UNIVERSITY OF SOUTHAMPTON

Energy-efficient design and implementation of turbo codes for wireless sensor network

by

Liang Li

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Physical and Applied Science
School of Electronics and Computer Science

November 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCE
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

by Liang Li

The objective of this thesis is to apply near Shannon limit Error-Correcting Codes (ECCs), particularly the turbo-like codes, to energy-constrained wireless devices, for the purpose of extending their lifetime. Conventionally, sophisticated ECCs are applied to applications, such as mobile telephone networks or satellite television networks, to facilitate long range and high throughput wireless communication. For low power applications, such as Wireless Sensor Networks (WSNs), these ECCs were considered due to their high decoder complexities. In particular, the energy efficiency of the sensor nodes in WSNs is one of the most important factors in their design. The processing energy consumption required by high complexity ECCs decoders is a significant drawback, which impacts upon the overall energy consumption of the system. However, as Integrated Circuit (IC) processing technology is scaled down, the processing energy consumed by hardware resources reduces exponentially. As a result, near Shannon limit ECCs have recently begun to be considered for use in WSNs to reduce the transmission energy consumption [1,2]. However, to ensure that the transmission energy consumption reduction granted by the employed ECC makes a positive improvement on the overall energy efficiency of the system, the processing energy consumption must still be carefully considered.

The main subject of this thesis is to optimise the design of turbo codes at both an algorithmic and a hardware implementation level for WSN scenarios. The communication requirements of the target WSN applications, such as communication distance, channel throughput, network scale, transmission frequency, network topology, etc, are investigated. Those requirements are important factors for designing a channel coding system. Especially when energy resources are limited, the trade-off between the requirements placed on different parameters must be carefully considered, in order to minimise the overall energy consumption. Moreover, based on this investigation, the advantages of employing near Shannon limit ECCs in WSNs are discussed. Low complexity and energy-efficient hardware implementations of the ECC decoders are essential for the target applications.

A systematic approach to the study is employed, by considering the star network topology before the more intricate multi-hop topology. The investigation concludes that in a star network, decoders are not typically required in the sensor nodes. Therefore, the near Shannon limit coding gain that is offered by a turbo-like code can provide a significant energy saving for the energy-constrained sensor nodes. By contrast, in the case of multi-hop networks, decode-and-forward is typically applied, requiring the decoder to be employed in the sensor nodes. These decoders consume additional energy that erodes the energy saving provided by the turbo-like code.

To realise a low complexity and a high energy efficiency for a turbo decoder, it is essential to find the most desirable fixed-point parameterisation for the hardware implementations. This must be chosen to have the best trade-off between the decoding performance and the energy consumption. Previous research has shown that the best choice is highly dependent on a wide range of factors. Therefore, an efficient method to find the optimal parameterisation is proposed for designing an energy efficient turbo code. More specifically, a framework using EXtrinsic Information Transfer (EXIT) chart analysis is developed to investigate desirable fixed-point parameterisation for turbo decoders.

Conventional turbo decoder architectures have been targeted for high throughput applications, such as the 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) and Digital Video Broadcasting (DVB) standards. However, the communication requirements of WSNs are different from those conventional turbo codes applications. Motivated by this difference, a low complexity energy-efficient Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) decoder architecture is proposed. The proposed architecture employs an order of magnitude fewer gates than the most recent LUT-Log-BCJR architectures, facilitating a 71% energy consumption reduction, compared to comparable conventional turbo decoders. Moreover, by considering the trade-off between the transmission and decoding energy consumption, the proposed architecture facilitates a 10% reduction in the overall energy consumption at transmission ranges above 39 m, compared with the other type of widely used turbo decoder, the Maximum Logarithmic Bahl-Cocke-Jelinek-Raviv (Max-Log-BCJR) decoder.

Finally, based on the studies above, a framework for estimating the decoding energy consumption of a turbo code at an early design stage is proposed. Furthermore, a holistic design method for parameterising the turbo code design and optimising the overall energy efficiency is proposed. This is in contrast to the conventional design method, which lacks the capability to estimate the decoding energy consumption of a turbo code during the code design stage. Therefore, computation complexity is conventionally used to trade off with the decoding performance, without optimising the parameterisations of a turbo code for the purpose of overall energy saving. In this thesis, an example turbo code design is developed using the proposed holistic design method, in order to minimise the overall energy consumption.

Acknowledgements

Numerous people have supported me during my graduate research in diverse ways, without whom the completion of this thesis would be impossible. Only a few words here could not adequately capture all my appreciation.

I would like to express my heartfelt gratitude to my supervisors, Prof. Lajos Hanzo, Prof. Bashir Al-Hashimi and Dr. Rob Maunder, for their exceptional supervision, insightful guidance and overall for their supreme friendship. Their constant inspiration and unfailing encouragement have greatly benefited me. Especially, their diligence and endless energy deserve my sincere respect. I may not complete this thesis without those thorough discussions with Dr. Rob Maunder. I am very grateful for his careful explanation and unreserved teaching.

Many thanks also to my colleagues and the staffs of the Communications Group and ESD Group for all the useful discussions and comments throughout my research. I wish to specially thank Dr. Soon Xin Ng, Dr. Amit Acharyya, Xin Zuo, Sheng Yang and Jatin N. Mistry for fruitful discussions as well as for their friendship. Many thanks to my flatmates Dr. Yang Qin and Ning Wang for their help and friendship throughout my Ph.D. study.

As always, I would express my sincere appreciation to my dear parents for their love, unconditional support as well as for their cultivation, without whom I would not have reached where I am.

To my family

List of Publications

1. **L. Li**, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, “An energy-efficient error correction scheme for IEEE 802.15.4 wireless sensor networks,” *IEEE Transactions on Circuits and Systems II*, vol. 57, no. 3, pp. 233-237, 2010.

2. **L. Li**, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, “Design of fixed-point processing based turbo codes using extrinsic information transfer charts,” in *Proceeding of IEEE Vehicular Technology Conference*, Ottawa, Canada, 2010, pp. 1-5.

3. **L. Li**, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, “A low-complexity turbo decoder architecture for energy-efficient wireless sensor networks, ” *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, in press.

Contents

List of Symbols

General notation

- The notation $\tilde{x}$ denotes an LLR value that pertains to the bit value x.
- The superscript $^{\mathrm{a}}$ is used to indicate an a priori LLR.
- The superscript $^{\mathrm{e}}$ is used to indicate an extrinsic LLR.
- The superscript $^{\mathrm{p}}$ is used to indicate an a posterior LLR.
- p and q are used to represent the operands of a calculation.

Special symbols

a Memory access rate.

$\mathbf{a}_i$ A bit sequence having the index $i \in [1, 2, 3, ...]$.

A Power amplifier efficiency.

B The number of BCJR operations performed.

C Turbo code complexity.

$\mathbf{b}_i$ A bit sequence having the index $i \in [1, 2, 3, ...]$.

d Transmission distance.

$D_{\mathrm{H}}^{\mathrm{min}}$ Minimum hamming distance.

E_{b} Energy per bit.

E^{pr} Decoding/processing energy consumption.

E^{tx} Transmission energy consumption.

E_{cyc} Energy consumption per clock cycle.

f Frequency.

G_{c} Coding gain

h The number of the component encoders in an MCTC.

I The number of decoding iterations performed.

k The number of inputs to each component encoder in a turbo code.

K Constraint length of a convolutional code.

m The number of memory elements (registers) employed in a convolutional code.

n The number of non-systematic outputs generated by each component encoder in a turbo code.

N The block/interleaver length of a turbo code.

$N_{\mathrm{b}i}$ The number of bits in the bit sequence $\mathbf{b}_i$.

N_0 Noise power spectral density.

P Power consumption.

P_{l} Path loss.

p Path loss exponent.

r Receiver noise figure.

R Coding rate.

s Decoding throughput of a decoder.

S A state in a turbo decoding trellis.

S_0 Minimum received SNR required to achieve the target BER in an uncoded wireless communication system.

S_c Minimum received SNR required to achieve the target BER in an coded wireless communication system.

t Time consumption.

T_i A transition in a turbo decoding trellis.

V, v Supply voltage.

w_{s} The sliding-window length of the sliding-window turbo decoder employed in the Log-BCJR algorithm.

w_{p} The pre-backward recursion length of the sliding-window turbo decoder employed in the Log-BCJR algorithm.

α The α values in the Log-BCJR algorithm.

β The β values in the Log-BCJR algorithm.

δ The δ values in the Log-BCJR algorithm.

γ The γ values in the Log-BCJR algorithm.

Chapter 1

Introduction

A Wireless Sensor Network (WSN) is a network composed of a number of sensor devices, each having multiple integrated functions [3]. These include sensing and processing the phenomenons that occur in the surrounding environment, as well as the ability to communicate wirelessly with each other and with higher-level networks. Figure 1.1 illustrates a typical WSN topology, in which a number of sensor nodes transmit information to the central node. This may then forward the information to a higher-level network, such as the Internet or a cellular network. Figure 1.1 includes examples of both single- and

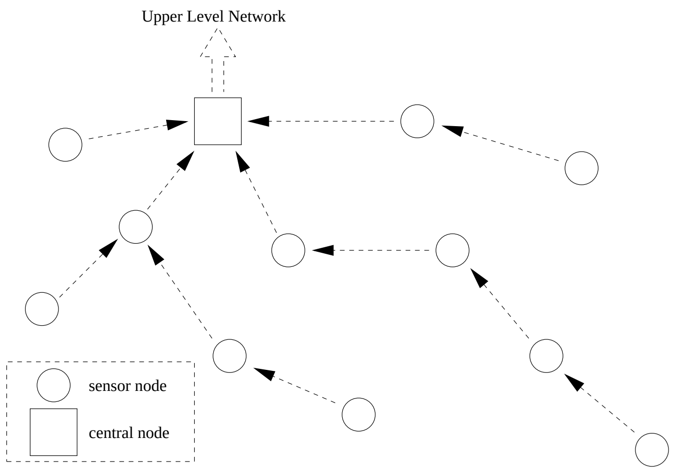


FIGURE 1.1: A typical WSN configuration.

multi-hop communications from the sensor nodes to the central node. A number of sensor nodes act as relays in the case of multi-hop communications, receiving messages from the direction of the source sensor node and retransmitting them in the direction of the central node. In some cases, the network topology may be more complicated than that shown in Figure 1.1. For example, there could be more than one central node for collecting the information and forwarding it to the higher-level network. Alternatively, the communication between a sensor node and a central node could take place over multiple routes, rather than just a single route. On the other hand, a simpler star network topology may be employed in some scenarios, where each sensor node uses only a single hop to transmit directly to the central node. As a result, the topologies of WSNs are highly flexible, allowing them to be deployed in different environments for various applications. One of the unique feature of WSNs, compared to other wireless communication applications, is that, typically, information only flows in one direction, from the sensor nodes to the central node. The sensor nodes are typically only required to receive when relaying each other's messages. In some applications, the sensor nodes may receive a small amount of control signaling from the central node, but the amount of information is so small when compared to the information transmitting from the sensor nodes, that it can be treated as a much simpler case, employing a much simpler communication scheme.

WSNs have become a popular technology during the last decade, which has been widely applied in industrial, medical and home applications. With the growing functions of the sensors, WSNs can be used for continuous monitoring environmental [4] or other conditions and events for a wide range of applications, from long range geophysical inspection [5] to short range personal health care [6]. This has been made possible owing to advances in wireless communications, digital electronics and Micro-Electro-Mechanical Systems (MEMS), which allow the sensor devices to be implemented with reduced size, energy consumption and cost [7, 8]. Wireless technology gives significant advantages compared to conventional wired sensor network solutions, including improved network coverage, scalability, reliability, installation and maintenance cost. These benefits are achieved by avoiding the requirement for wired connections between nodes.

In a WSN, each sensor node has processing and communication capabilities. They typically also include a number of MEMS sensors, which are able to monitor a selection of parameters and conditions. These include temperature, humidity, movement, lighting condition, pressure, soil makeup, noise level, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects, as well as the current characteristics such as speed, direction, and size of an object [8]. Because the scope of sensor node applications is vast, there are a wide variety of requirements for their characteristics, such as their coverage, data rate, sensor functions and lifetime, for example. Therefore, there is vast scale of the hardware and software solutions of sensor node implementations. However, some common features are shared by all sensor nodes [7, 9]. The sensors are deployed either inside the phenomenon being observed, or very close to it. To avoid interference with the phenomenon, the sensor nodes are typically required to be small and lightweight. In addition, the sensors are often deployed in large numbers and may be disposable, in which case the sensor nodes must be low cost. As a result, sensor nodes are typically powered by batteries that are small, lightweight and inexpensive, or by a limited energy harvesting ability. Furthermore, the recharging or replacement of the batteries is typically difficult and avoided, since the deployed sensors are often difficult to recollect for battery recharging or replacement. Therefore, the fundamental requirement of sensor nodes is low energy consumption. Maximising the energy efficiency of the sensor nodes is an important challenge for all WSN implementations. In particular, any optimisations made for energy saving purposes must not sacrifice the other specifications of the design that are required to maintain the WSN functionality. For example, a WSN designed for a particular scenario is required to meet the particular communication performance requirements, such as range, reliability and data rate. An optimal design of the communication system achieves a minimal energy consumption while meeting all the other requirements. In this way, the lifetime of the WSN may be maximised, without requiring the replacement or recharging of the sensor node batteries. Therefore, before studying optimisation approaches for minimising the energy consumption of WSN communication systems, their communication requirements must be considered carefully. In this chapter, the communication requirements of WSNs are investigated for different types of WSNs application. The real world applications of WSNs are summarised into three categories, namely our-door applications, in-door applications and Body Area Networks (BANs). All the three categories are discussed in order to give an overall investigation of the WSN communication requirements. Furthermore, based on the investigation results, turbo and turbo-like channel codes are proposed for reducing the communication energy consumption of sensor nodes. The potential advantages and disadvantages of this approach are discussed in Section 1.4.2. Finally, the outline of the thesis is given in Section 1.5.

1.1 Communications in out-door wireless sensor networks

As described above, the sensors in WSNs are required to monitor different parameters and conditions within the deployed phenomenon. For this reason, the main family of applications for WSNs is environmental monitoring. In out-door applications, examples include forest fire detection, water quality monitoring [10, 11], infrastructure monitoring [12], precision agriculture [13], geophysical inspection [5, 14, 15] and habitat monitoring for animals [16, 17]. The purpose of the sensors could be gathering and transmitting data to the central nodes for recording, or detecting unusual changes in the target phenomenon and rasing alarms.

In these WSNs applications, the sensor nodes are typically required to be deployed over a large area such a forest, a mine plant, a suspension bridge, a stretch of river or a farm, naturally resulting in a relatively long-range communication requirement. In addition, a long lifetime of the order of months or years is typically required without recharging or replacement of the sensor node batteries. However, the data rate requirement is typically low, since the monitored signals often have a low dynamic range and low frequency. Furthermore, in many cases, data delivery delays of several minutes do not prevent the WSN from fulfilling its purpose [4]. Table 1.1 summarises the communication specification of several WSN deployments for out-door environmental monitoring that have been reported in the literature. In the following sub-sections, the communication

Pub	Application	Number of sensors	Channel throughput[1]	Transmission frequency	Communication distance	Coverage area
[15]	Landslide detection	150	-	2.4 GHz	50 m	-
[14]	Volcano monitoring	16	100 Kb/s	2.4 GHz	200-400 m	3 km linear area
[13]	Precision agriculture	27	153 Kb/s	868 MHz	300 m to gateway node	5000 m^2
[16]	Habit monitoring	-	40 Kb/s	916 Mhz	3600 m	959 km^2
[4]	Cattle monitoring	28	72 Kb/s	433 MHz	500 m	-
[4]	Ground water quality monitoring	9	72 Kb/s	915 MHz	800 m	6 km^2
[18]	Mountain environmental monitoring	-	76 Kb/s	868 MHz	-	900 m linear area
[17]	Habit monitoring	98	128 Kb/s	453 MHz	-	9000 m^2

Table 1.1: A summary of the typical communication requirements of a selection of studies of environmental monitoring WSNs.

requirements of out-door environmental monitoring WSNs are discussed, based on the investigation results of previous work.

1.1.1 Coverage and communication range

As shown in Table 1.1, the coverage and the communication range requirements of WSNs are diverse, since they depend on the particular scenario. Here, the coverage depends upon the area of the target forest, farm or factory, for example, while the communication range additionally depends on the sensor nodes' deployment density that is required in order to obtain the desired monitoring accuracy when the multi-hop technique is employed. In many cases, the communication range requirements vary from tens to hundreds of metres. In some particular cases, the communication range can be thousands of metres. Figure 1.2 summarises the communication range, computational capacity and storage capacity of various sensor node devices. The specified communication ranges agree with the communication range requirements that are summarised in Table 1.1. Since the small size requirement of the sensor nodes imposes a corresponding limitation upon the transmitter, antenna and batteries, typical WSN communication range in out-door scenarios can be considered to be relatively long distances. Moreover, the path loss of the transmitted signal between a pair of sensor nodes may be as high as the fourth

[1]Note that the channel throughputs are the maximum channel throughput of the employed sensor node devices, not the actual requirements of the applications.

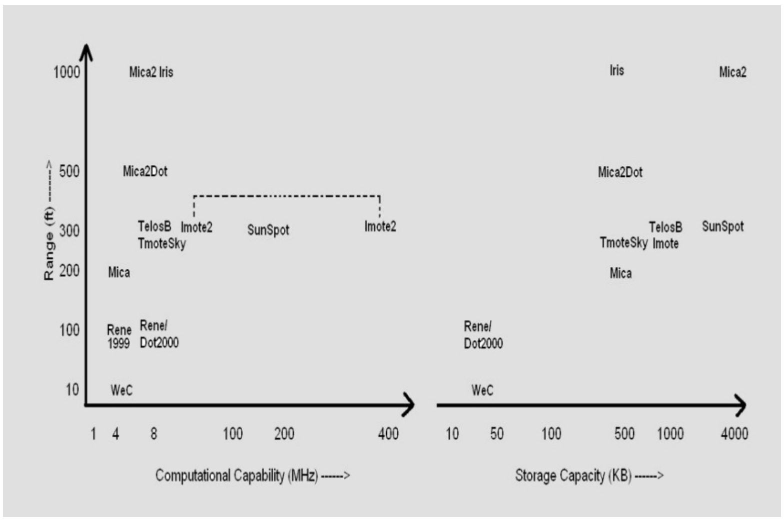


FIGURE 1.2: Classification of WSN sensor node devices based on communication range, computational capacity and storage capacity. ©V. Potdar *et al.* 2009 [19]

order exponent of the distance between them. This is because the antennas of the sensor nodes are close to the ground and because WSNs are typically deployed in complex environments [8]. Therefore, maintaining reliable communication across the WSN is a challenge when designing the communication systems of sensor nodes. In particular, the energy consumed by transmission is typically a large portion of the overall energy consumption of the sensor nodes. For example, the investigation in [20] shows that, in a Rayleigh fading channel with fourth power distance loss, the energy cost of transmitting 1 Kb over a distance of 100 metres is approximately the same as executing 3 million instructions by a 100 Million Instructions Per Second (MIPS) general-purpose processor. This is far beyond the signal processing requirement of a typical sensor node.

1.1.2 Date rate

Compared with other wireless applications, such as cellular networks and Wireless Local Area Networks (WLANs), the typical data rate of environmental monitoring WSNs is relatively low. In Table 1.1, the data rate of the considered schemes range from tens of Kb/s to hundreds of Kb/s. Moreover, the channel throughput is often significantly higher than the required data rate, enabling low duty cycle communications between sensors. In general, WSN applications are expected to have relatively low data rate requirements compared with conventional wireless communication applications, such as mobile phones and WPANs. This can be exploited to reduce the energy consumption of the sensor nodes.

1.1.3 Network topology

The network topologies of environmental monitoring WSNs vary from application to application. A simple star network is only suitable for small scale WSNs with limited numbers of sensor nodes and limited coverage. However, as discussed above, many applications are required to cover very large areas measured in km^2. Therefore, multi-hop technology is generally applied to reduce the communication range requirement for each transmitting node. Here, sensor nodes situated between the source node and the central node are employed as relays. The transmitted information is received and retransmitted by the relays, reducing the communication range of each transmission. For example, a 46-hop system comprising 64 sensor nodes has been deployed on the Golden Gate Bridge to monitor the health of its infrastructure [12]. This reduces the maximum communication range from 4200 feet to just 75 feet. Although the number of sensor nodes deployed in the examples of Table 1.1 are mostly several tens, these WSNs were developed for the purpose of research work. In real life applications, the number of sensor nodes and the network coverage area can be several orders of magnitude higher [8]. More than one central node or gateway node may be included in a WSN for collecting data from the sensors and communicating with higher-level networks. In addition, in some cases, a single network may comprise several interconnected sub-networks having different topologies [21]. Furthermore, for habitat monitoring applications, sensor nodes may be attached to the target animals, which causes their positions to change with time. In these cases, the network becomes ad-hoc, requiring self-organisation, since the network topology can be expected to change over time. Similarly, in other applications, the network topology can also change owing to sensor movement or temporary inaccessibility. In these cases, maintaining the network topology becomes a significant a challenge.

1.1.4 Frequency bands

The frequency bands used by WSNs may need to be available globally, if they are to be used world wide. Furthermore, WSNs must not interfere with other networks, since they are often deployed in places like factories, hospitals and office buildings, where other wireless communication services may be used at the same time. As listed in Table 1.2, Industrial Scientific and Medical (ISM) bands present an option, since they are license-free in most countries.

According to [22], certain hardware constraints imposed by the sensors and the tradeoff between antenna efficiency and power consumption limit the choice of carrier frequency for WSNs to the ultrahigh frequency range. Indeed, the 433.05 - 434.79 MHz, 902 - 928 MHz and 2400 - 2500 MHz bands are typically employed in environmental monitoring WSNs, as shown in Table 1.1. In addition, the 868 MHz band, which is approved by European Telecommunications Standards Institute (ETSI) in Europe, is also used in

Frequency bandwidth	Center frequency
30 kHz	6780 kHz
14 kHz	13,560 kHz
326 kHz	27,120 kHz
40 kHz	40.68 MHz
1.74 MHz	433.92 MHz
26 MHz	915 MHz
100 MHz	2450 MHz
150 MHz	5800 MHz
250 MHz	24.125 GHz
500 MHz	61.25 GHz
1 GHz	122.5 GHz
2 GHz	245 GHz

Table 1.2: Frequency bands available for ISM applications.

some WSNs. On the other hand, higher transmission frequencies allow smaller antennas [4]. As a result, the 902 - 928 MHz and 2400 - 2500 MHz bands have been popular choices in the latest environmental monitoring WSNs.

1.1.5 Scalability

The number of sensor nodes deployed in an environmental monitoring WSN depends upon the required network coverage and sensor deployment density. This may exceed thousands of sensor nodes [8]. Indeed, more than 1000 sensors were deployed in the experimental environmental monitoring WSN of [23] and it was suggested that more than 10000 sensors would be required for a real-life network having a coverage 5 km^2. In addition, the deployment densities of WSNs are also highly variable. In [24], the sensor nodes density was as high as 20 nodes/m^2. By contrast, in [14], only 16 sensor nodes were required to cover a 3 km linear area. Therefore, a high scalability is required for WSN protocols and algorithms, in order to deal with different challenges in different scales of network. For example, the high interference of high-density networks or the long communication range for low-density networks.

1.1.6 Lifetime

Depending on the application, the required lifetime of an environmental monitoring WSN may range from some hours to several years. For example, in [4,14,16], the lifetime specification of the deployed WSNs are 19 days, nine months and 1.5 years, respectively. In the general case, a maximal WSN lifetime is desirable during the design, which leads to a requirement for high energy efficiency and reliability. Due to the small size of the sensor nodes, maintaining a lifetime of the order of days is a significant challenge. In cases of lifetimes of the order of years, the sensors must enter a sleep mode for most of the time, when sensing and transmitting are not required [25].

1.1.7 Energy consumption

As discussed, the WSN sensor nodes may be very small, so that they are cheap to fabricate, easy to deploy and so that they do not interfere with the phenomenon that they are intended to monitor. With the latest development in MEMS, signal processing and wireless communication technologies, the sensor nodes can be manufactured with the volume of several cubic centimetres or even smaller. For example, in [4, 16], the size of the sensor nodes are reported as $4.9 \times 3.7 \times 1.2$ cm^3 and 50×60 mm^2, respectively. These small sizes render the corresponding WSNs naturally energy-constrained, even though most of the sensor node volume is occupied by batteries in many cases. In addition, WSNs are also expected to have a long lifetime without the requirement for battery replacement while performing reliable communications over the specified communication distance. Hence, on board energy harvesting systems have been developed for providing an energy supplement for WSNs. Potential energy sources include solar, piezoelectric, vibration, thermoelectric and acoustic noise [26]. However, owing to the limited size of sensor nodes, current embedded energy harvesting systems only offer limited compensation of the energy constrained situation of WSNs. Table 1.3 gives some examples of the power generation potential of several energy harvesting technologies for WSNs [27]. A summary of some previous works in [19] shows that the power consumptions of the

Energy harvesting technology	Power density
Solar cells (outdoors at noon)	15 mW/cm^2
Piezoelectric (shoe inserts)	330 μW/cm^3
Vibration (small microwave oven)	116 μW/cm^3
Acoustic noise (100 dB)	960 nW/cm^3

Table 1.3: Power densities of harvesting technologies.

signal processing systems in WSNs can vary from several milliwatts to several tens of milliwatts. The transmission power can reach 20 dBm (100mW) when the communication distance is of the order of tens of metres and increases exponentially as the distance increases. As a result, even solar cells do not generate as much power as is consumed by a typical wireless sensor in environmental WSNs. Therefore, the energy consumption of the communication system on sensor nodes is a very important issue for WSNs. On the other hand, in some cases the central node of a WSN has abundant energy resources, since it can typically afford a larger size than the sensor nodes or has access to mains power. If not, it is also easier to replace or recharge the batteries, since there are usually few central nodes in a WSN.

1.1.8 Path loss model

Path loss has a significant impact on the design of communication systems for WSNs. Therefore, the WSN communication scheme should exploit the begin channel characteristics offered by short range communications, but should also cope with the malign channel characteristics that are associated with long ranges. Depending on the environment of the target scenario, path loss can be modelled in a number of ways when using simulations to estimate the performance of the designed system [1,2,28,29]. However, regardless of the particular target environment, the path loss P_l is described as a function of the transmission distance d. A model that has been widely used [1,2,28–33] increases the mean path loss exponentially with transmission distance d, according to

$$P_l(d) \propto \left(\frac{d}{d_0}\right)^p, \tag{1.1}$$

where d_0 is a reference distance and p is the path loss exponent, which indicates how fast the path loss increases with transmission distance d. Typically, the path loss obtained using Equation 1.1 is expressed in the logarithmic domain, using decibels.

In [29], wireless communications in various environments for WSNs was investigated using carrier frequency of 915 MHz. The investigation particularly considered the situation when sensors lie on or near the ground as opposed to previously developed models, which assumed that a person is holding the transceiver at a height of 1.5 m from the ground. The study also considered the different multipath characteristics of urban environments, open terrain, woody terrain and woody/hilly terrain separately. The proposed path loss model is given by

$$P_l(d)[dB] = \mathrm{X} + 10p\log_{10}\left(\frac{d}{d_0}\right), \tag{1.2}$$

where X models the random shadow fading effect using a zero-mean Gaussian distribution having a standard deviation in σ_{X} expressed in decibels. The measured path loss exponents p and shadow fading standard deviation σ_{X} are summarised in Table 1.4.

Terrain	p (dB)	σ_{X} (dB)
Open	3.41	4.70
Woody	2.35	4.37
Woody and Hilly	2.90	4.17

Table 1.4: Summary of measured field results.

The most significant factor that can be abstracted from the environment to model the path loss is the path loss exponent p. During the design of the wireless communication systems for WSNs, an appropriate path loss model should be employed to evaluate the performance and the energy efficiency of the system over a range of typical transmission distances in the target scenario. This is particularly important for WSNs, since it is important to ensure both high quality performance and high energy efficiency. The evaluation of the WSN design using an appropriate path loss model can help to strike a good balance between these two characteristics of the design.

1.2 Communications in in-door wireless sensor networks

1.2.1 Home automation and smart environment applications

WSNs can be also applied for home applications. For example, home automation technology uses smart sensor nodes buried in appliances, such as vacuum cleaners, microwave ovens and refrigerators, to manage them [34]. Environmental monitoring WSNs can be also deployed for assisted living. In contrast to out-door WSNs, home-based WSNs can exploit other available wireless networks such as WLAN or Bluetooth networks, when the size of the sensor nodes is affordable. If these networks cannot be utilised, the communication requirements are similar to those of out-door WSNs applications, except that the communication distance is much shorter. More specifically, in the home environment, the communication range varies from several metres to several tens of metres. As a result, the required transmission power is typically much lower than in out-door WSNs, but interference becomes a more important issue, since the deployment density can be high within the relatively small area of a home, and the WSN must co-exist with other wireless networks.

1.2.2 Other applications

There are numerous of other applications for in-door WSNs, such as the environmental monitoring of office buildings, green houses and factories. The characteristics of these applications lie in-between those of out-door and home-base WSNs. For example, the typical communication distance in these applications is longer than in home-based WSNs but shorter than in long-range out-door WSNs. Here, the transmission distance varies from several metres to several hundreds of metres.

In addition, the propagation environment of in-door applications is different from the out-door applications. The reflection and absorption of the transmitting signals by walls and other in-door obstacles must be considered [35].

In [28], indoor wireless communication in multi-floored buildings is investigated for carrier frequencies of 914 MHz. This work also concluded that the path loss does not depend on the carrier frequency significantly in the range 900 MHz to 4 GHz. A path loss model is proposed, which is defined as the path loss from the transmitter to the reference distance d_0, plus the additional path loss described by Equation 1.3 in decibels,

$$P_l(d)[dB] = P_l(d_0)[dB] + 10p\log_{10}\left(\frac{d}{d_0}\right). \tag{1.3}$$

A $d_0 = 1$ m reference distance was chosen and $P_l(d_0)$ was assumed to be due to free space propagation. The study considered 14 different scenarios, including transmission within different types of buildings and through different number of floors. The path loss exponent p was found to vary from 1.81 to 5.04. A similar model is proposed in [1, 2], according to

$$P_l(d)[dB] = 20\log_{10}\left(\frac{4\pi}{\lambda}\right) + 10p\log_{10}(d), \tag{1.4}$$

where λ is wave length of the transmission signal and d_0 is assumed to adopt a default value of 1 m.

1.3 Body area networks

Body Area Networks (BANs), also referred to as Body Sensor Networks (BSNs), Body Area Sensor Networks (BASNs) or Wireless Body Area Networks (WBANs), are an emerging application of short range WSNs in the healthcare industry. These networks comprise a number of wireless sensors placed on or in the human body for continual monitoring of physiological parameters such as heart rate, ElectroCardioGraphy (ECG) data, ElectroEncephaloGraphy (EEG) data, blood pressure, body temperature, motion and levels of certain chemicals such as sugar, oxygen and medications in the blood [6]. The monitoring of these parameters could be important or even life critical for some people or patients, such as the ageing population, chronic disease patients, cerebrovascular and cardiovascular disease patients. Long-term monitoring and logging of patients' physiological parameters could help doctors to provide personalised treatment or discover risks earlier. For example, the blood sugar level and other related human body conditions of diabetes patients have to be regularly monitored by using examinations in hospital and self-examination by the patients. However, this approach is unreliable, inconvenient and not frequent enough [6]. With BAN solutions, the conditions of the patients can be monitored and logged continuously, without involving effort from the patients. For some dangerous diseases, such as coronary heart disease, which could cause sudden death, BANs are able to monitor and discover the potential danger, raise the alarm and even contact the hospital before the patient is even aware of the on-setting problem. BANs [36] can also be applied in sports training or gym applications to help instructors to monitor the physiological parameters of people undertaking exercise. Furthermore, BANs can be deployed in hospitals or home-based healthcare systems to create a more comfortable, convenient and economical way to perform the patient monitoring. Over the past few years, advancements in electronic systems and wireless technologies have enabled the development of small and intelligent medical sensors which can be attached to or implanted into the human body. The healthcare industry is becoming increasingly interested in using these technologies to develop practical BANs [37]. This has also stimulated interest in related research areas, such as energy harvesting, signal processing and communication. In 2008, the Institute of Electrical and Electronics Engineers (IEEE) 802.15 working group established the body area network task group (IEEE 802.15.6), to develop guidelines for using wireless technologies for BANs applications in various healthcare services.

The scenario of BANs has some unique features compared with the conventional WSN applications. Figure 1.3 illustrates a typical BAN scenario, which comprises a number of sensor nodes attached on or implanted in the human body to perform different functions for collecting different physiological parameters. A central device such as a smart phone receives the data from the sensor nodes over a wireless channel, forming a BAN. A BAN may connect to an external network for communicating with a higher-level system, such as the hospital, depending on the requirement of different scenarios. The number of

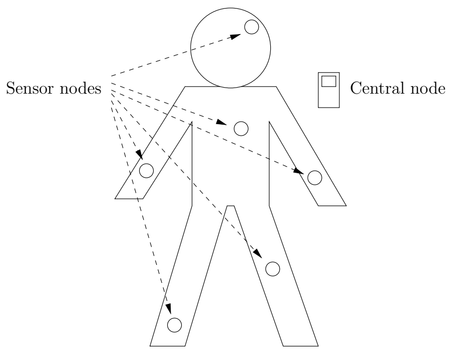


Figure 1.3: A typical BANs configuration.

sensors required could be variable depending on different applications. To monitor one particular disease, typically only a few (< 3) sensor nodes are required [37]. However, for more complicated situations, more sensors might be required. Especially, when motion detection is involved, for example, for people who need after treatment to help recover mobility. In this case, sensors might be required on every movable part of the body, with more than one sensor required where accurate motion detection is sought. According to [6], typically no more than 20 sensors is required for any one person.

Owing to the wide variety of characteristics in BAN applications, they must be carefully considered when developing communication technologies for BANs.

- Firstly, the size of the sensors is required to be as small as possible for comfort and convenience issues. Smaller nodes imply limited onboard resources, requiring the energy consumed by processing, storage, and communication to be traded-off against the fidelity, throughput and latency required for the transmitted data [38]. Since one of the purposes of BANs is to create a convenient healthcare system for patients without support from professionals, the wireless sensors on human bodies need to have long lifetimes without any need for maintenance. This leads to an extreme energy efficiency requirement, relying only on battery-stored or harvested energy.

- Secondly, it is assumed that all the sensors and the central device are always near to the patient, leading to a very short transmission range requirement for BANs. Some previous work [6, 39, 40] suggests that 2-5 metres communication range is sufficient for most BAN applications.
- Thirdly, the transmission power must be limited in order to avoid the detrimental effects of high power transmission on the health of humans within the vicinity.

The differences between the scenarios of BANs and the conventional WSNs lead to some unique communication requirements. Therefore, this section discusses these requirements in detail.

1.3.1 Frequency bands

As a special case of WSNs, the frequency bands that are appropriate for use in BANs are similar to those of conventional WSNs. For example, many previous works are based on IEEE 802.15.4 standards, which uses 868/915MHz and 2.4 GHz frequency bands [41–44]. Similar frequency bands are also chosen by [45–51]. An alternative solution is to use Ultra-WideBand (UWB)) technology, which is authorised for communication between 3.1 GHz to 10.6 GHz [52]. [53–57] discussed the potential and the advantages of applying UWB technology to BANs. The USA Ultra-WideBand (FCC) is considering several possible frequency bands for use by BANs [58]:

- 2300-2305 MHz and 2360-2395 MHz Band: The 802.15.TG6 Group and GE Healthcare (GEHC) propose to use this band for BANs. However, this band is currently used by several other services, including Aeronautical Mobile Telemetry (AMT), federal radio location and amateur radio users. This could pose interference and security problems. The FCC is considering the proposed potential use of these bands by BANs on a coexistence and non-interference basis.
- 2400-2483.5 MHz Band: This band is used by ISM equipment on a non-licensed basis under the FCC's rules. The FCC seeks comment on whether BANs could operate in this band under current rules or whether new rules would be required to regulate BANs using this band.
- Other Frequency Bands: The FCC is seeking comment on whether other frequency bands may be appropriate for BANs, including the 5150-5250 MHz band, which is now allocated for federal and non-federal aeronautical navigation and non-federal fixed-satellite use, as well as Unlicensed National Information Infrastructure (U-NII) devices.

According to the latest development of the IEEE 802.15.6 task group [59], three PHYsical layer (PHY) specifications are included, the NarrowBand (NB) PHY operating in 402 - 405 MHz, 420 - 450 MHz, 863 - 870 MHz, 902 - 928 MHz, 950 - 956 MHz, 2360-2400 MHz and 2400-2483.5 MHz, the UWB PHY operating at 3993.6 MHz and 7987.2 MHz with a bandwidth of 449.2 MHz, and the Human Body Communications (HBC) PHY operating in two frequency bands centred at 16 MHz and 27 MHz, with the bandwidth of 4 MHz.

1.3.2 Data rate

BANs typically require real-time low data rate communications. For example, the investigation results from [37], [39] and [60] are summarised in Table 1.5. Figure 1.4, quoted

Healthcare applications	H. Li [39]	B. Zhen [37]	M. Patel [60]
Heartbeat	<0.1 Kb/s	0.05 Kb/s	-
Body temperature	<0.1 Kb/s	0.05 Kb/s	<10 Kb/s
ECG	2.5 Kb/s	72 Kb/s	72 Kb/s
EEG	0.54 Kb/s	131.1 Kb/s	86.4 Kb/s
ElectroMyoGraphy (EMG)	-	1152 Kb/s	1.536 Mb/s
Blood pressure	<0.1 Kb/s	0.05 Kb/s	<10 Kb/s
Blood sugar level	<0.1 Kb/s	-	-
Other Blood analysis	-	8.192 Kb/s	-

TABLE 1.5: Data rate requirement of different applications in BANs.

from [61], summarised the data rate requirement possible ranges of several typical BAN applications. The investigation of [6] concluded that 500 Kb/s data rate is sufficient for

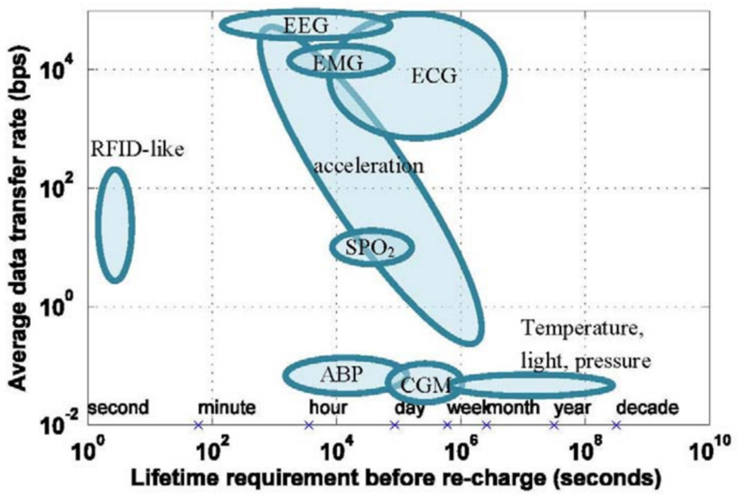


FIGURE 1.4: Summaries of data rate requirements of a selection of BAN applications. ©Y.-Q. Zhang 2011 *et al.* [61]

BAN applications. Note that for low-rate applications, such as heartbeat, body temperature and blood pressure, the specific data rate values that were identified in the various studies are relatively consistent. However, for some high-rate applications, such as ECG and EEG, the conclusions are quite varied. These discrepancies may be attributed to the different assumptions of how much pre-processing is applied to the signals before they are transmitted. For example, if a sensor transmits compressed data rather than raw data, the data rate requirement could be significantly reduced. In all the previous investigations into this topic, the highest data rate requirement is given by [60], which identified that up to 1.536 Mb/s may be required by EMG monitoring. On the other hand, some studies include video stream transmission in BANs scenarios, increasing the data rate requirement to 100 Mb/s [6, 60]. However, these scenarios are not typically considered by BAN applications. According to [62], the 802.15.6 standard has a limitation of the maximum data rate of r 971.4 Kb/s including the communication payload, which gives a effective throughput of 674.7 Kb/s.

1.3.3 Reliability, accuracy and latency

The performance of a BAN can be quantified by the delay profile, the information loss rate, the Bit Error Rate (BER) and Frame Error Rate (FER), which must be considered carefully during the design of the communication system. The prevalence of communication errors should be less than a strictly defined limit in order to avoid disastrous behaviour [37, 63, 64], since an erroneous or erased record could affect a doctor's judgement when making life-critical decisions, for example. On the other hand, transmission latencies of the order of seconds is not a critical issue in practice, since the reaction of the hospital, such as sending an ambulance, is associated with a higher order of magnitude delay.

In addition, when a human body moves, the attached sensors will change positions relative to each other. Furthermore, when the environment changes, the channel conditions will also change and affect the network performance. Despite these varying conditions, BANs are required to maintain reliability. Consideration must be given to the various different activities that can be conducted by human bodies, such as walking, running and turning, for example. The human body may move through various transmission environments, such as tunnels, subways and parks, for example. Each of these is associated with different propagation characteristics and different interference patterns from coexisting BANs and other networks.The design of BANs must be prepared for all practical scenarios.

1.3.4 Path loss model

Although BANs typically have short communication ranges of 2 - 5 m, the propagation environment is hostile, since human bodies strongly attenuate RF signals [64, 65]. A characterising feature of conventional WSNs is that path loss is highly variable, since human activities are typically performed nearby. The path loss model for BANs has the same mathematical expression as the conventional WSNs, as discussed in Section 1.1.8 In [30], the path loss exponent is reported to range from 3.2 to 4.9 depending on the transmission frequency bands. In [31], the study of channel models for Non-Line Of Sight (NLOS) propagation along the surface of a human body found that the path loss exponent has a value of about 3. In [32], a path loss exponent of 7 was found in NLOS situations for propagation around human bodies. In [33], the path loss exponent was reported to vary from 4.22 to 6.26. As these results demonstrate, the human body is not a benign environment for wireless communication.

A further challenge of BANs is that all the sensor nodes operate in each others' vicinity, potentially inducing an interference problem. Therefore, maintaining a reliable communication in BANs despite the unique unstable propagation environment is a particular challenge, even though the transmission range is short.

1.3.5 Energy consumption

The energy consumption of the BAN sensor nodes is a very crucial issue owing to their limited energy resources and the long life-time requirement. As mentioned before, the sensors in BANs must rely on energy harvesting or battery operation without recharging or replacement. Since the sensors are expected to be as small as possible, they cannot employ high capacity batteries or energy harvesters and so their energy resources are extremely limited. Therefore, every function of the sensors is required to be energy efficient, including the wireless communication mechanism. It is widely agreed that the low power requirement is one of the most challenging issues in developing BANs [6,37,66].

1.3.6 Network topology

Because of the small network scale and the one-way communication assumption, a star topology is an obvious solution of BANs [67, 68]. Here, the sensor nodes transmit directly to a single central sink node. The advantages of the star topology are its simple architecture and highly concentration of system complexity upon the central node, assumed to have access to plentiful energy resources [69]. However, it may be desirable to employ multi-hop relaying in some BANs applications that employ a sufficiently high number of sensor nodes. This is because the propagation environment near or inside human body is not benign for wireless communication and so multi-hop transmission has the benefit of increasing the communication reliability and reducing the transmission power [66]. Hence, many recent efforts have focused on multi-hop network solutions for BANs [70, 71].

1.3.7 Security

Security is another important issue in BANs for medical applications [40]. Safety and privacy should be considered for all involved parties, including doctors, nurses, patients, administrative personal and medical service providers. BAN devices also require authentication for security purposes. The interferences imposed by external devices or intentional attacks made by third parties must be considered. On the other hand, owing to the limited resources that are available in the sensor nodes and the requirement for the system to be user friendly, any security measures must be simple. In particular, the insertion and removal of a node in a BAN must be easy for the user to accomplish.

1.4 Error-correcting code solutions in wireless sensor networks

1.4.1 Energy-constrained challenge in wireless sensor networks

As discussed, the key challenge that applies to all WSNs is the energy constrained nature of the sensor nodes. Therefore, the energy consumption of a WSN's communication system must be considered throughout the whole design process. However, specifications such as the communication ranges, data rates, network topologies and carrier frequencies, are highly dependent on the targeted scenario. The optimisation of the communication system's energy efficiency must take these factors into account. Previous research has considered this subject in different layers of the communication system, including the Media Access Control (MAC) layer, the physical layer and the hardware architectures. Table 1.6 reviews the literature in this area.

Table 1.6: Previous contributions on energy-efficient wireless communication systems for WSNs.

Author(s)	Contribution
J. Heidemann and D. Estrin 2002 [72] T. V. Dam and K. Langendoen 2003 [73]	Different scheduling schemes are proposed on MAC layer to reduce the energy consumption on the sensor nodes in WSNs by scheduling the communication systems on the sensor nodes in sleeping mode periodically.

Continued on next page

Table 1.6 – *Continued from previous page*

Author(s)	Contribution
E. J. Coyle 2003 [74] D.-H. Nam and H.-K. Min 2007 [75] L. Kyounghwa *et al.* 2010 [76] M. C. M. Thein and T. Thein 2010 [77]	Different clustering techniques are proposed to cluster sensors into groups, so that sensors communicate information only to clusterheads and then the clusterheads communicate the aggregated information to the central node. As a result, the total energy spent in the network for communications is reduced.
M. Cardei and M. Tha 2005 [78]	A method that divides randomly deployed sensors into different sets and to be active alternatively while maintaining the required coverage is proposed, which can reduce the average energy consumption of the entire network and extend the network lifetime.
A. Wang and A. Chandrakasan 2002 [79]	It investigates the challenges of energy-efficient Digital Signal Processor (DSP) that are specific designed for WSNs.
X.-H. Li 2003 [80]	A novel communication scheme which uses two transmitting sensors and space-time block codes to provide transmission diversity in distributed WSNs with neither antenna-arrays nor transmission synchronisation is proposed for the purpose of saving transmission energy.
O. C. Omeni *et al.* 2007 [81]	A novel concept of a wake-up fall-back time is introduced to handle time slot overlaps while Listen-Before-Transmit (LBT) technique is used in BANs to minimise the overall energy consumption.
C. C. Enz *et al.* 2004 [82]	An ultra low-power platform for the implementation of WSNs that achieves low-power operation through a careful co-design approach is developed by the Swiss Centre for Electronics and Microtechnology.
C. Schurgers and M. B. Srivastava 2001 [83] H. Oh and K. Chae 2007 [84] M. Zhang *et al.* 2008 [85] W. N. W. Muhamad *et al.* 2009 [86]	Different routing algorithms are proposed to optimise the transmission routes in ad-hoc WSNs for reducing the overall transmission energy consumption.

Continued on next page

Table 1.6 – *Continued from previous page*

Author(s)	Contribution
X. Chen *et al.* 2009 [87]	A novel transmission scheme in which the sensor nodes' transmissions are ordered according to the magnitude of their measurements. Sensors having magnitudes, smaller than a threshold do not transmit. The results show that the proposed approach is very energy efficient, with only a negligible measurement error introduced.
N. Sadeghi *et al.* 2006 [2] S. L. Howard *et al.* 2006 [1] M. C. Vuran and I. F. Akyildiz 2009 [88] M. E. Pellenz *et al.* 2009 [89] A. Brokalakis and I. Papaefstathiou 2012 [90]	Error-correcting coding schemes are considered for employment in WSNs to reduce the transmission energy consumption.
J. Abouei *et al.* 2011 [91]	The energy efficiency of Luby Transform (LT) codes is analysed for the scenario of WSNs.
A. Brokalakis *et al.* 2011 [92]	An innovative platform which combines a standard wireless node with very low cost reconfigurable hardware is designed in order to to evaluate the efficiency of this pioneering approach. Three different networking and security protocols are implemented in the present system: a) Turbo coding, b) Blowfish encryption and c) XMesh routing.
J. Singh and D. Pesch 2011 [93]	A novel Error-Correcting Code (ECC) based adaptive error control strategy is proposed for employment with a cascaded fuzzy inference system, in order to combat the communication unreliability of indoor environments.
J. Rahhal 2012 [94]	An LDPC code is considered for employment in Multiple-Input and Multiple-Output (MIMO) WSNs to enhance the transmission and hence reduce the energy consumed by the sensor.

A key principle for reducing the energy consumption of a WSN communication system is to use the energy efficiently according to the communication requirements. For example, overachieving the communication requirements of the target applications, such as data rate or communication range may result in unnecessary energy waste. In addition, with an optimised specification based on the communication requirements, an efficient design, employing appropriate coding techniques, algorithms and hardware implementations, is also important for further reducing the system energy consumption.

1.4.2 Error-correcting code for energy efficient wireless communication

There are some existing standards which have been applied in previous WSNs applications and research works, such as IEEE 802.15.4 [19, 35, 95] and Medical Implant Communication Service (MICS) [96]. New standards are also being developing for potential BANs applications, such as IEEE 802.15.6. In contrast to standards for long range or high speed applications, such as WLAN, mobile phones and satellites communications, the WSN standards are designed to be simple and efficient. More specifically, they are intended to have a simple implementation requiring a small amount of hardware resource and having low energy consumptions. As a result, the existing communication protocols tend to employ simple codes that are easy to implement. For example, in the IEEE 802.15.4 protocol, a simple 4-bit symbol to 32-bit chip sequence spreading scheme is used [95].

This is in contrast to more sophisticated ECCs, namely near Shannon limit error correcting codes, such as turbo codes [97] and Low-Density Parity-Check (LDPC) codes [98]. The complex decoders of these sophisticated ECCs require more processing energy consumption E^{pr} by the hardware implementation, but facilitate a lower transmission energy consumption E^{tx}.

The ECCs that are adopted by existing WSN standards are appropriate for short range communications. For example, the typical transmission range of IEEE 802.15.4 is 10-75 metres [99]. At these communication ranges, the transmission energy consumption E^{tx} is low enough that the processing energy consumption E^{pr} introduced by the coding schemes is not negligible. Considering the overall energy consumption $E^{\mathrm{tx}}+E^{\mathrm{pr}}$, simple ECC schemes are motivated for these standards. In general, from the energy efficiency point of view, sophisticated ECC codes are more suitable to long range communication, since E^{tx} increases exponentially with the communication range. In this case, E^{pr} becomes negligible compared to E^{tx}. In these circumstances, a sophisticated ECC can reduce E^{tx} by a significant amount, which is greater than its energy consumption $E^{\mathrm{pr}}_{\mathrm{b}}$. Depending on the ECC scheme and its implementation, there is a critical distance, beyond which the ECC scheme reduces the overall energy consumption $E^{\mathrm{tx}} + E^{\mathrm{pr}}$ and can be considered to be energy efficient. The critical distance is also high dependent on the particular transmission environment, since the path loss exponent n can be highly variable, as discussed in Section 1.1.8 and 1.3.4.

However, as the Integrated Circuit (IC) process technology is scaled down, the energy consumption E^{pr} of a particular ECC reduces exponentially. This also reduces the critical distance of sophisticated ECCs, motivating their employment in WSNs applications. Moreover, as discussed in Section 1.1.1, some WSN applications have relatively long communication ranges, which prevent reliable communication using the short range standards, without employing an excessive transmission energy E^{tx}. As a result, near Shannon limit ECCs have recently been considered to improve the energy efficiency of WSNs [1,2,89,90]. The near Shannon limit performance of sophisticated ECCs facilitates greater coding gains than the simple ECCs that are employed in conventional communication protocols for WSNs. This allows a lower transmission energy to be employed, without increasing the transmission error rate. In [2], a selection of ECC schemes and their implementations were considered and found to give critical distances that vary from several metres to several tens of metres. In addition, near Shannon limit ECCs are attractive in scenarios where the permitted transmission energy is constrained. For example, in BANs, the transmission energy may be limited in order to avoid human body health issues. Furthermore, it is often challenging to achieve reliable communication in BANs scenarios, owing to their high path losses. Therefore, a near Shannon limit ECC is desirable, since it allows limited transmission energy communications over malignant channels with very low error rates.

In this thesis, the design and implementation turbo-like codes for WSNs is studied. Turbo-like codes are a family of near Shannon limit ECCs that has been widely employed in wireless communication systems. However, it has been hardly considered for use in energy-constrained systems such as WSNs before. For this reason, the algorithm and implementational design of turbo-like codes have been previously particularly focused on achieving a high coding gain and throughput, rather than a high energy efficiency. In this thesis, the potential of applying turbo-like code for the purpose of improving the communication system's energy efficiency is explored. In particular, the holistic design of turbo-like coding algorithms and implementations is considered for improving the energy efficiency of the system.

1.5 Objectives and organisation of the thesis

As discussed in Section 1.4.2, the near Shannon limit performance of turbo-like codes can facilitate low transmission energy consumption E^{tx} in wireless communication schemes. Energy-constrained applications such as WSNs may benefit from this reduction of transmission energy consumption E^{tx} in order to extend the lifetime of the system. However, in realistic scenarios, the additional processing energy consumption E^{pr} introduced by employing the turbo-like code in the system must be considered as an offset against the energy saving. The main objective of this thesis is to explore methods for holistically designing a turbo-like coding algorithm and implementation, in order to reduce the overall energy consumption of the wireless communication system.

The outline of the thesis is shown in Figure 1.5. The chapters are organised as follows.

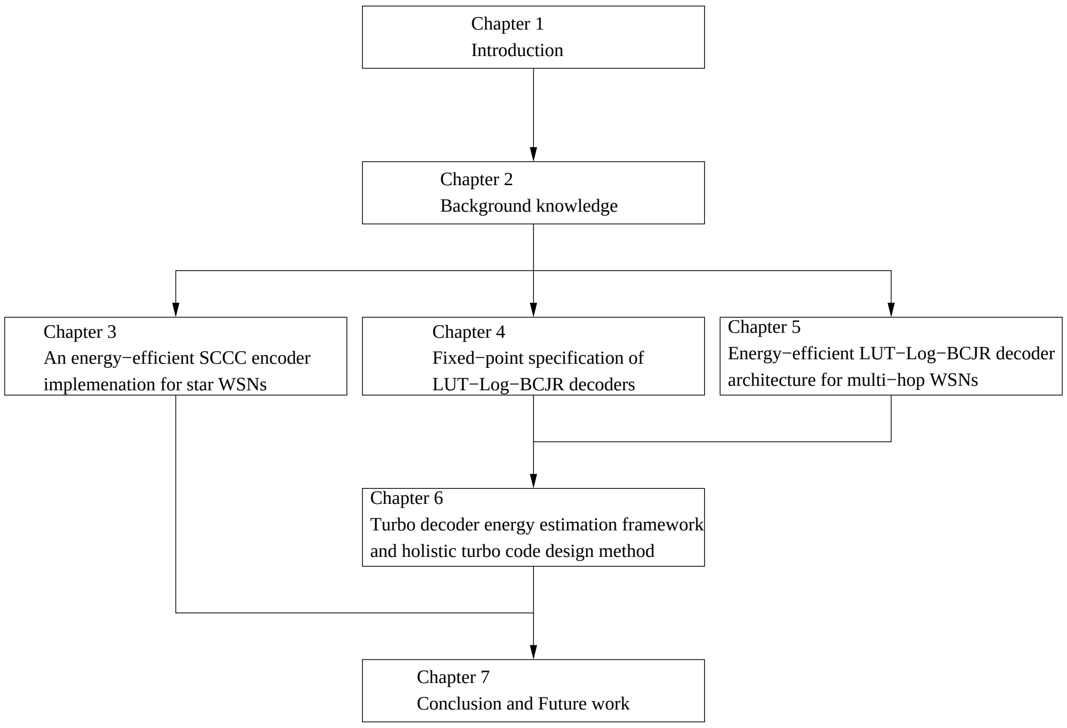

Chapter 1
Introduction
Chapter 2
Background knowledge
Chapter 3
An energy−efficient SCCC encoder implemenation for star WSNs
Chapter 4
Fixed−point specification of LUT−Log−BCJR decoders
Chapter 5
Energy−efficient LUT−Log−BCJR decoder architecture for multi−hop WSNs
Chapter 6
Turbo decoder energy estimation framework and holistic turbo code design method
Chapter 7
Conclusion and Future work

FIGURE 1.5: Thesis outline.

- Chapter 2 is a background chapter, providing the key knowledge related to this thesis. It commences with a full introduction to ECCs and turbo codes in particular. The EXtrinsic Information Transfer (EXIT) chart [100] analysis method for turbo-like codes is presented. The fixed-point implementational issues of hardware design for ECC algorithms are also discussed.

- In Chapter 3, the simple but widely-applied star network scenario of WSNs is investigated as a special case of WSNs. WSN applications having a star network topology have a unique character, namely that on one-way communication is required, from the sensor nodes to the central node. As a result, the employment of turbo-like codes in star WSNs requires only the encoding system to be implemented in the sensor nodes, the decoding system is only required in the central node. The encoders of turbo-like codes, such as the Serial Concatenated Convolutional Code (SCCC) [101] and turbo code typically have very low complexities. In this section, the benefits associated with saving energy consumption by employing a SCCC in star sensor networks is explored. In particular, an augmentation of the IEEE 802.15.4 [102] channel coding scheme is implemented for the investigation.

The results presented in Chapter 3 show that the proposed SCCC encoder has only an insignificant energy consumption compared to the transmission energy reduction that it affords. On the other hand, the decoder that is deployed on the central node has a high complexity and hence high energy consumption. Therefore, the result of employing the SCCC channel coding system in a star WSN is that the energy consumption of the system is redistributed from the sensor nodes to the central node. This energy redistribution is ideal for the purpose of extending the lifetime of the WSNs. This is because in WSNs, the sensor nodes typically have limited energy resources due to their small size. However, the central node can typically afford to have a larger size and is often integrated into a higher-level system having abundant energy resources.

- Following the star network investigation of Chapter 3, more complicated topologies are considered in Chapter 4, 5 and 6 of this thesis. In many applications, complicated network topologies involving multi-hop communications among the sensor nodes are required for WSNs. For example, when the WSN is required to cover a large area or include a large number of sensor nodes, as discussed in Section 1.1.1. Alternatively, multi-hop is required when the transmission power is limited by other issues, such as human health protection, as discussed in Section 1.4.2. In multi-hop networks, it becomes necessary to deploy high-complexity turbo-like decoders in not only the central node, but also the sensor nodes. As a result, the energy consumed by the decoder E^{pr} may significantly offset the energy saving gained by the reduced transmission energy consumption E^{tx}. According to previous studies on this topic [1, 2, 89, 90], the energy efficiency of the decoder is a key issue for applying turbo-like codes in WSNs in order to reduce the overall energy consumption of the system. Chapter 4 proposes a method for investigating the effect on the decoding performance when using different word length settings for a fixed-point turbo decoder. Since hardware complexity is directly related to the word length setting, determining the minimum word length requirement of a turbo decoder without significantly affecting its decoding performance is an important issue for achieving an energy-efficient hardware implementation.

- In the conventional design procedure of turbo codes, the processing energy consumed by the hardware is not typically considered in detail during the design. This is because turbo-like codes are conventionally applied in long range and high speed communication systems, such as the 3GPP Long Term Evolution (LTE) [103] and Digital Video Broadcasting (DVB) [104]. In these applications, the processing energy consumption is insignificant compared with the transmission energy consumption. As a result, the decoding throughput becomes the highest priority in conventional turbo decoder design. The hardware complexity and energy efficiency are traded off for high decoding throughput. In Chapter 5, a novel Application-Specific Integrated Circuit (ASIC) architecture is proposed for a widely used type of turbo decoder, namely the Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) [105] decoder, which has a low energy consumption and complexity when used in WSN applications.

- Conventionally, the turbo-like decoder's processing energy consumption can only be estimated at the hardware implementation stage of the design process, when it is already too late to make any changes to the code design. In Chapter 6, a holistic turbo code design method is proposed to consider both the transmission energy consumption E^{tx} and the processing energy consumption E^{pr}, aiming for an overall energy optimised design for WSN applications. In order to realise the holistic design method, a framework for estimating the energy consumption of the LUT-Log-BCJR decoder architecture at an early code design stage is proposed. By using the framework and a relevant path loss model for the transmission energy estimation, a holistic design method for designing a turbo code having an optimised overall energy consumption is proposed. This allows the hardware energy consumption to be considered during the design of the turbo coding algorithm. This overall optimisation of the energy consumption allows the communication system to take full advantage of the turbo code's near Shannon limit performance. A case study is presented, in which the proposed method is used to investigate a previous turbo code design work [106]. The advantages of using the holistic design method from the energy efficiency point of view, compared with the conventional design method, are demonstrated.
- Chapter 7 concludes the thesis and discusses opportunities for future work.

1.6 Novel Contributions

The novel contributions of this thesis are listed below.

- In Chapter 3, a SCCC channel coding scheme based on an augmentation of the IEEE 802.15.4 PHY is proposed by Dr. Rob Maunder [69] for WSN applications. The augmentation achieves a desirable redistribution of the energy consumption in WSNs from the sensor nodes to the central node. An ASIC design of the augmentation is proposed and implemented in this chapter. At the cost of increasing the decoding complexity of the central nodes, this approach significantly reduces the communication energy consumption of the sensor nodes.
- Chapter 4 proposes a novel EXIT chart analysis technique for investigating the trade-off between complexity and performance that is associated with the word length setting of fixed-point turbo code implementations. Conventionally, BER simulations are used for this task. However, the BER result can only provide the performance of a particular fixed-point turbo code implementation. By contrast,

EXIT chart analysis provides the convergence behaviour information of the decoder, which offers insights into the reasons causing the performance degradation due to the limited word length in fixed-point implementations. These analysis results may help the designer to find desirable parameterisations or adjust the design to have a lower word length requirement for certain applications. In addition, rather than providing results only for one paricular number of decoding iterations as a BER simulation does, an EXIT simulation allows an arbitrary number of iterations to be considered. In this way, the proposed approach provides an exhaustive investigation of the candidate turbo code with a significantly lower simulation time than the BER simulation based method.

- In Chapter 5, an energy-efficient LUT-Log-BCJR decoder architecture for ASIC implementation is proposed for WSN applications. Unlike the conventional LUT-Log-BCJR architecture, which uses dedicated modules to perform different tasks and achieve a high throughput, the proposed LUT-Log-BCJR architecture performs a number of Add-Compare-Select (ACS) operations in parallel during each clock cycle, using ACS units that are designed to have a low gate count and a short critical path. As a result, the proposed architecture uses the available hardware resources more efficiently. The low complexity and balanced data paths in the architecture reduces the wastage of energy that is caused by spurious transitions and clock tree distribution. The short critical path of the proposed architecture allows it to operate at a high clock frequency without requiring complicated combinational logic to control the data path lengths. It therefore reduces the energy wastage associated with static power consumption.

- In Chapter 6, the proposed LUT-Log-BCJR architecture is employed to derive a bottom-up energy estimation framework for estimating the turbo decoder energy consumption at an early design stage. The framework is based on generalised analysis and individual post-layout simulations of the sub-modules of the proposed architecture. The EXIT chart analysis method of Chapter 4 for investigating the fixed-point specification of the hardware implementation can be used to provide fixed-point specification information for the framework. As a result, the framework allows a turbo code designer to predict the energy consumption of their turbo decoder at early design stage, facilitating an overall energy optimised design.

- In Chapter 6, the proposed framework is used as the basis of a holistic design procedure for designing an overall energy optimised turbo code design for WSN applications. By considering both the transmission energy consumption E^{tx} and the processing energy consumption E^{pr}, the holistic design procedure allows a turbo code design to be overall energy optimised for a particular scenario, having a paricular path loss exponent and transmission range.

Chapter 2

Turbo codes, extrinsic information transfer charts and the fixed-point representation

This chapter introduces all the background information related to this thesis, including the basic principle of turbo-like codes, the encoding and decoding schemes of turbo codes, EXtrinsic Information Transfer (EXIT) chart analysis for turbo-like codes and fixed-point number representation in Application-Specific Integrated Circuit (ASIC) designs.

