UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

Serialization and Asynchronous Techniques for Reliable Network-on-Chip Communication

Simon Ogg

Thesis for the degree of Doctor of Philosophy

May 2009

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

SERIALIZATION AND ASYNCHRONOUS TECHNIQUES FOR RELIABLE NETWORK-ON-CHIP COMMUNICATION

by Simon Ogg

The Network-on-Chip (NoC) paradigm has been proposed as a potentially viable on-chip communication infrastructure for multiprocessor SoC. This thesis investigates the development and validation of efficient links that improve NoC performance, power consumption and reliability. There is emphasis on low-level simulation and validation of the NoC links throughout and gate level circuits are given to provide practical implementations.

The first part of the thesis investigates the use of compression in bit-serial point-to-point links as a means of increasing the available bandwidth of the links in NoC. A bit-serial link reduces the cost of interconnect by reducing the number of wires, but at the expense of reduced throughput. Compression is used to improve the throughput of the serial link by reducing the amount of data transmitted through unused significant bit removal. The compression is performed in real time and the overhead of the extra circuitry is small. The link is modelled in VHDL and simulated to check functionality and correct operation.

The second part of the thesis investigates the development of serial asynchronous links to overcome issues such as power and interconnect area overhead in NoC links. Serialization is used to reduce the interconnect cost of a link by reducing the number of wires needed. The combination of asynchronous circuitry and serialization allows for a lower wiring area and reduced power NoC link, in particular for increased link length. The serial asynchronous link is compared to a fully synchronous link of similar characteristics. Power, area and throughput is compared between the asynchronous and synchronous solutions. Validation is performed on FPGA to confirm the correct functionality of the serialized asynchronous link.

Unreliability due to soft errors is becoming an issue with scaling of technology. The third part of the thesis investigates a novel data coding technique for the asynchronous links developed earlier which offers resilience to soft errors. Resilience is achieved by coding the data using symbols for each bit and a common reference so that transient errors on the NoC link wires can be detected by comparing the symbols and reference to obtain validity of the data and the value of the data. Practical circuits are shown and simulated as well as the area and power estimates.

Contents

List of Figures

List of Tables

DECLARATION OF AUTHORSHIP

I, **Simon Ogg** declare that the thesis entitled

Serialization and Asynchronous Techniques for Reliable Network-on-Chip Communication

and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- none of this work has been published before submission, **or** [delete as appropriate] parts of this work have been published as: [please list references]

Signed: ……..

Date:…….

Acknowledgements

I would like to thank my supervisor Professor Bashir Al-Hashimi for his support and help throughout my PhD. I am also grateful to Professor Alex Yakovlev (Newcastle University) for the valuable technical input especially on asynchronous matters.

I would like to acknowledge Physical Sciences Research Council (EPSRC) for funding the project under grant no. EP/C512804.

I would also like to acknowledge Dr Tom Kazmierski and Dr. Koushik Maharatna for the feedback during the 9 month report and MPhil transfer report respectively.

List of Abbreviations

SoC	System-on-Chip
NoC	Network-on-Chip
MSB	Most Significant Bit
LSB	Least Significant Bit
LEDR	Level Encoded Dual Rail
LETS	Level Encoded Transition Signalling
FPGA	Field Programmable Gate Array
PE	Processing Element
DFT	Design For Test
DFM	Design For Manufacture
IP	Intellectual Property
IC	Integrated Circuit
AMBA	Advanced Microcontroller Bus Architecture
LCD	Liquid Crystal Display
ASB	Advanced System Bus
APB	Advanced Peripheral Bus
AHB	Advanced High Performance Bus
AXI	Advanced eXtensible Interface
UART	Universal Asynchronous Receiver/Transmitter
ADC	Analog to Digital Convertor
DAC	Digital to Analog Convertor
RAM	Random Access Memory
ROM	Read Only Memory
DSP	Digital Signal Processor

SERDES	Serializer / Deserializer
WAFT	WAve Front Train
RTL	Register Transfer Level
FSM	Finite State Machine

Chapter 1. Introduction

Demand for cheaper and higher performance electronic products increases year on year and is likely to keep increasing. One way of reducing the costs and improve performance is to integrate more and more functionality into a single microchip which would have previously required several discrete device on a circuit board to form a system. The integration of several devices to form a system on a single microchip is called system-on-chip (SoC) [1]. All these functional units need to be able to communicate with each other to pass data or control information in order for the system to work. They way these functional units of the SoC communicate with each other is termed as on-chip communication in this thesis. As SoC devices integrate more and more functional units the on-chip communication can become increasingly complex. A SoC could contain many different processing units, such as a digital signal processor (DSP), random access memory (RAM), read only memory (ROM), Microprocessor, analog to digital converter (ADC), digital to analog converter (DAC), universal asynchronous receiver/transmitter (UART) and various other elements. Fig. 1-1 shows an example High Definition TeleVision (HDTV) SoC from Fujitsu which integrates several different processing units and interfaces to form a complete video processing engine.

In future applications the SoC platforms could contain hundreds or possibly thousands of processing units. These could potentially be huge multi-processor arrays for highly parallel applications or perhaps custom SoCs that have many units to perform different functions. These functional units could be a mix of pre-designed blocks supplied from different design houses as well as custom designed circuitry. These functional units are often referred to as Intellectual Property (IP) cores [2]. Often the pre-designed blocks have been verified for a particular process technology and the issue of how best to connect all the functional units, or blocks, poses a challenge. When future SoC platforms are going to contain hundreds or thousands of processing units then bus based communication may be difficult to implement and scale poorly. A Network-on-Chip based communication system will be more desirable as the mechanism for passing data from one IP core to another. Such SoC platforms of the future could consist of multi-media processing applications that require real time streaming of data that require heavy bandwidth, systolic arrays of

processing units that pass data through several processing units each of which performs computation on the data or highly parallel multi-core processors.

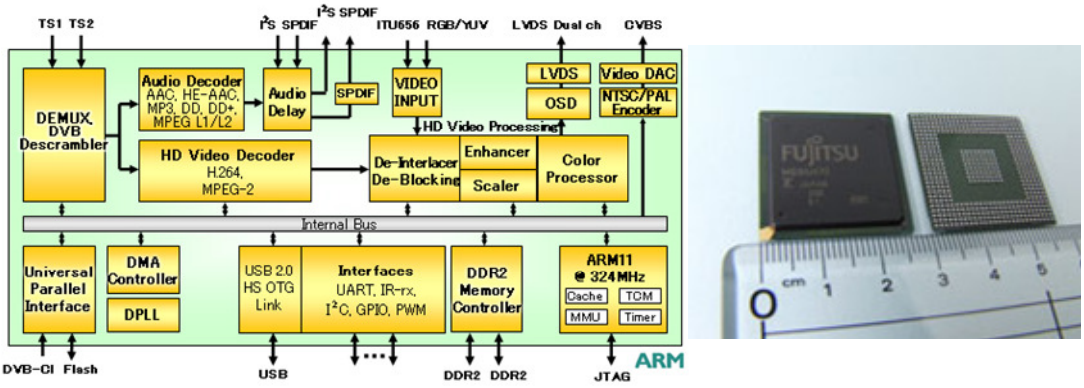


Fig. 1-1 Example SoC Fujitsu MB86H70 HDTV Processor [3]

Key challenges for future SoC designers include issues such as clock synchronization, signal integrity, process variation and power consumption [4]. Clock synchronization will become an issue due to the uncertainty of interconnect delay caused by process variation which is difficult to control. Signal integrity will be affected by the continuous smaller geometries that allow denser and more tightly packed circuits to integrate into smaller areas. Power consumption will require reduced power supplies and system level power saving mechanisms in order to reduce power. Leakage power will also become problematic as the power is wasted through currents flowing through transistors which are switched off increases with smaller geometries.

The International Technology Roadmap for Semiconductors (ITRS) [5] states that scaling of global interconnect and decreased reliability are two of the many challenges facing silicon design. The scaling of global interconnect performance relative to device performance will impact the communication mechanism and synchronization in large SoC designs. Research into Network-on-Chip is an active area which proposes the use of on-chip networks as a communication mechanism. Network based on-chip communication is a promising approach to overcome the global interconnect scaling problem stated by the ITRS. Consider Fig. 1-2, the local wires in the cores (A) reduce along with the size of the core and the transistors. The global wires that are used to connect between the cores (B) are still the same length. The cores and their associated local wiring are able to operate at faster speeds, whereas the speed to communicate between the cores remains relatively similar. Effectively the global wires remain fixed whereas the gates and local wires scale with

the process [6.]. As wire delays are effectively fixed by the particular technology being used global wires may have to have repeaters or registered buffers used along the length of the wires in order to pipeline the data so that several items of data can be travelling at the same time on different points along the length of the wire.

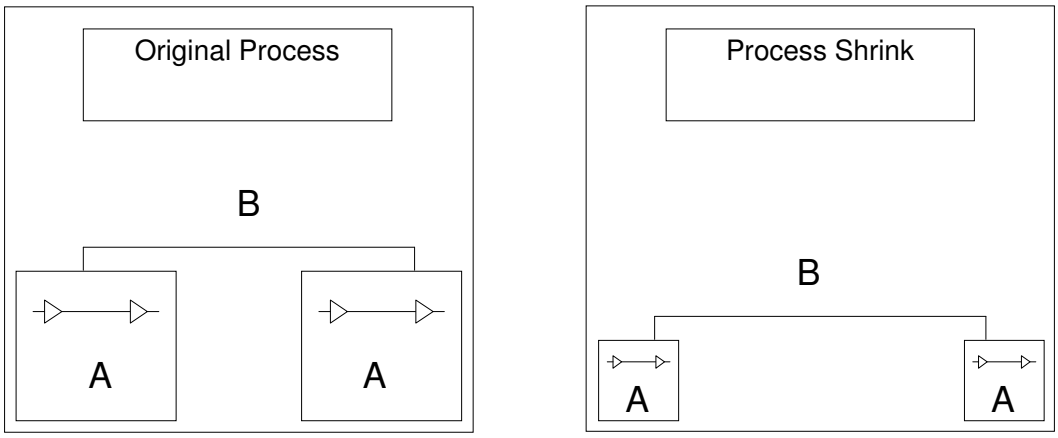


Fig. 1-2 Example of Geometry Size Reduction on Global Interconnect

Reliability is also a concern, the ITRS [5] states that technology scaling leads to more transient and permanent failures of signals, logic values, devices and interconnects. Making robust designs will become a priority as systems become too large to be effectively tested during manufacture. Such solutions include redundant logic and on-chip re-configurability for fault tolerance, adaptive and self-correcting circuits and software based fault tolerance. Shrinking geometries, lower power voltages and higher frequencies have a negative effect on reliability, intermittent faults arising from process variation and manufacturing are increasing and smaller transistors and lower power voltages means that circuits are more susceptible to neutron and alpha particles which cause transient faults [7].

This thesis addresses some of the challenges of performance and reliability in Multi-processor SoC based communication systems by providing solutions that can be implemented within the NoC framework. The rest of this chapter will explain the principles of on-chip communication and provide an overview of bus based and NoC communication. Section 1.1 introduces bus based communication. Network-on-Chip communication is discussed in section 1.2. The motivation for the work in this thesis is presented in section 1.3 along with the contributions and thesis structure in 1.4. Finally section 1.5 gives the publications that have arose from the work in this thesis.

1.1. Bus Based Communication

There currently exists many available on chip bus topologies [8-12]. Buses are the simplest and most widely used interconnection network [13]. In its simplest form a bus can be considered a shared medium with which the cores on the bus can transfer data to and from the other cores. Fig. 1-3, shows an example bus with a microprocessor, digital signal processor (DSP), random access memory (RAM) and an input-output device (I/O). Only one core on the bus can send a message at a time.

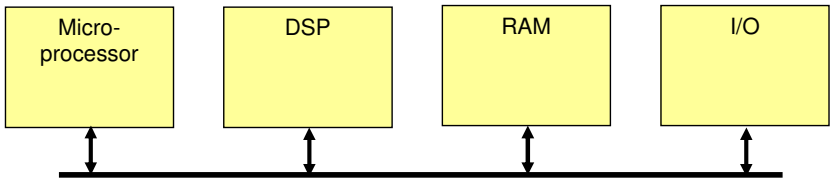


Fig. 1-3 Simple Shared Bus

Consider Fig. 1-4, if the DSP is transferring data to the RAM then the other cores cannot use the bus while this is happening. If the microprocessor tries to send a message to the I/O at the same time this will cause bus contention [14]. Typically arbitrators are used when there is more than a single core that can send a message. The arbitrator decides which core should have use of the bus when two or more cores need to use it at the same time. As only one core can send a message on the bus at any given time the bus is effectively reserved for that core until it decides to release the bus. This is one of the major problems of bus based systems and efforts have been made to try and alleviate this as much as possible. Prioritising can be used to ensure that important transactions, such as critical interrupts or control information, are performed before the less important ones, such as non-critical data transfers.

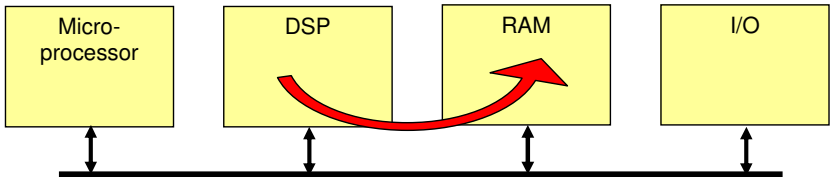


Fig. 1-4 DSP to RAM communication

Bus operations can be can be categorized into three units; cycles, messages and transactions [14]. Messages are a logical unit of information such as a read message or write message. A message requires a number of cycles to complete when being sent from the sender to receiver device. A transaction is a sequence of messages, for example to read from a memory the transaction consists of a read message and a reply with the data. Recently bus architectures have started to use split transactions, where

the request is separated from the reply. This is particularly useful for read requests where the target device may not be ready to send the data. Rather than have the bus being held waiting for the target core to send the data, the bus is released and other transactions can take place, the target core will send the data later. Bus based communication does not scale well [15] since the addition of more cores means more competition for the use of the bus, increasing the amount of time cores need to wait for control of the bus and also limiting the bandwidth. As there is some overhead in a bus transaction such as arbitration, addressing and possibly acknowledgement messages can be sent as a block or burst of information. Burst messages for example allow several data items to be read or written to a device across a bus without need to arbitrate and address each data item.

Bus bridges [16] are a mechanism that can be used to effectively split the bus into several sections. In Fig. 1-5 for example, a bridge could be introduced to split the DSP and RAM from the microprocessor and I/O. The DSP can send messages to the RAM at the same time the microprocessor can send messages to the I/O device. However, if transfers go through the bridge, such as the microprocessor sending messages to the RAM, then the transfers may have slightly increased latency due to the data having to go through the bridge and also both halves would be in use effectively blocking the use of the bus on both sides until the transfer is complete. Having bridges is also useful if slow cores are on one side of the bridge and fast cores on the other, since the fast side of the bus can operate at the highest possible speed which some cores on the slow side may not be able to operate at.

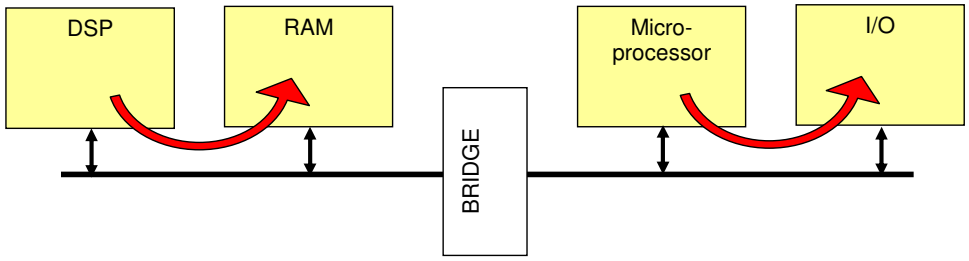


Fig. 1-5 Shared Bus

One of the most popular SoC bus solutions are the bus types defined in the Advanced Microcontroller Bus Architecture (AMBA) from ARM [8]. Over several years the AMBA specifications have been refined to meet the requirements of more complex SoC designs. The various bus types specified range from the simple bus architectures used to access peripherals to more complex multi-master high performance bus

architectures. A bus master is a core which initiates communication, a bus slave is the target core which responds or is the target of the communication. The main bus types are summarised briefly in [17, 18] which are:

- APB (Advanced Peripheral Bus) is a single master, non pipelined low speed synchronous bus used to interface to peripherals which must all be slaves. The bus can be implemented with dual read-write or tri-state. It does not support burst messages.
- ASB (Advanced System Bus) was the 1st generation AMBA system bus introduced in 1995. It is a synchronous multi-master bus and supports burst messages and any master can lock the bus as required. It does not support split transactions. It is a non-multiplexed bus with a single data bus.
- AHB (Advanced High-performance Bus) is the 2nd generation AMBA system bus introduced in 1999. It is a synchronous non-multiplexed multi-master bus. It is pipelined and supports burst messages. Also supported are split transactions where the slave can trigger the release of the bus and complete the transaction at a later time. It is a non tri-state multiplexer implementation.
- AXI (Advanced eXtensible Interface) is the 3rd generation AMBA bus introduced in 2003. It is a channel based architecture supporting multiple outstanding burst, out of order completion. Can be implemented as a shared bus, multi-layer or a mixture of both. Multi-layer is a term used to define a bus interconnect where some or all of the bus masters each have their own bus layer which connects to every slave on the bus.

The evolution of early general architecture for AMBA based designs is now discussed. Older systems which used tri-state buses which had several cores attached often had high capacitance on the bus [19] due the number of drivers attached. More modern buses are multiplexed based so that the capacitance seen by the drivers is not affected by the number of cores attached to the bus. Even with a multiplexed bus the performance of the bus is reduced when the number of cores using the bus increases, this is because the bus is effectively a shared communication medium where only one device on the bus can get control and use it at any point in time. The solution [19] was to partition the different cores onto separate buses. Fig. 1-6a shows example partitioning for early AMBA based designs. The main components of interest are the

ARM processor and the LCD Controller which all require high speed access to memory on the ASB bus. Most of the bus traffic such as CPU fetches from memory and LCD direct memory access (DMA) is on the ASB bus. The APB bus is separated or de-coupled from the ASB bus with a bridge. One of the problems with this architecture is the LCD competes with the ARM processor for external memory accesses. Devices could cause waits on the bus for a large amount of time locking out DMA accesses to DRAM.

Fig. 1-6b shows that the SoC designs from ARM split the external static memory and DRAM interfaces onto separate buses coupled via a bridge. This allows DMA to fetch data from DRAM while at the same time the processor can access ROM or peripherals. The critical path in this design is the ARM to DRAM controller which is split by the ASB to ASB bridge. A further iteration of the architecture is shown in Fig. 1-6c where a multi-port memory controller is used to alleviate issues where the LCD controller can access the SDRAM without interfering with the ARM processor transactions. Another advantage is that the high-bandwidth data transfer required for the LCD controller occurs on its own local bus, reducing power consumption.

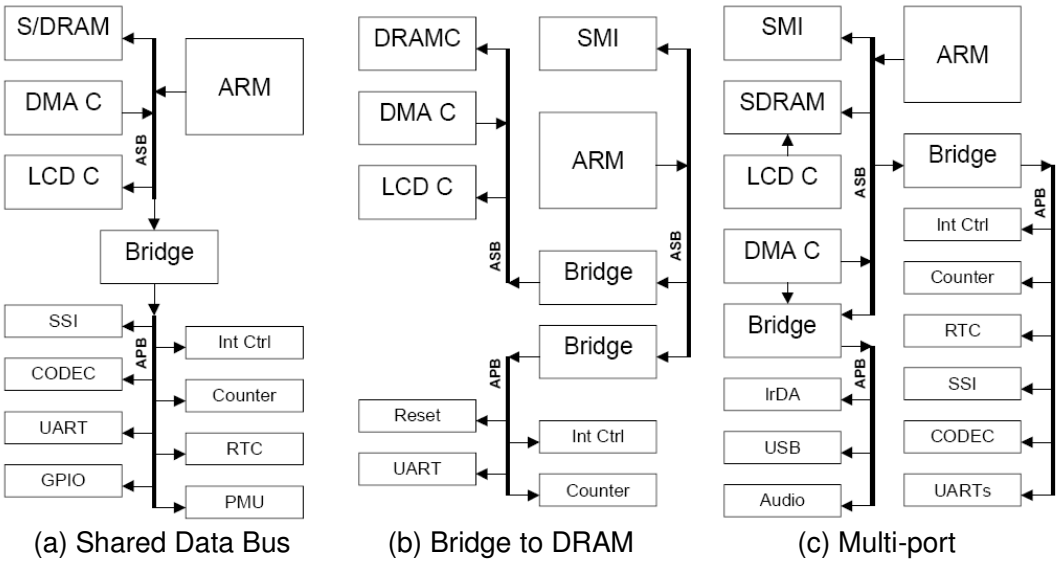


Fig. 1-6 General AMBA based design architectures [19]

The overall general trend to relieve bottlenecks in bus based communication designs is to split or partition the bus in some way. While partitioning the bus clearly alleviates some problems, over-partitioning and adding too many bridges will also cause problems such as additional latency through the bridges.

1.2. Network-on-Chip

Splitting a single bus into multiple buses and partitioning the cores onto the most appropriate bus is one way to raise throughput or avoid competition between bus masters. Another approach is to use a crossbar switch [20-22], which can connect one set of cores to another set. The crossbar switch can connect any core in one set to any core in another set and effectively become a point to point link between the two cores. Only one device can control or be controlled at any one time. The advantage of a crossbar switch is that it can support any number of simultaneous transactions between cores as long as no conflicts occur. The disadvantage of crossbar switches is that they are expensive, especially when the number of cores in a set increase as the area of the crossbar switch would increase squarely with the number of cores. Crossbar switches are also used in Network-on-Chip. However, the switches tend to have a smaller number of input and output channels and are distributed around the chip as opposed to one large crossbar switch which could connect each core with every other core. Having several smaller crossbar switches distributed around the chip as the communication mechanism also allows for better scalability since another switch can easily be added to the system without impacting the existing interconnect of switches too much.

Network-on-Chip (NoC) is a current area of research interest which proposes a network type architecture to allow the different functional units within a SoC to communicate with each other [23, 24]. A NoC typically consists of several point to point links connecting switches (routers) together and the functional blocks. For example, Fig. 1-7 shows 8 functional blocks (A-H) connected via 8 switches (1-8). Each functional block will interface to a switch through a network interface. The topology shown in Fig. 1-7 is a 2D mesh. The topology of the network is not fixed and could be a 2D mesh, Torus, Hypercube, Star [25, 26] or other common topology. The topology does not have to have a regular structure and could also be application specific topology with an irregular layout. Such application specific topologies could be optimized by the designer to arrange it so that low speed cores do not have the same level of interconnectivity or access to the network as high speed cores which require higher bandwidth. However, application specific NoCs could be restrictive if the application changes and different communication requirements are needed.

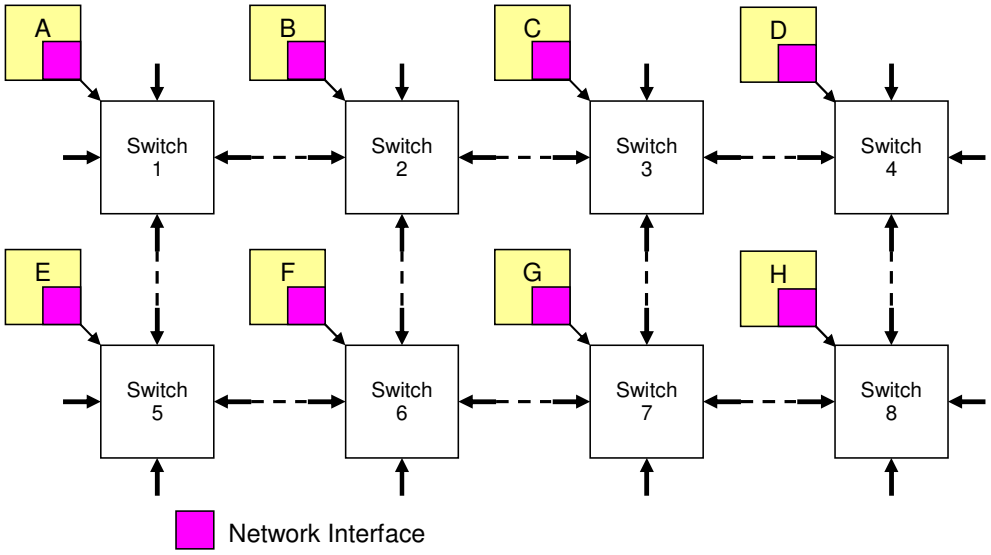


Fig. 1-7 Example Network-on-Chip

A generic network interface is shown in Fig. 1-8. The network interface converts the packet based communication to the protocols that is used by the IP cores. It is responsible for packetizing the data and scheduling the packets. The network interface will take the read and write requests from the core and transform them into packet based transactions that conform to the NoC packet protocol. Data is then moved from source to destination through the network via the switches as a packet.

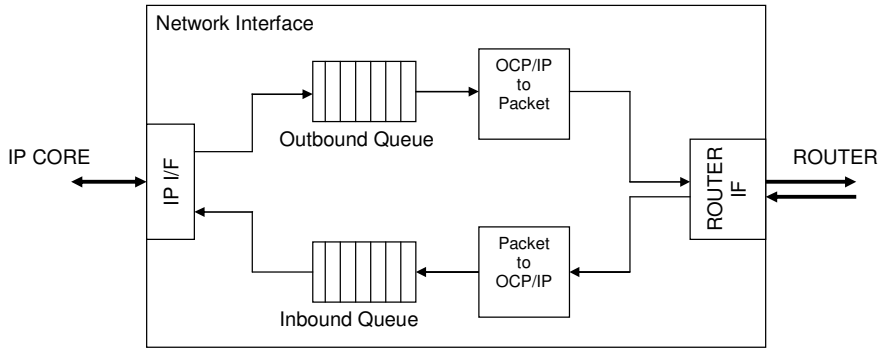


Fig. 1-8 Generic Network Interface [27]

A packet can be considered as a group of bytes consisting of header, payload and the tail [28], Fig. 1-9. The packet can be further broken down into Flits, a logical unit of certain width that the packet is broken down into. The packets of data are built by the network interface and then forwarded to the switch. The switch then looks at information in the header to decide where to forward the packet to. The packet may hop through one or more switches in order to arrive at the destination where the packet will be accepted and the data pushed back out into the destination core across a common IP core interface. A packet generally consists of:

- Header: generally this contains information about the path of the packet, the source, destination, type of packet etc. The contents and size of the header is dependent on the complexity of the NoC architecture.
- Payload: contains any data that is to be transferred from one core to another.
- Tail: contains termination codes that represent the end of packet. Also can contain a checksum that can be used for error detection.

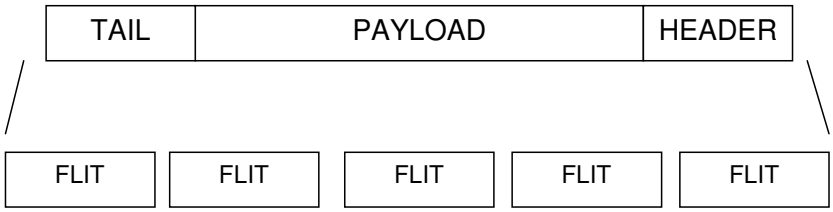


Fig. 1-9 Packets and Flits

There are three popular techniques for sending packets of data, these are Store and Forward, Virtual Cut Through and Wormhole [29]. Wormhole routing is the choice of technique for NoC in which each flit of a packet is sent, Fig. 1-10. Switch 1 receives a flit and asks switch 2 if it is ready to receive the flit, switch 2 acknowledges and the flit is sent. Each switch can hold a single or multiple flits. Wormhole routing does not suffer the latency problem of store and forward and also does not require each switch to have buffer space for the entire packet. Latency is defined in this case the time it takes for the first flit of data to go from the source to the destination. Naturally wormhole type routing will have lower latency as the flits will arrive faster at the destination since they are free to move from router to router without having to wait for the whole packet to fill the router in schemes such as store and forward.

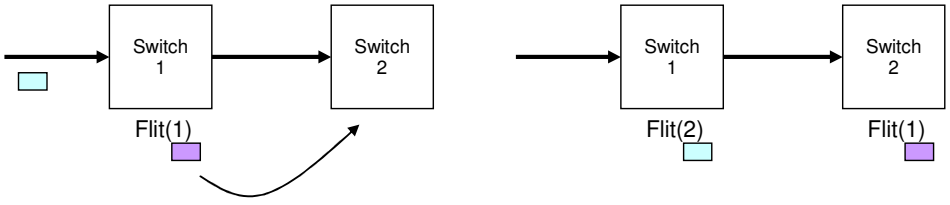


Fig. 1-10 Wormhole routing

The switches in the NoC structure are responsible for routing the packets in the correct direction based upon information in the header. Various schemes can be used from simple static XY routing [30-32] to complex dynamic (adaptive) schemes [33-35]. Simple static routing if often the choice for SoC designs due to the lower cost and more simple implementation of the system [36]. Static routing is often used when the

traffic around the NoC is known before implementation so an appropriate topology and bandwidth can be chosen. If traffic patterns are not known before design time then more complex dynamic routing that attempts to balance the routing to ease congestion could be used. Dynamic routing also is useful when faults exists such as a dead switch, a dynamic routing scheme could simply adapt and find a new route around the fault area. Dynamic routing could require complex and possibly impractical solutions which could lead to high overheads when implementing the NoC. Much work has been done with many publications and research groups focusing on this area and the reader is referred to literature [37-44] for further reading if required.