2.1 Turbo-like codes

The turbo principle is a concept of Error-Correcting Codes (ECCs) that include iterative decoding processes, also referred to as turbo decoding processes, such as serial or parallel concatenated codes [97, 101]. A unique feature of turbo-like codes is that they include two or more concatenated component codes. These types of codes were first proposed in [107]. The concatenation between the component codes in the encoding process could be parallel or serial, as shown in Figure 2.1. In a Parallel Concatenated Code (PCC), the inputs of the two encoders come from the same source; in a Serial Concatenated Code (SCC), one encoder's output provides the other encoder's input. An interleaver, which rearranges the data in a non-contiguous and random way, is used between the component encoders. The success of turbo-like codes is that they introduce an iterative decoding process to approach the optimal decoding performance. The two decoding schemes corresponding to the two encoding schemes are given in Figure 2.2. An iterative decoding process is performed between the two concatenated decoders by feeding the decoded results back to each other's input. In these schemes, the decoded result improves in each iteration, until the best result is achieved after a certain number of iterations.

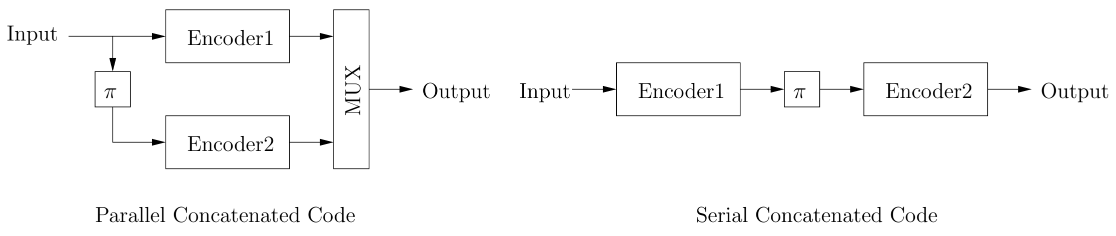


FIGURE 2.1: Two concatenation way of turbo-like codes.

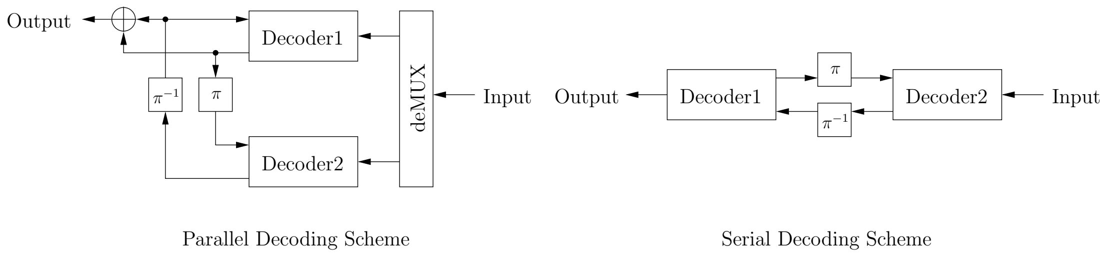


FIGURE 2.2: Two decoding schemes of two types of turbo-like codes.

The first version of concatenation codes was SCC. A famous example consists of a Reed-Solomon code [108] as the outer code (applied first and removed last) and followed by a convolutional code [109] as the inner code (applied last and removed first) [110]. In the early concatenated coding schemes, despite including two or more component codes, there is no iterative decoding process in the decoder. The decoder generates hard decisions (i.e. the determined bit results) directly. In a communication receiver, the demodulator can produce soft decisions in the demodulation process. Instead of giving the decoded bit results, soft decisions are reliability information expressed by the a posteriori probability of each bit. Soft decisions express not just what the most likely value of a bit is, but also how likely it is, while hard decisions only express the former. In the logarithmic domain, the soft decisions become Logarithmic Likelihood Ratios (LLRs) defined as:

$$\tilde{z}_i = \ln\left(\frac{P(z_i = 0)}{P(z_i = 1)}\right), \tag{2.1}$$

where $\tilde{z}_i$ is the soft decision of the received bit z_i. Before the turbo principle was discovered, a typical decoder would utilise the soft decisions in the decoding process and generate hard decisions at its output. This type of decoder is called a Soft-In Hard-Out (SIHO) decoder. Therefore, a straight forward way of decoding SCCs involves the use of a SIHO decoder for the inner decoder and a Hard-In Hard-Out (HIHO) decoder for the outer decoder. If a convolutional encoder is concerned, a Viterbi decoder [111] is used at the corresponding place to give hard decisions. As discussed in [112], the first drawback of such a structure is that the inner decoder generates hard decisions, thus preventing the outer decoder from utilizing its ability to accept soft decisions at its input. The second drawback is that if the inner decoder makes a continual error sequence, the outer decoder is typically unable to correct the errors. The second drawback can be conquered by inserting an interleaver between the inner and the outer encoder and correspondingly a deinterleaver between the inner and the outer decoder. The function of an interleaver is to rearrange the order of a sequence in a pseudo-random way. The function of a deinterleaver, with knowledge of the rearranging method of the corresponding interleaver, is to restore the order of an interleaved sequence. Thus, a continual error sequence in the inner decoder's output becomes dispersed in the input to the outer decoder. The transmission scheme is shown in Figure 2.3. However, if errors occur at the output of the

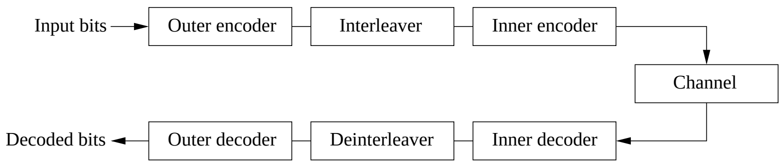


FIGURE 2.3: Transmission scheme of SCCs.

outer decoder, these would remain in the final decoded results. A turbo-like code can be considered to be a refinement of the concatenated encoding schemes employing an improved decoding process including iterative algorithms. The concept of turbo decoding is for a system with two component codes to pass soft decisions from the output of one decoder to the input of the other decoder, and to iterate this process many times in order to produce more reliable decisions. To obtain benefits from an iterative decoding process, it is required that the two decoders feed soft decisions to each other. This is because using hard decisions as an input to a decoder degrades its performance compared with soft decisions [113]. Therefore, turbo decoding requires Soft-In Soft-Out (SISO) decoders for the decoding of each component code. The introduction of turbo codes in [97] is also the first introduction of PCCs. It was reported that the scheme could achieve a Bit Error Rate (BER) of 10^{-5} using a rate 1/2 code over an Additive White Gaussian Noise (AWGN) channel and Binary Phase-Shift Keying (BPSK) modulation at an E_b/N_0 of 0.7 dB [97,114]. According to the discussion in [97,114], the near Shannon limit for BPSK modulation matches the throughput of this scheme at $E_b/N_0 = 0$ dB. Hence the performance is 0.7 dB from Shannon limit. Most importantly, owing to its iterative decoding scheme, the complexity of a turbo decoder is much less than that of a non-iterative decoder having the same performance. According to [115], the complexity required to allow the non-iterative codes to approach the Shannon limit would be not feasible to implement. The discovery of turbo codes revolutionised the field of error correcting codes since it allowed performance very close to the near Shannon limit in practice.

To evaluate the performance of a turbo or turbo-like code, a BER chart is a commonly used tool. A typical BER chart of turbo codes is shown in Figure 2.4. Here, the Y axis is the BER of the decoding result after a certain number of decoding iterations and the X axis is E_b/N_0, where E_b is the transmission energy per bit and N_0 is the noise power spectral density (i.e. noise power in a 1 Hz bandwidth). As shown in Figure 2.4, a typical turbo code can achieve very low BER when the E_b/N_0 exceeds a certain threshold. The region where the BER curve rapidly falls is called the turbo

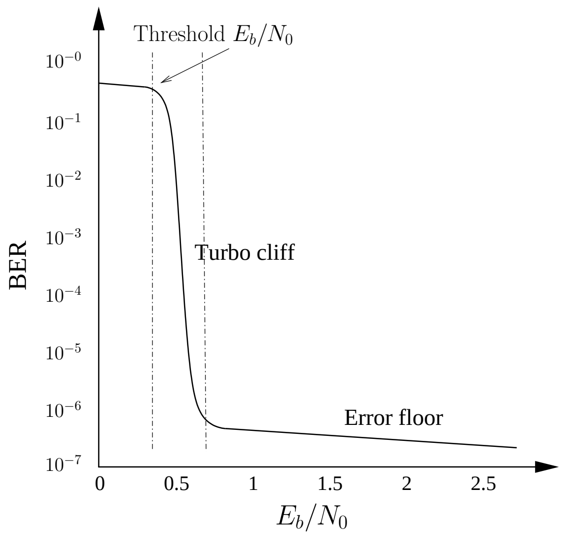


FIGURE 2.4: A typical BER chart for turbo codes.

cliff region and the region where the BER curve is flat at a very low BER is called the error floor region. To understand how the turbo codes outperform the earlier coding schemes, Figure 2.5 is quoted from [115]. This shows simulation results of the original

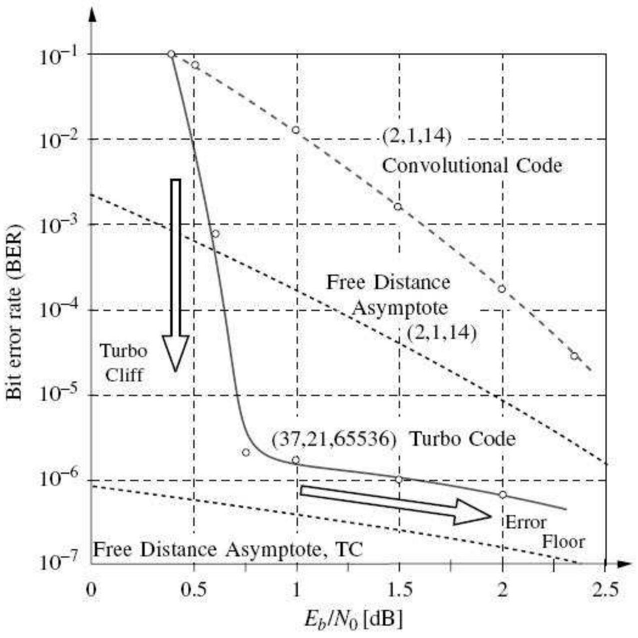


FIGURE 2.5: Performance comparison of a turbo code and a convolutional code. ©C. Schlegel *et al.* 2004 [115]

rate R=1/2 turbo code presented in [97] and a maximum free distance (MFD) R=1/2, memory $\nu = 14$ convolutional code with Viterbi Algorithm (VA). The simulation results show that the turbo code outperforms the convolutional code by 1.7 dB at a BER of 10^{-5}. The comparison is distinct, especially since a detailed complexity analysis reveals that the complexity of the turbo decoder is much smaller than the Viterbi decoder used for the convolutional code.

Based on the turbo principle, a classical turbo encoder is composed of two Recursive Systematic Convolutional (RSC) encoders, as shown in Figure 2.6. In addition to Figure 2.1,

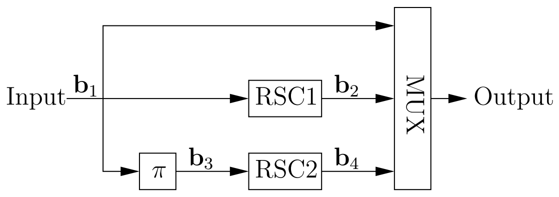


FIGURE 2.6: A classical turbo encoder scheme.

a systematic output is usually included in practice. The input information sequence is encoded twice by the two RSC encoders. The first encoder processes the information $\mathbf{b}_1$ in its original order, while the second encoder processes the same sequence, but in a different order $\mathbf{b}_3$ that is imposed by an interleaver. In this scheme, the systematic bit sequence is also transmitted to the decoder. As shown in the figure, sequences $\mathbf{b}_2$ and $\mathbf{b}_4$ are the outputs of each encoder. Sequence $\mathbf{b}_1$ is the systematic bit sequence and $\mathbf{b}_3$ is the interleaved systematic bit sequence. Note that $\mathbf{b}_3$ is not transmitted since it can be obtained by an identical interleaver in the decoder.

In the decoding process, as shown in Figure 2.7, two A Posteriori Probability (APP) decoders are used correspondingly for the two convolutional encoders in the encoding scheme. In the figure, $\tilde{\mathbf{b}}_1$, $\tilde{\mathbf{b}}_2$ and $\tilde{\mathbf{b}}_4$ are the LLR sequences corresponding to the bit

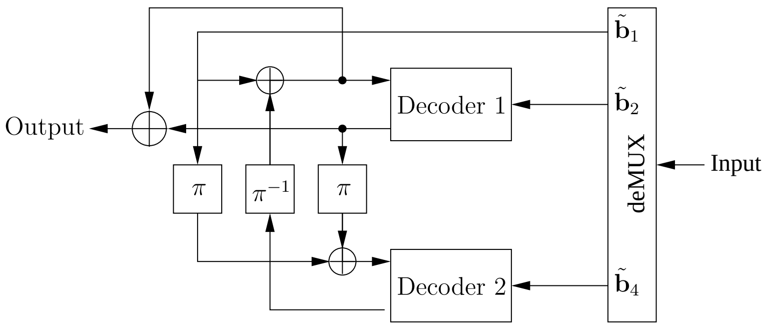


FIGURE 2.7: A classical turbo decoder scheme.

sequences $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_4$ in Figure 2.6. The purpose of an APP decoder is to compute a posteriori probabilities on either the information bits or the encoded symbols. Its application in turbo-like codes was made it become the major representative of the SISO decoders. The algorithm was originally invented by Bahl, Cocke, Jelinek and Raviv in 1972, as the so called Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [105]. Its capability of generating soft decisions is well suited for iterative decoding. In Figure 2.7, the two decoders are working alternatively in an iterative way. To get the correct order of the input sequences, an identical interleaver with the one used in the encoding scheme and a corresponding deinterleaver is used between the decoders. Furthermore, an extra interleaver is used for providing the systematic sequence for both of the decoders. The main advantage of this decoding process compared with using the Viterbi decoders is that it utilises the ability of the decoders to accept soft decisions at their input. However, in iterative decoding schemes, the information provided for one decoder from the other one, is extrinsic information instead of a posteriori information. The extrinsic information represents only the new information obtained by a decoder. The reason for using extrinsic information is to prevent the decoding scheme from being a positive feedback amplifier [112]. As shown in Figure 2.7, the a priori information from the systematic sequence is added to the input of the decoders. Since the a posteriori information already includes the a priori information from the previous decoding process from the other decoder, this would create a positive feedback amplifier in the loop. By using extrinsic information instead of a posteriori information, this problem can be solved. Therefore, the output of the decoders in Figure 2.7 is extrinsic information. It can be obtained using a simple subtraction between the a posteriori and the a priori LLRs. Alternately, it can also be generated directly by a modified BCJR algorithm. By receiving the new extrinsic information from the other decoder, the reliability of the decoding increases in each iteration. The whole decoding process stops when the required reliability is reached or until no further reliability can be gleaned.

A further improved version of the BCJR algorithm is called the Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm [115]. This is a transferred version of BCJR algorithm into the logarithmic domain. Its purpose is to avoid the high number of multiplication operations that are required in the original BCJR algorithm and more importantly, the Log-BCJR algorithm has variables with a much more manageable dynamic range than those of the BCJR algorithm, reducing the memory requirement and allowing fixed-point processing to be used. Since it avoids the complex circuit implementation due to many multipliers required by the original BCJR algorithm and requires much less memory, the Log-BCJR algorithm is widely used in practice. Hence, in this report, since only the practical applications of the BCJR algorithm in iterative decoding schemes is investigated, the discussed and simulated algorithm is the modified Log-BCJR algorithm that generate the extrinsic information directly. The detail of the algorithm is discussed in the next section.

The turbo principle can also be applied to SCCs, which is another primary category of turbo-like codes, referred to as Serial Concatenated Convolutional Codes (SCCCs) [101]. Instead of using the decoding scheme in Figure 2.3, a scheme similar to the turbo decoder is used, as shown in Figure 2.2. The two SIHO decoders are replaced with SISO decoders and an interleaver and a deinterleaver are required to form the iterative decoding loop. According to [112], serial turbo-like codes perform better than parallel turbo codes in the error floor region. On the other hand, in the turbo cliff region, parallel turbo codes perform better with the same overall coding rate. Moreover, Low-Density Parity-Check (LDPC) codes [98] is another recently widely applied near Shannon limit ECC that applies the turbo principle using an iterative decoding scheme.

2.2 Turbo encoder and decoder

Turbo-like codes generally have a simple encoding scheme and a relatively complicated decoding scheme owning to the high computation complexity of the Log-BCJR algorithm. In this section, the turbo code in 3rd Generation Partnership Project (3GPP) Universal Mobile Telecommunications System (UMTS)/Long Term Evolution (LTE) Standards [103,116] is used as an example to introduce the typical turbo coding schemes and the widely used decoding algorithm for turbo codes, the Log-BCJR algorithm. The UMTS/LTE turbo code is also used as an application example throughout this thesis, so the description in this section is frequently referred to in later chapters.

2.2.1 Universal Mobile Telecommunications System (UMTS)/Long Term Evolution (LTE) turbo encoder and decoder schematics

To simplify the description, assuming BPSK modulation is used in this case, each transmitted symbol conveys one bit. For other modulation methods, the transmitted bits here would be replaced by transmitted symbols. According to [103, 116], the concatenated RSC encoders of UMTS/LTE turbo code is a rate R=1/2, K=4 constraint length and m=3 memory convolutional code. Two identical RSC encoder of this type form the rate 1/3, 8-state Parallel Concatenated Convolutional Code (PCCC) illustrated in Figure 2.8. As shown in the figure, each rate R=1/2 encoder gives an encoded output sequence. In

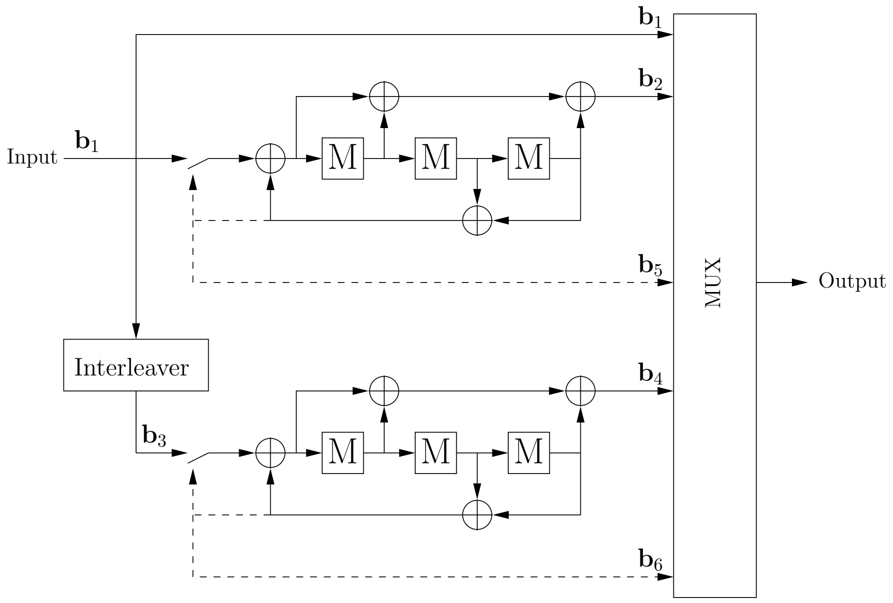


FIGURE 2.8: Scheme of the UMTS/LTE turbo encoder.

addition, the systematic sequence is also given as an output. Since the systematic output of both encoders are identical, the whole encoder has a rate approaching R=1/3. In the RSC encoder, the three memory bits forms an 8-state Finite-State Machine (FSM).

Notation N is defined to represent the length of a bit sequence. The length of uncoded sequence $\mathbf{b}_1$ is $N_{\mathrm{b}1}$. Before the encoding of the bit sequence $\mathbf{b}_1$ commences, the shift registers of each concatenated convolutional code are initialised in a state that is known to the receiver. Typically, the m=3 memory elements of each shift register are initialised with logic-zeros, placing them in what is referred to as the 'all zeros' state. However, following the encoding of the $N_{\mathrm{b}1}$ bits in the sequence $\mathbf{b}_1$, the shift registers will enter states that are not inherently known to the receiver. A number of techniques have been proposed to cope with this [112]:

- No termination: In this case, in the decoding process, the end of a block sequence is considered to have a equivalent possibility of each possible state. No information of the final state is provided to the decoder. The decoding process is then less effective for the last bits of the encoded data and the performance may be reduced. This degradation is a function of the block length. However, for some applications the degradation may be acceptable.

- Termination: This method involves the transmission of several extra bits (three bits in the UMTS/LTE example) at the end of each block sequence to force the encoder return to the 'all zero' state. The UMTS/LTE turbo code of Figure 2.8 is an example of this technique. The extra tail bits on $\mathbf{b}_2$ and $\mathbf{b}_4$, and the extra sequences $\mathbf{b}_5$ and $\mathbf{b}_6$, also need to be sent to the decoder. This method conquers the uncertain final state issue but induces another two drawbacks. Firstly, extra redundancy information is added to the transmission. Nevertheless, the redundancy is negligible except for very short blocks and it is useful for error correction. Secondly, for parallel codes, the tail bits $\mathbf{b}_5$ and $\mathbf{b}_6$ are not identical for each constituent code, which means in the iterative decoding process, the extrinsic information of the tail bits cannot be exchanged between the decoders. Hence, the data at the end of the block sequence will get less benefit from the turbo decoding process. The SCCC also has a similar problem.

- Tail-biting: [117] introduced a technique which allows any state of the encoder as the initial state. This method involves a double encoding process: Firstly, a normal encoding of the sequence starting from 'all zero' state is performed, but the output of the encoder is ignored. Only the final state of the encoder is stored. Secondly, the encoding process is performed again in order to actually generate the output. In this step, the initial state is a function of the final state previously stored. The result of this process is the final state of the encoder is equal to its initial state. The advantage of this method is no extra bits have to be added and transmitted. However, the double encoding process is the main drawback of this method. In addition, it only works for the convolutional codes where the BCJR algorithm is especially adapted.

In the UMTS/LTE turbo code, the termination technique is used as shown in Figure 2.8. The initial states of the shift registers are all set to zeros before starting to encode the bit sequence $\mathbf{b}_1$. Note that once the encoding of $\mathbf{b}_1$ is finished, the two switches in the figure switch down to form a closed loop in the two encoders. Following this is the termination process. The termination is performed by taking the tail bits from the shift register feedback after all information bits are encoded. It takes m bits to force the final state back to the 'all zero' state for each encoder. The output of the turbo encoder is $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_5$, $\mathbf{b}_4$ and $\mathbf{b}_6$, where $\mathbf{b}_1$ is the systematic bit sequence, $\mathbf{b}_2$ and $\mathbf{b}_4$ are the encoded bit sequences from the two encoders, respectively, and $\mathbf{b}_5$ and $\mathbf{b}_6$ are the termination sequence of the two encoders, respectively. As a result, in the case where $\mathbf{b}_1$ has $N_{\text{b}1}$ bits, $\mathbf{b}_2$ and $\mathbf{b}_4$ will have $N_{\text{b}2} = N_{\text{b}4} = N_{\text{b}1} + m$ bits each, while $\mathbf{b}_5$ and $\mathbf{b}_6$ will have $N_{\text{b}5} = N_{\text{b}6} = m$ bits each. The possible block length of the turbo code (i.e. the length of bit sequence $\mathbf{b}_1$) for the UMTS and the LTE standards are $N_{\text{b}1} \in [40, 5114]$ and $N_{\text{b}1} \in [40, 6144]$, respectively. For the interleaved sequence $\mathbf{b}_2$, the length is $N_{\text{b}2} = N_{\text{b}1}$. The termination bits $\mathbf{b}_5$ and $\mathbf{b}_6$ have a length equal to the number of memory bits in the RSC encoders, which is $m = 3$. Consequently, for the encoded sequence $\mathbf{b}_2$ and $\mathbf{b}_4$, there is $N_{\text{b}3} = N_{\text{b}4} = N_{\text{b}1} + 3$. Note that the additional termination bits ($\mathbf{b}_5$ and $\mathbf{b}_6$) make the coding rate R of the encoder is $\frac{N_{\text{b}1}}{3N_{\text{b}1}+12}$, which is slightly lower than 1/3.

To understand the operation of the FSM in the component encoders, the transitions between the stages can be shown as the transition diagrams in Figure 2.9. $\mathbf{b}_1 = \{b_{1,j}\}_{j=1}^{N_{\text{b}1}}$,

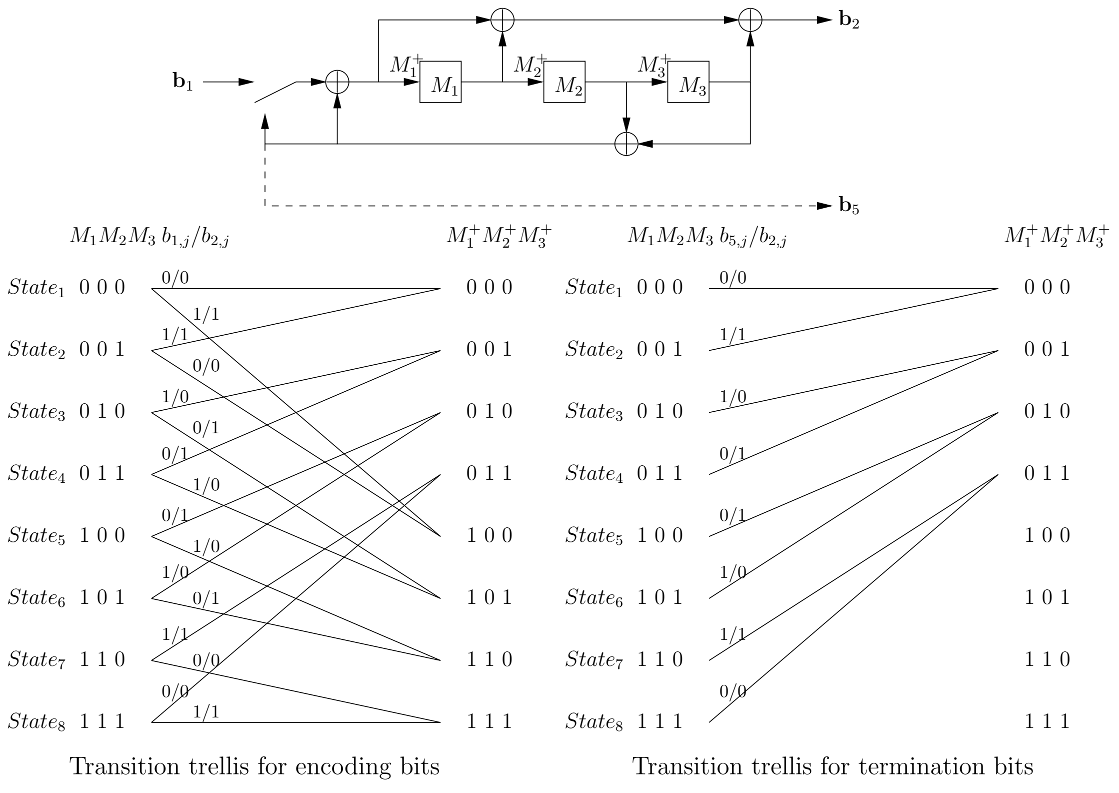


FIGURE 2.9: Scheme of the convolutional encoder and the state transition diagrams.

$\mathbf{b}_2 = \{b_{2,j}\}_{j=1}^{N_{\mathrm{b1}}+3}$ and $\mathbf{b}_5 = \{b_{5,j}\}_{j=N_{\mathrm{b1}+1}}^{N_{\mathrm{b1}+3}}$ are the input sequence and the output sequence. M_1, M_2 and M_3 are the current values of the three memory bits in the encoder. M_1^+, M_2^+ and M_3^+ are the next values of the three memory bits. The transition of the values and the decoding results can be expressed as the following equations.

- For encoding bits, where $j \in [1, N_{\mathrm{b1}}]$.

$$M_1^+ = b_{1,j} \oplus M_2 \oplus M_3 \tag{2.2}$$

$$M_2^+ = M_1 \tag{2.3}$$

$$M_3^+ = M_2 \tag{2.4}$$

$$b_{2,j} = M_1^+ \oplus M_1 \oplus M_3 \tag{2.5}$$

- For termination bits, where $j \in [N_{\mathrm{b1}} + 1, N_{\mathrm{b1}} + 3]$.

$$M_1^+ = 0 \tag{2.6}$$

$$M_2^+ = M_1 \tag{2.7}$$

$$M_3^+ = M_2 \tag{2.8}$$

$$b_{5,j} = M_2 \oplus M_3 \tag{2.9}$$

$$b_{2,N_{\mathrm{b1}}+j} = 0 \oplus M_1 \oplus M_3 \tag{2.10}$$

The eight possible states are corresponding to the $State_1$ to $State_8$ as shown in the figure. The trellis diagrams gives all the possible transitions of the FSM. The left trellis diagram shows the transitions for the encoding bits in a sequence. The right diagram shows the transitions for the termination bits in a sequence. Note that the first state in a transition sequence is 'all zero', which is the $State_1$ in the figure. With the termination technique, the last state in the sequence is forced back to $State_1$. This causes the possible transitions at the first three steps and the last three steps to be limited. A transition trellis diagram of a transition sequence is shown in Figure 2.10. The input sequence is $\{b_{1,j}\}$ and the output sequence $\{b_{2,j}\}$ and $\{b_{5,j}\}$ can be obtained by tracking the state transition in Figure 2.10. For instance, for a 5-bit input sequence example $\mathbf{b}_1 = [0, 1, 1, 0, 1]$, the transitions in the trellis is shown in Figure 2.11. Note that there are 8 steps in the trellis since $m = 3$ termination bits are included. The encoded bit sequence would be $\mathbf{b}_2 = [0, 1, 0, 0, 1, 0, 1, 1]$ and the actually transmitted systematic bit sequence is $[\mathbf{b}_1, \mathbf{b}_5] = [0, 1, 1, 0, 1, 1, 0, 1]$. The trellis diagram is not only helpful to understand the encoding operations of a convolutional code, but also useful to explain the Log-BCJR decoding algorithm, as shall be discussed later.

The architecture of the decoder is as shown in Figure 2.12. A feedback loop is formed between Decoder 1 and Decoder 2 to realise the iterative decoding process. Each iteration consists of two half iterations, one for each constituent RSC code. The two decoders operate alternately since the input of one decoder includes the output of the

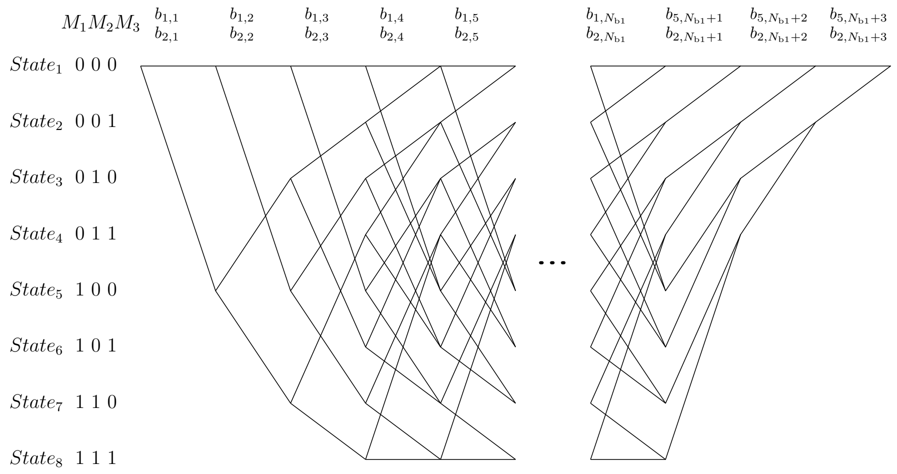


FIGURE 2.10: The trellis diagram of the convolutional code of Figure 2.9.

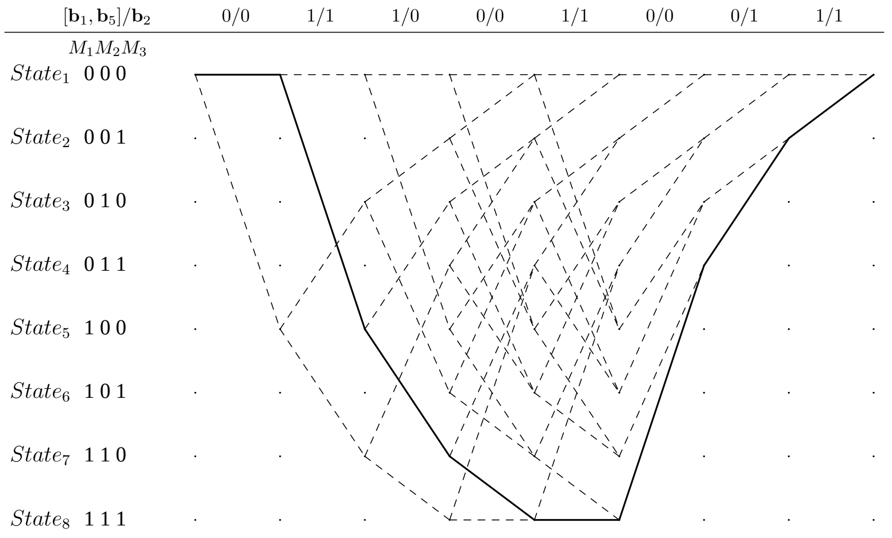


FIGURE 2.11: A example transition sequence in the trellis diagram of Figure 2.10.

other decoder from the previous half iteration. The operation of the RSC decoder (i.e. the Log-BCJR algorithm) is described in Section 2.2.2. In the figure, the input of the decoding scheme is assumed to be a sequence of LLRs. The five input $\tilde{\mathbf{b}}_1^{\mathrm{a}}$, $\tilde{\mathbf{b}}_2^{\mathrm{a}}$, $\tilde{\mathbf{b}}_4^{\mathrm{a}}$, $\tilde{\mathbf{b}}_5^{\mathrm{a}}$ and $\tilde{\mathbf{b}}_6^{\mathrm{a}}$ are the input LLR sequences corresponding to the coded output $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_4$, $\mathbf{b}_5$ and $\mathbf{b}_6$ in the encoding scheme. Each decoder receives two LLR sequences. One is the LLR sequence corresponding to the encoded sequence, which is received from the transmission channel directly ($\tilde{\mathbf{b}}_2^{\mathrm{a}}$ for Decoder 1 and $\tilde{\mathbf{b}}_4^{\mathrm{a}}$ for Decoder 2). The other is the uncoded a priori LLR sequence from the other decoder ($\tilde{\mathbf{c}}_1^{\mathrm{a}}$ for Decoder 1 and $\tilde{\mathbf{c}}_2^{\mathrm{a}}$ for Decoder 2), which is simply formed by adding the extrinsic information provided by the other decoder ($\tilde{\mathbf{b}}_1^{\mathrm{a}'}$ and $\tilde{\mathbf{b}}_2^{\mathrm{a}'}$) to the received systematic information ($\tilde{\mathbf{b}}_1^{\mathrm{a}}$). The extrinsic information ($\tilde{\mathbf{b}}_1^{\mathrm{e}}$ and $\tilde{\mathbf{b}}_2^{\mathrm{e}}$) generated by the decoders need to be rearranged by the proper interleaver (π) or deinterleaver (π^{-1}). For Decoder 1, the input LLR from the other decoder $\tilde{\mathbf{c}}_1^{\mathrm{a}}$ is the sum of the $\tilde{\mathbf{b}}_1^{\mathrm{a}'}$ and $\tilde{\mathbf{b}}_1^{\mathrm{a}}$ following with $\tilde{\mathbf{b}}_5^{\mathrm{a}}$ as shown in the figure. Because the two encoders have independent tails, the soft decisions of the tail bits are not

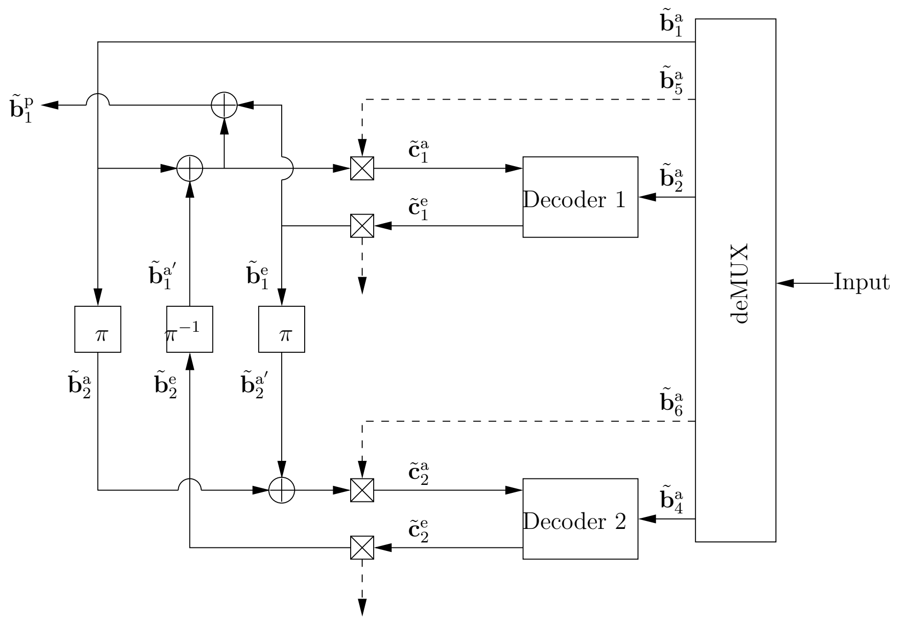

$\tilde{\mathbf{b}}_1^{\mathrm{a}}$
$\tilde{\mathbf{b}}_5^{\mathrm{a}}$
$\tilde{\mathbf{b}}_1^{\mathrm{p}}$
$\tilde{\mathbf{c}}_1^{\mathrm{a}}$
$\tilde{\mathbf{c}}_1^{\mathrm{e}}$
Decoder 1
$\tilde{\mathbf{b}}_2^{\mathrm{a}}$
$\tilde{\mathbf{b}}_1^{\mathrm{a}'}$
$\tilde{\mathbf{b}}_1^{\mathrm{e}}$
deMUX
Input
π
π^{-1}
π
$\tilde{\mathbf{b}}_2^{\mathrm{a}}$
$\tilde{\mathbf{b}}_2^{\mathrm{e}}$
$\tilde{\mathbf{b}}_2^{\mathrm{a}'}$
$\tilde{\mathbf{b}}_6^{\mathrm{a}}$
$\tilde{\mathbf{c}}_2^{\mathrm{a}}$
$\tilde{\mathbf{c}}_2^{\mathrm{e}}$
Decoder 2
$\tilde{\mathbf{b}}_4^{\mathrm{a}}$

FIGURE 2.12: Scheme of the UMTS/LTE turbo decoder.

passed between the decoders. Thus the information of the termination bits need to be considered carefully. The systematic information of the Decoder 1's termination bits $\tilde{\mathbf{b}}_5^{\mathrm{a}}$ need to be appended at the end for a complete $\tilde{\mathbf{c}}_1^{\mathrm{a}}$. Therefore, $\tilde{\mathbf{c}}_1^{\mathrm{a}} = [\tilde{\mathbf{b}}_1^{\mathrm{a}} + \tilde{\mathbf{b}}_1^{\mathrm{a}'}, \tilde{\mathbf{b}}_5^{\mathrm{a}}]$. On the other hand, for the extrinsic information generated by Decoder 1, $\tilde{\mathbf{c}}_1^{\mathrm{e}}$, the information of the termination bits need to be removed before it is interleaved and passed to Decoder 2, as shown in the figure. Therefore, the length of $\tilde{\mathbf{c}}_1^{\mathrm{a}}$ and $\tilde{\mathbf{c}}_1^{\mathrm{e}}$ are $N_{\mathrm{c}1} = N_{\mathrm{b}1} + 3$. For Decoder 2, respectively, the uncoded a priori information is the sum of $\tilde{\mathbf{b}}_2^{\mathrm{a}'}$ (i.e. interleaved $\tilde{\mathbf{b}}_1^{\mathrm{e}}$) and interleaved systematic information $\tilde{\mathbf{b}}_2^{\mathrm{a}}$. Therefore, $\tilde{\mathbf{c}}_2^{\mathrm{a}} = [\tilde{\mathbf{b}}_2^{\mathrm{a}} + \tilde{\mathbf{b}}_2^{\mathrm{a}'}, \tilde{\mathbf{b}}_6^{\mathrm{a}}]$ and the same processing of the termination bits is applied on $\tilde{\mathbf{c}}_2^{\mathrm{a}}$ and $\tilde{\mathbf{c}}_2^{\mathrm{e}}$. The length of $\tilde{\mathbf{c}}_2^{\mathrm{a}}$ and $\tilde{\mathbf{c}}_2^{\mathrm{e}}$ are $N_{\mathrm{c}2} = N_{\mathrm{b}2} + 3 = N_{\mathrm{b}1} + 3$. In the first iteration, $\tilde{\mathbf{b}}_2^{\mathrm{e}}$ is initialised with a sequence of zero valued LLRs which implies that the values of the corresponding bits are completely unknown. $\tilde{\mathbf{b}}_1^{\mathrm{a}}$ is the received systematic information. Note that two identical copies of the interleaver employed in the encoding scheme and a corresponding deinterleaver are used between the decoders, in order to give the correct ordering of the input sequences. As discussed before, in the Log-BCJR algorithm, the extrinsic information is directly generated by the decoding algorithm, which is done inside the decoder. After all the iterations are completed, the a posteriori output of the decoding scheme is obtained by adding the final extrinsic output to the final a priori input of the Decoder 1, $\tilde{\mathbf{b}}_1^{\mathrm{p}} = \tilde{\mathbf{b}}_1^{\mathrm{a}} + \tilde{\mathbf{b}}_1^{\mathrm{e}} + \tilde{\mathbf{b}}_1^{\mathrm{a}'}$, as shown in the figure. And the SISO decoding process is then completed. Based on the soft decisions, hard bit decisions can be taken to give the final decoding result.

2.2.2 Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv algorithm

As discussed in Section 2.1, the Log-BCJR algorithm [118] is typically used in practice, because compared with the original BCJR algorithm [105], it does not require multiplications and its internal variables have a much lower dynamic range, allowing a fixed-point number representation to be employed. For reducing the computation complexity, there are two popular variations of the Log-BCJR algorithm, namely the Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) and the Maximum Logarithmic Bahl-Cocke-Jelinek-Raviv (Max-Log-BCJR) algorithms. In this section, the Log-BCJR algorithm is described in detail. The two variations are also introduced.

The two fundamental operations of the original BCJR algorithm are addition and multiplication. For $p = \ln(P)$ and $q = \ln(Q)$, multiplication in the normal domain becomes addition in the logarithmic domain.

$$\ln(PQ) = \ln(e^p e^q) = p + q \tag{2.11}$$

Addition in the normal domain can be solved by the Jacobian logarithm in the logarithmic domain, which is defined using $\max^*$, as given in Equation 2.12.

$$\ln(P + Q) = \ln(e^p + e^q) = max(p, q) + \ln(1 + e^{-|p-q|}) = \max^*(p, q). \tag{2.12}$$

The $\max^*$ function is usually computed by successive pairwise operations when there are more than two operands, since it is associative. In practice, the function $f_c = \ln(1 + e^{-|p-q|})$ can be implemented by a Look-Up Table (LUT), so the function can be performed by a select operation in the LUT. The LUT realised version of the Log-BCJR algorithm is called the LUT-Log-BCJR algorithm. In the LUT-Log-BCJR, the max operation can be performed by a compare operation between p and q. As a result, all the operations required by the LUT-Log-BCJR algorithm are either 'add', 'compare' or 'select', namely the Add-Compare-Select (ACS) operations. The other variation of the Log-BCJR algorithm, the Max-Log-BCJR algorithm, simply replaces the $\max^*$ operation with a max operation, which further reduces the computation complexity, but also induces a performance degradation.

In order to present the Log-BCJR algorithm, the RSC code employed in the UMTS/LTE turbo code is taken as an example. Figure 2.13 shows the example trellis diagram provided in Section 2.2. For Decoder 1 in Figure 2.12, $\mathbf{b}_1 = \{b_{1,j}\}_{j=1}^{N_{\mathrm{b1}}}$ are the systematic bits; $\mathbf{b}_5 = \{b_{5,j}\}_{j=N_{\mathrm{b1}}+1}^{N_{\mathrm{b1}}+3}$ are the tail bits and $\mathbf{b}_2 = \{b_{2,j}\}$ are the encoded bits. There are three tail bits used for termination, as shown in the trellis, in order to drive the encoder back into the 'all zero' state, $State_1$. For simplifying the notation, sequence $\{b_{1,j}, b_{5,j}\}$

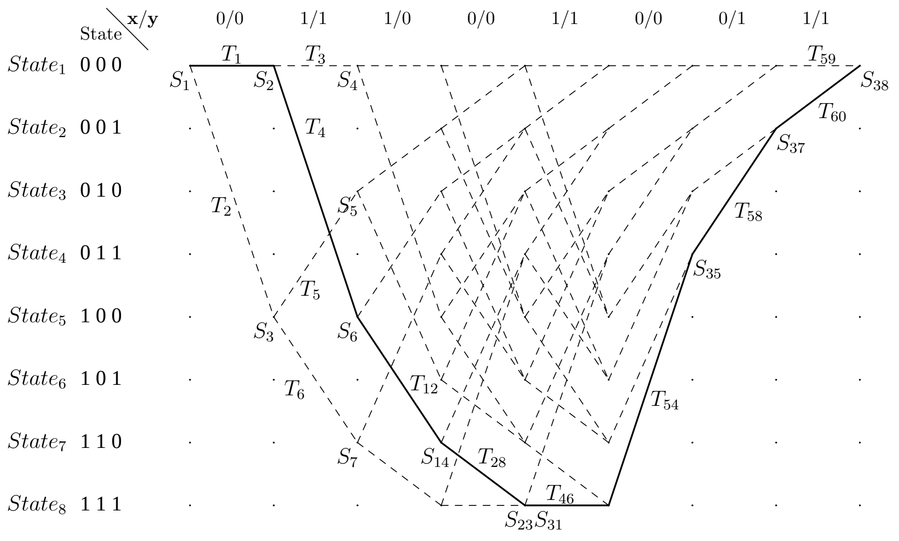


FIGURE 2.13: A example transition sequence of a short terminated trellis.

is replaced by $\mathbf{x} = \{x_j\}_{j=1}^{N_{b1}+3}$ and sequence b$_2$ = $\{b_{2,j}\}$ is replaced by $\mathbf{y} = \{y_j\}_{j=1}^{N_{b1}+3}$. The same decoding trellis can be also applied to decoder 2 in Figure 2.12.

In each step of the trellis diagram, there are 16 possible transitions, except for in the case of the three initial steps at the start and the three termination steps at the end. For a certain input systematic sequence $\{x_j\}$ and the corresponding encoded sequence $\{y_j\}$, only one transition is used in each step in a encoding trellis, as exemplified in Figure 2.13. The corresponding systematic bit and encoded bit of each transition is also given in the figure. To represent each state in Figure 2.13, notation $\{S_1, S_2, S_3, ..., S_{38}\}$ is used to identify the possible states in the trellis following the order from top to bottom and from left to right, as shown in the figure. Similarly, notation $\{T_1, T_2, T_3, ..., T_{60}\}$ is used to notate each possible transition in the trellis following the same order. In addition, notation t_j is used to represent the transition employed in the encoder trellis for the j^{th} bit. Similarly s_j is the state entered by the encoder after the j^{th} bit. Therefore, for the example sequence of $\mathbf{x}$ and $\mathbf{y}$ in Figure 2.13, the traced transition sequence is $\mathbf{t} = \{T_1, T_4, T_{12}, T_{28}, T_{46}, T_{54}, T_{58}, T_{60}\}$ and the traced state sequence is $\mathbf{s} = \{S_1, S_2, S_6, S_{14}, S_{23}, S_{31}, S_{35}, S_{37}, S_{38}\}$. For describing the algorithm, the following notations are defined.

- fr(T) is the starting state of the transition T. For example, in Figure 2.13, fr(T_1)=S_1 and fr(T_3)=S_2.
- to(T) is the ending state of the transition T. For example, in Figure 2.13, to(T_1)=S_2 and to(T_2)=S_3.
- fr(S) is the set of all transitions that start from state S. For example, in Figure 2.13, fr(S_2)=$\{T_3, T_4\}$.

- $to(S)$ is the set of all transitions that end at the state S. For example, in Figure 2.13, to(S_{38})=$\{T_{59}, T_{60}\}$.
- $\mathcal{B}_x(T)$ is the value for the bit in $\mathbf{x}$ that is implied by the transition T. For example, in Figure 2.13, $\mathcal{B}_x(T_1) = 0$ since transition T_1 implies that $x_1 = 0$. Similarly, there is $\mathcal{B}_x(T_4) = 1$.
- $\mathcal{B}_y(T)$ is the value for the bit in $\mathbf{y}$ that is implied by the transition T. For example, in Figure 2.13, there are $\mathcal{B}_y(T_1) = 0$ and $\mathcal{B}_y(T_2) = 1$.
- $\mathcal{J}(T)$ is the bit index associated with the transition T. For example, in Figure 2.13, $\mathcal{J}(T_1) = \mathcal{J}(T_2) = 1$ and $\mathcal{J}(T_3) = \mathcal{J}(T_4) = \mathcal{J}(T_5) = \mathcal{J}(T_6) = 2$.

Using the notations defined above, the Log-BCJR algorithm is described as follows. The ultimate purpose of the algorithm is to calculate the extrinsic LLRs of the decoded sequence $\tilde{\mathbf{x}}^{\mathrm{e}}$, based on the a priori LLRs $\tilde{\mathbf{x}}^{\mathrm{a}}$ and $\tilde{\mathbf{y}}^{\mathrm{a}}$ that are passed to the decoder. The calculation of the extrinsic LLRs $\tilde{\mathbf{x}}^{\mathrm{e}}$ leads to the calculations of another three groups of internal variables, γ, α and β.

- The γ values are conditional transition probabilities. In this case, the γ values is divided into two sub-groups, the a priori transition probability γ_x and the channel transition probability γ_y. They corresponding to each transition in the trellis. For each transition in each step, there is a $\gamma_x(T)$ and a $\gamma_y(T)$. For example, for a traced transition sequence $\mathbf{t}$, $\gamma_x(T)$ represents the probability $\ln[P(t_{\mathcal{J}(T)} = T|\tilde{x}^{\mathrm{a}}_j)]$, and $\gamma_y(T)$ represents the probability $\ln[P(t_{\mathcal{J}(T)} = T|\tilde{y}^{\mathrm{a}}_j)]$.
- The α values are corresponding to each state in each step in the trellis. It is the conditional probability that in step j (i.e. the decoding process is working on the trellis step that corresponding to the received x^{a}_j and y^{a}_j), the traversed path reaches a particular state S, that is $\alpha(S)$ represents the probability $\ln[P(S_{\mathcal{J}(T)} = S|\{\tilde{x}^{\mathrm{a}}_j\}_{j=1}^{\mathcal{J}(T)}, \{\tilde{y}^{\mathrm{a}}_j\}_{j=1}^{\mathcal{J}(T)})]$.
- The β values, on the other hand, are the conditional probabilities of the traversed path starting from a particular state, that is $\beta(S)$ represents the probability $\ln[P(S_{\mathcal{J}(T)} = S|\{\tilde{x}^{\mathrm{a}}_j\}_{j=\mathcal{J}(T)+1}^{N_x}, \{\tilde{y}^{\mathrm{a}}_j\}_{j=\mathcal{J}(T)+1}^{N_y})]$.

Finally, the three groups of variables can be used to calculate the probability that the encoder traversed a specific transition T in the trellis. Notation δ is defined to represent such a probability. For calculating the extrinsic information, a group of δ_x is considered here. For a particular T, $\delta_x(T)$ represents the probability $\ln[P(t_{\mathcal{J}(T)} = T|\{\tilde{x}^{\mathrm{a}}_j\}_{j-1, j\neq\mathcal{J}}^{N_x}, \{\tilde{y}^{\mathrm{a}}_j\}_{j-1}^{N_y})]$. It is the joint probability of the corresponding γ_y, α and β of the transition T.

For computing all these internal variables, the Log-BCJR algorithm is composed of the following four parts.

1. γ calculation: The values of γ depend on the inputs of the Log-BCJR decoder. There are two inputs, the uncoded LLRs input $\tilde{\mathbf{x}}$ and the encoded LLRs input $\tilde{\mathbf{y}}$. As shown in Figure 2.8, the encoded LLR input is the LLRs of the encoded sequence received from the channel $\tilde{\mathbf{y}}_j^{\mathrm{a}}$. The uncoded LLR input is $\tilde{\mathbf{x}}_j^{\mathrm{a}}$. For a transition T, the γ_x and γ_y can be calculated as:

$$\gamma_x(T) = (1 - \mathcal{B}_x(T))\tilde{x}^{\mathrm{a}}_{\mathcal{J}(T)}, \tag{2.13}$$

$$\gamma_y(T) = (1 - \mathcal{B}_y(T))\tilde{y}^{\mathrm{a}}_{\mathcal{J}(T)}. \tag{2.14}$$

2. α calculation: The values of α depend on the γ values and α values from the previous step in the trellis. Hence, it requires a forward recursion in the trellis to obtain all the α values. For a state S, in step j, the function to calculate α is:

$$\alpha(S) = \max_{T \in to(S)}{}^{*}(\gamma_x(T) + \gamma_y(T) + \alpha(\mathrm{fr}(T))). \tag{2.15}$$

   where $\alpha(S_1) = 0$, in the case of Figure 2.13.

3. β calculation: The values of β depend on the γ values and β values from the next step in the trellis. Hence, it requires a backward recursion in the trellis to obtain all the β values. For a state S, in step j, the function to calculate β is:

$$\beta(S) = \max_{T \in fr(S)}{}^{*}(\gamma_x(T) + \gamma_y(T) + \beta(\mathrm{to}(T))). \tag{2.16}$$

   where $\beta(S_{38}) = 0$, in the case of Figure 2.13.

4. δ_y calculation: The values of δ_x can be calculated according to (2.17).

$$\delta_x(T) = \gamma_y(T) + \alpha(\mathrm{fr}(T)) + \beta(\mathrm{to}(T)) \tag{2.17}$$

5. Finally, the extrinsic information can be calculated based on δ values. The extrinsic LLRs of the uncoded bits $\tilde{\mathbf{x}}^{\mathrm{e}}$ are:

$$\tilde{x}^{\mathrm{e}}_j = \max_{T \left| \substack{\mathcal{B}_x(T)=0 \\ \mathcal{J}(T)=j} \right.}{}^{*} \big(\delta_x(T)\big) - \max_{T \left| \substack{\mathcal{B}_x(T)=1 \\ \mathcal{J}(T)=j} \right.}{}^{*} \big(\delta_x(T)\big). \tag{2.18}$$

The algorithm is accomplished.

2.3 Extrinsic information transfer chart

As mentioned in Section 2.1, the BER chart is a powerful tool for analysing the performance of turbo-like codes. However, it is unable to characterise the convergence behaviour of a turbo-like code, for example at the onset of the turbo cliff. This requires a different analysis tool, namely the EXIT chart [100]. An EXIT chart uses Mutual Information (MI) measurement to quantity the quality of the extrinsic information exchanged between the constituent decoders in an iterative decoding system. The MI $I(\tilde{\mathrm{b}}, \mathrm{b})$ is a scalar in the range of $[0, 1]$, where zero is low quality and one is high quality information. There are a number of different methods of measuring MI. The averaging method [119] uses the equation:

$$I(\tilde{\mathrm{b}}, \mathrm{b}) = 1 + \frac{1}{N_b} \sum_{j-1}^{N_b} \sum_{b'=0}^{1} \frac{(1-b')e^{\tilde{b}_j}}{1+e^{\tilde{b}_j}} \log_2 \left[\frac{(1-b')e^{\tilde{b}_j}}{1+e^{\tilde{b}_j}} \right] \tag{2.19}$$

This method has the advantage of not requiring any knowledge of the bit sequence $\mathbf{b}$. This is achieved by assuming that the LLRs in $\tilde{\mathbf{b}}$ satisfy the consistency condition [100], that is the LLRs do not express too much confidence or too little confidence. Since the averaging method 'believe' what the LLRs say, it does not need to consider the true values of the bits in $\mathbf{b}$. However, this assumption is only valid if there are no sub-optimalities in the receiver design. This requires perfect channel estimation, perfect carrier recovery, perfect synchronisation, perfect equalisation and optimal decoding using the Log-BCJR algorithm. By contrast, The histogram method [119] of measuring MI does not make the described assumption and is therefore better suited when a sub-optimal receiver is employed. This method uses knowledge of the true values of the bits in $\mathbf{b}$ to avoid having to 'believe' what the LLRs say.

The EXIT chart is comprised of two curves for the two decoders in the system. Each curve plots the MI of the extrinsic LLRs versus the MI of the a priori LLRs of one decoder in the system. This measures the quality of the output as a function of the quality of the input. For example, taking the UMTS/LTE decoding scheme as an example, for the first decoder, the EXIT curve plots $I(\tilde{\mathbf{b}}_1^{\mathrm{e}}, \mathbf{b}_1)$ as a function of $I(\tilde{\mathbf{b}}_1^{a'}, \mathbf{b}_1)$ as shown in Figure 2.14, where $I(\tilde{\mathbf{b}}_1^{\mathrm{a}}, \mathbf{b}_1)$ is the MI between $\tilde{\mathbf{b}}_1^{\mathrm{a}'}$ and $\mathbf{b}_1$, while $I(\tilde{\mathbf{b}}_1^{\mathrm{e}}, \mathbf{b}_1)$ is the MI between $\tilde{\mathbf{b}}_1^{\mathrm{e}}$ and $\mathbf{b}_1$. For plotting the EXIT curve, a simulator is employed to generate sequences of a priori LLRs $\tilde{\mathbf{b}}_1^{\mathrm{a}'}$ having a range of MI $(0 < I(\tilde{\mathbf{b}}_1^{\mathrm{a}'}, \mathbf{b}_1) < 1)$. Using simulations that include the channel model, the modulation model and the BCJR decoder, the extrinsic output $\tilde{\mathbf{b}}_1^{\mathrm{e}}$ can be obtained and measured. The MI is averaged over many frames. If $I(\tilde{\mathbf{b}}_1^{\mathrm{e}})$ is used to represent $I(\tilde{\mathbf{b}}_1^{\mathrm{e}}, \mathbf{b}_1)$ and $I(\tilde{\mathbf{b}}_1^{\mathrm{a}'})$ to represent $I(\tilde{\mathbf{b}}_1^{\mathrm{a}'}, \mathbf{b}_1)$, the EXIT function $I(\tilde{\mathbf{b}}_1^{\mathrm{e}}) = F(I(\tilde{\mathbf{b}}_1^{\mathrm{a}'}))$ of the UMTS/LTE turbo code is shown in Figure 2.15. In the simulation, the exact convolutional code shown in Figure 2.14 is chosen, with BPSK modulation and an AWGN channel. The Signal-to-Noise Ratio (SNR) is -4dB. The SNR is defined as:

$$SNR = \frac{E_s}{N_0}, \tag{2.20}$$

where E_s is the energy per symbol and N_0 is the noise power spectral density.

Similarly, the EXIT curve for the other decoder can be plotted. For a turbo code, owing to the symmetry of the two concatenated codes, the EXIT function of the lower decoder

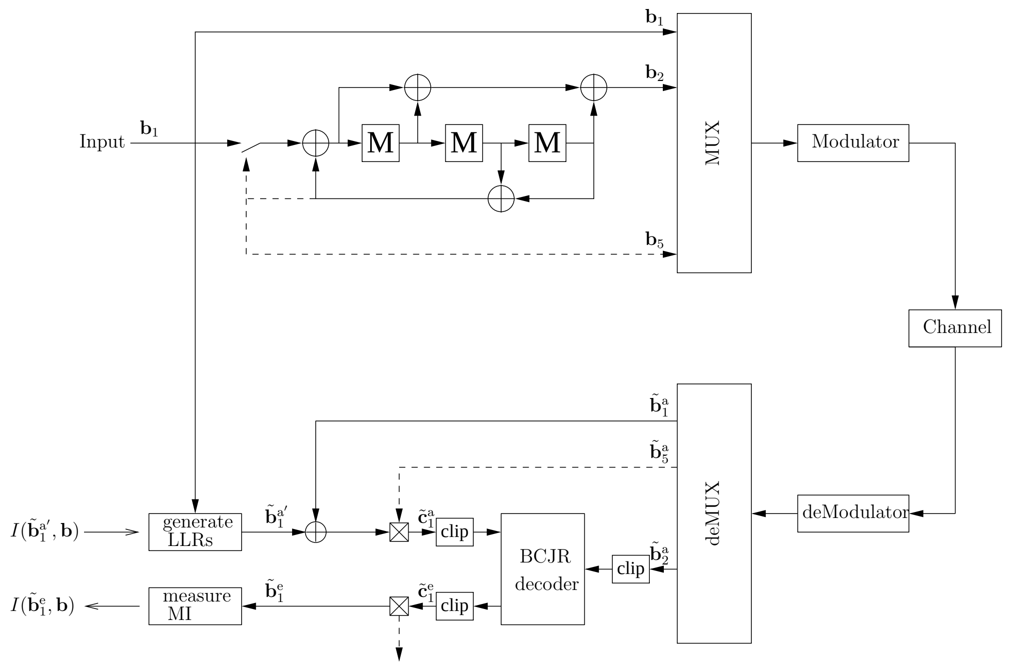

Input
$\mathbf{b}_1$
M
M
M
$\mathbf{b}_1$
$\mathbf{b}_2$
$\mathbf{b}_5$
MUX
Modulator
Channel
deModulator
deMUX
$\tilde{\mathbf{b}}_1^{\mathrm{a}}$
$\tilde{\mathbf{b}}_5^{\mathrm{a}}$
$\tilde{\mathbf{b}}_2^{\mathrm{a}}$
$I(\tilde{\mathbf{b}}_1^{\mathrm{a}'}, \mathbf{b})$
generate LLRs
$\tilde{\mathbf{b}}_1^{\mathrm{a}'}$
$\tilde{\mathbf{c}}_1^{\mathrm{a}}$
clip
BCJR decoder
clip
$I(\tilde{\mathbf{b}}_1^{\mathrm{e}}, \mathbf{b})$
measure MI
$\tilde{\mathbf{b}}_1^{\mathrm{e}}$
$\tilde{\mathbf{c}}_1^{\mathrm{e}}$
clip

FIGURE 2.14: Schematic of an EXIT chart simulation.

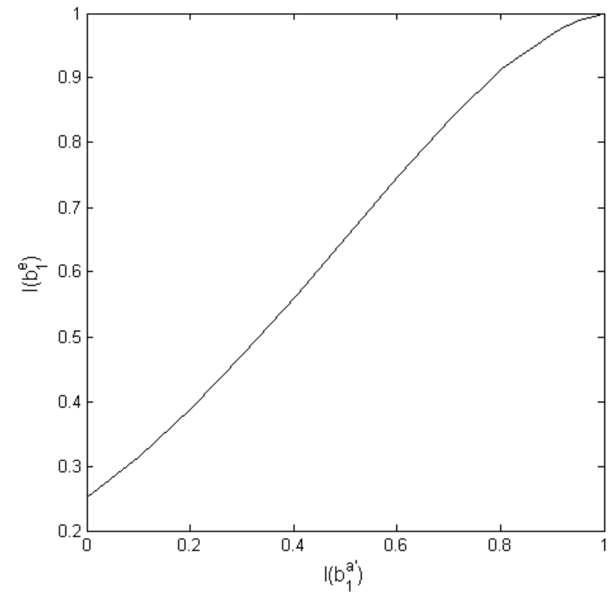


FIGURE 2.15: One EXIT curve $I(\tilde{\mathbf{b}}_1^{\mathrm{e}}) = F(I(\tilde{\mathbf{b}}_1^{\mathrm{a}'}))$ of UMTS/LTE turbo code using BPSK to transmit over an AWGN channel having an SNR of -4 dB.

is identical to that of the upper decoder. In an EXIT chart, the second curve is plotted with the inverted axes, that is the horizontal axis plots the MI of the extrinsic output and the vertical axis plots the MI of the a priori input. The reason for displaying the second curve with the inverted axes is because in the iterative decoding process, the output of one decoder becomes the input of the other decoder in the next iteration. By putting the input of the decoder and the output of the other decoder in the same axis, the interaction of the two concatenated decoders can be predicted on an EXIT chart. The complete EXIT chart of the UMTS/LTE turbo code generated is given in Figure 2.16.

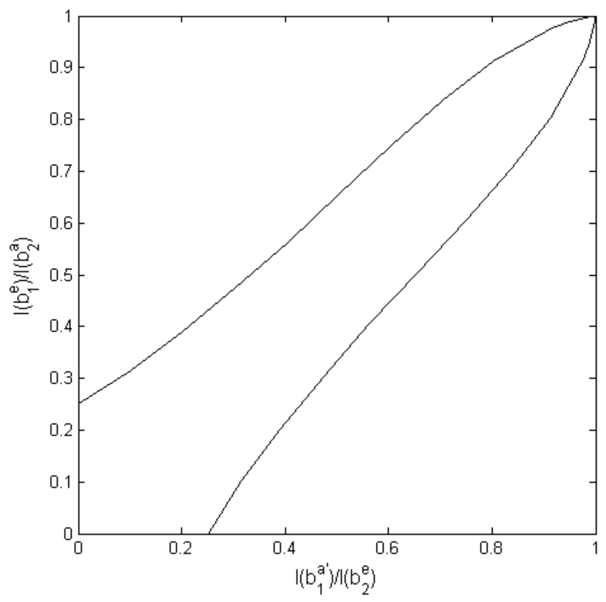


Figure 2.16: EXIT chart of the UMTS turbo decoder.

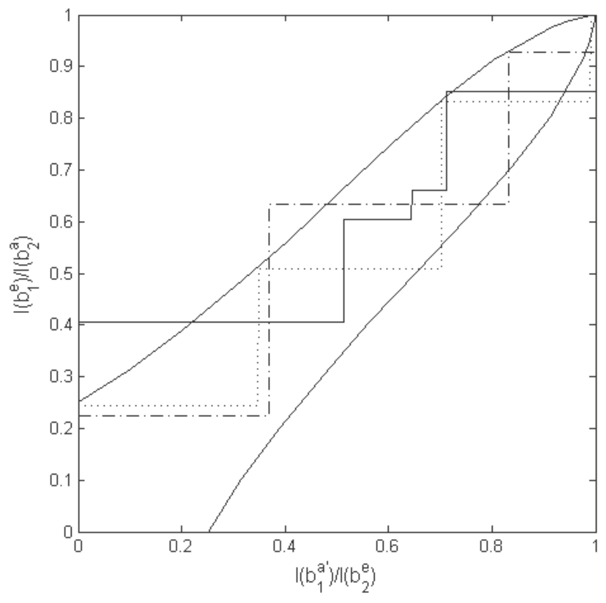


Figure 2.17: Example decoding trajectories in the EXIT chart.

The iterative decoding process of the turbo code can be revealed by plotting decoding trajectories in the EXIT chart, as shown in Figure 2.17. A decoding trajectory begins at (0,0) point in the bottom left corner of the EXIT chart, since no a priori information is provided by the other decoder at the start of the decoding process. The MI of the output of the first decoder can be obtained by the upper curve in the EXIT chart and is provided as the input of the second decoder. Based on the MI provided by the first decoder, the MI of the output of the second decoder can be obtained by the lower curve in the EXIT chart. The decoding performance of the next iteration can be obtained by the same way. Thus, a decoding a decoding trajectory can be obtained as a staircase that propagates upwards to the upper curve to rightwards towards to the lower curve. Note that the condition for the decoding trajectory to have a high probability of reaching the (1,1) point is that the EXIT chart has open tunnel, that is the two curves do not cross each other before they reach the (1,1) point. By reaching the (1,1) point, the BER will be in the error floor region. However, since the EXIT chart represents the statistical analysis of a large number of simulated samples, the trajectories will typically vary from the EXIT chart in practice. Different input sequences have different traces of trajectories. As demonstrated in Figure 2.17, the three trajectories are all different and depart from the EXIT chart. The EXIT chart gives the average convergence behaviour of the investigated code. An EXIT chart allows the two concatenated codes to be considered in isolation of each other. Since EXIT charts can predict the iterative interaction of the two codes, the iterative decoding process does not need to be simulated in order to draw an EXIT chart. Thus, EXIT charts can be obtained faster than BER/Frame Error Rate (FER) charts.

2.4 Fixed-point numerical representation

The fixed-point representation of numerical values is a widely used technique in ASIC digital signal processing circuits. For the hardware implementation of the turbo decoders, it is a very important technique to keep the hardware complexity low. This section provides an introduction to fixed-point representation in hardware design. Fixed-point representation, compared with floating-point representation, is easily implemented in a small memory space and it is fast to execute arithmetic calculations. Therefore, it is well-suited to real-time or low-power applications. Internally, the computation of fixed-point numbers take the values as integers, but considers the integer part and fraction part separately with an imaginary point.

Two's complement representation is the most widely used fixed-point representation in practice. A two's complement binary number is divided into three parts, a sign bit, an integer part and a fraction part. First, consider the two's complement representation of signed integers before considering the representation of numbers having a fraction part. The most significant bit is used as the sign bit, where 0 is used to represent positive signs and 1 is to represent negative signs. The rest of the bits represent the magnitude of the number. A negative number is represented by its absolute value complemented bit by bit and incremented by 1. For example the 3-bit representation of 2 is 010. The complement of this is 101. Adding 1 to this gives the two's complement representation of -2, namely 110. The complete set of 3-bit two's complement representations is given in Table 2.1. In addition, another two signed integer representation methods, sign and absolute value notation and one's complement notation, are also given as examples in the table for comparison. As shown, compared with the other two methods, the two's

Binary number	000	001	010	011	100	101	110	111
Sign and absolute value	+0	+1	+2	+3	-0	-1	-2	-3
One's complement	+0	+1	+2	+3	-3	-2	-1	-0
Two's complement	+0	+1	+2	+3	-4	-3	-2	-1

TABLE 2.1: Different representation methods for integer numbers

complement representation avoids the double representation of zero. As a consequence, the range of negative values is greater than the range of positive values by one. The main advantage of two's complement notation is the ability to perform the addition of negative numbers, without needing to take the sign of the operands into consideration. Furthermore, in two's complement notation, subtraction is achieved by performing the complement and adding. For example, in 3-bit representation, $2 - 3$ can be performed by calculating the sum of 2 (010) and -3 (101).

$$2 - 3 = 2 + (-3) = 010 + 101 = 111 = -1 \tag{2.21}$$

For the subtractions in two's complement notation, letting the result overflow is necessary. Take the following calculation as an example:

$$2 - 2 = 2 + (-2) = 010 + 110 = (1)000 = 000 = 0 \tag{2.22}$$

When the overflowed part is removed, the calculation gives the correct result naturally. By contrast, subtraction in the other two notation methods is more complicated since complement and adding does not give the correct result. For example, in one's complement notation:

$$3 - 2 = 3 + (-2) = 011 + 101 = 000 = 0 \tag{2.23}$$

Therefore, the addition of different signed components need to be considered carefully and extra correction is required.

For a fractional fixed-point number, A , an imaginary point is set at a certain place. A 3-bit two's complement fixed-point number with a 2-bit fraction part is exemplified in Table 2.2. In the table, the imaginary point is placed after the most significant bit in the binary representation. For n-bit two's complement representation, notation Q$y.z$

Binary number	0.00	0.01	0.10	0.11	1.00	1.01	1.10	1.11
Two's complement	+0.00	+0.25	+0.50	+0.75	-1	-0.75	-0.5	-0.25

TABLE 2.2: Two's complement representation method for fraction numbers

is defined to represent the point setting, where y represents the number of bits in the integer component and z represents the number of bits in the fraction component. The total number of bits is given by $n = y + z + 1$. For example, an 8-bit two's complement number with a imaginary point after the 5th bit is 01100.010. The integer part is 12 and the fraction part is 0.25, thus the decimal value of 01100.010 is 12.25. The representation range of the representation are given by (2.24), and the resolution r is given by (2.25).

$$-\frac{2^{p+q}}{2^q} \leq A \leq \frac{2^{p+q}-1}{2^q} \tag{2.24}$$

$$r = 2^{-q} \tag{2.25}$$

Owing to the limited representation range and the limited resolution of fixed-point representation, errors occur when the values are outside of the range or the resolution, causing overflow and underflow, respectively. For example, for a Q3.2 fixed-point representation, the representation range is $-8 \leq A \leq 7.75$. The resolution is 0.25. To calculate $6 + 7$, the binary process is $0110 + 0111 = 1101$. The result in binary 1101 is -3 in decimal, which is incorrect because of the overflow. For a value that is out of the resolution limit, take 1.23 as an example, in the fixed-point representation, it has to be approximated to the nearest quantised value, which is 1.25. The 0.02 inaccuracy is caused by the underflow.

2.5 Chapter Summary

This chapter commenced with the introduction of the widely used type of near Shannon limit ECCs, namely turbo-like codes. Secondly, a powerful analysis tool for investigating the designs, the performance and the convergence behaviours of turbo-like codes, namely EXIT charts, is presented. Finally, an important issue that directly related to the hardware complexity and energy consumption of the turbo-like codes' implementations, the fixed-point numerical representation, is discussed. The key points of this background chapter are as follows.

- Section 2.1 introduced typical configurations and the BER performance of the two types of turbo-like codes, namely SCCCs and PCCCs. By employing SISO component decoders and an iterative decoding process to exchange extrinsic information between the component decoders, turbo-like codes are capable of achieving a near Shannon limit performance.

- In Section 2.2, using the UMTS/LTE turbo code [103] as an example, the typical turbo encoder and decoder schemes are presented. Furthermore, a widely used decoding algorithm of turbo codes, namely the LUT-Log-BCJR algorithm, is described in detail.

- In Section 2.3, the EXIT chart [100] is introduced for visualising the EXIT function of a LUT-Log-BCJR decoder. The demonstration illustrates how that EXIT charts are capable of characterising the convergence behaviour of turbo-like codes. The typical simulation scheme for generating EXIT charts and decoding trajectory chart is presented.

- In Section 2.4, the fixed-point numerical representation is introduced. The fixed-point representation has a lower complexity when compared to the floating-point representation, facilitating practical ASIC implementations of LUT-Log-BCJR decoders. This section presented three fixed-point representation methods, namely sign and absolute value, one's complement and two's complement. The advantage of using the two'c complement fixed-point representation instead of the other two representation methods is discussed.

Chapter 3

A serially concatenated convolutional code scheme for star topology wireless sensor networks[1]

3.1 Introduction

As discussed in Chapter 1, turbo-like codes have the capability to reduce the transmission energy consumption of the wireless communication systems in Wireless Sensor Networks (WSNs), thanks to their near Shannon limit performance. However, the additional energy consumption introduced by the encoder and decoder precessing must be considered, when determing the *overall* energy consumption. In this chapter, a simple but widely used scenario, namely the star network, is investigated on this subject. A Serial Concatenated Convolutional Code (SCCC) scheme based on an augmentation of the PHYsical layer (PHY) of the Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 standard is used for the investigation.

The sensor nodes of a WSN are typically required to maintain sporadic but reliable data transmissions for extended periods of time. Furthermore, in some applications, the sensor nodes are required to be small, preventing the use of bulky batteries [3]. Therefore, improvements in the energy efficiency of the sensor nodes are also required in order for WSNs to find further diverse applications. A star WSN topology is often employed in low power and low rate applications, with all data frames being transmitted to a central node, which coordinates the reactions of a higher-level system to the sensed

[1]This work is collaborated with Dr. Rob Maunder. The presented SCCC scheme is proposed in Dr. Rob Maunder's previous work [69].

data. Many previous efforts have focused on the energy consumption of star topology WSNs [81, 120, 121]. In these scenarios, the central node typically has abundant energy resources, owing to its integration into the higher-level system. It is therefore beneficial to redistribute energy consumption from the sensor nodes to the central node, whenever possible.

In [69], an augmentation to the PHY of the IEEE 802.15.4 standard for WSNs [102] was introduced in order to redistribute energy consumption in the manner described above. This was achieved by deploying a sophisticated decoder of the employed Error-Correcting Code (ECC) within the central node. This reduced the transmission energy required to achieve a desirable Frame Error Rate (FER) of 10^{-3} by 3.22 - 6.75 dB, depending on the length of the PHY payload [69]. Although, additional energy is consumed by the additional error correction encoding operations that are performed in the transmitting sensor nodes, the analysis in [69] demonstrated that the proposed approach can reduce the *overall* energy consumption by more than 17.4 - 23.3%. However, this result was based on the assumption that the additional error correction encoding operations are performed by software running on the 8051 microprocessor of a Chipcon CC2430 [122] sensor node. This is a conservative assumption since the required functionality is significantly simpler than the capability of an 8051 microcontroller. A sensor node energy consumption reduction of more than 20% could therefore be expected if error correction encoding was performed in hardware. This chapter therefore introduces and characterises a dedicated Application-Specific Integrated Circuit (ASIC) design that was implemented for this purpose.

The rest of this chapter is organised as follows. Section 3.2 reviews the augmented IEEE 802.15.4 PHY and the analysis of its energy savings that was detailed in [69]. In Section 3.3, the novel deterministic interleaver design that was employed to obtain the results of [69] is detailed for the first time. This interleaver design is suitable for all possible PHY payload lengths without imposing an excessive memory requirement. A novel Genetic Algorithm (GA) [123] is employed for parameterising the interleaver in order to maximise its performance, as detailed for the first time in Section 3.4. Section 3.5 discusses a novel hardware implementation of the ECC encoder, which employs parallel 'just-in-time' processing in order to achieve a low processing latency and energy consumption. This energy consumption is analysed in Section 3.6 and the analysis shows that it is insignificant compared to the transmission energy saving that it affords. As a result, the augmented PHY is shown to offer net energy consumption savings of 24.8 – 31.4%, which are significantly greater that those reported in [69] of 17.4 – 23.3%. Finally, conclusions are offered in Section 3.7.

3.2 Review of the augmented PHYsical layer

As detailed in [69], the proposed augmentation to the IEEE 802.15.4 PHY can be employed to convey the payloads of IEEE 802.15.4 data frames using a reduced transmission energy. However, the augmented PHY imposes additional interleaving and rate-1 encoding operations upon the transmitting sensor nodes. As shown in the schematic of Figure 3.1, these operations are performed between the Pseudo Noise (PN) spreading and Offset Quadrature Phase-Shift Keying (O-QPSK) operations of the standard IEEE 802.15.4 PHY [102].

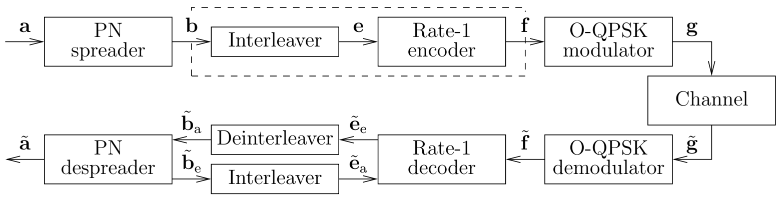


FIGURE 3.1: Schematic of the augmented IEEE 802.15.4 PHY.

An interleaver and a rate-1 encoder is proposed to be concatenated to the PN spreader from the original IEEE 802.15.4 PHY, as shown in the dashed line box in Figure 3.1. The augmented PHY applies PN spreading to the M-byte PHY payload **a**, where $M \in [10 \ldots 127]$ [102]. This is achieved by decomposing the payload **a** into sets of $k = 4$ consecutive bits and mapping these to $n = 32$-bit PN sequences [102], as in the standard PHY. These PN sequences are concatenated to obtain the N-bit sequence **b**, where $N = \frac{8Mn}{k}$. The interleaver of the augmented PHY then employs a three-step so-called Dithered Relative Prime (DRP) process [124] to rearrange the order of the bits in **b**. This process is detailed for the first time in Sections 3.3 and 3.4 of this chapter. As shown in Figure 3.1, the resultant N-bit sequence **e** is rate-1 encoded [125], as detailed in Section 3.3. Finally, the encoded bit sequence **f** is O-QPSK modulated, as in the standard PHY. As detailed in Section 3.3, the input of the augmented PHY's O-QPSK modulator **f** comprises the same number N of bits as the output of the PN spreader **b**, like in the standard PHY. For this reason, the PN spreader and O-QPSK modulator remain completely unchanged, when the augmented PHY is employed in the transmitting sensor nodes.

In the receiving central node, additional rate-1 decoding and deinterleaving operations are employed by the augmented PHY. This employs iterative decoding [126], which repeatedly alternates the operation of the PN despreader and the rate-1 decoder, as shown in Figure 3.1. This is in contrast to the receiver of the standard PHY, which employs only the 'one-shot' operation of the PN despreader. Since the augmented PHY invests more decoding complexity in the central node than the standard PHY, it can achieve a desirable FER using a reduced sensor node transmission energy. This is demonstrated by the simulation results of Figure 3.2, which considers transmission over a Line-Of-Sight (LOS) channel in the presence of Additive White Gaussian Noise (AWGN) having a constant power spectral density N_0, in common with [102]. These results show that when transmitting $N = 640$-bit payloads, the augmented PHY can achieve a desirable FER of 10^{-3} at a transmission energy per bit E_c that is 3.22 dB lower than that required by the standard PHY. Furthermore, this gain increases to 6.75 dB, when $N = 8128$-bit payloads are employed, owing to the augmented PHY's interleaver gain [69], which is obtained when transmitting longer payloads.

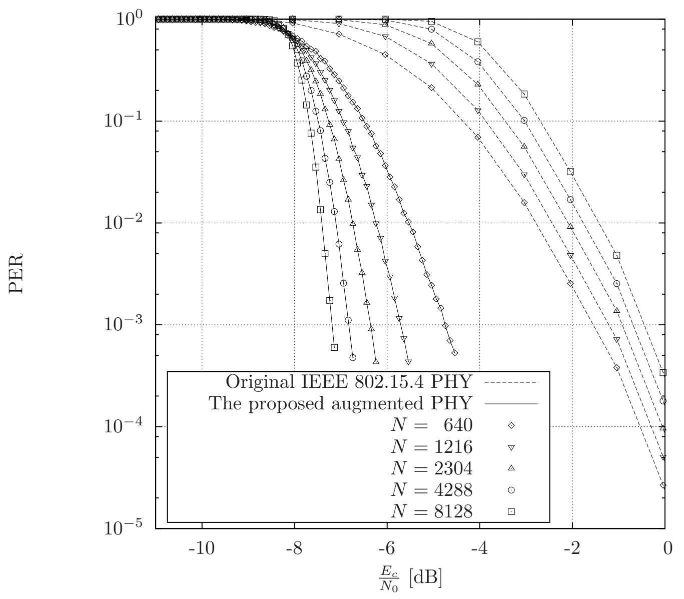


Figure 3.2: FER performance of the standard and augmented PHYs for payloads comprising various numbers of bits N, when communicating over LOS AWGN channels, having a range of values for the SNR per payload bit E_c/N_0.

In order to assess the practical sensor node energy saving, the augmentation of the Chipcon CC2430 PHY [122] was considered in [69]. The energy consumed during transmission is given by $E^{\text{tx}} = I^{\text{tx}} \cdot V \cdot t^{\text{tx}}$, where $t^{\text{tx}} = N/f^{\text{tx}}$ is the transmit duration and the IEEE 802.15.4 transmission rate is $f^{\text{tx}} = 2 \cdot 10^6$ bits per second [102]. As may be expected, the current I^{tx} consumed during the transmission of a data payload depends on the particular transmit energy per bit E_c employed. In its maximum transmit power mode of 0.6 dBm, the Chipcon CC2430 consumes $I^{\text{tx}}_{\text{std}} = 32.4$ mA [122]. At this transmit power, the amount of energy $E^{\text{tx}}_{\text{std}} = I^{\text{tx}}_{\text{std}} \cdot V \cdot t^{\text{tx}}$ consumed by the standard PHY without augmentation is illustrated in Figure 3.3 for payloads comprising various numbers N of bits. As described above, the augmented PHY reduces the transmission energy required to achieve a desirable FER of 10^{-3} by 3.22 – 6.75 dB, depending on the length of the payload N. Corresponding reductions from the Chipcon CC2430's maximum transmit power of 0.6 dBm allow its current consumption $I^{\text{tx}}_{\text{aug}}$ to be lowered from 32.4 mA to 21.7 – 23.7 mA [122]. Figure 3.3 shows the amount of transmission energy $E^{\text{tx}}_{\text{aug}} = I^{\text{tx}}_{\text{aug}} \cdot V \cdot t^{\text{tx}}$ consumed by the augmented PHY for various values of N. These results show that the augmented PHY facilitates *gross* sensor node energy savings of $(E^{\text{tx}}_{\text{std}} - E^{\text{tx}}_{\text{aug}})$ that are 27.0 – 33.0% of $E^{\text{tx}}_{\text{std}}$, depending on the payload length N.

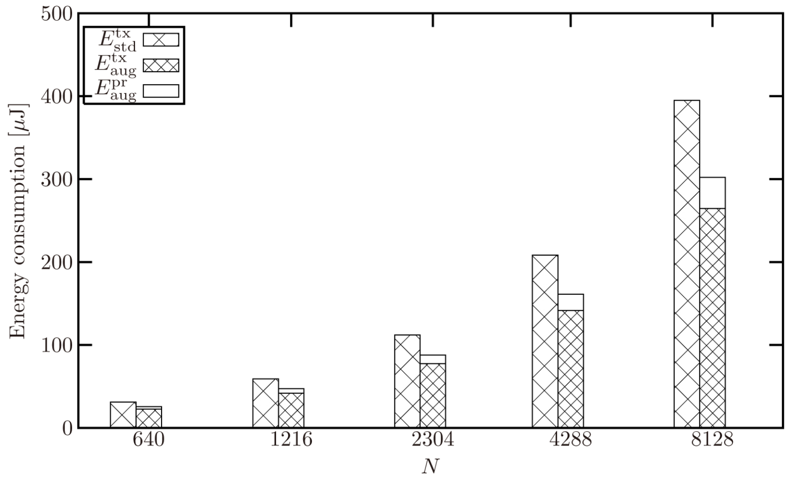


FIGURE 3.3: Total energy consumed in the standard Chipcon CC2430 PHY $E^{\text{tx}}_{\text{std}}$ and in the software implementation of the augmented PHY ($E^{\text{tx}}_{\text{aug}} + E^{\text{aug}}_{\text{pr}}$).

However, in order to determine the *net* sensor node energy saving $[E^{\text{tx}}_{\text{std}} - (E^{\text{tx}}_{\text{aug}} + E^{\text{pr}}_{\text{aug}})]$ that is afforded by the augmented PHY, it is necessary to additionally consider the energy $E^{\text{pr}}_{\text{aug}}$ consumed during the operation of the interleaver and rate-1 encoder that are boxed in Figure 3.1. In [69], it was assumed that these operations were performed by software running on the 8051 processor of a Chipcon CC2430 sensor node. Since the interleaver of Figure 3.1 employs a three-stage process and rate-1 encoding can be completed in a single step, it was assumed that each bit in the N-bit payload $\mathbf{b}$ can be processed using 4 clock cycles, requiring $C^{\text{pr}}_{\text{aug}} = 4N$ clock cycles in total. The duration of this processing is therefore $t^{\text{pr}}_{\text{aug}} = C^{\text{pr}}_{\text{aug}}/f^{\text{pr}}_{\text{aug}}$, where $f^{\text{pr}}_{\text{aug}}$ is the system's clock frequency. Here, $f^{\text{pr}}_{\text{aug}}$ is assumed to be 32 MHz, which is the clock frequency on the Chipcon CC2430 [122]. The energy consumed by interleaving and rate-1 encoding is given by $E^{\text{pr}}_{\text{aug}} = I^{\text{pr}}_{\text{aug}} \cdot V \cdot t^{\text{pr1}}_{\text{aug}}$. Here, a supply voltage $V = 3$ Volts equal to that of the Chipcon CC2430 and a current consumption of $I^{\text{aug}}_{\text{pr1}} = 12.3$ mA equal to the peak consumption of the on-board 8051 processor [122] were conservatively assumed. The resultant processing energy consumptions $E^{\text{pr}}_{\text{aug}}$ are shown in Figure 3.3 for a variety of payload lengths N. Using this software implementation of the augmented PHY, net energy savings $[E^{\text{tx}}_{\text{std}} - (E^{\text{tx}}_{\text{aug}} + E^{\text{pr}}_{\text{aug}})]$ that correspond to 17.4 – 23.3% of $E^{\text{tx}}_{\text{std}}$ are afforded, as reported in [69].

As described in Section 3.1 however, a more efficient implementation of the augmented PHY in a transmitting sensor node would resemble the Chipcon CC2430, but with the addition of a ASIC module dedicated to performing interleaving and rate-1 encoding. Since a dedicated module would be much simpler than an 8051 processor and because it could benefit from parallel processing, this approach would offer a reduced processing energy consumption $E^{\text{pr}}_{\text{aug2}}$ and therefore an increased net energy saving. The following sections detail the design, parametrisation, implementation and characterisation of a hardware module for this purpose.

3.3 Module design

As described in Section 3.2, the proposed module implements the interleaver and rate-1 encoder that are boxed in Figure 3.1. Here, the standard IEEE 802.15.4 PN spreader [102] provides the input bit sequence $\mathbf{b}$. As described in Section 3.2, this comprises $8M/k$ number of $n = 32$-bit PN sequences [102], where M is the number of bytes in the data payload $\mathbf{a}$. Since this has 118 possible values $M \in [10 \dots 127]$, there are 118 possible lengths $N = \frac{8Mn}{k} \in \{640, 704, 768, \dots, 8128\}$ for the bit sequence $\mathbf{b}$ [69].

When repositioning the bits in the sequence $\mathbf{b} = \{b_i\}_{i=0}^{N-1}$, the interleaver of Figure 3.1 is required to desirably 'randomise' the order of the bits in the resultant sequence $\mathbf{e} = \{e_i\}_{i=0}^{N-1}$. In order to achieve this, the interleaver must fully exploit the grade of freedom for re-positioning the bits, which increases with the number of bits N. As a result, different interleaver designs are required for each of the 118 possible values of N and the associated parameters of the interleaver design must be stored in Read-Only Memory (ROM).

A naive interleaver design would be parameterised by 118 arrays $\{\pi_{640}$, π_{704}, π_{768}, $\dots$, $\pi_{8128}\}$, each of which would comprise N unique integers in the range of $[0, N-1]$. The operation of the naive interleaver can be formally specified as $e_i = b_{\pi_N[i]}$, where $\pi_N[i]$ is the i^{th} element in the array π_N and $i \in [0, N-1]$. However, this approach would require approximately 800 KB of ROM, which is considered to be excessive, since memory accesses are typically associated with relatively high energy consumptions in systems [127].

This problem was solved in the implementations detailed in [128–130] by employing deterministic interleaver designs. These require the storage of only a limited number of parameters, which are employed to compute the elements of the interleaver pattern in an on-line manner, as and when they are required. However, the designs detailed in [128–130] are optimised for turbo codes and are not suitable for the augmented PHY. This is because the interleaver is required to mitigate a higher level of correlation within the soft information exchanged in the augmented PHY, since the PN despreader of Figure 3.1 operates on relatively long blocks, comprising $n = 32$ bits.

For this reason, a deterministic design resembling a DRP interleaver [124] is employed, which has been shown to effectively 'randomise' the order of the bits in the sequence $\mathbf{b}$ without requiring an excessive amount of ROM. Indeed, only 12 KB of ROM are required to store the parameters of the interleaver design, as listed in Table 3.1. Like a DRP interleaver, the proposed design is implemented in three stages, which are referred to as 'Interleaver 1', 'Interleaver 2' and 'Interleaver 3' in the discussions below. As exemplified by the criss-crossing arrows in Figure 3.4, these interleavers are employed to 'randomise' the order of the intermediate bit sequences $\mathbf{c}$, $\mathbf{d}$ and $\mathbf{e}$, respectively.

Note that the intermediate bit sequences comprise the same number of bits as the input sequence $\mathbf{b}$, namely N.

TABLE 3.1: Parameters of the interleavers shown in Figure 3.4.

$\{\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{253}\}$	Each of the 254 arrays $\mathbf{r}_u$ is comprised of $n = 32$ unique integers from the range $[0, n-1]$.
$\{s_{640}, s_{704}, s_{768}, \ldots, s_{8128}\}$	Each of the 118 integers s_N has a value from the range $[0, n-1]$.
$\{p_{640}, p_{704}, p_{768}, \ldots, p_{8128}\}$	Each of the 118 integers p_N has a value from the range $[1, N-1]$.
$\{W_{640}, W_{704}, W_{768}, \ldots, W_{8128}\}$	Each of the 118 integers W_N has a value from the range $[1, N]$.
$\{\mathbf{w}_{640}, \mathbf{w}_{704}, \mathbf{w}_{768}, \ldots, \mathbf{w}_{8128}\}$	Each of the 118 arrays $\mathbf{w}_N$ is comprised of W_N unique integers from the range $[0, W_N-1]$.

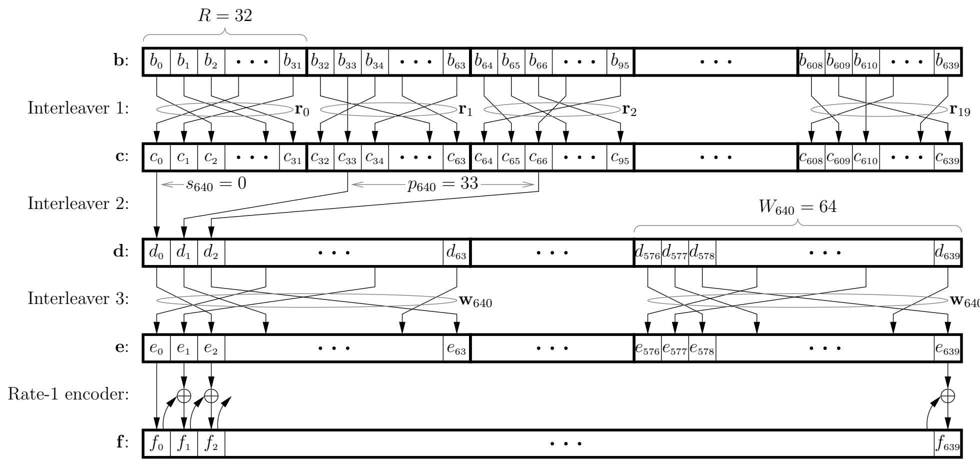


FIGURE 3.4: Example operation of the interleaver and rate-1 encoder of Figure 3.1 for $N = 640$.

Interleaver 1 Similarly to the first stage of a DRP interleaver, Interleaver 1 of Figure 3.4 employs a block-based rearrangement of the bits in the input sequence $\mathbf{b} = \{b_i\}_{i=0}^{N-1}$ in order to generate the sequence $\mathbf{c} = \{c_i\}_{i=0}^{N-1}$. More specifically, each block of $n = 32$ bits in $\mathbf{c}$ is provided by rearranging the order of the corresponding $n = 32$-bit PN sequence in $\mathbf{b}$. As shown in Figure 3.4, a different rearrangement is employed for each $n = 32$-bit PN sequence, as specified by the parameters $\{\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{253}\}$ of Table 3.1. Note that 254 rearrangements are required, because the sequence $\mathbf{b}$ comprises $N/n = 254$ PN sequences, when it has a maximal length of $N = 8128$ bits. The operation of Interleaver 1 can be formally specified as $c_i = b_{j_i}$, where

$$j_i = n \cdot u + \mathbf{r}_u[v], \tag{3.1}$$

$\mathbf{r}_u[v]$ is the v^{th} element in the array $\mathbf{r}_u$, $v = i \bmod n$, $u = i \operatorname{div} n$ and $i \in [0, N-1]$. Here, the 'div' operator indicates integer division, while 'mod' is employed to represent the modulo operator.

Interleaver 2 Similarly, the operation of Interleaver 2 from Figure 3.4 can be specified as $d_i = c_{j_i}$, where

$$j_i = (s_N + p_N \cdot i) \bmod N \tag{3.2}$$

and $i \in [0, N-1]$. As in the second stage of a DRP interleaver, s_N identifies the index of the bit in $\mathbf{c}$ that provides the first bit in $\mathbf{d} = \{d_i\}_{i=0}^{N-1}$, as shown in Figure 3.4. The subsequent bits in $\mathbf{d}$ are provided by employing successive hops of p_N bits (modulo N) to select the corresponding bit in the sequence $\mathbf{c}$. Here, p_N is required to be a relative prime of N in order to ensure that each bit in $\mathbf{c}$ provides exactly one bit for $\mathbf{d}$. Note that the particular values that are employed for s_N and p_N depend upon the length N of the bit sequence, as shown in Table 3.1.

Interleaver 3 Similarly, the parameters employed for Interleaver 3 of Figure 3.4 depend upon the length N of the bit sequence. This interleaver employs a block-based rearrangement of the bits in $\mathbf{d}$ in order to obtain the sequence $\mathbf{e}$, similarly to Interleaver 1. However in contrast to Interleaver 1, Interleaver 3 employs the same rearrangement for each block of W_N bits in the sequence $\mathbf{d}$, as shown in Figure 3.4. As seen in Table 3.1, this rearrangement and the block length are described by the parameters $\mathbf{w}_N$ and W_N, respectively. Clearly, W_N is required to be a factor of the bit sequence length N. For example, $W_{640} = 16$, $W_{1216} = 32$, $W_{2304} = 64$, $W_{4288} = 64$ and $W_{8128} = 64$. The operation of Interleaver 3 can be formally specified as $e_i = d_{j_i}$, where

$$j_i = W_N \cdot u + \mathbf{w}_N[v], \tag{3.3}$$

$\mathbf{w}_N[v]$ is the v^{th} element in the array $\mathbf{w}_N$, $v = i \bmod W_N$, $u = i \operatorname{div} W_N$ and $i \in [0, N-1]$.

Rate-1 encoder Finally, for each bit in the sequence $\mathbf{e} = \{e_i\}_{i=0}^{N-1}$, the rate-1 encoder of Figure 3.1 generates one bit for its output sequence $\mathbf{f} = \{f_i\}_{i=0}^{N-1}$. More specifically, $f_0 = e_0$ and $f_i = e_i \oplus f_{i-1}$ for $i \in [1, N-1]$, as shown in Figure 3.4, in which $\oplus$ indicates the modulo-2 addition of two binary bits. As a result, the output bit sequence $\mathbf{f}$ input to the standard IEEE 802.15.4 O-QPSK modulator [102] also comprises N bits.

3.4 Module parametrisation

The off-line algorithm employed to design values for the interleaver parameters of Table 3.1 is described in this section, in order to ensure that the order of the bits in the sequence $\mathbf{b}$ is effectively 'randomised'. Note that the N-bit input sequence $\mathbf{b}$ has $2^{N/8}$ legitimate permutations, since the PN spreader of Figure 3.1 has a coding rate of $k/n = 1/8$ [102]. As described above, the module detailed in this section maps each of these permutations to a different permutation of the output bit sequence $\mathbf{f}$. The particular mapping that is employed depends upon the parameters of the interleavers. The off-line algorithm used for designing these parameters attempts to maximise the minimum Hamming distance $D_{\mathrm{H}}^{\mathrm{min}}$ between the legitimate permutations of $\mathbf{f}$, as detailed below. In this way, the number of bit errors that is required to transform the transmitted permutation of $\mathbf{f}$ into any other legitimate permutation is maximised. This maximises the probability that transmission errors can be detected and corrected by the iterative decoder of Figure 3.1, optimising its performance.

Though it is beyond the scope of this chapter, it can be shown that $D_{\mathrm{H}}^{\mathrm{min}}$ can be as low as six if the interleaver does not effectively 'randomise' the order of its bits. However, it can be shown that $D_{\mathrm{H}}^{\mathrm{min}}$ will increase to at least 24, provided that the interleaver parametrisation satisfies two conditions. The first condition requires the interleaver to separate every pair of bits from each $n = 32$-bit PN sequence in $\mathbf{b}$ with at least two bits from other PN sequences, when they are re-positioned in $\mathbf{e}$. For example, this condition will *not* be satisfied, if the bits b_{66} and b_{98} (which are constituent of the same $n = 32$-bit PN sequence in $\mathbf{b}$, as shown in Figure 3.4) are interleaved to positions e_{343} and e_{341}, respectively (which are separated by only one other bit position, namely e_{342}). The second condition of achieving $D_{\mathrm{H}}^{\mathrm{min}} \geq 24$ requires each $n = 32$-bit PN sequence in $\mathbf{b}$ to have no more than one bit in $\mathbf{e}$ that is adjacent to a bit within each of the other PN sequences. For example, if the bits b_{32} and b_{34} are interleaved to positions e_{512} and e_{125}, respectively, and the bits b_{610} and b_{639} are interleaved to e_{124} and e_{513}, respectively, then the second condition will not be satisfied.

The described conditions of achieving $D_{\mathrm{H}}^{\mathrm{min}} \geq 24$ motivated the design of a GA [123], which was employed for selecting beneficial values for the parameters of Table 3.1. The first goal of the proposed GA was to maximise the minimum positional separation between any two bits in $\mathbf{e}$ that originate from the same $n = 32$-bit PN sequence in $\mathbf{b}$. The GA's second goal was to minimise the maximum number of occurrences that any two $n = 32$-bit PN sequences in $\mathbf{b}$ have bits that are positioned next to each other in $\mathbf{e}$. Clearly, in order to achieve these goals, the grade of freedom for the interleavers to re-position the N bits in the sequence $\mathbf{b}$ must be fully exploited, as described above.

3.5 Module implementation

This section describes a ASIC module that implements the interleaver and rate-1 encoder that are boxed in Figure 3.1. The schematic of Figure 3.5 is employed for the proposed module, which could be integrated between the PN spreader and the O-QPSK modulator of a standard IEEE 802.15.4 implementation, such as the Chipcon CC2430. In the following discussions, the proposed module's Input and Output (I/O) interface, datapath, ROM and controller are detailed.

The proposed module is specifically designed to avoid imposing any changes upon the I/O interfaces of the standard PN spreader and O-QPSK modulator. These exchange $n = 32$-bit PN sequences [102], at a rate of $f_{\mathrm{tx}}/n = 62.5 \cdot 10^3$ per second, where $f_{\mathrm{tx}} = 2 \cdot 10^6$ bits per second, as described in Section 3.2. These features motivate the proposed module's employment of a $f_{\mathrm{pr2}}^{\mathrm{aug}} = 62.5$ kHz clock, which is supplied using the 'Clk' port shown in Figure 3.5, as well as the 32-bit I/O ports 'Data_in' and 'Data_out'.

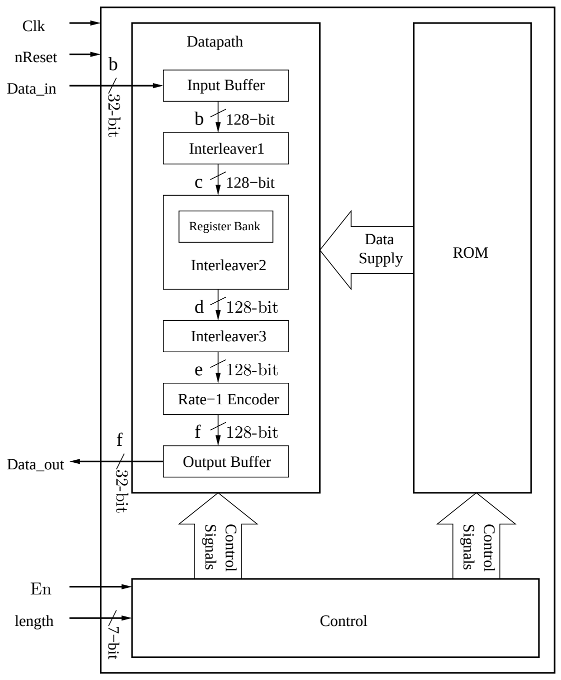


FIGURE 3.5: Schematic of the proposed hardware implementation.

Note that the proposed module has three other ports, as shown in Figure 3.5. The module begins operating when a logic one is placed upon the 'En' port, allowing its input port to be synchronised with the PN spreader's output port. As described in Section 3.3, the module operates in one of 118 different modes, depending on the length N of the bit sequence **b**. The value of N can be extracted from the 7-bit 'frame-length field' employed in the IEEE 802.15.4 PHY header [102], which conveys the number of bytes in the PHY payload **a**, namely M. This 'frame-length field' is provided to the module using its 7-bit 'Length' port. Finally, the 'nReset' port may be used to reset the registers employed within the proposed module.

A serial structure is employed within the datapath block of the proposed module, as shown in Figure 3.5. Here, the interleaver of Figure 3.1 is implemented in three stages, namely Interleaver 1, Interleaver 2 and Interleaver 3, as described in Section 3.3. These stages interleave multiple bits in parallel [131], in order to process the bits at the same rate that they are supplied by the PN spreader, as shown in the timing diagram of Figure 3.6 and detailed below. This approach facilitates a low processing latency and 'just-in-time' processing, which reduces the number of registers that are required to store intermediate results.

A uniform 128-bit dataflow width is chosen within the datapath block of Figure 3.5. This uniform width allows Interleaver 2, Interleaver 3 and the rate-1 encoder to be operated

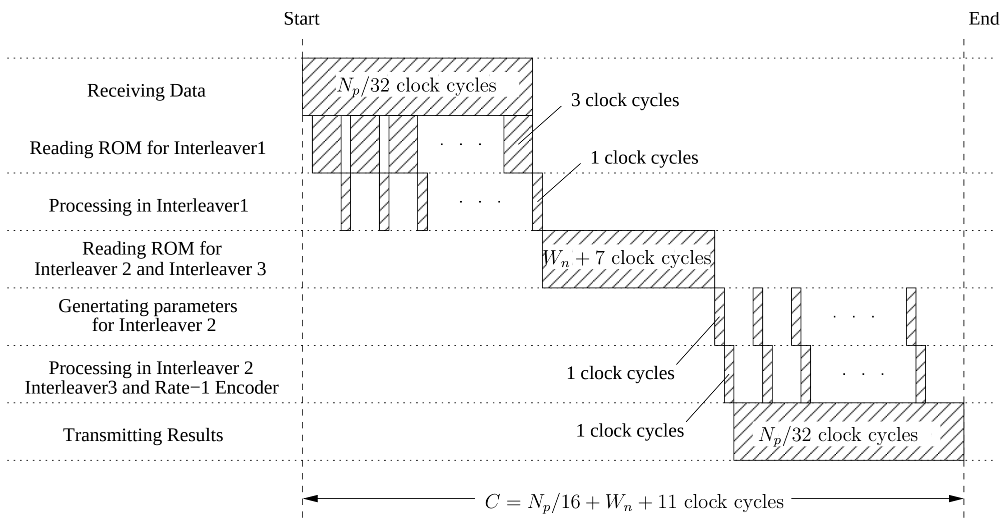

Start
End
Receiving Data
Reading ROM for Interleaver1
Processing in Interleaver1
Reading ROM for Interleaver 2 and Interleaver 3
Genertating parameters for Interleaver 2
Processing in Interleaver 2 Interleaver3 and Rate−1 Encoder
Transmitting Results
$N_p/32$ clock cycles
3 clock cycles
1 clock cycles
$W_n + 7$ clock cycles
1 clock cycles
1 clock cycles
$N_p/32$ clock cycles
$C = N_p/16 + W_n + 11$ clock cycles

FIGURE 3.6: Timing diagram of the proposed hardware implementation.

within a single clock cycle, without the need for intermediate registers. However, owing to its 8128-bit register bank, Interleaver 2 cannot be tightly connected to Interleaver 1. More specifically, this register bank must collect all of the bits from the sequence **c**, before Interleaver 2 can commence generating the sequence **d**, owing to the relative prime number based hops that are required, as discussed in Section 3.3. The 128-bit input and output buffers shown in Figure 3.5 are required owing to the different port widths employed inside and outside of the datapath block.

As described in Section 3.3, the three interleaving stages are parameterised as described in Table 3.1. In the proposed module, these parameters are stored in the ROM block shown in Figure 3.5. Combinational logic is employed to convert these parameters into the bit indices j_i, according to (3.1), (3.2) and (3.3) for Interleaver 1, Interleaver 2 and Interleaver 3, respectively. Here, the complexity of calculating (3.2) can be reduced, if it is implemented recursively according to

$$j_i = (j_{i-1} + p_N) \bmod N, \tag{3.4}$$

where $i \in [1, N-1]$ and $j(0) = s_N$. As shown in Figure 3.5, Interleaver 2 is required to interleave 128 bits in parallel. Note that 128 adders could be chained together to perform the required calculations of (3.4). However, the necessary combinational logic path would be too long to be performed within a single clock cycle. For this reason, a parallel adder chains is employed, each employing 16 adders to determine a different subset of the 128 bit indices j_i in a single clock cycle, striking a trade-off between bit area and speed.

As described above, 128-bit subsets of the sequence **b** are processed by Interleaver 1 as and when they are supplied to the input buffer, at a rate of 32 bits per clock cycle. However, 128 elements from the arrays $\mathbf{r}_u$ of Table 3.1 are required in order to calculate

(3.1) for each 128-bit subset of **b**. For this reason, a multiport ROM is employed to provide these 128 parameters in just three clock cycles, as shown in the timing diagram of Figure 3.6. In a fourth clock cycle, the 128 bits that have been loaded into the input buffer during the previous four clock cycles are interleaved and stored in the register bank of Interleaver 2. Note that during one of the four clock cycles in which Interleaver 1 is operated, it is necessary to both read from and write to the input buffer of Figure 3.5. When this process completes, the register bank of Interleaver 2 will store the bit sequence **c**, as shown in Figure 3.4.

Next, the registers that were used to store the parameters of Interleaver 1 are reused for storing the parameters of Interleaver 2 and Interleaver 3. Since the register bank of Interleaver 2 can store the bit sequence **c** indefinitely, there is no need to use multiport ROM accesses, when loading the parameters of Interleaver 2 and Interleaver 3. Hence, as shown in Figure 3.6, $W_N + 7$ clock cycles are used to load the parameters s_N, p_N and W_N, as well as the W_N elements of the array $\mathbf{w}_N$ of Table 3.1.

Following this, the bits of the sequence **c** are processed by Interleaver 2, Interleaver 3 and the rate-1 encoder in order to obtain the sequence **f**, as shown in Figure 3.4. Here, 128 bits are processed and written into the output buffer of Figure 3.5 at a time. As shown in Figure 3.6, 'just-in-time' processing is employed to populate the output buffer at the same rate that it supplies bits to the O-QPSK modulator of Figure 3.1, namely at 32 bits per clock cycle. Hence, one clock cycle is employed to recursively perform the calculations of (3.4), one clock cycle is employed to process the 128 bits and two idle clock cycles are employed.

As shown in Figure 3.6, the proposed module employs a total of $C^{\text{aug}}_{\text{pr2}} = N/16 + W_N + 10$ clock cycles to perform interleaving and rate-1 encoding.

3.6 Energy consumption analysis

In this section, the energy consumption results of the proposed module $P^{\text{pr}}_{\text{aug2}}$ are investigated within the context of the overall energy consumption $E^{\text{tx}}_{\text{aug}} + E^{\text{pr}}_{\text{aug2}}$. The effect that integrating this module into the Chipcon CC2430 hardware [122] is estimated. This estimation will be compared with that of [69], which considered the implementation of the interleaver and rate-1 encoder of Figure 3.1 in software running on the Chipcon CC2430's 8051 processor, as described in Section 3.1.

The Synopsys Design Compiler was employed to synthesise a gate-level implementation of the module detailed in Section 3.5. The synthesis employs a STMicroelectronics 0.12 μm technology standard cell library, resulting in a 1.6 mm^2 bit area, including the ROM. Synopsys PrimeTime was employed to determine the resultant implementation's average power consumption, which was found to be $P^{\text{pr}}_{\text{aug2}} = 666.9$ μW, where a supply voltage of $V = 3$ Volts is assumed. Assuming that the proposed module is shut down, when it is deactivated by placing a logic zero upon its En port, as shown in Figure 3.5. Hence, the duration $t^{\text{pr}}_{\text{aug2}}$ for which the proposed module consumes current is given by $t^{\text{pr}}_{\text{aug2}} = C^{\text{pr}}_{\text{aug2}}/f^{\text{pr}}_{\text{aug2}}$, where $C^{\text{pr}}_{\text{aug2}} = N/16 + W_N + 10$ and $f^{\text{pr}}_{\text{aug2}} = 62.5$ kHz, as described in Section 3.5. Finally, the proposed module's energy consumption is given by $E^{\text{pr}}_{\text{aug2}} = P^{\text{pr}}_{\text{aug2}} \cdot t^{\text{pr}}_{\text{aug2}}$, as shown in Figure 3.7.

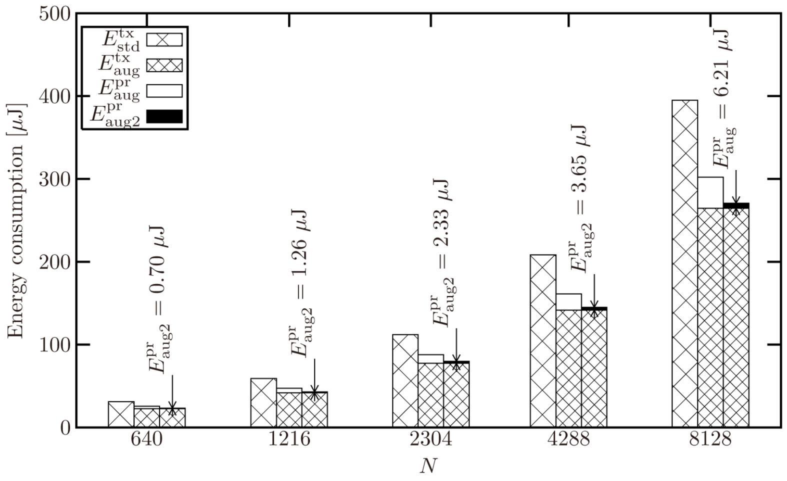


FIGURE 3.7: Total energy consumed in the standard Chipcon CC2430 PHY $E^{\text{tx}}_{\text{std}}$ and in the proposed ASIC implementation of the augmented PHY ($E^{\text{tx}}_{\text{aug}} + E^{\text{pr}}_{\text{aug2}}$).

Figure 3.7 provides the energy consumed by the proposed module $E^{\text{pr}}_{\text{aug2}}$ for payloads comprising various numbers N of bits. These energy consumptions are 76.7 – 83.4% lower than the $E^{\text{pr}}_{\text{aug}}$ values estimated in [69] for the case where interleaving and rate-1 encoding are performed in software running on the 8051 processor of a Chipcon CC2430. As a result, the energy savings $[E^{\text{tx}}_{\text{std}} - (E^{\text{tx}}_{\text{aug}} + E^{\text{pr}}_{\text{aug2}})]$ afforded by employing the augmented PHY are increased from 17.4 – 23.3% to 24.8 – 31.4% of $E^{\text{tx}}_{\text{std}}$, as shown in Figure 3.3. Indeed, the energy $E^{\text{pr}}_{\text{aug}}$ consumed by the proposed module only erodes 4.8 – 8.3% of the transmission energy reduction $(E^{\text{tx}}_{\text{std}} - E^{\text{tx}}_{\text{aug}})$ that it facilitates, which is considered to be an attractive engineering trade-off.

3.7 Conclusions

In this chapter, a detailed design of an augmentation to the IEEE 802.15.4 physical layer of WSN sensor nodes has been introduced. This augmentation significantly reduces the transmission energy required to achieve a target FER of 10^{-3} at the cost of requiring some additional processing within the sensor nodes. Using a gate-level implementation of the design, the overall energy consumption of the augmented PHY was estimated. The results showed that the energy consumed by the additional processing was negligible compared to the transmission energy saving that it facilitate. Indeed, the augmented PHY facilitates a significant reduction in the sensor nodes' overall energy consumption of 26.65 – 32.78%. Based on the study in this chapter, it can be concluded that with the proper management of the additional processing energy consumption that is introduced by a sophisticated ECC, such as turbo-like codes, the overall energy consumption of sensor nodes in WSNs can be reduced. The energy saving is significant in a star network because the encoders tend to be simple and only consume an insignificant part of the overall energy consumption. However, in order to apply the same philosophy in decode-and-forward multi-hop networks [132], the decoder must be considered as part of the energy trade-off between the transmission and processing energy consumptions. In the following chapters, the widely used turbo decoder, the Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) decoder considered in this regard. Its energy-efficient hardware design and its effect on the overall energy consumption in a WSN are fully investigated.

Chapter 4

Extrinsic information transfer chart based fixed-point turbo code parameter design

4.1 Introduction

As discussed in Section 1.4.2, when turbo-like codes are applied in multi-hop Wireless Sensor Networks (WSNs), decoding-and-forwarding is required at the sensor nodes. Hence, additional energy consumption is imposed. As a result, the overall energy consumption can only be reduced, when the transmission distance is sufficiently high, so that the transmission energy saving achieved by the employed Forward Error Correction (FEC) code is sufficiently high to overcome the additional energy consumption of the decoders. This challenging topic was investigated in [1, 2] for the sake of extending the life time of WSNs. Therefore, the energy consumption of the decoder's hardware implementation became an important factor in the design of a wireless sensor node. In this chapter, an important issue that is directly related to the energy consumption of a decoder's hardware implementation, namely the choice of fixed-point operand-word lengths is considered. An EXtrinsic Information Transfer (EXIT) chart based method is proposed for investigating the trade-offs between the operand-word lengths of the decoder and the achievable decoding performance. In contrast to the conventional method of using Bit Error Rate (BER) simulations for solving this design problem, the proposed method provides deeper insights into the specific causes of the performance degradations encountered. Additionally, the EXIT chart based method is less time-consuming than the BER based method, because the latter requires the consideration of a range of channel Signal-to-Noise Ratios (SNRs) in the associated Monte-Carlo simulations.

The performance of the turbo-like codes is usually evaluated based on floating-point simulation during the design process. However, in practical implementations, for reasons of energy efficiency, a fixed-point number representation would be preferable for Digital Signal Processors (DSPs), Field-Programmable Gate Arrays (FPGAs) or for Application-Specific Integrated Circuit (ASIC) implementations [133]. This is because compared to floating-point implementations, fixed-point implementations facilitate significant energy consumption reductions at a modest performance degradation [134].

As discussed in Section 2.1, one of the most important advantages of using the Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) algorithm is the reduced dynamic range of both its internal variables and of the Logarithmic Likelihood Ratios (LLRs). In practice, this allows a fixed-point representation to be used. In fixed-point implementations, the hardware complexity typically increases linearly with the internal operand-width of the data, since the operand-width determines the size of all the data paths and of the computing resources in the architecture [112]. Moreover, the iterative decoding process of turbo-like coding schemes requires a large amount of memory for storing the LLRs and the internal variables. Using less bits for each LLR and each variable is capable of significantly reducing the memory requirement, hence reducing the energy consumption of the decoder. Therefore, for a low-power implementation, minimising the number of bits required for representing the fixed-point quantities in the algorithm is an important issue. However, the information lost due to reducing of the operand-width lengths will degrade the performance. Hence, there is a trade off between the performance attained and the hardware complexity imposed, which has to be carefully considered, when aiming for a low-power design.

Various previous studies investigated the fixed-point implementation of turbo decoders by exploring the minimum word lengths of the different quantities to ensure a tolerable BER/Frame Error Rate (FER) degradation [133–141]. However, no universal conclusion has been obtained. Even though some of the studies considered the same specifications, namely those of the Universal Mobile Telecommunications System (UMTS)/Long Term Evolution (LTE) turbo decoder combined with Binary Phase-Shift Keying (BPSK) for transmission over an Additive White Gaussian Noise (AWGN) channel, the conclusions were somewhat different [133, 134, 136, 138, 140, 141]. This is because, in fixed-point implementations, a range of different issues affect the achievable decoding performance and diverse techniques were investigated for dealing with these issues.

The performance degradations caused by fixed-point implementations are primarily imposed by the underflow and overflow phenomena. In the context of underflow, the operand-fraction accuracy limits the overall computational accuracy of the calculations in the algorithm. Since the Jacobian logarithm in the LUT-Log-BCJR algorithm [142] is realised using piece-wise linear approximation stored in a Look-Up Table (LUT), the precision of the fixed-point representation is directly limited to the number of entries in the LUT. As discussed in Section 2.2.2, the $\max^*$ operator of the Jacobian algorithm is defined as:

$$\max^*(p,q) = \ln(e^p + e^q) \tag{4.1}$$
$$= \max(p,q) + \ln(1 + e^{-|p-q|}) \tag{4.2}$$
$$= \max(p,q) + f_c(|p-q|), \tag{4.3}$$

where the function $f_c = \ln(1 + e^{-|p-q|})$ may be readily approximated using a LUT. The number of entries (the quantised outputs) in the LUT is determined by how many quantised levels are defined by the specific fixed-point representation for covering the value range of the function f_c, $0 < \ln(1 + e^{-|p-q|}) < \ln(2)$. The fixed-point quantisation of f_c is shown in Figure 4.1. For example, using a $z = 3$-bit fraction-word length gives

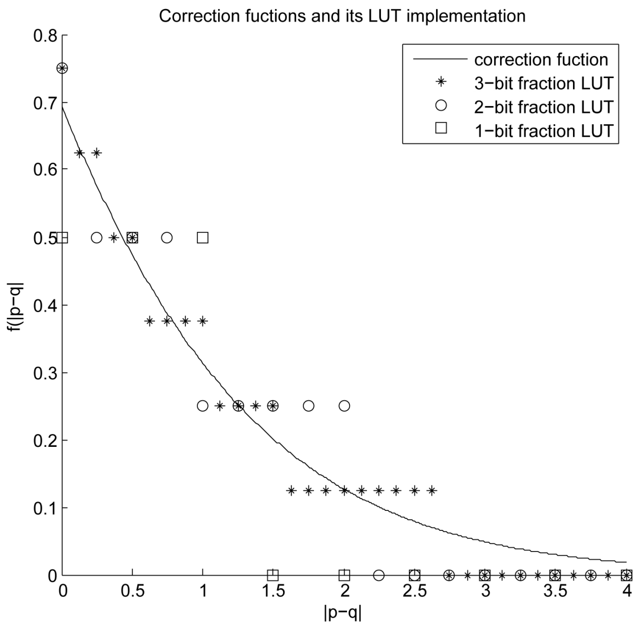


FIGURE 4.1: The floating-point correction function f_c and its fixed-point implementations using different fraction operand-widths.

a LUT resolution of 0.125, hence the LUT has the following seven elements, {0, 0.125, 0.25, 0.375, 0.5, 0.625, 0.75}. Similarly, a $z = 2$-bit fraction-word length gives a four-element LUT and a $z = 1$-bit fraction-word length defines a two-element LUT, as shown in Figure 4.1. Hence, the fraction-word length affects not only the width of databus, the computing resources and the memory requirements, but also the size of the LUT used.

In contrast to the above-mentioned underflow-issues, the occurrence of overflow depends on both the dynamic range of the variables and on the number of bits assigned to the integer part of the fixed-point representation. In the event of overflow, the lost information might become fatal for the success of the decoding all together. However, the dynamic range of the variables is difficult to predict and sometimes becomes quite wide, hence requiring a large number of bits in fixed-point representations to guarantee that the entire range is covered. In LUT-Log-BCJR decoders, there are only three different operations in the Add-Compare-Select (ACS) operations. The 'compare' and 'select' operations cannot induce any overflow. However, the 'add' operations might result in overflows. Consider the LUT-Log-BCJR algorithm of Section 2.2.2. In the decoding trellis of Figure 2.13, each α is given by the sum of two γ, an α value from the previous step and the correct function f_c of Equation 4.3. Since it includes an α from the previous step, the calculation of α is constituted by an accumulation of the α values in the trellis. Therefore, the values of α would increase without limits as the coded block-length is increased. The overflow imposed by having a limited word length is the most significant effect to be considered. The calculation of β imposes the same problem. According to Equation 2.17, the calculation of δ is constituted by the sum of α, β and γ, which might also result in an overflow. To deal with these issues, a number of different techniques have been proposed [136, 139].

The first approach is to arrange for the saturation of the over flowing data during its processing. This method is widely used in fixed-point digital filters [143, 144]. A disadvantage of this approach is that it requires some additional hardware within each computing element that might cause an overflow, such as adders. The simulation results of Section 4.3 will demonstrate that in its own right this technique is unsuitable for the LUT-Log-BCJR algorithm when applied in isolation, but it can work well in collaboration with a second technique, namely with normalisation [136].

More particularly, normalisation is applied in the context of the LUT-Log-BCJR algorithm for the sake of dealing with the overflow on the α and β internal variables. It scales down the increasing metrics during each step, in order to prevent them from increasing without a bound. This reduces the occurrence probability of overflows and allows the word length required for representing the variables to be further reduced. As discussed above, the α and β values are accumulated, as traversing through in the decoding trellis. For example, a decoding trellis of the UMTS/LTE turbo decoder is given in Figure 4.2. Based on the algorithm described in Section 2.2.2, $\alpha(S_4)$, $\alpha(S_5)$, $\alpha(S_6)$ and $\alpha(S_7)$ of Figure 4.2 accumulate from $\alpha(S_2)$ and $\alpha(S_3)$, which in turn accumulate from $\alpha(S_1)$. This accumulation continues as the forward recursion proceeds, with subsequent α values increasing without limits and hence potentially resulting in overflows. However, the decoding results do not depend on the absolute values of α — they only depend on the difference between the α values of the states at the same trellis stage [139]. For example, the calculation of the extrinsic LLR x_2 in Figure 4.2 is insensitive to the specific values of $\alpha(S_4)$, $\alpha(S_5)$, $\alpha(S_6)$ and $\alpha(S_7)$, but sensitive to the difference $\alpha(S_4) - \alpha(S_5)$, $\alpha(S_4) - \alpha(S_6)$, $\alpha(S_4) - \alpha(S_7)$ and so on. The same conclusion can also be applied to the β values. Therefore, the above-mentioned normalisation technique can be applied for controlling the dynamic range of the path metrics without affecting the decoding performance. This normalisation may be achieved by subtracting a constant from all the path metrics at a specific trellis stage [136, 139, 145]. A range of, different approaches have been used in previous studies. In [136], the path metrics

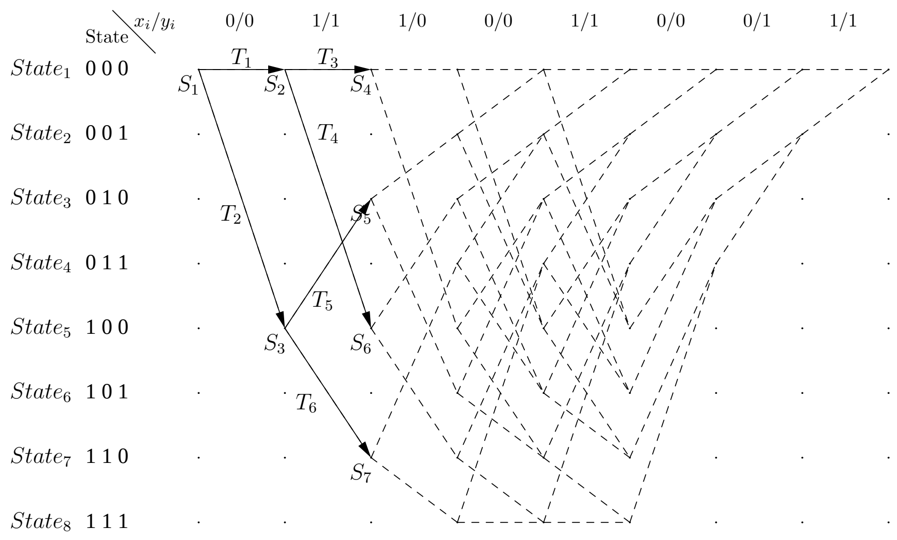
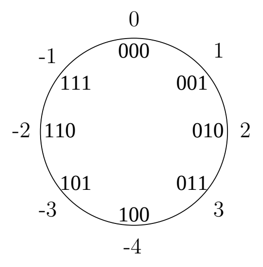


FIGURE 4.2: Possible accumulation routes in the an example decoding trellis of the UMTS/LTE turbo decoder.

were reduced by an amount corresponding to the minimum path metric at each trellis stage. By contrast, in [139], the path metrics were reduced by the maximum amongst them at each trellis stage. Both of these two approaches require extra computations for finding the minimum or the maximum path metric and for performing the subtractions. In [145], a modified version was advocated, where instead of searching for the smallest or largest metric at each step, the first trellis state metric is subtracted at each trellis stage. All the different approaches required different word lengths in the conclusions.