Circuit switching and packet switching techniques are the main techniques to create a connection between source and destination [45]. Packet switching is more common and is referred to as packet switching because the information to tell the switch where to send the data next is embedded in the packet. Circuit switching is when the connection is setup before the data is sent and maintained until the connection is terminated. Advantages of circuit switching are stable connection, high bandwidth but suffer from an initial circuit setup penalty when setting up the connection. Packet switching advantages include congestion avoidance and fault tolerance as each packet can take different routes, disadvantages are that there is a penalty for each packet due to header information, nodes need buffering and difficult to guarantee quality of service (QoS) [46].

NoC schemes can also exploit circuit switching and packet switching together in order to get the best of each world. Circuit switching techniques which establish a connection before the data is sent allows the user to have a Guaranteed Throughput (GT) since the bandwidth has been basically reserved until the connection is removed. Packet switching uses wormhole routing which generally provides a Best Effort (BE) approach. AEtheral is a proposed NoC that takes advantage of both GT and BE architecture [47]. This is achieved by using a GT and BE router used in parallel, the GT router has a higher priority for use of the links that the BE router. The BE router can only use the links if the GT router is not using them. Using this approach it is possible to guarantee a QoS for certain application which require a guaranteed throughput.

As technology scales down soft errors are also becoming a concern [48-51]. A soft error is where a signal or piece of data is wrong within a circuit, but the circuit itself is not broken or faulty. A soft error can occur because of alpha particles, cosmic rays and thermal neutrons [52], as well as crosstalk and signal integrity problems. Radiation hardening and error detection and correction techniques are often used to alleviate soft error problems. Radiation hardening is where the designers increase the capacitance of certain nodes in a circuit by increasing the transistor sizes so that it is less likely a particle can upset the node and affect the circuit. Error detection and correction can be done through data coding that adds redundancy to the data in order to be able to see if a single or multiple bit error has occurred.

1.3. Motivation

A general overview and principles of communication structures, both bus based and NoC based, has been given in section 1.1 and 1.2. Bus based systems are already well established and different standards are supported by many IP companies. It is likely that as the number of cores on bus based systems grow the approach to overcome the communication bottlenecks will be to partition or split the bus into several segments. Network-on-Chip may be the way forward to replace the traditional bus based infrastructure, especially as the number of cores increase. Evidence from industry shows that Network-on-Chip has already become reality. Intel has produced an 80-core chip, the teraflops research chip [53]. The chip contains 80 simple processor cores each of which contains a 5-port messaging passing router. They are connected together with a 2D mesh network. In addition each fine grain power management allows the compute engines or routers of each core to be activated or put to sleep depending on the performance required. Other companies such as Philips and Arteris [47, 54] are also active in Network-on-Chip research.

NoC appears to provide a more structured and scalable solution to the communication bottleneck in SoC. A regular topology means that partitioning like in bus designs is not needed. The packetizing of data and the fact that the packets may have to be forwarded through several switches may mean higher latencies in some situations. Switch complexity is also an issue, the more complex you make a switch the more resources such as power and area are used. Working NoCs that offer significant advantages that outweigh the shortcomings will need to be demonstrated in

order to gain a foothold in the commercial world. Some encouraging research chips from Intel and Philips may signal the start of a trend towards Network-on-Chip.

Considerable work is being undertaken in Network-on-Chip which is now a very active research area. Much of the research focuses on high-level issues such as routing and traffic performance. The motivation of the research presented in this thesis falls broadly into three areas, compression to improve bandwidth, asynchronous techniques to improve the power and simplify clocking and finally data coding to improve reliability of NoC links.

- Compression – Recent research [55, 56] has shown that compression is useful to increase the available bandwidth and also a way to decrease power. This research explores the use of compression in bit-serial links for NoC with the aim to provide a simple compression scheme that is tightly integrated into serial transmission schemes. Power and area of the compression hardware will be examined as well as the reduction in transmitted data size.
- Asynchronous - Most of the work on Network-on-Chip has been synchronous interconnect [28, 57, 58]. The application of asynchronous techniques coupled with serialization is investigated with the intention of reducing the number of wires between switches of the NoC.
- Reliability – Soft errors pose an increasing problem as technology shrinks [59, 60], with up to 80% of errors being transient. The research investigates data coding that is compatible with the asynchronous NoC links. The coding schemes are introduced as a way of increasing the resilience of single event transients on asynchronous links

It is important when evaluating the benefits of NoC that high level issues such as routing and scheduling need to be considered together with the low level issues such as physical link design, data transfer and communication protocol. This forms the focus of this thesis.

1.4. Contributions and Thesis Structure

This thesis investigates low-level improvements to Network-on-Chip communication links. With a particular focus on serialised links, compression is examined and also asynchronous techniques are considered. Chapter 2 provides a literature review and

discusses a range of recent research in on-chip communication and what is being proposed to further improve certain aspects of communication such as power, throughput and latency.

Chapter 3 presents a simple real-time compression scheme that can be used in a bit-serial link. A bit-serial link would reduce the cost of interconnect by reducing the number of wires, but at the expense of reduced throughput. To improve the throughput compression is used. The compression is based on a differential encoding scheme that is applied to a certain number of data items. The number of data items can be fixed or dynamic. For dynamic sizing and algorithm has been developed and presented. Experimental results have been shown for the transfer of decoded mpeg picture data across a link. Power simulations were performed.

Chapter 4 extends on the theme of serialization and presents a link architecture and circuitry for serialization in the asynchronous domain. Serialization is used to reduce the interconnect cost of a link and asynchronous circuitry is used to provide some of the advantages of asynchronous solutions such as the removal of global clocks in the NoC. The circuits were simulated and compared to a fully synchronous link of similar characteristics. Power, area and throughput were compared between the asynchronous and synchronous solutions. Validation on FPGA was performed to check the functionality of the circuits.

Chapter 5 presents a novel coding scheme for transient error resilience on asynchronous links. Data is transmitted using data symbols and a reference symbol, the phase relationship between the data and reference symbol is used to determine the data and the validity.

1.5. Publications

The research presented in this thesis has lead to the following publications:

- "Improved data compression for serial interconnected network on chip through unused significant bit removal", Ogg, S.; Al-Hashimi, B, 2006., 19th International Conference on VLSI Design, Hyderabad, India.
- "Serialized Asynchronous Links for NoC", Ogg, S.; Valli, E.; Al-Hashimi, B.; Yakovlev, A.; D'Alessandro, C.; Benini, L., 2008, Design Automation and Test Europe (DATE), Munich, Germany.

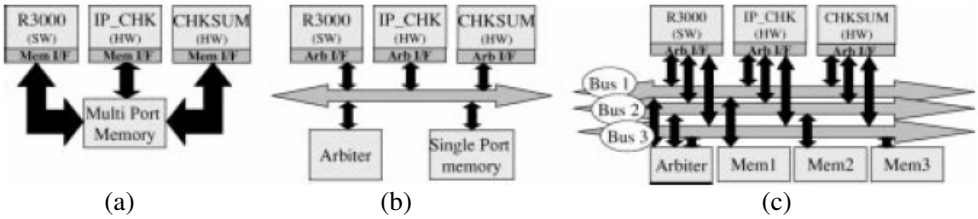
- “Asynchronous Transient Resilient Links for NoC”, Ogg S, Al-Hashimi B., Yakovlev A., 2008, CODES+ISSS, Atlanta, USA.
- “Reducing Interconnect Cost in NoC through Serialized Asynchronous Links”, Ogg, S.; Valli, E.; D’Alessandro, C.; Yakovlev, A.; Al-Hashimi, B.; Benini, L., 2007, First International Symposium on Networks-on-Chip, Princeton, USA. (POSTER)

Chapter 2. Literature Review

In the previous chapter the principles of bus based and NoC based communication have been discussed. Current and previous research on how to improve on-chip communication in SoC will be presented in this literature review. The literature review is classified into 3 sections. Section 2.1 discusses bus based systems. Section 2.2 discusses current research on NoC based systems. Reliability is discussed in section 2.3. Concluding remarks given section 2.4.

2.1. Bus Based Communication

Bus based communication has been used and still is the conventional choice for on-chip communication offering high performance buses and standardised interfaces [8-11]. The concept of a bus is well established so much of recent research concentrates on aiming to gain improvements through changing the bus topologies for improved performance, smarter handling of transactions for performance and coding of the data to improve power. Bus topologies have been explored in [61-63] for SoC applications, presenting methodologies to optimize the communication architecture for several communicating cores. An example TCP/IP interface system was used in [62] to illustrate how selection of bus topology can impact the performance. Three communication architectures were considered; a dedicated point to point link to a single multi-port memory, a shared 128 bit bus to a single-port memory and a 3x 32 bit split bus to multiple single port memories, Fig. 2-1. In their TCP/IP application the single shared bus (Fig. 2-1b) performed worse than the dedicated links (Fig. 2-1a) in terms of processing time which was up to 40% higher because of the waiting time introduced when two components try to access the same memory on the shared bus. A split bus approach (Fig. 2-1c) reduces bus conflicts but also reduces bandwidth when compared to the shared bus architecture.



(a) (b) (c)

Fig. 2-1 Communication Architectures for TCP/IP network interface system [62]

Lahiri [64] has proposed a method of optimizing communication architecture at run-time. Additional layers of circuitry called communication architecture tuners sit in an existing topology. The extra circuitry allows the more critical data to be handled differently which can lower communication latencies. The results for their examples show that the number of deadlines missed and average processing time for a system can be improved with the inclusion of communication architecture tuners.

Bus encoding is another method that is often used to reduce power or avoid crosstalk. Stan [65, 66] proposes techniques for reducing the switching power on the data and address buses. An extra bit is introduced to the regular data bus called the 'invert bit' which signifies whether the data on the bus should be inverted or not. The data is inverted if more than half of the bits change on the subsequent data item. This is a simple and effective method of ensuring that no more than 50% of the data lines will switch because if there is more than half of the bits changing in the subsequent data item the data will be inverted. This technique effectively caps the switching activity to a maximum of 50%. This could be applied to NoC and would also require extra wires to signify if the data is inverted or not. The technique could be attractive on wider parallel links as adding an extra wire would not be much additional overhead, but for narrow or bit serial links the overhead of adding and extra wire may make it not worthwhile.

The use of grey coding for address buses has also been suggested [66]. Since the addresses are generally sequential, grey coding would help to make sure that only one or a few bits change on subsequent address values. The author recommends both grey coding and bus inversion for the address bus to allow for situations when the addressing is non-sequential such as interleaving accesses or branch/jump situations. Working-Zone encoding [67] has been proposed as a way to exploit locality between addresses on the address bus. This technique used a table to keep track of preferred address areas and if the address matched a certain space instead of using the whole address it could be expressed as a working zone area with an offset. Bus encoding which exploits localities in addresses is probably unsuitable for NoC applications since there is no address bus, the address will be embedded in the header of a packet followed by the data.

Osbourne [68] extends the bus invert idea and uses it in an AMBA based bus. Instead of a single invert bit, four invert bits are used for each of the four byte lanes

on the data bus. Each byte of the 32 bit data bus can be independently inverted allowing a finer-grain control of what part of the data gets inverted. Power savings were achieved with their example sets of data reporting a 20% saving in power for 32 bit transfers. Bus invert techniques to reduce crosstalk noise, delay and power has been proposed in [69], indicating that average power on the bus can be reduced by almost 10%. Aghaghiri [70] proposes an encoding technique for memory buses using sector based coding techniques. Sectors spaces are defined that correspond to certain address areas. The data within a sector is then encoded with respect to the sector head which allows the encoding to exploit localities. Other coding techniques are used for error control schemes, Bertozzi [71] presents the idea that energy in the communication link can be reduced which has the side effect of reducing reliability. Coding is then introduced as a way of compensating for the reduced reliability allowing detection or correction to flag or recover corrupted data. It is shown that error control coding can enhance communication reliability while allowing a reduction in energy.

The Working-Zone encoding [67] is extended to include data buses in [72] which sends a 1-hot encoded offset if the data is similar or differs slightly from the previously sent data. Bus encoding to reduce cross-talk is proposed in [73] by effectively mapping data onto codewords which reduce or stop transitions on neighbouring wires. An adaptive dictionary based encoding scheme ADES is proposed in [74]. This reduces the power consumption of data buses by effectively splitting the word into three parts, an upper, index and lower part. It then maintains a dictionary of the most frequently occurring words. If the transmitted word occurs in the dictionary it sends an index and the lower part of the word. The upper part of the word is not sent. The receiver then matches the index to its own copy of the dictionary and attaches the upper part of the word. Table based encoding, where the sender and receiver keeps a table of the most frequently used data could be useful at the network interface level where only 1 destination expects to communicate with a single source. It would probably be difficult to use at the router level since sequential packets arriving at the router may not be from the same source (or sequential flits in the case of circuit switched routers) which would mean that commonality or locality of the data is reduced as the router would be effectively seeing interleaved data from several different sources each having their own most frequently occurring data words. There

is also the issue of coherence, the sources do not know about the most frequently occurring words of the other sources.

2.2. NoC Based Communication

Introduction and concepts of network on chip have been covered in chapter 1 and more detailed NoC architectures have been proposed in [15, 28, 47, 57, 58, 75-81]. Numerous publications deal with issues such as improving the routing algorithms, providing deadlock free mechanisms, fault tolerance and energy efficiency which have been proposed in the literature [37, 39-42, 82-84]. Bus based systems become difficult to scale when more cores are added. By the end of the decade the major challenges faced by designers will be to provide functionally correct, reliable operation of the interacting components [23]. On-chip communication will be a limiting factor for performance and possibly energy consumption. Synchronization of future chips will be difficult with a single clock source so a globally asynchronous locally synchronous (GALS) approach may be beneficial and are a current active topic of interest [85-87]. It is not clear from these publications if considering the high level NoC issues such as routing algorithms are beneficial without considering the low level implementation issues, in particular, the links. Higher level optimization such as more complex routing strategies, or smarter switches will almost certainly lead to more additional circuitry in the switch and increase the overhead. Lower level optimization such as the physical links could be beneficial. As the number of cores on a chip with a NoC communication structure increases, so does the number of links and switches. Future chips with many cores may need a considerable number of switches and links in order to be fully connected, especially so in a mesh type structure. Section 2.2.1 discussed NoC link level improvements and section 2.2.2 discusses serialization of NoC links.

2.2.1 NoC Link Level Interconnect

The Network-on-Chip links which connect the switches have received some research attention. Since the links are generally identical between all the switches any refinement in a link should improve the whole network. The long wires in NoC and short clock periods is problematic since the delay for driving data across a long wire grows with the resistance and capacitance [88]. At the simplest level the wires can be pipelined by dividing the long wire into several buffered segments. This buffering

could be achieved with simple inverters or registered buffers in a synchronous environment. Greenstreet [88] presents a novel way of long wire signalling in which the buffers along the length of the wire are so-called 'soft latches' that help keep all events in a close relationship to the reference event or 'fast' signal, Fig. 2-2. The basic operation of the circuit is that if the request or reference signal arrives before the data the 'fast' signal activates an extra set of transistors in the inverter in the data path and decreases the delay in the data path thereby attracting the data to the reference signal.

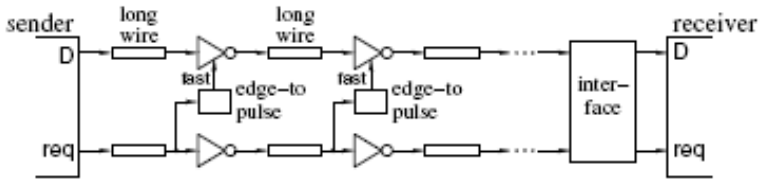


Fig. 2-2 Surfing Link [88]

Asynchronous NoC architecture has been proposed in [89, 90] where the whole NoC infrastructure is completely asynchronous. Advantages such as the removal of global clocking, lower power and higher skew tolerance have been suggested. A more localised asynchronous solution which proposes an asynchronous point to point link has been shown in [91] using wrapper circuitry and clock pausing techniques to minimise the risk of meta-stability. A delay insensitive chip area interconnect has been proposed in [92]. The scheme uses a 1-hot encoding technique to transmit 2 bits of data at a time on a 1 of 4 wires with a 5^{th} wire signifying the end of a packet. The 1-hot encoding minimizes crosstalk since only 1 of the 4 wires will have a transition on it.

The future of wires has been investigated in [93]. In this work the authors show how delay scales non-linearly with long wires and how adding repeaters can improved and make the delay linear with length. Fig. 2-3. shows how the wire delay can be changed by adding repeaters. The addition of repeaters does not come free. The authors note that adding repeaters can increase the number of vias from the upper layer metal down to the substrate and usually repeaters tend to be used in certain areas or clusters on the device, rather than allowed to be placed anywhere. Repeating a whole bus is not a trivial task and requires a considerable amount of area.

Source synchronous links have been used in [94] where the source transmits data synchronised to a local clock along with a strobe signal for timing reference at

the receiver end. This has the advantage that the clock which drives the various processing units does not necessarily have to be phase aligned with each other easing the global clock skew constraints.

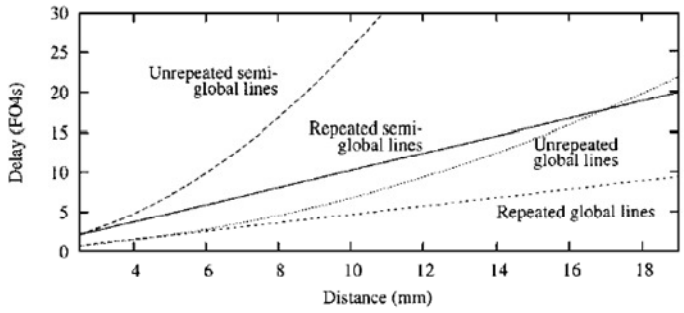


Fig. 2-3 Repeater effects on long wires [93]

Pulse based on-chip interconnect has been investigated in [95] with emphasis on reducing the global metal area footprint using serial transmission. In this work the author develops minimally spaced global wire interconnect and analyzes the effects of repeater optimization for throughput and latency. Increasing the number of repeaters increases the throughput as this effectively pipelines the wire, but the latency can also increase with the addition of repeaters and requires careful consideration to optimize.

2.2.2 Serialization of NoC Links

Serialization of the data to reduce the interconnect is also a current area of interest. Serialization leads to fewer wires, better spacing within the same area, lower static power and the possibility of better timing and synchronization. Fewer wires mean that the wiring area can be reduced, it also has the added effect of reducing the number of repeaters and metal vias that are associated with the wiring. If the wiring area is reduced it may be possible to take advantage of the freed up area by spacing out the wires more and thus reduce capacitive crosstalk between the wires since it is based on the distance between the wires. The capacitive crosstalk between wires is shown in Fig. 2-4. If the distance between the wires A and B increases then the capacitance between the wires (Cwire) should reduce. Crosstalk can increase the propagation delay of signals travelling down the wire decreasing the performance and also affect the signal integrity of adjacent wires by inducing a voltage on adjacent wires that could cause a transient which if wide enough may cause a logic level change. Crosstalk energy minimization by coding the data has been proposed in [96, 97]. However while coding reduces crosstalk power by reducing the number of

simultaneous transitions it does not reduce crosstalk itself [98], which can only be minimized by spacing the wires and reducing the capacitance.

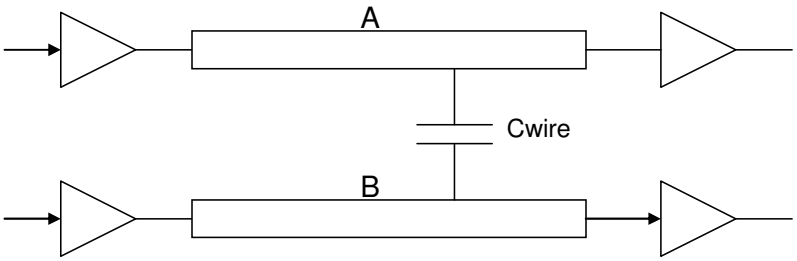


Fig. 2-4 Capacitive Crosstalk

Lower static power is achieved using serialization. Since the number of wires and therefore the repeaters are reduced the static power consumed by the repeaters is reduced. This reduction of static power is proportional the reduction in wiring for a constant repeater size. Better timing margins and synchronization through serialization are an effect of reduction or removal of wire to wire capacitance. Reducing the wire capacitance reduces the propagation delays which would allow tighter timing as clock periods can be shortened to take into account that the signal propagate along the wire faster.

Some research between the trade-offs of parallel and serial interconnect has been shown in [99] for NoC applications. The authors modelled a physical link between two points. The model of the interconnect consisted of a cascade output driver and several repeaters distributed along the length of the wire. The work shows that the leakage current per driver could increase by a factor of 5x when moving from 130 nm to 70 nm technology. Hence the more parallel drivers that are used the more leakage current could impact the interconnect between the two switches. The work continues to suggest that in their work the improvements in power in and area for serial links could be 5x and 17x respectively due to the lower number of wires and repeaters in serialized links.

A serialized scheme [100] shows a serial data transfer method implemented on chip. The serial link consists of a data and control line, Fig. 2-5. Data is synchronised by ring oscillators in the transmitter and receiver which is activated by the control line. Data is shifted out serially and counters in the in the transmit and receive circuits ensure that there is a fixed number of oscillations each time the control line is

activated. The circuit has been fabricated in 0.6 um and a 1 GHz operating frequency has been achieved on a 40 mm line length.

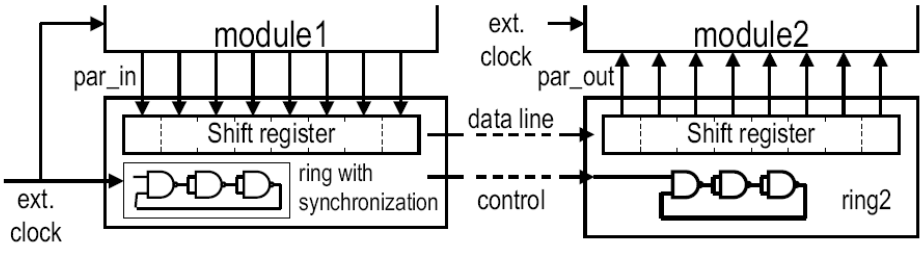


Fig. 2-5 Serialized Scheme using ring oscillators [100]

Serial solutions and ways to improve them have been carried out by Lee [101, 102]. The work shows the implementation of a 5 mm link with a 1.6 um wire pitch. Using low swing differential signalling they managed to carry a packet at 1.6 GHz with a power consumption of 0.35 pJ/bit. If a full swing link was used it is suggested that the power would increase threefold. Novel signalling and drivers can be costly to implement but using serial interconnect would mean that only a few need to be used in comparison to a fully parallel interconnect. Also proposed by Lee is SILENT [103], a method of coding the data to reduce energy. The coding scheme is differential and basically puts a '1' in the bit position when that particular bit has changed. Fig. 2-6, for example shows five 8 bit words that are to be transmitted bit-serially. The first word (W0) has 5 transitions, the second word (W1) has 7 transition in the original data. If the first word is 01010001 and the second word is 01010010 we can see that the only bits that change from W0 to W1 are the two least significant bits. The first and second words (W0 and W1) can be XORed together to form a new word (EW1) which represents the difference between the two words which will be 00000011. When the silent encoding is applied it can be seen that the second word in the encoded data (EW1) now only has 2 transitions since the only the difference has been encoded. Effectively each data is being XORed with the previous data, encoding the differences. Reduction in transitions for data values are shown to be around 40% for their examples and they do show a 80% reduction for transitions on an instruction address bus. Since NoC is a packet based method of transaction, there is no separate data and address bus so the 80% reduction in transitions would be unlikely in a NoC environment.

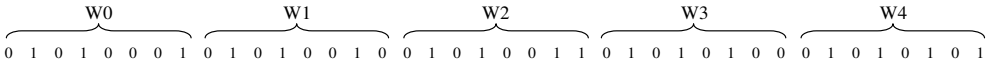
Original Data

	Data	# Tr
W0	01010001	5
W1	01010010	7
W2	01010011	5
W3	01010100	7
W4	01010101	7

Data coded with Silent

	Data	# Tr
EW0	01010001	5
EW1	00000011	2
EW2	00000001	2
EW3	00000111	2
EW4	00000001	2

Original Data sent serially = 31 Transitions



Data Coded with Silent sent serially = 13 Transitions

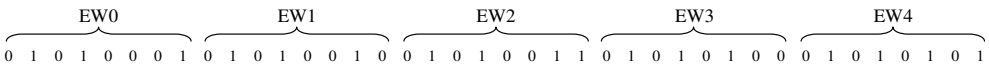


Fig. 2-6 SILENT Encoding Scheme [103]

More advanced serial techniques are presented in [104, 105]. These rely on wire pipelining where the next bit or bits of data is present on the wire before the previous bit has been consumed by the receiver. Wave-pipelining has also been shown in [106]. On-chip transmission lines can be found in [107, 108], this should allow for very fast speeds which standard lossy resistive wires cannot achieve. The fast asynchronous shift register presented in [104] uses level encoded dual rail which basically uses 2 wires for data and 2 wires for control in a differential manner, giving a total of 4 wires for a fully bit-serial data channel. The encoding works by pushing the data out serially and only switching the control if the subsequent data bit remains the same. This means that for each bit transferred two transitions will occur on either the data or the control. A very fast data rate is reported of 67 Gbps. Lee [105] has also shown a novel circuit for fast serialisation and de-serialisation called Wave Front Train (WAFT). In the work presented the conventional D-type flip flops common in shift-registers are replaced with delay elements for timing and uses signal propagation as the shifting mechanism. The WAFT serializer/des-serializer (SERDES) operates based on the fact that the delays of the serializer and de-serializer are the same, variations between the two will produce jitter and degrade the performance of the SERDES. Operation speed of 3 Gbps using 0.18 um technology has been proved to be feasible. A wave pipelined NoC interconnect implemented on FPGA has been shown in [109] using synchronous and asynchronous techniques.

2.3. Reliability in NoC

As technology scales down and integration increases errors will become more prominent [59]. Errors can be transient or permanent, permanent errors tend to be related to the manufacturing process, whereas transient errors are generally caused by surrounding environment of the affected circuitry. In synchronous systems the data is sampled every clock edge, so the data is only affected if the transient fault occurs around the same time as the clock edge. Asynchronous circuits on the other hand are always waiting for signals to change and generally rely on a change of data or control signals to function and are therefore more at risk from transient faults since no clock is used and a transient can affect the circuit at any time. Dual-rail, 1-of-4, Level Encoded Dual-Rail (LEDR) and Level Encoded Transition Signalling (LETS) [110-113] are all commonly proposed ways of data coding that allow asynchronous data transfer with completion detection. Completion detection is a means of detecting when the data is valid or not, as in the asynchronous domain there is no clock to say when signals are going to be sampled, so coding the data is often used to differentiate between 1, 0 and no data. For example, Fig. 2-7a shows a data signal associated with a clock, when the clock goes high the data is sampled as the data will be valid on the clock edge. If a clock is not used it is impossible to tell when the data should be sampled as there is no reference point to indicate when the data is valid. Fig. 2-7b, shows a common data coding method used in asynchronous circuits called Dual Rail. With dual rail two signals are used, it is a return to zero signalling method meaning that the signals have to return to zero after each valid data. If a '0' is to be transmitted one of the signals goes high and returns to zero (DATA_A) and if a '1' is to be transmitted the other signal goes high (DATA_B). Completion detection can easily be accomplished in dual-rail by ORing the two signals together which can be used to latch data. If a transient occurs on these dual rail signals then unwanted invalid data will be seen at the receiver end if the transient is large enough to generate a valid signal through completion detection.

Fig. 2-7 Synchronous with clock and Asynchronous Dual Rail

Multiple rail phase encoding [114] has shown inherent resilience to single event transients due to the nature of the of the coding such that the information is retrieved when two edges of a set of signals change in close relationship to each other and the arrival order of those signals dictate the data. If the transient occurs outside the time when the group of signals are changing there will be no effect at the receiver end as the receiver has to see all of the signals change before data is validated. A single event upset hardened pipeline interconnect is shown in [115]. A single event upset is a change in state of a node within an electronic circuit which causes an error. An scheme which uses two coding techniques combined, one for crosstalk minimisation and the other for transient resilience, is reported in [116]. This self-correcting green coding scheme uses triplication to make the data resilient to transients by using 3 signals per data bit and a majority voter decoder then should still be able to recover the correct data even if a transient is present on one of the signals. At a slightly higher level, fault tolerance can be built into the routers, such as in [117] where default backup paths (DBP) are used as a method to bypass the routers main circuitry, Fig. 2-8a. As can be seen if the crossbar switch or other critical part of the router is faulty the DBP can be used as a simple connection to bridge one input port to one output port in a permanent fashion. While this would not offer flexibility in terms of routing direction it would offer connectivity in a uni-directional path. For example in Fig. 2-8b shows a mesh NoC structure with several routers and processing elements (PE). Fig. 2-8c shows the worst case scenario if all routers fail, the DBP provides an effective uni-directional ring that would allow data to be passed around the NoC albeit at a much reduced throughput. A region based routing scheme has been proposed in [118] which combined with a segment routing algorithm can be used for regular mesh network topologies in the presence of link failures.