A third approach proposed in [136] explores the nature of the two's complement representation, namely that of the wrap-around technique. The basis of the wrap-around technique is that the decoding results of the LUT-Log-BCJR only depend on the difference between the path metrics, as discussed above. To demonstrate the wrap-around nature of the two's complement representation, three examples of the difference calculation are portrayed in Figure 4.3. Note that in the calculation of (1+3)-2 and (2+2+2)-3, both results in the brackets are overflown in a 3-bit two's complement representation, but the equivalent calculation in two's complement representation still gives the correct answer, as long as the result does not overflow. Therefore, even the calculation result of a path metrics is overflowed, as long as the difference between the two involved path metrics is still within the dynamic range of the chosen fixed-point parameterisation, the decoder can still give the correct result. However, this technique has its limitation on the tolerance of overflow. For the third calculation, the difference in the last calculation step is overflown. In this situation, the two's complement representation fails to provide the correct result, as shown in Figure 4.3. This technique was first introduced for Viterbi decoders in [146] and then it was also invoked for Soft-In Soft-Out (SISO) decoders [146]. In the LUT-Log-BCJR algorithm, it can be shown that all possible differences between the pairs of path metrics are upper bounded [146]. Therefore, as long as the difference



(1+3)-2=2
(001+011)-010=100-010=100+110=010=2

(2+2+2)-3=3
(010+010+010)-011=110-011=110+101=011=3

(2+2+2)-1=5
(010+010+010)-001=110-001=110+111=101=-3

Figure 4.3: Example of difference calculation in two's complement representation.

between two metrics is not above the largest possible value that can be represented by the specified word length in two's complement representation, the subtraction can be performed correctly using modulo 2^n arithmetic by simply ignoring the overflow of the operands. The advantage of the wrap-around technique is that no additional hardware requirements are imposed. According to [136], one or two more bits may be required for this approach compared to the subtractive normalisation and saturation, because it only has a limited capability of tolerating overflows, as demonstrated by the third example of Figure 4.3. The trade-off between introducing extra hardware for the normalisation/saturation and using the wrap-around technique has to be carefully considered, depending on the turbo code specifications.

In conclusion, to implement a LUT-Log-BCJR algorithm in fixed-point representation, the different techniques have different word length requirements. In the similar previous contributions of [133–138, 140, 141], the different environment, design and implementation configurations lead to different conclusions. A brief summary of the configurations of the above eight papers is provided in Table 4.1. Three similar turbo codes are considered in these papers, as shown in Figure 4.4. The type-2 configuration corresponds

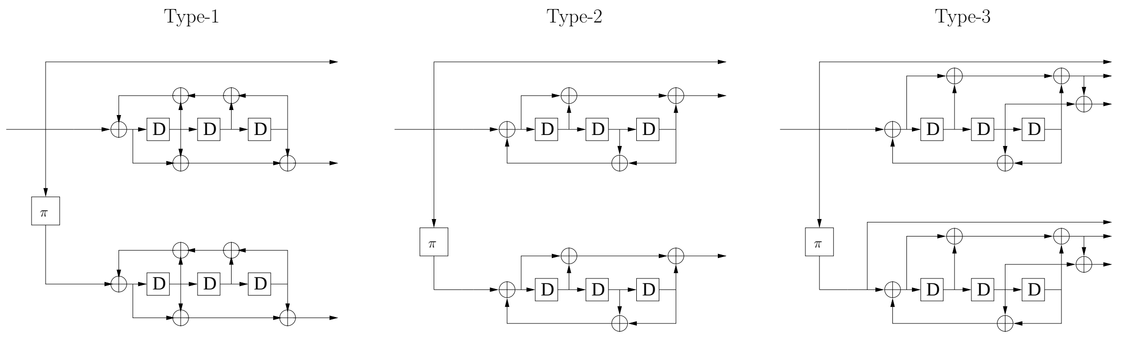


Figure 4.4: Three different types of turbo codes investigated in the previous studies of Table 4.1.

to the UMTS/LTE turbo encoder discussed in Chapter 2.

Only a few papers discussed the effects of finite word length based on mathematical techniques [136, 138, 139]. However, it was shown in [136, 139] that mathematical techniques are inadequate for deciding the desirable word lengths in practice. More specifically, mathematical techniques are capable of determining the upper bounds of the path metrics that will never be exceeded [139], nonetheless, practical BER/FER simulation results

Authors	[135]	[138]	[133]	[136]
Encoder	Type-1	Type-2	Type-2	Type-2
Modulation	BPSK	BPSK	BPSK	BPSK
Channel	AWGN	AWGN	AWGN/Rayleigh	AWGN/Rayleigh
Interleaver	helical	N/A	N/A	3GPP compliant
Block length (bit)	216	4828	600	600
Iteration times	5	10	5/10	5/7/10
Overflow control	Normalisation	Saturation	N/A	Normalisation Saturation
Look-Up-Table	16 elements	22 elements	7/10 elements	N/A
Authors	[134]	[141]	[137]	[140]
Encoder	Type-2	Type-2	Type-3	Type-2
Modulation	BPSK	BPSK	BPSK	BPSK
Channel	AWGN	AWGN	AWGN/Rayleigh	AWGN/Rayleigh
Interleaver	block prime	N/A	N/A	ideal
Block length (bit)	1024	N/A	640	2896
Iteration times	3/8	5/8	N/A	7/18 half
Overflow control	Saturation	N/A	N/A	Normalisation
Look-Up-Table	2 elements	7 elements	2/4/8 elements	N/A

Table 4.1: A summary of previous studies on fixed-point parameterisations of turbo decoders.

show that the further reduction of the word lengths beyond the theoretical bounds can be tolerated in practice, resulting only in a modest reduction of the achievable performance [136]. As a result, the appropriate fixed-point parametrisation of a decoder cannot be carried out purely on the basis of mathematical analysis.

Traditional BER/FER Monte-Carlo simulation based methods are time-consuming. Additionally, the BER/FER curves do not provides any insight into the iterative decoding convergence of the process. The different types of variables in a decoding scheme tend to have different desirable word length requirements. Hence, a large number of combinations have to be tested with the aid of time-consuming simulations to identify the desirable settings. Hence, considering the effects of different techniques with the aid of Monte-Carlo simulations is on the verge of being unfeasible. In this chapter, an EXIT chart [100] analysis based technique is proposed for determining the desirable fixed-point parameterisations of turbo-like decoders. The EXIT chart based investigations are naturally less time-consuming compared to BER/FER simulations. Moreover, the BER simulation results only characterises the decoding performance of a particular number of iterations in the decoding process, while an EXIT chart explicitly characterise the convergence behaviour of the decoder, hence allowing an arbitrary number of iterations to be considered.

The results of Section 4.3 will demonstrate that based on the proposed EXIT chart analysis method, not only the decoding performance can be investigated, but also the reasons for imposing a performance degradation by a specific fixed-point implementation may be identified. This insightful information assists the designers in determining the appropriate technique of preventing any degradation, hence determining the desired word lengths more promptly. For presenting the proposed method, the 3rd Generation Partnership Project (3GPP) UMTS/LTE turbo decoder's [116] desirable word length settings are investigated. The results are compared to these of previous works. Additionally, the investigations show that the proposed method may be readily applied for any turbo code and potentially any turbo-like code, including iterative decoding schemes which can be analysed by EXIT charts.

As introduced in Section 2.3, the EXIT chart constitutes a powerful tool of analysing the convergence behaviour of iterative systems, such as turbo-like decoders. Unlike BER/FER simulations, generating an EXIT chart is less time-consuming, since the simulation of the interleaver in the decoder and that of the actual iterative decoding process is not required. Although the effects of a realistic finite-duration sub-optimal interleaver cannot be explicitly revealed, an interleaver only changes the order of the information bits, but no information is lost during the interleaving process due to a fixed-point implementation. Since the objective of the proposed method is to investigate the performance degradation imposed by fixed-point implementations, the interleaver would not affect the results of the proposed method. Nonetheless a disadvantage of the EXIT chart based method is that it only considers a fixed SNR while a BER/FER chart considers a wide range of SNR or E_b/N_0 values. The target SNR can be chosen to be that particular value, where the tunnel is narrow and the performance is most sensitive to the fixed-point representation's limitations. Additionally, the EXIT simulations are less time-consuming than BER/FER simulations. Naturally, it is possible to draw different EXIT charts for different SNRs if necessary. In summary, EXIT charts constitute a more efficient tool than BER/FER charts, when aiming for determining the desirable word lengths for a fixed-point decoder.

The effect on EXIT chart simulations recorded for different word lengths of a turbo decoder were first provided in [147], albeit no convincing conclusions were given. Hence, in this chapter, a detailed analysis method of using EXIT charts for determining the desirable word length setting of fixed-point implementations of turbo-like decoders is proposed for the first time, by providing comprehensive investigations for the design example of the UMTS/LTE turbo decoder [116]. In Section 4.2, the proposed method is applied for investigating the desirable word length setting for the UMTS/LTE turbo decoder under the comprehensive consideration of fixed-point implementation techniques. The conclusions are then compared to those of previous treatises in Section 4.3. Section 4.4 concludes the chapter.

4.2 Extrinsic information transfer chart analysis of the fixed-point UMTS/LTE turbo Decoder

The specifications and structure of the UMTS/LTE encoder and decoder pair were presented in Chapter 2. BPSK modulated transmission over an AWGN channel was considered, initially assuming SNR = -4 dB. As shown in Section 4.3, at this SNR, the EXIT chart of the UMTS/LTE turbo code has a moderately open tunnel, hence the performance degradations imposed by fixed-point operands may be readily observed. An SNR = -4.83 dB is also used, where the EXIT tunnel is almost closed, because both the onset of the turbo cliff in the EXIT chart and the BER performance are most sensitive to the limitations imposed by the fixed-point representation, which allows the desirable operand-width to be critically appraised. Random bit sequences are fed into the input of the turbo encoder. An interleaver length of 453-bit is used in the simulations, which is the geometric mean of the minimum and maximum block length in the UMTS standard. This value is opted because the performance degradation of turbo codes is proportional to the block length [148]. The shortest (40-bit) and longest (5114-bit) frame lengths in the UMTS Standard are also used in the simulations for investigating the effect of the frame length on the achievable performance. Additionally, the conclusions emerging from previous works [133–138, 140, 141] are summarised for the sake of demonstrating the validity of the proposed method.

Firstly, three different versions of the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm are investigated using floating-point operand representations. These three algorithms are the LUT-Log-BCJR algorithm relying on the exact calculation of the Jacobian logarithm, the LUT-Log-BCJR algorithm using the eight-element look-up-table (LUT) based Jacobian logarithm and Maximum Logarithmic Bahl-Cocke-Jelinek-Raviv (Max-Log-BCJR) algorithm [118]. It has been shown in [145] that the performance loss imposed by using the Jacobian logarithm is less than 0.1 dB relative to the exact log-domain calculation, which is usually considered acceptable. The performance degradation of the Max-Log-BCJR UMTS/LTE turbo decoder has also been well documented [149, 150]. According to [150], the E_b/N_0 performance degradation observed at a BER of 10^{-5} between the LUT-Log-BCJR and Max-Log-BCJR algorithms is a moderate 0.3 dB for a block length of 640 bits and 0.54dB for the 5114-bit block length for transmission over AWGN channels and higher in Rayleigh fading channel [149], which is considered significant. The purpose of this work is to have similar fixed-point EXIT chart results to those of the floating-point LUT-Log-BCJR technique, while considering an EXIT chart similar to that of the floating-point Max-Log-BCJR unacceptable.

Secondly, the effects of having a limited fraction-part on the overall fixed-point representation are investigated. Since the limitation of fraction-part also limited not only the accuracy, but also the number of elements in the LUT, the effects of the number of elements in the LUT is also considered.

Thirdly, the effects of a limited integer-part on the fixed-point representation are investigated. The three overflow control approaches discussed in Section 4.1 are also investigated.

Finally, based on the analysis to be provided in Section 4.3.2.4, the desirable combinations of fraction-length and integer-length are investigated under the assumptions of different interleaver block lengths. The effect of trellis termination techniques will also be investigated in Section 4.3.2.4. The BER simulation results for the selected specifications are also provided for validating the proposed method. The conclusions obtained by the proposed method are then compared to those of previous studies [133–138, 140, 141].

4.3 Design study

4.3.1 Comparison of different floating-point log-domain methods

Figure 4.5 portrays the EXIT chart of the UMTS/LTE turbo decoder using the abovementioned three variations of the log-domain BCJR decoding algorithms, namely the Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) algorithm, the LUT-Log-BCJR algorithm and the Max-Log-BCJR algorithm. The open EXIT tunnel between the two

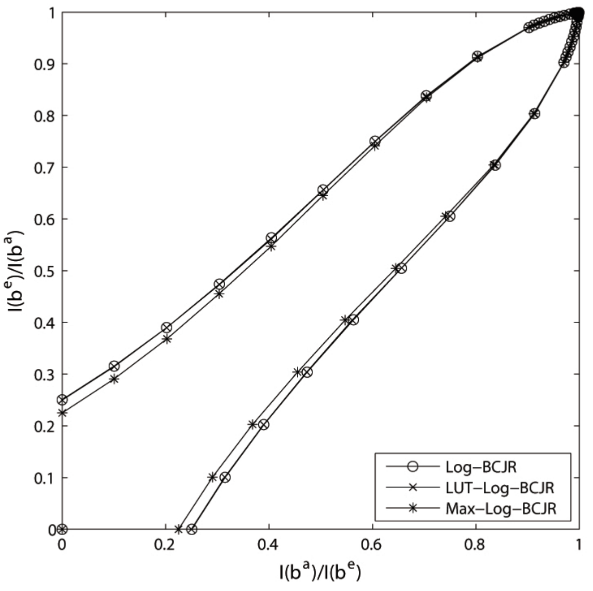


FIGURE 4.5: The EXIT chart of three variations of the log-domain BCJR decoding algorithm.

curves becomes narrower due to the information-loss imposed by the Max-Log-BCJR. Therefore, given a certain number of decoding iterations, the mutual information associated with the Max-Log-BCJR remains lower than that of the LUT-Log-BCJR algorithm, viewing the associated degradation from a different perspective, in order to obtain a certain target BER, more decoding iterations might be required by the Max-Log-BCJR algorithm. Therefore, it can be anticipated that the BER degradation due to the information lost by the fixed-point implementation can be reflected in the EXIT chart. Further simulation results will underline this conclusion.

4.3.2 Comparison and analysis of fixed-point implementations

In order to investigate the decoding performance of using fixed-point operand representations in the UMTS/LTE turbo decoder, fixed-point operands are used for all the variables in the proposed EXIT chart based investigations. As described in Chapter 2, the operand-width specification includes those of the integer-part and of the fraction-part. Additionally, according to the previous studies in [133–138,140,141], the dynamic ranges, which are determined by the operand-width of the integer-part in fixed-point representation, of the input LLRs and of the internal variables of a turbo decoder have to be specified separately for the sake of obtaining the desirable operand-widths.

As described in Section 4.1, the gradually increasing values of the internal variables is caused by the accumulated additions of the input LLRs, which is the typical cause of the overflow during the decoding process. Therefore, clipping the input LLRs to a smaller integer operand-width than that of the internal variables is capable of preventing an overflow. As a result, there are three parameters that have to be determined for computing the operand-width specifications of a turbo decoder, namely the integer operand-width of the input LLRs, the integer operand-width of the internal variables and the fraction operand-width of all variables in the simulation. The two integer operand-widths dominate the effect of the overflow during the decoding process. The fraction operand-width determines the effect of the underflow during the decoding process. The total number of possible specifications is given by all the combinations of the legitimate values of these three parameters. In order to reduce the number of specifications that have to be characterised, the operand-widths of both the integer-part and of the fraction-part are investigated separately.

Firstly, a long operand-width of 32 bits is assumed for both the integer operand-widths, while using a limited fraction operand-width for investigating the performance of a limited operand-precision. Secondly, the opposite scenario is considered in order to investigate the degradation imposed by the limited dynamic range. For the sake of presenting the simulation results, the notation FP(x, y, z)-Log-BCJR is used for specifying the parameters of the fixed-point approximation in the Log-BCJR decoder. In the above notation, z is the fraction operand-width of the fixed-point representation applied, while y is the integer operand-width, including the sign bit, of the internal variables. The parameter combination of $x < y$ identifies the integer operand-width of the input LLRs, when they are input to the LUT-Log-BCJR decoder. This is necessary, for the sake of mitigating the effects of overflow. For example, in the FP(3, 4, 2)-Log-BCJR scenario, an *a priori* LLR having the value of 1011.11 would be clipped to 1100.00, which is -4 in its decimal representation.

Firstly, for an n-bit fraction part representation, up to 2^n elements are used in the LUT, as described in Section 4.1. The corresponding EXIT chart results are shown in Figure 4.6. The simulation results show that using a coarse 1-bit fraction-part and two elements in the LUT imposes an observable degradation on the EXIT chart, while a 2-bit fraction-part associated with four elements in the LUT gives almost no observable degradation in the EXIT chart. As shown in Figure 4.6, 2-bit fraction-length guaran-

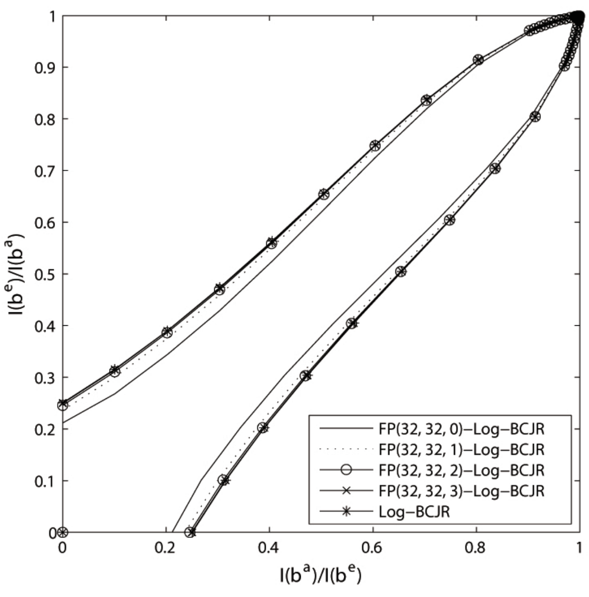


FIGURE 4.6: The EXIT chart of fixed-point LUT-Log-BCJR decoder using different fraction operand-widths.

tees a virtually identical result compared to the floating point result. Similarly, a 1-bit fraction-length also results in an EXIT chart similar to the floating-point result. The associated degradation is lower than that imposed by the floating-point Max-Log-BCJR algorithm. Note that having 0-bit fraction-length effectively removes the LUT, transforming it from the LUT-Log-BCJR to the Max-Log-BCJR. However, the associated EXIT chart degradation is more severe than that of the Max-Log-BCJR which is a consequence of the low resolution used for the variables.

Considering the trade-off between the complexity imposed and the performance attained, relying on 1-bit or 2-bit fraction-length may be deemed appropriate for most applications. Further discussion are deferred, until the combined simulation results associated with a limited fraction and integer lengths are considered. The BER analysis of different fraction lengths is given in [134, 138, 141]. Both [134, 138] concluded that a 2-bit fraction-length is capable of approaching the performance of the floating-point decoder. Nonetheless, the authors of [141] suggested that a 3-bit fraction-length provides a better performance, which only incurs a SNR penalty of 0.015 dB. The simulation results of [138] demonstrated that a 1-bit fraction length only causes a loss of 0.1 dB for medium-to-low SNRs, but has no detrimental consequences on the error floor performance. In the eight papers studies considered [133–138, 140, 141], five opted for a 2-bit fraction-length as the most desirable choice and three of them chose a 3-bit fraction-length.

Numerous authors have investigated the desirable operand-width of the different internal variables much as the input LLRs, α, β, γ and λ. However, in practice, it is inconvenient to store and process different variables using different accuracy [140, 141]. Although the authors of [133] claimed that further operand-width minimisation applied to the different variables is capable of reducing the switching activity, which has an influence on the energy consumption, as the integration density increases, the associated total power consumption contribution by the dynamic power component becomes smaller and smaller. Thus, the benefits of reducing the operand-width of the different variables considered is reduced. A further potential disadvantage of different operand-widths is that the employment of such a strategy requires additional extension and clipping mechanism for the data-buses in the datapath, which increases the design complexity. Therefore, a single operand-width setting is considered. Nonetheless, it is necessary to consider the input LLRs and the internal variables of the SISO decoder separately, because the limited accuracy of the input LLRs directly affects the dynamic range of the internal variables. According to [138], the possible differences α and β between the pairs of path metrics Δ_{MAX} (i.e. the possible differences between the α values or β values in the decoding trellis), plays an influential role in the LUT-Log-BCJR decoder, as discussed in Section 4.1, which are upper-bounded by the following function of the input LLRs' dynamic range:

$$\Delta_{MAX} = min[w \times M_u + d_{min}(w) \times M_c], \tag{4.4}$$

where $d_{min}(w)$ is the minimum Hamming-weight of the bit sequences generated by the input bit sequences having a weight of w, $\pm M_u$ and $\pm M_c$ are the dynamic ranges of the SISO decoder's two inputs, namely of the extrinsic information and of the LLRs output by the soft demodulator. Hence, $d_{min}(w)$ depends on the bit sequence considered. Furthermore, $\pm M_u$ and $\pm M_c$ are simply related to the operand-width of the integer part of the input LLRs. As discussed in the previous section, it is important to keep the difference between pairs of metrics within a limited range for the sake of maintaining an unimpaired performance. However, the different overflow control techniques require different operand-widths for guaranteeing this condition. Therefore, based on the discussions in [138], the operand-width of the internal variables typically requires more bits than that of the input LLRs. As a result, the operand-width specifications of the fixed-point representation based turbo decoders has to be carefully considered on an individual basis for any specific code, because the specifications derived for any specific turbo code cannot directly applied to another code.

Furthermore, given the different operand-width settings for the input LLRs and the internal variables, the conversion between different representations has to be carefully managed. Extending a shorter operand-width to a longer one would not induce any problem since the values remain unaltered. More explicitly, an extension mechanism simply attaches zeros to solve this problem. This may be readily realised in hardware and no extra operations are required. However, conversion in the other direction may inflict a performance degradation. Moreover, the Most Significant Bit(MSB) in two's complement representation determines the sign of the operand. Hence, simply ignoring the extra bits during the conversion may in fact change the polarity of the operand, which would significantly affect the success of the decoding process. Hence, a clipping mechanism associated with saturation is required during the conversion. If the original

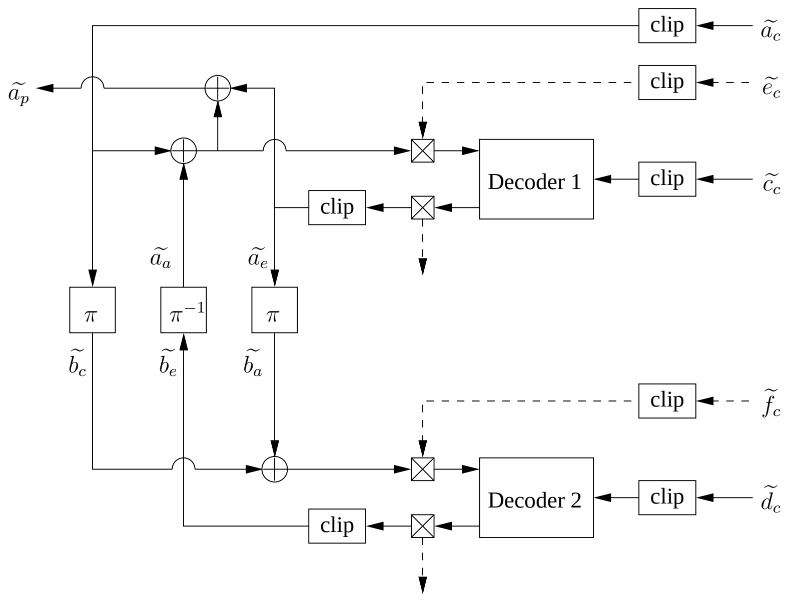


FIGURE 4.7: The schematic of the UMTS/LTE turbo decoder relying on operand-clipping.

operand value is over the affordable operand-width, the converted operand must be set to the maximum legitimate value. This method requires extra hardware in practice and extra operations in the simulations. Figure 4.7 portrays the schematic of the clipping operations used in the decoder.

4.3.2.1 Wrapping technique

As discussed in Section 4.1, the two's complement representation can naturally prevent the errors caused by the overflows for LUT-Log-BCJR algorithm, since the overflowed data may be considered as being wrapped in a circle. Hence the difference between two operands remains the same. The benefit of this wrapping technique is that no extra hardware is required. Therefore, it is suitable for the scenarios where sufficient memory is available or a low-dimensional datapath is required. Note however that the clipping of the input LLRs is still required. The wrapping technique detailed in Section 4.1 is only suitable for the calculation of the internal variables.

Figure 4.8 shows the associated EXIT chart results for the FP(5, 7, 32)-Log-BCJR, FP(4, 7, 32)-Log-BCJR and FP(3, 7, 32)-Log-BCJR scenarios. The simulation results of

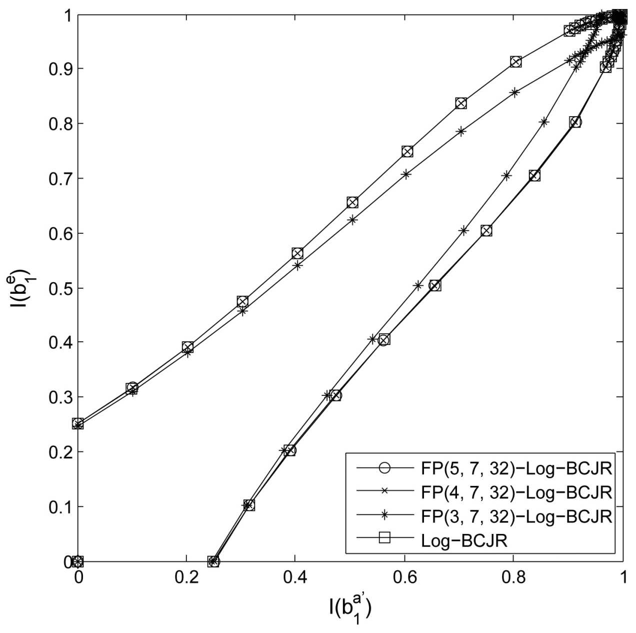


FIGURE 4.8: The EXIT chart for different integer operand-widths, including FP(5, 7, 32)-Log-BCJR, FP(4, 7, 32)-Log-BCJR and FP(3, 7, 32)-Log-BCJR, in conjunction with the wrapping technique.

Figure 4.8 suggest that for the FP(5, 7, 32)-Log-BCJR algorithm almost no degradation is observed in the EXIT chart compared to the floating-point result. By contrast, the EXIT chart result of the FP(3, 7, 32)-Log-BCJR scenario indicates that the tunnel closed before the curves reached the (1,1) point, which suggests that the decoding performance would be significantly degraded.

The FP(5, 7, 32)-Log-BCJR and FP(4, 7, 32)-Log-BCJR scenarios provide different conclusions based on the EXIT chart simulations, which are shown in Figure 4.9. More specifically, Figure 4.9 is a zoomed version of Figure 4.8. For the FP(5, 7, 32)-Log-BCJR algorithm, the EXIT function $I_e(I_a)$ reaches a peak value at a certain I_a abscissa values and then starts decreasing, which results in a closed tunnel. Hence, the BER performance of the FP(4, 7, 32)-Log-BCJR scenario cannot compete with that of the FP(5, 7, 32)-Log-BCJR algorithm.

Note that in Figure 4.9, the EXIT tunnels of the FP(5, 7, 32)-Log-BCJR and FP(3, 7, 32)-Log-BCJR scenarios become closed in different ways, which reveals the different reasons for their closures. For FP(3, 7, 32)-Log-BCJR, the closure is caused by the lower gradient of the curves $I_e(I_a)$ seen in Figure 4.9. Since the only difference between the FP(3, 7, 32)-Log-BCJR and FP(4, 7, 32)-Log-BCJR schemes is having a 1 bit lower

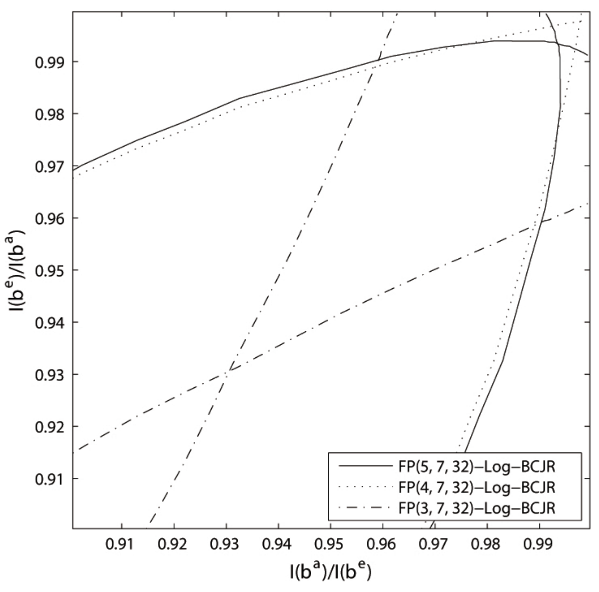


FIGURE 4.9: The EXIT chart's zoomed-in top-right corner extracted from Figure 4.8 for different integer operand-widths, including FP(5, 7, 32)-Log-BCJR, FP(4, 7, 32)-Log-BCJR and FP(3, 7, 32)-Log-BCJR, in conjunction with the wrapping technique.

integer operand-width for the input LLRs, it may be conjectured that the lower $I_e(I_a)$ is due to the LLR degradation imposed by the reduced operand-width. Thus, having a 4-bit integer operand-width is the minimum operand-width for the input LLRs. For the FP(5, 7, 32)-Log-BCJR scheme, the closure is due to the decay of the curves $I_e(I_a)$ beyond their peak points. The EXIT chart result shows that the bit-width specification of the FP(4, 7, 32)-Log-BCJR is adequate for all the variables. Hence it may be inferred that the performance degradation of FP(5, 7, 32)-Log-BCJR is due to the insufficient integer operand-width difference between the input LLRs and the internal variables. More explicitly, when the number of iterations is increased, the mutual information converged by the *a priori* LLRs is also increased, which means that the average absolute value of the LLRs is also increased. Due to the add and accumulate operations of the input LLRs in LUT-Log-BCJR algorithm, having an insufficient difference between the operand-width of the input LLRs and the internal variables [1] may cause a serious overflow problem in the calculations of the internal variables. Hence for abscissa values above 0.95 the function $I_e(I_a)$ starts to decrease. This reveals that the overflows of the internal variables are beyond the tolerance limit of the wrapping technique, which results in the EXIT curve failing to reach the (1,1) point. This effect may be explicitly observed in Figure 4.10 and Figure 4.11.

Specifically, for the FP(5, 7, 32)-Log-BCJR and FP(6, 7, 32)-Log-BCJR scenarios the peak point of the curves occur in Figure 4.10 earlier due to the even more limited operand-width difference between the input LLRs and the internal variables. With a

[1]The difference may be calculated by $y - x$ for FP(x, y, z)-Log-BCJR.

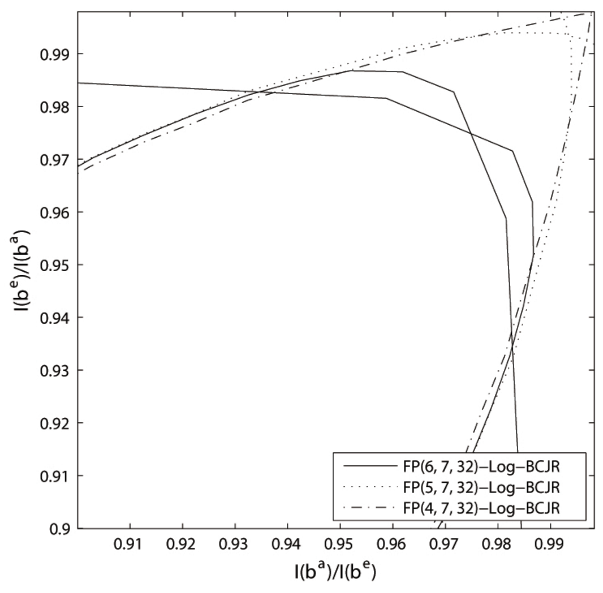


FIGURE 4.10: The EXIT chart's zoomed-in top-right corner for different integer operand-widths, including FP(6, 7, 32)-Log-BCJR, FP(6, 7, 32)-Log-BCJR and FP(4, 7, 32)-Log-BCJR, in conjunction with the wrapping technique.

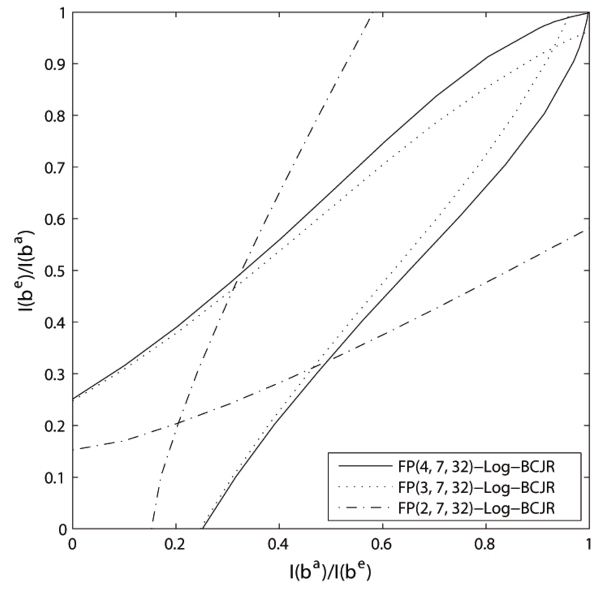


FIGURE 4.11: The EXIT chart for different integer operand-widths, including FP(4, 7, 32)-Log-BCJR, FP(3, 7, 32)-Log-BCJR and FP(2, 7, 32)-Log-BCJR, in conjunction with the wrapping technique.

further reduction of the integer operand-width of the input LLRs, the best performance is achieved by FP(4, 7, 32)-Log-BCJR. On the other hand, as shown in Figure 4.11, when the integer operand-width of the input LLRs becomes less than 4 bits, the performance starts to degrade. However, since the difference remains sufficient, no decaying tendency is observed for the curves of FP(3, 7, 32)-Log-BCJR and FP(2, 7, 32)-Log-BCJR. Only the positive gradient of the curves is lower due to the insufficient operand-width of the input LLRs, which caused the closure of the open EXIT tunnel. If the internal variables' integer operand-width is further reduced to 6 bits, the EXIT tunnel always becomes closed before reaching the (1,1) point, regardless of the integer operand-width of the LLRs.

In conclusion, having a 4-bit integer operand-width for the input LLRs is the minimum acceptable setting for UMTS/LTE turbo decoders. For the wrapping technique, the minimum difference between the input LLRs and the internal variables is 3 bits. Therefore, the desirable integer length setting is constituted by the FP(4, 7, 32)-Log-BCJR scenario.

4.3.2.2 Saturation technique

The wrapping technique requires no additional operations or hardware for handling the overflow of the internal variables. Another simple overflow control technique often used in practice is the saturation technique. As mentioned before, the input LLRs are clipped due to the reduced operand-width of the internal variables in the specification. The same technique may also be applied to the internal variables. However, this approach has a disadvantage, when applied to the LUT-Log-BCJR decoders. The problem is namely that the LUT-Log-BCJR algorithm is using the distance between different groups of the internal variables for determining the output of the decoder, i.e. the extrinsic LLRs, as described before. The saturation technique clips all the overflowing data values to the maximum or minimum value that can be represented by the specified fixed-point representation. The related variation imposed on the values of the overflowing data may change the distance between two internal variables. Under extreme conditions, when two variables overflow in the same direction, both of the variables are clipped to the maximum or minimum value of the specific fixed-point representation considered, hence the distance between them becomes 0. The simulation results of Figure 4.12 demonstrated that this problem makes the results of the saturation technique even worse than those of the wrapping technique. Figure 4.12 portrays the simulation results for using the saturation technique, which may be contrasted to Figure 4.8 using the wrapping technique. Since the conditions for the input LLRs remain unchanged, the minimum integer operand-width of them remains 4 bits. However, the required operand-width difference between the input LLRs and the internal variables is significantly increased due to the application of the saturation technique.

Although it may be observed in Figure 4.12 that for the setting of FP(4, 12, 32)-Log-BCJR, the EXIT tunnel is closed before reaching the (1,1) point, as shown in the figure, the point of closure is close to the (1,1) point and the EXIT curves have almost no difference with respect to the floating point result. Hence, it may be concluded that for Figure 4.12 that FP(4, 12, 32)-Log-BCJR is the desirable integer operand-width setting

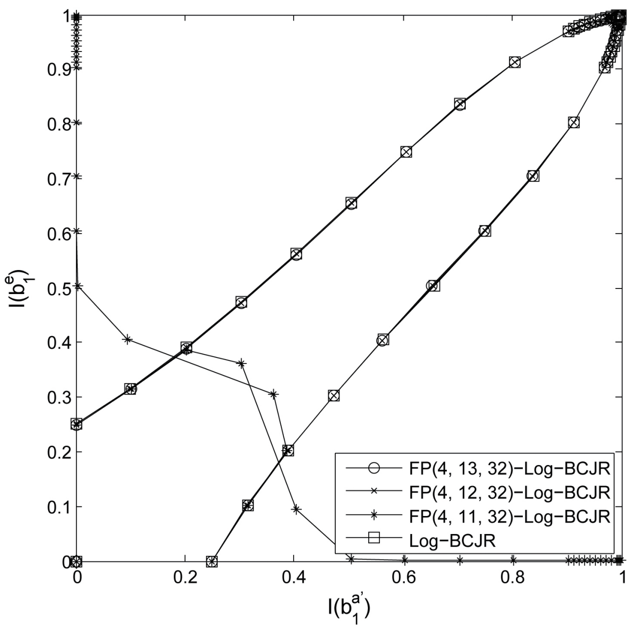


FIGURE 4.12: The EXIT chart for different integer operand-widths in conjunction with the saturation technique.

for the saturation technique. The corresponding EXIT chart result is almost identical to the floating-point result in Figure 4.5. For the FP(4, 11, 32)-Log-BCJR scenario the EXIT tunnel becomes closed far before reaching the (1,1) point.

Note in Figure 4.12 that, in contrast to Figure 4.8, the results of using the wrapping technique, when the integer operand-width of the internal variables is reduced to 11 bits, the function $I_e(I_a)$ falls to near 0 values soon after reaching the peak point. As discussed in the context of Figure 4.9, the reason for forcing an EXIT chart curve (i.e. the function $I_e(I_a)$) to decay is the insufficient difference of the integer lengths between the input LLRs and internal variables. Since the EXIT chart of the FP(4, 13, 32)-Log-BCJR scenario can reach the (1,1) point, in Figure 4.12, having a 4-bit integer operand-width for the input LLRs is still sufficient for the saturation technique. Hence, the saturation technique increases the required integer operand-width difference between the input LLRs and the internal variables, as seen by comparing Figure 4.9.

As mentioned before, the decoding result only depends on the difference between the path metrics (i.e. internal variables), but not on their actual values. The saturation technique clipped the overflowing variables to the positive and negative limits. It may be speculated that while the overflowing internal variables clipped to their limited values, the difference between the path metrics becomes 0. Hence no reliable soft outputs can be obtained. Once a number of internal variables overflow, the EXIT chart curves rapidly decay to 0, as shown in Figure 4.12 for the result of FP(4, 11, 32)-Log-BCJR scenario.

In conclusion, the saturation technique is not suitable for LUT-Log-BCJR decoders. However, in next section, further simulation results not included here showed that the employment of saturation is necessary in combination with the normalisation technique for controlling the overflow in LUT-Log-BCJR decoders [145]. Only a delicate combination of the saturation and normalisation techniques is capable of obtaining the most desirable operand-width setting.

4.3.2.3 Normalisation technique

A limitation of the wrapping technique is that if the difference between the path metrics exceeds the dynamic range, the associated path metric subtraction would not give the correct result. The goal of the saturation technique is to eliminated this problem. However, demonstrated by the simulation results seen in Figure 4.12, for the saturation technique, it imposed another problem, namely that many of the overflowing variables are clipped to the same value. Hence, a normalisation technique is introduced for dealing with this problem. The simulation results in this section show that given the appropriate combination of saturation and normalisation, the integer operand-width requirement of the internal variables can be further reduced. As the variables α and β are gradually accumulated, the largest one of them is subtracted from them in each step. For example, in the UMTS/LTE turbo decoder described in Chapter 2, the decoder's trellis has eight α variables at each trellis stage of Figure 2.10. Their values depend on the specific α and γ values of the previous trellis stage of Figure 2.10. Therefore, as the α values are accumulated along the trellis, during the decoding process, the eight α values of the current step are reduced by the largest one of them, before calculating the α values of the next step. Therefore, the accumulation of the α values can be decelerated. The dynamic range of the α values may be significantly reduced. The same process is also applied for the calculations of the β values. The corresponding EXIT chart results are shown in Figure 4.13. The desirable operand-width setting of the integer length is FP(4, 5, 32)-Log-BCJR, which necessitates two bits less for the internal variables.

When employing the saturation and normalisation techniques for the UMTS/LTE turbo decoder of Chapter 2, the operand-width required for the internal variables is 1 bit lower than for the wrapping technique of Section 4.3.2.1. The drawback of the saturation and normalisation approach is that extra operations are required for the decoding process in addition to the LUT-Log-BCJR algorithm. Hence, the extra hardware has to be employed in practical implementations. Additionally, the extra calculations may also require extra processing time, which inevitably reduces the decoding speed. As a result, both approaches has their own benefits and disadvantages, hence, they may be suitable for different turbo decoders used in diverse applications. The trade-offs between employing the saturation and normalisation approach or the wrapping approach have to be carefully considered bearing in mind the different constraints of the specific application, the affordable complexity, energy consumption, decoding speeds, and so on.

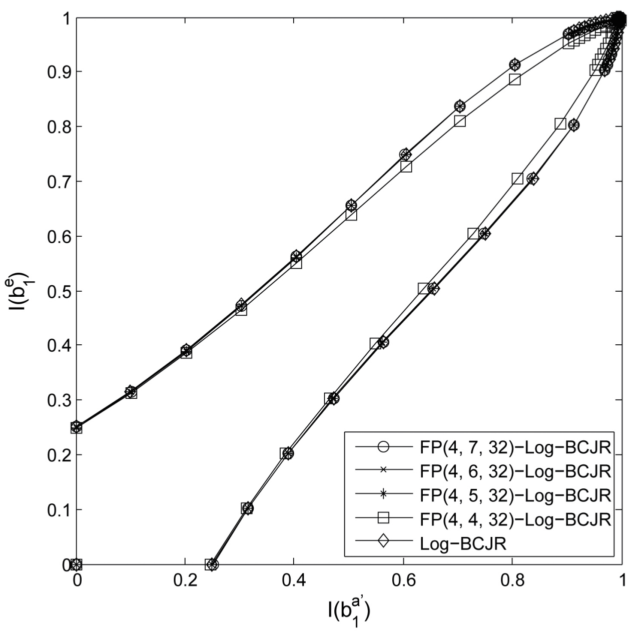


FIGURE 4.13: The EXIT chart of different integer operand-widths in conjunction with the normalisation technique.

4.3.2.4 Final validation

In order to determine and validate the desirable operand-width setting for the fixed-point implementation of the UMTS/LTE turbo code, the proposed EXIT chart based investigations considered various combinations of integer and fraction lengths. Since the simulation results of Figure 4.6 only limited the fixed-point fraction length, in the overall study of this section, both 1-bit and 2-bit fraction lengths are considered in the final validation simulations. The fraction length settings combined with the desirable integer length setting and applied for both the wrapping technique FP(4, 7, 32)-Log-BCJR as well as for the normalisation technique FP(4, 5, 32)-Log-BCJR are evaluated here. Moreover, for the different settings, different turbo code specifications are considered, which include the longest 5114-bit block length, the shortest 40-bit block length and the most critical SNR value of -4.83 dB where the EXIT becomes just open.

When considering the normalisation technique of Section 4.3.2.3, the final validation is shown in Figure 4.14 for the longest block length, in Figure 4.15 for the shortest block length and in Figure 4.16 for the most sensitive SNR value of -4.83 dB. According to the results of Figure 4.14 to 4.16, FP(4, 5, 2)-Log-BCJR attains almost the same performance, as the floating-point solution, despite considering different situations. By contrast, the FP(4, 5, 1)-Log-BCJR imposes further degradations due to the combined effects of the limited integer and fraction lengths, albeit this degradation is not as severe as for the Max-Log-BCJR. Moreover, according to the simulation results Figure 4.14 and 4.15, the block length does not have a significant effect on the EXIT chart. Consequently, a single compromise specification can perform adequately for any block length.

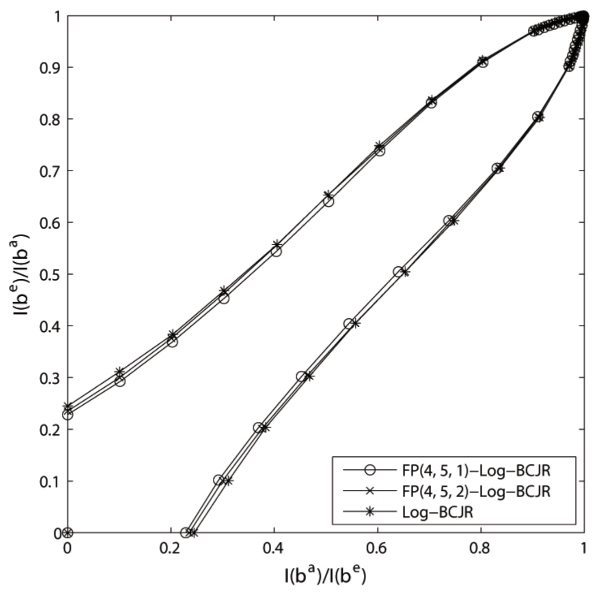


FIGURE 4.14: The EXIT chart of 5114-bit block length relying on fixed-point operand-width representation, using the normalisation technique in comparison to the floating-point solution.

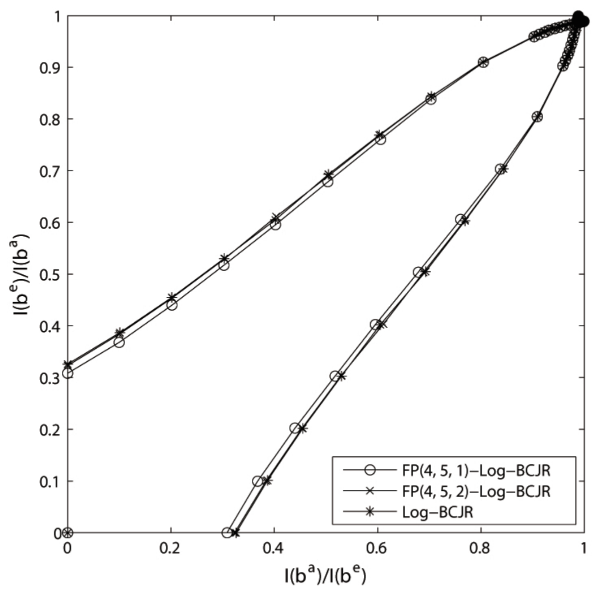


FIGURE 4.15: The EXIT chart of 40-bit block length relying on fixed-point operand-width representation, using the normalisation technique in comparison to the floating-point solution.

Therefore, it is concluded for the normalisation technique that FP(4, 5, 2)-Log-BCJR represents the desirable operand-width setting for the UMTS/LTE turbo decoder.

The final validation results are shown in Figure 4.17 for the wrapping technique for the longest block length, while in Figure 4.18 for the shortest block length. Finally,

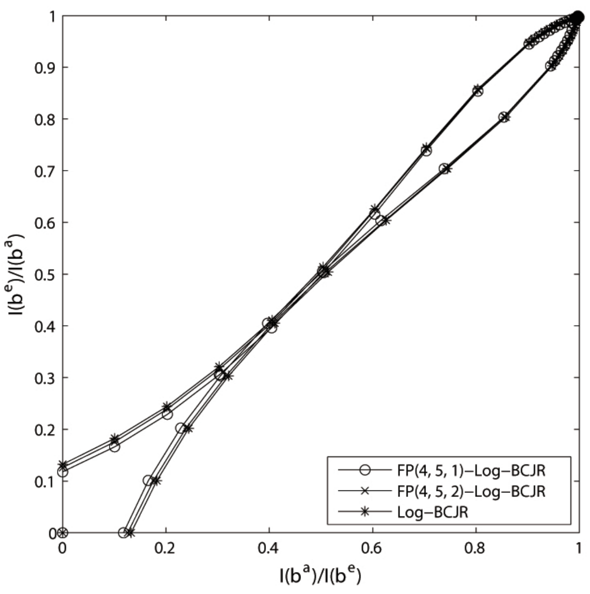


FIGURE 4.16: The EXIT chart of SNR = -4.83—dB/453-bit block length relying on fixed-point operand-width representation, using the normalisation technique in comparison to the floating-point solution.

Figure 4.19 portrays the corresponding result for the most sensitive SNR value of -4.83 dB. It emerges from these results that the 2-bit fraction part represents the best

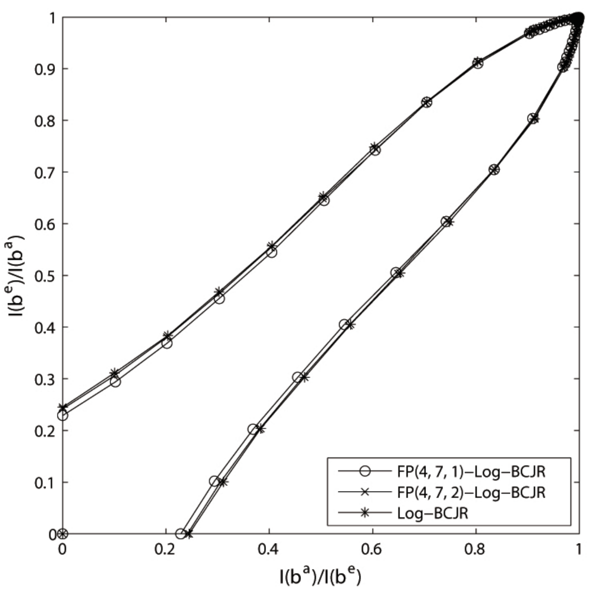


FIGURE 4.17: The EXIT chart of 5114-bit block length relying on fixed-point operand-width representation, using the wrapping technique in comparison to the floating-point solution.

option in this case. Hence, FP(4, 7, 2)-Log-BCJR is the desirable operand-width setting of the UMTS/LTE turbo decoder for the wrapping technique.

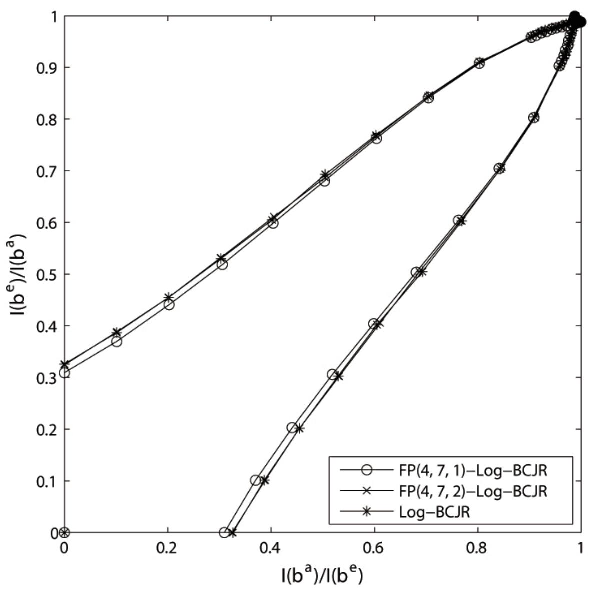


FIGURE 4.18: The EXIT chart of 40-bit block length relying on fixed-point operand-width representation, using the wrapping technique in comparison to the floating-point solution.

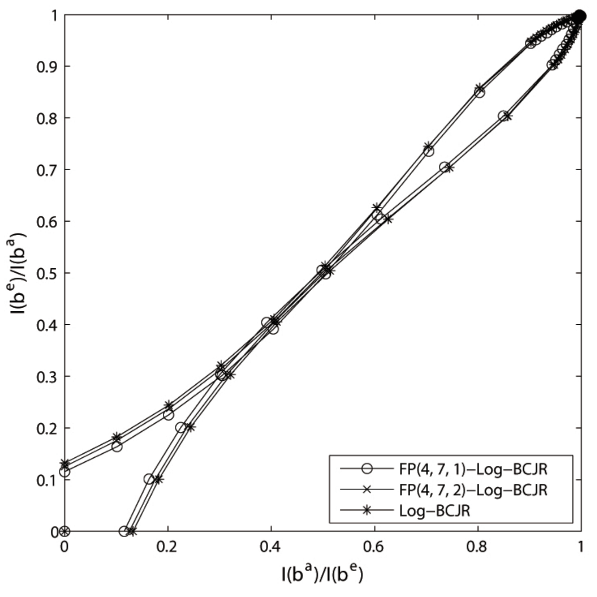


FIGURE 4.19: The EXIT chart of SNR = -4.83 dB/453-bit block length relying on fixed-point operand-width representation, using the wrapping technique in comparison to the floating-point solution.

Finally, a range of selected BER simulation results are provided in Figure 4.20 for validating that the EXIT chart analysis results are indeed confirmed by the BER performance results. As shown in the figure, the BER results of both FP(∞, ∞, 0)-Log-BCJR and of FP(3, ∞, ∞)-Log-BCJR exhibit a performance degradation compared to the floating-point results due to having an insufficiently high operand-width for both the fraction

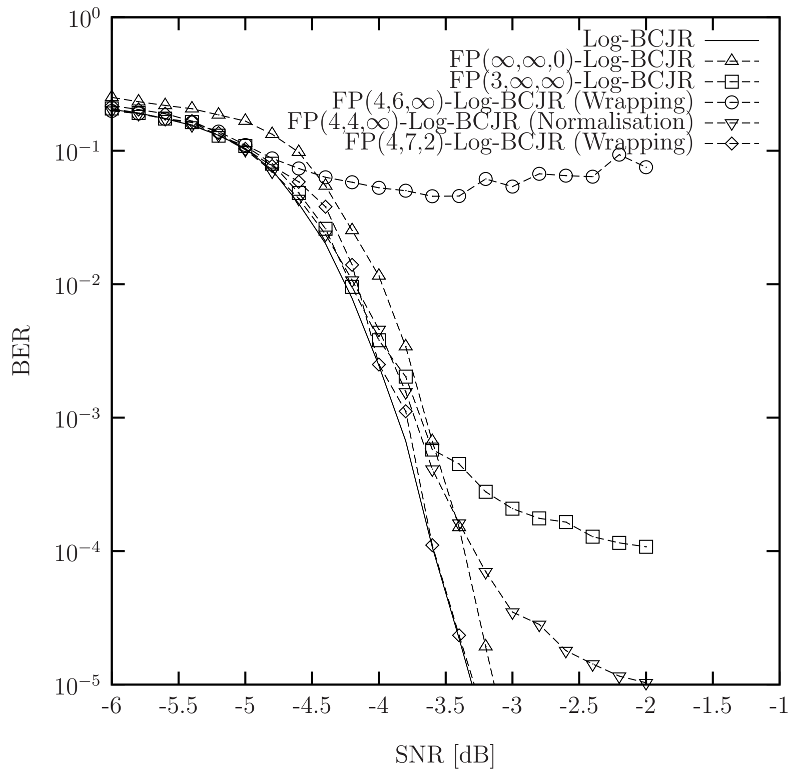


FIGURE 4.20: A selection of BER results for a 453-bit block length using different operand-width specification and different overflow control techniques for validating the proposed method.

part and for the integer part of the LLRs. The BER results of FP(4, 6, ∞)-Log-BCJR confirmed with the wrapping technique and of the FP(4, 4, ∞)-Log-BCJR relying on the normalisation technique exhibit a performance degradation owning to having an the insufficient operand-width difference between the LLRs and the internal variables. The BER result of Figure 4.20 for FP(4, 7, 2)-Log-BCJR using the wrapping technique shows almost no performance degradation which agrees with the conclusion given by the proposed method.

Note that while the diverse causes of performance degradation detailed above impose different effects upon the EXIT functions during the analysis, they all have the a similarly detrimental effect upon the BER results shown in Figure 4.20. Therefore, it may be argued that the EXIT chart analysis offers deeper insights that are not revealed by BER analysis, hence allowing desirable parameterisations to be found without relying on a brute-force full search across the entire parameter space.

In order to compare the results of this chapter to the selected previous work seen in Table 4.1, the authors of [133,138,140] claimed that having a 3-bit integer operand-width is sufficient for the input LLRs. However, they only considered the input LLRs received from the channel. In the EXIT chart simulations, the input LLRs considered constitute the *a priori* input of the concatenated decoders, which includes the channel's output and the extrinsic information gleaned from the other decoder of Figure 2.14. Since they are both input to the concatenated decoders, it is reasonable to assume that they have the same operand-width. Hence, the simulation results in this chapter showed that 4-bit integer operand-width constitute a desirable setting for the input LLRs. The authors of [137] arrived at the same conclusion concerning the integer operand-width of the input LLRs. By contrast, the authors of [134–136, 141] did not consider the input LLRs separately. For the internal variables, the studies in [135,137] considered all the different internal variables (i.e. γ, α, β and δ) separately. The authors of [135, 137] considered eight bits for the longest integer operand-width of the internal variables, while the study in [141] concluded that assigning 7-bit was the desirable setting, while the conclusion of [140] was that of using six bits. The investigations of [133, 134] suggested that using five bits was the most desirable setting. The reason for arriving at diverse conclusions is the choice of experimental different circumstances, as shown in Table 4.1. For example, the authors of [140] used normalisation in their simulations, but only at a couple of specific processing steps in the decoding process, not at each step.

4.4 Conclusions

An EXIT chart based technique was conceived for investigating the desirable operand-width setting of the fixed-point representation used in a turbo decoder. Using the appropriate operand-width setting is important for the hardware implementation of the turbo decoders, since it has a direct impact on their complexity, energy consumption and speed. The desirable codec parameterisation depends on the turbo code's design, on the specific decoding algorithm invoked and a range of other factors. The proposed EXIT chart analysis based technique substantially simplifies the conventional BER analysis. The employment of the proposed technique for the design of the UMTS/LTE turbo decoder demonstrated the advantage of the proposed method compared to the conventional method based on time-consuming BER analysis. Furthermore, the EXIT chart based technique reveals the essential reasons for any specific performance degradation caused by the inappropriately chosen operand-width specifications. The investigation of this chapter using the EXIT chart based fixed-point turbo code parameter design technique reveals three reasons that may impose a performance degradation by a specific fixed-point implementation, which the conventional BER analysis failed to provide.

1. When the fraction operand-width is insufficient, the decoding performance degrades due to reduced resolution of the LUT in the LUT-Log-BCJR algorithm.

2. When the integer operand-width of the LLRs is insufficient, the LUT-Log-BCJR decoder fails to receive enough information from the input LLRs and hence the decoding performance degrades.

3. When the integer operand-width is not sufficiently longer than the integer operand-width of the LLRs, the overflows during the decoding process will become severe,

hence degrading the decoding performance. The required difference between the integer operand-widths of the internal variables and the LLRs varies depending on the overflow control techniques employed.

As a result, for a specific turbo decoder associated with an appropriately chosen overflow control technique, the three parameters of the fixed-point implementation should be investigated individually, in the order of the fraction operand-width of all the variables, the integer operand-width of the LLRs and the integer operand-width of the internal variables, as demonstrated in Section 4.3. Furthermore, during the design process of the parameter specification of a fixed-point turbo decoder, the reasons for the performance degradation high lighted by the EXIT charts can be used for finding the desirable parameterisation without time-consuming brute-force search across a vast design space. Finally, the conclusions obtained for the UMTS/LTE turbo decoder were compared to those of previous studies on the same subject for the sake of validating the benefits of the proposed method.

In conclusion, the findings of this chapter indicate for the UMTS/LTE turbo decoders with a BPSK modulation for transmission over an AWGN channel that the FP(4,7,2)-Log-BCJR setting using the wrapping technique and the FP(4,5,2)-Log-BCJR relying on the normalisation technique employ the lowest possible operand-widths that avoid a significant performance degradation and therefore offer the most attractive trade-off between energy-consumption and performance. Note that this specific example is used for demonstrating the proposed method and comparing with previous work. In practice, this method can be generally applied to different turbo codes with different modulations and channel models.

However, the operand-width setting is not the only important issue in developing a low-complexity and hence energy-efficient turbo decoder. The next stage of the implementation process, namely the hardware architecture design of the decoder is also important. Since turbo codes have not been considered for employment in energy-constrained applications, such as WSNs, in Chapter 5 a low complexity energy-efficient turbo decoder architecture is proposed.

Chapter 5

A low complexity energy-efficient turbo decoder architecture

5.1 Introduction

As discussed in Chapter 1, the near Shannon limit performance of turbo-like codes can be exploited to reduce the transmission energy consumption in a wireless communication system. Hence, they are finding applications in Wireless Sensor Networks (WSNs), where the energy consumption is dominated by wireless communication, and the communication systems are energy-constrained [1, 2, 89, 90]. Chapter 3 showed that turbo-like codes can be easily applied in star topology WSNs for reducing transmission power when only one-way message transmission from the sensor nodes to the central node is required. This is because the high complexity turbo-like decoder is only required on the central node, which typically has sufficient energy supply, and the extra energy consumption cost by inducing the required encoder on the sensor nodes is negligible compared with the transmission energy reduction that it affords owing to its improved Bit Error Rate (BER) performance. However, in some WSN scenarios, a high number of sensors and a long average transmission distance are employed. As discussed in Chapter 1, in WSNs designed for environmental monitoring, a sensor network including hundreds of sensor nodes could be deployed into a large and variable environment. The sensor nodes are required to communicate with each other over ranges from several metres to hundreds of metres depending on the application for extended periods of time, while relying on batteries that are small, lightweight and inexpensive. In these applications, multi-hop network topologies or two-way communication may be applied, requiring the decoder of the applied Error-Correcting Code (ECC) to be embedded on the energy-constrained sensor nodes. However, the computational complexity of turbo-like decoders are typically significantly higher than the corresponding encoders. For the typical transmission distance of WSNs, the energy consumption of the turbo decoders may not be negligible compared with the overall energy consumption of the communication systems. Therefore, in these cases, the transmission energy saving by introducing the turbo code can be partially or even completely offset by the energy consumption of the decoder. In order to benefit from employing turbo codes, the trade-off between different coding gains and computational complexities of the decoding algorithms as well as the average transmission distance of the target application has to be carefully considered. In addition, minimising the energy consumption of the decoder at the hardware implementation level is important for improving the energy saving offered by employing turbo codes.

Compared with conventional turbo codes applications, energy-constrained applications exploit the near Shannon limit performance of turbo codes in a different way. Conventionally, turbo codes have found application in cellular and broadcast standards, such as 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE) [103] and Digital Video Broadcasting (DVB) [104] standards. These schemes aim for a high spectral efficiency (bit/s/Hz) and in particular a high throughput (bit/s) in order to facilitate high data rate real-time communication. Therefore, previous Application-Specific Integrated Circuit (ASIC) implementations [151–157] of turbo codes have been designed for achieving a high processing throughput, at the cost of having a relatively high energy consumption. On the other hand, in energy-constrained scenarios, instead of increasing the transmission throughput towards the theoretical upper limit, turbo codes can be employed to maintain a particular throughput and instead reduce the required transmission energy towards the corresponding lower limit. The coding gain offered by turbo codes allows them to achieve a particular transmission spectral efficiency β at a lower Signal-to-Noise Ratio (SNR) per bit E_b/N_0. However, in these scenarios only a modest throughput is required, since relatively low transmission throughputs of less than 1 Mb/s are typical [4,18], particularly if transmission duty-cycling is employed. Therefore, having a low complexity and low energy decoder becomes a higher priority than having a high throughput when applying turbo codes for the purpose of energy saving. For this reason, the trade-off between the energy consumption and processing throughput of ASICs iterative soft decoders designed for these energy-efficient scenarios is different from that of area spectrally efficient applications. Therefore, the previously proposed Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) decoder architectures [151–157] are not suitable for energy-efficient applications, since they prioritise throughput over energy efficiency.

In this chapter, the trade off between the decoding and transmission energy consumption in WSNs when employing turbo codes in the communication systems is investigated in Section 5.2. The energy efficiency of the conventional architecture for turbo decoder hardware implementation is analysed in Section 5.3. Based on the investigation, a low-complexity energy-efficient turbo decoder architecture is proposed in Section 5.4. In Section 5.5, a LTE turbo decoder is implemented using the proposed architecture, its energy efficiency is analysed and compared with similar previous works. Finally, Section 5.6 concludes the chapter.

5.2 Trade off of employing turbo codes for energy saving

As mentioned in Section 5.1, when employing turbo codes in energy-constrained applications, the transmission energy $E_{\mathrm{b}}^{\mathrm{tx}}$ (measured in nJ/bit) can be reduced owing to the turbo coding gain. However, the reduction in $E_{\mathrm{b}}^{\mathrm{tx}}$ is partially offset by the additional energy that is consumed by the turbo decoder $E_{\mathrm{b}}^{\mathrm{pr}}$. For these reasons, turbo decoders conceived for energy constrained scenarios are required to facilitate a low *overall* energy consumption of $E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{pr}}$. As discussed in Chapter 2, there are a number of variations of the turbo decoding algorithm, which provide different coding gains and computational complexities. The coding gain determines the reduction in transmission energy $E_{\mathrm{b}}^{\mathrm{tx}}$ and the computational complexity is directly related to the decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$. The investigation in this section will show that the differences between different decoding algorithms can have a significant effect on the overall energy consumption, when employing turbo codes in wireless communication systems. In particular, this section considers the two most popular variations of the turbo decoding algorithm, the LUT-Log-BCJR algorithm and the Maximum Logarithmic Bahl-Cocke-Jelinek-Raviv (Max-Log-BCJR) algorithm. As discussed in Chapter 2, The Max-Log-BCJR algorithm has a 50% lower computational complexity than the LUT-Log-BCJR algorithm, at the cost of 0.5 dB less coding gain in an uncorrelated Rayleigh fading channel [149]. The much lower complexity of the Max-Log-BCJR algorithm has a significant effect on the hardware implementation. Table 5.1 summarises several state of the art designs for the LUT-Log-BCJR and Max-Log-BCJR decoders of the LTE standard turbo code. As shown in Table 5.1, when only considering the energy consumption of the decoder, the Max-Log-BCJR decoders achieve a much higher throughput and a lower energy consumption than the LUT-Log-BCJR decoders.

However, the Max-Log-BCJR decoder has a 0.5 dB BER performance degradation compared to the LUT-Log-BCJR decoder, as the simulation results show in Figure 5.1. This performance degradation requires a 0.5 dB higher transmission energy consumption, in order to maintain the same BER in wireless communication. The absolute transmission energy consumption can be estimated using the transmission power model of [1,2]. Here, the environmental parameters and WSN system specifications in Table 5.2 are assumed. In the table, note that $N_0 = 10 \times \log(k * T)$, where $k = 1.3806503 \times 10^{-23}$ is Boltzmann constant and $T = 300K$ is room temperature.

The path loss can be estimated by,

$$P_{\mathrm{l}}(d)[\mathrm{dB}] = 20 \log_{10}\left(\frac{4\pi}{\lambda}\right) + 10p\log_{10}(d), \tag{5.1}$$

Table 5.1: Comparison of the implemented Turbo decoder.

	LUT-Imp-1	LUT-Imp-2	LUT-Imp-3	Max-Imp-1	Max-Imp-2
Publication	[154]	[156]	[158]	[159]	[160]
Algorithm	LUT-Log	LUT-Log	LUT-Log	Max-Log	Max-Log
Block size (bit)	5114	5114	5114	6144	6144
Technology (nm)	180	180	180	65	120
Supply voltage (V)	1.8	1.8	1.8	-	1.2
Area A (mm^2)	9	14.5	8.2	2.1	3.57
(Scaled for 90 nm)	(2.25)	(3.63)	(2.05)	(4.0)	(2.0)
Gate count (exclusive of memory)	85k	410k	65k	-	553k
Memory required (Kb)	239	450	161	-	129
Clock frequency F (MHz)	111	145	100	300	390.6
Decoding iterations	10	8	6.5	6	5.5
Throughput T (Mb/s)	2	10.8	4.17	150	390.6
Power consumption (mW)	292	956	320	300	788.9
(Scaled for 90 nm)	(36.5)	(119.4)	(40)	(796.4)	(332.8)
Energy consumption (nJ/bit/iteration)	14.6	11.1	12.7	0.31	0.37
(Scaled for 90 nm)	(1.8)	(1.4)	(1.59)	(0.81)	(0.16)
$E_b^{tx} + E_b^{pr}$ (nJ/bit) when transmitting over 39 m (5 iterations)	17.16	15.16	16.06	13.42	10.17
$E_b^{tx} + E_b^{pr}$ (nJ/bit) when transmitting over 58 m (5 iterations)	48.92	46.92	47.82	49.88	46.63
$E_b^{tx} + E_b^{pr}$ (nJ/bit) when transmitting over 100 m (5 iterations)	361.7	359.7	360.6	409.0	405.8

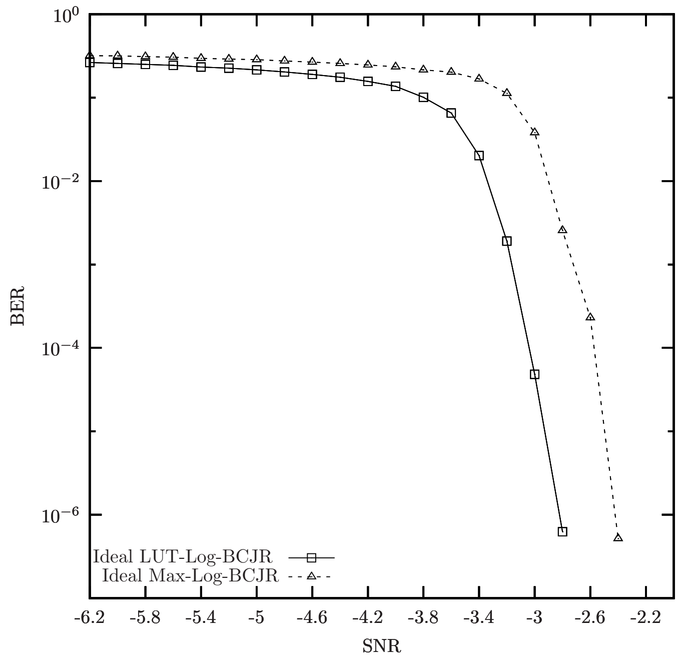


FIGURE 5.1: The BER performance of the LUT-Log-BCJR decoder and the Max-Log-BCJR decoder.

Transmission frequency (f)	5.8 GHz
Power amplifier efficiency (A)	33%
Receiver noise figure (r)	4
Path loss exponent (p)	4
BER target	10^{-4}
Uncoded system minimum received SNR at the target BER (S_0)	34 dB
Temperature	300 K
Thermal noise (N_0)	-203.8 dBm

TABLE 5.2: Environment assumptions and system specification of the estimated WSN.

where $\lambda = c/f$ is the wave length of the transmission signals carrier, $c = 2.998 \times 10^8$ m/s is the speed of light and d is the transmission distance.

The transmission SNR required to achieve the target BER when no coding is employed is given by

$$S_\mathrm{u} = N_0 + S_0 + r + P_\mathrm{l} + A. \tag{5.2}$$

The improved SNR when employing a turbo code having a coding gain G_c is given by

$$S_\mathrm{c} = S_\mathrm{u} - G_\mathrm{c}, \tag{5.3}$$

Finally, the transmission energy in nJ/bit is given by

$$E_\mathrm{b}^{tx} = 10^{S_\mathrm{c}/10} \tag{5.4}$$

Based on this model, the transmission energy consumption when using the LUT-Log-BCJR decoder and the Max-Log-BCJR decoder can be calculated and compared. To investigate the overall energy consumption of the different turbo decoders, the four state-of-the-art LUT-Log-BCJR and Max-Log-BCJR decoder implementations of Table 5.1 and their energy consumptions $E_{\mathrm{b}}^{\mathrm{pr}}$ are used as examples. Their transmission energy consumptions, $E_{\mathrm{b}}^{\mathrm{tx}}$, are estimated using the described model and the simulation results of Figure 5.1. The overall energy consumptions, $E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{pr}}$, associated with using the four different turbo decoder implementations are plotted as a function of transmission range d in Figure 5.2.

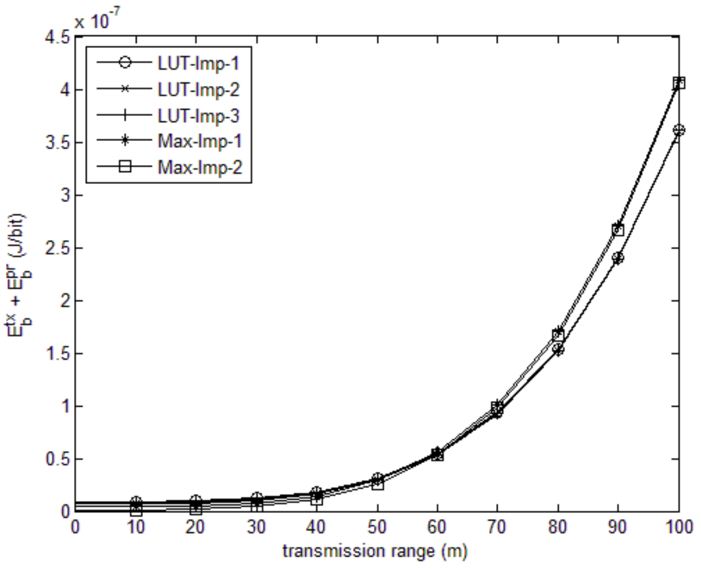


FIGURE 5.2: The comparison of the overall energy consumptions $E_{\mathrm{b}}^{\mathrm{tx}} + E_{\mathrm{b}}^{\mathrm{pr}}$ of the five chosen implementations in Table 5.1.

As shown in Figure 5.2, when transmission distance is relatively low ($d < 20m$), the decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ makes the major contribution to the overall energy consumption. Therefore, there is no significant increase in the overall energy consumption as the transmission range is increased from 0 to 20 m. In these case, the Max-Log-BCJR schemes give the lowest overall energy consumption. However, as the transmission distance is increased above 20 m, the transmission energy consumption becomes the major contributor to the overall energy consumption. When the transmission distance is relatively long ($d > 75m$), the improved coding gain of the LUT-Log-BCJR decoder allows it to use a lower transmission energy, which leads to a lower overall energy consumption. The results in Figure 5.2 show that the LUT-Log-BCJR decoders give at least a 10% reduction in the overall energy consumption when the transmission distance is $d > 100m$. There are critical distances in the range $25m < d < 75m$, where the LUT-Log-BCJR decoders begin to offer lower overall energy consumptions than the Max-Log-BCJR decoders, as shown in Figure 5.3. Note that these results are based on the analysis of the Universal Mobile Telecommunications System (UMTS)/LTE turbo code, which is not specifically designed for energy-constrained applications. As will be demonstrated in Chapter 6, if a turbo code scheme was designed with the consideration of the overall energy consumption, the resulting decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ may be significantly lower than the example considered above. Therefore, the difference in $E_{\mathrm{b}}^{\mathrm{pr}}$ between the LUT-Log-BCJR and Max-Log-BCJR decoders may be smaller in such scheme, which implies that the critical distances discussed above may be even shorter in practice.

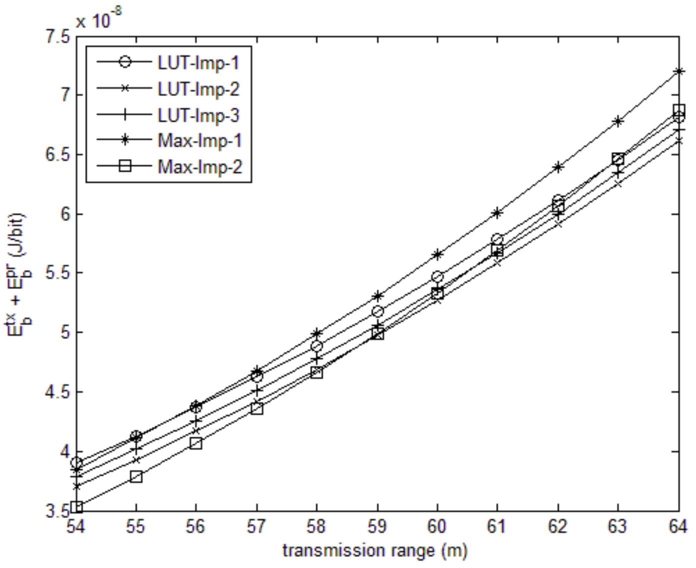


FIGURE 5.3: A zoomed in version of Figure 5.2 at where the overall energy consumptions of the LUT-Log-BCJR decoders become lower than the Max-Log-BCJR decoders.

At the cost of requiring 10% more overall energy consumption, the Max-Log-BCJR decoders have a low computational complexity. The higher computational complexity of the LUT-Log-BCJR algorithm means that they cannot achieve the same high throughputs, which are required by the conventional turbo codes applications, such as the LTE and DVB standards. Therefore, a 10% increase in on overall energy consumption is a reasonable compromise for making the decoders meet the high throughput requirement by the applications. Since turbo codes have previously only found application in high throughput schemes, the literature shows that, since 2003, most of the turbo decoder architecture research has focused on the high throughput Max-Log-BCJR decoder implementations [159–163]. However, for low throughput energy-constrained applications, the avove mentioned compromise is not necessary. As discussed in Chapter 1, low throughput energy-constrained applications, such as environmental WSNs, employ a wide variety of transmission ranges. For most of these possible communication ranges, the LUT-Log-BCJR decoders are more desirable than the Max-Log-BCJR decoder, as shown in Figure 5.2. Figure 5.3 shows that depending on the decoding energy consumption difference between the implementations of the LUT-Log-BCJR decoder and the Max-Log-BCJR decoder, there is a critical distance, where the LUT-Log-BCJR decoder yields a lower overall energy consumption than the Max-Log-BCJR decoder. Furthermore, there are some other variations of the LUT-Log-BCJR decoding algorithm, such as the constant-Log-BCJR [164, 165] and the Enhanced-Max-Log-BCJR [166] algorithms. These variations provide different trade offs between the computational complexity and the BER performance. Based on the principle discussed above, since the LUT-Log-BCJR decoder gives the best BER performance of all the variations, it can be concluded that the LUT-Log-BCJR decoder is the most desirable type of Bahl-Cocke-Jelinek-Raviv (BCJR) decoder for low throughput energy-constrained wireless applications, except for when the transmission range is very short.

5.3 Conventional architecture

The previous section discussed the overall energy efficiency of wireless communication systems using the LUT-Log-BCJR and the Max-Log-BCJR turbo decoders. The LUT-Log-BCJR decoder is more desirable for most communication ranges from the overall energy consumption point of view, owing to its improved BER performance compared with the Max-Log-BCJR decoder. However, for very short communication ranges, the Max-Log-BCJR decoder offers better energy-efficiency since the transmission energy consumption is relatively low and the Max-Log-BCJR decoder tends to have a lower decoding energy consumption, thanks to its low computational complexity. The simulation results show that depending on the difference of the decoding energy consumptions of two types of decoders, there is a critical distance, beyond which the LUT-Log-BCJR decoder becomes more energy-efficient. Therefore, if the decoding energy consumption of the LUT-Log-BCJR decoder can be further reduced, not only can the overall energy consumption can be reduced, but also the critical distance can be shortened. This maximises the benefits of applying turbo codes in wireless communication systems using the LUT-Log-BCJR decoder, particularly when the possible communication range of the target scenario is highly variable from a very short distance to a very long distance. In this section, opportunities to improve the energy efficiency of the LUT-Log-BCJR and the Max-Log-BCJR decoders are discussed. The analysis of the conventional turbo decoder architecture shows that due to its relatively high complexity, the LUT-Log-BCJR decoder's energy consumption can be further reduced significantly by introducing a new architecture.

As described in Section 2.2.1, a turbo encoder [97] comprises a parallel concatenation of two convolutional encoders, each of which has a structure comprising m number of memory elements, as exemplified for $m = 3$ in Figure 5.4, which is the convolutional code that is used in the LTE standard [103]. Each encoder converts an uncoded bit sequence $\mathbf{b}_1 = \{b_{1,j}\}_{j=1}^{N}$ into the corresponding encoded bit sequence $\mathbf{b}_2 = \{b_{2,j}\}_{j=1}^{N}$, where N is the block length of the bit sequences.

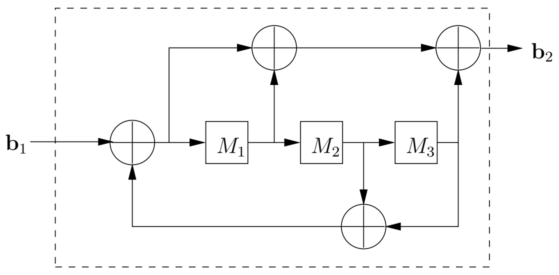


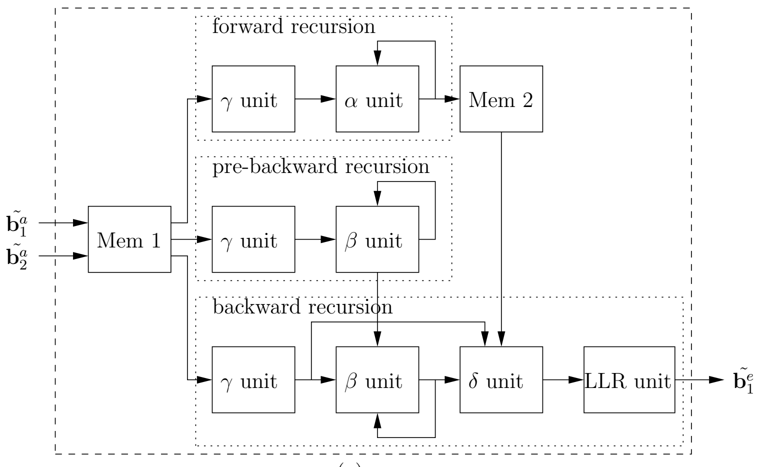
Figure 5.4: The convolutional encoder scheme employed in the UMTS/LTE turbo decoder.

Correspondingly, a turbo decoder [167, 168] comprises a parallel concatenation of two LUT-Log-BCJR decoders. Rather than operating on bits, each LUT-Log-BCJR decoder processes Logarithmic Likelihood Ratios (LLRs) [97], where each LLR $\tilde{b} = \ln \frac{P(b=0)}{P(b=1)}$ quantifies the decoder's confidence concerning its estimate of a bit b from the bit sequence $\mathbf{b}_1$ or $\mathbf{b}_2$. Each LUT-Log-BCJR decoder processes two *a priori* LLR sequences, namely $\tilde{\mathbf{b}}_1^{\mathrm{a}} = \{\tilde{b}_{1,j}^{\mathrm{a}}\}_{j=1}^{N}$ and $\tilde{\mathbf{b}}_2^{\mathrm{a}} = \{\tilde{b}_{2,j}^{\mathrm{a}}\}_{j=1}^{N}$, which are converted into the extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{e}} = \{\tilde{b}_{1,j}^{\mathrm{e}}\}_{j=1}^{N}$. This extrinsic LLR sequence is iteratively exchanged with that generated by the other LUT-Log-BCJR decoder, which is used as the *a priori* LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{a}}$ in the next iteration [118].

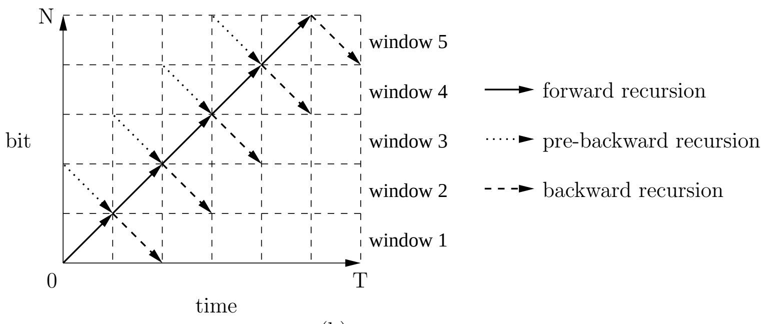
A LUT-Log-BCJR decoder typically employs the sliding-window technique [169] to generate the LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{e}}$ as the concatenation of w_{s} number of equal length subsequences. Each of these windows is generated separately, using a forward, a prebackward and a backward recursion, as shown in Figure 5.5. These three different recursions are performed concurrently for three different windows, as exemplified in Figure 5.5 (b) for $w_{\mathrm{s}} = 5$. This schedule results in the completion of the windows in their natural order, starting with that containing the first LLR $\tilde{b}_{1,1}^{\mathrm{e}}$ and ending with the one containing the last LLR $\tilde{b}_{1,N}^{\mathrm{e}}$.

When the forward recursion is performed for a particular window, one pair of its corresponding *a priori* LLRs $\tilde{b}_{1,j}^{\mathrm{a}}$ and $\tilde{b}_{2,j}^{\mathrm{a}}$ is read from Mem 1 of Figure 5.5 (a) and processed per clock cycle, in the ascending order of the bit index j. The forward recursion of the LUT-Log-BCJR algorithm can be performed in two pipelined steps using the corresponding dedicated hardware components of Figure 5.5 (a). The notations used in this chapter are described in Chapter 2.

1. Firstly, the γ values that correspond to the current window are generated. Here, each γ value $\gamma_i(T)$ corresponding to one transition in the decoding trellis is set either equal to the corresponding *a priori* LLR $\tilde{b}_{i,\mathcal{J}(T)}^{\mathrm{a}}$ or to zero, depending on the particular pair of states that the transition T is between and on the Generator Polynomials (GPs) of the encoder, as discussed in Section 2.2.2.
2. Next, the α values that correspond to the current window are generated. Here, each α value corresponds to a state S in the current step in the decoding trellis.



(a)



(b)

Figure 5.5: (a) The conventional LUT-Log-BCJR architecture. (b) The timing schedule of the sliding-window technique.

The calculation of α is given by

$$\alpha(S) = \overset{*}{\max_{T \in \mathrm{to}(S)}} \epsilon(T) \left(\alpha\big(\mathrm{fr}(T)\big) + \sum_{i=1}^{k+n} \gamma_i(T) \right), \tag{5.5}$$

where, $T \in to(S)$ represents the set of transitions T that end at S, depending on the GPs of the encoder. Note that the forward recursion for the first window is initialised independently. By contrast, the forward recursion for the other windows is initialised using α values that were obtained during the forward recursion of the preceding window. It is for this reason that the windows must be processed in order, as shown in Figure 5.5. The $\max^*$ operation is used to represent the Jacobian logarithm detailed in [170, 171], which is defined for two parameters p and q as

$$\max^*(\tilde{p}, \tilde{q}) = \max(\tilde{p}, \tilde{q}) + \ln[1 + \exp(-|\tilde{p} - \tilde{q}|)] \tag{5.6}$$

and can be extended to three or more parameters using associativity.

One set of α values is written to Mem 2 of figure 5.5 (a) per clock cycle in the ascending order of the bit index j.

When the backward recursion is performed for a particular window, one pair of its corresponding *a priori* LLRs $\tilde{b}^{\mathrm{a}}_{1,j}$ and $\tilde{b}^{\mathrm{a}}_{2,j}$ is read from Mem 1 of Figure 5.5 (a) and processed per clock cycle, in the descending order of the bit index j. Simultaneously, the corresponding set of α values are read from Mem 2 and processed per clock cycle. As a result, a particular window's backwards recursion cannot be performed until after its forward recursion has been completed, as shown in Figure 5.5 (b). The backward recursion of the LUT-Log-BCJR algorithm can be performed in four pipelined steps using the corresponding dedicated hardware components of Figure 5.5 (a).

1. Firstly, the γ values that correspond to the current window are re-generated, as described above.

2. Next, the β values that correspond to the current window are generated. Here, each β value is given by

$$\beta(S) = \overset{*}{\max_{T\in\mathrm{fr}(S)}} \eta(T)\left(\beta\big(\mathrm{to}(T)\big) + \sum_{i=1}^{k+n}\gamma_i(T)\right). \tag{5.7}$$

   Note that the backward recursion for the last window is initialised independently. By contrast, the backward recursions for the other windows are initialised using β values that were previously obtained during the prebackward recursion of the next window. This is achieved using step 1 and 2 of the backward recursion and initialising the latter independently. It is for this reason that the prebackward recursions of Figure 5.5 (b) are performed before the backward recursions of the preceding windows.

3. Next, the δ values that correspond to the current window are generated, according to

$$\delta_i(T) = \alpha\big(\mathrm{fr}(T)\big) + \beta\big(\mathrm{to}(T)\big) + \left(\sum_{i'=1,i'\neq i}^{k+n}\gamma_{i'}(T)\right) \tag{5.8}$$

4. Finally, the value of each extrinsic LLR in the current window of the sequence $\tilde{\mathbf{b}}^{\mathrm{e}}_1$ is generated according to

$$\tilde{b}^{\mathrm{e}}_{1,j} = \overset{*}{\max_{T\left|\substack{\mathcal{B}_1(T)=0\\ \mathcal{J}(T)=j}\right.}}\big(\delta_1(T)\big) - \overset{*}{\max_{T\left|\substack{\mathcal{B}_1(T)=1\\ \mathcal{J}(T)=j}\right.}}\big(\delta_1(T)\big). \tag{5.9}$$

A typical conventional LUT-Log-BCJR architecture is based on Figure 5.5 (a). By implementing each module in the figure individually, the forward, pre-backward and backward recursions are performed by separate dedicated hardware blocks. This architecture is able to generate one extrinsic LLR per clock cycle, by following the processing schedule of Figure 5.5 (b). Moreover, some techniques to improve the throughput based on the architecture have been developed. For example, parallel processing can be used to achieve a very high decoding throughput, by including more than one architecture of Figure 5.5 (a) in the decoder [172]. A radix-4 architecture has been proposed, which increases the throughput of the LUT-Log-BCJR decoder by executing more decoding operations per clock cycle [156]. However, since these techniques aim for improving the throughput and not the energy efficiency, they are not discussed any further in this section.

As described above, the conventional LUT-Log-BCJR architecture executes as many operations as possible in one clock cycle by pipelining the forward, prebackward and backward recursion according to Figure 5.5 (b). To achieve this, the recursions involve calculations that must be performed in series, resulting in a very long critical path in the hardware implementation. As a result, the conventional architecture cannot achieve a high energy efficiency for three reasons. Firstly, the lengthening of the critical path implies a greater variety of data path lengths. The differences between the data path lengths in the circuit can cause significant energy wastage owing to spurious transitions (glitches) [173]. Spurious transitions can account for a significant part of the dynamic power in a ASIC implementation [174]. Reducing spurious transitions requires the lengths of the paths that converge at each gate in the circuit to be roughly equal. Secondly, a long critical path prevents the decoder from using a high clock frequency. To implement the conventional LUT-Log-BCJR architecture at a high clock frequency, it is necessary to employ additional hardware during the synthesis in order to shorten the critical path. This is achieved by employing more complicated functional circuits such as the 'look-ahead adder' to minimise their long critical paths. This increases the size of the datapath, resulting in a higher energy consumption. On the other hand, operating at a lower clock frequency would make the utilisation of the circuit lower. This would cause hardware resources to idle for longer, increasing the static energy consumption. The energy wasted by the static energy consumption becomes more and more critical when the process technology is scaled down [175]. Thirdly, the high complexity of the conventional architecture due to its dedicated hardware for different tasks increases the requirements of the clock tree and the buffers for multiple loaded input signals. Hence, this may become a significant additional consumer of energy in the decoder. In summary, the conventional architecture naturally results in energy wastage, since it is not designed with consideration of the energy efficiency optimisation as a high priority.

5.4 Proposed LUT-Log-BCJR architecture

In this section, a novel LUT-Log-BCJR architecture is proposed for energy-constrained scenarios, which avoids the wastage of energy that is inherent to the conventional architecture of Section 5.3. The philosophy of the proposed architecture is to redesign the timing of the conventional architecture in a manner that allows its components to be efficiently merged. This produces an architecture comprising only a low number of low-complexity functional units, which are collectively capable of performing the entire LUT-Log-BCJR algorithm with a high hardware utility. Further wastage is avoided since the critical paths of the proposed functional units are naturally short- and equally-lengthed, eliminating the requirement for additional hardware to manage them. Furthermore, the proposed approach naturally results in a low area and a high clock frequency, which implies a low static energy consumption.

5.4.1 Decomposition of the LUT-Log-BCJR algorithm into Add-Compare-Select operations

As discussed in Section 2.1, the LUT-Log-BCJR algorithm is attractive in practical hardware implementations because the low dynamic range of its variables allows them to be represented using fixed-point binary numbers, which are associated with a low hardware complexity. It is demonstrated that 9-bit two's complement fixed-point binary numbers including $z = 2$ fraction bits are sufficient to approach the LUT-Log-BCJR decoding performance that is offered by 64-bit floating-point numbers. In addition, it is unnecessary to prevent overflow in fixed-point implementations of the LUT-Log-BCJR because it is inherently resilient to the resultant error. Besides its suitability for fixed-point representation, the LUT-Log-BCJR is attractive in practical implementations of turbo decoders because it only requires calculations that can be decomposed into Add-Compare-Select (ACS) operations, which are associated with a low computational complexity. More specifically, the LUT-Log-BCJR algorithm can be performed entirely on the basis of three calculations, namely addition, subtraction and the max* calculation of Equation 5.10.

Note that, in the case where $z = 2$ fraction bits are used in a fixed-point representation, the right hand term of the max* calculation from Equation (5.6) can be approximated using a Look-Up Table (LUT) comprising $2^z = 4$ entries according to

$$\max{}^*(\tilde{p}, \tilde{q}) = \max(\tilde{p}, \tilde{q}) + \begin{cases} 0.75 & \text{if } |\tilde{p} - \tilde{q}| = 0 \\ 0.5 & \text{if } |\tilde{p} - \tilde{q}| \in \{0.25, 0.5, 0.75\} \\ 0.25 & \text{if } |\tilde{p} - \tilde{q}| \in \{1, 1.25, 1.5, 1.75, 2\} \\ 0 & \text{otherwise} \end{cases}, \tag{5.10}$$

Note that all of the values used in the Look-Up Table (LUT) are multiples of 0.25, since this is the lowest non-zero positive value that can be represented using $z = 2$ fraction bits.

While each addition and subtraction calculation performed in the LUT-Log-BCJR can be considered to be a single ACS operation, each max* calculation can be considered to require four ACS operations:

1. one to simultaneously calculate $\max(\tilde{p}, \tilde{q})$ and $|\tilde{p} - \tilde{q}|$;
2. one to determine if $|\tilde{p} - \tilde{q}| > 0.75$;
3. one to determine if $|\tilde{p} - \tilde{q}| > 0$ or $|\tilde{p} - \tilde{q}| > 2$, depending on the outcome of operation (2);
4. one to add $\max(\tilde{p}, \tilde{q})$ to the value selected from the set $\{0.75, 0.5, 0.25, 0\}$.

Note that in the general case where $z > 0$, a total of $z + 2$ ACS operations are required to complete the max* calculation. The LUT-Log-BCJR algorithm is transformed into the Max-Log-BCJR algorithm in the special case of $z = 0$, where only the first of the above listed ACS operations is required.

5.4.2 Proposed architecture

While the conventional LUT-Log-BCJR architecture of Section 5.3 employs different hardware components for different steps in the LUT-Log-BCJR algorithm, this section proposes an architecture which uses the ACS unit of Section 5.4.3 to perform the whole decoding process. As shown in Figure 5.6, the proposed architecture employs M number of Calculation Units (CUs) in parallel. Each CU includes an ACS unit, three dedicated registers (R1, R2 and R3) and the interconnection structure between them.

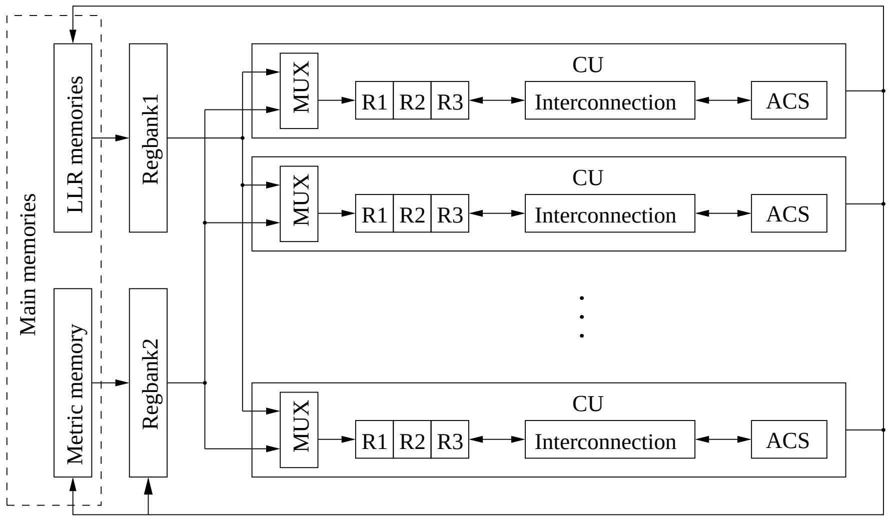


FIGURE 5.6: The proposed LUT-Log-BCJR architecture.

Furthermore, in order to minimise the number of costly accesses to the main memory, which consume a large amount of energy, the proposed architecture additionally employs two register banks, namely Regbank1 and Regbank2. The combined usage of the dedicated registers and the register banks facilitate an efficient max* calculation and allow an entire stage of the trellis to be processed without accessing the main memory.

The register banks, Regbank1 and Regbank2, are used to temporarily store the LUT-Log-BCJR variables from one trellis stage j to the next. This facilitates an entire step of the LUT-Log-BCJR algorithm to be processed without further accesses to the main memory. More specifically, Regbank1 stores the pair of *a priori* LLRs from the *a priori* LLR sequences $\tilde{\mathbf{b}}_1^{\mathrm{a}}$ and $\tilde{\mathbf{b}}_2^{\mathrm{a}}$ that correspond to the currently considered trellis stage j, as well as the seven LUT constants of Equation 5.10. Meanwhile, the α, β and δ values are stored in Regbank2. In addition, some internal results for β calculations during the backward recursion may also be stored in Regbank2. The required number of registers in Regbank2 is 2^m.

While the second register level facilitates the LUT-Log-BCJR processing of one step j of the LUT-Log-BCJR algorithm, the main memory facilitates the processing of the entire algorithm. More specifically, it stores the two *a priori* LLR sequences $\tilde{\mathbf{b}}_1^{\mathrm{a}}$ and $\tilde{\mathbf{b}}_2^{\mathrm{a}}$, as well as the extrinsic LLR sequence $\tilde{\mathbf{b}}_1^{\mathrm{e}}$ that is generated in response. Furthermore, following the completion of the forward recursion, the α values are stored in the main memory for the duration of the backward recursion. As in the efforts of [151, 153, 176, 177], the throughput of the proposed architecture can be significantly increased by decomposing the main memory into $M + 2$ parallel blocks. More specifically, one memory access port is employed for each of the M CUs, as well as two ports for loading the two *a priori* LLR sequences into cache.

Since the proposed architecture supports a fully parallel arrangement of any number of CUs, it can be readily applied to any LUT-Log-BCJR decoder, regardless of the corresponding convolutional encoder parameters that are employed. These parameters include the number of uncoded bit sequence k and non-systematic encoded bit sequences n, the constraint length notation and the generator polynomials notation. Furthermore, the proposed architecture can be readily adapted to support any number z of fraction bits in the two's complement representation, as well as the Max-Log-BCJR algorithm, as discussed in Section 5.4.1. However, a specialised controller is required for any particular LUT-Log-BCJR decoder. Furthermore, the gate count of these controllers are typically higher than those of the conventional architectures. However, it will be shown in Section 5.5 that the controller accounts for less than 10% of the proposed architecture's relatively low gate count. Let us now discuss the parametrisation of the proposed architecture for the implementation of a specific LUT-Log-BCJR decoder, namely that of LTE [103].

5.4.3 Proposed ACS unit and calculation unit

Section 5.4.1 demonstrates that the addition, subtraction and $\max^*$ calculations of the LUT-Log-BCJR algorithm can all be decomposed into individual ACS operations. In order to minimise the number of gates required in the LUT-Log-BCJR decoder, this section proposes the novel ACS unit of Figure 5.7, which can perform one ACS operation per clock cycle. This ACS unit can perform two types of calculation results, namely fixed-point addition or subtraction operations from port $\tilde{r}$ or 1-bit comparison results from the three 1-bit registers $\mathrm{C} = \{\mathrm{C}_0, \mathrm{C}_1, \mathrm{C}_2\}$.

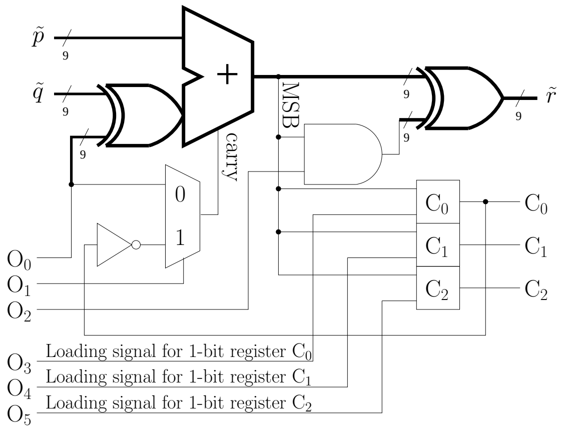


FIGURE 5.7: The proposed ACS unit.

As shown in Figure 5.7, the addition calculations $\tilde{r} = \tilde{p} + \tilde{q}$ of the LUT-Log-BCJR algorithm may be performed in the ACS unit by using the *operation code* $\mathrm{O} = \{\mathrm{O}_0, \mathrm{O}_1, \mathrm{O}_2, \mathrm{O}_3, \mathrm{O}_4, \mathrm{O}_5\} = 000000_2$. Note that a subscript of 2 is applied to the operation codes and register contents in order to indicate that these are binary numbers. By contrast, decimal values are used for all other variables in the following discussion. Meanwhile, the operation code $\mathrm{O} = 100000_2$ may be used to perform the LUT-Log-BCJR algorithm's subtraction calculations $\tilde{r} = \tilde{p} - \tilde{q}$. As shown in Figure 5.7, the variables $\tilde{r}$, $\tilde{p}$ and $\tilde{q}$ have a 9-bit word length, for the reasons discussed in Chapter 4. The procedure of performing a max* operation using the proposed ACS unit is described as follow. Three general purpose registers, R1, R2 and R3, as given in Figure 5.8, are required for providing inputs or storing internal results of the ACS unit during the procedure.

1. In this clock cycle, the max* calculation is begun by using the operation code $\mathrm{O} = 101100_2$ and loading the ACS unit operands $\tilde{p}$ and $\tilde{q}$ from the registers R1 and R2, respectively. The result $\tilde{r}$ is stored in the register R3, according to

$$\mathrm{R3} = \begin{cases} \mathrm{R1} - \mathrm{R2} & \text{if } \mathrm{R1} \geq \mathrm{R2} \\ (\mathrm{R2} - \mathrm{R1}) - \underline{0.25} & \text{if } \mathrm{R1} < \mathrm{R2} \end{cases}. \tag{5.11}$$

   This operation approximates the $|\mathrm{R1} - \mathrm{R2}|$ calculation that is implied by Equation (5.10). Note however, that when $\mathrm{R1} < \mathrm{R2}$, it is the value of $\overline{\mathrm{R1} - \mathrm{R2}}$, which is the one's complement result of $\mathrm{R1} - \mathrm{R2}$, that is calculated and stored in R3. Owing to the two's complement representation, this result is equivalent to decrementing the binary representation of $\mathrm{R2} - \mathrm{R1}$. Since $z = 2$ fraction bits are employed, this decrementation is equivalent to subtracting 0.25, which is the lowest non-zero positive value that can be represented. In order to emphasise that this value of 0.25

is caused by decrementation, it is underlined in Equation (5.11). Note that this inaccuracy affords a simpler ACS unit than would otherwise be required. Besides, it can be trivially cancelled in the following clock cycles.

In order to identify max(R1, R2), the Most Significant Bit (MSB) of the adder's output, is stored in the register $\mathrm{C_0}$. This gives

$$\mathrm{C_0} = \begin{cases} 0_2 & \text{if R1} \geq \text{R2} \\ 1_2 & \text{if R1} < \text{R2} \end{cases} . \tag{5.12}$$

Having approximated |R1 − R2| and determined max(R1, R2), the first ACS operation described in Section 5.4.1 is completed.

2. The LUT comparison that is performed during the second ACS operation is achieved using the operation code $\mathrm{O} = 110010_2$. The value stored in R3 is used for the ACS unit's operand $\tilde{q}$, while $\tilde{p}$ uses the constant decimal value 0.75 obtained from Regbank1, which is the second entry of the LUT described in Equation (5.10). This value is effectively compared with |R1 − R2|, according to

$$\tilde{r} = \begin{cases} 0.75 - \mathrm{R3} & = 0.75 - (\mathrm{R1} - \mathrm{R2}) \quad \text{if } \mathrm{C_0} = 0_2 \\ 0.75 - \mathrm{R3} - \underline{0.25} & = 0.75 - (\mathrm{R2} - \mathrm{R1}) \quad \text{if } \mathrm{C_0} = 1_2 \end{cases} . \tag{5.13}$$

Note that the ACS unit's operation is dictated by the value stored in $\mathrm{C_0}$. In the case where $\mathrm{C_0} = 1_2$, the ACS unit takes the opportunity to cancel the associated decrementation that was introduced in the previous clock cycle. This is achieved by using a carry in of carry $= 0_2$ for the adder, rather than a value of 1_2, as is conventional when performing a subtraction. The result of the LUT comparison is stored in the register $\mathrm{C_1}$ by considering the value of MSB in the adder's output, according to

$$\mathrm{C_1} = \begin{cases} 0_2 & \text{if } \tilde{r} \geq 0 \\ 1_2 & \text{if } \tilde{r} < 0 \end{cases} . \tag{5.14}$$

Following this clock cycle, the register $\mathrm{C_1}$ stores the outcome of the test $|\mathrm{R1}-\mathrm{R2}| > 0.75$, as required by the second ACS operation described in Section 5.4.1.

3. In analogy with the previous clock cycle, the result of the test $|\mathrm{R1} - \mathrm{R2}| > 0$ or the test $|\mathrm{R1} - \mathrm{R2}| > 2$ is determined depending on whether it was previously decided that $|\mathrm{R1} - \mathrm{R2}| > 0.75$. More specifically, $\mathrm{O} = 110001_2$ for the operation code is employed, the value stored in R_3 for the ACS unit's operand $\tilde{q}$ and the constant value 0 or 2 for $\tilde{p}$, as appropriate. As shown in the Equation (5.10), these constant values are the first and third entries of the LUT. The comparison

is achieved according to

$$\tilde{r} = \begin{cases} 0 - \mathrm{R3} & = 0 - (\mathrm{R1} - \mathrm{R2}) \quad \text{if } \mathrm{C}_0 = 0_2, \mathrm{C}_1 = 0_2 \\ 0 - \mathrm{R3} - \underline{0.25} & = 0 - (\mathrm{R2} - \mathrm{R1}) \quad \text{if } \mathrm{C}_0 = 1_2, \mathrm{C}_1 = 0_2 \\ 2 - \mathrm{R3} & = 2 - (\mathrm{R1} - \mathrm{R2}) \quad \text{if } \mathrm{C}_0 = 0_2, \mathrm{C}_1 = 1_2 \\ 2 - \mathrm{R3} - \underline{0.25} & = 2 - (\mathrm{R2} - \mathrm{R1}) \quad \text{if } \mathrm{C}_0 = 1_2, \mathrm{C}_1 = 1_2 \end{cases} . \tag{5.15}$$

The result of the LUT comparison is stored in the register C_2 by considering the value of MSB in the adder's output, according to

$$\mathrm{C}_2 = \begin{cases} 0_2 & \text{if } \tilde{r} \geq 0 \\ 1_2 & \text{if } \tilde{r} < 0 \end{cases} . \tag{5.16}$$

4. The max* calculation is completed in the fourth clock cycle by using the operation code $O = 000000_2$. Here the operand $\tilde{p}$ is provided by the maximum of R1 and R2, as identified by C_0. Meanwhile, a value for the operand $\tilde{q}$ is selected from the set $\{0.75, 0.5, 0.25, 0\}$ that stored in Regbank1, depending on the contents of C_1 and C_2. As a result,

$$\tilde{r} = \max(\mathrm{R1}, \mathrm{R2}) + \begin{cases} 0.75 & \text{if } \mathrm{C}_1 = 1_2, \mathrm{C}_2 = 1_2 \\ 0.5 & \text{if } \mathrm{C}_1 = 1_2, \mathrm{C}_2 = 0_2 \\ 0.25 & \text{if } \mathrm{C}_1 = 0_2, \mathrm{C}_2 = 1_2 \\ 0 & \text{if } \mathrm{C}_1 = 0_2, \mathrm{C}_2 = 0_2 \end{cases} , \tag{5.17}$$

as required.

The ACS unit works with the memory components of Figure 5.6, including the main memory, the register banks and the general purpose registers (R1, R2 and R3). Corresponding structures are required to connect between the ACS unit and the memory components. The CU of Figure 5.8 is designed for this purpose. As described above, the ACS unit can perform the max* calculation of Equation (5.10) in four clock cycles, namely one for each of the ACS operations that are described in Section 5.4.1. In between these clock cycles, three 9-bit registers R1, R2 and R3 are used to store intermediate results. The ACS unit takes input from two databuses, 'databus1' and 'databus2', as shown in Figure 5.8. The output 'res' gives the calculation results, which can be either fed back to the general purpose registers (R1, R2 and R3) or output to the main memory through 'data_out'. The output 'C' gives the 1-bit comparison results, which can be either transmitted to the multiplexer controlled by 'C_0' for determining the result of a max operation or transmitted to the LUT in Regbank1 to determine the correction value in a max* operation. As shown in Figure 5.8, the inputs from 'databus1' and 'databus2' are highly flexible. A group of tri-state buffers provides different possible

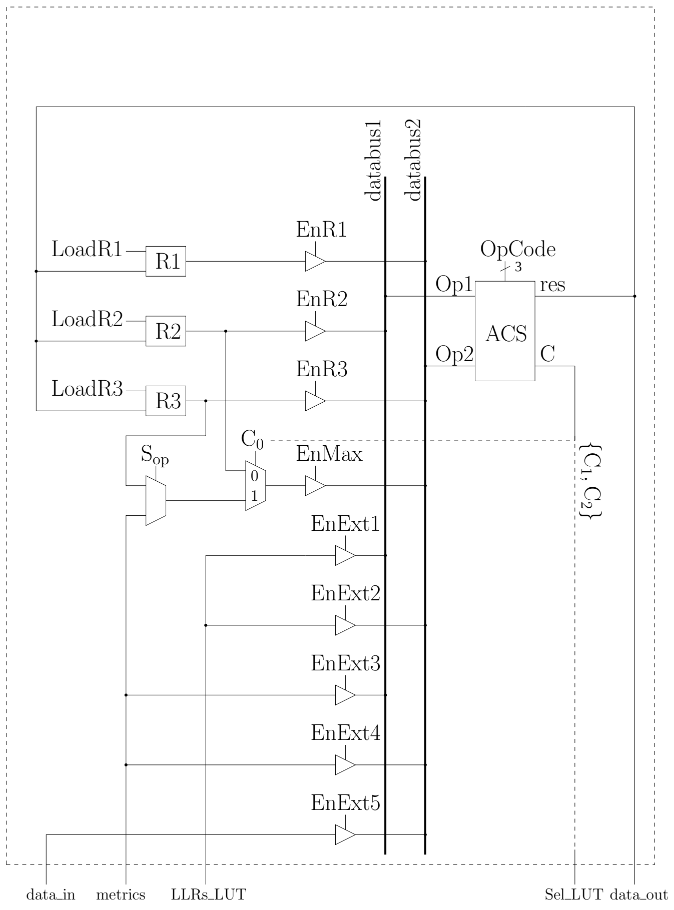


FIGURE 5.8: the proposed CU.

combinations of the inputs from all the memory components in the architecture. This allows the CU to provide the required operands from any memory components to the ACS unit. This avoids additional clock cycles wasteful for moving the data between differen memory components.

5.4.4 Optimal number of parallel calculation units

Intuitively, it may seem that the energy efficiency of the proposed architecture is independent of the number of parallel CUs M that it employs. For example, it may seem that doubling M would double the power consumption, but halve the total processing time, maintaining a constant energy efficiency. However, the situation is complicated by the data dependencies of the LUT-Log-BCJR algorithm. More specifically, a high number of parallel CUs facilitates the just-in-time calculation of the LUT-Log-BCJR variables, reducing the amount of storage that they require. However, the different steps of the LUT-Log-BCJR algorithm are suited to different amounts of parallelism. This is because each stage of the trellis is associated with 2^m α and β values, but 2^{m+1} δ values and only a single extrinsic LLR. Therefore, a high number of parallel CUs cannot be exploited at all times. This reduced hardware activity diminishes the energy efficiency owing to the static power that is consumed by inactive gates. Therefore, there exists a particular number of parallel CUs which optimises this trade-off, maximising the energy efficiency.

Since the number of the transitions in each stage of the trellis is 2^{m+1}, to equally distribute the calculations of γ, α, β and δ, the candidate values for M are $\{1, 2, 2^2, \ldots, 2^m\}$. In order to determine the optimal number M of parallel CUs, the components of the proposed architecture, including CUs, Regank1 and Regbank2, were implemented using the Taiwan Semiconductor Manufacturing Company (TSMC) 90 nm process technology. The areas a required for the final implementation of the components, which reflect their hardware complexity, were obtained by post-layout simulation. For a particular M, based on the operation distributions in the CUs, the time consumption of decoding one extrinsic LLR and their hardware complexity, the hardware utilisation efficiency can be quantified.

For example, when $M = 2^m$ in the LTE turbo code, the operation distribution can be presented by the timing diagram of Figure 5.9. Note that addition and subtraction calculations can be performed in one clock cycle, while the max* calculations require four clock cycles, as described in Section 5.4.3. As described in Section 5.3, the proposed

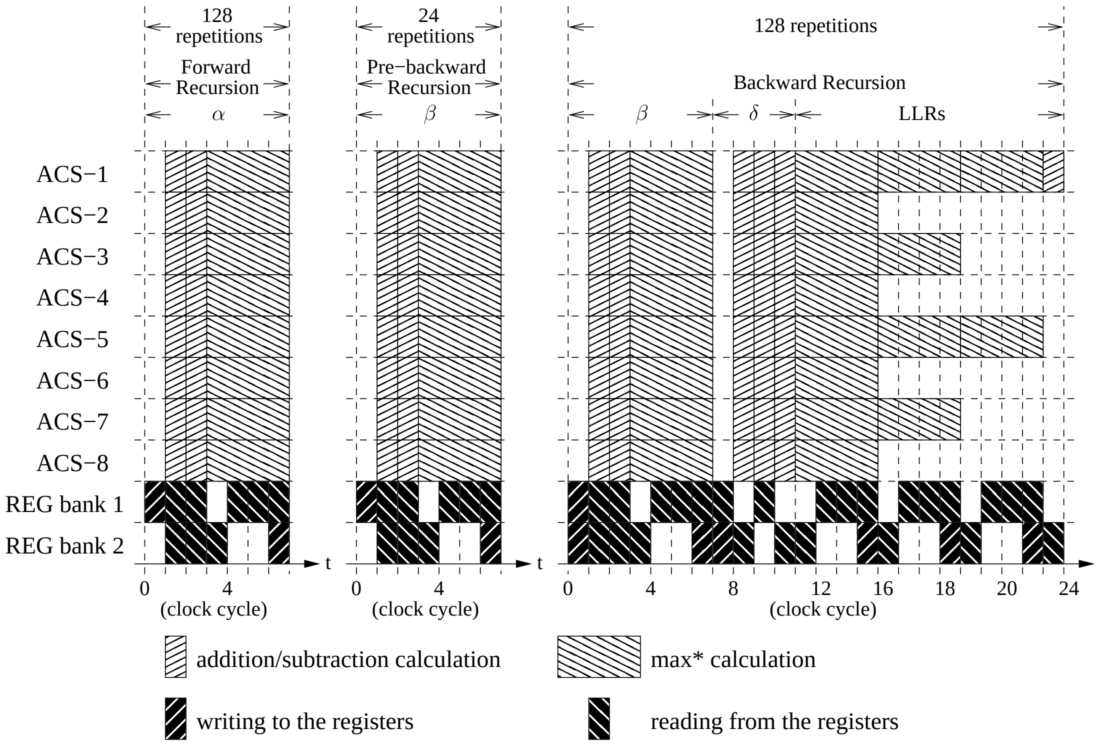


FIGURE 5.9: The timing diagram for the operation distribution in the LTE decoder's components, when $M = 2^m$.

architecture implements the sliding-window LUT-Log-BCJR decoding algorithm by repeatedly cycling through three different recursions, namely the forward, pre-backward and backward recursions. These are applied in turn to $\lfloor N/128 \rfloor$ number of windows comprising 128 trellis stages, as well as to a final window comprising the remainder of the N trellis stages. As shown in Figure 5.9, the forward recursion determines the set of α values for each of the 128 trellis stages in the current window using seven clock cycles. More specifically, the $M = 8$ CUs are employed to calculate the $2^m = 8$ α values in parallel, with the assistance of Regank1 and Regbank2. Similarly, the pre-backward recursion uses seven clock cycles to determine the set of β values for each of the 24 trellis stages beyond the right-hand edge of the current window. Finally, the backward recursion requires 24 clock cycles to calculate the β, δ and extrinsic LLR values for each of the 128 trellis stages in the current window, as shown in Figure 5.9. On average, it can be calculated that 32.3 clock cycles are required to calculate each extrinsic LLR. Therefore, the time consumption c in clock cycles per bit can be obtained using the timing diagram of Figure 5.9. Moreover, the timing diagram shows for what fraction of time r each of the components is operational. The hardware utilisation rate u of the decoder is defined as the average of the r values, weighted by the a values of each component in the decoder, according to

$$u = \frac{\sum r \times a}{\sum a}. \tag{5.18}$$

In addition, the total datapath area can be estimated as $A = \sum a$.

To evaluate the hardware utilisation efficiency, the metric $u \times c \times A$ is proposed to quantify the hardware resources that are actually used to decode each extrinsic LLR. A lower value of $u \times c \times A$ represents a decoder using less hardware resources to decode one extrinsic LLR. For the LTE turbo code, the metric values corresponding to different values of M are given in Table 5.3.

M	$2^{m-2} = 2$	$2^{m-1} = 4$	$2^m = 8$
c	106.3	55.6	32.3
A	5564	8550	13377
u	75.21%	77.56%	71.08%
$u \times c \times A$	4.45×10^5	3.69×10^5	3.07×10^5

Table 5.3: Datapath characteristics for various numbers M of parallel CUs.

In the case where fewer than 2^m parallel CUs are employed, the calculations of Figure 5.9 are rearranged into fewer rows. This has the benefit of increasing the hardware activity during the extrinsic LLR calculations, as shown in Table 5.3. However, Table 5.3 shows that a greater number of clock cycles per LLR c are required. Furthermore, just-in-time processing is prevented, requiring Regbank2 to be enlarged in order to store additional intermediate variables, as shown in Table 5.3. Table 5.3 shows that $u \times c \times A$ is minimised for $M = 2^m$. For this reason, in Section 5.5, $M = 8$ is adopted in order to implement an energy-efficient LUT-Log-BCJR decoder for the turbo code of LTE.

5.5 Turbo decoder complexity and energy analysis

To analyse the complexity and the energy efficiency of the proposed architecture, an LTE turbo decoder was implemented to the post-layout stage using TSMC 90 nm technology. The turbo decoder comprises four parts, namely a LUT-Log-BCJR decoder, an interleaver, a controller and the memory. The interleaver was implemented according to the latest low-complexity LTE interleaver designs [178, 179]. The memory employs one 128×64-bit on-chip single-port Static Random-Access Memory (SRAM) module for storing the α values. Similarly, it employs five 6144×6-bit on-chip single-port SRAM modules for storing the two sets of *a priori* LLRs, the two sets of extrinsic LLRs and the single set of systematic LLRs. The layout of the decoder is provided in Figure 5.10. As shown in Figure 5.10, the hardware complexity of the proposed architecture is so low

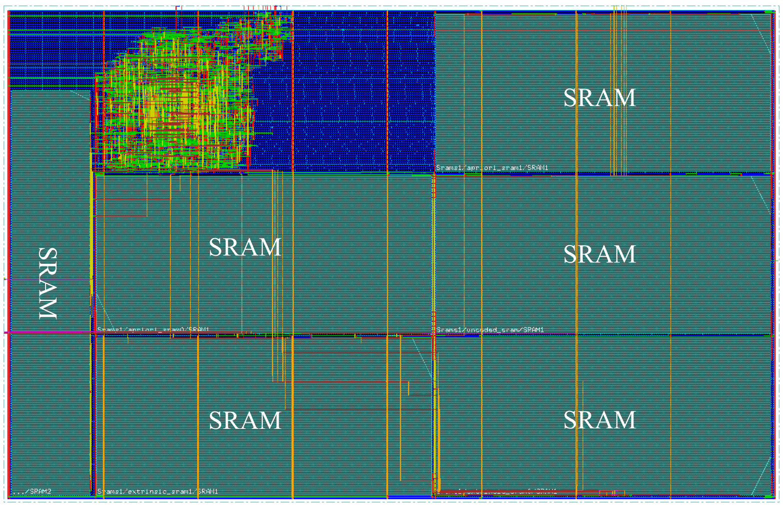


FIGURE 5.10: Chip layout of the decoder.

that the chip area is actually mostly limited by the memory modules. In contrast, in conventional architectures, the major portion of the chip area is occupied by the datapath, even though the conventional architectures normally employ at least as much memory as the proposed architecture. For energy constrained WSNs, it may be desirable to employ block lengths for the turbo code that are shorter than those of the LTE standard, since a short packet length is usually required in these applications. In these case, significant reduction of the proposed architecture's chip area can be expected owing to the reduced memory requirement. Table 5.4 compares the proposed architecture with the latest LUT-Log-BCJR and Max-Log-BCJR decoder architectures. The area and energy consumptions are estimated based on post-layout simulation. The implementation results arising from different technologies are also scaled[1] to give a fair comparison. As shown in Table 5.4, the energy consumption of the proposed architecture is significantly lower than that of the conventional LUT-Log-BCJR architectures and is comparable to the

[1]The energy consumption and area are adjusted using scaling factors of $1/s^3$ and $1/s^2$ respectively, where s is the ratio of the old technology scale to the new one [160].

TABLE 5.4: Comparison of the implemented turbo decoders.

Publication	Proposed	[154]	[156]	[158]	[159]	[160]
Algorithm	LUT-Log	LUT-Log	LUT-Log	LUT-Log	Max-Log	Max-Log
Block size (bit)	6144	5114	5114	5114	6144	6144
Technology (nm)	90	180	180	180	65	120
Supply voltage (V)	1.0	1.8	1.8	1.8	-	1.2
Area A (mm^2)	0.35	9	14.5	8.2	2.1	3.57
(Scaled for 90 nm)		(2.25)	(3.63)	(2.05)	(4.0)	(2.0)
Gate count (exclusive of memory)	7.5k	85k	410k	65k	-	553k
Memory required (Kb)	188	239	450	161	-	129
Clock frequency F (MHz)	333	111	145	100	300	390.6
Decoding iterations	5	10	8	6.5	6	5.5
Throughput T (Mb/s)	1.03	2	10.8	4.17	150	390.6
Power consumption (mW)	4.17	292	956	320	300	788.9
(Scaled for 90 nm)		(36.5)	(119.4)	(40)	(796.4)	(332.8)
Energy consumption (nJ/bit/iteration)	0.4	14.6	11.1	12.7	0.31	0.37
(Scaled for 90 nm)		(1.8)	(1.4)	(1.59)	(0.81)	(0.16)
$E_b^{tx}+E_b^{pr}$ (nJ/bit) when transmitting over 39 m (5 iterations)	10.16	17.16	15.16	16.06	13.42	10.17
$E_b^{tx}+E_b^{pr}$ (nJ/bit) when transmitting over 58 m (5 iterations)	41.92	48.92	46.92	47.82	49.88	46.63
$E_b^{tx}+E_b^{pr}$ (nJ/bit) when transmitting over 100 m (5 iterations)	354.7	361.7	359.7	360.6	409.0	405.8

recent Max-Log-BCJR decoders. In addition, because the proposed decoder's datapath is simple and has a low energy consumption, the simulation results show that 40% of the energy consumption of the implemented decoder can be attributed to the memory. Considering that energy-constrained WSNs require shorter packet lengths than the LTE standard, the energy consumption of the proposed architecture could be further reduced in these applications where less memory is required.

To analyse the overall energy consumption $E_b^{tx} + E_b^{pr}$ of the LUT-Log-BCJR and the Max-Log-BCJR decoders, the BER performance of the proposed architecture and the ideal performance of the two types of the decoders are quantified in Figure 5.11[2]. As

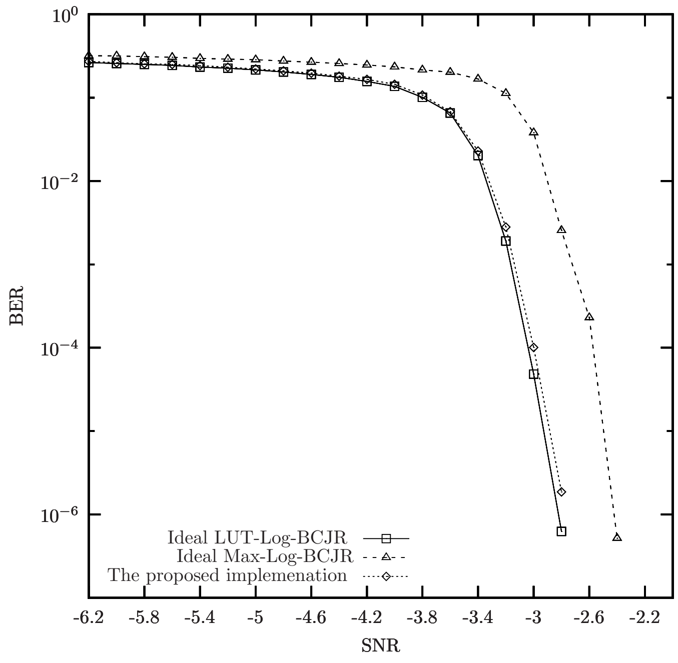


FIGURE 5.11: BER simulation results, in the case where 5 iterations are employed to decode a 6144-bit LTE block, which was transmitted over an uncorrelated Rayleigh fading channel.

discussed in Section 5.2, the high energy efficiency of the Max-Log-BCJR is achieved at the cost of requiring a 0.5 dB higher transmission energy per bit to achieve a BER of 10^{-4}, as shown in Figure 5.11. As a result, the LUT-Log-BCJR algorithm facilitates an *overall* energy consumption - including the energy consumed during both transmission and decoding - that is 10% lower than that of the Max-Log-BCJR at long transmission ranges, where the energy consumption of the turbo decoder is negligible compared to the transmission energy required. Indeed, the analysis in [1,2] reveals that a small difference in BER performance has a significant effect on the overall energy consumption $E_b^{tx}+E_b^{pr}$. Some estimation results of the overall energy consumptions $E_b^{tx} + E_b^{pr}$ for each decoder in Table 5.4 are provided for a variety of transmission distances. Figure 5.12 shows the difference in overall energy consumption between the proposed architecture and the

[2]Since different simulation parameters and channel models are used in previous publications, the BER performance of the proposed architecture is compared with the idealised upper-bound performance of the various algorithms, which was obtained using floating-point simulation.