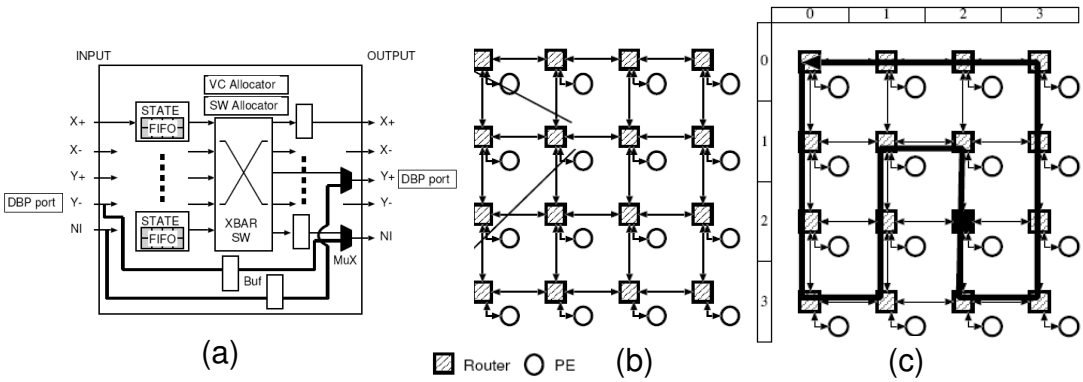


Fig. 2-8 Default Backup Paths in NoC [117]

Multi-path routing has been proposed in [119] for an in-order type packet delivery with integrated support for tolerance against transient and permanent errors. Multiple copies of the packets are routed on different paths from the source to the sink. This uses spatial and temporal redundancy to reduce the risk of the packet being affected by faulty links or routers that could otherwise corrupt or block a packet being sent along a path.

Permanent or manufacturing faults are beyond the scope of this thesis but a brief overview is given for completeness. Reliability analysis for on-chip networks has performed in [120] where models for NoC link failure due to manufacturing variation have been explored. Manufacture test for NoC has been covered in [121-124] where the NoC is used as the test access mechanism to the cores to check for manufacturing faults such as stuck at faults. Numerous cores may make boundary scan become too slow, so the NoC is used to inject and retrieve the test patterns to the various cores on chip. Recently [125] has shown a design for test (DFT) architecture for asynchronous NoC. Each router is surrounded by an asynchronous test wrapper which is used to insert test vectors or retrieve the responses, Fig. 2-9. The Generator-Analyzer-Control (GAC) unit is responsible for test vector generation, configuration and analysis of the results. The Wrapper Control Module (WCM) controls the setup and configuration the associated wrapper.

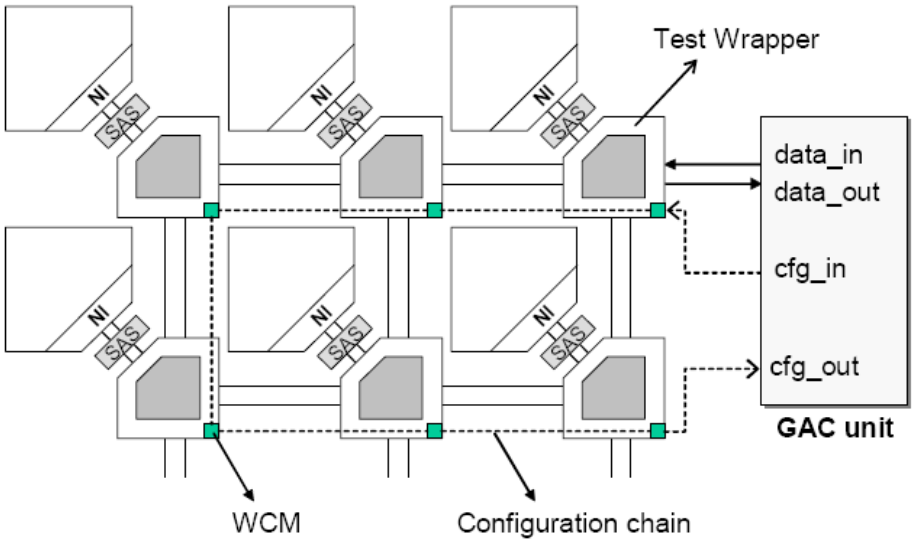


Fig. 2-9 Asynchronous Test Wrappers [125]

2.4. Concluding Remarks

Considerable work has been achieved to improve the performance and reliability of on-chip communication. The literature review has highlighted areas of research that address these problems. In bus based systems different bus topologies and bus bridges have been introduced to split the bus into several segments so that localised communication between devices on each segment do not interfere with each other. Power reduction using data coding is a popular proposed technique to reduce switching power on bus based systems. Research activity in NoC based communications has lead to several possible areas that could enhance or improve certain aspects of the NoC. Routing methods and algorithms have been proposed to improve the efficiency of packet routing. Asynchronous techniques have been introduced which help reduce power and clock skew. Serial point to point links between the NoC switches have also been considered, to reduce the interconnect cost in terms of area and power. Reliability in NoC is another promising area with coding techniques and error detection being proposed.

The focus of the work in this thesis is on the links that connect the switches together in a NoC communication structure. While the work focuses on a small part in the field of NoC, small improvements on a single link could prove valuable when considered as part of a larger NoC. The next three chapters present work that has attempted to address issues associated with interconnect cost of the links for NoC. The area of work covered covers compression, serialization asynchronous and reliability and is split among the chapters as shown in Fig. 2-10.

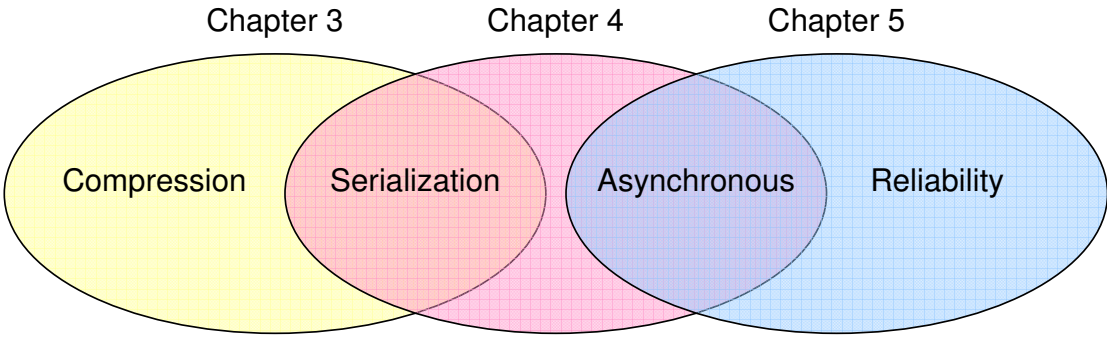


Fig. 2-10 Organisation of Chapters

Chapter 3. Bit-Serial Compression using Unused Significant Bit Removal

Long parallel links provide high data rates at the cost of large wiring area, routing difficulty and noise [126]. Leakage power in parallel links is also high relative to serial links due to the many repeaters and buffers used on long links. Serial links could also have reduced dynamic power since long parallel links will have a higher capacitance due to the wire to wire capacitance associated with long runs of closely spaced metal wires requiring higher power drivers and repeaters to achieve the same propagation delay compared to a single wire without any wire to wire capacitance. Serial links in Network-on-Chip provide advantages in terms of reduced wiring area, reduced switch complexity and routing and potential power savings [99]. Wiring area reduction reduces the real-estate cost of the interconnect and can reduce the number of repeaters required for a NoC link. Routing is made easier in serial links due to the reduced amount of wires and associated vias and repeaters. Crosstalk can also be reduced as the link does not require several parallel wires to transmit data which can have a large wire to wire capacitance and couple signals together in a parallel link.

However, serial links offer lower bandwidth in comparison to parallel schemes. Poor bandwidth increases the risk of congestion and possible lower throughput or stalling of data. This chapter proposes a simple yet effective real-time compression technique, based on removing unused significant bits which reduces the amount of data sent over serial links. The proposed technique reduces the number of bits and the number of transitions when compared to the original uncompressed data. A case study showing the results of compression on two MPEG1 coded picture data shows bit and transition reductions of data over a bit-serial link.

Section 3.1 motivates the work and section 3.2 highlights some of the existing compression techniques. Section 3.3 describes the proposed compression technique and the experimental results are given in section 3.4. Concluding remarks are given in section 3.5

3.1. Motivation

Data can be transferred from one point to another using parallel and serial schemes. Parallel transfer is when each bit of the data is transferred at the same time on different wires. Bit-Serial is when each bit of the data is transferred one after the other on the same wire.

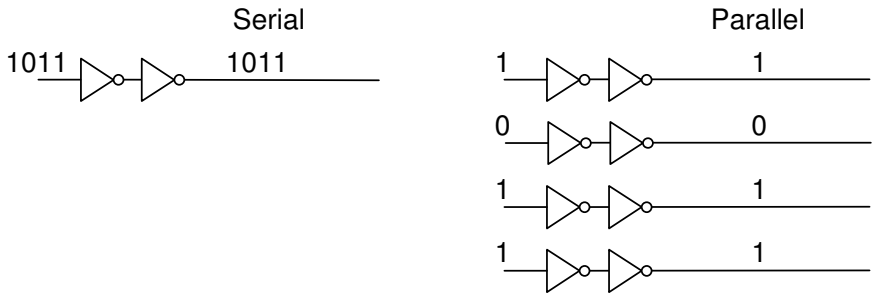


Fig. 3-1 Serial versus Parallel example

Consider the situation where the inverting buffers in the serial and parallel examples are the same in Fig. 3-1. Some initial high level conclusions can be drawn about the two methods by observation if the buffers are the same. Serial techniques will use less wires and drivers so the overall area of the communication channel could be smaller and wire area is reduced. The overall speed of parallel techniques will be faster, the data 1011 will require 4 cycles to transfer serially compared to just 1 cycle for parallel. Crosstalk will be reduced for serial since serial techniques will have no data wires next to each other whereas parallel data will have crosstalk between adjacent wires. Crosstalk is when a signal transmitted on one wire creates an unwanted effect on another wire and is caused by capacitive, inductive or conductive coupling. Serial techniques will possibly have a higher dynamic power depending on the data, consider that 1011 has just been transferred in both case and 1010 will be transferred next. The serial bus will have 4 transitions to transfer 1010. The parallel bus will just have 1 transition. The only time on the serial bus when no transitions occur is when the serial bus is transmitting all 1's or all 0's. The parallel bus has a number of buffers, whereas the serial bus only has one, hence the parallel bus could have higher leakage power. Leakage power occurs in the transistors inside logic gates due to a small amount of current that still flows even though the transistor is off [127]. Various methods can be used to improve parameters such as power, skew and transfer speed in serial links. Consider the simple methods of Fig. 3-2.

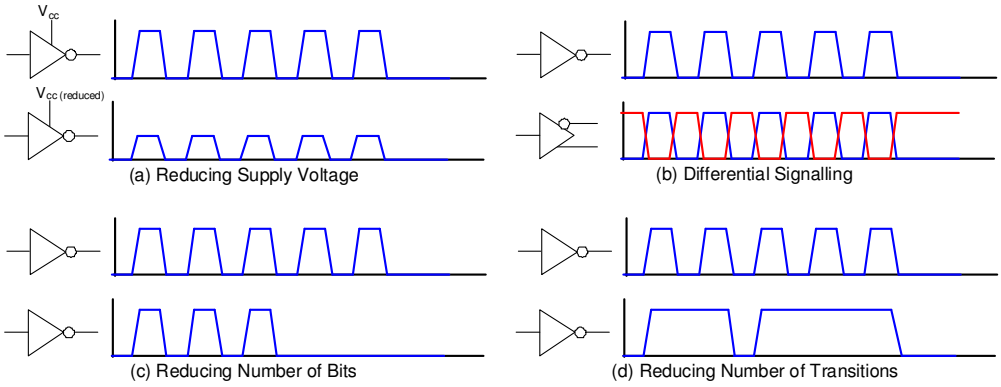


Fig. 3-2 Various Improvements to Serial Links

The voltage swing can be reduced by lowering the supply voltage of the drivers in the serial link shown in Fig. 3-2a. This has the effect of lowering the dynamic power as switching power is related to voltage, $P_{dynamic} = \frac{1}{2} \cdot V_{dd}^2 \cdot f_{clock} \cdot C_L \cdot P_{01}$ [128], where V_{dd} is the supply voltage, f_{clock} is the clock frequency, C_L is the load capacitance and P_{01} is the probability of a 0 to 1 transition.

It can be seen that reducing V_{dd} will have a square law reduction on dynamic power, this is referred to as voltage scaling. However, one must be careful as lowering V_{dd} will also mean longer rise and fall times due to the threshold voltage, which means potentially slower clock speeds. Rise time is given by $t_r = \frac{3 \cdot C_L}{K_p \cdot V_{dd}}$ [128] where C_L is the load capacitance, K_p is the CMOS process constant and V_{dd} is the supply voltage.

The driver and receiving buffer could be implemented differentially [129] as shown in Fig. 3-2b. This would make the link more immune to common mode noise. Common mode noise is when external interference affects two or more parts of a circuit in a similar way. In this case any noise affecting one wire would affect the other the wire to the same extent and because the data is obtained from the difference of the two wires the noise does not impact the ability to retrieve data. Consider Fig. 3-3, the unwanted noise (shown as spikes on the data waveform) on the standard single ended signalling could interfere with the signal enough to cause errors at the receiver end, especially if the noise causes the amplitude to cross the switching threshold the receiver circuitry. The differential signalling uses two wires, one with the signal (DATAp) and one which is the complement (DATAn). If the same noise affects both these signals a clean data signal is still retrieved by taking the difference

(DATAp-DATAn) of the differential pair as the common mode noise will be cancelled out by the process of taking the difference.

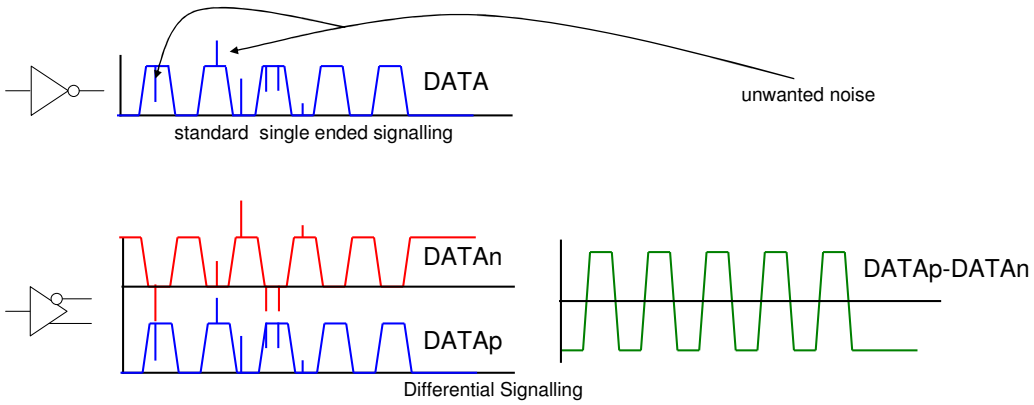


Fig. 3-3 Common mode Noise on Differential Signal

The problem with this method is that now two wires are needed and careful balancing of the wire lengths and the driving transistors would be required to keep symmetry. Reducing or removing bits in the serial link, shown in Fig. 3-2c, for a given amount of data to be transferred is a way to possibly lower power and definitely increase the transfer speed. Consider the situation in Fig. 3-4 where the data sequence is transmitted across a serial link. Each 8 bit data is shifted out LSB first across the serial link in turn. There is considerable redundancy here because the 4 MSBs in this sequence, outlined by the box, do not change. So a way of exploiting this redundancy could be used to reduce the number of bits that is sent which is the essence of the technique discussed in this chapter.

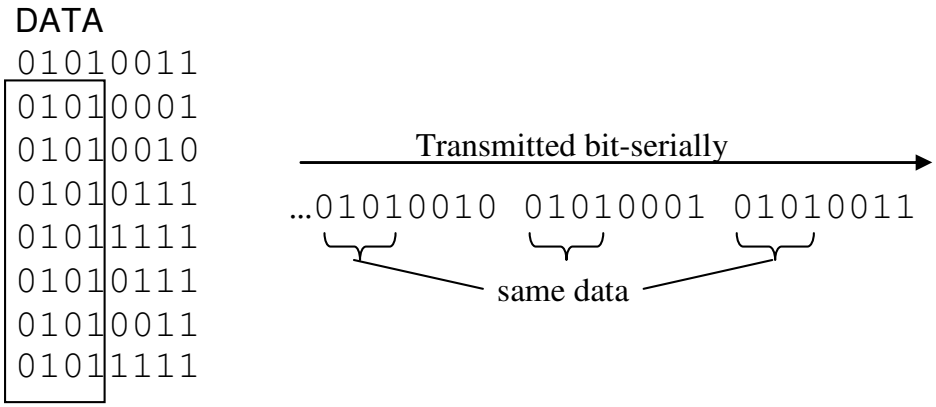


Fig. 3-4 Example of Redundant Bits

Reducing transitions is an effective way of reducing the dynamic power, Fig. 3-2d. Referring to $P_{dynamic} = \frac{1}{2} \cdot V_{dd}^2 \cdot f_{clock} \cdot C_L \cdot P_{01}$ [128] it is shown that the probability of a

0 to 1 transition occurring, P_{01}, is directly related to dynamic power. Reducing the 0 to 1 transitions will reduce power. Transition encoding which encodes data only when it changes can reduce the number of transitions. If the first data is sent and then only the differences are sent in subsequent data the number of transitions should be reduced and is the method used in SILENT [103] and is shown in Fig. 3-5. Observing the figure it can be seen that the original un-coded data has 5 transitions in the first byte and 6 transitions in the second and third byte. Consider the original data and the bits which are underlined, these are the only bits that change on each subsequent byte. If the data is encoded so that '1' is used to show that the bit has changed from the previous data we get the resulting encoded data shown. When this is transmitted bit-serially it is shown that there are 5 transitions in the first byte and 2 transitions in the second and third byte. The transition encoding reduces the number of transitions that are present in the data when the data is sent.

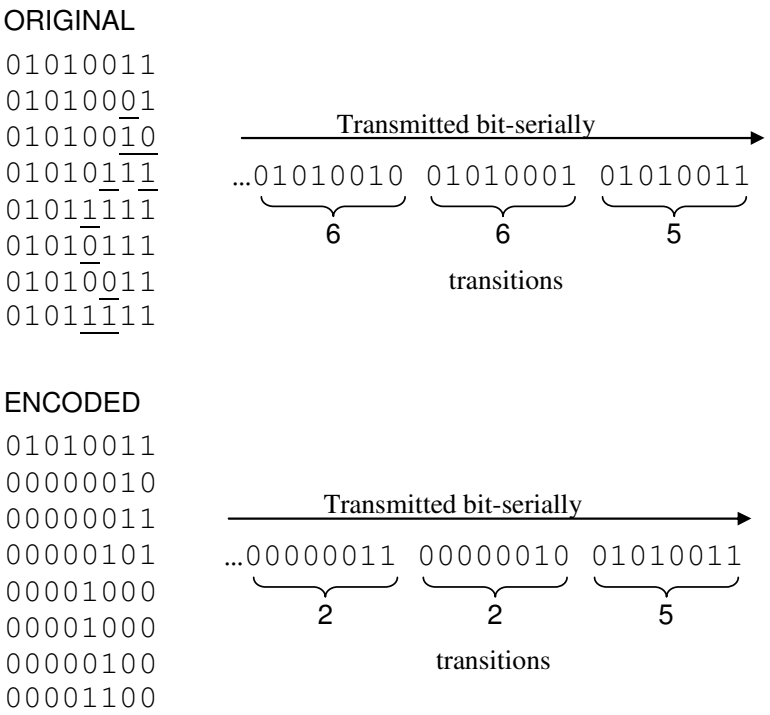


Fig. 3-5 Example of Transition Reductions

Transmission Minimized Differential Signalling (TMDS) is used in the Digital Video Interface standard, [130] is another method of reducing transitions. It works by serially XORing or XNORing the data from the LSB to the MSB. The XORed or XNORed word is selected by inspecting the number of transitions and inserting a 9^{th} bit to signify if the XORed or XNORed word has been used. Finally a 10^{th} bit is

added to signify if the 9 bits will be inverted or not as the transmitter attempts to keep an average equal number of 1's and 0's to balance the line.

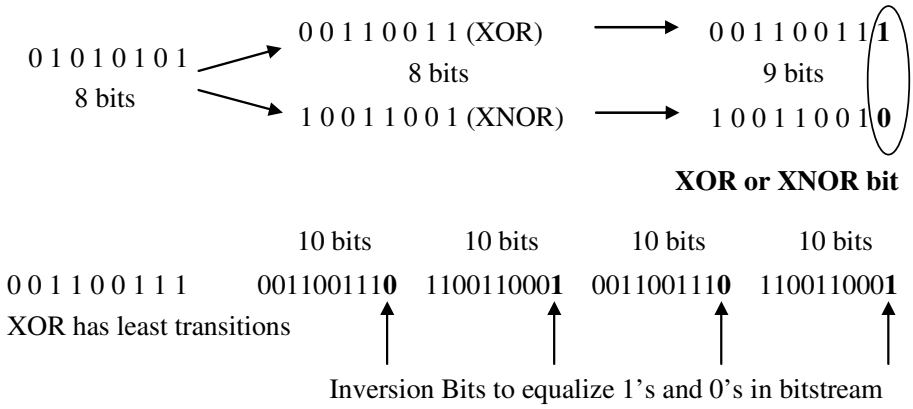


Fig. 3-6 Transmission Minimized Differential Signalling

TMDS is useful to reduce the number of transitions and to remove any DC imbalance on the cables but it comes at the cost of using 10 bits for every 8 bits of data, effectively increasing the amount of bits to be sent by 25%.

Several methods have been considered with respect to NoC. Voltage scaling by reducing the supply voltage [71], for instance, could reduce power in a single serial link or certain sets of links could be grouped together to share a supply voltage which could be altered. Differential signalling could be implemented on-chip and provide a low-swing signalling that can provide some benefits with respect to common mode noise immunity. Differential signalling would require a differential driver and receiver for each signal as well as two physical wires. Bit reduction and transition reduction appear to be promising ways of improving certain aspects of NoC serial links, such as switching power reduction and reducing the required bandwidth needed to send information. Reducing the number of transitions will reduce dynamic power and reducing the number of bits should free up bandwidth. To reduce the number of bits and transitions compression can be used.

Compression schemes such as run-length encoding could be used where there are long runs of '1's and '0's where the number of uninterrupted '1's and '0's are sent. However this would be inefficient for NoC data where unlikely that the data contains long runs of the same bits.

Huffman coding is where frequently occurring fixed sized data words are converted into shorter sized data words and rarely occurring fixed size data words are

converted into longer sized data words [131]. Huffman coding is suitable for data when knowledge of the data is known beforehand and that certain patterns of data will occur more regularly than others. This results in a smaller number of bits transmitted when the uncompressed data stream contains large amounts of the often occurring fixed size data words. One problem with Huffman coding is that if the original source data starts to contain many of the less frequently occurring fixed sized data words the resulting data to be transmitted could become greater. For example, consider the 3 bit fixed sized data to variable sized code table in Table 3-1. If the source data to be sent was 000 010 001 000 000 001 then by sending the code words 0 1110 10 0 0 10 instead of the original data we send 11 bits instead of 18 bits. However, if the original data was 101 100 011 000 011 100 the resulting data to be sent would increase to 11010 11000 1111 0 1111 11000 resulting in an expansion to 24 bits.

Table 3-1 Example Huffman Coding

Original	Code word
000	0
001	10
010	1110
011	1111
100	11000
101	11010
110	11011
111	11001

Sub-word encoding could be used to compress data. This could be achieved by sending the data and then sending only the sub-words that change. For example 8 bit words could be split into two 4 bits words and 2 extra bits could be sent to signify if the sub-word is being sent or not. Consider the example in Table 3-2, the original data consists of 6 x 8 bit words. The resulting encoding data consists of 2x bits to show if the left or right 4 bit sub-word is present or not and the sub-words themselves if they are to be sent. The resulting data is 44 bits compared to the original 64 bits. This compression technique is suitable for data where there is localisation of the data between one word and the next. The encoding does require a small amount of overhead to signify which sub-words are present for each data word. The amount of overhead is dependent on how many sub-words the data word will be split into.

Table 3-2 Sub-word Encoding

Original Data	Sub-word info + words
0000 0001	11 0000 0001
0000 1001	01 1001
1011 0111	11 1011 0111
1010 0111	10 1010
1010 0111	00
1011 0011	11 1011 0011

To study the effects of compression a simple bit-serial point to point link is studied. A simple overview of a bit serial point to point link is shown in Fig. 3-7. It consists of shift registers to perform the operation of load and unloading of parallel data and shifting data in and out serially.

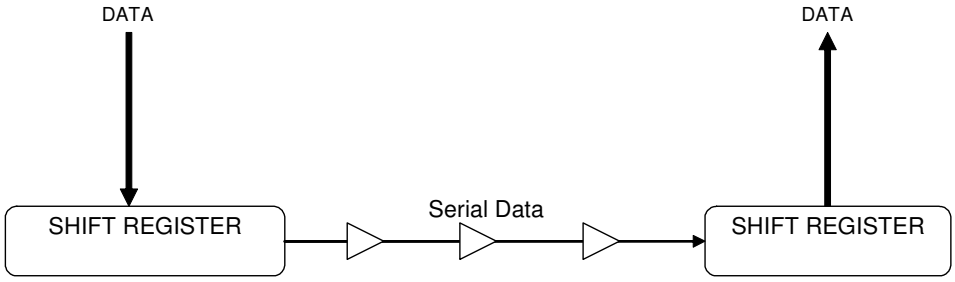


Fig. 3-7 Bit-Serial link

To compress data a compressor and de-compressor must be used. The compressor and de-compressor will add extra hardware overhead to the solution, however, the reduction in the amount of data that is sent over the serial link means that a reduced number of bits will be sent which also can lead to a reduced number of transitions that may help reduce power in the link. The area, power and performance trade offs will be considered in the experimental section of this chapter. The remainder of this chapter investigates the use of compression for bit-serial links and proposes a technique to compress data for bit-serial transmission, called Unused Significant Bit Removal.

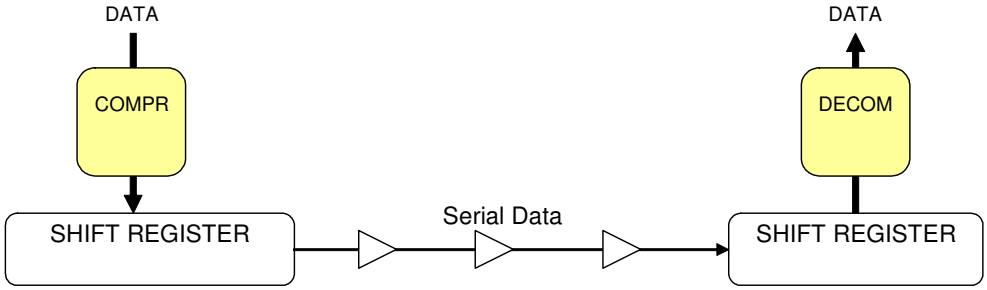


Fig. 3-8 Bit-Serial with compression

3.2. Compression

Communication bandwidth is becoming a major bottleneck in terms of system performance [55]. Compression is becoming more common for the transfer and storage of data within systems. Compression is the reduction in size of data used to represent some particular information. Compression can broadly fall into two main categories, lossy and lossless. Lossy compression is where the compression is achieved through removing unwanted or unperceivable parts of information. An example of this is JPEG [132] where high compression ratios of pictures are obtained by the removal of information which is generally not noticeable with the human eye. In effect the decompressed data is a representation that is similar to the original data.

Lossless compression is where the decompressed data is exactly the same as the original data, i.e. no information is lost. Losing information, especially instruction code or critical system data within a SoC environment would almost certainly cause a system failure and therefore lossless compression schemes will be considered for SoC communication. There are many different compression algorithms ranging from simple differential compression to schemes based on the more complex Lempel-Ziv method [55, 74, 133-136]. The different compression algorithms vary in complexity and performance in terms of compression ratio and overhead. The more complex compression algorithms tend to achieve better compression ratios at the expense of processing time or hardware, Fig. 3-9.

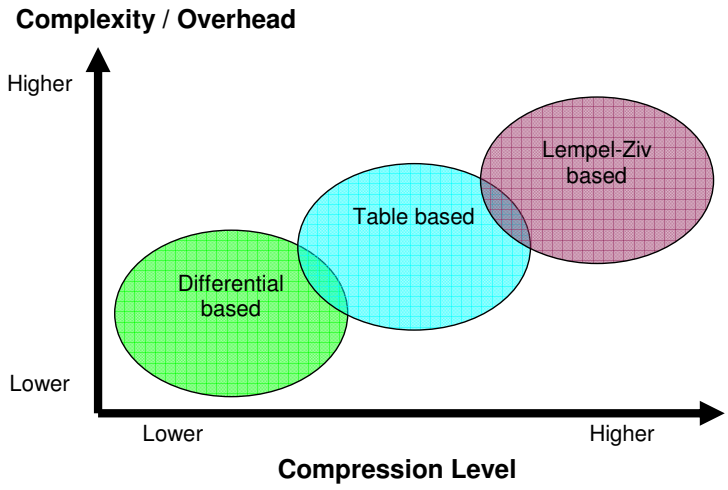


Fig. 3-9 Snapshot of common lossless compression schemes

Perhaps the simplest method of compression is differential based compression [55]. Differential based compression is where data is sent and then the difference between that and subsequent data is sent. This relies on that fact that only some of the bits will change between subsequent data. Assume there is a number of N data words of width W bits, the total number of bits needed to send would be:

$Total = N \times W$ bits.

Now assume that all the data words are examined and find that several of the bits remain the same over the N data words, let the amount of bits that remain the same be $BITS_{SM}$. There will also be some extra overhead that signifies which bits remain the same and which change, let the size of the overhead be $OVHD$ bits. The data could now be transmitted by sending the information about which bits change, the first entire data word and then just transmitting only the bits that change, thus the new total number of bits that would need to be sent would now be:

$Total_{new} = OVHD + 1 \times W + (N-1) \times (W - BITS_{SM})$ bits.