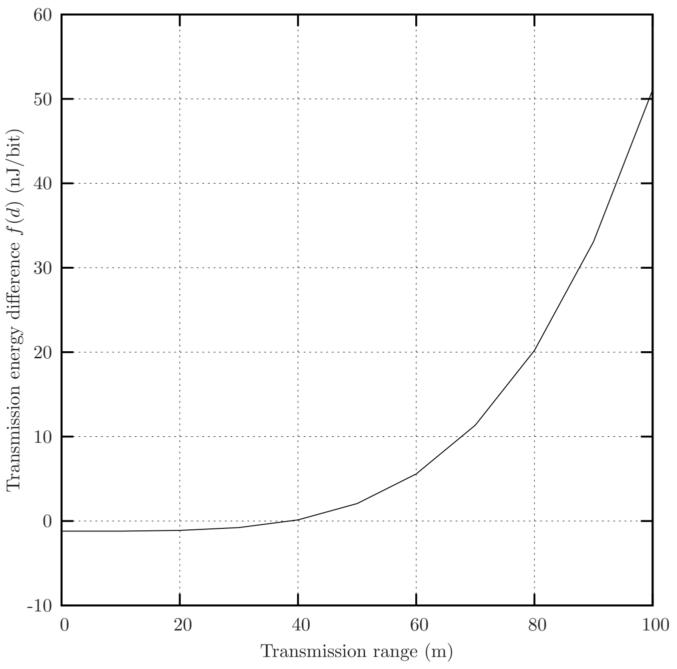


FIGURE 5.12: The energy consumption difference between the proposed architecture and the Max-Log-BCJR decoder in [160] at BER = 10^{-4}.

Max-Log-BCJR of [160], $f(d) = (E_b^{tx} + E_b^{pr})_{LUT-Log-BCJR} - (E_b^{tx} + E_b^{pr})_{Max-Log-BCJR}$. As shown in Figure 5.12, when the function $f(d)$ has a negative value, the Max-Log-BCJR decoder offers a greater energy efficiency than the proposed architecture. By contrast, when $f(d)$ has a positive value, the proposed architecture offers a greater energy-efficient than the Max-Log-BCJR decoder. The magnitude of the value is the difference in overall energy consumption between the two schemes. As a result, the proposed architecture offers an overall energy saving that increases exponentially beyond a range of 39 m, relative to the state-of-the-art Max-Log-BCJR decoder [160]. Further calculation shows that, compared with the most energy-efficient design [160] of Table 5.4, which has an energy consumption of 0.16 nJ/bit/iteration, the proposed LUT-Log-BCJR decoder achieves more than 10% overall energy saving when the transmission distance reaches 58 m.

5.6 Conclusions

This chapter commenced by investigating the potential energy saving that can be offered by employing turbo codes in WSNs. Since turbo decoders have a relatively high complexity, their energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ must be considered in these energy-constrained scenarios. The saving in transmission energy $E_{\mathrm{b}}^{\mathrm{tx}}$ that is facilitated by turbo codes must overcome $E_{\mathrm{b}}^{\mathrm{pr}}$ in order to achieve an overall energy saving. As a result, overall energy consumption $E_{\mathrm{b}}^{\mathrm{pr}} + E_{\mathrm{b}}^{\mathrm{tx}}$ is proposed to evaluate the energy efficiency of a turbo code which is employed in a WSN. An energy-efficient turbo decoder is essential for minimising $E_\mathrm{b}^\mathrm{pr} + E_\mathrm{b}^\mathrm{tx}$, when employing turbo codes in WSNs.

The evaluation of $E_\mathrm{b}^\mathrm{pr}+E_\mathrm{b}^\mathrm{tx}$ is applied to five state-of-the-art ASIC turbo decoder designs, employing either the LUT-Log-BCJR or the Max-Log-BCJR algorithms. The study reveals that Max-Log-BCJR turbo decoders typically have lower E_b^pr than LUT-Log-BCJR turbo decoders because of their lower computational complexity. However, LUT-Log-BCJR decoders facilitate 10% lower E_b^tx than Max-Log-BCJR decoders because they achieve higher coding gains. As a result, when the transmission distance is sufficiently short, E_b^pr makes a greater contribution than E_b^tx to the overall energy consumption and Max-Log-BCJR decoders are more energy efficient than LUT-Log-BCJR decoders. However, for any pair of LUT-Log-BCJR and Max-Log-BCJR decoders, there is a critical transmission distance at which E_b^pr is sufficiently high and the LUT-Log-BCJR decoder becomes beneficial for the overall energy efficiency. Therefore, the key issue for designing LUT-Log-BCJR decoders is making them more energy efficient over a wider transmission range in order to achieve a low E_b^pr for the ASIC implementation.

For the purpose of designing a hardware architecture for LUT-Log-BCJR turbo decoders having a low E_b^pr, this chapter has also analysed the energy efficiency of conventional turbo decoder architectures. Based on the analysis, a low-complexity energy-efficient turbo decoder architecture was proposed specifically for WSNs. The LUT-Log-BCJR algorithm is decomposed into its fundamental operations, namely the ACS operations. A novel ACS unit is designed for performing all the operations during the decoding process. A datapath architecture is proposed to support a fully parallel arrangement of any number of CUs, with a low hardware complexity. Finally, the proposed architecture is demonstrated and validated by implementing the LTE turbo decoder. The implementation results show that the proposed architecture provides an overall energy saving compared to both the conventional LUT-Log-BCJR and Max-Log-BCJR decoder architectures, when the transmission range exceeds 39 m. Moreover, the proposed architecture is highly flexible and can be readily applied to any turbo code.

Chapter 6

A turbo decoder energy estimation model allowing overall energy optimisation

6.1 Introduction

As discussed in Chapter 1, turbo codes can benefit in Wireless Sensor Networks (WSNs) by reducing their transmission energy consumption, since the sensor nodes in WSNs typically require a relatively long lifetime and have constrained energy resources. The investigation of Chapter 3 concludes that employing a Serial Concatenated Convolutional Code (SCCC) in a star topology WSN can significantly reduce the sensor nodes' energy consumption, and hence increase their lifetime. In this case, the only extra device required on the sensor nodes is the SCCC encoder which uses forward error correction to allow a significant transmission power reduction. As discussed in Chapter 1, this is applicable in WSNs where information only flows in one direction, from the sensor nodes to the central node. Since the encoder of a SCCC is very low complexity, it consumes only a negligible amount of energy compared with the energy saving associated with the transmission power reduction provided by the employed SCCC. The trade-off is the high complexity SCCC decoders that are required on the central node, which have a relatively large energy consumption. Since WSN typically have only one or a few central nodes having abundant energy resources, the described trade-off is desirable.

However, in more complicated network topologies, multi-hop transmission is typically employed [180] and the sensor nodes need to receive and retransmit the signals. In decode-and-forward schemes [132], the high complexity decoders are required on the sensor nodes and will consume extra energy. As a result, whether or not the employed turbo code can still provide energy saving for the sensor nodes depends on whether or not the transmission energy saving can overcome the decoder's energy consumption.

The investigation in Chapter 5 shows that when the communication distance is long enough between two nodes in the WSN, the transmission energy consumption becomes dominant compared with the energy consumed by the decoders integrated into the sensor nodes. Therefore, the situation becomes similar to that of the star network, since both the encoder and the decoder's energy consumption is negligible compared with the transmission energy saving granted by the turbo code. As a result, the overall energy consumption of the sensor nodes can be reduced and the lifetime of the WSN can be extended. However, this condition limits the applicable minimum transmission range when adopting a turbo-like code in multi-hop WSNs.

The energy consumed by the turbo-like decoder is an important factor that affects the minimum transmission range for which an overall energy reduction can be achieved. As discussed in Chapter 5, there are two significant parts of energy consumption in wireless communication systems that relate to the use of channel coding schemes, namely the transmission energy saving $\Delta E_{\mathrm{b}}^{\mathrm{tx}}$ and the decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$. Therefore, the overall energy saving of the sensor nodes can be considered to be $\Delta E_{\mathrm{b}}^{\mathrm{tx}} - E_{\mathrm{b}}^{\mathrm{pr}}$. Therefore, to design a turbo code for WSNs, both $\Delta E_{\mathrm{b}}^{\mathrm{tx}}$ and $E_{\mathrm{b}}^{\mathrm{pr}}$ are very important specifications that need to be considered carefully. The transmission energy saving $\Delta E_{\mathrm{b}}^{\mathrm{tx}}$ can be estimated by using Bit Error Rate (BER) analysis and a relevant path loss model, as demonstrated in Chapter 5.

However, it is difficult to estimate the decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ during the code design stage. Therefore, conventionally, it is not considered in the design process [91]. Instead, the turbo code is designed using simulation (BER, EXtrinsic Information Transfer (EXIT) simulation) to meet a particular performance requirement, such as a particular coding gain with a particular computation complexity. Then the hardware implementation is designed to minimise the energy consumption under the constraint of achieving the required decoding throughput, cost, area, etc. However, this design procedure is not suitable for turbo codes designed to be applied in WSNs. By only considering the performance in the code design stage, the energy consumption of the coding system is not holistically optimised. The design result may achieve a great reduction on transmission energy consumption, $\Delta E_{\mathrm{b}}^{\mathrm{tx}}$, but also impose a high decoding energy consumption, $E_{\mathrm{b}}^{\mathrm{pr}}$, which cancels out the energy saving overall. If the decoding energy consumption, $E_{\mathrm{b}}^{\mathrm{pr}}$, can be considered from the start of the design process, the disadvantage of the conventional design method can be prevented. Therefore, a methodology providing the capability to estimate the decoding energy consumption from the start is required.

Energy estimation is a key issue for Application-Specific Integrated Circuit (ASIC) design procedures. Methodologies and tools can be developed to perform energy estimation at various stages of the design procedure. A typical ASIC design procedure is shown in Figure 6.1. Typically, the later the design stage in which the energy estimation is performed, the higher the accuracy that can be gained, because more information about the hardware implementation is available [181]. Therefore, conventionally, the energy

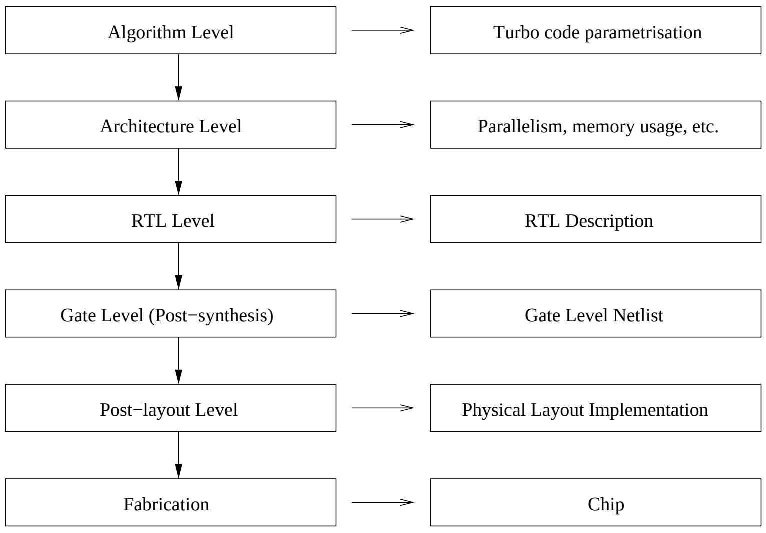

Algorithm Level
Turbo code parametrisation
Architecture Level
Parallelism, memory usage, etc.
RTL Level
RTL Description
Gate Level (Post−synthesis)
Gate Level Netlist
Post−layout Level
Physical Layout Implementation
Fabrication
Chip

FIGURE 6.1: The typical ASIC design procedure and the outcome of each stage.

estimation at the post-synthesis level or post-layout level is widely used in practice due to its accuracy and generality [182]. However, there is less opportunity to rethink the design at these design stages. Previously, [183,184] proposed energy estimation methods based on the computation complexity of a design, such as the number and the type of arithmetic/Boolean operations in the behavioural description and the number of states and/or transitions in a controller description. These complexity based models rely on the assumption that the complexity of a circuit can be estimated using the number of equivalent gates, which limits the accuracy of the estimation. This is because even with the same number of equivalent gates, different compositions and behaviours of the gates has a significant effect on the energy consumption [185,186]. In addition, these methods require additional analysis of the target design, in order to convert the information from the algorithm level to hardware complexity. However, this analysis is not required in the conventional design procedure and so this approach is inconvenient [187]. In this chapter, the architecture proposed in Chapter 5 is used to derive a bottom-up energy estimation framework. Using a detailed investigation of the energy consumed by the sub-modules in the proposed architecture following post-layout simulations, energy models are produced based on the measurements. These energy models of the sub-modules can be used during the early design stage, in order to estimate the energy consumption of a turbo decoder without requiring knowledge from later design stages. The motivation for using a bottom-up energy estimation framework based on the proposed architecture is because its high configurability allows the characteristics of the hardware implementation to be predicted at an early design stage. The object of this framework is to provide energy consumption information for the turbo code designer, in order to help determine the parameters of the codes. In particular, the framework allows the designer compare the energy consumption of different turbo code designs.

The remaining sections of this chapter are organised as follows. Section 6.2 presents a generalisation of the architecture proposed in Section 5. In particular, a redesigned controller is proposed in order to allow the architecture to be readily reconfigured for different sets of turbo code parameters. In Section 6.3, the above-mentioned energy estimation framework is proposed. In Section 6.4, a holistic design procedure is demonstrated which uses the proposed estimation framework and a path loss model to optimise the selection of turbo code parameters for the scheme of [106]. In [106], different turbo code parameterisations are investigated using the conventional BER evaluation method. The demonstration of this section shows that the proposed holistic design procedure gives an overall optimisation for the turbo code design, which the conventional design procedure cannot provide. The conclusions are provided in Section 6.5.

6.2 Generalised architecture

As described in Section 5.4, the proposed architecture can be reconfigured to implement any Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) decoder. This is because the proposed architecture is based on five sub-modules, where different decoder designs employ different combinations of the sub-modules. The top-level of the architecture is shown in Figure 6.2[1]. There are five sub-modules in the top-level, namely the Calculation Units (CUs), two register banks (Regbank1 and Regbank2), the Logarithmic Likelihood Ratio (LLR) memories and the metric memory. The

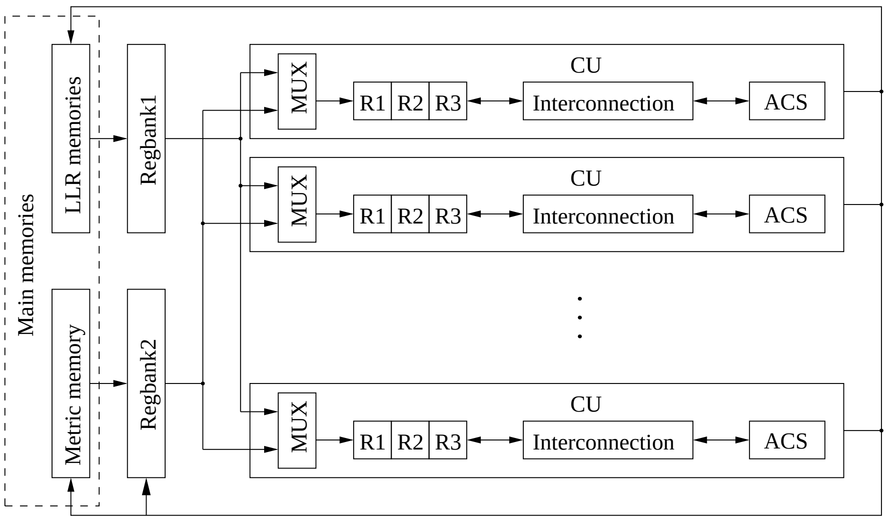


FIGURE 6.2: The configuration of the proposed LUT-Log-BCJR decoder architecture.

generalisation of the architecture proposed in this section requires no change to the architecture shown in Figure 6.2. However, the generalisation includes a redesigned usage of Regbank2 and a redesigned controller. Furthermore, for different LUT-Log-BCJR decoders, the reconfiguration of the proposed architecture may adjust the number of Add-Compare-Select (ACS) units employed in parallel, the number of registers in Regbank1

[1]This figure is also provided in Section 5.4, Figure 5.6.

and Regbank2 and the redesign of the controller to correctly schedule the operations that take place in the ACS units and the register banks.

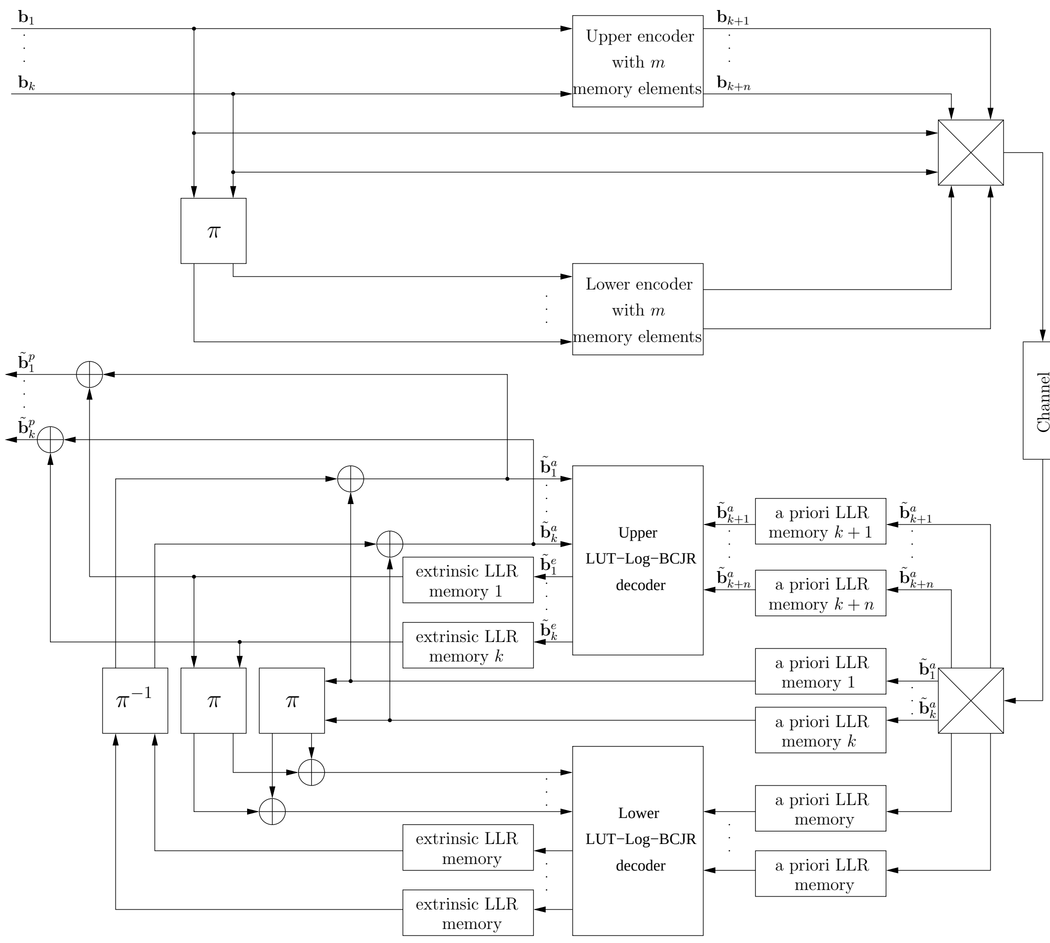


FIGURE 6.3: The configuration of a typical turbo code scheme.

The configuration of the generalised architecture depends on the parametrisation of the turbo code design. As shown in Figure 6.3, the parameters of a turbo code includes the number of input sequences k provided to each component encoder, the number of the memory elements m employed in each component encoder and the number of non-systematic output sequences n provided by each component encoder. Note that in Figure 6.3 each of the k input sequences corresponds to one of the k systematic output sequences. Moreover, since the sliding-window technique [169] is recommended as described in Section 5.2, the sliding-window length w_s and the pre-backward recursion length w_p are parameters of the specification. In the following sub-sections, each sub-module of Figure 6.2 is discussed individually. In Section 6.2.4, a controller design that is more appropriate for general cases is proposed. The study shows that the configuration of the turbo decoder implementation can be determined in the earliest turbo code design stage.

6.2.1 Register banks

There are two register banks, Regbank1 and Regbank2, in the architecture, as shown in Figure 6.2. Regbank1 is composed of both registers and dummy registers. The dummy registers have hard-wired constant values, which correspond to the Look-Up Table (LUT) elements. They are considered as registers in the architecture level, in order to give a concise data organisation and facilitate a simpler controller design for the decoder. Regbank1 provides four types of input for the CUs, namely the a priori LLRs, the index values of the LUT, the output values of the LUT and a constant zero value. The formation of Regbank1 is given in Figure 6.4. In the figure, $\{\tilde{b}^{\mathrm{a}}_{1,j},\ \tilde{b}^{\mathrm{a}}_{2,j},\ \ldots\ \tilde{b}^{\mathrm{a}}_{k+n,j}\}$

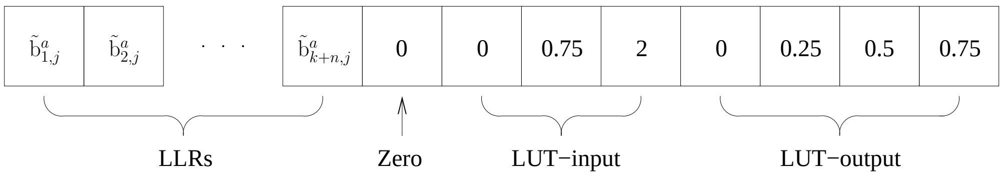


FIGURE 6.4: The components of Regbank1.

are the a priori LLRs of the current decoding step j of each a priori LLR sequences, $\{\tilde{\mathbf{b}}^{\mathrm{a}}_1,\ \tilde{\mathbf{b}}^{\mathrm{a}}_2,\ \ldots\ \tilde{\mathbf{b}}^{\mathrm{a}}_{k+n}\}$, as shown in Figure 6.3, where $\tilde{\mathbf{b}}^{\mathrm{a}}_i = \{\tilde{\mathbf{b}}^{\mathrm{a}}_{i,j}\}_{j=1}^{N}$. Here, N is the block length, namely, the number of LLRs in each sequence. As shown in Figure 6.4, only the $k+n$ number of LLRs require actual registers in the hardware design. The others are all dummy registers. Note that some of the dummy registers store the same values, but are kept separated in order to facilitate a concise controller design.

In Chapter 5, the turbo decoder was designed for a special case, where each component encoder has $k = 1$ input and $n = 1$ non-systematic output sequences. Depending on which part of the LUT-Log-BCJR algorithm is being processed, Regbank2 is used to store different internal metrics. During the forward recursion, Regbank2 stores the set of $\alpha(S)$ values (Equation 5.5) that are calculated for a particular step j in the trellis. Similarly, a set of $\beta(S)$ (Equation 5.7) values is stored during the pre-backward and backward recursions. On other occasions, Regbank2 stores some internal calculation results, as discussed in Section 5.4.2. The proposed decoder architecture requires Regbank2 to store a set of metrics corresponding to one step j in the decoding trellis at a time. Moreover, in this chapter, the proposed architecture is generalised for any LUT-Log-BCJR decoder. Because Regbank2 is also used for storing the internal results for the $\delta_i(T)$ calculations (Equation 5.8), these internal results define the register requirement in Regbank2 in the general case where $k \geq 1$ or $n \geq 1$. The number of registers required in Regbank2 is given by $2^m(2^k - 1)$, as discussed in Section 6.2.4.

6.2.2 Calculation unit

The CUs perform all the ACS operations of the Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) decoding process. Each CU consists of a ACS unit, three general purpose registers, the interconnections between them and the Multiplexor Unit (MUX) unit for providing the input values from the register banks. The connections between the general purpose registers and the ACS unit is given in Figure 6.5[2]. The function of the ACS

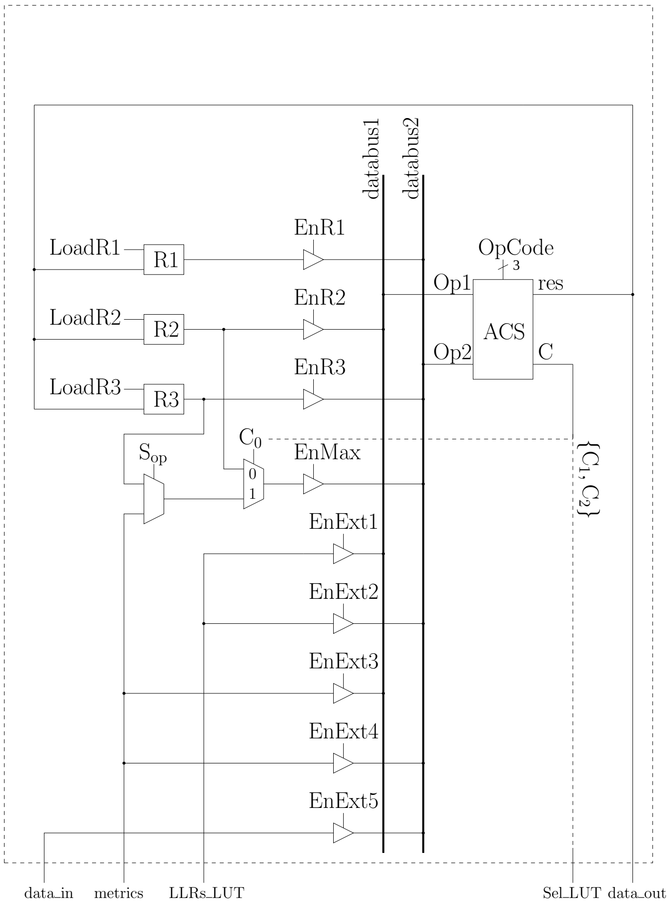


FIGURE 6.5: CU structure.

unit is to perform the fundamental ACS operations, including addition/subtraction, max* and selection, as described in Section 5.4.3. As discussed in Section 5.4.4, the optimal number of CUs to employ in the decoder, is 2^m, where m is the number of the memory elements in the component encoder. In the general case, the only modification required is to place the correct number of CUs in parallel. There is no need to reconfigure the internal structure of the CUs.

As shown in Figure 6.2, the MUXs are used to connect the register banks and the CUs. They select the input for the CUs from the values stored in the register banks. The required number of MUXs is equal to 2^m, since each of the 2^m CU requires a MUX

[2]This figure is also provided in Section 5.4.3, Figure 5.8.

to select its input individually. For each MUX, two individual multiplexors are used respectively for Regbank1 and Regbank2. The number of inputs to each multiplexor is determined by the number of the registers in the corresponding register bank. For Regbank1, the dummy registers need to be considered as well. As a result, the number of inputs to the MUX for Regbank1 can be derived from Figure 6.4, namely, $k+n+8$. For Regbank2, a $2^m(2^k-1)$ input multiplexor is required, as will be explained in Section 6.2.4.

6.2.3 Memories

As shown in Figure 6.2, the memories required by the proposed architecture may be divided into two types, namely, the LLR memory blocks and the metric memory block. Each LLR memory block stores one a priori or extrinsic LLR sequence. For a particular LUT-Log-BCJR decoder, the required number of extrinsic LLR memory blocks is k. The required number of a priori LLR memory blocks depends on whether the implemented turbo code is systematic or not. For a systematic code, $k+n$ a priori LLR memory blocks are required. By contrast, only n a priori LLR memory blocks are required for a non-systematic code, since the systematic bits are not transferred to the decoder. The use of the LLR memories in a turbo decoder is shown in Figure 6.3. All of the LLR memory blocks have the same size and the same word length, which is determined by the block length N of the turbo code design and the word length specification of the input LLRs. Here, the word length specification depends on the fixed-point setting of the LLRs, as investigated in Chapter 4.

For the metric memory, only one block is used for the proposed architecture. Between the operation of the forward and backward recursions, the metric memory stores 2^m number of $\alpha(S)$ values from Equation 5.5, for each of the w_s steps in the current window of the trellis. Since the $\alpha(S)$ values are always accessed in groups of 2^m at the same time, each group is stored in one word of the memory, in order to simplifying the design. Therefore, the word length of the metric memory is required to be long enough to store 2^m $\alpha(S)$ values at once.

6.2.4 Controller design and decoding time consumption

In this section, the redesigned controller for the generalised LUT-Log-BCJR architecture is proposed. The decoding time consumption is discussed along with the controller design, since this information is required for the energy estimation framework in Section 6.3.

The controller controls the operations performed during the decoding process. As discussed in Section 5.4.4, it can be described by a schedule chart like Figure 5.9. The redesign that is proposed in this section adjusts the calculation of δ values and the use of Regbank2. The purpose of the LUT-Log-BCJR decoder is to calculate the extrinsic LLRs in the sequence $\tilde{b}^{\mathrm{e}}_{i,j}$, where $i \in [1, 2, ..., k+n]$ and $j \in [1, 2, ..., N]$. In order to determine the time required to perform the decoding, the adjusted algorithm and the number of ACS operations performed during the calculation of the $\gamma_i(T)$, $\alpha(S)$, $\beta(S)$, $\delta_i(T)$ and $\tilde{b}^{\mathrm{e}}_{i,j}$ values are summarised as follows.

1. $\gamma_i(T)$ values

$$\gamma_i(T) = \begin{cases} 0 & \text{if } \mathcal{B}_i(T) = 1 \\ \tilde{b}^{\mathrm{a}}_{i,\mathcal{J}(T)} & \text{if } \mathcal{B}_i(T) = 0 \end{cases} \tag{6.1}$$

   $\gamma_i(T)$ values are the fundamental components of the $\alpha(S)$ and $\beta(S)$ calculations, as shown in Equation 6.1, 6.3 and 6.5. There is no calculation required to obtain the $\gamma_i(T)$ values, since these adopt the value of either $\tilde{b}^{\mathrm{a}}_{i,j}$ or 0, depending on the value of $b_{i,j}(T)$ in the decoding trellis, as shown in Equation 6.1. Owing to this, half of the γ_i values have a value of zero and do not need to be considered by subsequent addition operations. Note that there are $k+n$ number of $\gamma_i(T)$ values for each transition T, namely one corresponding to each LLR sequence $i \in [1, k+n]$.

2. $\alpha(S)$ values

$$\epsilon(T) = \alpha\big(\mathrm{fr}(T)\big) + \sum_{i=1}^{k+n} \gamma_i(T) \tag{6.2}$$

$$\alpha(S) = \overset{*}{\max_{T \in \mathrm{to}(S)}} \epsilon(T) \tag{6.3}$$

   Each $\alpha(S)$ value corresponds to a state S in the decoding trellis. In the general case, the calculation of an $\alpha(S)$ value is given by Equation 6.3. Since 2^k transitions merge into each state in the trellis, a total of $2^k - 1$ $\max^*$ operations are required to calculate each $\alpha(S)$ value. The additions required by $\alpha(S)$ calculations are those of Equation 6.1 and 6.2. However, since many $\gamma_i(T)$ values have a value of zero, the number of additions required by each $\alpha(S)$ value is different. Considering that half of the $\gamma_i(T)$ values are equal zero, a total of $2^{k-1}(k+n)$ additions are required to obtain all of the $\alpha(S)$ calculations in each step j of the trellis.

3. $\beta(S)$ values

$$\eta(T) = \beta\big(\mathrm{to}(T)\big) + \sum_{i=1}^{k+n} \gamma_i(T) \tag{6.4}$$

$$\beta(S) = \overset{*}{\max_{T \in \mathrm{fr}(S)}} \eta(T) \tag{6.5}$$

   The calculation of a $\beta(S)$ value is similar to $\alpha(S)$ values and the required number of ACS operations is the same.

4. $\delta_i(T)$ values

$$\delta_i(T) = \alpha\big(\mathrm{fr}(T)\big) + \eta_j(T) - \gamma_i(T) \tag{6.6}$$

For each particular input sequence $i \in [1, k+n]$, there is one $\delta_i(T)$ value corresponding to each transition T in the trellis. A total of 2^{k+m} $\delta_i(T)$ values are required to calculate each extrinsic LLR. The generalised calculation of a $\delta_i(T)$ value is given in Equation 6.6, which includes two addition/subtraction operations. Therefore, a total of 2^{k+m+1} addition/subtraction operations are required to perform all the $\delta_i(T)$ calculations in any particular step j of the trellis.

Note that the original δ calculation, from Equation 5.8, is given in Equation 6.7.

$$\delta_i(T) = \alpha\big(\mathrm{fr}(T)\big) + \beta\big(\mathrm{to}(T)\big) + \Bigg(\sum_{i'=1, i'\neq i}^{k+n} \gamma_{i'}(T)\Bigg) \tag{6.7}$$

According to Equation 6.7, when $k+n$ becomes a large number, the number of additions required by the $\delta_i(T)$ calculations increases significantly. To reduce the number of additions required by the $\delta_i(T)$ calculations, Equation 6.6 is proposed for the redesigned controller. This introduces the use of the internal variables $\eta(T)$. As discussed in Chapter 2, the $\beta(S)$ calculation used in each step j of the decoding trellis, includes the calculation of a variable $\eta(T)$ corresponding to each transition T in the trellis. Each $\delta_i(T)$ value is also corresponds to a particular transition in the trellis. Therefore, using the $\eta(T)$ values, the calculation $\delta_i(T)$ values can be simplified according to Equation 6.6. When $k > 1$ or $n > 1$, the number of additions required by Equation 6.6 is significantly reduced compared with Equation 6.7. As mentioned in Section 6.2.1, the use of Equation 6.6 to calculate the $\delta_i(T)$ values when $k > 1$ causes the number of registers required in Regbank2 to increase. During the backward recursion, all the $\eta(T)$ values calculated for the current step j in the trellis must be stored for the $\delta_i(T)$ calculations later. There are 2^{m+k} transitions in each step j of the trellis, so 2^{m+k} $\eta(T)$ values are required to be stored at any one time. The controller design proposed in this section allows each CU to store one $\eta(T)$ value in one of its general purpose registers. Since there are 2^m CUs in the datapath, 2^m $\eta(T)$ values can be stored in the general purpose registers. The remaining $\eta(T)$ values must be stored in Regbank2. Therefore, $2^m(2^k - 1)$ registers are required in Regbank2. Note that when $k = 1$, only $2^m(2^k - 1) = 2^m$ registers are required in Regbank2, which is the same number that was used in the implementation of Chapter 5. Therefore, the datapath implementation of $k = 1, n = 1$ in Chapter 5 is a special case of the proposed generalised architecture.

5. Extrinsic LLRs

$$\tilde{b}^{\mathrm{e}}_{i,j} = \overset{*}{\max_{T\left|\substack{\mathcal{B}_i(T)=0\\ \mathcal{J}(T)=j}\right.}} \big(\delta_i(T)\big) - \overset{*}{\max_{T\left|\substack{\mathcal{B}_i(T)=1\\ \mathcal{J}(T)=j}\right.}} \big(\delta_i(T)\big) \tag{6.8}$$

The extrinsic LLRs are the final output of the decoder. The calculation of the extrinsic LLR is given in Equation 6.8. For each extrinsic LLR, 2^{k+m} $\delta_i(T)$ values are divided into two groups, in order to calculate the difference of two max* results. A total of $2(2^{k+m-1}-1)$ max* operations and one subtraction operation are required. In each step j of the trellis, there are k extrinsic LLRs that need to be calculated.

As discussed, the decoding process is divided into three recursion processes, namely, the forward recursion, the pre-backward recursion and the backward recursion. The time consumption of the redesigned controller in each of the three recursions is summarised as follows.

1. Forward recursion:
The generalised controller design of the forward recursion can be described as shown in Figure 6.6 (a). Regbank2 is divided into groups for the controller, where

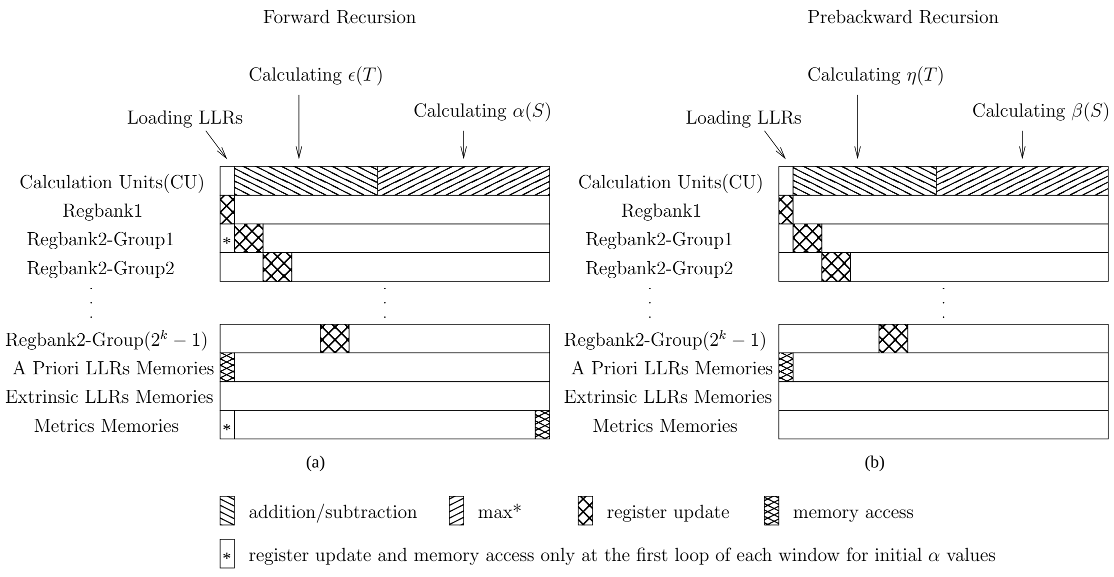


FIGURE 6.6: The decoding schedules of the forward and prebackward recursions.

each of the 2^k-1 groups contains 2^m registers. In this way the registers are always updated or read one group at a time, as shown in Figure 6.6 (a). Each step j of the forward recursion begins with a clock cycle in which the corresponding a priori LLRs $\{\tilde{b}^{\mathrm{a}}_{1,j}, \tilde{b}^{\mathrm{a}}_{2,j}, ...\tilde{b}^{\mathrm{a}}_{k+n,j}\}$ are loaded into Regbank1. Next, the $\epsilon(T)$ values are calculated. Similarly to the $\eta(T)$ calculation, 2^m $\epsilon(T)$ values can be stored of the general purpose registers of each CU. The remaining $2^m(2^k-1)$ values are stored in Regbank2. Since there are 2^m CUs in the architecture, the decoder calculates

2^m operands at a time. The results are then stored into one of the subgroups in Regbank2, as shown in Figure 6.6 (a). Following this, the max* operations can be performed to calculate the $\alpha(S)$ values, since all the required $\epsilon(T)$ values are available in the registers. The resultant $\alpha(S)$ values are stored in the metric memory block of Figure 6.2. Note that for the first step in each window, the set of $\alpha(S)$ values calculated for the previous step need to be restored from the metric memory, in order to initial the forward recursion.

As shown in Figure 6.6, each add/sub operation requires one clock cycle and each max* operation requires four clock cycles. For each step j, in the forward recursion, a total of $2^{k-1}(k+n) + 4(2^k - 1)$ clock cycles are required. In addition, one clock cycle is required to load the corresponding a priori LLRs from the memory. Therefore, the total number of clock cycles required for each step of the forward recursion, is given by

$$T_{\text{forward}} = 2^{k-1}(k+n) + 2^{k+2} - 3. \tag{6.9}$$

2. Pre-backward recursion:
   The pre-backward recursion is shown in Figure 6.6 (b). It is the same as the forward recursion, except that there is no need to store the $\beta(S)$ values, since only the results from the last step in the window are required in order to initialise the backward recursion. Furthermore, there is no need for the initial values for each pre-backward window since the initial values of $\beta(S)$ are all zero.

   The number of clock cycles required for each step j of the pre-backward recursion is equal to that of the forward recursion, as shown in Figure 6.6, $T_{\text{prebackward}} = T_{\text{forward}}$.

3. Backward recursion:
   The backward recursion is shown in Figure 6.7. It starts with $\beta(S)$ calculation, as in the pre-backward recursion. Following the $\beta(S)$ calculation is a loop for the extrinsic LLR calculation, including the $\delta_i(T)$ calculation and extrinsic LLR calculation, as shown in Figure 6.7. There are k extrinsic LLRs $\{\tilde{b}^{\text{e}}_{1,j}, \tilde{b}^{\text{e}}_{2,j}, ...\tilde{b}^{\text{e}}_{k,j}\}$ which need to be calculated, so the loop repeats k times.

   As discussed in Chapter 5, the use of the CUs for the max* operations in the extrinsic LLR calculations is different from all the other calculations in the decoding process. Each set of max* operations is distributed among the CUs using a binary tree structure, in order to perform Equation 6.8, as shown in Figure 6.8. Figure 6.8 provides a detailed schedule of the CUs for the last stage in Figure 6.7, including the max* operations and the final subtraction of the extrinsic LLRs, according to Equation 2.18. In general case, there are 2^{m+k} $\delta_i(T)$ values per trellis step j. The process to calculate the extrinsic LLR using the $\delta_i(T)$ values can be divided into three stages, as shown in Figure 6.8. In the first stage, all the CUs perform max*

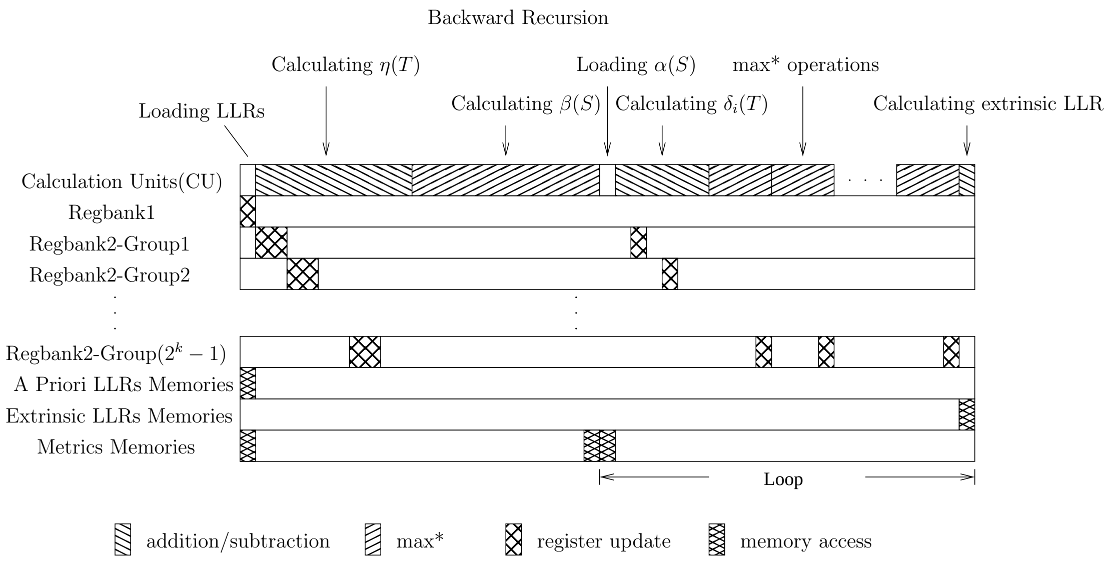


FIGURE 6.7: The decoding schedule of the backward recursion.

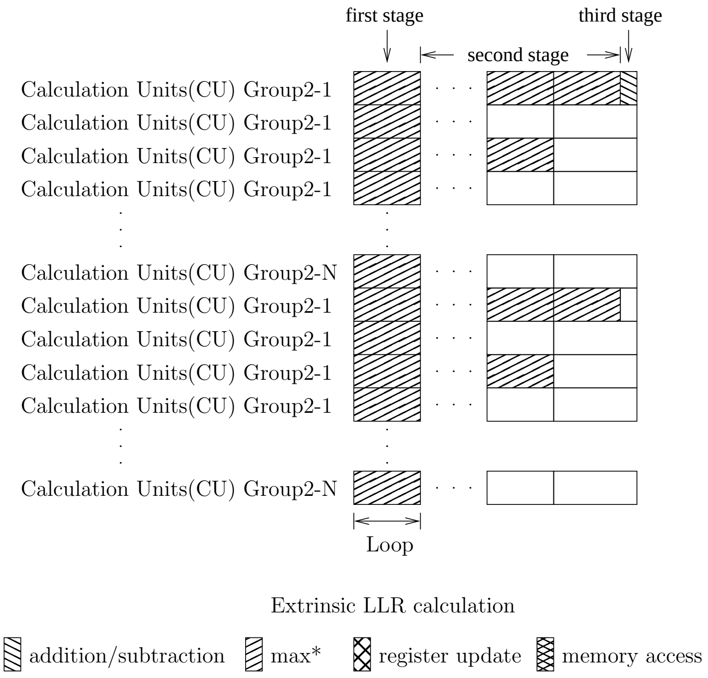


FIGURE 6.8: The decoding schedule of the extrinsic LLR calculations.

operations in a loop until there are not enough operands for all of the CUs to work together. The second stage starts with only half of the CUs performing the max* operations. The number of working CUs reduces by half after each max* cycle, until there are only two CUs working at the same time. Then, in the final stage, the first CU performs a subtraction of the results from the two CUs in previous max* cycle, in order to obtain the extrinsic LLR.

In the backward recursion, the $\beta(S)$ calculation requires the same time consumption as the forward and pre-backward recursions. For $\delta_i(T)$ calculations, one clock cycle is required for loading the required $\alpha(S)$ values from the metric memory. A further 2^{k+1} clock cycles are required to calculate the $\delta_i(T)$ values. A total of $2^{k+1}+1$ clock cycles are required. In the first stage of the extrinsic LLR calculations, the number of max* operands reduces from 2^{m+k} to 2^{m+1}, requiring k max* cycles. In the second stage, $\log_2 2^{m-1} = m-1$ max* cycles are required to reduce the number of max* operands from 2^m to 4. In the third stage, one clock cycle is used to perform the subtraction of the two results from the second stage. There are k extrinsic LLRs that need to be calculated for one step j in the trellis, this requires the loop shown in Figure 6.8 to be repeated k times. The time consumption to calculate all the $\delta_i(T)$ values and the extrinsic LLRs for one step j in the trellis is therefore $k(2^{k+1}+4(k+m-1)+1) = k(2^{k+1}+4k+4m-3)$ clock cycles. The total time consumption for one loop of the backward recursion is given by

$$T_{\text{backward}} = 2^{k-1}j + 2^{k+2} + k(2^{k+1}+4k+4m-3) - 2 \tag{6.10}$$

The generalised controller design for the architecture proposed in Chapter 5 can be adapted to any LUT-Log-BCJR decoder. For the special case of $k=1, n=1$, the controller is same as the design of Chapter 5, except for the proposed alternative $\delta_i(T)$ calculation.

In order to obtain the throughput of the LUT-Log-BCJR decoder, it is necessary to consider the sliding window length w_{s} and the pre-backward window recursion length w_{p}. In each backward recursion, k number of extrinsic LLRs are calculated. Therefore, the average number of clock cycles for calculating each extrinsic LLR can be calculated as

$$T_{\text{e}} = \frac{w_{\text{s}}(T_{\text{forward}} + T_{\text{backward}}) + w_{\text{p}}T_{\text{prebackward}}}{w_{\text{s}} \times k}. \tag{6.11}$$

In summary, the configuration and the hardware structure of the proposed architecture can be predicted at the early design stage. This conclusion is the foundation of the proposed energy estimation framework of this chapter. In the next section, the proposed energy estimation framework is presented based on the discussion in this section.

6.3 Energy estimation framework

In this section, the proposed architecture is employed as the basis of a framework for estimating the energy consumption of a LUT-Log-BCJR decoder, as well as a turbo decoder. As mentioned in Section 6.1, the objective of this framework is to provide quantified energy consumption information of the LUT-Log-BCJR decoder at the turbo code design stage, in order to assist the designer in making decisions about the parameters of the code.

In order to obtain quantified energy estimation results, some assumptions from the later implementation stages are required. The Integrated Circuit (IC) fabrication process technology [160], the supply voltage and the clock frequency of the implemented circuit may all have significant impacts on the energy consumption. Therefore, Taiwan Semiconductor Manufacturing Company (TSMC) 90nm technology is chosen in this work, to provide typical energy consumption information for the hardware implementation. The proposed framework can be extended to different process technologies by employing the triple scaling factor of Table 5.4, which is a widely used method to approximately estimate the energy consumption of a particular architecture, when using a different process technology. The input variables of the framework are summarised in Table 6.1. When

k	the number of inputs of each component encoder
m	the number of memory elements of each component encoder
n	the number of non-systematic outputs of each component encoder
w_{s}	the sliding-window length
w_{p}	the pre-backward recursion length
N	the block length
I	the number of decoding iterations performed
v	the supply voltage
f	the clock frequency
x, y and z	the word length specifications of the decoder, as described in Chapter 4

Table 6.1: Summary of the variables in the energy estimation framework.

using the framework to estimate the LUT-Log-BCJR decoder's energy consumption, the designer is able to change the values of the variables to investigate their impact on the energy consumption. Note that the choices of the word length specifications, x, y and z, are fully investigated in Chapter 4. Therefore, they are not discussed in detail in this chapter. Instead, the word length specifications identified in Chapter 5, $x = 4$, $y = 7$ and $z = 2$, are assumed during the formation of the framework. However, the option of estimating the energy consumption for a different x, y and z specification setting is introduced at the end of the framework.

The proposed architecture of Figure 6.2 is considered as the combination of CUs, Regbank1 and Regbank2. Here the CUs contain all of the combinational logic structures. Regbank1 and Regbank2 are left with pure register arrays. Since synthesis results are highly dependent on the data path lengths in the design, the described sub-module division guarantees that all the data paths are contained in one sub-module, the CU. Therefore, the described approach minimises the impact of variations imposes by the synthesis process. The energy consumption of the architecture's datapath can be calculated as the sum of each sub-module's energy consumption, as detailed in the following sections.

6.3.1 Calculation unit

In this section, the typical energy consumption of the CU sub-module is modelled as a function of the parameters k, m, n, w_s, w_p, v and f. By taking the four parameters as the input, the proposed energy model is able to calculate the typical energy consumption of the CU in nJ/Clock Cycle, $E_\mathrm{cyc}^\mathrm{CU}$.

The CU sub-modules includes an ACS unit, three general purpose registers, a interconnection unit and a MUX unit. The operating clock frequency and the supply voltage have a direct impact on the energy consumption. As discussed in Section 6.2.2, the parameters k, m and n have an impact on the MUX unit, which affects the hardware complexity. The parameters k, m, n, w_s and w_p determine the decoding throughput, which is also directly related to the energy consumption. To model the energy consumption of the CU, three types of energy consumption are investigated, namely, the energy consumed by the CU when performing add/sub operations, max* operations and in the idle state. Moreover, the variation of different parameters are investigated.

As a first step in deriving an energy model for the CU, a fixed set of parameter values $k = 1$, $m = 3$, $n = 1$ and the typical supply voltage of TSMC 90nm technology $v = 1.2$ V are chosen, in order to analyse the energy impact of varying the operating clock frequency f. For a particular hardware structure, the power consumption increases almost linearly with the clock frequency f [188], unless the data path lengths have longer delays than the clock period, requiring additional optimisation of the hardware implementation by the synthesis tool. Since the proposed architecture maintains control of the combinational logic data path lengths, the post layout simulation results show a very good linearity between the power consumption and the clock frequencies in the range 0 to 500 MHz, as shown in Figure 6.9. Here, $P(v)$ is defined as the power consumption of a circuit when the supply voltage is v. Since the power consumption is proportional to the square of the supply voltage, the power consumption for supply voltages other than $v = 1.2$ V can be estimated based on the $P(1.2)$ results of Figure 6.9 according to Equation 6.12.

$$P(v) = \frac{v^2}{1.2^2} \times P(1.2). \tag{6.12}$$

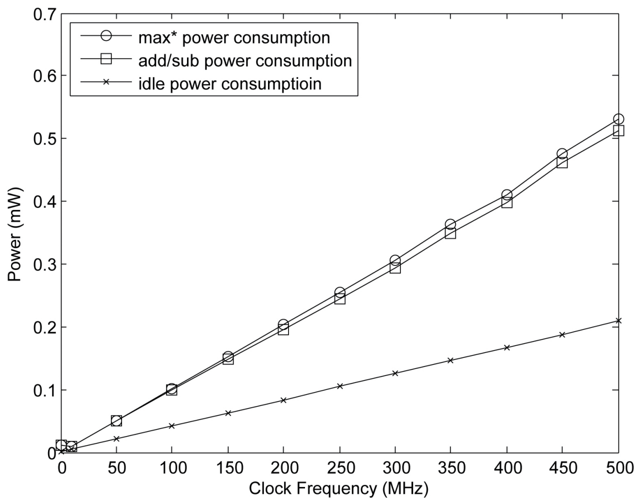


FIGURE 6.9: CU's power consumption results versus clock frequency, when $k = 1$, $m = 3$, $n = 1$ and $v = 1.2$ V.

However, a reduction of the supply voltage v has a similar impact on the hardware implementation as increasing the clock frequency. More specifically, this increases the delays in the circuit. When the delays in the data paths become longer than the clock periods, the synthesis tool will introduce additional circuitry, in order to reduce the data path lengths, at the cost of an increased hardware complexity. Therefore, Equation 6.12 is only applicable for a certain range of clock frequencies. For TSMC 90 nm technology, the specified supply voltage range is 0.84 V to 1.32 V [189]. To obtain the applicable clock frequency range of Equation 6.12 for the proposed architecture, the worst case $v = 0.84$ V is investigated. Figure 6.10 compares the power consumption obtained by

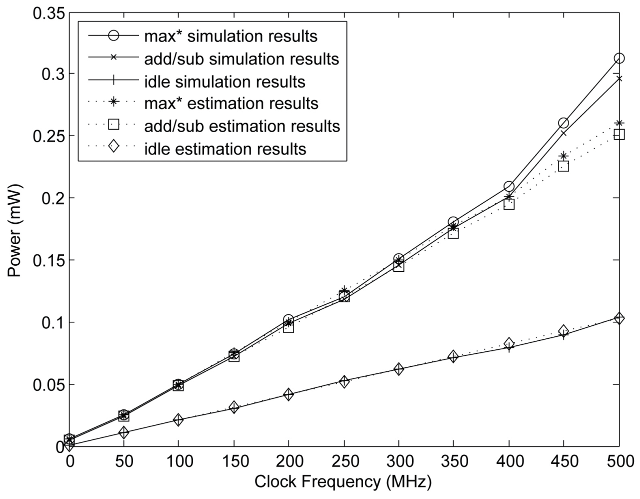


FIGURE 6.10: The comparison of post layout simulation results and the estimation results by Equation 6.12, when $k = 1$, $m = 3$, $n = 1$ and $v = 0.84$ V.

post layout results with the prediction made by Equation 6.12. As shown in the figure, the power consumption of the actual implementation results becomes higher than the estimation results from clock frequencies above $f = 400$ MHz. Therefore, when the proposed architecture is implemented using TSMC 90 nm technology, Equation 6.12 can be only used when $f \in (0, 400 \text{ MHz}]$.

In this work, the objective of the propose framework is to estimate the energy consumed by the LUT-Log-BCJR decoder per bit of information in nJ/bit. In Section 6.2.4, the throughput of the decoder was determined as a function of the clock period. Here, the energy consumption per clock cycle E_{cyc} is fundamental to the proposed energy estimation framework. This can be obtained by dividing the power consumption P of Figure 6.9 by the corresponding clock frequency f, according to,

$$E_{\text{cyc}} = \frac{P}{f}. \tag{6.13}$$

The results obtained from post layout simulation results show that when the hardware structure is implemented in TSMC 90 nm technology, E_{cyc} can be approximately considered to be a constant value that is independent from f. For example, the results obtained for four different combinations of the parameters $\{k+n, m\}$ are shown in Figure 6.11, including $\{2,3\}$, $\{3,3\}$, $\{2,5\}$ and $\{7,3\}$. As discussed in Section 6.2.2, the parameters k, m and n determine the size of the MUX used in the CU, which has only an insignificant effect on the energy consumption results, as shown in Figure 6.11. The

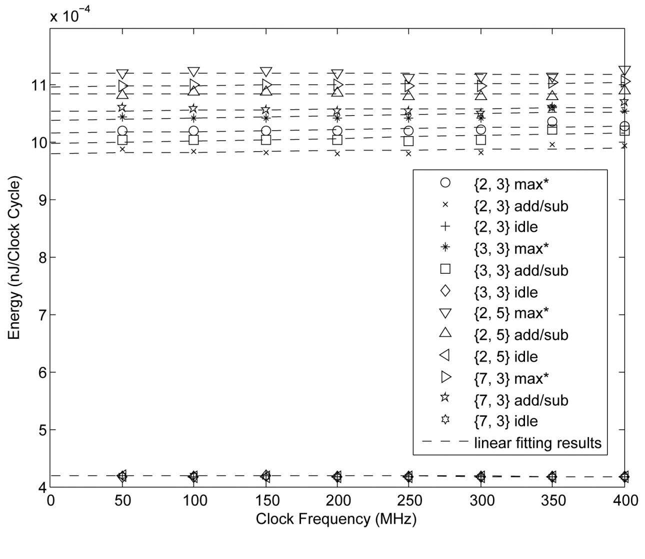


FIGURE 6.11: E_{cyc} results of the CU with four different combination settings of $k+n, m$, where $v = 1.2$ V.

dashed lines in Figure 6.11 are lines of best fit for each group of E_{cyc} results. The results show that, for a certain supply voltage v, E_{cyc} of the CU depends on $\{k+n, m\}$ and is approximately independent from f.

To model the relationship between E_{cyc} and $\{k+n, m\}$, the impact on the CU's hardware structure by the variation of $\{k+n, m\}$ must be considered. As discussed, parameters $k+n$ and m each affect the number of inputs required for a different multiplexor in the MUX unit. Therefore, the effects of $k+n$ and m are independent from each other, and so these parameters can be investigated individually. The simulation results of CU's E_{cyc} for different values of $k+n$ and m are provided in Figure 6.12 and 6.13. Note that when one parameter is changed, the other is fixed at its minimum value, namely $k+n=2$ or $m=1$. The simulation results show that the parameters $k+n$ and m have

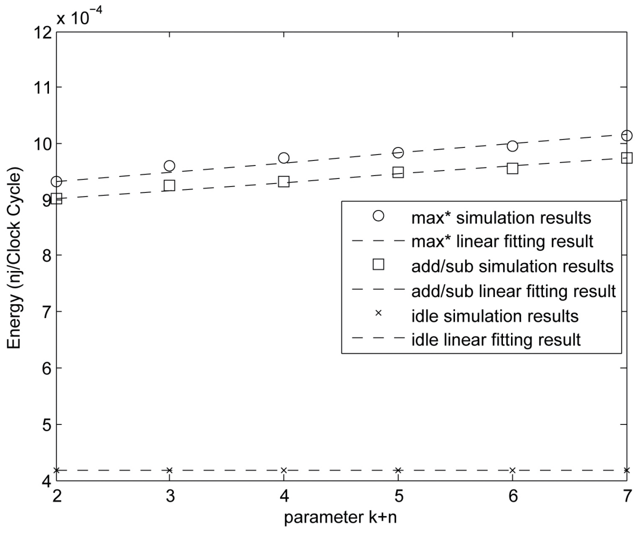


FIGURE 6.12: E_{cyc} results of the CU for different $k+n$, where $m=1$, $v=1.2$ V and $f=200$ MHz.

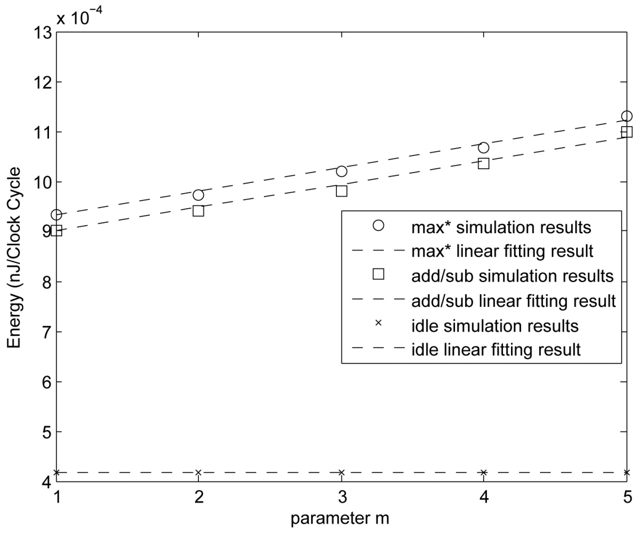


FIGURE 6.13: E_{cyc} results of the CU for different m, where $k=n=1$, $v=1.2$ V and $f=200$ MHz.

only a negligible effect on E_{cyc} when the CU is idling. Therefore, E_{cyc} can be considered to have a constant value which is independent from $k+n$ and m when the CU is idling. When the CU is performing max* or add/sub operations, the results in Figures 6.12 and 6.13 show that E_{cyc} increases linearly, when one of the parameter values is increased. Linear fitting results are provided in the figures accordingly. The effect on $k+n$ and m on the CU's energy consumption can be modelled as the energy consumption observed for $k+n=2$ and $m=1$, plus additional energy consumptions caused by the target $k+n$ and target m respectively. Therefore, the linear fitting results given in Figure 6.12 and 6.13 are constrained so that the functions cross the $k+n=2$ and $m=1$ points. The energy consumption E_{cyc} of the CU is modelled as a function $E_{\mathrm{cyc}}^{\mathrm{CU}}(k,n,m,v)$ of parameters $k+n$, m and v. The slopes of the four linear fitting results in Figures 6.12 and 6.13 and the value of $E_{\mathrm{cyc}}(1,1,1,1.2)$ for different operations are given in Table 6.2.

Operation	max*		add/sub	
Parameter	k+n	m	k+n	m
Slope	1.69×10^{-5}	4.73×10^{-5}	1.47×10^{-5}	4.64×10^{-5}
$E_{\mathrm{cyc}}(1,1,1,1.2)$ (nJ/Clock Cycle)	9.32×10^{-4}		9.015×10^{-4}	

TABLE 6.2: The slope values of linear fitting results in Figure 6.12 and 6.13 and $E_{\mathrm{cyc}}(1,1,1,1.2)$ for different operations.

Therefore, the CU's E_{cyc} can be modelled as

$$E_{\mathrm{cyc}}^{\mathrm{CU,max}^*}(k,n,m,v)=\frac{v^2}{1.2^2}(9.32\times10^{-4}+1.69\times10^{-5}(k+n-2)+4.73\times10^{-5}(m-1)), \tag{6.14}$$

$$E_{\mathrm{cyc}}^{\mathrm{CU,add}}(k,n,m,v)=\frac{v^2}{1.2^2}(9.015\times10^{-4}+1.47\times10^{-5}(k+n-2)+4.64\times10^{-5}(m-1)). \tag{6.15}$$

As discussed above, the only significant effect on E_{cyc} while the CU idling is given by parameter v. Therefore, it can be modelled by

$$E_{\mathrm{cyc}}^{\mathrm{CU,idle}}(v)=\frac{v^2}{1.2^2}0.418\times10^{-3}, \tag{6.16}$$

since $E_{\mathrm{cyc}}^{\mathrm{CU,idle}}(1.2)=0.418\times10^{-3}$. A comparison between the results estimated using these models for the examples of Figure 6.11 and the corresponding simulation results is given in Figure 6.14.

Based on Equation 6.14, 6.15 and 6.16, the energy consumption of the CU in a LUT-Log-BCJR decoder can be calculated, by investigating the three recursions in the decoding process.

1. CU energy analysis for the forward recursion.
   As shown in Figure 6.6, the operations of the CU in the forward recursion can be divided into three stages, including idle stage, addition stage and max* stage. As discussed in Section 6.2.4, one clock cycle is required for the idle stage, $2^{k-1}(k+n)$ clock cycles are required for the addition stage and $4(2^k-1)$ clock cycles are required for the max* stage. The average E_{cyc} of the CU during the forward

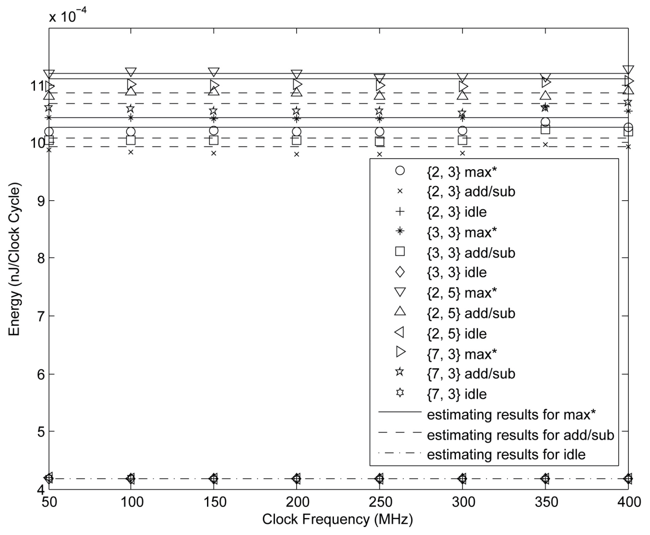


FIGURE 6.14: The comparison of E_{cyc}'s simulation results and estimated results of the CU with the same examples in Figure 6.11.

recursion can be calculated as

$$E_{\text{cyc}}^{\text{CU,forward}} = \frac{E_{\text{cyc}}^{\text{CU,idle}} + 2^{k-1}(k+n)E_{\text{cyc}}^{\text{CU,add}} + 4(2^k-1)E_{\text{cyc}}^{\text{CU,max}^*}}{T_{\text{forward}}}. \tag{6.17}$$

2. CU energy analysis for the pre-backward recursion.
   As shown in Figure 6.6, the operations performed by the CUs during the pre-backward recursion are the same as those of the forward recursion, so

$$E_{\text{cyc}}^{\text{CU,prebackward}} = E_{\text{cyc}}^{\text{CU,forward}}. \tag{6.18}$$

3. CU energy analysis for the backward recursion.
   As discussed in Section 6.2.4, the backward recursion includes the calculation of the $\beta(S)$ values, the $\delta_i(T)$ values and the extrinsic LLRs. The $\beta(S)$ calculation is the same as in the pre-backward recursion. The $\delta_i(T)$ calculation includes one clock cycle in idle and 2^{k+1} clock cycles of add/sub operations. The calculation of the extrinsic LLRs can be divided into three stages. In the first stage, each CU perform k max* operations, taking $4k$ clock cycles. In the second stage, the operations in each CU are different. A total of $4(m-1)$ clock cycles are required to perform the $m-1$ max* operations, as shown in Figure 6.8. On average, each CU spends a fraction $\frac{\sum_{i=1}^{m-1} 2^i}{(m-1)2^m}$ of the time performing max* operations and a fraction $1 - \frac{\sum_{i=1}^{m-1} 2^i}{(m-1)2^m}$ of the time in idle. In the third stage, only one CU is operated, performing a subtraction in one clock cycle. On average, each CU spends $\frac{1}{2^m}$ of the time performing subtractions and $1 - \frac{1}{2^m}$ of the time in idle. The loop in Figure 6.8 is repeated k times in each backward recursion. Therefore, the average

E_{cyc} for the backward recursion is given by

$$E_{\text{cyc}}^{\text{CU,backward}} = \frac{T_{\text{add}} \times E_{\text{cyc}}^{\text{CU,add}} + T_{\text{max}^*} \times E_{\text{cyc}}^{\text{CU,max}^*} + T_{\text{idle}} \times E_{\text{cyc}}^{\text{CU,idle}}}{T_{\text{backward}}}, \tag{6.19}$$

where

$T_{\text{add}} = 2^{k-1}(k+n) + (2^{k+1} + \frac{1}{2^m})k$,

$T_{\text{max}^*} = 4(2^k - 1) + \Big(4k + (\frac{\sum_{i=1}^{m-1} 2^i}{(m-1)2^m})4(m-1)\Big)k$,

$T_{\text{idle}} = 2 + \Big(1 - \frac{1}{2^m} + (1 - \frac{\sum_{i=1}^{m-1} 2^i}{(m-1)2^m})4(m-1)\Big)k$.

Overall, the average E_{cyc} of one CU is given by

$$E_{\text{cyc}}^{\text{CU}} = \frac{w_{\text{s}}(E_{\text{cyc}}^{\text{CU,forward}} + E_{\text{cyc}}^{\text{CU,backward}}) + w_{\text{p}} E_{\text{cyc}}^{\text{CU,prebackward}}}{2w_{\text{s}} + w_{\text{p}}}. \tag{6.20}$$

6.3.2 Register banks

In this section, the typical energy consumption of the register banks is estimated. Two internal variables are introduced for the energy model, namely the update rate u of a register in the unit of update times per clock cycle and the number of registers in a register bank r. As discussed in Section 6.2.1, the values of r for Regbank1 r_1 and Regbank2 r_2 are given by

$$r_1 = k + n, \tag{6.21}$$

$$r_2 = 2^m(2^k - 1). \tag{6.22}$$

In this section, the calculation of the values of u for Regbank1 and Regbank2 are discussed. The energy consumption of a register bank $E_{\text{cyc}}^{\text{Regbank}}$ can be calculated by multiplying the energy consumption of one register $E_{\text{cyc}}^{\text{Reg}}$ by the number of registers r it contains. Here, $E_{\text{cyc}}^{\text{Reg}}$ is considered to be a function of the parameter u in the energy model.

For Regbank1, all the registers are updated together, as described in Section 6.2.4. As shown in Figure 6.6 and 6.7, one update is performed in each loop of all three recursions. Therefore, u of Regbank1 can be calculated as

$$u_1 = \frac{2w_{\text{s}} + w_{\text{p}}}{w_{\text{s}}(T_{\text{forward}} + T_{\text{backward}}) + w_{\text{p}} T_{\text{prebackward}}} \tag{6.23}$$

As described in Section 6.2.4, Regbank2 is divided into $2^k - 1$ groups of 2^m registers. The registers in each group always updated together. As shown in Figure 6.6, for each register is updated an average of two times in each of the forward and the pre-backward recursions, giving the update rates

$$u_{2,\text{forward}} = \frac{k+n}{2T_{\text{forward}}}, \tag{6.24}$$

$$u_{2,\text{prebackward}} = \frac{k+n}{2T_{\text{prebackward}}}. \tag{6.25}$$

For the backward recursion, each group is updated $\frac{(k+n)(2^k-1)}{2} + 2^k + k(k+m-1) - 2$ times in total, according to Figure 6.7. The average update rate is therefore

$$u_{2,\text{backward}} = \frac{\frac{(k+n)(2^k-1)}{2} + 2^k + k(k+m-1) - 2}{(2^k-1)T_{\text{backward}}}. \tag{6.26}$$

Finally, the overall u of Regbank2 can be calculated as

$$u_2 = \frac{w_{\text{s}}u_{2,\text{forward}} + w_{\text{s}}u_{2,\text{backward}} + w_{\text{p}}u_{2,\text{prebackward}}}{2w_{\text{s}} + w_{\text{p}}}. \tag{6.27}$$

Note that for energy estimation, only real registers are considered. The dummy registers of Regbank1 need no consideration, since they are hard wired constant values in the hardware implementation, as discussed in Section 6.2.1. The energy model of a register bank can be built based on the energy analysis of a single register. Post-layout simulation results of the energy consumed by a single register having a various update rates u, are shown in Figure 6.15. Similar to the CU's results, the energy consumption of a

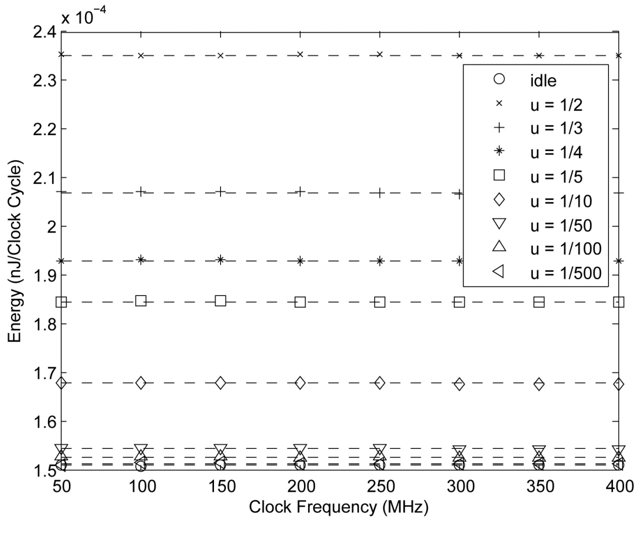


FIGURE 6.15: Energy consumptions of a single register operating at different clock frequencies, for $v = 1.2$ V.

register can be considered to be a constant value that is independent from the clock frequency. The dash lines in the figure are the average values of the simulation results. The results show that the energy consumption significantly increases when u increases. In Figure 6.16, these average values are used to model the energy consumption of a register, as a function of u. The dashed line in the figure provides the linear fitting

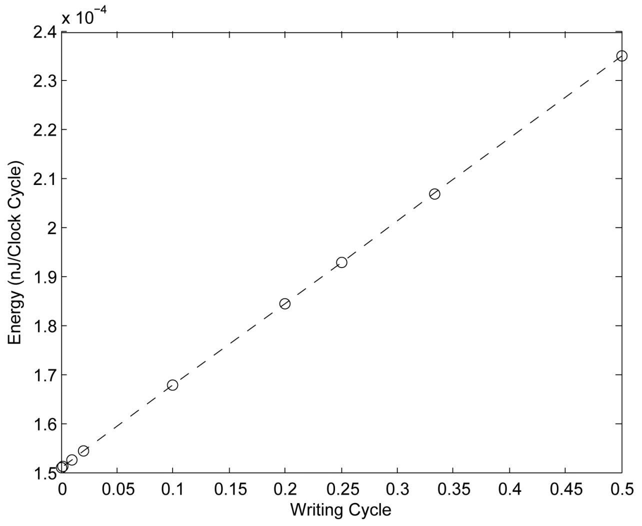


FIGURE 6.16: Energy consumptions of a single register with different update rate u.

result, which demonstrates the linearity of the energy consumption with u as modelled by

$$E_{\text{cyc}}^{\text{Reg}} = (0.168u + 0.1511) \times 10^{-3}. \tag{6.28}$$

Note that when $u = 0$, $E_{\text{cyc}}^{\text{Reg}}$ represents the energy consumption of the register in idle. Using the linear fitting result, the energy consumption of a register bank having r number of registers can be calculated as

$$E_{\text{cyc}}^{\text{Regbank}} = \frac{v^2}{1.2^2} r(0.168u + 0.1511) \times 10^{-3}. \tag{6.29}$$

Since Regbank1 has $k + n$ registers and Regbank2 has $2^m(2^k - 1)$ registers, their energy consumption are given by

$$E_{\text{cyc}}^{\text{Regbank1}} = \frac{v^2}{1.2^2}(k + n)(0.168u_1 + 0.1511) \times 10^{-3}. \tag{6.30}$$

$$E_{\text{cyc}}^{\text{Regbank2}} = \frac{v^2}{1.2^2} 2^m(2^k - 1)(0.168u_2 + 0.1511) \times 10^{-3}. \tag{6.31}$$

To verify the energy model, an error bar plot generated by the simulation results and the estimation results of $r = 8$ and $v = 1.2$ V is given in Figure 6.17. Here, the error bars show the range of energy consumption values that result when the clock frequency is varied in the range of 50 MHz to 400 MHz.

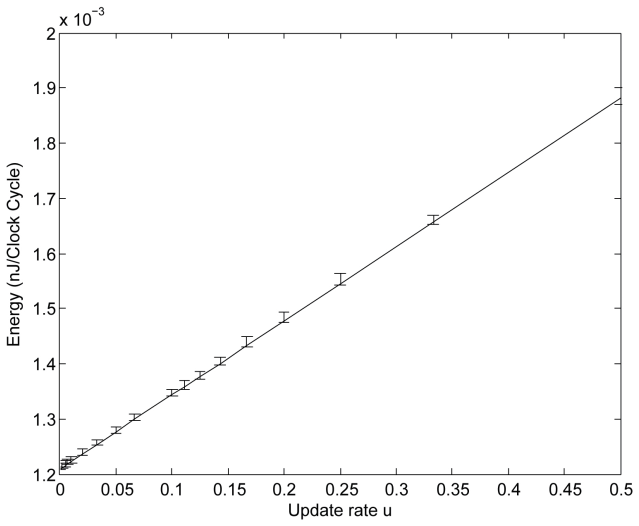


FIGURE 6.17: The error bar result, $r = 8$ and $v = 1.2$ V.

6.3.3 Datapath

There are 2^m CUs in the proposed architecture. Based on the energy model of Sections 6.3.1 and 6.3.2, the energy consumption in nJ/Clock Cycle can be calculated as

$$E_{\mathrm{cyc}}^{\mathrm{Datapath}} = 2^m E_{\mathrm{cyc}}^{\mathrm{CU}} + E_{\mathrm{cyc}}^{\mathrm{Regbank1}} + E_{\mathrm{cyc}}^{\mathrm{Regbank2}}. \tag{6.32}$$

This framework proposes to use the unit nJ/bit to evaluate the energy consumption of a Log-LUT-BCJR decoder. More specifically, this quantifies the average energy consumed to decode one extrinsic LLR $E_{\mathrm{e}}^{\mathrm{Datapath}}$. The energy efficiency of a datapath $E_{\mathrm{e}}^{\mathrm{Datapath}}$ may be obtained by multiplying $E_{\mathrm{cyc}}^{\mathrm{Datapath}}$ with the number of clock cycles required to calculate one extrinsic LLR, T_{e}, which can be obtained using Equation 6.11. The word length of the fixed-point representation used in the datapath is $y + z$, as defined in Table 6.1. The results so far have been based on the assumption that $y + z = 9$. Therefore, for a datapath with a different fixed-point setting, $E_{\mathrm{e}}^{\mathrm{Datapath}}$ can be calculated according to

$$E_{\mathrm{e}}^{\mathrm{Datapath}} = \frac{y+z}{9} E_{\mathrm{cyc}}^{\mathrm{Datapath}} \times T_{\mathrm{e}}. \tag{6.33}$$

6.3.4 Controller

In typical ASIC design processes, intricate knowledge of the controller's hardware implementation cannot be obtained before synthesis. This is because unlike the datapath and the memory blocks, the controller design is based on the behaviour model. As a result, the energy consumption of the controller is difficult to estimate in an early design stage [185, 190].

In this framework, a method to approximately estimate the controller's energy consumption is proposed. A configurable Register-Transfer Level (RTL) model of the proposed architecture's controller is designed for investigating the energy consumption of the controller with different specifications of the design parameters. The parameters that affect the controller includes k, m, n, w_s, w_p and N. This RTL module is not an actual controller for any LUT-Log-BCJR decoder, but it is designed to include the state machine, which can be abstracted from Figure 6.6 and 6.7, and part of the combination logics of the control signals, which can be generalised for any decoder. The RTL module can be easily reconfigured by changing the parameters for the investigation. It represents most of the energy consumption of an actual decoder. For example, for the Long Term Evolution (LTE) decoder, simulation results show that the energy consumption of the RTL module's simulation result is more than 95% of the actual design's. The remaining 5% combinational logic that defines the output signals of the controller, but which cannot be generalised, since the design is different for different decoders. For the proposed architecture, this inaccuracy in the controller's energy estimation is acceptable, since the controller typically contributes only a small fraction (less than 5%) of the total energy consumption of the decoder, as discussed in this section.

Using the proposed RTL module, the energy consumption of the proposed architecture's controller is investigated. In Figure 6.18, the energy consumption simulation results of the controller in nJ/Clock Cycle, $E_\mathrm{cyc}^\mathrm{Control}$ are given for $k = 1$, $m = 1$ and $n = 1$, with different operating clock frequency f and interleaver length N. Here, $w_\mathrm{s} = 128$, $w_\mathrm{p} = 24$ and the typical supply voltage for TSMC 90 nm technology $v = 1.2$ V are assumed. It

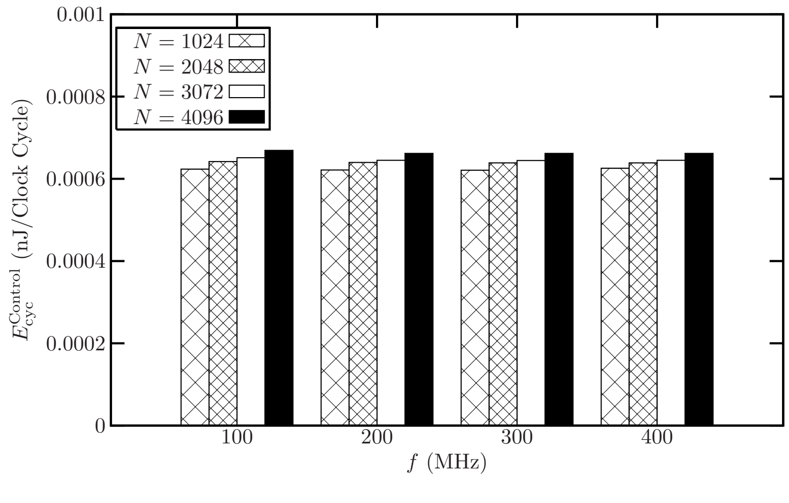


FIGURE 6.18: $E_\mathrm{cyc}^\mathrm{Control}$ (nJ/Clock Cycle) with different f and N, when $k = 1$, $m = 1$ and $n = 1$.

is shown that the energy consumption variation caused by different clock frequencies f is insignificant. Therefore, $E_\mathrm{cyc}^\mathrm{Control}$ is considered to be independent from f.

In the controller design, the block length N only affects a bit counter in the controller. This bit counter identifies which bit is being processed and determines the end of a block, which has a length N. Therefore, the bit counter register must be able to store values of up to N. According to the binary number representation, the required word length is defined by $\lceil \log 2(N+1) \rceil$. As a result, N only causes an apparent increment of $E_{\text{cyc}}^{\text{Control}}$ when $\lceil \log 2(N+1) \rceil$ increases. To investigate the effect, $E_{\text{cyc}}^{\text{Control}}$ obtained from post-layout simulation results with an increasing N are given in Figure 6.19, together with a linear fitting result. According to Figure 6.19, when $f = 400$ MHz, $v = 1.2$ V,

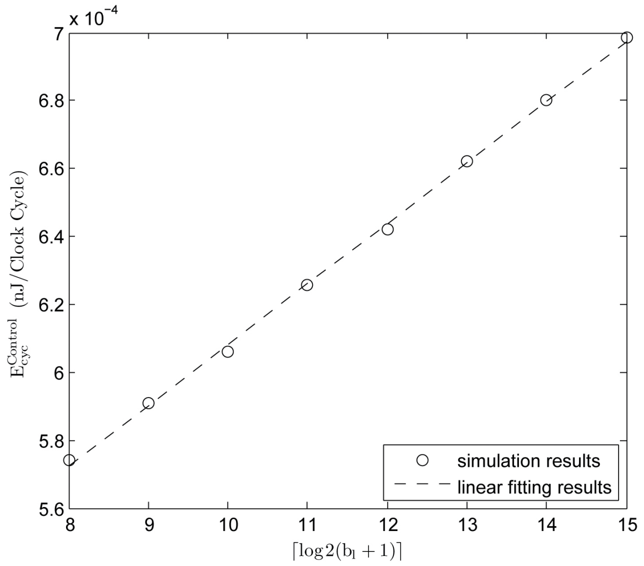


FIGURE 6.19: The increment of $E_{\text{cyc}}^{\text{Control}}$ when N increases.

$w_{\text{s}} = 128$, $w_{\text{p}} = 24$, $k = 1$, $m = 1$ and $n = 1$, $E_{\text{cyc}}^{\text{Control}}$ can be modelled by Equation 6.34.

$$E_{\text{cyc,N}}^{\text{Control}}(N) = (0.01788\lceil \log 2(N+1) \rceil + 0.4293) \times 10^{-3}. \tag{6.34}$$

The proposed architecture recommends always using $w_{\text{s}} = 128$ and $w_{\text{p}} = 24$, except when $N \leq 128$. In this case, that the sliding window technique is not required and the situation it is equivalent to $w_{\text{s}} = N$ and $w_{\text{p}} = 0$ for the design. However, this exception does not affect the controller's energy consumption, according to the simulation results. For example, for $N = 240$ (WiMax) [191], $E_{\text{cyc}}^{\text{Control}}$ is 5.925×10^{-4} nJ/Clock Cycle when using the sliding window technique and 5.9125×10^{-4} nJ/Clock Cycle, otherwise.

Since parameter N only affects the bit counter in the controller, its impact on $E_{\text{cyc}}^{\text{Control}}$ can be considered to be independent from that of the other parameters. However, k, m and n have a more intricate effect on $E_{\text{cyc}}^{\text{Control}}$. Unlike N, their effects on the controller's implementation are not independent from each other. As a result, their combined effect on the controller's energy consumption cannot be modelled independently. An estimation method based on simulation results is proposed estimating $E_{\text{cyc}}^{\text{Control}}$ as functions of the parameters k, m and n. The calculation is based on four groups of simulation results. Here, $f = 400$ MHz, $v = 1.2$ V, $N = 1024$, $w_{\text{s}} = 128$, and $w_{\text{p}} = 24$ are assumed in these results. Table 6.3 provides the first group of results, which were obtained by increasing

k, while maintaining $m = 1$ and $n = 1$. The second group of results of Table 6.4 show

k	1	2	3	4
$E_{\text{cyc}}^{\text{Control}}$	6.255×10^{-4}	6.4075×10^{-4}	6.6×10^{-4}	6.9725×10^{-4}
k	5	6	7	8
$E_{\text{cyc}}^{\text{Control}}$	7.2475×10^{-4}	7.3625×10^{-4}	7.545×10^{-4}	7.815×10^{-4}

TABLE 6.3: $E_{\text{cyc}}^{\text{Control}}$ (nJ/Clock Cycle) simulation results of k increasing, when $m = 1$ and $n = 1$.

the effect of increasing m, while maintaining $k = 1$ and $n = 1$. The third group of

m	1	2	3	4
$E_{\text{cyc}}^{\text{Control}}$	6.255×10^{-4}	6.3275×10^{-4}	6.3725×10^{-4}	6.675×10^{-4}
m	5	6	7	8
$E_{\text{cyc}}^{\text{Control}}$	6.21×10^{-4}	6.39×10^{-4}	6.3775×10^{-4}	6.3225×10^{-4}

TABLE 6.4: $E_{\text{cyc}}^{\text{Control}}$ (nJ/Clock Cycle) simulation results of m increasing, when $k = 1$ and $n = 1$.

results of Table 6.5, were obtained by increasing n, while maintaining $k = 1$ and $m = 1$. Finally, the fourth group of results of Table 6.6, were obtained for $k = m = n$, with all

n	1	2	3	4
$E_{\text{cyc}}^{\text{Control}}$	6.255×10^{-4}	6.205×10^{-4}	6.145×10^{-4}	6.39×10^{-4}
n	5	6	7	8
$E_{\text{cyc}}^{\text{Control}}$	6.3325×10^{-4}	6.2925×10^{-4}	6.2825×10^{-4}	6.4025×10^{-4}

TABLE 6.5: $E_{\text{cyc}}^{\text{Control}}$ (nJ/Clock Cycle) simulation results of n increasing, when $k = 1$ and $m = 1$.

of them increasing at the same time. To estimate $E_{\text{cyc}}^{\text{Control}}$ for a specific combination

$k = m = n$	1	2	3	4
$E_{\text{cyc}}^{\text{Control}}$	6.255×10^{-4}	6.3925×10^{-4}	6.8625×10^{-4}	7.0175×10^{-4}
$k = m = n$	5	6	7	8
$E_{\text{cyc}}^{\text{Control}}$	7.465×10^{-4}	7.5925×10^{-4}	7.7225×10^{-4}	7.7795×10^{-4}

TABLE 6.6: $E_{\text{cyc}}^{\text{Control}}$ (nJ/Clock Cycle) simulation results of $k = m = n$.

of k, m and n, firstly, $E_{\text{cyc},k}^{\text{Control}}(k)$, $E_{\text{cyc},m}^{\text{Control}}(m)$, $E_{\text{cyc},n}^{\text{Control}}(n)$ and $E_{\text{cyc},kmn}^{\text{Control}}(kmn)$ are used to define the results in Table 6.3 to Table 6.6. For a certain specification of $\{k, m, n\}$, $kmn = \min(k, m, n)$ is defined and $E_{\text{cyc}}^{\text{Control}}$ is estimated as Equation 6.35.

$$\begin{aligned} E_{\text{cyc},k,m,n}^{\text{Control}}(k,m,n) = {} & E_{\text{cyc},kmn}^{\text{Control}}(kmn) + \big(E_{\text{cyc},k}^{\text{Control}}(k) - E_{\text{cyc},k}^{\text{Control}}(kmn)\big) \\ & + \big(E_{\text{cyc},m}^{\text{Control}}(m) - E_{\text{cyc},m}^{\text{Control}}(kmn)\big) + \big(E_{\text{cyc},n}^{\text{Control}}(n) - E_{\text{cyc},n}^{\text{Control}}(kmn)\big). \quad (6.35) \end{aligned}$$

Combining the models for N, k, m, n and v allows $E_{\text{cyc}}^{\text{Control}}$ to be estimated as

$$E_{\text{cyc}}^{\text{Control}} = \frac{v^2}{1.2} E_{\text{cyc},k,m,n}^{\text{Control}}(k, m, n) + 0.01788(\lceil \log_2(N+1) \rceil - 11) \times 10^{-3}. \tag{6.36}$$

Using $E_{\text{cyc}}^{\text{Control}}$ and T_{e}, the energy efficiency of the controller in nJ/LLR can be calculated as

$$E_{\text{e}}^{\text{Control}} = E_{\text{cyc}}^{\text{Control}} \times T_{\text{e}}. \tag{6.37}$$

To verify the model, the estimation results and the simulation results of $E_{\text{e}}^{\text{Control}}$ for four example applications are given in Table 6.7, where $f = 400$ MHz and $v = 1.2$ V.

Application	WiMax [191]	CDMA2000 [192]	LTE [193]	Deep space communication [194]
Simulation result	0.0285	0.0241	0.0224	0.0247
Estimation result	0.0286	0.0243	0.0224	0.0246

Table 6.7: Comparison of the estimation results and the simulation results of $E_{\text{e}}^{\text{Control}}$ (nJ/LLR) of example applications.

6.3.5 Energy estimation of LUT-Log-BCJR decoders and results validation

Based on the results in Section 6.3.3 and 6.3.4, the overall energy efficiency of the LUT-Log-BCJR decoder $E_{\text{e}}^{\text{BCJR}}$ can be calculated as

$$E_{\text{e}}^{\text{BCJR}} = E_{\text{e}}^{\text{Datapath}} + E_{\text{e}}^{\text{Control}}. \tag{6.38}$$

To validate the proposed framework, two LUT-Log-BCJR decoders for the two different turbo codes of [106] were implemented using the generalised architecture. Post-layout simulations are performed for obtaining the post-layout energy estimation results. Design-I has the specification of $k = 1$, $m = 3$ and $n = 1$. By contrast, design-II has the specification of $k = 1$, $m = 2$ and $n = 1$. Both design have block lengths of $N = 6144$. In addition, $x = 4$, $y = 7$, $z = 2$, $w_{\text{s}} = 128$, $w_{\text{p}} = 24$, $f = 400$ MHz and $v = 1.2$ V are assumed, in both cases. The simulated energy consumption and the estimated energy consumption are compared in Table 6.8. The results show that the estimated results are with more than 98% of the post-layout simulation results.

	Design-I	Design-II
Simulation result	0.3208 nJ/LLR	0.154 nJ/LLR
Estimation result	0.3156 nJ/LLR	0.1516 nJ/LLR

Table 6.8: Comparison of the estimation results and the simulation results of $E_{\text{e}}^{\text{BCJR}}$ (nJ/LLR) of the example designs.

6.3.6 Memories

As discussed in Section 6.1, the proposed framework aims to give energy estimation results that closely predict the post-layout simulation results. For the memory modules, the databook provided by the standard library developer [195] provides specifications which allow the energy consumption to be calculated. According to the TSMC 90 nm databook [195], the power consumption of a particular memory module can be estimated by considering the accessing rate a in units of accesses per clock cycle, the clock frequency f and the supply voltage v. According to [195], writing and reading operations are considered to have the same energy consumption. In the standard cell library, the power consumption of the Static Random-Access Memory (SRAM) used in the architecture can be estimated using the reference table of [195]. In the reference table, for memory blocks with various word-widths and word-lengths, typical accessing power consumption p_a and leakage current I_l are given. The power consumption P_a can be used to calculate the dynamic energy consumption when the memory is being accessed. The leakage current I_l can be used to calculate the static energy consumption of the memory when it is idle. As is typical for standard cells, the reference table only provides the reference data for several typical supply voltages. However, the voltage scaling method used for the previous model can still applied. In this case, the typical specifications for TSMC 90 nm SRAM operating at 1.2 V are used.

The memories required by the proposed architecture are divided into three types, the a priori LLR memory, the extrinsic LLR memory and the metric memory, as shown in Figures 6.2 and 6.3. The accessing rate a of these three memories types are defined as a_a, a_e and a_m, respectively. The calculation of these values can be derived from Figure 6.6 and 6.7, as

$$a_\mathrm{a} = \frac{2w_\mathrm{s} + w_\mathrm{p}}{w_\mathrm{s}(T_\mathrm{forward} + T_\mathrm{backward}) + w_\mathrm{p} T_\mathrm{prebackward}}. \tag{6.39}$$

$$a_\mathrm{e} = \frac{w_\mathrm{s}}{w_\mathrm{s}(T_\mathrm{forward} + T_\mathrm{backward}) + w_\mathrm{p} T_\mathrm{prebackward}}. \tag{6.40}$$

$$a_\mathrm{m} = \frac{4w_\mathrm{s}}{w_\mathrm{s}(T_\mathrm{forward} + T_\mathrm{backward}) + w_\mathrm{p} T_\mathrm{prebackward}}. \tag{6.41}$$

As a result, the average power consumption of a memory block can be calculated as

$$p^\mathrm{M} = \frac{v^2}{1.2^2}(fvp_\mathrm{a}a + vI_\mathrm{l}) \times 10^{-3}, \tag{6.42}$$

where $a \in \{a_\mathrm{a}, a_\mathrm{e}, a_\mathrm{m}\}$. The energy consumption per Clock Cycle is given by

$$E^\mathrm{M}_\mathrm{cyc} = \frac{(\frac{v^3}{1.2^2} fp_\mathrm{a}a + vI_\mathrm{l}) \times 10^{-3}}{f}. \tag{6.43}$$

For example, for the standard 128×64 bit memory block of the TSMC 90 nm library [195], when $v = 1.2$ V, $p_\mathrm{a} = 33.65$ μA/mW and $I_\mathrm{l} = 0.887$ μA. Figure 6.20 gives both the simulation results and the estimation results, in order to verify this memory energy model. Similar to the datapath energy efficiency, the memory block energy efficiency

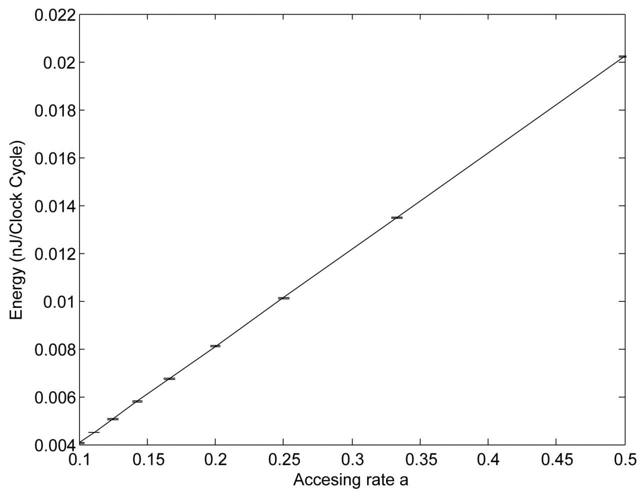


FIGURE 6.20: The error bar result of 128 × 64 bits memory, $v = 1.2$ V.

$E_{\mathrm{e}}^{\mathrm{M}}$ can be calculated as

$$E_{\mathrm{e}}^{\mathrm{M}} = T_{\mathrm{e}} \times E_{\mathrm{cyc}}^{\mathrm{M}}. \tag{6.44}$$

To estimate the memory's energy efficiency in the Log-LUT-BCJR decoder, the memory specification must be chosen, according to the discussion in Section 6.2.3. Then Equation 6.39 to 6.44 can be used to estimate the energy efficiency.

6.3.7 Interleaver

The interleaver is typically designed independently from the turbo code. As a result, it is not possible to devise a general model for estimating the energy consumption of the interleaver in a turbo decoder, owing to the many different types of interleaver that can be used. However, as mentioned in Chapter 5, in the proposed architecture, the rate that the interleaver is required to generate addresses is very low. As a result, it is straightforward to implement a low complexity interleaver, having an insignificant energy consumption compared to the turbo decoder. Therefore, a less accurate estimation of the interleaver's energy consumption does not significantly impact the overall estimation of the proposed framework. To simplify the energy estimation of the interleaver, further assumptions for the employed framework may be chosen. Firstly, the interleaver may be limited to supporting only a single length. Secondly, the LTE interleaver design may be chosen for the estimation. These assumptions allow a relatively simple energy model to be obtained for the interleaver and are reasonable for WSN applications. The simulation and estimation results presented in this section will show that due to the low address generating speed requirement in the proposed architecture, the energy consumption of the interleaver is low and insignificant in the overall energy estimation.

The energy consumption of an LTE interleaver supporting only a single length is affected by interleaver length (which is equal to the block length of the turbo code), N, and the address generation rate g (which is measured in the number of addresses that are generated per clock cycle, as determined by the controller design of Section 6.2.4). The simulation results show that the effect of the interleaver length N on the interleaver's energy consumption is insignificant. The simulated energy consumptions of interleavers having different interleaver lengths, including 512, 1024, 2048 and 4096 bits have the standard deviations that are no more than 0.8% of the average results, irrespective of the address generation rate. Therefore, the proposed energy model for the interleaver's energy consumption is based on only the address generation rate g. The average values of the simulated energy consumptions of interleavers having lengths of 512, 1024, 2048 and 4096 bits, are shown in Figure 6.21, for various values of g. A linear fitting result

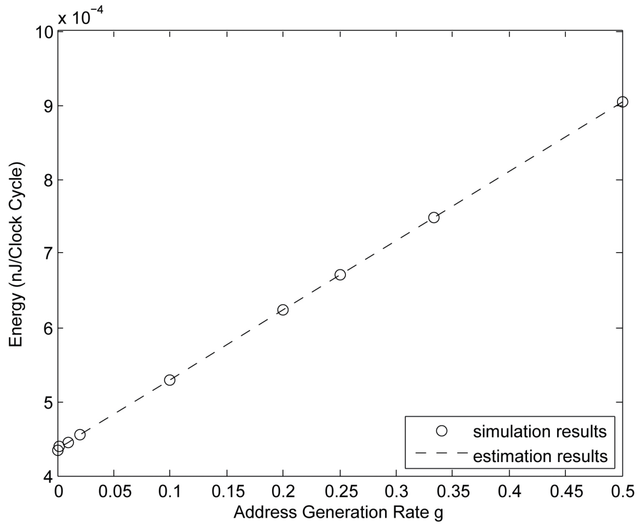


FIGURE 6.21: Energy consumptions of the interleaver with different address generation rates g, where $V = 1.2$ V and $f = 400$ MHz.

is also shown in the figure. Similarly to the modelling methods that were proposed for the register banks and the CU, the energy consumption of the interleaver in nJ/Clock Cycle can be estimated as

$$E_{\text{cyc}}^{\text{Interleaver}} = \frac{v^2}{1.2^2}(0.9382g + 0.4359) \times 10^{-3}. \tag{6.45}$$

Based on the description of Section 6.2.4, in each forward, pre-backward and backward recursion loop of the LUT-Log-BCJR decoder, the interleaver is required to generate one memory address. As a result, the address generation rate can be calculated using the time consumption of the three recursion loops, according to

$$g = \frac{2w_{\text{s}} + w_{\text{p}}}{w_{\text{s}}(T_{\text{forward}} + T_{\text{backward}}) + w_{\text{p}}T_{\text{prebackward}}}. \tag{6.46}$$

Finally, using $E_{\mathrm{cyc}}^{\mathrm{Interleaver}}$ and T_{e}, the energy efficiency of the interleaver in nJ per extrinsic LLR can be calculated as

$$E_{\mathrm{e}}^{\mathrm{Interleaver}} = E_{\mathrm{cyc}}^{\mathrm{Interleaver}} \times T_{\mathrm{e}}. \tag{6.47}$$

In Table 6.9, the same designs of Table 6.8 are chosen to demonstrate the insignificance of the interleaver's energy consumption. The estimation results of the energy consumed by the LUT-Log-BCJR decoders $E_{\mathrm{e}}^{\mathrm{BCJR}}$, the memories $E_{\mathrm{e}}^{\mathrm{Mem}}$ and the interleavers $E_{\mathrm{e}}^{\mathrm{Interleaver}}$, per extrinsic LLR are given.

	Design-I	Design-II
$E_{\mathrm{e}}^{\mathrm{BCJR}}$	0.3156 nJ/LLR	0.1516 nJ/LLR
$E_{\mathrm{e}}^{\mathrm{Mem}}$	0.2761 nJ/LLR	0.276 nJ/LLR
$E_{\mathrm{e}}^{\mathrm{Interleaver}}$	0.0166 nJ/LLR	0.0148 nJ/LLR

Table 6.9: The estimation results of $E_{\mathrm{b}}^{\mathrm{BCJR}}$, $E_{\mathrm{e}}^{\mathrm{Mem}}$ and $E_{\mathrm{e}}^{\mathrm{Interleaver}}$ (nJ/LLR) of the example designs.

6.3.8 Energy estimation of the turbo decoders and results validation

Using the energy estimation models for the LUT-Log-BCJR decoder and the memories, the energy estimation of a complete turbo decoder can be performed. As implied in Figure 6.3, the two LUT-Log-BCJR decoders employed by a turbo decoder are typically identical. As a result, the iteration decoding process between them can be performed by one LUT-Log-BCJR decoder in practice. The energy consumption of a LUT-Log-BCJR decoder is defined as $E_{\mathrm{e}}^{\mathrm{BCJR}}$ in Section 6.3.5, which is the energy consumed by the LUT-Log-BCJR decoder for decoding one extrinsic LLR. Therefore, the energy required by of the LUT-Log-BCJR decoder to decode one bit information $E_{\mathrm{b}}^{\mathrm{BCJR}}$ can be estimated as

$$E_{\mathrm{b}}^{\mathrm{BCJR}} = 2I \times E_{\mathrm{e}}^{\mathrm{BCJR}}, \tag{6.48}$$

where I is the number of iterations performed in the turbo decoding process. Similarly,

$$E_{\mathrm{b}}^{\mathrm{Interleaver}} = 2I \times E_{\mathrm{e}}^{\mathrm{Interleaver}}, \tag{6.49}$$

To estimate the memories' energy consumption, the LLR memories in the turbo decoding scheme of Figure 6.3 are divided into three groups. The a priori LLR memories with indices 1 to k are defined as Group-1. The a priori LLR memories with indices $k + 1$ to $k + n$ are defined as Group-2. Finally, the extrinsic LLR memories with indices 1 to k are defined as Group-3.

As shown in Figure 6.3, Group-1 is shared by the two LUT-Log-BCJR decoders. Regardless of which one is working, the behaviour of Group-1 memories is exactly the same. Therefore, the accessing rate of Group-1 memories a_{g1} is same as a_{a}, as discussed in Section 6.3.6. This is given by

$$a_{\mathrm{g1}} = \frac{2w_{\mathrm{s}} + w_{\mathrm{p}}}{w_{\mathrm{s}}(T_{\mathrm{forward}} + T_{\mathrm{backward}}) + w_{\mathrm{p}}T_{\mathrm{prebackward}}}. \tag{6.50}$$

Using Equation 6.50, the energy consumed by each of the k memory blocks in Group-1, $E_{\mathrm{e}}^{\mathrm{g1}}$, can be calculated using Equation 6.44. When performing I decoding iterations, the energy consumed per bit by Group-1 is given by

$$E_{\mathrm{b}}^{\mathrm{g1}} = 2I \times k \times E_{\mathrm{e}}^{\mathrm{g1}}. \tag{6.51}$$

For Group-2, the memories are only in use when the corresponding LUT-Log-BCJR decoder is operating, so their accessing rate is only half of a_{a}, according to

$$a_{\mathrm{g2}} = \frac{2w_{\mathrm{s}} + w_{\mathrm{p}}}{2(w_{\mathrm{s}}(T_{\mathrm{forward}} + T_{\mathrm{backward}}) + w_{\mathrm{p}}T_{\mathrm{prebackward}})}. \tag{6.52}$$

However, there is another identical group of memories that are employed by the other LUT-Log-BCJR decoder, as shown in Figure 6.3. The energy consumption per bit of both groups can be estimated as

$$E_{\mathrm{b}}^{\mathrm{g2}} = 4I \times n \times E_{\mathrm{e}}^{\mathrm{g2}}. \tag{6.53}$$

The Group-3 memories not only perform as the extrinsic LLR memories for one LUT-Log-BCJR decoder, but also perform as the a priori LLR memories for the other LUT-Log-BCJR decoder. Therefore, their accessing rate is the average of a_{a} and a_{e}, which is given by

$$a_{\mathrm{g3}} = \frac{3w_{\mathrm{s}} + w_{\mathrm{p}}}{2(w_{\mathrm{s}}(T_{\mathrm{forward}} + T_{\mathrm{backward}}) + w_{\mathrm{p}}T_{\mathrm{prebackward}})}. \tag{6.54}$$

Similar to Group-2, there are two identical groups of extrinsic LLR memories corresponding to each LUT-Log-BCJR decoder, as shown in Figure 6.3. The energy consumption of both groups can be estimated as

$$E_{\mathrm{b}}^{\mathrm{g3}} = 4I \times k \times E_{\mathrm{e}}^{\mathrm{g3}}. \tag{6.55}$$

The situation for the metric memory is similar to that of the LUT-Log-BCJR decoder. The energy consumption of a turbo decoder's the metric memory is given by

$$E_{\mathrm{b}}^{\mathrm{m}} = 2I \times E_{\mathrm{e}}^{\mathrm{m}}, \tag{6.56}$$

where $E_{\mathrm{e}}^{\mathrm{m}}$ can be calculated using Equation 6.44.

The total energy consumption of the memories in the turbo decoder $E_{\mathrm{b}}^{\mathrm{Mem}}$ is the sum of all the memories' energy consumptions, which is given by

$$E_{\mathrm{b}}^{\mathrm{Mem}} = E_{\mathrm{b}}^{\mathrm{g1}} + E_{\mathrm{b}}^{\mathrm{g2}} + E_{\mathrm{b}}^{\mathrm{g3}} + E_{\mathrm{b}}^{\mathrm{m}}. \tag{6.57}$$

The turbo decoder's total energy consumption per bit can be calculated as

$$E_{\mathrm{b}}^{\mathrm{Turbo}} = E_{\mathrm{b}}^{\mathrm{BCJR}} + E_{\mathrm{b}}^{\mathrm{Mem}} + E_{\mathrm{b}}^{\mathrm{Interleaver}}. \tag{6.58}$$

Finally, for the same designs used in Section 6.3.5, the turbo decoders' simulated energy consumptions and the estimated energy consumptions per information bit (nJ/bit) are compared in Table 6.8. The results show that the estimated results are within 95% of the post-layout simulation results.

	Design-I	Design-II
Simulation result	6.3955 nJ/bit	4.7686 nJ/bit
Estimation result	6.0826 nJ/bit	4.4244 nJ/bit

TABLE 6.10: Comparison of the estimation results and the simulation results of the energy consumptions (nJ/bit) of the example designs.

6.4 Holistic design method of turbo codes for energy-constrained communication systems

In this section, the proposed energy estimation framework is applied to investigate the overall energy efficiency of different turbo codes. By considering both the transmission energy consumption $E_{\mathrm{b}}^{\mathrm{tx}}$ and the decoding process energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$, a holistic turbo code design method for energy-constrained applications, such as WSNs, is demonstrated using the proposed architecture. Compared with the conventional design method, this method allows the turbo code to be used for the purpose of reducing the overall energy consumption of a wireless communication system. The proposed method focuses on the encoder and decoder design of the turbo code.

The design procedure consists of two parts. In the first part, BER simulations and the transmission power model are used to estimate the required transmission energy consumption $E_{\mathrm{b}}^{\mathrm{tx}}$. In the second part, the energy estimation framework proposed in Section 6.3 is used to estimate the required decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$. Finally, the overall energy consumption can be obtained by combining $E_{\mathrm{b}}^{\mathrm{tx}}$ and $E_{\mathrm{b}}^{\mathrm{pr}}$ and used to evaluate the energy efficiency for different turbo codes.

As mentioned in Section 6.1, the objective of the proposed design method is to determine the particular parametrisation of the turbo code design that optimises the overall energy consumption of the system. The component encoder of the design is specified by the parameters k, m and n, as well as the generator polynomial. Further parameters may be considered when variations of the turbo code are considered for the design. For example, when multiple transmissions of the encoded bits are introduced, the coding rate R is another parameter that should be considered individually. Furthermore, when Multiple-Component Turbo Codes (MCTCs) [106] are considered, the number of parallel component encoders that are employed becomes a parameter of the scheme. In addition, the number of decoding iterations performed also affects the decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ and the required minimum transmission energy consumption $E_{\mathrm{b}}^{\mathrm{tx}}$ significantly. To present the holistic design method, a particular design example [106] is considered in the following sections. The approach adopted here is in contrast to that of [106] where an MCTC is designed by comparing different parameterisations of a turbo code scheme. Using the conventional design method, the decisions in [106] were based on EXIT and BER performance alone. In this chapter, by using the holistic design method, an overall energy optimised design is obtained for the same scheme.

6.4.1 Decoding energy estimation

As mentioned above, a selection of different Twin-Component Turbo Codes (TCTCs) and MCTCs were investigated in [106] using the conventional design method, based on EXIT chart and BER analysis alone. The MCTC encoding scheme is shown in Figure 6.22. In contrast to the TCTC, an MCTC encoder employs more than two compo-

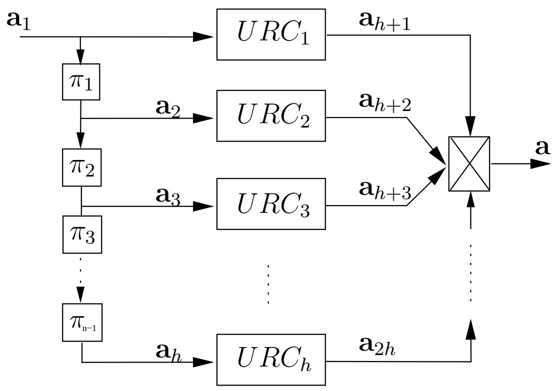


FIGURE 6.22: The MCTC encoder schematic [106].

nent encoders in parallel. As shown in Figure 6.22, in an MCTC employing h component encoders, the uncoded sequence $\mathbf{a}_1$ and its interleaved copies $\{\mathbf{a}_2, \mathbf{a}_3, ...\mathbf{a}_h\}$ are encoded by Unity Rate Coding (URC) component encoders in order to generate the encoded outputs $\{\mathbf{a}_{h+1}, \mathbf{a}_{h+2}, ...\mathbf{a}_{2h}\}$. Hence, the coding rate is given by $R = 1/h$. Figure 6.23

gives the corresponding MCTC decoder schematic, which comprises h corresponding Log-BCJR decoders.

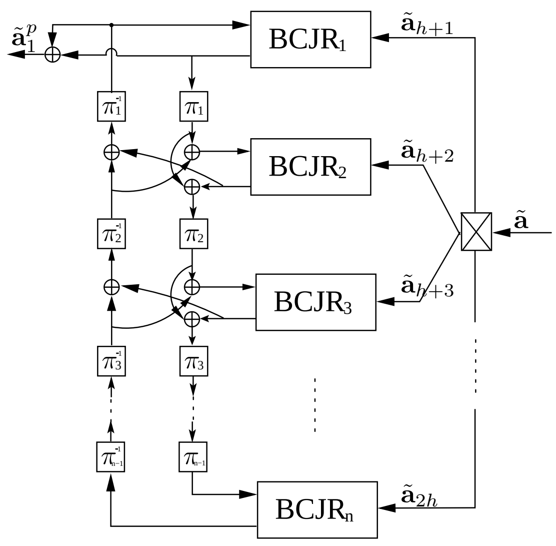


FIGURE 6.23: The MCTC decoder schematic [106].

Moreover, owing to the lack of an energy estimation measure, computational complexity is chosen for comparing different code designs, in [106], as well as in many similar previous publications. In [106], the computational complexity C is defined as

$$C = 2^m * B, \tag{6.59}$$

where m is the number of the memory elements in the component encoders. However, the proposed energy estimation framework shows that the complexity measure considered in the conventional design method does not offer fair comparisons, since it cannot accurately represent the energy consumption of different turbo code designs.

In [106], twelve different turbo code schemes were investigated, including four MCTCs, four systematic TCTCs and four non-systematic TCTCs, as listed in Table 6.11. For each scheme, three different values of B were considered. Using BER simulations, the performance of the chosen schemes were compared. An evaluation of the schemes was then made, based on the BER results and the complexity C. The parametrisation of the schemes chosen in [106] includes parameters k, m, n, B, the coding rate R and the generator polynomial. All the schemes have the same block length of $N = 2048$ bits. Note that normally, for systematic codes, $R = \frac{k}{n+n}$, and for non-systematic codes, $R = \frac{k}{n}$. However, R can be further increased by retransmitting some or all of the outputs generated by the component encoders. Different coding rates R for the same code design yield different performance and transmission energy consumption. in order to estimate the decoding energy consumption of the coding schemes when employing the generalised

[3]The requried Signal-to-Noise Ratio (SNR) in dB of the chosen schemes at BER $= 10^{-5}$.

candidate	k	m	n	R	B	polynomial	$E_{\mathrm{b}}^{\mathrm{pr}}$ (nJ/bit)	s (Mb/s)	C	SNR[3]
sysTCTC-1	1	3	1	1/3	3	$(17, 15)_o$	1.825	4.003	24	-0.2
sysTCTC-1	1	3	1	1/3	6	$(17, 15)_o$	3.65	2.001	48	-2.1
sysTCTC-1	1	3	1	1/3	12	$(17, 15)_o$	7.299	1.001	96	-2.5
sysTCTC-2	1	3	1	1/4	3	$(17, 15)_o$	1.825	4.003	24	-2
sysTCTC-2	1	3	1	1/4	6	$(17, 15)_o$	3.65	2.001	48	-3.3
sysTCTC-2	1	3	1	1/4	12	$(17, 15)_o$	7.299	1.001	96	-4
sysTCTC-3	1	3	1	1/5	3	$(17, 15)_o$	1.825	4.003	24	-3.3
sysTCTC-3	1	3	1	1/5	6	$(17, 15)_o$	3.65	2.001	48	-4.8
sysTCTC-3	1	3	1	1/5	12	$(17, 15)_o$	7.299	1.001	96	-5.2
sysTCTC-4	1	3	1	1/6	3	$(17, 15)_o$	1.825	4.003	24	-4.1
sysTCTC-4	1	3	1	1/6	6	$(17, 15)_o$	3.65	2.001	48	-5.6
sysTCTC-4	1	3	1	1/6	12	$(17, 15)_o$	7.299	1.001	96	-6.2
TCTC-1	1	3	1	1/3	3	$(10, 17)_o$	1.825	4.003	24	4.1
TCTC-1	1	3	1	1/3	6	$(10, 17)_o$	3.65	2.001	48	0.7
TCTC-1	1	3	1	1/3	12	$(10, 17)_o$	7.299	1.001	96	-0.5
TCTC-2	1	3	1	1/4	3	$(10, 17)_o$	1.825	4.003	24	2.6
TCTC-2	1	3	1	1/4	6	$(10, 17)_o$	3.65	2.001	48	-1
TCTC-2	1	3	1	1/4	12	$(10, 17)_o$	7.299	1.001	96	-1.5
TCTC-3	1	3	1	1/5	3	$(10, 17)_o$	1.825	4.003	24	0.6
TCTC-3	1	3	1	1/5	6	$(10, 17)_o$	3.65	2.001	48	-2.2
TCTC-3	1	3	1	1/5	12	$(10, 17)_o$	7.299	1.001	96	-2.6
TCTC-4	1	3	1	1/6	3	$(10, 17)_o$	1.825	4.003	24	-0.2
TCTC-4	1	3	1	1/6	6	$(10, 17)_o$	3.65	2.001	48	-3.3
TCTC-4	1	3	1	1/6	12	$(10, 17)_o$	7.299	1.001	96	-4.3
MCTC-1	1	2	1	1/3	6	$(4, 7)_o$	2.655	2.274	24	1
MCTC-1	1	2	1	1/3	12	$(4, 7)_o$	5.309	1.137	48	-2
MCTC-1	1	2	1	1/3	24	$(4, 7)_o$	10.619	0.568	96	-3
MCTC-2	1	2	1	1/4	6	$(2, 3)_o$	2.655	2.274	24	-3
MCTC-2	1	2	1	1/4	12	$(2, 3)_o$	5.309	1.137	48	-4
MCTC-2	1	2	1	1/4	24	$(2, 3)_o$	10.619	0.568	96	-4
MCTC-3	1	2	1	1/5	6	$(2, 3)_o$	2.655	2.274	24	-4
MCTC-3	1	2	1	1/5	12	$(2, 3)_o$	5.309	1.137	48	-5
MCTC-3	1	2	1	1/5	24	$(2, 3)_o$	10.619	0.568	96	-6
MCTC-4	1	2	1	1/6	6	$(2, 3)_o$	2.655	2.274	24	-4
MCTC-4	1	2	1	1/6	12	$(2, 3)_o$	5.309	1.137	48	-6
MCTC-4	1	2	1	1/6	24	$(2, 3)_o$	10.619	0.568	96	-7

TABLE 6.11: The chosen turbo code designs.

architecture of Section 6.2, a clock frequency of $f = 400$ MHz is assumed in order to obtain the maximum throughput and a supply voltage $v = 1.2$ V is assumed, since this is the standard setting of the TSMC 90 nm process technology. The decoding throughput of the decoder s in Mb/s can be obtained as

$$s = \frac{f}{T_\mathrm{e} \times B}, \tag{6.60}$$

where T_e is the average number of clock cycles for a Log-BCJR decoder to calculate each extrinsic LLR, as discussed in Section 6.2.4. The specifications and the estimation results of all the chosen schemes are given in Table 6.11.

6.4.2 Transmission energy estimation

In this section, the BER simulation results of the chosen schemes are used to determine the corresponding transmission energy consumptions E_b^tx using a transmission power model. The path loss model given in Equation 5.1 to 5.4 is used to calculate the transmission energy consumption per information bit E_b^tx. The same environment assumptions and system specification of the target scenario of Table 5.2 are applied. The BER simulation results of the chosen schemes [106] are given in Figure 6.24. To

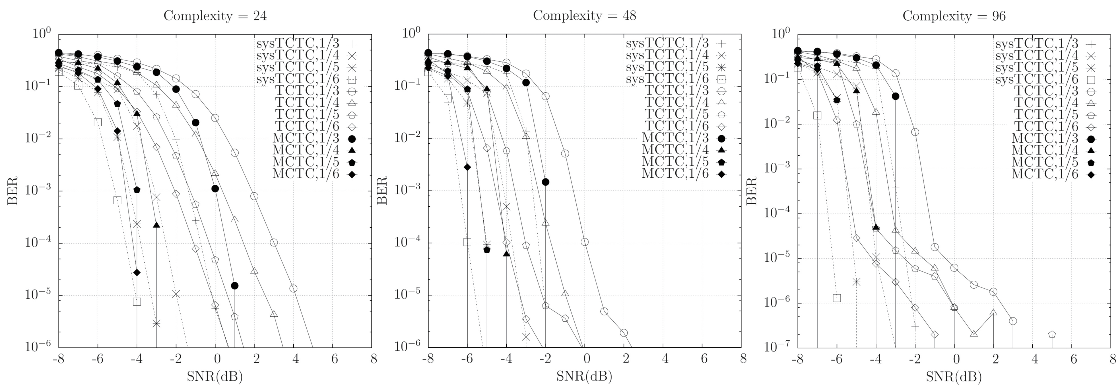


FIGURE 6.24: The BER simulation results of the chosen schemes. ©H. Chen *et al.* 2010 [106]

evaluate the transmission energy, a maximum BER requirement must be specified based on the target application. Here, a BER of 10^{-5} is assumed to be the maximum BER that can be tolerated. The minimum SNRs required to achieve these BERs for each scheme are summarised in Table 6.11. Using this transmission energy model and the results of Table 6.11, the transmission energy consumption E_b^tx of the chosen schemes can be obtained as a function of the transmission distance, as shown in Figure 6.25.

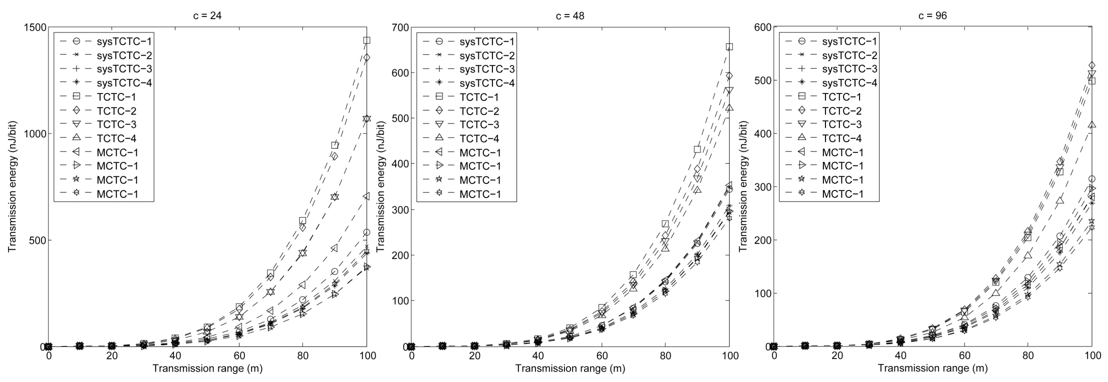


FIGURE 6.25: The transmission energy consumption of the chosen schemes.

6.4.3 Overall energy efficiency analysis

Based on the results of Sections 6.4.1 and 6.4.2, the overall energy consumption $E_\mathrm{b}^\mathrm{tx}+E_\mathrm{b}^\mathrm{pr}$ can be obtained. Figures 6.26, 6.27 and 6.28 provide the $E_\mathrm{b}^\mathrm{tx}+E_\mathrm{b}^\mathrm{pr}$ estimation results of the chosen schemes at transmission ranges of $d = 20, 30$ and 40 m, which are typical transmission ranges of WSNs that are deployed in buildings or small areas.

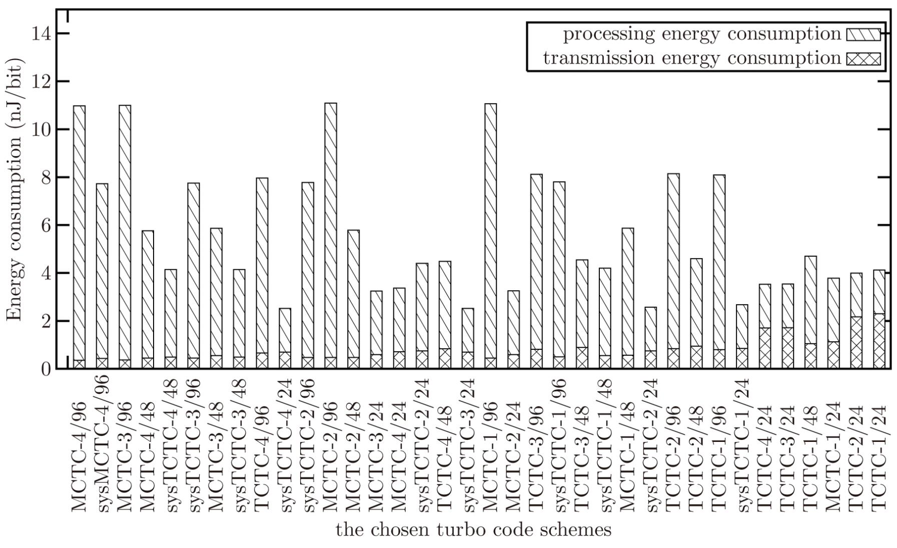


FIGURE 6.26: Overall energy consumption $E_\mathrm{b}^\mathrm{tx}+E_\mathrm{b}^\mathrm{pr}$ of the chosen schemes, when $d = 20$ m.

In Figures 6.26, 6.27 and 6.28, the chosen schemes are referred to using the 'scheme name/complexity' format. The schemes are arranged in descending order of the SNR required to achieve BER $= 10^{-5}$ from left to right, according to the results given in Table 6.11. As shown in Figures 6.26, 6.27 and 6.28, neither the required SNR nor the complexity is correlated with the overall energy consumption. Therefore, the overall energy efficiency of a turbo code scheme cannot be adequately analysed using the conventional methods. However, using the proposed energy estimation framework and path loss model, the overall energy consumption results can be obtained, as shown in Figure 6.26, 6.27 and 6.28.

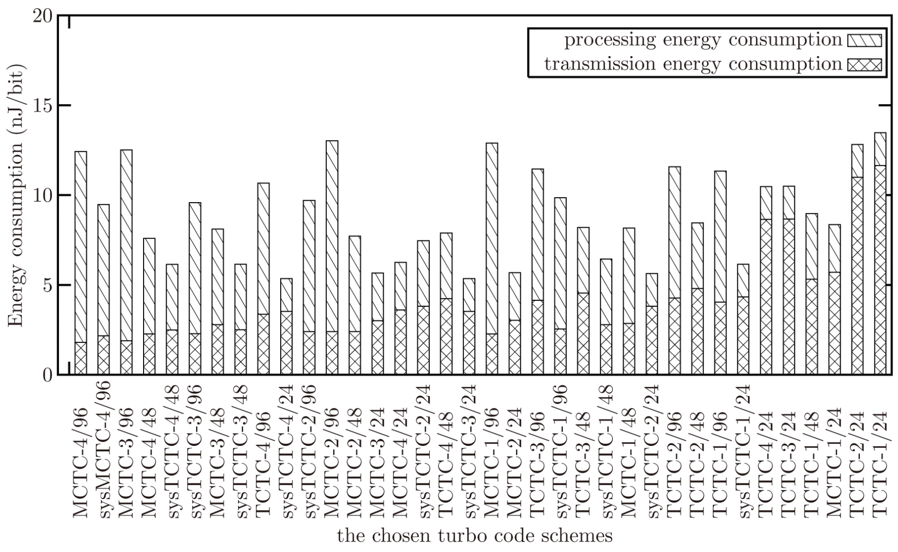


FIGURE 6.27: Overall energy consumption $E_{\text{b}}^{\text{tx}} + E_{\text{b}}^{\text{pr}}$ of the chosen schemes, when $d = 30$ m.

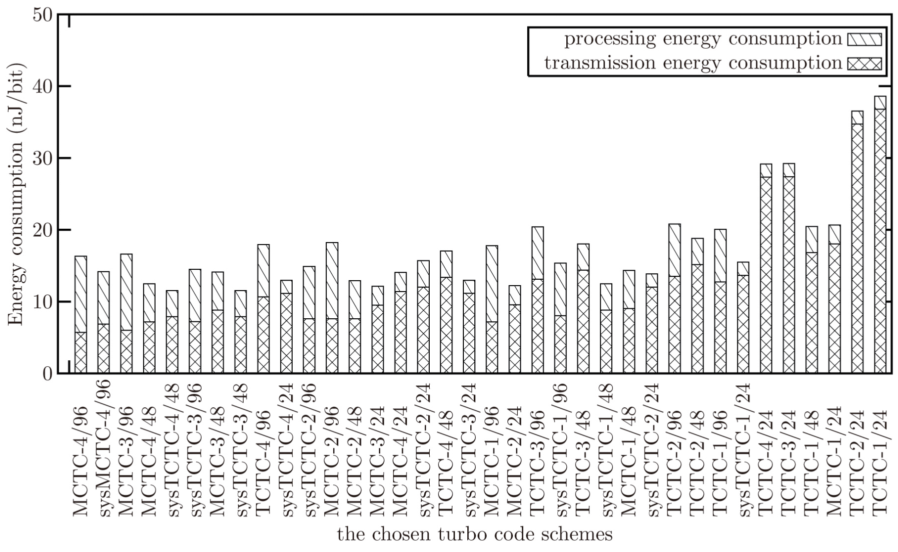


FIGURE 6.28: Overall energy consumption $E_{\text{b}}^{\text{tx}} + E_{\text{b}}^{\text{pr}}$ of the chosen schemes, when $d = 40$ m.