For example, assume 20 data items of 32 bits each. This would require in total 640 bits to be transferred. Now assume that the 8 most significant bits of the data items are the same and the information about how many bits remain the same uses an additional 32 bit word (meaning a overhead of 32 bits), the new total number of bits that need to be transmitted is:

$Total_{new} = 32 + 1 \times 32 + (20-1) \times (32-8) = 520$ bits

So 520 bits could now be sent instead of 640 bits. Slightly more complex is table based compression [55, 137]. In this case both a table of frequently occurring data is kept at the source and also the destination. The table could be static if the most popular data words are known before hand or it could be updated during operation for a dynamic approach where knowledge of that data is not known before hand. For each data item the source would check if it matches a entry in the table. If a match is found, rather than sending the actual data a shorter codeword can be sent which says which entry in the table should be used for the data item. This technique would suit situations where certain data words are more popular than others. It would require a table of popular words both at the transmitter and receiver side and the size of the table will affect how many entries can be used and therefore the amount of

compression that can be achieved. There exists other compression schemes such as the more complex Lempel-Ziv algorithm [133] which builds on the table based schemes even further. In this case a dictionary is maintained and a tree like structure containing strings of data is obtained. Schemes based in the Lempel-Ziv method required a large amount of data to operate on in order to achieve worthwhile compression. Schemes such as these are unlikely to be suitable for SoC communication, which is the essence of this thesis, due to the complexity of the algorithms which may lead to a large implementation overhead. Table based compression could be useful in a point to point application but within a NoC communication structure a table would be needed at a destination for every possible source that could transfer data to it. This would add considerable overhead to the NoC as the tables would probably be implemented using content addressable memory (CAM) based memory elements and several of these at every destination point would mean considerable hardware overhead.

In NoC, data is routed from one core to another through switches, the links between the switches could be parallel or serial, each of which has advantages and disadvantages. A serial link, for example, has lower wiring density and reduced crosstalk, but reduced bandwidth when compared to parallel. As discussed in Chapter 2, Morgenshtein [99] analysed serial and parallel links in NoCs and concluded that on-chip interconnects could benefit from serial links and Kimura [100] and Lee [102] have both implemented serial links in practice and shown they are viable for use in high speed, low power on chip network communication. Whilst there exists publications that deal with reducing power of parallel links [65, 66, 68], there is little reported work on reducing power in serial links apart from Lee [103] who has recently proposed a coding technique, SILENT, for serial transmission on NoC that reduces power effectively. With their example application the number of transitions is reduced by 40%. However, it is important to note that in their scheme the original amount of data is not reduced.

The previous work [99, 101, 103] has demonstrated the benefits in terms of area and power when employing serial links to connect the switches of a NoC. The motivation of this chapter is to investigate and develop a technique that will allow data transfers to overcome bandwidth limitations associated with serial links. This is

achieved through the proposed simple compression technique that exploits the bit level similarities of successive data words.

3.3. Proposed Compression Technique: Unused Significant Bit Removal

As the bandwidth in the NoC is limited there exists a motivation for compressing data in serial linked NoC to reduce the overall bits being transmitted. The reduction in bits transmitted by Unused Significant Bit Removal (USBR) would directly give spare bandwidth capacity within the network communication structure. The proposed technique is aimed at a block of data where the most significant bits are less likely to change than least significant bits, such as situations where the variance of the data is sometimes small for a certain number of words. If the variance of the data is small then it is likely that the significant bits will change less and USBR can be used to compress the data. Fixed block sizes and dynamic block sizing are considered when applying the compression to a block of data. Consider Fig. 3-10, the binary data given has the most significant bit that changes in each word underlined (Fig. 3-10a).

For the rest of this chapter data is shown pictorially as a group of squares representing bits with the most significant bit change shown as a shaded square (Fig. 3-10b). It can be seen that the two most significant bits in this example do not change, so redundant information is present which need not be transferred (Fig. 3-10c). The USBR technique removes these bits which do not change and sends some extra bits which signify what bits change and how long the block of data is. The extra bits added can be considered as additional overhead. This scheme does impose an additional penalty in terms of latency as the data is buffered up as each word is analyzed. The additional latency will depend on the block size over which the compression is applied.

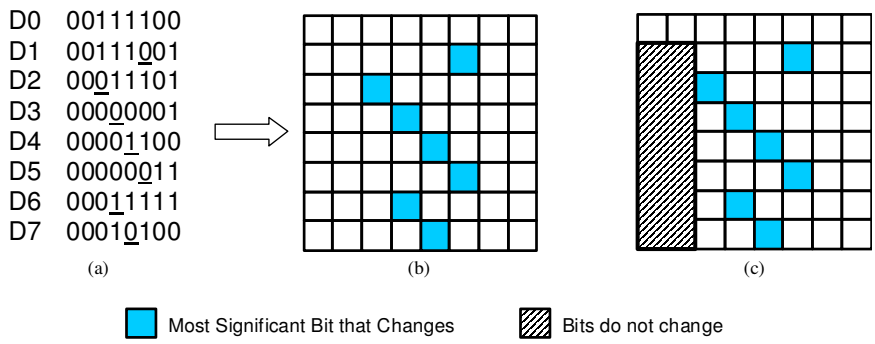


Fig. 3-10 Example 8x8bit block of data

3.3.1 Fixed Block Sizing

As an example, assume an overhead of 8 bits is used, 3 of which signify what bits change (2^3 can signify that 1 to 8 bits change) and 5 of which signify the block length (2^5 = 32, in length if necessary) giving a total overhead of 8 bits. Referring to the example in Fig. 3-11 it can be seen that 14 bits are removed from the block (Fig. 3-11b). The overhead of 8 bits is then added to the start of the block (Fig. 3-11c). The overhead would show that the 6 least significant bits change in the block and the block length is 8. Note that the 1st data word stays complete as a starting reference point for the subsequent data words which have been reduced to 6 bits each. The number of bits is reduced from 64 to 58 in this example. It is important to note at this point that the compression method is suitable for data where the most significant bits change less often then the least significant bits. For data where the least significant bits change less often a transformation of the data could be done before hand so the least significant bits are swapped with the most significant bits. For random data where all the bits change compression may not be suitable.

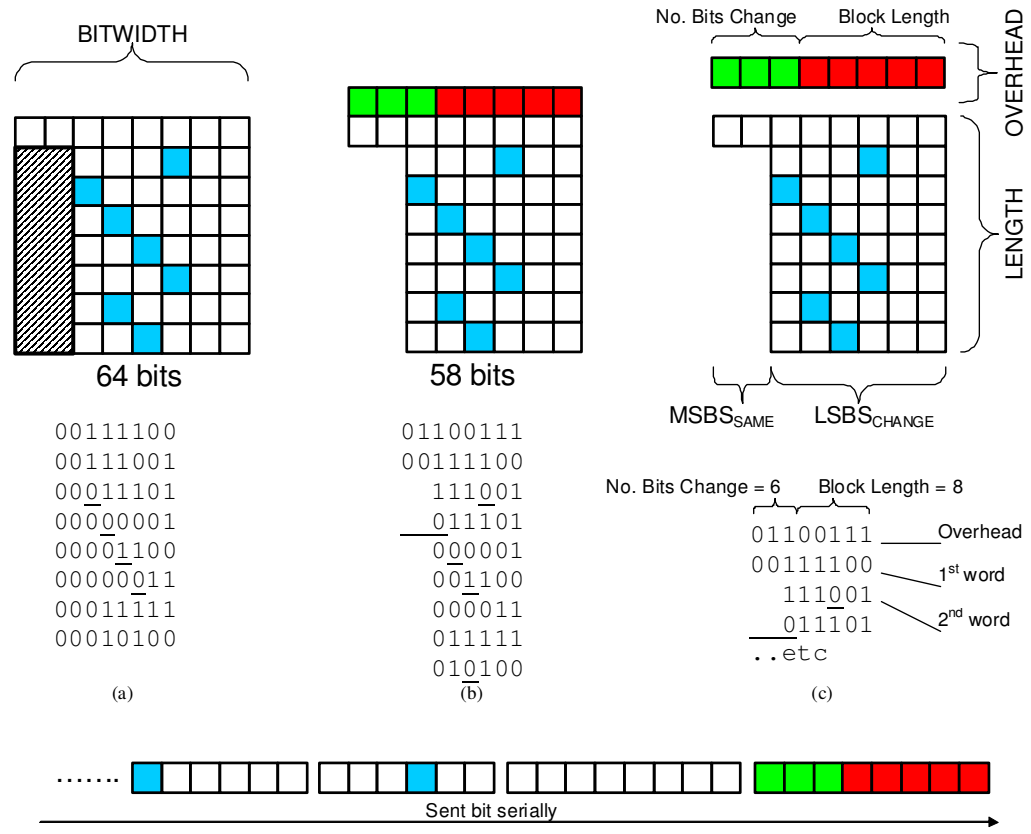


Fig. 3-11 Example of compression

A block diagram of a fixed block size compression scheme is shown in Fig. 3-12. The data is written into a FIFO which acts as a buffer or queue for the data. At the same

time the data is monitored by the MASK unit to see which bits change as the data is being written into the FIFO. The MASK unit is basically a module which examines each piece of data in turn and determines which bits have changed over a certain amount of data and then generates information that signifies which significant bits remain the same, this information can then be used to 'mask off' the bits so they are not transmitted. Once the FIFO is full the mask value is generated which says which bits have changed and which have stayed the same. This mask information is considered as additional overhead and is then loaded into the parallel to serial shift register first and shifted out. The controller FSM then loads the 1st data word into the parallel to serial shift register and shifts out the 1st word serially. The subsequent data words are each loaded into the parallel to serial shift register but the FSM now only clocks out the bits that have changed before loading the next parallel data word. At the receiving end the opposite occurs, firstly the mask value which holds the information about which bits change is loaded. The 1st data word is then shifted in and clocked out as parallel data. The FSM then uses the mask information to control the serial to parallel loading at the correct time when the appropriate number of subsequent data bits have been shifted in. As the subsequent bits are shifted in they are written to only the bit positions they correspond to and form the next data word which is then clocked out in a parallel format. With fixed block sizing the amount of data that is collected before a decision is made about which bits changed is fixed. The FIFO that collects this data must be large enough to accommodated this.

In the examples it has been assumed the overhead data, which contains information about the bits that change and in the case of dynamic block sizing the block length, is the same as the bit width. So for example if we have 16 bit data then our overhead will also be 16 bits. For fixed block sizing if the block size is hard coded into the transmitter and receiver we can provide information about which bits change using 4 bits (2^4 = 16) which leaves 12 bits of the overhead unused. However, hard coding the block size into the transmitter and receiver means less flexibility should the block size need to be changed.

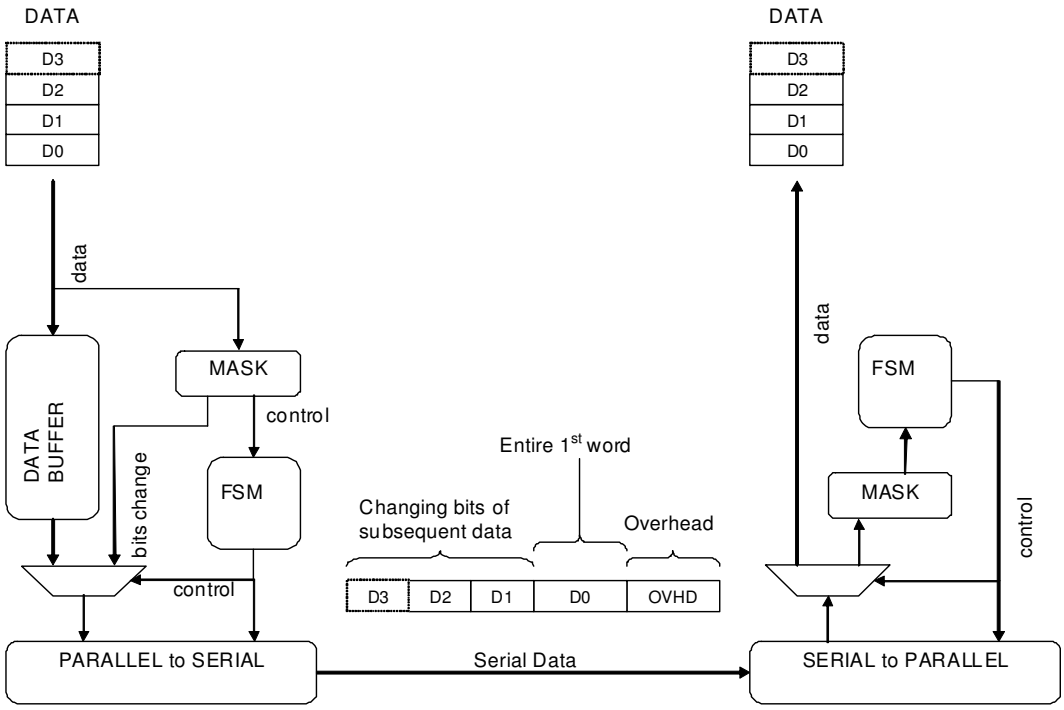


Fig. 3-12 Generic Diagram of Compression Scheme

Up to now the block length has been considered as a fixed. The block length in the example is 8. To determine what the optimum block length is without knowledge of the data to be compressed is not an easy task. A general guideline is to make the fixed block size similar to any inherent groups of consecutive data that show minimal data variance. As stated in Appendix A, a fixed block size of 64 is used for MPEG picture data. This is because picture data is often processed in units that consist of six groups of 64 words, the less complex the picture, the less variance there generally is within each group. It is important not to make the block size too large as this would impact the size of any buffer which has to hold the data. Increasing the buffer size will increase the area cost of implementation. Conversely, making the fixed block size too small will lead to compression inefficiencies where the any potential gains from removing redundant bits would be impacted by an increased amount of overhead bits being used for the increase in the number of blocks resulting from higher fragmentation of the data.

3.3.2 Dynamic Block Sizing

An alternative to fixed block length is to dynamically alter the length on a block by block basis. This is useful for situations where there is no inherent grouping of data that can be seen. If the block length is fixed the implementation of performing the proposed compression technique becomes simple since the block length stays

constant. Using an algorithm to allow dynamic block sizing based on information about the data can be done but this will impact the area of the transmitter and receiver since the circuitry would be more complex. Dynamic block sizing will now be discussed in more detail.

It is common to buffer data within the network switch interface to store data before being packetized to make sure data is transferred efficiently. For dynamic block sizing a queue such as a FIFO could be used to provide information about how to compress the buffered data. The information in the queue would consist of some overhead information which signifies the block length and the number of bits which change or stay the same. Extra circuitry would be needed to gather information about the most suitable way to organize the data into blocks and provide the number of data words over which compression is applied. This extra information about the length of the block would also have to be transmitted along with the number of bits that change so that the receiver would know how many data words to decompress using the current information of how many bits change and how many remain the same. The extra circuitry would require some sort of numeric addition and comparison in order to find out what block size or number of words to apply compression. In order to make a decision about block sizing an algorithm is used to determine the block size base on the number of bits that change over a particular number of data words. In order to keep it simple the algorithm only examines each data word once when it is written into the FIFO. It may be possible to achieve better optimization of the algorithm if several passes of examining the data are done but this would require the data to effectively wait in a memory while there was several sweeps through the data in order to find the best block sizing and therefore the transmission of the data would not take place until the block sizing had taken place. By just examining each data word once when it is being written into the FIFO we know that the data will be ready to be transmitted as soon as the algorithm determines the first block size.

A simple algorithm is proposed that effectively uses pointers which point to the start of 3 successive minimum length blocks in the data. Using 3 pointers allows the algorithm to make two decisions, to continue to merge blocks together or to store information about the current merged block and start a new block. To get the minimum length for a block it is necessary to know when compression will be

worthwhile. Referring to Fig. 3-13 the following equation must be satisfied in order to achieve compression.

$$((LENGTH - 1) \times LSBS_{CH}) + BITWIDTH + OVHD < LENGTH \times BITWIDTH$$

where, LENGTH = length of the block,

$LSBS_{CH}$ = number of Least Sig. Bits that change

OVHD = overhead in bits

BITWIDTH = bit width of the data words

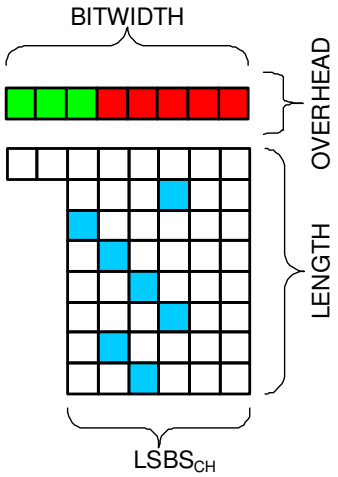


Fig. 3-13 Compressed Data Format

The basic outline of the algorithm is given in Fig. 3-14. It consists of an initialization phase followed by an evaluation and update loop.

```
1     // Initialize the pointers by getting the first three minimum blocks
2     GetNextBlock(p0)
3     GetNextBlock(p1)
4     GetNextBlock(p2)
5
6     while (not at end of data) // Main loop
7     {
8           // bias = 4
9
10          // Net savings merging p0 + p1
11          // = - potential savings lost + overhead + bias
12
13          // Net savings merging p1 + p2
14          // = - potential savings lost + overhead
15
16          // Evaluate the merging options
17          if merging block 0 and 1 gives best savings
18                p0 = Merge(p0,p1)
19                p1 = p2
20                GetNextBlock(p2)
21
22          else if merging block 1 and 2 gives best savings
23                store p0 info in queue
24                p0 = Merge(p1,p2)
25                GetNextBlock(p1)
26                GetNextBlock(p2)
27
28    } end while
29
30    GetNextBlock(p)
31    { // Gets the minimum sized block that will achieve compression }
32
33    Merge(p,p)
34    { // Merges the two blocks together}
```

Fig. 3-14 Algorithm for dynamic block sizing

An example initialization, evaluation and update cycle is shown in Fig. 3-15. First, three minimum length blocks are found. These three minimum length blocks are then evaluated for two possible merging options, merging block 0 and block 1 or merging block 1 and block 2. In the example shown in Fig. 3-15, merging block 0 and block 1 results in the loss of 4 bits. However, the merge operation would also remove one set of overhead bits, in this case 8 bits. Furthermore a bias value is used when calculating the potential savings when merging block 0 and block 1. This is done to encourage the algorithm not to fragment the data into too many blocks. The resulting net savings for merging block 0 and block 1 will therefore be -4+8+4 giving a total of +8. Merging block 1 and block 2 in Fig. 3-15 shows that 9 bits are lost and 8 bits are saved through removing overhead bits. This gives a total net saving of -1. Merging block 0 and 1 gives the best overall net bit savings. The merge is performed and the pointers are updated. This whole process is continuously performed again. Anytime block 1 and block 2 is merged the information about block 0, the length and number of bits that change, is stored in a queue to allow the already buffered data to be compressed. The queue will contain the length and changing bits information that allows the data in the buffer to be compressed.

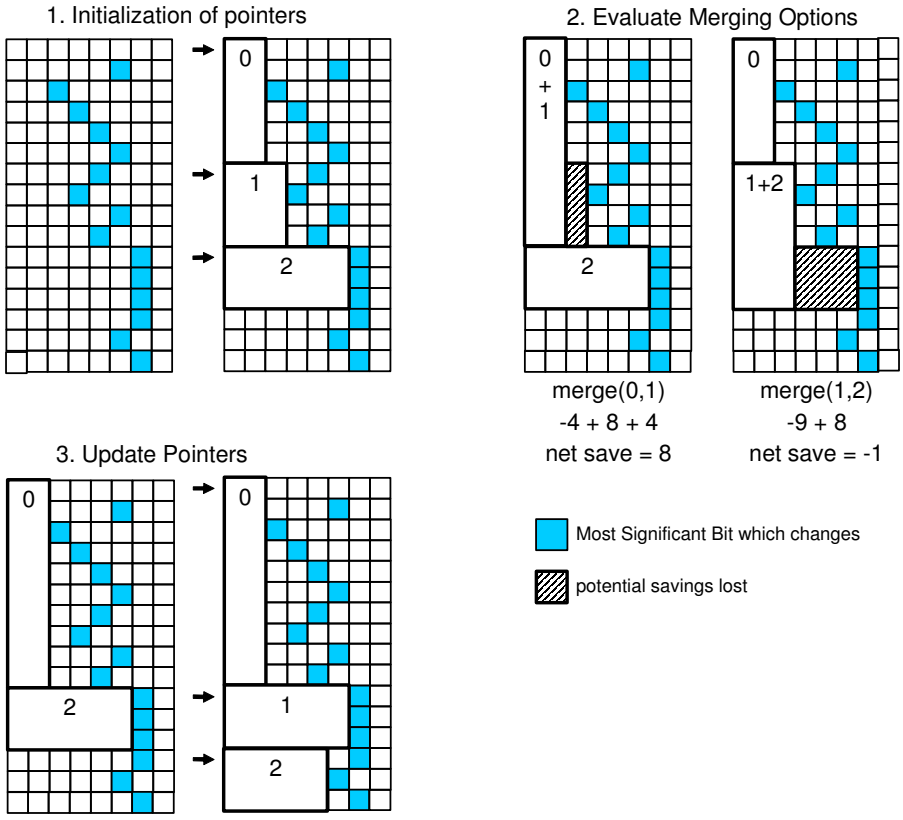


Fig. 3-15 Example initialization, evaluation and update cycle

Fig. 3-16 shows a possible implementation of the USBR compression technique in hardware. The packet header generation is ignored in this diagram and just the payload data is considered for clarity. The general structure outlined shows a transmitter consisting of a buffer, block sizing unit, queue, controller and parallel to serial converter. The receiver consists of a serial to parallel converter where each bit can be addressed, and a controller. For the transmitter, the data is written into a buffer from the core as normal but at the same time the block sizing unit is collecting information on the data and working out ways to try and compress the data. As the block sizing unit finds the best way to split the data into blocks the information is written into a queue. The controller then takes information from the queue regarding the length and number of bits that change and uses these to drive the parallel to serial converter correctly. The parallel to serial converter shifts out the overhead information from the queue, the entire first data word and then only the bits that change within the block for each word.

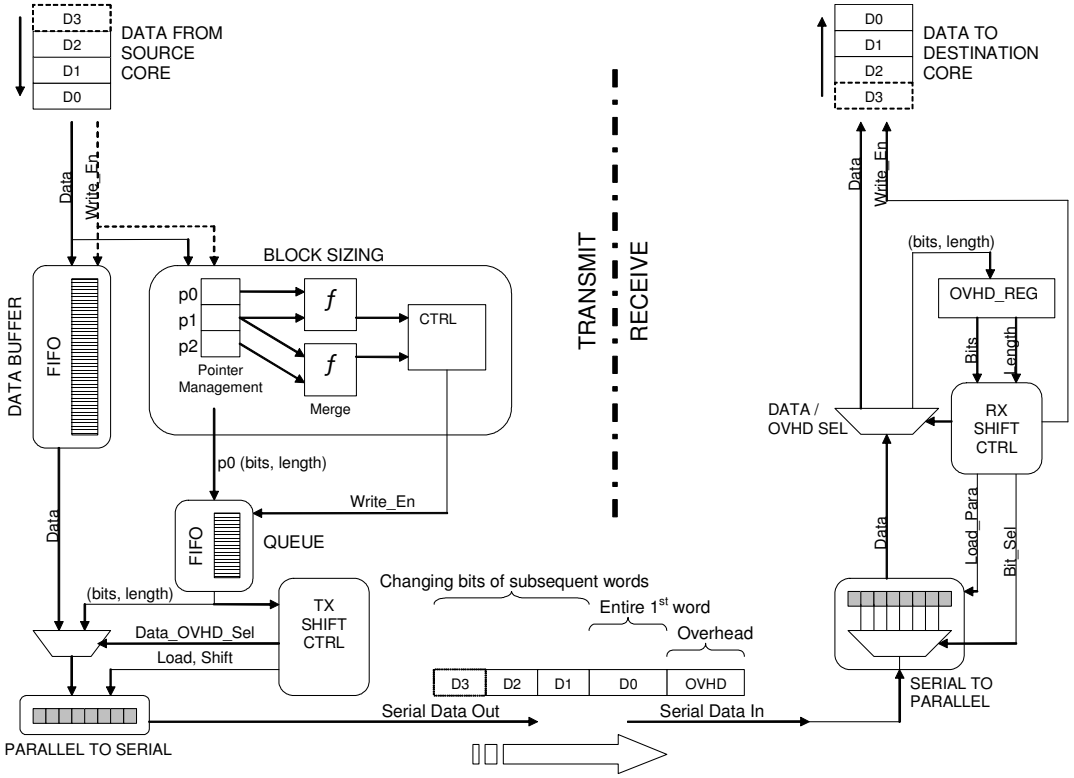


Fig. 3-16 Implementation for dynamic block sizing, USBR

The receiver shifts in the overhead bits that contain the information which specifies the bits that stay the same and the length of the block. The first data word is then shifted in and on subsequent data words only the bits which change are shifted in to the appropriate bit position. Each time enough bits have been shifted in to form a

valid word whole uncompressed data is clocked out in parallel to be used by the receiving core. This continues until the end of the block is reached and then the whole process is repeated on the next block. An interesting observation with this compression method is that the receiver does not have to buffer data in order to perform decompression. Each word can be extracted in turn as soon as the necessary bits have been shifted in. This is useful with memory accesses since as soon as the packet has finished being sent across the serial link the memory should contain the updated data.

The number of bits in the overhead in our examples has been set the same as the bit width of the data. However, it is possible to optimize this further as we can use some of the bits to say what bits have changed and then optimize the remaining bits to say the block length if we constrain the block length to particular sizes. For example with a data width of 16 bits we can use 4 bits to say which bits have changed leaving 12 bits for the block size, 12 bits allowing a block size of up to 4096. Using 8 bits for the block size we could have block lengths of up to 256. So if the block sizes were constrained to a maximum length of 256 we could just use 12 bits in the overhead (4 bits to say which bits change in the data and 8 bits for block length).

3.4. Experimental Results

In order to confirm the effectiveness of USBR in compressing data over a serial link we applied the technique to two MPEG intra-coded pictures shown in Fig. 3-17, a series of samples from a sinwave with 88 samples per complete sinwave and some randomly generated data. Each example data is used with a 16 bit and 10 bit fixed point precision. The resulting bit and transition percentage reduction for each example is presented. The amount of overhead was set at 16 bits per block for the fixed block size compression and for the dynamic block sizing compression and the bit-width set at 16.

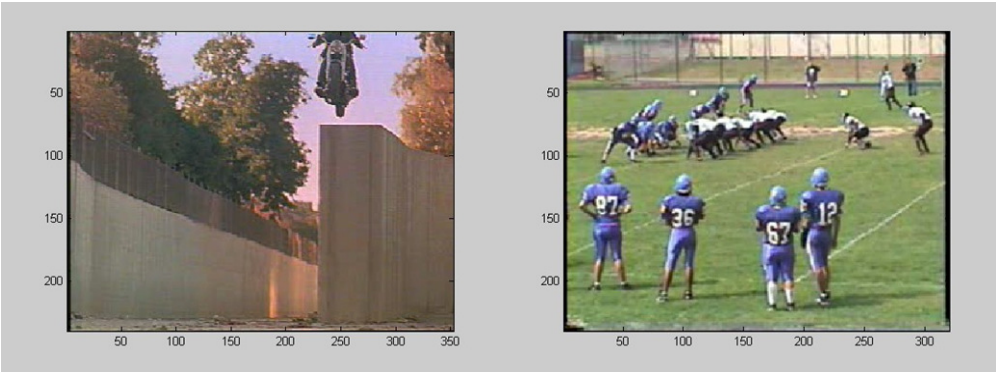


Fig. 3-17 Intra-coded pictures from MPEG stream bike.m1v and football.m1v

To count the number of bit or transitions a RTL synthesisable VHDL model of the uncompressed, Fixed and SILENT implementations was synthesized and a gate level net-list was generated, a brief overview of the VHDL modules can be found in Appendix A. A C# application was written for the dynamic scheme to count the bit reductions for the dynamic algorithm using the same source data that is supplied to the VHDL testbench. In the test bench two modules were used which monitored the serial link, one to count the transitions and the other to count the number of bits. The test bench and net-list were simulated in Modelsim. The test bench and input data was common for all implementations so the stimulus and test data remained the same. The test bench was run on the VHDL implementations and the bit reduction and transition reduction obtained for:

- Uncompressed; Serial link without compression
- Fixed; Serial link with fixed sized block lengths
- Fixed + SILENT; Serial link with transition reduction [103]
- Dynamic; Serial link with dynamic sized block lengths

The reduction in number of bits is shown in Table 3-3 and the reduction in transitions is shown in Table 3-4. The example source data that was used was sent through the link via the test bench was generated in several different ways. The arnie1 and football1 data was generated by extracting an intra-coded picture from an MPEG video stream, the data is basically a text file consisting of integer numbers which represent the chrominance and luminance macro block information in an 8 column format. The sinwave data consisted of values generated from an excel spreadsheet which created a sin wave with 88 sample points per sin wave period. The random data

was generated using the random function in excel. The integer numbers range between 0 and 65535 for 16 bit precision and 0 to 1023 for 10 bit precision.

Table 3-3 Amount of Data Transferred (Bits)

SOURCE	Uncompressed	Fixed	Fixed+Silent	Dynamic
arnie1_10bit	2027520	999792	999792	1144214
arnie1_16bit	2027520	1683279	1683279	1720560
football1_10bit	1843200	1055646	1055646	1096359
football1_16bit	1843200	1719288	1719288	1663466
sinwave88samples_10bit	11264	7786	7786	7552
sinwave88samples_16bit	11264	11440	11440	11283
1024random_10bit	16384	10592	10592	10358
1024random_16bit	16384	16640	16640	16416

Table 3-4 Number of Transitions

SOURCE	Uncompressed	Fixed	Silent	Fixed+Silent
arnie1_10bit	675890	483225	338558	326238
arnie1_16bit	990919	830473	643514	637716
football1_10bit	629130	506292	347348	338026
football1_16bit	903604	839816	646606	645462
sinwave88samples_10bit	3760	3672	2634	2634
sinwave88samples_16bit	5648	5652	4846	4854
1024random_10bit	5718	5217	5640	5173
1024random_16bit	8153	8159	8269	8275

Fig. 3-18 shows the average reduction of the bits being transmitted for the example data for 10 and 16 bit precision with a block bit-width of 16. The uncompressed labelled bar shows the original amount of data normalised to 100%. The other two bars show the resulting transition reduction for Fixed and Dynamic implementations. SILENT is excluded here as it reduces transitions not the amount of data and therefore would be the same as uncompressed. For 16 bit random and sinewave data there is no compression (Fig. 3-18, 1024random_16bit and sinwave88samples_16bit), in fact there is slight expansion in size. This is to be expected since random data cannot be compressed and a full swing sinewave will use all the available values between 0 to 65535 for 16 bit data and therefore all bits will generally change within the 64 data samples resulting in no MSBs that stay the same and no compression. The Dynamic algorithm performs slightly better on the 10 bit random and sinewave data as well as the 16bit football picture with a 1-3% reduction is data size. However on the remaining picture examples the Fixed scheme performs superior with a 2-7% reduction in size. The fixed scheme performs quite well in

comparison to the dynamic, but this could be due to the fact that the fixed scheme uses a block size of 64 over which to compress the data, this is exactly the same size as the yuv block size in decoded mpeg data so there is some natural synergy in terms of the localization of the yuv data fitting perfectly into the block size of the compressor. When the data does not have any inherent relation to the block size of the compressor (such as the 10 bit random data or sinwave) the dynamic scheme does show a slight improvement, albeit 1-2%.

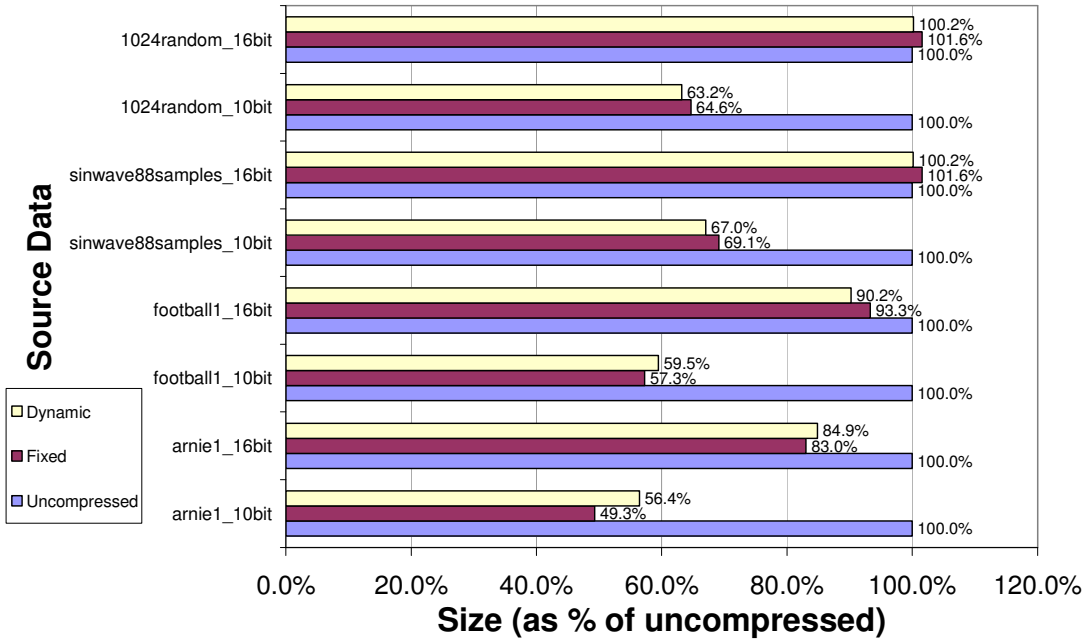


Fig. 3-18 Average Reduction in Bits Transmitted

To examine how the proposed technique reduces the number of transitions and therefore the power, Fig. 3-19 shows the average reduction of transitions for example data for our proposed compression using a fixed block size of 64 and using SILENT[103] which is a technique specifically used for reducing transitions and also a combination of both to show the effect of compression and transition reduction techniques together. As can be seen, in the cases where the data does not compress such the random and sinwave 16 bit examples (Fig. 3-19, 1024random_16bit and sinwave88samples_16bit) the proposes compression results in no transition reductions, in fact a slight increase is seen of 0.1% which is due to the extra transitions within the extra overhead. It can be seen that the transition reduction works well on the yuv picture data (Fig. 3-19, football1 and arnie1) resulting in around 50% of the original amount of transitions for the 10 bit precision arnie1 picture data and 72% of the original transitions for the 16 bit football picture data. Our proposed

compression does reduce the number of transitions also, 72% for the arnie1 picture data and 92% for the football1 picture, but does not achieve the same performance as SILENT.

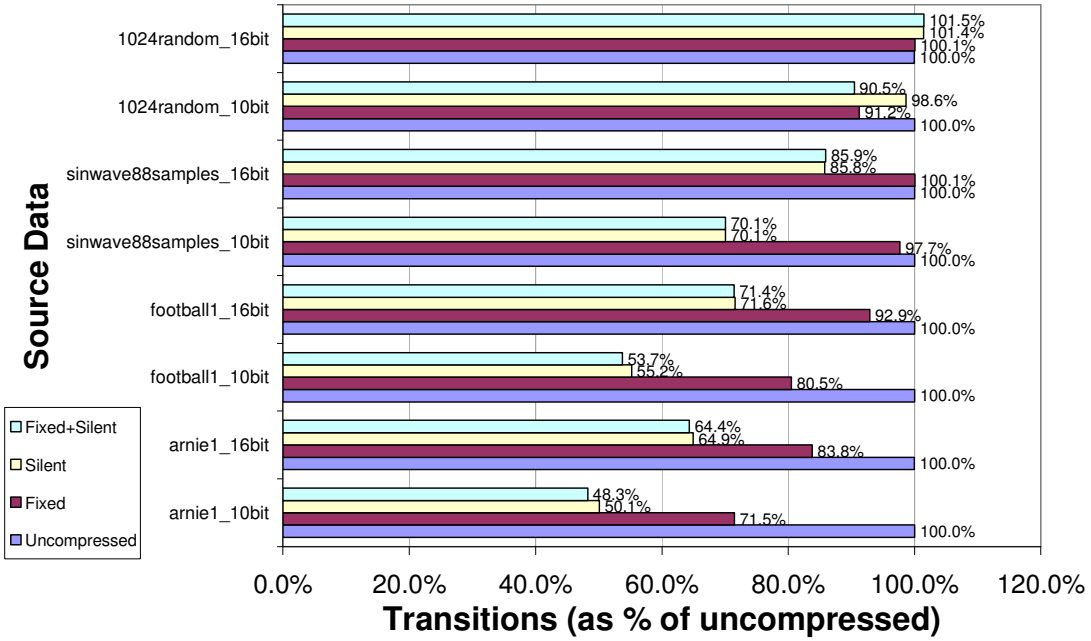


Fig. 3-19 Average Reduction in Transitions

In order to achieve similar transition reduction as SILENT we can combine the proposed compression with SILENT and as can be seen the number of transitions is now reduced to an amount similar or slightly better than SILENT, 48% for arnie1 10 bit picture and 71% for football1 16 bit picture. For the sinwave data our proposed compression does very little to the amount of transitions, but SILENT does decrease the transitions to 70% and 86% for 10 and 16 bit precision respectively. For the 10 bit random values SILENT does not perform well, which is to be expected as if the data is random and the difference between successive data will also be random too and since SILENT works by encoding the difference their will be little or no transition reductions. The random 16 bit data shows some interesting properties in that the number of transitions slightly increases in all cases. It is believed that because the random data cannot be compressed the extra transitions in the overhead will cause the number of transitions to increase. The proposed compression does reduce transitions by nature of the fact that the number of bits are being reduced. However, if further transition reduction is required then is it shown that the technique can be combined with SILENT to reduce the transitions further.

To give an idea of the cost of the proposed compression in terms of power and area, the transmitter and receiver (for fixed block size) net lists generated from the RTL VHDL models where used to obtain gate count and used for the basis of a gate level power simulation.

To show how power could be saved within a NoC link a FIFO type buffer which was coded and used as a connection between the transmitter and receiver, Fig. 3-20, as much of the power in a switch is used by the buffers [138, 139]. In Xpipes [28] the buffers are distributed along the length of the wire in order rather than have the buffers in the switches themselves. This distributed buffers system allows the switches to be physically smaller and the buffering occurs along the NoC links. The buffers have the capacity to hold two flits each and this is the basis for comparison. To give a similar level of capacity the buffers used in the simulation examples consisted of a 32 entry 1 bit wide FIFO to allow up to two flits to be buffered if each flit is 16 bits. When compressed the flit size could be smaller than 16 bits, but there is no guarantee it will be so the maximum possible size of 16 bits must be taken into consideration.

In this architecture it is assumed that the flits will not be interleaved and that a packet of data will be allowed to transfer in a wormhole fashion. Interleaving could be introduced but would most likely require extra logic in the switches to monitor the size of the compressed flits and strip out the interleaved compressed flits as necessary to rebuild the separated packets. This would impact the complexity of the switches as the compressed flit size could change on a packet by packet basis and each interleaved packet could have different compressed flit sizes.

The transmitter and receiver was synthesised with Synplify-ASIC using ST 0.12µm CORE9GPLL library. The synthesised design was then used in conjunction with the picture data from the bike example of Fig. 3-17 and run through Synopsys Primepower to provide gate level power estimations for the design. The simulation time for the runs was the same for both implementations to allow a comparison of the average power.

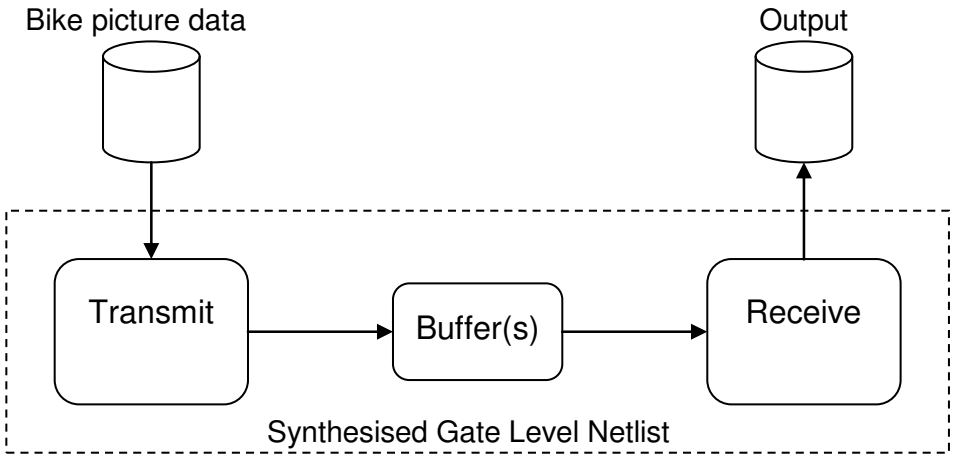


Fig. 3-20 Test bench and power simulation setup

As shown in Table 3-5, for the uncompressed implementation the area of the hardware is 97403 μm^2. For fixed block sizing the area is 107750 μm^2 effectively an increase of 10.6%. If we use fixed+SILENT in the implementation to further reduce the transition the area of the hardware is 111555 μm^2 which is a 14.5% increase in area.

Table 3-5 Area of design for standard and fixed block size of 64 (µm2)

	Transmitter	Receiver	Buffer	Total
Standard	91916	1596	3891	97403
Proposed Fixed	99488	4365	3897	107750
SILENT[103]	93519	4024	3877	101420
Proposed Fixed+SILENT	101068	6590	3897	111555

Table 3-6 shows the power for the uncompressed implementation and the fixed block size implementation. The two implementations show similar power usage when the data goes through a single switch, 0.3949 mW for the uncompressed and 0.4007 mW for the fixed block sizing, at first there seems to be no gain in using the compression since the power increase in the transmitter section is slightly more than the power saved in the buffer. However, this power is for a single buffer only. If the data has to go through more than one switch then the additional power saving of each additional buffer within each switch exceeds the power increase in the transmitter due to compression. For example in a NoC system if the data from the transmitting core has to pass through at least 3 switches to arrive at the receiving core then the power is reduced from 0.7031 mW down to 0.6332 mW, a power saving of around 10%.

Table 3-6 Power used when transferring the bike picture data example (mW)

1 Buffer	**Transmitter**	**Receiver**	**Buffer**	**Total**
Standard	0.1813	0.0595	0.1541	0.3949
Fixed	0.2208	0.0635	0.1164	0.4007
2 Buffers				
Standard	0.1813	0.0595	0.3082	0.5490
Fixed	0.2208	0.0635	0.2328	0.5168
3 Buffers				
Standard	0.1813	0.0595	0.4623	0.7031
Fixed	0.2208	0.0635	0.3492	0.6332

3.5. Concluding Remarks

Recent research is indicating that NoC with serial interconnect provides benefits from power and area point of view. This chapter has presented an effective compression technique that can be employed with such NoC, improving the bandwidth bottleneck of bit-serial links. It has been shown that it is possible to compress data over a serial link to reduce the amount of data that needs to be transmitted. Fixed and dynamic block sizing of the data to which the compression is applied to has been considered. Fixed block sizing shows a slight advantage in terms of reducing the amount of data when the original uncompressed data has some regular pattern or localisation of data words over similar length as the block size. In the case of MPEG the smallest unit of picture data that is stored in memory is 64 words (8x8 matrix portion of a picture) so data will tend be similar in this 64 word portion of data. Fixed block sizing has shown to be effective for the MPEG picture information showing that it is possible to reduce the amount of data by a further 6.7% compared to the dynamic block sizing in the example test data (Fig. 3-18, arnie1_10bit). Dynamic block sizing tends to perform slightly better where the similar localised data is not grouped into regular lengths as is the case with MPEG picture data. General guidelines for determining a suitable fixed block length and an algorithm for dynamic block sizing has been developed.

The proposed technique exploits the fact that unused significant bits do not need to be transmitted. Furthermore, the technique offers transmission with less transition count leading to the potential of lower power. Experimental results have been provided to show the transition reductions. A possible implementation of the proposed compression technique has been outlined and the area overhead costs from synthesised results have been presented showing that a power saving of 10% can be achieved for the link with 3 buffers. This power saving is achieved at the cost of an increase in area of 10.6%.

Chapter 4. Asynchronous Serialized NoC Links

Chapter 3 considered compression for bit-serial transmission in order to reduce the amount of data sent across a NoC link. However, the compression scheme is only useful for situations where the most significant bits change infrequently as shown in the results section of Chapter 3. The bit-serial compression may not be suitable for other types of data and so the option to serialize the data into slices and not use a full bit-serial link is explored to allow a higher bandwidth than a fully bit-serial link but also offer reduced wiring area than a fully parallel link. This chapter also investigates the use of asynchronous techniques to build on serialization to allow for power reduction and simplify the clock distribution of the NoC interconnect.

Synchronous circuit design relies on a common global clock which is used to maintain timing and provide a mechanism to allow all signals to be sampled at a well defined timing interval defined by the clock period. Synchronous circuits are deterministic in their operation and as such are relatively easy to design compared to asynchronous which does not have a common timing reference. The majority of NoC architectures that have been proposed are synchronous [28, 47, 140]. Recently there has been studies of asynchronous NoC [90, 92, 141] which highlight some of the problems with synchronous NoC such as global clock power consumption, clock skew and electro-magnetic interference. Global clock power can be a significant part of the total power for SoC. Processors such as the Pentium Pro or Alpha the clock power can be 10-30% of the total chip power [142, 143]. The Intel 80 core teraflops processor in which the cores are arranged in a 2D mesh uses about 28% of the total power on communication [144].

Interconnect between the NoC switches has also received attention in the asynchronous domain. An asynchronous point-to-point link that can be used for communication has been investigated in [91]. This scheme uses clock pausing techniques to pass data from the synchronous to asynchronous domains and provides a meta-stable safe interface between synchronous and asynchronous domains. However, if number of input/output interfaces increase arbitration is needed so that each input/output interface is able to pause the clock, this could lead to an increased chance of the clock being paused as the number of interfaces grow which could lead to an increase in latency and starting and stopping of data transfers through the

interface. Also the scheme needs a carefully calibrated delay line in the consumer and producer interface in order to function.

Interconnect cost, in terms of the number of wires required between switches, could also be considerable in NoC architectures since each switch is effectively connected by a point-to-point link to a neighbouring switch. The high cost of parallel links has been shown in [99], especially when inter-wiring spacing, shielding and repeaters are considered. The number of point-to-point links between the switches of a NoC will grow as more cores are integrated into a system.

Serialized transmission for NoC application been demonstrated successfully in [103]. Leakage and dynamic power reduction is possible using serialization [126]. Leakage power is reduced because of the reduction in the number of repeaters and buffers due to the reduced amount of wires. Dynamic power can also be possibly reduced since a large parallel link has large wire to wire capacitance which would require larger drivers and repeater to obtain the same propagation delay in comparison to a serialized linked with reduced wire to wire capacitance. Shielding, if used, on parallel links will also add to the total capacitance. However, it is important to take into account that the serializer and de-serializer circuitry uses power which may be more than the additional power used to driver parallel links. As the length of the links increase it may be more favourable to use serialized links as the power increase caused by the serializer and de-serializer is offset by the saving of the drivers seeing a lighter capacitive load per wire. It has been suggested that for 65 nm that serial links are preferred for wire length greater than 2-4 mm when compared to wave pipelined and register pipelined parallel links [126]. It is important to note that many factors, such as wire spacing, repeater sizes, wire capacitance, data patterns and switching activity all have an effect on the power so there is no clear way of knowing if dynamic power can be reduced without taking these factors into account and analyzing the system as a whole.

This chapter proposes the application of serialization as a means of reducing the interconnect cost in NoC, leading to reduction of wire congestion around the switches and the possibility of reducing the spacing between cores if over-cell routing cannot be used. Furthermore, the work investigates feasibility and design requirements arising from the interfacing of routing units of a fully-synchronous NoC scheme with an underlying asynchronous serial physical link.

The motivation for the work is given in section 4.1. Section 4.2 given details of the asynchronous link. Section 4.3 shows how the proposed link can be modified for word level acknowledgement and Section 4.4 determines the upper bound throughput of the asynchronous link. Experimental results are given in Section 4.5 and a summary of the acknowledgement schemes in 4.6. Section 4.7 provides practical validation of the asynchronous link using FPGA technology. Concluding remarks are given in Section 4.8.

4.1. Motivation

To study the feasibility of how an asynchronous serialized link can fit into an existing synchronous NoC architecture, a typical synchronous point-to-point link with wire pipelining buffers along the length of the wire was used. This would be used in a synchronous NoC where the switches and the wire pipelining buffer are clocked together such as [28]. This allows for high throughput of data due to the pipelining and the use of existing synthesis tools to implement the design. The fundamental reasoning for buffered pipelined wires can be found in Appendix C. A single link is shown in Fig. 4-1, the two switches are connected together with a wire segmented by a series of synchronous two-slot buffers.

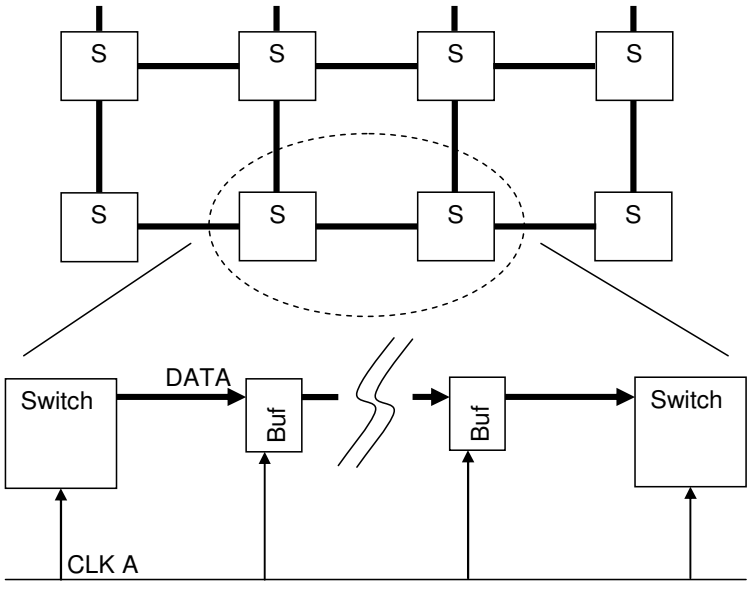


Fig. 4-1 NoC with Synchronous Link

To reduce the number of wires in such synchronous links the data can be serialised. Consider a simple serialization scheme as shown in Fig. 4-2, the number of wires required would reduce from the original amount *m* to the reduced amount *n*.

However, this would also mean that the 2nd clock (Clock B) driving the serializer, de-serializer and wire-buffers would have to be introduced. Clock B would have to be *m*/*n* times faster which could mean a 2nd clock tree spanning the chip area covering the NoC structure. Also, if no first-in first-out (FIFO) buffer or clock pausing mechanisms are used to pass data between the two clock domains the two clocks would have to be tightly phased locked to each other and clock B would have to be an integer value times faster than clock A in order that no timing violations occur when data or control signals pass between the two domains.

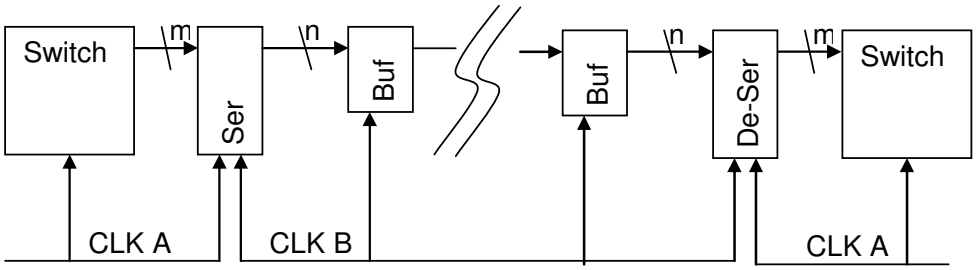


Fig. 4-2 Synchronous with Serialization Link

The introduction of asynchronous elements to the link would allow an architecture as shown in Fig. 4-3. The switch would interface directly to a synchronous to asynchronous interface and then go through a synchronous serializer. The benefit of this approach is that the data is serialized and thus saves wire area but also does not require a second higher speed clock to be fed into the serialization circuits and to the wire-pipeline buffers. The probability of the meta-stability at the interface between synchronous and asynchronous domain that may cause a synchronisation failure is significantly smaller in our case because of the relatively low frequency of Clock A, compared to frequency of the serialized link. The immediate drawback, however, is that extra overhead is introduced by the additional circuitry. This could impact area, power, latency and throughput.

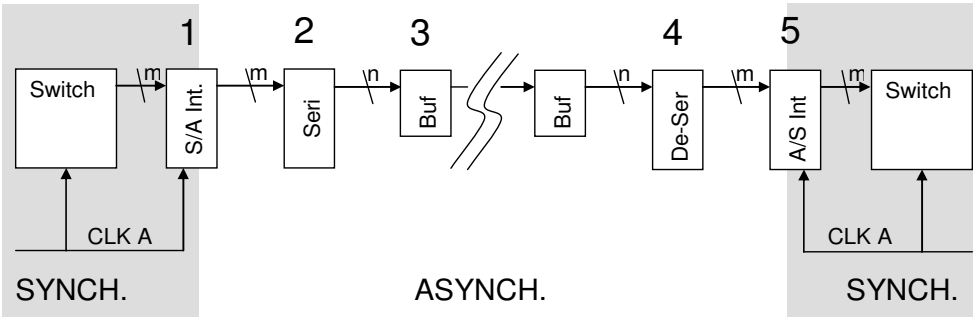


Fig. 4-3 Proposed Serialized Asynchronous Architecture

4.2. Asynchronous Link

The simplest asynchronous operation is perhaps bundled-data [145] where the data is sent with a request signal. A problem with bundled data is that mismatches in the delays of the data or request lines could cause meta-stability at the receiver. Techniques to improve bundled data such as surfing interconnect have been covered in [88]. Other schemes such as 1-of-4 coding [110], LEDR [111] and phase-encoding [146] encode the data in such a way that the receiver can recognize valid data, but usually require several wires per bit in order to function. Wave pipelining approaches are being proposed such as WAFT where a number of wave-fronts are present at one time on the data lines. The presented work shows a proof-of-concept implementation of an asynchronous link using a bundled-data link. The circuits are kept simple in order to provide a fair comparison to synchronous circuit. The remainder of this section describes the interface between the synchronous and the asynchronous domains and the physical implementation of the link. A more detailed overview of the proposed architecture is shown in Fig. 4-4. The architecture consists of a synchronous to asynchronous interface, a serializer, a wire buffer, a de-serializer and an asynchronous to synchronous interface. Circuits have been designed for the implementations of the synchronous to asynchronous interfaces and the serializer and de-serializer. The design of each of the modules will be described in detail in this section.

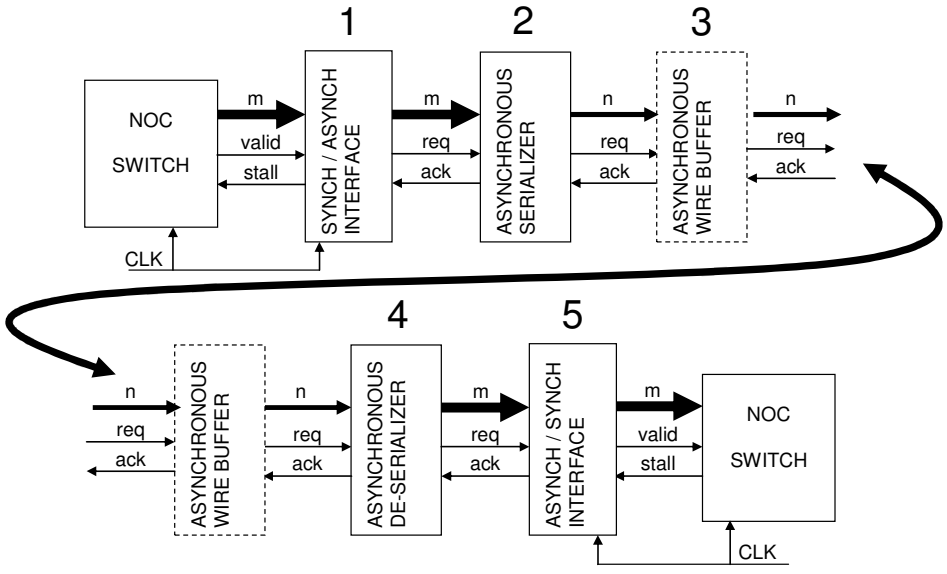


Fig. 4-4 Block Diagram Asynchronous Link

The asynchronous point-to-point link is implemented using standard logic cells and two common asynchronous cells, the C-Element [147] and the David Cell [148], which are shown in Fig. 4-5 and Fig. 4-7 respectively.

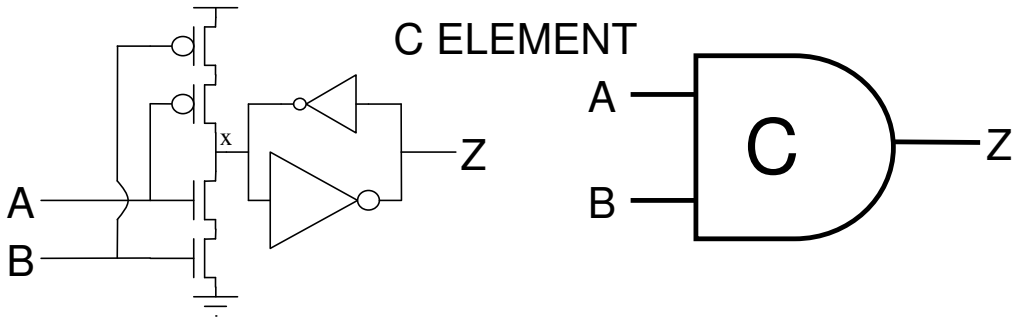


Fig. 4-5 C-Element

The C-Element is a component which has hysteresis where the output reflects the state of the inputs when states of the inputs all match. The output remains the same until all the inputs change to the opposite state. The C-Element consists of two invertors connected back to front in order to hold the output Z at a given state. The inputs A and B are connected to transistor which controls the other side of the holding invertors, point x in Fig. 4-5. If A and B are both low point x is pulled high and the output of the invertors Z goes low. If A and B are both high then point x is pulled low and the output Z goes high. If A is high and B is low, or vice versa, then point x remains unchanged as does the output Z. The C-Element can be used to establish that two events have happened in 4 phase handshaking asynchronous circuits, as the output Z will not trigger until both A and B have been set. The output Z will only reset once both A and B have reset following the 4-phase handshaking rule. An example of 4-phase and 2-phase handshaking is shown in Fig. 4-6 in the 4-phase example the request is set, the acknowledge is set, the request reset and the acknowledge reset. In the 2-phase example the request is toggled and the acknowledge is toggled in response.