As shown in Figure 6.26, 'sysTCTC-4/24', 'sysTCTC-3/24' and 'sysTCTC-2/24' schemes have the least overall energy consumption among the candidates when $d = 20$ m. When $d = 30$ m, 'sysTCTC-4/24' and 'sysTCTC-3/24' schemes have the least overall energy consumption. Finally, 'sysTCTC-4/48' and 'sysTCTC-3/48' schemes have the least overall energy consumption when $d = 40$ m. However, the overall energy consumption of the 'sysTCTC-4/24' and 'sysTCTC-3/24' schemes are only slightly higher than those of the 'sysTCTC-4/48' and 'sysTCTC-3/48' schemes. As a result, 'sysTCTC-4/24' and 'sysTCTC-3/24' can be considered to be the most energy efficient schemes for transmission ranges between 20 m and 40 m. According to Equation 6.60, these two schemes also have high decoding throughputs of 4.003 Mb/s. However, as shown in Figure 6.26, 6.27 and 6.28, the 'sysTCTC-4/24' scheme consumes more transmission energy and less decoding energy than the 'sysTCTC-3/24' scheme. In general, this distinction does not affect the choice between these two schemes. However, it may be helpful, when there are special constraints imposed on transmission energy consumption or onboard processing energy consumption.

The case study of [106] offers a simple example for the purpose of demonstrating the philosophy of the proposed holistic design method. Many details, such as the assumption of the environment and the WSN system specification included in the analysis, are simplified for avoiding distraction from the demonstration. The proposed design method is capable of helping the designer to optimise a turbo code design in many different aspects. For example, besides the basic parameters of turbo code schemes that were considered in the example, the longest block length N of a turbo code determines the memory requirement of the hardware implementation, which contributes an significant part of the total decoding energy consumption, as shown in Section 6.3.6. The number of decoding iterations performed in the decoding process has a significant effect on both the BER performance and the decoding energy consumption. In addition, the number of hops employed in a multi-hop network determines the average transmission range and the sensor densities. All of these aspects directly affect the transmission and the decoding energy consumption. As a result, the proposed design method can be used to optimise a wide variety of related specifications for the purpose of improving the system energy efficiency.

6.5 Conclusions

In this chapter, the LUT-Log-BCJR architecture of Chapter 5 is generalised so that it can be reconfigured to support any encoder design. A redesigned controller is introduced which is optimised for the general case. The proposed generalised architecture may be reconfigured by simply changing the number of the CUs and the number of registers in the register banks. The behaviour and the decoding time consumption of the architecture is fully predictable based on the redesigned controller.

Secondly, an energy estimation framework based on the LUT-Log-BCJR architecture is proposed that allows the designer to estimate the energy consumption of a turbo code design at an early design stage. The framework fully utilises the reconfigurability and predicability of the generalised architecture. As a result, the input of the framework is the parameters of the encoder design of the turbo code. Using a series of equations, the energy consumption in nJ/bit of the turbo decoder can be calculated.

Finally, using the decoding energy estimation framework and the transmission power model used in Chapter 5, the two parts of the energy consumption that related to the using of turbo code, namely the transmission energy consumption $E_{\mathrm{b}}^{\mathrm{tx}}$ and the decoding energy consumption $E_{\mathrm{b}}^{\mathrm{pr}}$ can be estimated during the code design stage. This estimation cannot be obtained by any conventional design procedure. By taking advantage of this, a holistic design method for turbo code design in energy constrained communication systems is presented. The presented method allows the designer to determine the parameters of a turbo code for the purpose of optimising the overall energy efficiency at early design stage. The method is demonstrated by using it to compare the candidate designs proposed in a conventional turbo code design work [106]. The results show that the tools used in the conventional design method, including the BER performance and computation complexity analysis, are not suitable for analysing the energy efficiency of turbo codes. However, the proposed holistic design method is shown to achieve this goal. The discussion in this section concludes that the proposed design method allows the designer to optimise the parametrisation of a turbo code design in many different aspect for the purpose of improving the system energy efficiency.

Chapter 7

Conclusions and future work

The conclusions provided in this chapter constitute an amalgam of the conclusions drawn from each previous chapter and their logical connections. Some ideas for the possible future work are provided thereafter.

7.1 Conclusions

Chapter 1 reviewed the communication requirements of Wireless Sensor Networks (WSNs). The key issue for these communication systems is the limited energy resources that are typically available on the sensor nodes. Moreover, other communication requirements, such as transmission range, data rate, reliability, accuracy and latency, must be considered when designing an energy efficient communication system. It is challenging to meet all the communication requirements while maintaining a sufficient lifetime for the sensor nodes. Turbo-like codes are then proposed for reducing the transmission energy consumption of the sensor nodes. The advantages and the drawbacks of employing turbo-like codes in channel coding are discussed. Based on the investigation of Chapter 1, the objective of this thesis was stated as developing low-complexity and energy-efficient hardware implementation of turbo codes and to design turbo codes with the overall energy consumption considered in a holistic way. Previous contributions on this topic are reviewed. The chapter is concluded by outlining the novel contributions of the thesis.

In Chapter 2, the background knowledge of this thesis is introduced. Firstly, the basic principle of turbo-like codes is introduced, including the typical encoding and decoding schemes of Serial Concatenated Convolutional Codes (SCCCs) and turbo codes. Secondly, the principle of the Logarithmic Bahl-Cocke-Jelinek-Raviv (Log-BCJR) decoding algorithm was briefly reviewed. The most widely used variations of the algorithm, namely the Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv (LUT-Log-BCJR) and the Maximum Logarithmic Bahl-Cocke-Jelinek-Raviv (Max-Log-BCJR) algorithms were presented. Thirdly, the EXtrinsic Information Transfer (EXIT) chart tool and its use for analysing the performance of turbo-like codes are presented. Finally, the fixed-point number representation system for hardware design and implementation was introduced.

In Chapter 3, a SCCC channel coding scheme is proposed for star WSNs, in order to redistribute the energy consumption from the sensor nodes to the central node for the purpose of extending the system's lifetime. The SCCC is an augmentation of Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 channel coding scheme. It is designed and synthesised using STMicroelectronics 0.12 μm technology for validation. Assuming the implementation of the proposed design as a dedicated module in Chipcon CC2430 sensor nodes, the analysis shows that the overall energy consumption of sensor node transmissions can be reduced by 26.65 - 32.78%. The investigation results of this chapter demonstrated that a significant energy saving could be obtained by adopting sophisticated Error-Correcting Codes (ECCs) in wireless communication systems. Since ECCs encoders typically consume an insignificant amount of energy compared with the rest of the system, a star network is naturally suited for using sophisticated ECC to give an overall energy saving. However, for more complicated network topologies where multi-hop communication is involved, the situation is different. In these cases, the employment of high complexity ECC decoders may be required on the energy-constrained sensor nodes. The trade-off between the energy saving provided by the coding gain of the employed ECC and the extra energy that is consumed by the ECC decoders must be carefully investigated. In addition, the design and implementation of energy-efficient ECC decoders becomes an important consideration, when sophisticated ECC are used to reduce the overall energy consumption of the wireless communication system. In the rest of this thesis, a widely used near Shannon limit ECC, the turbo code [97], is considered for these applications.

As discussed in Chapter 1, to employ turbo codes or turbo-like codes for reducing the overall energy consumption of a WSN, it is important to have an energy efficient implementation of the decoder. As a result, in fixed-point hardware implementations, the word length specifications are highly related to the hardware complexity and hence the energy efficiency. In Chapter 4, a framework based on EXIT chart analysis is proposed to investigate the trade-off between the word length specifications of the fixed-point representation used in a turbo decoder and the decoder's Bit Error Rate (BER) performance. The framework aims to provide a design approach that gives insight into the relationship between the fixed-point specification and the decoders' performance. This is in contrast to the conventional BER analysis solution for avoiding energy wastage by employing a redundant and excessive word length in fixed-point hardware implementations.

In Chapter 5, the turbo decoder's effect on the overall energy efficiency of a wireless communication system is further investigated. The unsuitability of conventional turbo decoder architectures for WSNs is discussed. Motivated by this, a low complexity and energy-efficient LUT-Log-BCJR decoder architecture is proposed, specifically for WSN scenarios. The implementation results show that the proposed LUT-Log-BCJR architecture has a significantly lower energy consumption than conventional LUT-Log-BCJR decoder architectures. Furthermore, the proposed architecture provides greater overall energy efficiency than either the conventional LUT-Log-BCJR and Max-Log-BCJR decoder architectures when the transmission range exceeds 39 m.

In Chapter 6, based on the comprehensive investigation of energy-efficient turbo decoder design and implementation in the previous chapters, a holistic design method for turbo codes is proposed. This allows the optimisation of the overall energy consumption of the turbo code from the very start of the design. The aim of this method is to help the designer to determine the specifications of turbo codes for energy-constrained applications, such as WSNs. The fundamental approach of this design method is an energy estimation framework for turbo decoders. By employing the proposed LUT-Log-BCJR architecture of Chapter 5, a bottom-up energy estimation framework is proposed. Combining the use of the framework, BER analysis and a path loss model of the transmission power in wireless communication, the overall energy efficiency of a turbo coding system can be evaluated at an early design stage. By including this investigation during the code design procedure, the turbo code can be specifically optimised for a particular scenario with the purpose of improving the overall energy efficiency. In this thesis, the Universal Mobile Telecommunications System (UMTS)/Long Term Evolution (LTE) turbo code used throughout the thesis as an example for demonstrating the proposed analysis methods and hardware architecture for the purpose of giving comparison with typical previous work. All the proposed techniques are generally applied to characterise any turbo code.

7.2 Future work

This thesis has explored different aspects of energy-efficient turbo code design and implementation, including their parameterisations, hardware architecture and design methodology. All of the developed techniques are transferable to Low-Density Parity-Check (LDPC) codes [98]. For example, the EXIT chart analysis based method proposed in Chapter 4 for parameterising the fixed-point hardware implementation is applied for LDPC codes in [196]. Since turbo codes and LDPC codes have similar logarithmic decoding algorithms, the proposed architecture of Chapter 5 and the holistic design method of Chapter 6 can be also applied for LDPC codes.

An LDPC decoder can be described by a Tanner graph, as shown in Figure 7.1. The decoding process is an iterative process between the variable nodes and the check nodes. Each edge between the variable nodes and the check nodes of Figure 7.1 represents a two-way data exchange. During the first half of an iteration, the variable nodes generate

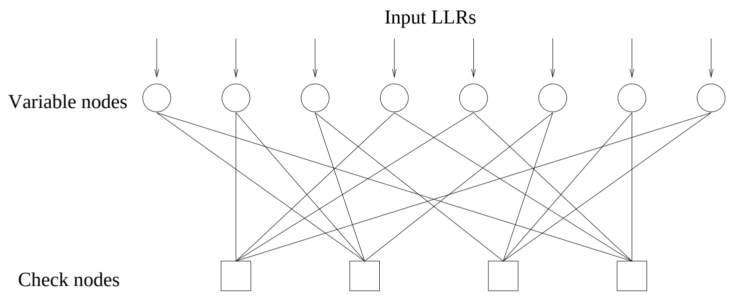


FIGURE 7.1: The Tanner graph of an example LDPC decoder.

extrinsic Logarithmic Likelihood Ratios (LLRs) corresponding to each edge in the figure and passes them to the check nodes. During the second half of an iteration, the check nodes take those extrinsic LLRs as input and generate new extrinsic LLRs corresponding to each edge in the Tanner graph. This is then passed to the variable nodes for use in the next iteration. The output generated at each port of a variable node is obtained as the sum of all the inputs provided at all other ports. Likewise the output generated at each port of each check node is obtained as the min* of all the inputs provided at all other ports [98]. The min* operation of a Min-Sum LDPC decoding algorithm is defined as

$$\min^*(\tilde{p},\tilde{q}) = \text{sign}(\tilde{p})\text{sign}(\tilde{q})\min(|\tilde{p}|,|\tilde{q}|) + \log\left(1+e^{-|\tilde{p}+\tilde{q}|}\right) - \log\left(1+e^{-|\tilde{p}-\tilde{q}|}\right). \quad (7.1)$$

The fundamental reason why the proposed LUT-Log-BCJR decoder architecture can also be transformed to be a Look-Up Table based Min-Sum (LUT-Min-Sum) LDPC decoder is that both of these decoding algorithms consist of only Add-Compare-Select (ACS) operations. Similar to the max* operation, the log components of the min* operation can be realised using a Look-Up Table (LUT). According to [196], fixed-point representation with 2-bit fraction part is recommended. As a result, the two correction functions having the form of $\log\left(1+e^{-\tilde{x}}\right)$ can be implemented by a LUT, according to

$$\log\left(1+e^{-\tilde{x}}\right) \approx \begin{cases} 0.75 & \text{if } \tilde{x}=0 \\ 0.5 & \text{if } \tilde{x}=\{0.25,0.5,0.75\} \\ 0.25 & \text{if } \tilde{x}=\{1,1.25,1.5,1.75,2\} \\ 0 & \text{otherwise} \end{cases}. \quad (7.2)$$

According to the discussion given in Chapter 5, it can be concluded that as long as an algorithm can be decomposed into a sequence of ACS operations, the proposed architecture is capable of performing it.

Using the same design philosophy of Chapter 5 for designing a Calculation Unit (CU) for the max* operation, a new CU can be specifically designed for the min* operation, as shown in Figure 7.2 and 7.3. This new CU is designed based on some modifications of Figure 5.7 and Figure 5.8. Here, MSB(X) is defined as the most significant bit of

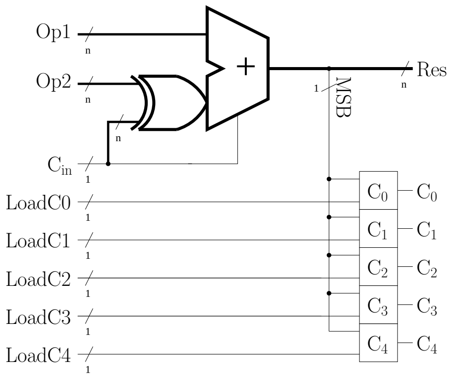


Figure 7.2: ACS unit.

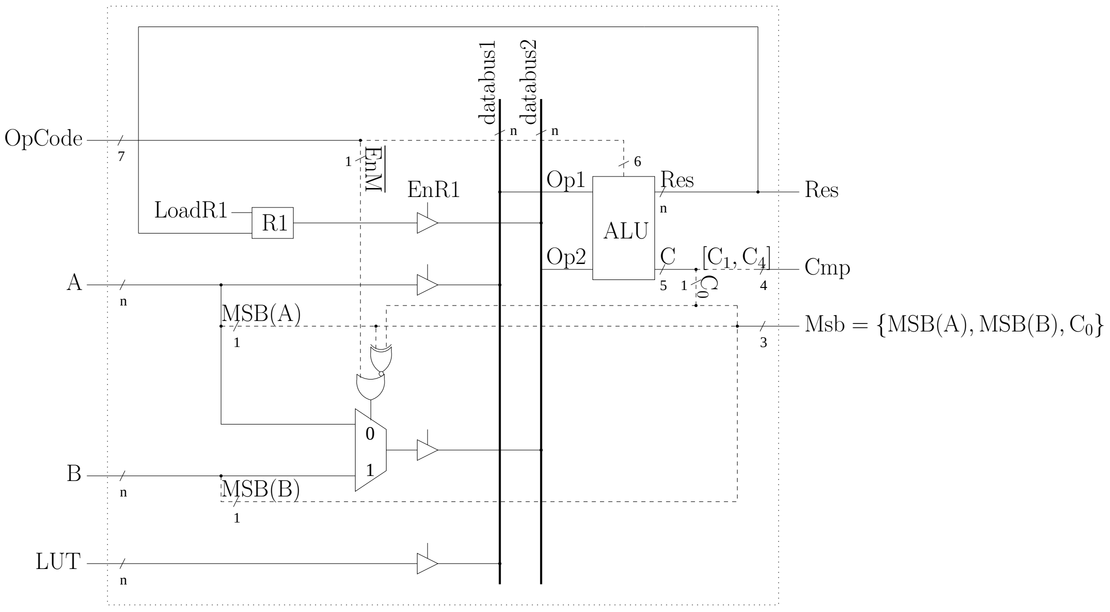


Figure 7.3: Calculation unit.

signal X. There are four input signals and three output signals for the CU. Signals A and B are the two operands of the target operation. The signal LUT provides all the constant values required by the CU, including the input and output elements of the LUT and a constant zero-valued input. The signal OpCode is the control signal of the CU, where OpCode $= \{\overline{\mathrm{EnM}}, \mathrm{C_{in}}, \mathrm{LoadC0}, \mathrm{LoadC1}, \mathrm{LoadC2}, \mathrm{LoadC3}, \mathrm{LoadC4}\}$. The output signal Res gives the calculation results for the addition, subtraction or min* operation. The output signal Cmp gives the comparison results required by the min* operations, which are stored in the 1-bit registers C_1 to C_4. Finally, the output signal Msb $= \{C_0, \mathrm{MSB(A)}, \mathrm{MSB(B)}\}$ is required by the controller for controling the CU during the min* operation, as explained below. Similar to the process of performing the max* operation in the CU of Chapter 5, the min* operation is decomposed into a sequence of ACS operation and performed by the proposed CU in nine clock cycles, completing one ACS operation per clock cycle, as follows.

1. In the first clock cycle,

$$\text{OpCode} = \begin{cases} 1110000_2 & \text{if MSB(A)} \oplus \text{MSB(B)} = 0, \\ 1010000_2 & \text{if MSB(A)} \oplus \text{MSB(B)} = 1. \end{cases} \tag{7.3}$$

The operation result of this clock cycle is:

$$\text{R1} = \begin{cases} \text{A} - \text{B} & \text{if MSB(A)} \oplus \text{MSB(B)} = 0, \\ \text{A} + \text{B} & \text{if MSB(A)} \oplus \text{MSB(B)} = 1; \end{cases} \tag{7.4}$$

$$\text{C}_0 = \begin{cases} \text{MSB(A} - \text{B)} & \text{if MSB(A)} \oplus \text{MSB(B)} = 0, \\ \text{MSB(A} + \text{B)} & \text{if MSB(A)} \oplus \text{MSB(B)} = 1. \end{cases} \tag{7.5}$$

Based on the results in R1, C_0 and the signs of A and B, the further operations in the CU to perform the $\min^*$ operation can be determined. Assuming $a = |\text{A}|$ and $b = |\text{B}|$, there are four possible combinations of the signs of A and B, as summarised in Table 7.1. The three internal results of $\min^*(\text{A}, \text{B})$, namely $\min(\text{A}, \text{B})$, $|\text{A} + \text{B}|$

A	B	R1	Msb			sign(A)sign(B) min (|A|, |B|)	$\|\text{A}+\text{B}\|$	$\|\text{A}-\text{B}\|$
			MSB(A)	MSB(B)	C_0			
a	b	$\text{A}-\text{B}$	0	0	0	b	a+b	a-b
					1	a	a+b	-(a-b)
a	-b	$\text{A}+\text{B}$	0	1	0	b	a+b	a-b
					1	-a	-(a+b)	a-b
-a	b	$\text{A}+\text{B}$	1	0	0	a	a+b	-(a-b)
					1	-b	-(a+b)	-(a-b)
-a	-b	$\text{A}-\text{B}$	1	1	0	-a	-(a+b)	-(a-b)
					1	-b	-(a+b)	a-b

TABLE 7.1: Summarisation of the four different possible situations of Signal A and B.

and $|\text{A} - \text{B}|$, can be calculated based on the results on the output signal MSB, as shown in the table. In this way, the absolute value operation can be avoided, removing the requirement for any dedicated hardware resource for this purpose.

2. From the second to the sixth clock cycle, the task for the CU is to perform the comparison operations according to Equation 7.2 for the two correction terms of Equation 7.1. Four comparisons are required in total. Similar to the proposed LUT-Log-BCJR decoder, the comparison results are stored as 1-bit binary numbers in the 1-bit registers C_1 to C_4. Based on Equation 7.1, two absolute values $|\text{A} + \text{B}|$ and $|\text{A} - \text{B}|$ are compared with the input elements of the LUT from Equation 7.2. The particular one that is compared first depends on the current value stored in R1, according to Equation 7.4. Once the comparisons for the current value in R1 are completed, the other one is calculated, stored in R1 and compared with the LUT contents.

The method used to compare an absolute value $|x|$ with an LUT input element y is to calculate $y-|x|$. The most significant bit of the result is then stored in a 1-bit register as the comparison result. However, as with the current value in R1, the CU can only calculate $A+B$ or $A-B$ without performing an absolute operation. In order to perform the equivalent calculation of $y-|x|$, the CU calculates $y-x$ or $y+x$, according to the sign of x. The sign of $A+B$ or $A-B$ can be obtained by the value of MSB, as shown in Table 7.1. The controller takes MSB as an input and generates the corresponding OpCode for the CU to perform the comparison. The comparisons required for a 4-output LUT are given in Section 5.4.3. As a result, two clock cycles are required to perform the comparisons for the current value in R1. To complete the calculation of the other correction function, one clock cycle is then used to calculate and store the value of $|\mathrm{A}+\mathrm{B}|$ or $|\mathrm{A}-\mathrm{B}|$, whichever has not been considered yet. Two more clock cycles are required to perform the remaining two comparison. Five clock cycles are required in total to complete the comparison task of a $\min^*$ operation.

3. In the seventh clock cycle, the first component of Equation 7.1 is calculated and stored in R1. The CU performs one of the calculation from $0+\mathrm{A}$, $0-\mathrm{A}$, $0+\mathrm{B}$ and $0-\mathrm{B}$, according to Table 7.1.

4. Finally, in the eighth and ninth clock cycles, two additions are performed to add the correction terms in Equation 7.1 to R1, according to the results stored in C_1 to C_4.

Based on the proposed CU, a highly scalable parallel architecture can be realised, offering an attractive trade-off between the decoding throughput and the energy efficiency. As demonstrated above, the proposed CU is capable of performing the functions of both the variable nodes and the check nodes. Ideally, a fully parallel architecture can be implemented for the LDPC decoder, where each edge in Figure 7.1 employs dedicated CU for calculating the extrinsic LLR. The CUs can operate as either variable nodes or check nodes iteratively during the decoding process. However, for a LDPC code with a long interleaver length, the number of CUs required may be too large, and the hardware complexity of the decoder may become excessively high. Therefore, the trade-off between the scale of parallelism of the architecture and the decoding throughput can be explored in future work.

Glossary

3GPP	3rd Generation Partnership Project.
ACS	Add-Compare-Select.
AMT	Aeronautical Mobile Telemetry.
APP	A Posteriori Probability.
ASIC	Application-Specific Integrated Circuit.
AWGN	Additive White Gaussian Noise.
BAN	Body Area Network.
BCJR	Bahl-Cocke-Jelinek-Raviv.
BER	Bit Error Rate.
BPSK	Binary Phase-Shift Keying.
CU	Calculation Unit.
DRP	Dithered Relative Prime.
DSP	Digital Signal Processor.
DVB	Digital Video Broadcasting.
ECC	Error-Correcting Code.
ECG	ElectroCardioGraphy.
EEG	ElectroEncephaloGraphy.
EMG	ElectroMyoGraphy.
ETSI	European Telecommunications Standards Institute.

EXIT	EXtrinsic Information Transfer.
FCC	Federal Communications Commission.
FER	Frame Error Rate.
FPGA	Field-Programmable Gate Array.
FSM	Finite-State Machine.
GA	Genetic Algorithm.
HBC	Human Body Communications.
HIHO	Hard-In Hard-Out.
IC	Integrated Circuit.
IEEE	Institute of Electrical and Electronics Engineers.
ISM	Industrial Scientific and Medical.
LBT	Listen-Before-Transmit.
LDPC	Low-Density Parity-Check.
LLR	Logarithmic Likelihood Ratio.
Log-BCJR	Logarithmic Bahl-Cocke-Jelinek-Raviv.
LTE	Long Term Evolution.
LUT	Look-Up Table.
LUT-Log-BCJR	Look-Up Table based Logarithmic Bahl-Cocke-Jelinek-Raviv.
MAC	Media Access Control.
Max-Log-BCJR	Maximum Logarithmic Bahl-Cocke-Jelinek-Raviv.
MCTC	Multiple-Component Turbo Code.
MEMS	Micro-Electro-Mechanical Systems.
MI	Mutual Information.

MICS	Medical Implant Communication Service.
MIPS	Million Instructions Per Second.
MUX	Multiplexor Unit.
NB	NarrowBand.
NLOS	Non-Line Of Sight.
O-QPSK	Offset Quadrature Phase-Shift Keying.
PCC	Parallel Concatenated Code.
PCCC	Parallel Concatenated Convolutional Code.
PDA	Personal Digital Assistant.
PHY	PHYsical layer.
PN	Pseudo Noise.
ROM	Read-Only Memory.
RSC	Recursive Systematic Convolutional.
RTL	Register-Transfer Level.
SCC	Serial Concatenated Code.
SCCC	Serial Concatenated Convolutional Code.
SIHO	Soft-In Hard-Out.
SISO	Soft-In Soft-Out.
SNR	Signal-to-Noise Ratio.
SRAM	Static Random-Access Memory.
TCTC	Twin-Component Turbo Code.
TSMC	Taiwan Semiconductor Manufacturing Company.
UMTS	Universal Mobile Telecommunications System.

UWB	Ultra-WideBand.
VA	Viterbi Algorithm.
WLAN	Wireless Local Area Network.
WSN	Wireless Sensor Network.

Bibliography

[1] N. Sadeghi, S. Howard, S. Kasnavi, K. I. V. C. Gaudet, and C. Schlegel, "Analysis of error control code use in ultra-low-power wireless sensor networks," in *Proceedings of International Symposium on Circuits and Systems*, Island of Kos, 2006, pp. 3558–3561.

[2] S. L. Howard, C. Schlegel, and K. Iniewski, "Error Control Coding in Low-Power Wireless Sensor Networks: When is ECC Energy-Efficient?" *EURASIP Journal of Wireless Communications and Networking, Special Issue: CMOS RF Circuits for Wireless Applications*, vol. 2006, Arti, pp. 1–14, 2006.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, 2002.

[4] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore, "Environmental Wireless Sensor Networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1903–1917, Nov. 2010.

[5] K. Stone, B. Hoenes, and T. Camp, "Hardware Platform for Wireless Geophysical Monitoring," in *10th International Conference on Information Processing in Sensor Networks (IPSN)*, Chicago, IL, USA, Apr. 2011, pp. 157–158.

[6] S. Drude, "Requirements and Applications Scenarios for Body Area Networks," in *Mobile and Wireless Communications Summit 16th IST*, Budapest, 2007, pp. 1–5.

[7] S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.

[8] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 52, pp. 292–422, 2008.

[9] T. Arampatzis, J. Lygeros, and S. Manesis, "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks," in *Proceedings of the IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control.* IEEE, 2005, pp. 719–724.

[10] B. O'Flynn, R. Martinez-Catala, S. Harte, C. O'Mathuna, J. Cleary, C. Slater, F. Regan, D. Diamond, and H. Murphy, "SmartCoast: A Wireless Sensor Network for Water Quality Monitoring," in *32nd IEEE Conference on Local Computer Networks.* IEEE, Oct. 2007, pp. 815–816.

[11] Z. Rasin and M. R. Abdullah, "Water Quality Monitoring System Using Zigbee Based Wireless Sensor Network," *International Journal of Engineering & Technology IJET*, vol. 9, no. 10, pp. 24–28, 2009.

[12] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks - IPSN '07.* New York, USA: ACM Press, Apr. 2007, p. 254.

[13] M. Martinelli, L. Ioriatti, F. Viani, M. Benedetti, and A. Massa, "A WSN-based solution for precision farm purposes," in *IEEE International Geoscience and Remote Sensing Symposium.* IEEE, 2009, pp. 469–472.

[14] G. Werner-allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, Berkeley, CA, USA, 2006, pp. 381—-396.

[15] M. V. Ramesh, "Real-Time Wireless Sensor Network for Landslide Detection," in *Third International Conference on Sensor Technologies and Applications.* Athens/Glyfada, Greece: IEEE, June 2009, pp. 405–409.

[16] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02.* New York, USA: ACM Press, Sept. 2002, p. 88.

[17] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys '04*, ser. SenSys '04. New York, New York, USA: ACM Press, 2004, p. 214.

[18] G. Barrenetxea, F. Ingelres, G. Schaefer, and M. Vetterli, "Wireless Sensor Networks for Environmental Monitoring: The SensorScope Experience," in *IEEE International Zurich Seminar on Communications*, Zurich, 2008, pp. 98–101.

[19] V. Potdar, A. Sharif, and E. Chang, "Wireless Sensor Networks: A Survey," *2009 International Conference on Advanced Information Networking and Applications Workshops*, pp. 636–641, May 2009.

[20] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, May 2000.

[21] F. L. Lewis, "Wireless Sensor Networks," in *Smart environments: technologies, protocols, and applications.* Wiley-Interscience, 2005, vol. 43, ch. 2.

[22] A.-S. Porret, T. Melly, C. C. Enz, and E. A. Vittoz, "A low-power low-voltage transceiver architecture suitable for wireless distributed sensors network," in *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353).* Presses Polytech. Univ. Romandes, pp. 56–59.

[23] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, M. Sridharan, S. Kumar, N. Seddon, C. Anderson, T. Herman, N. Trivedi, M. Nesterenko, R. Shah, S. Kulkarni, M. Aramugam, M. Gouda, D. Culler, P. Dutta, C. Sharp, G. Tolle, M. Grimmer, B. Ferriera, and K. Parker, "ExScal: Elements of an Extreme Scale Wireless Sensor Network," in *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.* IEEE, 2005, pp. 102–108.

[24] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking - MobiCom '01.* New York, New York, USA: ACM Press, July 2001, pp. 272–287.

[25] Z.-H. Long and M.-J. Gao, "Survey on network lifetime research for wireless sensor networks," in *2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology*, Oct. 2009, pp. 899–902.

[26] S. Roundy, P. K. Wright, and J. M. Rabaey, *Energy Scavenging for Wireless Sensor Networks.* New York: Springer, 2004.

[27] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in *4th International Symposium on Information Processing in Sensor Networks.* IEEE, 2005, pp. 457–462.

[28] S. Y. Seidel and T. S. Rappaport, "914 MHz path loss prediction models for indoor wireless communications in multifloored buildings," *IEEE Transactions on Antennas and Propagation*, vol. 40, no. 2, pp. 207–217, 1992.

[29] A. Fanimokun and J. Frolik, "Effects of natural propagation environments on wireless sensor network coverage area," in *Proceedings of the 35th Southeastern Symposium on System Theory.* Auburn, AL, U.S.: IEEE, 2003, pp. 16–20.

[30] W. Huan, "Wireless body area networks path loss characterization analysis," in *2nd International Conference on Computer Engineering and Technology.* IEEE, 2010, pp. 163–164.

[31] E. Reusens, W. Joseph, G. Vermeeren, L. Martens, B. Latre, I. Moerman, B. Braem, and C. Blondia, "Path loss models for wireless communication channel along arm and torso: measurements and simulations," in *IEEE Antennas and Propagation International Symposium.* IEEE, June 2007, pp. 345–348.

[32] A. Fort, J. Ryckaert, C. Desset, P. De Doncker, P. Wambacq, and L. Van Biesen, "Ultra-wideband channel model for communication around the human body," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 927–933, Apr. 2006.

[33] K. Sayrafian-Pour, W.-B. Yang, J. Hagedorn, J. Terrill, and K. Y. Yazdandoost, "A statistical path loss model for medical implant communication channels," in *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications.* Tokyo, Japan: IEEE, Sept. 2009, pp. 2995–2999.

[34] V. Z. Groza, D. Makrakis, D. C. Petriu, N. D. Georganas, and E. M. Petriu, "Sensor-based information appliances," *IEEE Instrumentation & Measurement Magazine*, vol. 3, no. 4, pp. 31–35, 2000.

[35] M. Dermibas, "Wireless sensor networks for monitoring of large public buildings," *University at Buffalo, Tech. Rep*, 2005.

[36] T. G. Zimmerman, "Personal Area Networks: Nearfield intrabody communication," *IBM System Journal*, vol. 35, pp. 609–617, 1996.

[37] B. Zhen, H. Li, and R. Kohno, "IEEE Body Area Netwokrs for Medical Applications," in *Wireless Communication Systems, 2007. ISWCS 2007. 4th International Symposium on*, 2007.

[38] M. A. Hanson, H. C. Powell, A. T. Barth, K. Ringgenberg, B. H. Calhoun, J. H. Aylor, and J. Lach, "Body Area Sensor Networks: Challenges and Opportunities," *Computer*, vol. 42, no. 1, pp. 58–65, Jan. 2009.

[39] H. Li, K. Takizawa, B. Zhen, and R. Kohno, "Body Area Network and Its Standardization at IEEE 802.15.MBAN," in *Mobile and Wireless Communications Summit, 2007. 16th IST*, 2007, pp. 1–5.

[40] J. A. D. Moutinho, "Wireless Body Area Network," 2011.

[41] J. Ryckaert, P. De Doncker, R. Meys, A. De Le Hoye, and S. Donnay, "Channel model for wireless communication around human body," *Electronics Letters*, vol. 40, no. 9, p. 543, 2004.

[42] N. F. Timmons and W. G. Scanlon, "Analysis of the performance of IEEE 802.15.4 for medical sensor body area networking," in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks.* IEEE, 2004, pp. 16–24.

[43] J. Rousselot, A. El-Hoiydi, and J.-D. Decotignie, "Performance Evaluation of the IEEE 802.15.4a UWB Physical Layer for Body Area Networks," in *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, 2007.

[44] D. Domenicali and M.-G. D. Benedetto, "Performance Analysis for a Body Area Network Composed of IEEE 802.15.4a Devices," in *Proceedings of 4th Workshop on Positioning, Navigation and Communication 2007(WPNC'07), Hannover, Germany*, 2007, pp. 273–276.

[45] Q. Zhang, P. Feng, Z. Geng, X. Yan, and N. Wu, "A 2.4-GHz Energy-Efficient Transmitter for Wireless Medical Applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 5, no. 1, pp. 39–47, Feb. 2011.

[46] J. Grosinger and M. Fischer, "Indoor on-body channel measurements at 900MHz," in *2011 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications*. IEEE, Sept. 2011, pp. 1037–1040.

[47] J. Masuch and M. Delgado-Restituto, "A 350 W 2.3 GHz integer-N frequency synthesizer for body area network applications," in *2011 IEEE 11th Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*. IEEE, Jan. 2011, pp. 105–108.

[48] D. Kurup, W. Joseph, E. Tanghe, G. Vermeeren, and L. Martens, "Extraction of antenna gain from path loss model for in-body communication," *Electronics Letters*, vol. 47, no. 23, p. 1262, 2011.

[49] S. L. Cotton, A. McKernan, A. J. Ali, and W. G. Scanlon, "An experimental study on the impact of human body shadowing in off-body communications channels at 2.45 GHz," in *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, 2011, pp. 3133–3137.

[50] V. De Santis and M. Feliziani, "Intra-body channel characterization of medical implant devices," in *EMC Europe 2011 York*, 2011, pp. 816–819.

[51] D. Kurup, W. Joseph, G. Vermeeren, and L. Martens, "In-body Path Loss Model for Homogeneous Human Tissues," *IEEE Transactions on Electromagnetic Compatibility*, no. 99, pp. 1–9, 2011.

[52] "Revision of part 15 of the commission's rules regarding ultra-wideband transmission systems: First report and order," Federal Communications Commission, Washington DC, Tech. Rep., 2002.

[53] J. Ryckaert, C. Desset, V. De Heyn, M. Badaroglu, P. Wambacq, G. V. der Plas, and B. V. Poucke, "Ultra-WideBand Transmitter for Wireless Body Area Networks," in *Proceeding on 14th IST Mobile & Wireless Communications Summit*, 2005.

[54] T. S. P. See, J. Y. Hee, C. T. Ong, L. C. Ong, and Z. N. Chen, “Inter-body channel model for UWB communications,” in *3rd European Conference on Antennas and Propagation*, 2009, pp. 3519–3522.

[55] Q. Wang, T. Tayamachi, I. Kimura, and J.-Q. Wang, “An On-Body Channel Model for UWB Body Area Communications for Various Postures,” *IEEE Transactions on Antennas and Propagation*, vol. 57, no. 4, pp. 991–998, Apr. 2009.

[56] L. Betancur, N. Cardona, A. Navarro, and L. Traver, “A statistical channel model for on body Area networks in Ultra Wide Band Communications,” in *2009 IEEE Latin-American Conference on Communications.* IEEE, Sept. 2009, pp. 1–6.

[57] K. Y. Yazdandoost and K. Hamaguchi, “Very small UWB antenna for WBAN applications,” in *2011 5th International Symposium on Medical Information and Communication Technology.* IEEE, Mar. 2011, pp. 70–73.

[58] M. L. R. Fox, H. Symons, S. Berson, and H. Westphal, “FCC proposes rules for body area networks (MBAN),” 2009.

[59] K. S. Kwak, S. Ullah, and N. Ullah, “An overview of IEEE 802.15.6 standard,” in *2010 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL 2010).* IEEE, Nov. 2010, pp. 1–6.

[60] M. Patel and J. Wang, “Applications, challenges, and prospective in emerging body area networking technologies,” *Wireless Communications, IEEE*, vol. 17, no. 1, pp. 80–88, 2010.

[61] Y.-Q. Zhang, Y. Shakhsheer, A. T. Barth, H. C. P. Jr., S. A. Ridenour, M. A. Hanson, J. Lach, and B. H. Calhoun, “Energy Efficient Design for Body Sensor Nodes,” *Journal of Low Power Electronics and Applications*, vol. 1, no. 1, pp. 109–130, 2011.

[62] S. Ullah and K. S. Kwak, “Throughput and delay limits of IEEE 802.15.6,” in *2011 IEEE Wireless Communications and Networking Conference.* IEEE, Mar. 2011, pp. 174–178.

[63] V. M. Jones, R. G. A. Bults, D. Konstantas, and P. A. M. Vierhout, “Healthcare PANs: Personal Area Networks for trauma care and home care,” in *In 4th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2001, pp. 1369–1374.

[64] M. Soini, J. Nummela, P. Oksa, L. Ukkonen, and L. Sydänheimo, “Wireless Body Area Network for Hip Rehabilitation System,” *Ubiquitous Computing and Communication Journal*, vol. 3, p. 7, 2008.

[65] L. Huang, M. Ashouei, R. F. Yazicioglu, J. Penders, R. J. M. Vullers, G. Dolmans, P. Merken, J. Huisken, H. De Groot, C. V. Hoof, and B. Gyselinckx, “Ultra-Low

Power Sensor Design for Wireless Body Area Networks - Challenges, Potential Solutions, and Applications," *Journal of Digital Content Technology and its Applications*, vol. 3, no. 3, pp. 136–148, 2009.

[66] B. Latré, I. Moerman, B. Dhoedt, and P. Demeester, "Networking in wireless body area networks," in *in 5th FTW PHD Symposium, Interactive poster session*, 2004, p. 113.

[67] C. K. Singh and A. Kumar, "Performance evaluation of an IEEE 802.15.4 sensor network with a star topology," *Wireless Networks*, vol. 14, no. 4, pp. 543–568, 2008.

[68] S. Choi, S. Song, K. Sohn, H. Kim, J. Kim, J. Yoo, and H. Yoo, "A Low-power Star-topology Body Area Network Controller for Periodic Data Monitoring Around and Inside the Human Body," in *10th IEEE International Symposium on Wearable Computers*, 2006, pp. 139–140.

[69] R. G. Maunder, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, and L. Hanzo, "Iterative Decoding for Redistributing Energy Consumption in Wireless Sensor Networks," in *Proceedings of the IEEE Int Conf on Computer Communications and Networks.* IEEE, 2008, pp. 623–628.

[70] A. G. Ruzzelli, R. Jurdak, G. M. P. O'Hare, and P. V. D. Stok, "Energy-efficient multi-hop medical sensor networking," in *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*, 2007, pp. 37–42.

[71] B. Latre, B. Braem, I. Moerman, C. Blondia, E. Reusens, W. Joseph, and P. Demeester, "A Low-delay Protocol for Multihop Wireless Body Area Networks," in *4h Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, 2007, pp. 1–8.

[72] J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.* IEEE, 2002, pp. 1567–1576.

[73] T. V. Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the First international conference on Embedded networked sensor systems - SenSys.* New York, New York, USA: ACM Press, Nov. 2003, p. 171.

[74] E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3. IEEE, 2003, pp. 1713–1723.

[75] D.-H. Nam and H.-K. Min, "An Energy-Efficient Clustering Using a Round-Robin Method in a Wireless Sensor Network," in *5th ACIS International Conference on Software Engineering Research, Management & Applications.* IEEE, Aug. 2007, pp. 54–60.

[76] L. Kyounghwa, L. Joohyun, L. Hyeopgeon, and S. Yongtae, "A Density and Distance based Cluster Head Selection algorithm in Sensor Networks," in *The 12th International Conference on Advanced Communication Technology (ICACT)*, 2010, pp. 162–165.

[77] M. C. M. Thein and T. Thein, "An Energy Efficient Cluster-Head Selection for Wireless Sensor Networks," in *International Conference on Intelligent Systems, Modelling and Simulation.* IEEE, Jan. 2010, pp. 287–291.

[78] M. Cardei and M. Thai, "Energy-efficient target coverage in wireless sensor networks," in *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2005, pp. 1976–1984.

[79] A. Wang and A. Chandrakasan, "Energy-efficient DSPs for wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 19, no. 4, pp. 68–78, July 2002.

[80] X.-H. Li, "Energy efficient wireless sensor networks with transmission diversity," *Electronics Letters*, vol. 39, no. 24, p. 1753, 2003.

[81] O. C. Omeni, O. Eljamaly, and A. J. Burdett, "Energy Efficient Medium Access Protocol for Wireless Medical Body Area Sensor Networks," in *4th IEEE/EMBS International Summer School and Symposium on Medical Devices and Biosensors.* IEEE, Aug. 2007, pp. 29–32.

[82] C. C. Enz, A. El-Hoiydi, J.-D. Decotignie, and V. Peiris, "WiseNET: an ultralow-power wireless sensor network solution," *Computer*, vol. 37, no. 8, pp. 62–70, Aug. 2004.

[83] C. Schurgers and M. B. Srivastava, "Energy efficient routing in wireless sensor networks," in *Proceedings Communications for Network-Centric Operations: Creating the Information Force*, vol. 1. IEEE, 2001, pp. 357–361.

[84] H. Oh and K. Chae, "An Energy-Efficient Sensor Routing with low latency, scalability in Wireless Sensor Networks," in *International Conference on Multimedia and Ubiquitous Engineering.* IEEE, 2007, pp. 147–152.

[85] M. Zhang, S.-P. Wang, C. Liu, and H.-B. Feng, "An Novel Energy-Efficient Minimum Routing Algorithm (EEMR) in Wireless Sensor Networks," in *4th International Conference on Wireless Communications, Networking and Mobile Computing.* IEEE, Oct. 2008, pp. 1–4.

[86] W. N. W. Muhamad, N. F. Naim, N. Hussin, N. Wahab, N. A. Aziz, S. S. Sarnin, and R. Mohamad, “Maximizing Network Lifetime with Energy Efficient Routing Protocol for Wireless Sensor Networks,” in *Fifth International Conference on MEMS NANO, and Smart Systems*. IEEE, 2009, pp. 225–228.

[87] X. Chen, R. Blum, and B. M. Sadler, “A new scheme for energy-efficient estimation in a sensor network,” in *43rd Annual Conference on Information Sciences and Systems*. IEEE, Mar. 2009, pp. 799–804.

[88] M. C. Vuran and I. F. Akyildiz, “Error Control in Wireless Sensor Networks: A Cross Layer Analysis,” *IEEE/ACM Transactions on Networking*, vol. 17, no. 4, pp. 1186–1199, Aug. 2009.

[89] M. E. Pellenz, R. D. Souza, and M. S. P. Fonseca, “Error control coding in wireless sensor networks,” *Telecommunication Systems*, vol. 44, no. 1-2, pp. 61–68, 2009.

[90] A. Brokalakis and I. Papaefstathiou, “Using hardware-based forward error correction to reduce the overall energy consumption of WSNs,” in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, Apr. 2012, pp. 2191–2196.

[91] J. Abouei, J. D. Brown, K. N. Plataniotis, and S. Pasupathy, “On the Energy Efficiency of LT Codes in Proactive Wireless Sensor Networks,” *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 1116–1127, Mar. 2011.

[92] A. Brokalakis, G.-G. Mplemenos, K. Papadopoulos, and I. Papaefstathiou, “RESENSE: An Innovative, Reconfigurable, Powerful and Energy Efficient WSN Node,” in *2011 IEEE International Conference on Communications (ICC)*. IEEE, June 2011, pp. 1–5.

[93] J. Singh and D. Pesch, “Towards Energy Efficient Adaptive Error Control in Indoor WSN: A Fuzzy Logic Based Approach,” in *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*. IEEE, Oct. 2011, pp. 63–68.

[94] J. S. Rahhal, “LDPC coding for MIMO wireless sensor networks with clustering,” in *2012 Second International Conference on Digital Information and Communication Technology and it’s Applications (DICTAP)*. IEEE, May 2012, pp. 58–61.

[95] J. Misic, “Enforcing Patient Privacy in Healthcare WSNs Using ECC Implemented on 802.15.4 Beacon Enabled Clusters,” in *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, 2008.

[96] M. R. Yuce, “Implementation of Body Area Networks Based on MICS/WMTS Medical Bands for Healthcare Systems,” in *IEEE Engineering in Medicine and Biology Society Conference (IEEE EMBC08)*, Aug. 2008, pp. 3417–3421.

[97] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes," in *Proceedings of the IEEE International Conference on Communications*, vol. 2, Geneva, Switzerland, 1993, pp. 1064–1070.

[98] R. Gallager, "Low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[99] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella, "Performance study of IEEE 802.15.4 using measurements and simulations," in *Proceeding of the IEEE Wireless Communications and Networking Conference*. IEEE, 2006, pp. 487–492.

[100] S. Ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[101] S. Benedetto and G. Montorsi, "Serial concatenated of block and convolutional codes," *Electronics Letters*, vol. 32, no. 10, pp. 887–888, 1996.

[102] "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," 2006.

[103] "3GPP LTE Turbo Reference Design," Altera Corporation, Tech. Rep., 2011.

[104] "Digital Video Broadcasting (DVB); Framing Structure, channel coding and modulation for Satellite Services to Handheld devices (SH) below 3 GHz," 2010.

[105] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. 20, no. 3, pp. 284–287, 1974.

[106] H. Chen, R. G. Maunder, and L. Hanzo, "An Exit-Chart Aided Design Procedure for Near-Capacity N-Component Parallel Concatenated Codes," in *Proceedings of the IEEE Global Telecommunications Conference GLOBECOM*. Miami, Florida, US: IEEE, Dec. 2010, pp. 1–5.

[107] G. D. J. Forney, "Concatenated Codes," Massachusetts Institute of Technology Research Lab of Electronics, Tech. Rep., 1966.

[108] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *SIAM Journal of Applied Math*, vol. 8, pp. 300–304, 1960.

[109] P. Elias, "Coding for noisy channels," in *IRE Convention Record Pt. 4*, 1955, p. 37.

[110] J. H. Yuen, M. K. Simon, W. Miller, F. Pollara, C. R. Ryan, D. Divsalar, and J. C. Morakis, "Modulation and coding for satellite and space communications," in *Proceedings of the IEEE*, vol. 78, no. 7, 1990, pp. 1250–1265.

[111] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 493–497, 1967.

[112] E. Boutillon, C. Douillard, and G. Montorsi, "Iterative Decoding of Concatenated Convolutional Codes: Implementation Issues," in *Proceedings of the IEEE*, vol. 95, no. 6, 2007, pp. 1201–1227.

[113] B. Sklar, *Fundamentals of Turbo Codes, Digital Communications: Fundamentals and Applications, Second Edition.* Prentice-Hall, 2001.

[114] C. Berrou and A. Glavieux, "Near Optimum Error Correcting Coding and Decoding: Turbo-Codes," *IEEE Trans. on Communications*, vol. 44, no. 10, pp. 1261–1271, 1996.

[115] C. Schlegel and L. Perez, *Trellis and Turbo Coding*, ser. IEEE Press Series on Digital & Mobile Communication, J. B. Anderson, Ed. John Wiley & Sons, 2004.

[116] *Universal Mobile Telecommunications System (UMTS); Multiplexing and Channel Coding (FDD)*, European Telecommunications Standards Institute Std., 1999.

[117] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Transactions on Information Theory*, vol. 47, no. 10, pp. 366–386, 2001.

[118] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and Sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings of IEEE International Conference of Communication*, vol. 2, Seattle, WA, USA, 1995, pp. 1009–1013.

[119] J. Sayir, "Measuring EXIT Charts for Low Complexity Decoders," in *International Symposium on Communication Theory and Applications*, Ambleside, UK, 2009.

[120] B. Krishnamachari and C. S. Raghavendra, "Performance evaluation of the IEEE 802.15.4 MAC for low-rate low-power wireless networks," in *IEEE International Conference on Performance, Computing, and Communications.* IEEE, 2004, pp. 701–706.

[121] J. Hoffert, K. Klues, and O. Orjih, "Configuring the IEEE 802.15.4 MAC Layer for Single-sinkWireless Sensor Network Applications," Washington University, St. Louis, Missouri, Tech. Rep., 2005.

[122] "A True System-on-Chip Solution for 2.4 GHz IEEE 802.15.4 / Zig- Bee(TM) Datasheet," Chipcon, Tech. Rep., 2007.

[123] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA, USA: Addison-Wesley, Oct. 1989.

[124] S. Crozier and P. Guinand, “High-performance low-memory interleaver banks for turbo-codes,” in *Proceeding of IEEE 54th Vehicular Technology Conference. VTC Fall*, vol. 4. IEEE, 2001, pp. 2394–2398.

[125] D. Divsalar, S. Dolinar, and F. Pollara, “Serial Concatenated Trellis Coded Modulation with Rate-1 Inner Code,” in *In Proceeding of GLOBECOM*, San Francisco, 2000, pp. 777–782.

[126] S. Benedetto and G. Montorsi, “Iterative decoding of serially concatenated convolutional codes,” *Electronics Letters*, vol. 32, no. 13, pp. 1186–1188, 1996.

[127] T. Okuma, Y. Cao, M. Muroyama, and H. Yasuura, “Reducing access energy of on-chip data memory considering active data bitwidth,” in *Proceedings of the International Symposium on Low Power Electronics and Design.* IEEE, 2002, pp. 88–91.

[128] Z.-F. Wang and Q.-W. Li, “Very Low-Complexity Hardware Interleaver for Turbo Decoding,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 7, pp. 636–640, July 2007.

[129] J.-H. Kim and I.-C. Park, “Double-Binary Circular Turbo Decoding Based on Border Metric Encoding,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 1, pp. 79–83, Jan. 2008.

[130] M. Martina, M. Nicola, and G. Masera, “A Flexible UMTS-WiMax Turbo Decoder Architecture,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 4, pp. 369–373, Apr. 2008.

[131] R. Dobkin, M. Peleg, and R. Ginosar, “Parallel interleaver design and VLSI architecture for low-latency MAP turbo decoders,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 4, pp. 427–438, Apr. 2005.

[132] M. Kakitani, G. Brante, R. D. Souza, and A. Munaretto, “Comparing the energy efficiency of single-hop, multi-hop and incremental decode-and-forward in multi-relay wireless sensor networks,” in *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications.* IEEE, Sept. 2011, pp. 970–974.

[133] H. Michel and N. Wehn, “Turbo-Decoder Quantization for UMTS,” *IEEE Communication Letters*, vol. 5, no. 2, pp. 55–57, 2001.

[134] M. A. Castellon, I. J. Fair, and D. G. Elliott, “Fixed-point turbo decoder implementation suitable for embedded applications,” in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, Saskatoon, Canada, 2005, pp. 1065–1068.

[135] J. Hsu and C. Wang, "On finite-precision implementation of a decoder for turbo codes," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 4, Orlando, FL, USA, 1999, pp. 423–426.

[136] A. Worm, H. Michel, F. Gilbert, G. Kreiselmaier, M. Thul, and N. Wehn, "Advanced Implementation Issues of Turbo-Decoders," in *Proceedings of the International Symposium on Turbo-Codes and Related Topics*, Brest, France, Sept. 2000, pp. 351–354.

[137] T. K. Blankenship and B. Classon, "Fixed-point performance of low-complexity turbo decoding algorithms," in *Proceddings of the IEEE Vehicular Technology Conference*, vol. 2, Rhodes, Greece, 2001, pp. 1483–1487.

[138] G. Montorsi and S. Benedetto, "Design of Fixed-Point Iterative Decoders for Concatenated Codes with Interleavers," in *IEEE Journal on Selected Areas in Communications*, vol. 2, San Francisco, CA, USA, 2000, pp. 801–806.

[139] Y. Wu, B. D. Woerner, and T. K. Blankenship, "Data width requirements in SISO decoding with modulo normalization," *IEEE Transactions on Communications*, vol. 49, no. 11, pp. 1861–1868, 2001.

[140] R. Hoshyar, A. R. S. Bahai, and R. Tafazolli, "Finite precision turbo decoding," in *Proceedings of the International Symposiumon on Turbo Codes and Related Topics*, Brest, France, 2003, pp. 483–486.

[141] A. Morales-Cortes, R. Parra-Michel, L. F. Gonzalez-Perez, and T. G. Cervantes, "Finite Precision Analysis of the 3GPP Standard Turbo Decoder for Fixed-Point Implementation in FPGA Devices," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs*, Cancun, Mexico, 2008, pp. 43–48.

[142] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A Soft-Input Soft-Output APP Module for Iterative Decoding of Concatenated Codes," *IEEE Communications Letters*, vol. 1, no. 1, pp. 22–24, 1997.

[143] V. Singh, "Elimination of overflow oscillations in fixed-point state-specе digital filters using saturation alrithmetic," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 6, pp. 814–818, 1990.

[144] D. A. Balley and A. A. Beer, "Simulation of filter structures for fixed-point implementation," in *Proceeding of the 28th Southeastern Symposium on System Theory*, Baton Rouge, LA, USA, 1996, pp. 270–274.

[145] G. Masera, *Turbo Code Applications: a journey from a paper to realization*, K. Sripimanwat, Ed. Springer Netherlands, 2005.

[146] A. Hekstra, "An Alternative to Metric Rescaling in Viterbi Decoders," *IEEE Transcations on Communications*, vol. 37, no. 11, pp. 1220–1222, 1989.

[147] B. Riaz and J. Bajcsy, “Impact of Finite Precision Arithmetics on EXIT Chart Analysis of Turbo Codes,” in *Proceedings of the IEEE Consumer Communications and Networking Conference*, Las Vegas, NV, USA, 2008.

[148] S. Dolinar, D. Divsalar, and F. Pollara, “Turbo code performance as a function of code block size,” in *Information Theory 1998 Proceedings 1998 IEEE International Symposium on*, 1998, pp. 32–.

[149] P. Robertson, P. Hoeher, and E. Villebrun, “Optimal and Sub-Optimal Maximum A Posteriori Algorithms Suitable for Turbo Decoding,” *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119–125, 1997.

[150] M. C. Valenti and J. Sun, “The UMTS turbo Code and an Efficient Decoder Implementation Suitable for Software-Defined Radios,” *International Journal of Wireless Information Networks*, vol. 8, no. 4, pp. 203–215, 2001.

[151] G. Masera, G. Piccinini, M. R. Roch, and M. Zamboni, “VLSI Architectures for Turbo Codes,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 3, pp. 369–379, 1999.

[152] E. Yeo, P. Pakzad, B. Nikolic, and V. Anantharam, “VLSI Architectures for Iterative Decoders in Magnetic Recording Channels,” *IEEE Transactions on Magnetics*, vol. 37, no. 2, pp. 748–754, 2001.

[153] T. Miyauchi, K. Yamamoto, T. Yokokawa, M. Kan, Y. Mizutani, and M. Hattori, “High-performance programmable SISO decoder VLSI implementation for decoding turbo codes,” in *Global Telecommunications Conference*, vol. 1, San Antonio, TX , USA, 2001, pp. 305–309.

[154] M. A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L. M. Davis, G. Woodward, C. Nicol, and R.-H. Yan, “A unified turbo/Viterbi channel decoder for 3GPP mobile wireless in 0.18-μm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1555–1564, 2002.

[155] G. Masera, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, “Architectural Strategies for Low-Power VLSI Turbo Decoders,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 3, pp. 279–285, 2002.

[156] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol, “A 24Mb/s radix-4 Log-MAP turbo decoder for 3GPP-HSDPA mobile wireless,” in *IEEE International Solid-State Circuits Conference*, vol. 1, 2003, pp. 150–484.

[157] Y. Zhang and K. K. Parhi, “High-throughput radix-4 logMAP turbo decoder architecture,” in *Asilomar conference on Signals, System and Computers*, Pacific Grove, CA, USA, 2006, pp. 1711–1715.

[158] F.-M. Li, C.-H. Lin, and A.-Y. Wu, "Unified Convolutional/Turbo Decoder Design Using Tile-Based Timing Analysis of VA/MAP Kernel," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 10, pp. 1063–8210, 2008.

[159] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE Turbo Code Decoder," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2010, pp. 1420–1425.

[160] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE," *IEEE Jouranal of Solid-State Circuits*, vol. 46, pp. 8–17, 2011.

[161] I. Ahmed and T. Arslan, "VLSI Design of Multi Standard Turbo Decoder for 3G and Beyond," in *Asia and South Pacific Design Automation Conference*, Yokohama, Japan, 2007, pp. 589–594.

[162] C.-H. Lin, C.-Y. Chen, and A.-Y. Wu, "High-throughput 12-mode CTC decoder for WiMAX standard," in *IEEE International Symposium on VLSI Design, Automation and Test*, Hsinchu, Taiwan, 2008, pp. 216–219.

[163] C. Wong, Y. Lee, and H. Chang, "A 188-size 2.1mmˆ2 Reconfigurable Turbo Decoder Chip with Parallel Architecture for 3GPP LTE System," in *2009 Symposium on VLSI Circuits*, Kyoto, Japan, 2009, pp. 288–289.

[164] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Soft-Output Decoding Algorithms in Iterative Decoding of Turbo Codes," TDA Progress Report 42-124, Tech. Rep., 1996.

[165] W. J. Gross and P. G. Gulak, "Simplified MAP algorithm suitable for implementation of turbo decoders," *Electronics Letters*, vol. 34, no. 16, p. 1577, 1998.

[166] J. Vogt and A. Finger, "Improving the Max-Log-MAP turbo decoder," *Electronics Letters*, vol. 36, no. 23, p. 1937, 2000.

[167] L. Hanzo, T. H. Liew, B. L. Yeap, R. Tee, and S. X. Ng, *Turbo Coding, Turbo Equalisation and Space-Time Coding.* John Wiley & Sons Inc, 2011.

[168] L. Hanzo, J. P. Woodard, and P. Robertson, "Turbo decoding and detection for wireless applications," in *Proceedings of the IEEE*, vol. 95, no. 6, 2007, pp. 1178–1200.

[169] C. Schurgers, F. Catthoor, and M. Engels, "Memory Optimization of MAP Turbo Decoder Algorithms," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 2, pp. 305–312, 2001.

[170] J. A. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures," in *IEEE Global Telecommunications Conference*, San Diego, CA, 1990, p. 704.

[171] A. J. Viterbi, "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 162–264, 1998.

[172] R. Dobkin, M. Peleg, and R. Ginosar, "Parallel VLSI architecture for MAP turbo decoder," in *The 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, 2002, pp. 384–388.

[173] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proceedings of the 32nd ACM/IEEE conference on Design automation conference - DAC '95.* New York, New York, USA: ACM Press, Jan. 1995, pp. 242–247.

[174] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proceedings 29th ACM/IEEE Design Automation Conference.* IEEE Comput. Soc. Press, 1992, pp. 253–259.

[175] T. Karnik, S. Borkar, and V. De, "Sub-90nm technologies: challenges and opportunities for CAD," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design - ICCAD '02.* New York, New York, USA: ACM Press, Nov. 2002, pp. 203–206.

[176] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and Optimization of an HSDPA Turbo Decoder ASIC," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 98–106, 2009.

[177] J.-H. Kim and I.-C. Park, "A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE," in *IEEE Custom Integrated Circuits Conference*, San Jose, CA, 2009, pp. 487–490.

[178] S.-G. Lee, C.-H. Wang, and W.-H. Sheen, "Architecture Design of QPP Interleaver for Parallel Turbo Decoding," in *IEEE Vehicular Technology Conference*, Taipei, Taiwan, 2010, pp. 1–5.

[179] Y. Sun and J. R. Cavallaro, "Efficient Hardware Implementation of A Highly-Parallel 3GPP LTE, LTE-Advance Turbo Decoder," *Integration, the VLSI Journal*, vol. 44, no. 1, pp. 1–11, 2010.

[180] K. Nakano, M. Sengoku, K. Mase, and S. Shinoda, "Network structure of multi-hop radio networks," in *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies and Industrial Opportunities (Cat. No.00CH37141)*, vol. 2. IEEE, pp. 1159–1164.

[181] K. M. Buyuksahin and F. N. Najm, "Early power estimation for VLSI circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1076–1088, July 2005.

[182] F. N. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 446–455, Dec. 1994.

[183] K. D. Muller-Glaser, K. Kirsch, and K. Neusinger, "Estimating essential design characteristics to support project planning for ASIC design management," in *IEEE International Conference on Computer-Aided Design Digest of Technical Papers.* IEEE Comput. Soc. Press, 1991, pp. 148–151.

[184] M. Nemani and F. N. Najm, "High-level area and power estimation for VLSI circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 6, pp. 697–713, June 1999.

[185] A. Raghunathan, S. Dey, and N. K. Jha, "Register-transfer level estimation techniques for switching activity and power consumption," in *Proceedings of International Conference on Computer Aided Design.* IEEE Comput. Soc. Press, 1996, pp. 158–165.

[186] A. Raghunathan, S. Dey, and N. Jha, "High-level macro-modeling and estimation techniques for switching activity and power consumption," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 4, pp. 538–557, Aug. 2003.

[187] M. Pedram, "Power simulation and estimation in VLSI circuits," in *The VLSI handbook*, W.-K. Chen, Ed. Boca Raton, FL, USA: The CRC Press and the IEEE Press, 1999.

[188] C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, June 1994.

[189] "TSMC 90nm Core Library Databook," 2005.

[190] P. Surti and L.-F. Chao, "Controller power estimation using information from behavioral description," in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, vol. 4. IEEE, pp. 679–682.

[191] D.-F. Zhao, Y.-P. Wu, and N.-N. Tong, "The Applied Research of Convolutional Turbo Code Based on WiMAX Protocol," in *4th International Conference on Wireless Communications, Networking and Mobile Computing.* IEEE, Oct. 2008, pp. 1–3.

[192] Q. Li and N. S. Ramesh, "Channel coding performance in cdma2000 systems," in *IEEE Emerging Technologies Symposium on Broadband, Wireless Internet Access. Digest of Papers (Cat. No.00EX414).* IEEE, 2000, p. 5.

[193] X.-M. Yu, Y.-M. Kang, and D.-F. Yuan, "Performance analysis of turbo codes in wireless rician fading channel with low rician factor," in *IEEE 12th International Conference on Communication Technology.* IEEE, Nov. 2010, pp. 48–51.

[194] D. Divsalar and F. Pollara, "Turbo Codes for Deep-Space Communications," Tech. Rep., 1995.

[195] "TSMC 90nm Low Power High Density Synchronous Single Port with Redundancy SRAM Compiler Databook," 2007.

[196] X. Zuo, R. G. Maunder, and L. Hanzo, "Design of Fixed-Point Processing Based LDPC Codes Using EXIT Charts," in *Proceeding of IEEE Vehicular Technology Conference*, San Francisco, CA, USA, 2011.

Author Index

Index