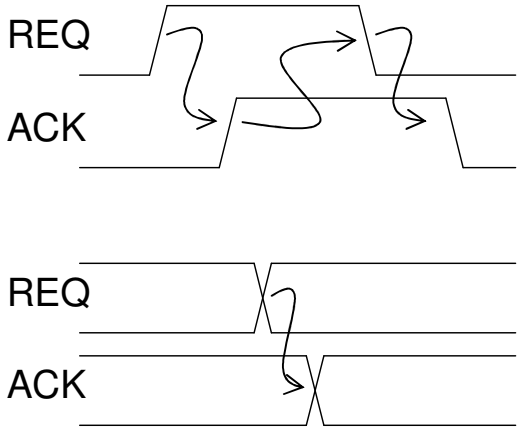


Fig. 4-6 4 Phase (top) and 2 Phase (bottom) Handshaking

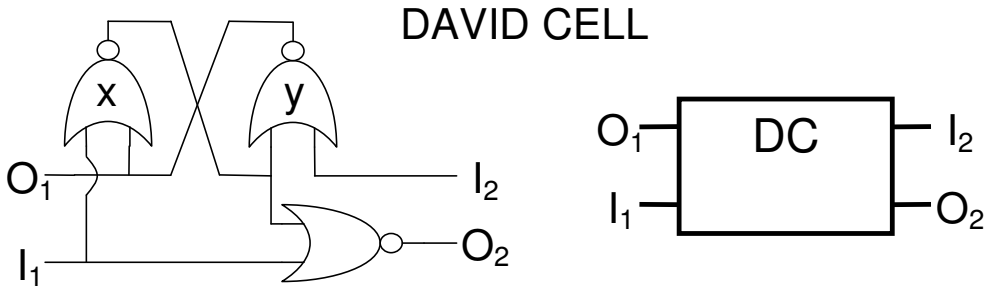


Fig. 4-7 David Cell

The David Cell consists of 3 NOR gates two of which are cross-coupled (x & y) and a third which acts as a gating mechanism for the output O2. The David-Cell can be used to sequence events or operations and can be chained together to form a 1-hot sequencing structure. The David-Cell is in a set condition if the output of gate x = 0 and y = 1, and a reset condition when x =1 and y = 0. Consider a series of David-Cells connect together as shown in Fig. 4-8. David Cell DC0 is set and DC1 and DC2 are reset. A and B are both some arbitrary circuits that perform a task when requested by the REQ signal and acknowledges completion of the task by the ACK signal. REQ to circuit A is currently high requesting that the circuit performs some operation. The sequence of event is as follows:

(a) Circuit A acknowledges is is finished by taking ACK to DC1 high.
(b) DC1 takes it output 01 high.
(c) REQ to circuit A goes low.
(d) ACK to DC1 goes low.
(e) REQ to circuit B goes high.

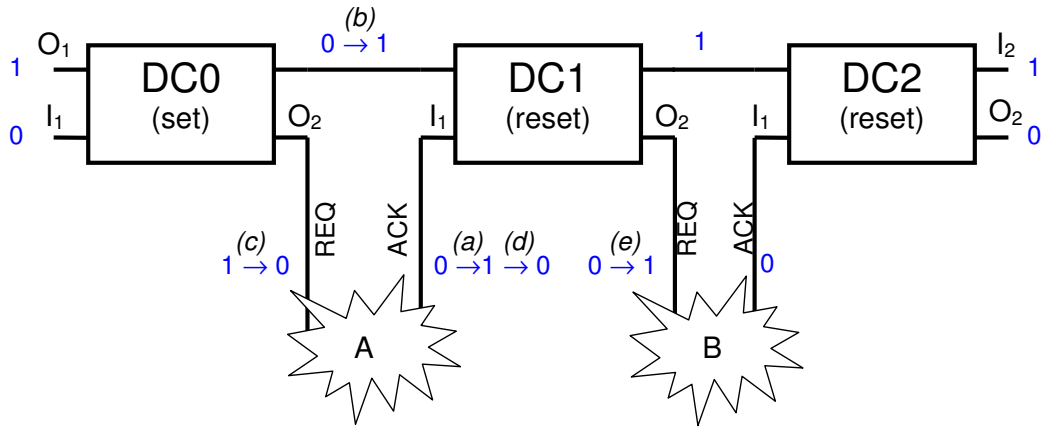


Fig. 4-8 Chain of David Cells

4.2.1 Synchronous to Asynchronous Interface

The synchronous to asynchronous interface is basically a FIFO type structure with a synchronous side that can write and an asynchronous side that can read. The FIFO can be considered 32 bit wide and 4 registers deep. A FIFO is used to effectively break the dependency of the of the asynchronous side from the synchronous side. The synchronous side can write to the FIFO at the same time the asynchronous side can read from it. A 4 deep a FIFO was used in the synchronous to asynchronous interface (marked 1 in Fig. 4-4) and asynchronous to synchronous interface (marked 5 in Fig. 4-4) to give a total of 8 possible spaces for data along the link, the same as the synchronous link. The synchronous to asynchronous interface (Fig. 4-9) can be considered in two halves, the synchronous register writer side and the asynchronous register reader side.

The synchronous register writer is comprised of four registers which can be synchronously written to when the appropriate WR_EN(x) signal is active. For each register there is an associated flag. The flag consists of a clocked D-Type flip flop with data enable. The input of D-Type is attached to VDD so that when WR_EN(x) is high a '1' is clocked onto the output of the D-Type. The output of the D-Type is the asynchronous flag FLAG_A(x). This is also fed through two registers flip flops to give the synchronous flag FLAG_S(x). The use of two flip-flops to build a synchronizer out of standard logic components is known to ensure sufficient level of protection against synchronization failures due to meta-stability, more on that can be found in [149]. The flag can be asynchronously cleared by using CLEAR(x) which is gated with the asynchronous reset attached to the D-Type. The VALID and STALL

signal is used to determine if there is space for the data on FLITIN to be written into one of the registers.

The asynchronous register reader comprises of several David-Cells and C-Elements. At reset DC(0) output O_2 is logic '1' and DC(1-3) output O_2 is logic '0'. The chain of David-Cells effectively form a 1-hot sequencer where one of them is always active. The C-Elements control the request and acknowledge handshaking and trigger the David-Cells in sequence. The multiplexer selects which of the registers will be output to the next stage, the output O_2 of the David-Cells control the multiplexer.

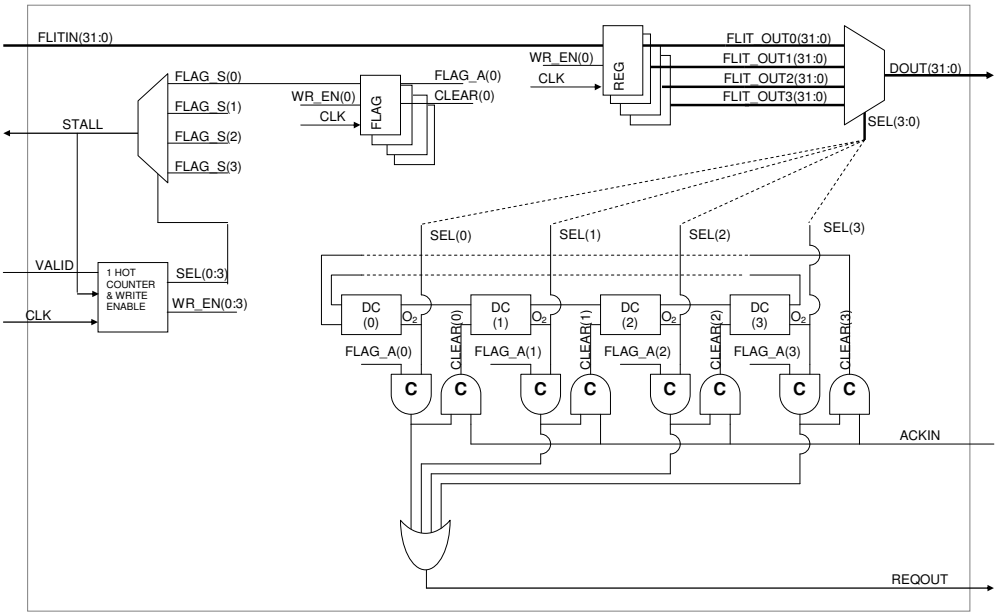


Fig. 4-9 Synchronous to Asynchronous Interface

4.2.2 Asynchronous Serializer

The asynchronous serializer, Fig. 4-10, consists of several David-Cells which select each 8 bit slice of the 32 bit data word in turn. At reset the output O_2 of DC(0) is logic '1' and output O_2 of DC(1-3) are logic '0'. The REQIN signal gated with SEL(0) triggers the start of the REQOUT / ACKIN sequence which is performed 4 times, each time the next 8 bit slice of the 32 bit data word is selected and latched at the output. The circuit can easily be modified to serialize more and break the 32 bit word in smaller slices by increasing the number of David-Cells and making the data path DOUT narrower.

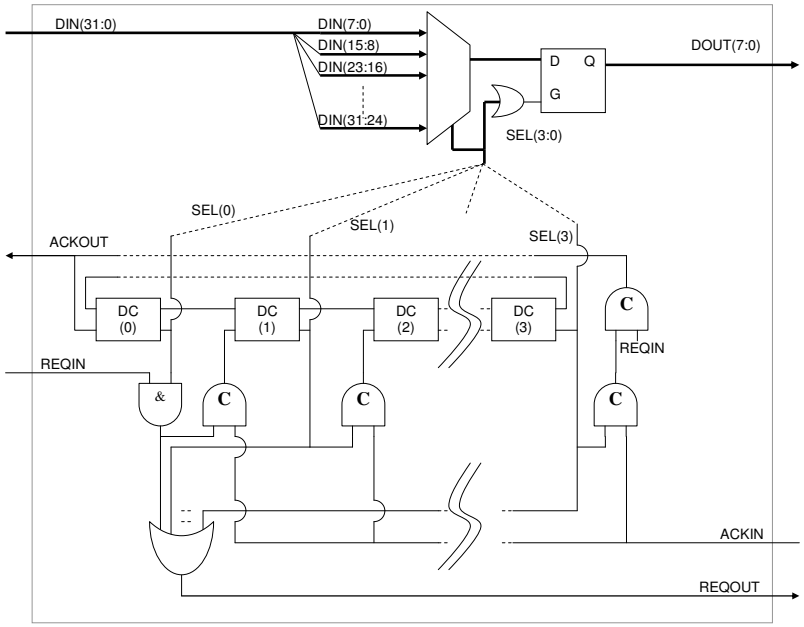


Fig. 4-10 Asynchronous 32 to 8 Bit Data Serializer

4.2.3 Asynchronous Wire-Buffer

The asynchronous wire buffer, Fig. 4-11 is based on a simple four phase latch control circuit [150]. It essentially latches the data on the falling edge of REQIN. The C-Element regulates the request and acknowledge handshaking safely. One point to note about this circuit is that the REQIN/ACKOUT side is not fully de-coupled from REQOUT/ACKIN side. If several of the wire-buffers are chained together then at best only every other buffer in the chain will be in use at a time. This does not present a problem in our case as the wire-buffering is a mechanism for transporting data rather than storage.

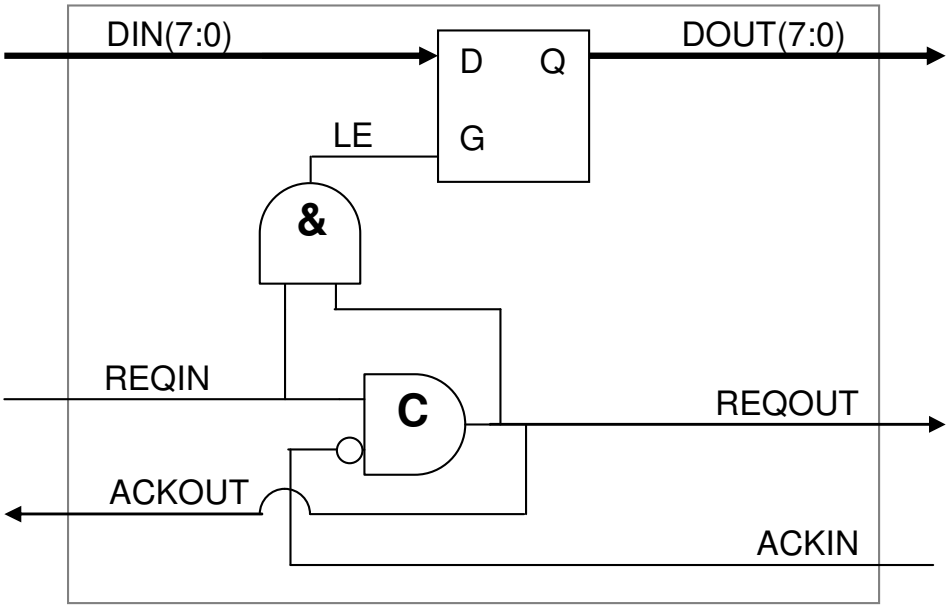


Fig. 4-11 Asynchronous Wire Buffer

4.2.4 Asynchronous De-Serializer

The asynchronous de-serializer shown in Fig. 4-12 takes 4 slices of 8 bits and re-constructs the original 32 bit data. At reset the output O2 of DC(0) is logic '1'. REQIN will go high signifying the first 8 bit slice is valid on DIN. The output of the C-Element LE(0) will then trigger and go high and latch the 8 bit slice into place. The REQIN/ACKOUT cycle is repeated 4 times until the 32 bit word is re-built and then the REQOUT is taken high to signify to the next stage the valid 32 bit data is ready. Again, like the serializer, the circuit can easily be altered for larger slice widths by reducing the number of David-Cells in the chain and altering the data path width.

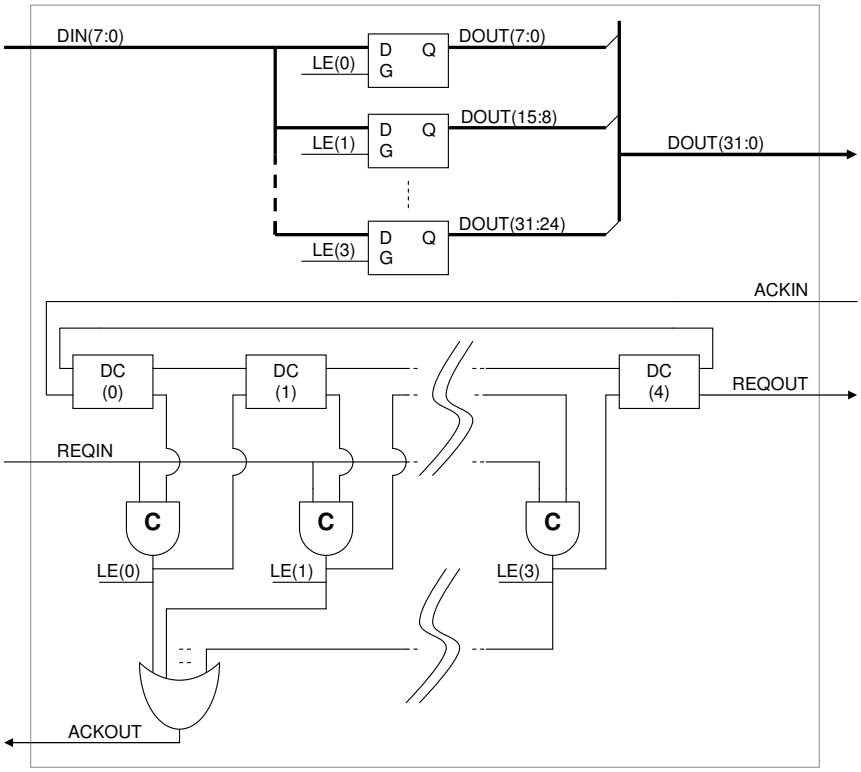


Fig. 4-12 Asynchronous 8 to 32 Bit Data De-Serialiser

4.2.5 Asynchronous to Synchronous Interface

The asynchronous to synchronous interface is also a FIFO type structure, with four latches. The design is very similar to the synchronous to asynchronous interface and again can be considered in two parts, the asynchronous latch writer and the synchronous latch reader. Like the asynchronous register reader, the asynchronous latch writer, Fig. 4-13 bottom, consists of 4 David-Cells and several C-Elements. Four of the C-Elements in this design are asymmetric, denoted by the '+' sign on one of the inputs, which means that the output will only be affected by this input going high and

ignored going low. Again, at reset, the output O_2 of DC(0) is logic '1' and the output O_2 of DC(1-3) is logic '0'. The request acknowledge sequence is triggered by REQIN. The output O_2 of DC(0) and REQIN is fed into a C-Element, which in turn is merged with inverse of FLAG_A(0). If the latch is empty FLAG_A(0) is '0' and the C-Element output LE(0) goes to a '1' and ACKOUT is asserted. REQIN will go low and FLAG_A(0) will get set to a '1' which will ripple through and take the input I_1 of DC(1) high. This will make DC(0) inactive and ripple through the two C-Elements taking LE(0) to '0' and taking ACKOUT low. The input I1 of DC(1) now goes to '0' and the output O_2 of DC(1) is now '1'. The sequence repeats for each consecutive request acknowledge handshake.

The synchronous latch reader is very similar to the synchronous register writer. There are four latches into which data is latched from the asynchronous latch writer. There are four flag modules which allow an asynchronous set of the FLAG_A(x) output and a synchronous clear. FLAG_S(x) is a synchronized version of FLAG_A(x) done by passing through two clocked flip-flops. A multiplexer allows one of the four data in the latches to be passed to the switch interface and a small controller controls the SEL(x), CLEAR(x) and VALID signals based on the FLAG_S(x) and STALL signals.

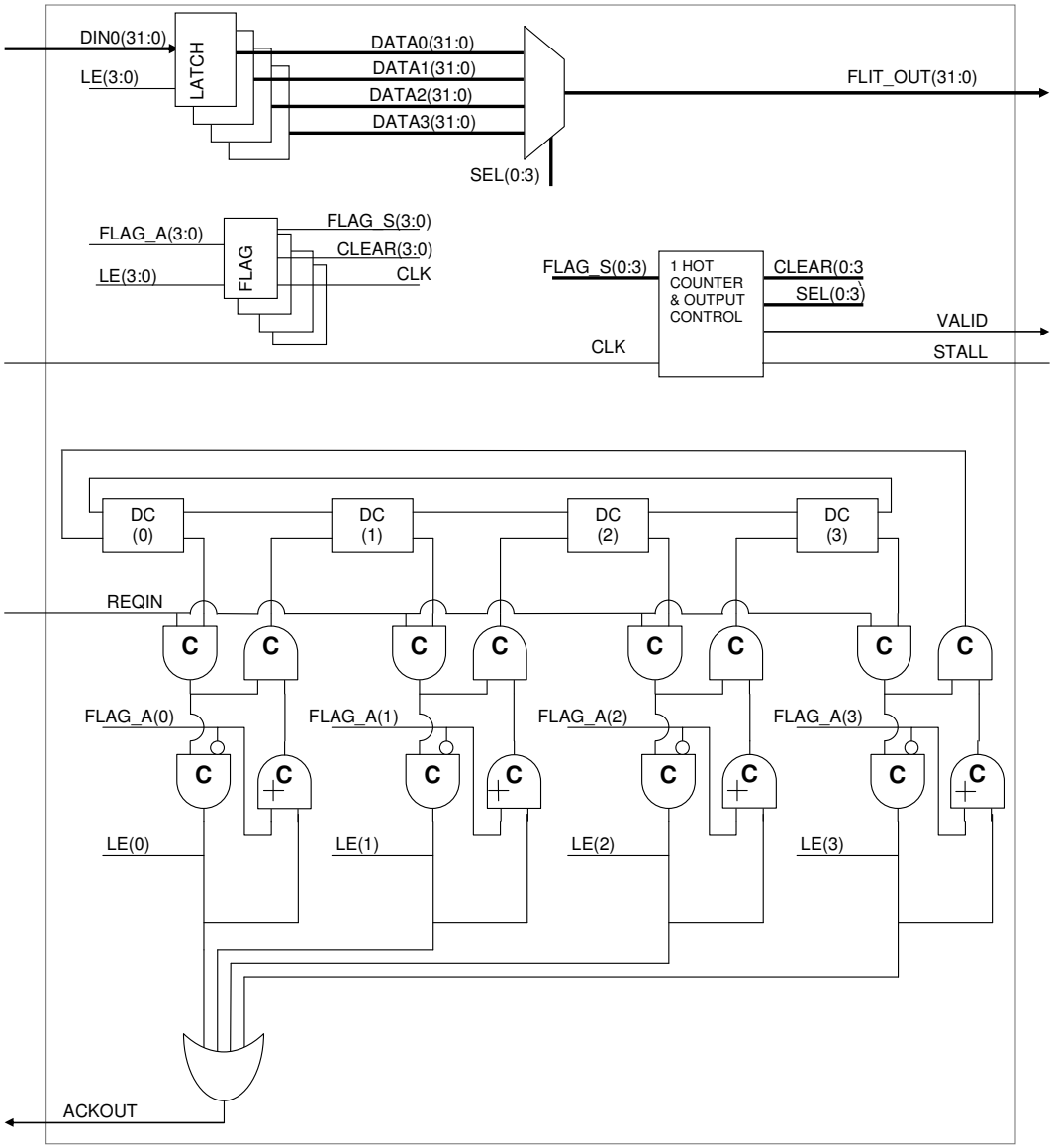


Fig. 4-13 Asynchronous to Synchronous Interface

4.3. Word Level Acknowledgement

One of the problems associated with a per-transfer acknowledgement is the need for the receiver or wire buffers to acknowledge every transfer. As the parallel data gets more and more serialised the number of request-acknowledge cycles per word increases. One possible way around this is to use a coarser grain acknowledgement that acknowledges at the word level, Fig. 4-14. Intuitively it can be seen that a coarser grain acknowledge at the very least removes 3 acknowledgements in our scheme. Removing 3 acknowledgements that would be required to send a full word will shorten the word transfer cycle time and increase throughput.

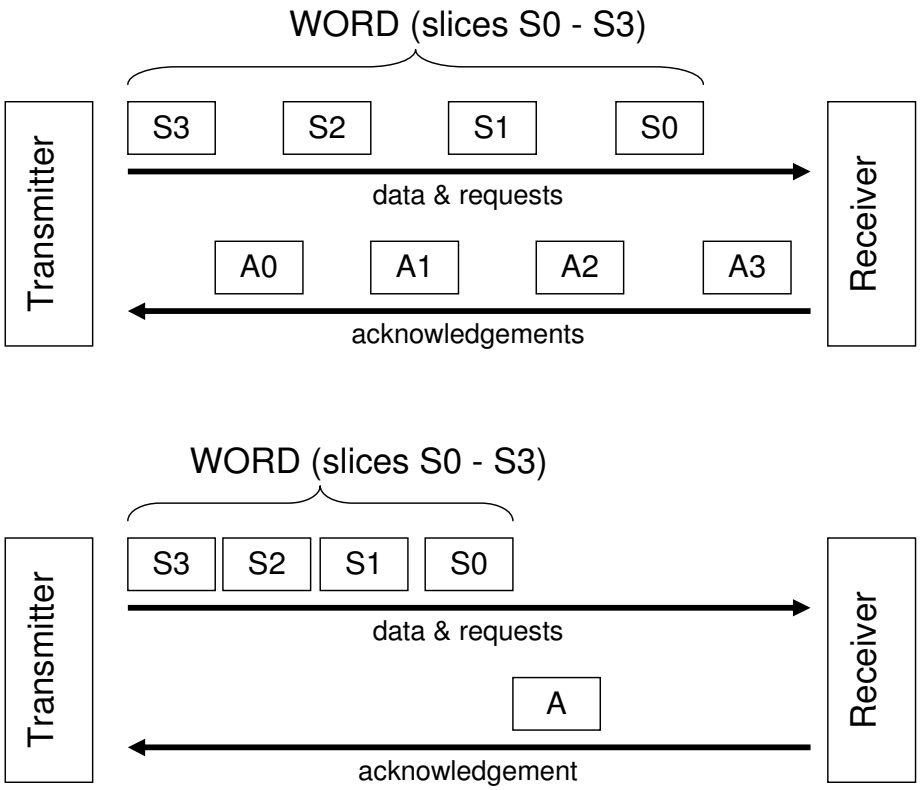


Fig. 4-14 Ack. every transfer(top) vs ack. every word (bottom)

Word level acknowledgement has some implications such as timing closure at the receiver which must be able to receive multiple transfers correctly and the need for some self regulated timing mechanism, such as a clock, at the transmitter to space the burst transfers out such that there is no timing violations incurred at the receive end. The proposed link (section 4.2) can be modified to use a per-word acknowledgement scheme by altering the serializer and de-serializer to perform several transfers per acknowledgement. Fig. 4-15 shows the proposed link with word level acknowledgement by modifying the serializer, de-serializer and wire buffer.

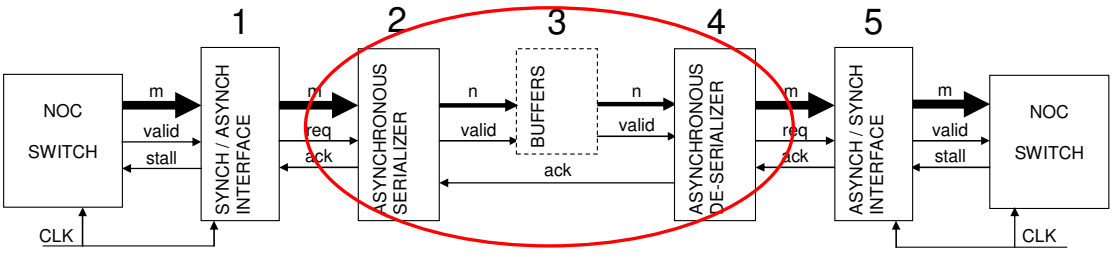


Fig. 4-15 Serial Asynchronous word-level acknowledgement

The buffers along the length of the wire can be replaced by simple buffers or an even number of invertors. The serializer (Fig. 4-16) uses a multiplexer with each slice of a word being selected in turn. The VALID signal goes high when there is valid data on

DOUT and signified to the receiver end that the data can be used. The VALID signal goes high 4 times, once for each slice of the word. The timing of the VALID signal is derived from the ring oscillator constructed by 5 back to back invertors. To adjust the frequency of the burst the number of invertors can be altered or different sizes can be used depending upon requirements. To ensure that VALID only goes high when the DATA is valid the respective timing between DATA and VALID can also be tuned by selecting different taps off the ring oscillator if necessary. Furthermore, if tolerance becomes problematic the VALID signal generation can be combined with the SELect signals to increase robustness.

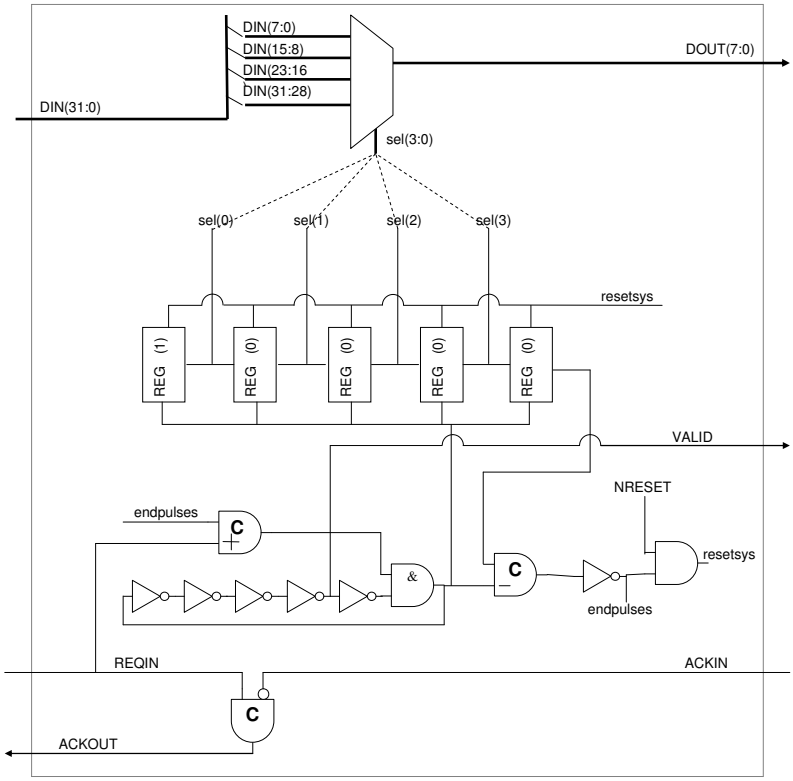


Fig. 4-16 Word level serializer

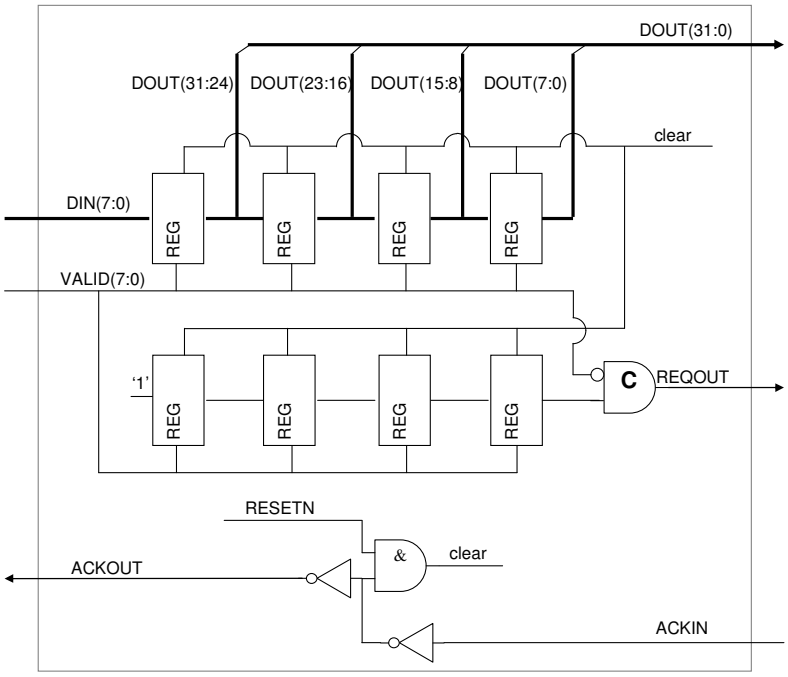


Fig. 4-17 Word level de-serializer

The de-serializer (Fig. 4-17) employs a shift register. This was done to see the effects of a shift register based de-serializer versus the original mux based de-serializer. The data is shifted in on DIN every time VALID goes high and the data slices are serially shifted onto DOUT. At the same time a single bit pulse is shifted down a single bit shift register of the same length to provide a REQOUT signal to the next asynchronous block to inform it the whole word has been built and is valid. ACKIN clears the single bit shift registers and removes REQOUT completing the handshake.

4.4. Calculation of Upper Bound Throughput

In order to give insight into the maximum throughput of the per-transfer and per-word scheme it is necessary to be able to calculate the upper bound rate for the asynchronous request/acknowledge handshaking. This can be done by examining the handshaking timing of the data through the link. To evaluate the accuracy of the per-transfer and per-word performance two equations have been developed which can be used to calculate the time taken to transfer a word across the link and therefore find the upper bound of the throughput.

4.4.1 Per Transfer Acknowledgement

For the per-transfer acknowledge scheme (Fig. 4-18) the following equations can be used to calculate the cycle delay.

$$Ser = \frac{SwitchDataWidth}{LinkDataWidth}$$

$D = Ser \times (4 \times Tp + Treqreq + Treqack + Tackack + Tackout) + Tnextflit$, where

- ***Ser*** is the serialization ratio
- ***Tp*** is the propagation time along the wires.
- ***Treqreq*** is the time of the request to write data into the buffer to the request to write the data out to the next buffer.
- ***Treqack*** is the time to request to write data into the buffer to the acknowledgment of the data.
- ***Tackack*** is the acknowledgement into the buffer to the acknowledgement out to the previous buffer
- ***Tackout*** is the acknowledgement into the buffer to the output of a new slice of data.
- ***Tnextflit*** is the time taken to get the next flit to be ready on the outputs of the transmitter.

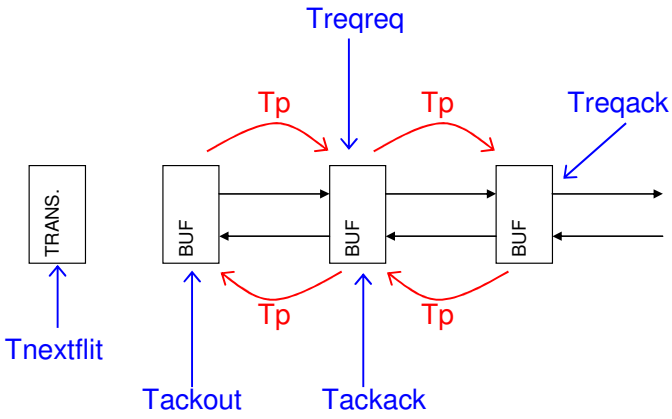


Fig. 4-18 Cycle Delay for the Per-transfer

4.4.2 Per Word Acknowledgement

For the per-word acknowledge scheme (Fig. 4-19) the cycle delay can be calculated using:

$D = Seg \times 2 \times Tp + (Seg - 1) \times Tinv + Tvalidwordack + Tackout + Tburst$, where:

- ***Seg*** is the number of wire segments.
- ***Tp*** is the wire propagation delay.
- ***Tinv*** is the inverter gate delay.
- ***Tvalidwordack*** is the delay from receiving a valid word to acknowledge output.
- ***Tackout*** is the acknowledge in to new flit output.
- ***Tburst*** is the burst period of the all the slices of the flit.

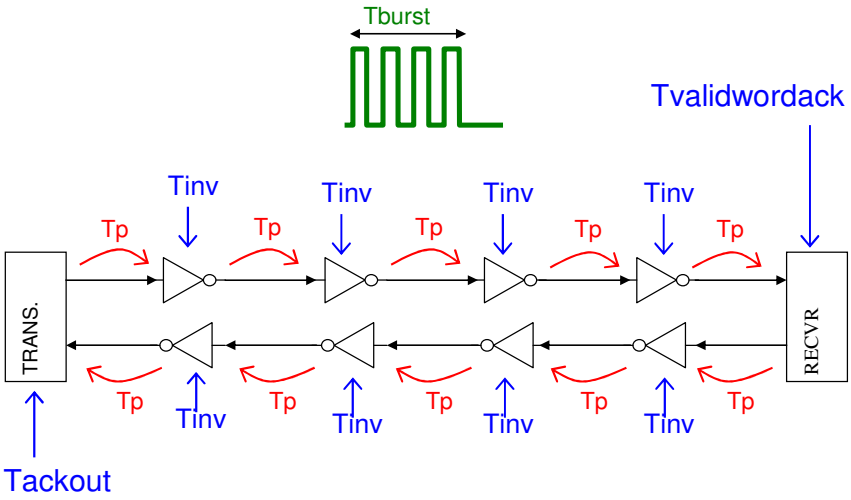


Fig. 4-19 Delay for per-transfer and per-word

The per-word equation can be checked using an example. Consider, Tp=0 since the simulation is gate level and does not take into account wire delays, Tinv=0.011 ns from the ST 0.12 CORE9GPLL datasheet, Tburst ~ 1.1 ns from simulation, Tvalidwordack ~ 0.7 ns and Tackout ~ 1.4 ns also from simulation and Seg = 5 since four wire buffers were used in simulation meaning the number of wire segments were five. Using these values the per-word delay is 3.21 ns from which we obtain an upper bound throughput of around 311 MFlits/s which matches with the supported bandwidths shown in Fig. 4-23 and Fig. 4-31 of the experimental section. It is important to note that the simulations do not incorporate wire delay information, but these can easily be introduced by using data from ITRS (International Technology Roadmap for Semiconductors) for the wire delay Tp.

With the equation we can now predict the upper bound throughput for different wire segment lengths. Using a global wiring metal pitch of 0.44 µm we can

look at the ITRS 2003 (Table 81a) and get an RC delay of approximately 50 ps for a 1 mm global wire at 0.44 µm pitch. A simple RC wire delay equation is:

$$T = \frac{RC}{2} L^2$$ where T is the delay, L is the wire length.

Using this equation we can find that RC = 0.0001 when L = 1 mm and T= 50 ps. Using this value of RC we can now put this back into the equation and see the effect of wire length versus wire delay for our global wire pitch of 0.44 µm, shown in Fig. 4-20.

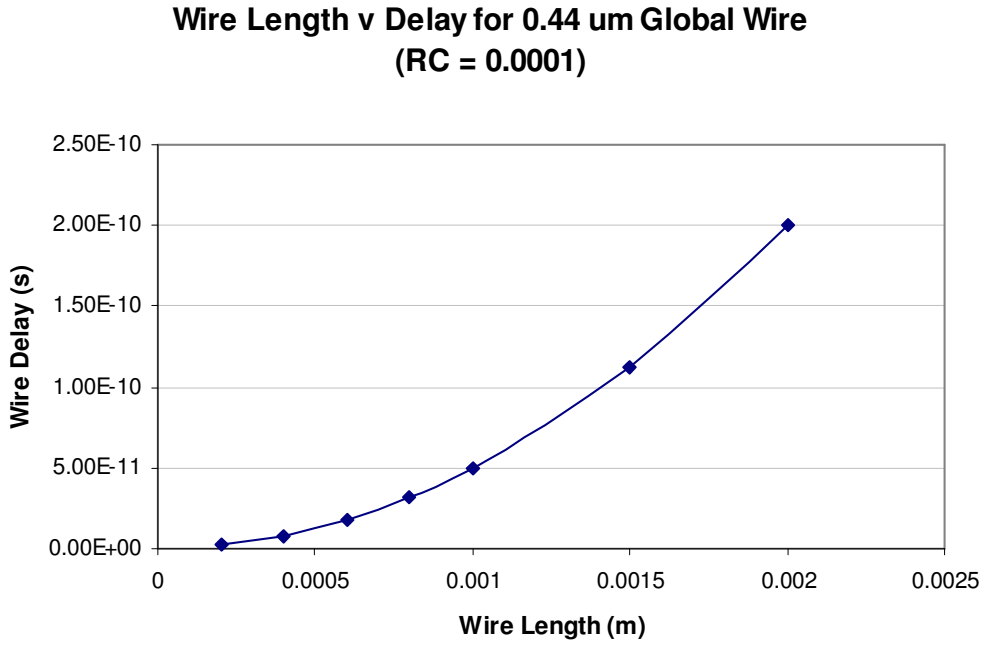


Fig. 4-20 Wire Delay for 0.44 um pitch global wire

The wire delay can further by used to calculate the upper bound throughput of the proposed per-word acknowledgement scheme using the delay cycle equation (2). Using our example of four buffers along the length of the wire the inverter and gate delays in the equation remain constant while Tp (the wire propagation delay) changes. The predicted upper bound throughput for a 0.44 µm pitch global wire using the per-word acknowledge scheme and four equally spaced buffers along the length is shown in Fig. 4-21. As can be seen the length of the wire has a square law effect on the wire delay, this is especially prominent on longer wire lengths. The upper bound throughput stays relatively flat at around 300 MFlits/s up to 2 mm length, then the throughput starts to fall off drastically.

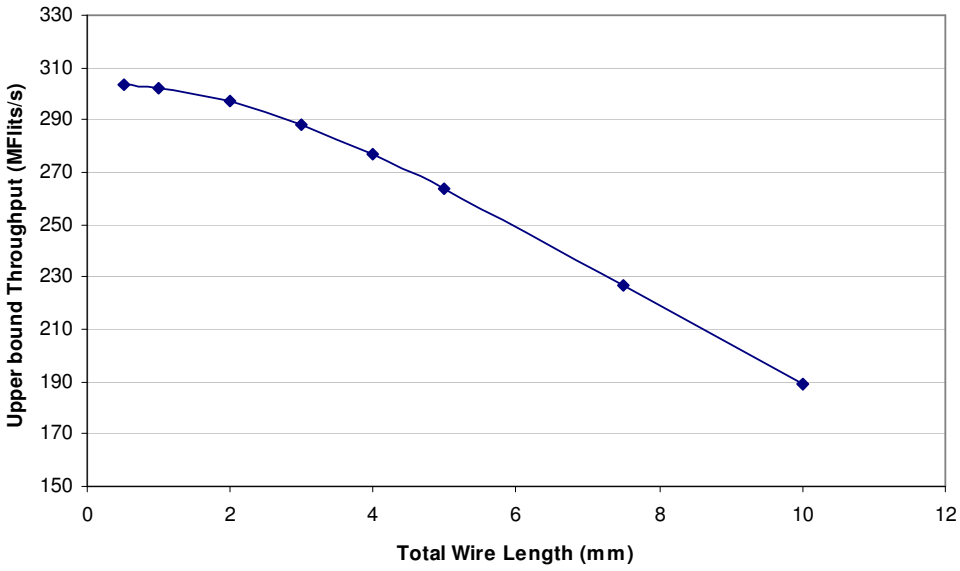


Fig. 4-21 Wire Length versus Throughput

4.5. Experimental Results

The simulations and comparisons are base lined from the XPIPES [28] NoC packet switched router and synchronous link which was obtained from University of Bologna. In XPIPES the input or output buffers are distributed along the length of wire rather than in the router itself. This distributed virtual buffer storage allows flits to use the buffers as storage and also to pipeline the wire. Each buffer can hold 2 flits, it is constructed from 2 registers each of which is 1 flit wide and some control logic which allows basic stall/go flow control of the flits down the length of the link. The synchronous links has 4 buffers along the length of the wire which means it can hold up to 8 flits. In the asynchronous serialised links the capacity is the same but the storage is in the synchronous to asynchronous interface which can hold 4 flits in the FIFO and in the asynchronous to synchronous interface which can hold another 4 flits in a FIFO, giving a total of 8 flits capacity, the same as the synchronous link. The reason the flits can be stored in the FIFO for the asynchronous link is because the FIFO is already exists in the synchronous/asynchronous interfaces and is used to allow the data to cross the synchronous/asynchronous domains. In the case of the per word asynchronous acknowledgement scheme the buffers along the length of the wire are non-registered and could not be used for storage anyway.

Circuits for the complete link (Fig. 4-9 to Fig. 4-13) were entered into the schematic editor in Cadence using gate level cells. The gate level cells use transistor

level models for analog level simulation. The results are based on three implementations that have been compared, Fig. 4-22. First a fully synchronous link with no serialisation and 32 bit wide data transfer (I1), second our proposed asynchronous per transfer acknowledge link which serialises the data down to 8 bits (I2) and thirdly our proposed asynchronous per-word link, also 8 bits (I3). Effects on power, area throughput are shown. The simulations were performed with foundry transistor level cells in ST 0.12 µm HCMOS9 technology with the Spectre simulator in Cadence.

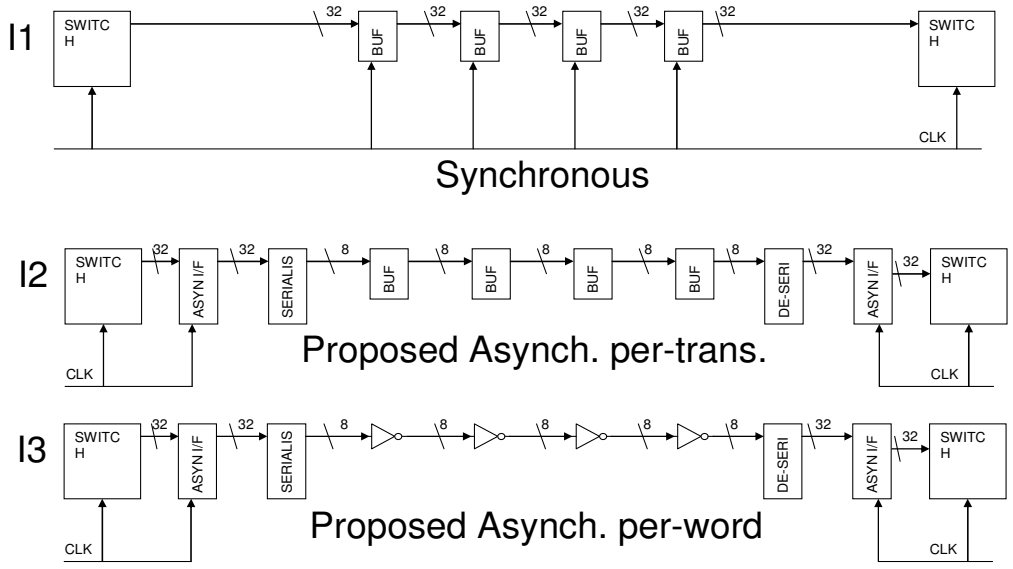


Fig. 4-22 Simulated Implementations

Fig. 4-23 shows the number of wires needed to achieve a certain bandwidth across a link. The synchronous link with 100, 200 and 300 MHz clock speeds are shown with the proposed link. As is seen the number of wires increase dramatically in the synchronous link as bandwidth increases. As the bandwidth required increases the number of wires for the synchronous link (I1) increases. This is because if the bandwidth requirement increases and the synchronous clock speed remains the same then the only way to increase the bandwidth is to make the data path wider by increasing the number of wires. Essentially the bandwidth is fixed by the number of data wires and the clock frequency if 1 bit of data is transferred per wire on each clock as follows:

$$Bandwidth = Wires \times Freq_{CLOCK}$$

The asynchronous link on the other hand is not governed by a synchronous clock, but is limited by an upper-bound throughput or cycle time of the asynchronous

circuitry. The upper-bound throughput is determined by the asynchronous handshaking cycle time. In this example of a 32 to 8 serialized asynchronous link the upper-bound throughput is ~ 300 MFlit/s, so the asynchronous link will operate correctly up to this point. For increased bandwidth the asynchronous link could halve the serialization going from 32 to 16 bits instead of 32 to 8 bits. This would effectively double the upper-bound throughput to ~600 MFlit/s at the expense of using 16 wires instead of 8 along the length of the link.

4.5.1 Area overhead

Fig. 4-23 shows that it is possible to achieve the same performance as the synchronous link but with less wires. For example, the proposed link (I3) can support 300 MFlits/s using a 300 MHz switch clock with 8 wires whereas the synchronous link (I1) would need 32 wires at 300 MHz which is a 75% reduction. It is interesting to note that the number of wires in the synchronous link would need to increase if the switch clock speed was reduced from 300 MHz to 100 MHz and maintain the same throughput, this would require an increase to 96 wires at 100 MHz.

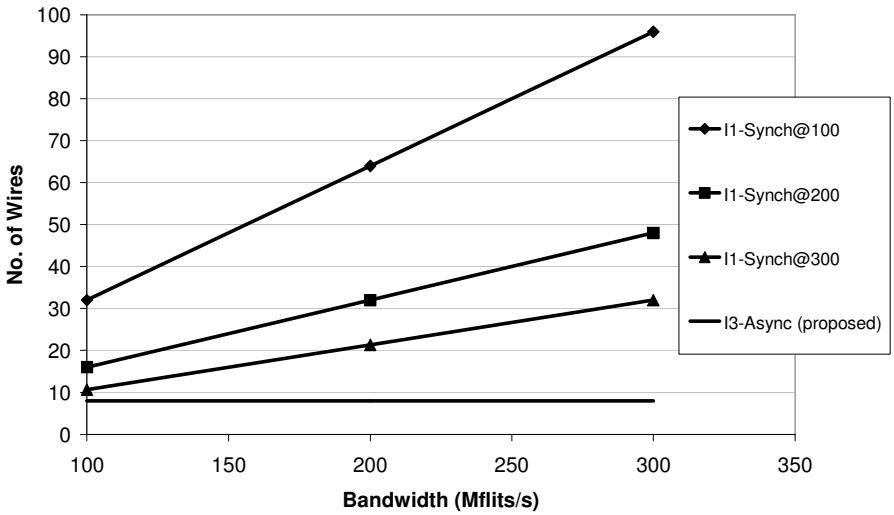


Fig. 4-23 Bandwidth vs. Wires

Fig. 4-24 shows the wire length and area for the implementations. It is important to note that a 1:1 trade off between area used by circuitry and area used by routing is not necessarily true as technology processes with many metal layers can route the interconnect over the top of the cells. The benefit of reducing the number of wires can clearly be seen, especially for longer wire lengths. For example, assuming a wire length of 1000 µm, implementation I3 has a wiring area cost of approximately

7,500 µm^2 whereas the synchronous implementation I1 is approximately 30,000 µm^2. As the wire length increases the proposed asynchronous link schemes (I2 and I3) have a moderate increase in area cost, unlike the synchronous implementation I1.

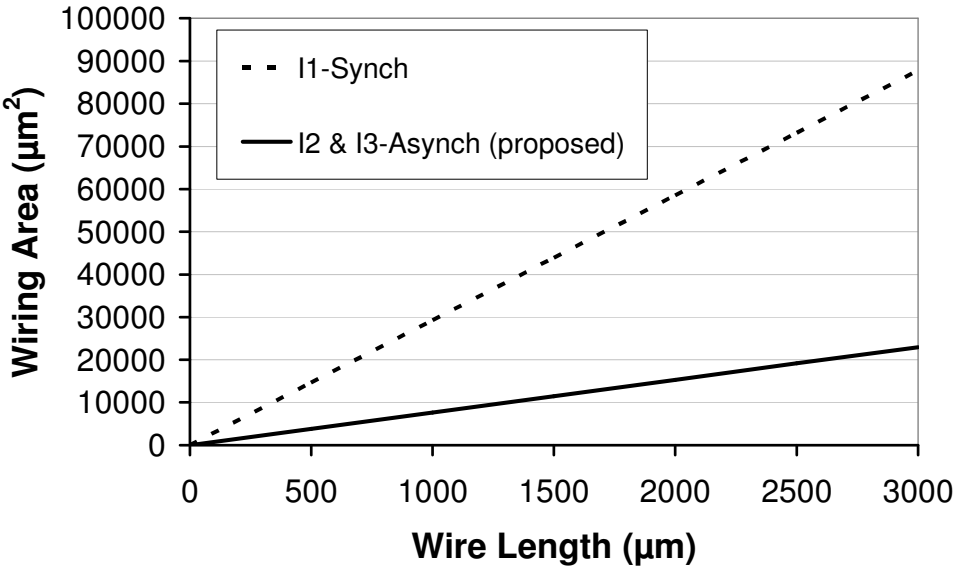


Fig. 4-24 Wire Area

For the ST 0.12 µm process the minimum metal width for a high layer (METAL6) which is typically used for global routing is 0.44 µm and the minimum gap between metal is 0.46 µm. The minimum wire area for the data path with N wires can be calculated by the following equation using *Mwidth* = 0.44 µm and *Mgap* = 0.46 µm.

$$METAL(\mu m) = N \times Mwidth$$
$$GAP(\mu m) = (N+1) \times Mgap$$
$$AREA_{DataWires} = Length \times (METAL + GAP)$$

Note that the equation is for a single layer of metal only. For multiple layers via and there associated design rule areas would have to be taken into account to provide a more accurate estimation.

The circuit area overhead of the synchronous and proposed asynchronous links are given in Table 4-1. To find out which portions of the asynchronous link use most resource a breakdown of the circuit or cell area used for each module for the implementations I1 through to I3 are shown in Table 4-2 to Table 4-4. The proposed architectures (I2 and I3) have an area increase of approximately 20% compared to the synchronous link (I1).

Table 4-1 Area overhead of the synchronous and proposed link

Implementation	Area (μm^2)
Synchronous (I1)	15864
Asynchronous per-transfer ack. (I2)	19193
Asynchronous per-word ack. (I3)	18396

Table 4-2 Breakdown of implementation I1

Module	Area (μm^2)	Qty.
32 Bit Synch Wire Buffer	3966	4
Total	15864	

Table 4-3 Breakdown of Implementation I2

Module	Area (μm^2)	Qty.
Synch to Asynch interface	9408	1
Asynch 32 to 8 serializer	869	1
Asynch 8 wire buffer	294	4
Asynch 8 to 32 de-serializer	1030	1
Asynch to Synch interface	6710	1
Total	19193	

Table 4-4 Breakdown of implementation I3

Module	Area (μm^2)	Qty.
Synch to Asynch interface	9408	1
Asynch 32 to 8 serializer	734	1
Asynch 8 wire buffer	61	4
Asynch 8 to 32 de-serializer	1301	1
Asynch to Synch interface	6710	1
Total	18396	

4.5.2 Power Consumption

The synchronous and asynchronous link implementation was compared in terms of power. The synchronous and asynchronous links each had 4 buffers along the length of the wire. On the asynchronous link the buffers were 8 bits wide and on the synchronous link 32 bits wide. All links had the same capacity to hold up to 8 flits. The average power was calculated for the transfer of 4 data items (0xAA55AA55, 0x55AA55AA, 0xA5A5A5A5, 0x5A5A5A5A) which exercises the data wires as much as possible and gives a high switching activity. The time the link is in use when transferring the 4 data items is approximately 70 ns on the original synchronous implementation running at 100 MHz. Using this base line transfer time of 70 ns the

simulation runs were set to 140 and 280 ns. This allows the average power for 50% and 25% usage to be obtained. The link can be considered 'in use' when one or more of the buffers is occupied by a flit. For example consider Fig. 4-25 showing flits F1 through to F4 occupying the buffers in order of arrival, the link usage time is basically the time when flit F1 enters the 1st buffer to the time flit F4 exits the 4th buffer. The same simulation run times of 140 and 280 ns was used for the asynchronous implementations in order to provide a fair comparison so that the average power can be seen to transfer the same data in the same period of time. The power for each block was obtained through Spectre simulations by taking the average of the supply voltage multiplied by the current over the simulation run time.

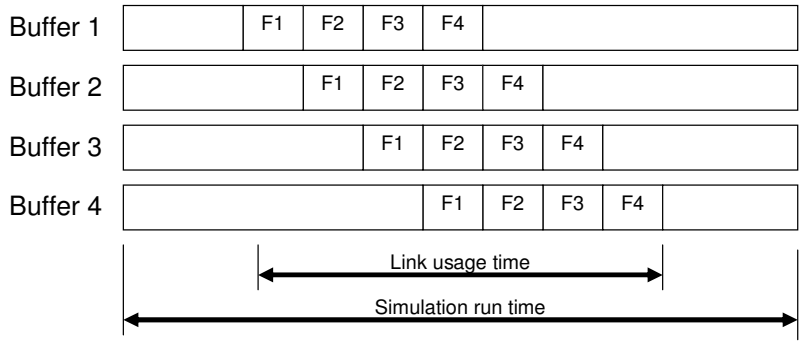


Fig. 4-25 Definition of Usage in our Simulations

The power consumption of the synchronous and the proposed asynchronous link are shown in Fig. 4-26 with switch clock speed of 100 MHz for different numbers of buffers in the link. As expected when a small number of buffers are used, such as 2, the synchronous implementation uses less power compared to the asynchronous due to the extra overhead of the synch/asynch converters and serializers. When the amount of buffers increase, the power in the synchronous implementation increases, unlike the asynchronous implementation which remains relatively similar. Comparing 2 buffers against 8 buffers for the wire link it can be seen the that power for the synchronous implementation (I1) increases 300% from 372 µW to 1498 µW which is expected since there is four times the number of synchronous buffers. The asynchronous per-transfer scheme (I2) shows a small power increase of 20% of the 589 µW to 712 µW, while the per-word acknowledgement scheme (I3) shows the least power increase of 2%, 623 µW to 637 µW, due to invertors being used along the length of the wire instead of latched buffer elements. Similar power consumption results can be obtained when the switch clock speed is increased to 300 MHz (Fig. 4-27). As expected the synchronous link power increases with clock frequency and it

can be seen that power increases from 1498 µW to 3229 µW for 8 buffers. The best power saving is obtained when the switch clock speed is 300 MHz and the number of buffers is 8, power is reduced by 65% from 3229 µW to 1110 µW when going from synchronous to asynchronous in this case.

Static power consumption was obtained by re-running the simulation with both stimulus and clocks being held low so no activity was present within the link. Static power consumption is the power consumed by the gates when no inputs are changing. The static power consumption is shown in Fig. 4-28. In the synchronous link the static power doubled from 23.7 µW to 47.4 µW going from 2 to 4 buffers and doubling again to 94.8 µW going from 4 to 8 buffers, this is to be expected as doubling the number of buffers which are the same is going to double the static power. The asynchronous implementations also increase in static power as the number of buffers increase, but at a much reduced rate since the buffers are more simple and in the case of the word-level acknowledgement are just inverters. For example going from 2 to 4 buffers in the per-transfer acknowledgement scheme increases the power from 50.15 µW to 50.97 µW and in the per-word scheme the static power increases from 66.83 µW to 67.14 µW. Each buffer in the per-transfer scheme uses 0.41 µW of static power and in the per-word scheme 0.17 µW or 0.14 µW depending on if the inverter is being held constantly high or constantly low due to differences in the size of the pmos and nmos transistors in the inverters.

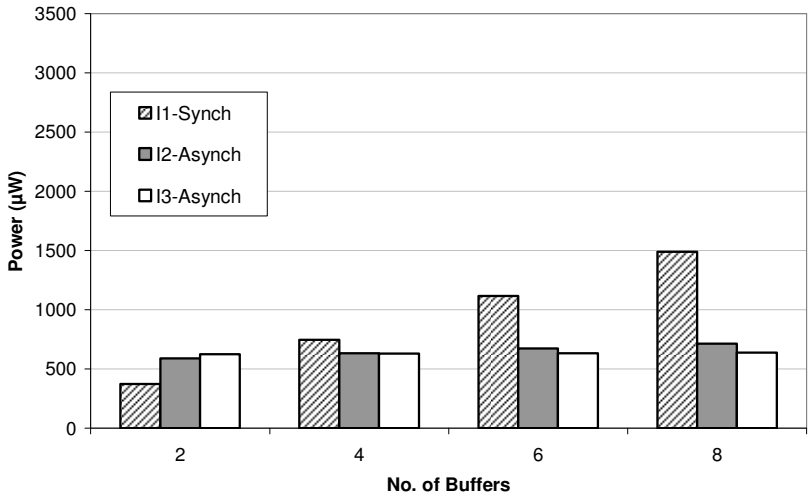


Fig. 4-26 Number of Buffers vs. Power @ 100 MHz

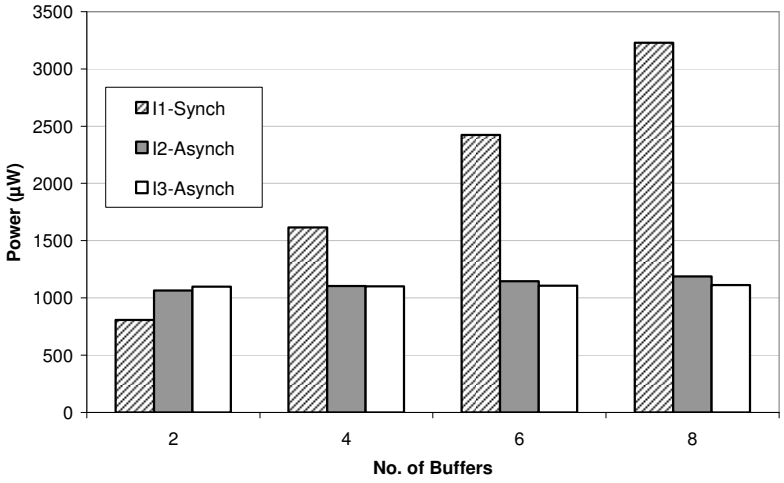


Fig. 4-27 Buffers v Power @ 300 MHz

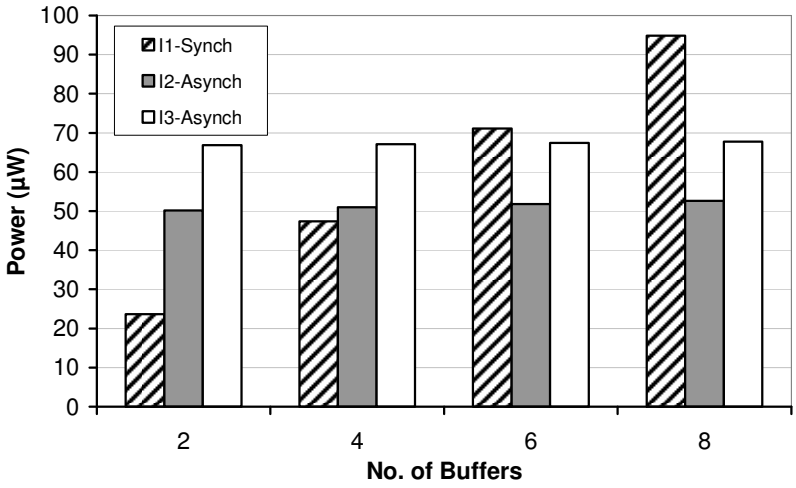


Fig. 4-28 Buffers versus Static power

To give insight as to where the power consumption is in the various components of the links, Fig. 4-29 shows a breakdown of the power when 1 or more buffers are occupied by flits 50% of the time (i.e. 50% usage). It can be seen that the dominant power in the asynchronous implementations (I2 and I3) are the asynch/synch and synch/asynch conversion circuits. This is expected since these circuits contain clocked synchronous parts. Comparing the proposed asynchronous links I2 and I3 which serializes down to 8 bits, it can be seen that that power used is similar. The I3 buffer power is considerably smaller than I2 at 9 µW versus 82 µW due to the fact that the buffers are simple invertors along the length of the wire and not latched elements as is the case for I2 and I3. The de-serializer uses more power I3 as a shift register based implementation is used instead of de-multiplexer, so all four registers are being

latched every time a slice of the flit arrives as opposed to just one register being latched in the de-multiplexer version.

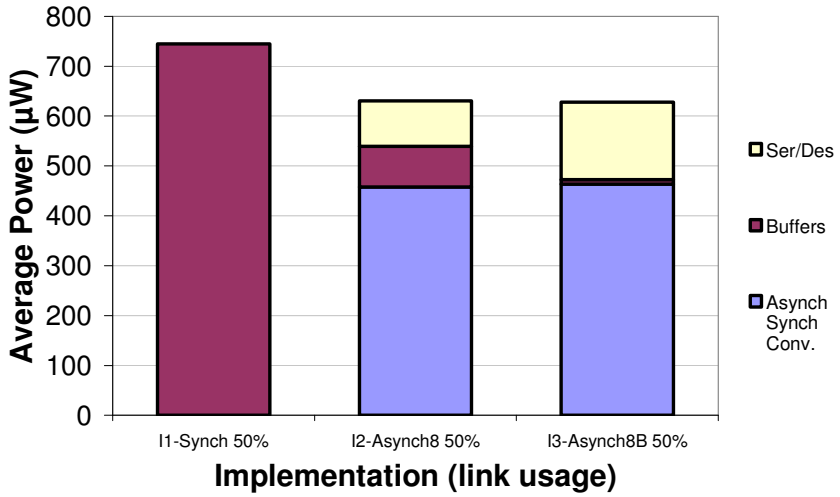


Fig. 4-29 Average Power for 50% usage

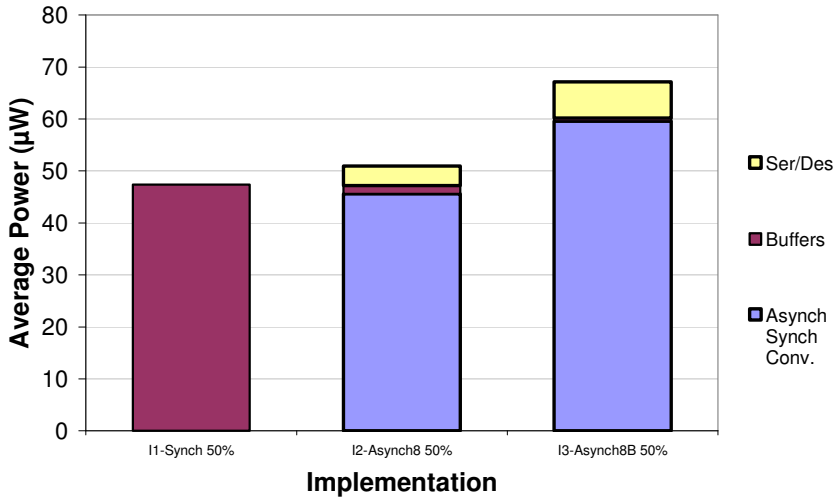


Fig. 4-30 Average Static power breakdown

4.5.3 Maximum Throughput

Fig. 4-31 shows the simulated maximum throughput of the three implementations, I1 the synchronous implementation, I2 the asynchronous per-transfer acknowledgement and I3 the per-word acknowledgement. This was achieved by simulating with increasing clock speeds until the throughput saturated. The back to back transfer time of a single flit was then measured to give the maximum throughput. As can be seen the per-word acknowledgement (~300 MFlits/s) has a 50% maximum throughput improvement over the per-transfer acknowledgement (~200 MFlit/s). The removal of the fine grain per-transfer acknowledgement and replacement with a coarser grained per-word acknowledgement has clearly improved the maximum throughput. The

synchronous implementation will not be limited by flow control acknowledgements but by the limits of the technology and will reach the maximum throughput limit when the setup and hold times start to be violated.

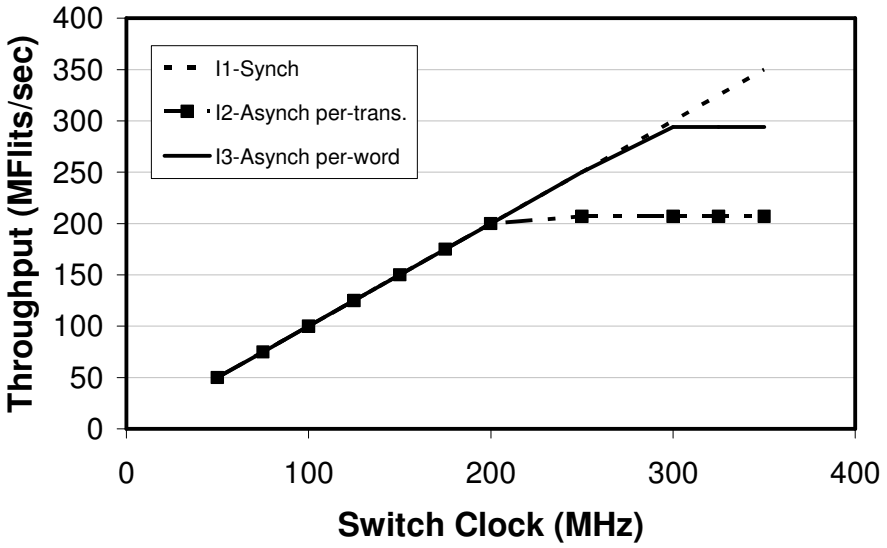


Fig. 4-31 Switch clock speed versus Throughput

4.5.4 Latency

The latency through the link, Fig. 4-32, can be considered at the time it takes the data to get from the synchronous to asynchronous interface on the transmit side to the asynchronous to synchronous interface on the receiver side (tAsynchpath) plus 2 to 3 clocks in order to resynchronize into the synchronous domain (tSynch). The reason for the 2 to 3 clocks to synchronize into the synchronous domain is because inside the asynchronous to synchronous interface the flag signals which signify data has arrived goes through two registers in order to synchronize the flag.

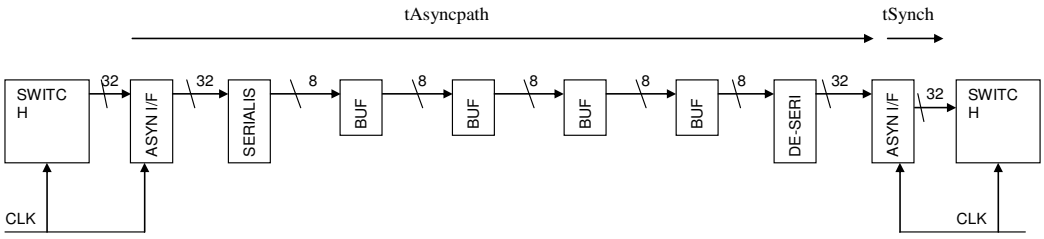


Fig. 4-32 Latency through the link

4.6. Summary of per-word and per-transfer schemes

Comparing the two techniques of per-transfer (Section 4.2) and per-word (Section 4.3) acknowledgement it is clear that the per-word acknowledgement offers higher

throughput due to the reduction of the request/acknowledge cycles needed to transfer a flit. Fig. 4-31 shows that the per-transfer throughput upper bound is around 200 MFlits/s whereas the per-word throughput is bound at around 300 MFlits/s, a 50% increase.

When considering area both techniques reduce the wiring area by 75%, but the circuit area of the per-word shown in Table 4-1 is 18396 μm^2 which is slightly smaller than the per-transfer scheme at 19193 μm^2. This mainly due to the buffers along the length of the wire being latched elements in the case of the per-transfer scheme and invertors in the per-word scheme.

The power use of the two techniques are very similar, Fig. 4-26 and Fig. 4-27. Observing the per-transfer scheme (I2) and the per-word scheme (I3) it can be seen that for 2 wire buffers the per-transfer scheme has a slight advantage in terms of lower power. However, as the number of buffers in the wire increase to 6 or more the advantage of lower power swings in favour to the per-word scheme. This is because the serializer and de-serializer in the per-word scheme uses more power but is offset against the wire buffers in the per-transfer scheme which are latched based. As the number of buffers increase, the power in the wire buffers in the per-transfer scheme overshadows the extra power used by the per-word serializer and de-serializer. Note that the per-word scheme wire buffers do not increase in power at the same rate due to the fact that they are simple invertors along the length of the wire and not latched based elements. The per-transfer acknowledgement scheme does ensure that every transfer is acknowledged, whereas the per-word scheme needs to ensure that the rate at which the slices of flits are transferred do not exceed the rate at which the receiver can consume them. In this respect the per-transfer scheme may be simpler as the transfers are regulated by the acknowledge signal so the receiver says when it is ready to receive the next slice of data.

For applications which can be satisfied with the lower per-transfer throughput and short wire lengths where only 2 buffers are used the per-transfer scheme will give the benefits of slightly less power usage. Also the per-transfer scheme would be used in more fault tolerant applications where the data is line coded and requires an acknowledge response for each valid data detected, such as dual-rail or m-of-n codes where an acknowledge is needed to signify that the data is valid and has been received correctly. This is because the per-transfer scheme acknowledges every transfer,

whereas the per-word scheme needs a burst of transfers before acknowledging, making the per-word scheme unsuitable for line-coded schemes. For applications which require a higher throughput that cannot be satisfied by the per-transfer scheme or long wire runs with many buffers that require lower power usage the per-word scheme would be desirable as this scheme has the power advantage when more and more buffers are used along the length of the wire.

In future technologies (sub 45 nm) if the tolerances of the on-chip transistors varies considerably the per-transfer scheme may be more desirable even though the throughput is lower. This is because with the per-word scheme the wire buffers are a series of invertors, each of which could have varying tolerances, so as the data and control signals propagate down the length of the wire the relative timing between them could drift apart to such an extent that improper latching of signals that are not yet valid could take place, Fig. 4-33. With the per-transfer scheme the data and control signals are effectively regenerated at each wire buffer due to the wire buffers being latched elements with request and acknowledge handshaking. Any drift between the data and control signals seen at the receiver will come from only the last buffer, whereas with the per-transfer scheme the drift would be accumulated from all the buffers along the wire.

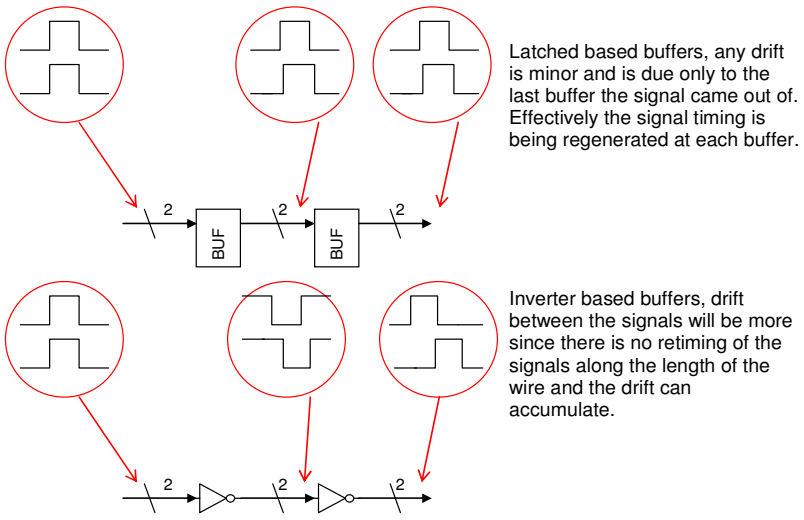


Fig. 4-33 Relative timing drift

The per-word scheme could be improved further by acknowledging the 1st slice of the flit rather than acknowledging the last slice of the flit, Fig. 4-34. Acknowledging the 1st slice of the flit reduces the dead time or waiting between back to back flit transfers as the acknowledgement circuitry always take a finite amount of

time to generate the acknowledge signals. Furthermore, a NACKing scheme could be used where the data is continuously transferred across when available and only stopped if the receiver buffer becomes too full, the receiver would have to send a stop or nack signal back to the transmitter to halt the transmission of data. This would require more buffering of data at the receiver end to ensure that no flits are lost or overwritten in the time it takes to stop the flow of data.

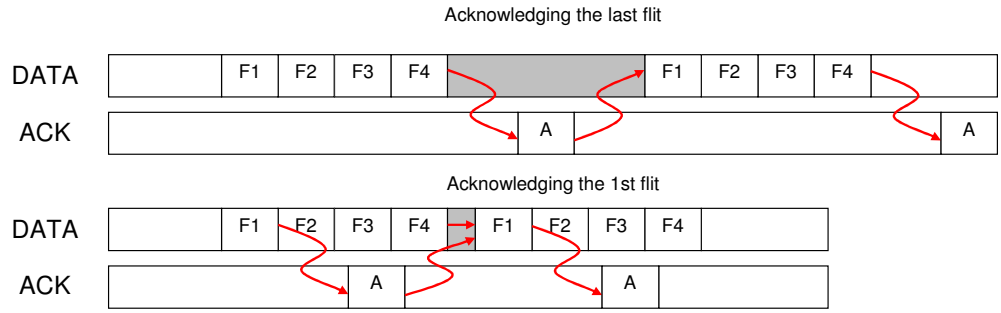


Fig. 4-34 First and Last flit acknowledgement

Serialization at the network interface [151] may give better results in power and latency as the serialization and de-serialization would only need to take place once at the network interfaces, rather than at each link. This would mean that the routers would have to operate faster at be at the same speed as the link. It is difficult to quantify what effect serialization at the network interface would do to our proposed link as the XPIPES router is treated as a black box and the link just interfaces to the outside of the router with no alteration to the router itself. To use serialization at the network interface a new router would have to be designed and implemented. If a new router was to be used it may even be beneficial to use an asynchronous router so that no conversion between synchronous and asynchronous domains would occur once inside the network.

4.7. Practical Validation of the Proposed Link

The previous section has shown simulation results based on transistor level models showing power, area and performance. The rest of this section discusses the implementation on FPGA. Section 4.7.1 shows the functional checking of the link.

To give an insight into how the proposed links operate in practice, practical implementation on hardware was considered. Two possible technologies for implementation of digital circuits are Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA). An ASIC is an integrated circuit implemented with standard cell logic gates. The cells are used from a pre-existing cell library that is supplied from the ASIC vendor. The ASIC has to be fabricated by the vendor once the design has been completed. An FPGA is a pre-fabricated silicon chip that has many small blocks of logic interconnected by many wires. Each logic block can be configured to perform logic functions and the wires can be used to selectively connect these logic blocks together to form a circuit. ASIC implementations allow for complete control of the placement of gates on silicon and should lead to faster speed as the logic and propagation delays can be kept to a minimum through optimization of the gate placements. However, ASICs require several months to design and fabricate, also if any errors are present the whole design and fabrication process may need to be repeated. FPGA implementations can be achieved in a short time since the synthesised design can be mapped to the FPGA structure automatically by place and route software. This rapid prototyping of designs also allows any errors in the design to be quickly corrected as well as trying out different circuit configurations. FPGA does allow functional validation but performance may be slower than ASIC implementations since the designer is constrained to use the pre-defined FPGA logic block which may not allow the optimum realisation of a circuit. In order to validate the link it was decided that FPGA implementation would offer rapid validation and low risk. While the FPGA implementation will not have the performance of an ASIC design it does give some confidence of the circuit working, albeit with much lower throughput.

As the link flow control is effectively governed by the handshaking which itself is controlled by C-Elements in the control logic it is reasonable to assume that the handshaking will occur in the correctly ordered sequence regardless of any delay

between the inputs of the C-Elements. There potentially could be delay between the bundled reference and the data which could cause the data to be latched at the wrong time, but this could be tuned out by tuning the relative delay between the reference signal and data signals so that the worst case and best case timing margins are covered. The functional testing of the link on FPGA should give reasonable confidence that the handshaking and the control logic is performing correctly but it is difficult to check that the data will remain aligned to the bundled reference signal over a large temperature range.

In order to validate the proposed link the circuits (Fig. 4-9 to Fig. 4-13) were coded in RTL VHDL, synthesised and targeted to a Xilinx Virtex FPGA. A Digilent XUP Virtex-II Pro Development System[1] was used as the target test platform. The development board consists of a XC2VP20 Virtex-II Pro FPGA with various connectors and peripherals attached on board. The FPGA was utilised along with a pair of connecters which were used to supply the logic analyzer with the output trace of the links. An Agilent 16000 series Logic analyzer was used to capture the output from the FPGA. All 32 data bits, valid signal and clock were routed to the connecter and the stall signal tied low to allow the logic analyzer to be able to capture the data free running. A simple walking '1' pattern was used as the stimulus to the link and was hard-coded as a small pattern generator in the synthesised code. The output of the link was connected to the I/Os of the FPGA which were routed to a connector on the test board to allow for capture by the logic analyzer. The code was partitioned for the synchronous and asynchronous link as shown in Fig. 4-35 and Fig. 4-36 respectively. The RTL code was contained in TOP.vhd was synthesized and tb_TOP.vhd was a test bench wrapper use for RTL and post place and route simulation. The DATA, VALID and STALL signals were mapped to FPGA I/O pins along with a copy of the clock and the reset signal mapped to a push button switch to allow reset of the circuit.

[1] More info obtained: http://www.digilentinc.com/Data/Products/XUPV2P/XUPV2P_User_Guide.pdf

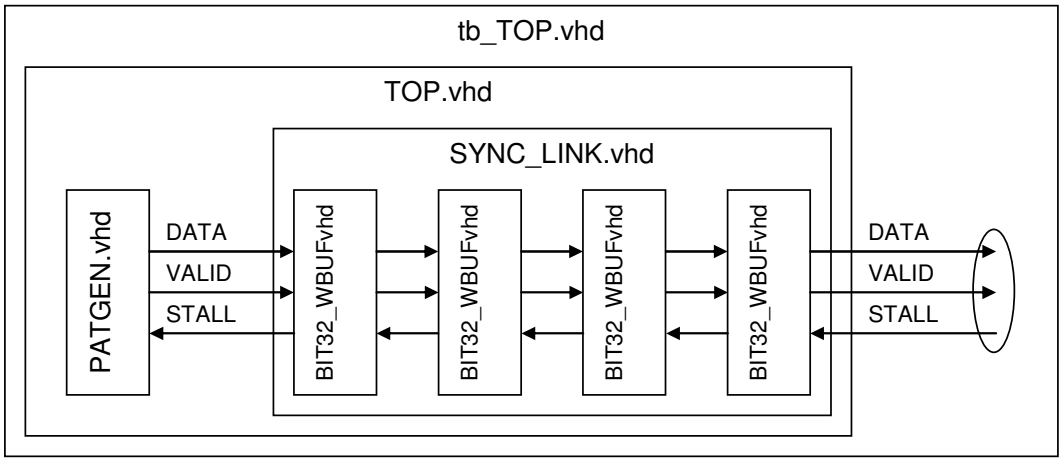


Fig. 4-35 Synchronous Link RTL & Test Bench

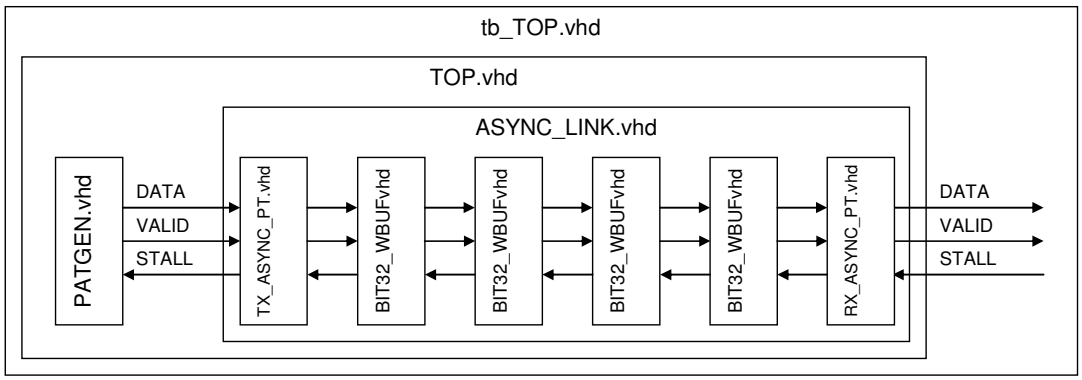


Fig. 4-36 Asynchronous Link TRL & Test bench

The wire buffers in the link were constrained to certain areas on the FPGA (Fig. 4-37a) in order to have a long wiring length between the buffers. The connectivity between the buffers and the I/O is shown in Fig. 4-37b where the wiring can be seen clearly going from BUF0 to BUF1 to BUF2 to BUF3 and finally to the I/O outputs of the FPGA. The constraints forced the place and router to put the logic for the wire buffers in their respective areas to try and emulate long wire lengths. It is unknown how long the distance between the buffers and this die size information of FPGAs is not generally available. Some discussion forums[1] on the internet suggest the die size for the XC2VP20 is 14mm × 11.4 mm which even allowing for I/O pads it would be reasonable to assume that the distance between the buffers from the left hand side to the right hand side of the chip is > 1 mm. The design flows for the synchronous and asynchronous implementations are shown in Appendix D.

[1] http://www.fpgarelated.com/usenet/fpga/show/36765-1.php

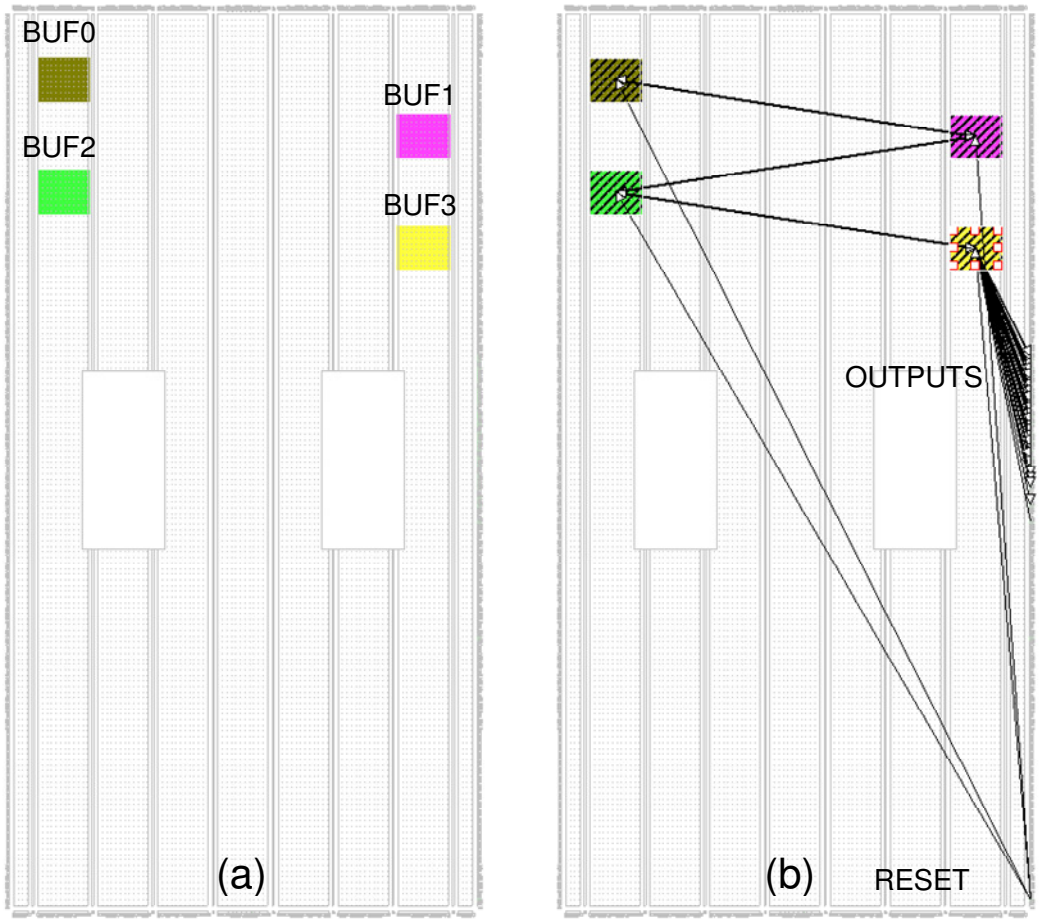


Fig. 4-37 Floorplan Constraints of FPGA

4.7.1 Functional Checking

The 32 bit DATA output and VALID signals were routed to I/O ports that were routed to a connecter on the development board. The logic analyzer probes were hooked up to the connecter and used to capture the waveforms. The captured waveform was checked to make sure that it was a walking '1' pattern to verify that the data transmitted along the link from the pattern generator logic was correct. Fig. 4-38 shows the captured waveform. Only bits 31 to 20 of the DATA bus has been shown for clarity, but it is clear that the walking '1' patterns is being received at the end of the link. One observation is that the DATA bits appear to have 2 transitions on them. This can be explained by considering the post-place and route simulation waveform shown in Fig. 4-39. In this simulation waveform the DATA output when VALID goes high is circled in a solid line, the 'old' values in that get observed at the output are circle in a dashed line. The reason a second high transition is seen on the DATA bits is because the synchronous side of the receiver is operating faster than the asynchronous link and the control logic in synchronous side of the receiver is

selecting a FIFO entry and multiplexing the contents out onto the DATA bus before the asynchronous side has written new data into it. This does not cause any error at the receiver end though as that VALID signal only goes high when the data is correct. This can be confirmed by observing the data capture as a list format shown in Fig. 4-40. It is shown that the DATA goes from 0x00000001, 0x00000002, 0x00000004 and so forth which is a walking one pattern in hexadecimal format.

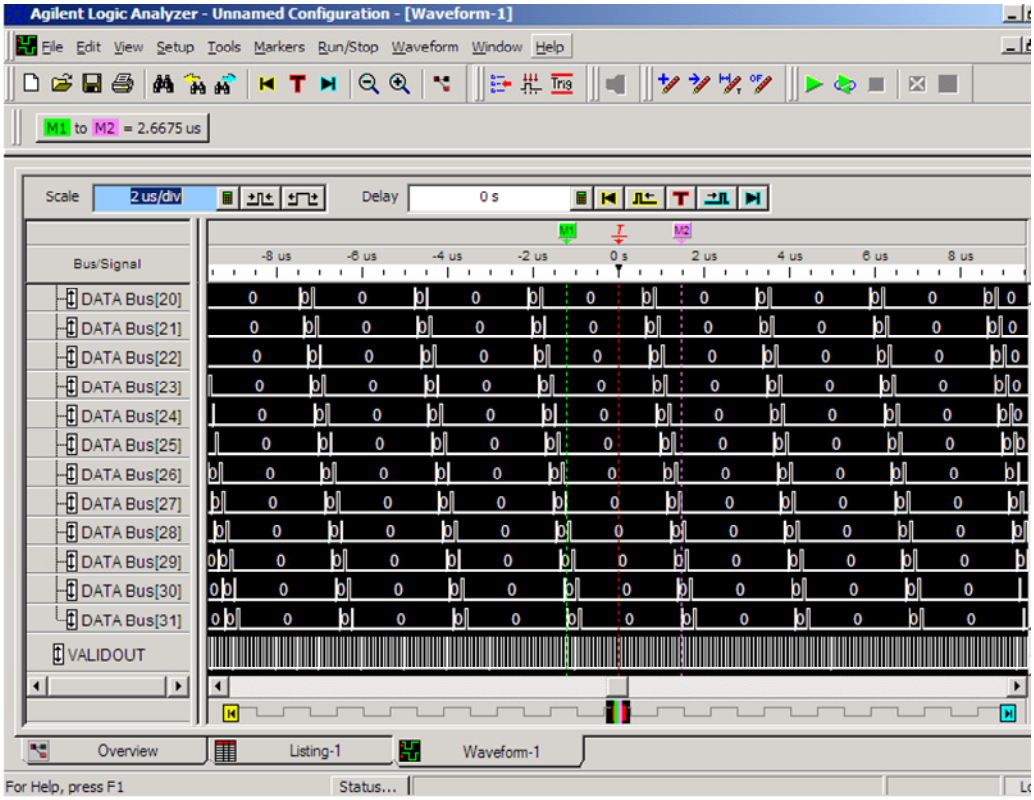


Fig. 4-38 Timing Capture of Asynchronous Per-Transfer Link

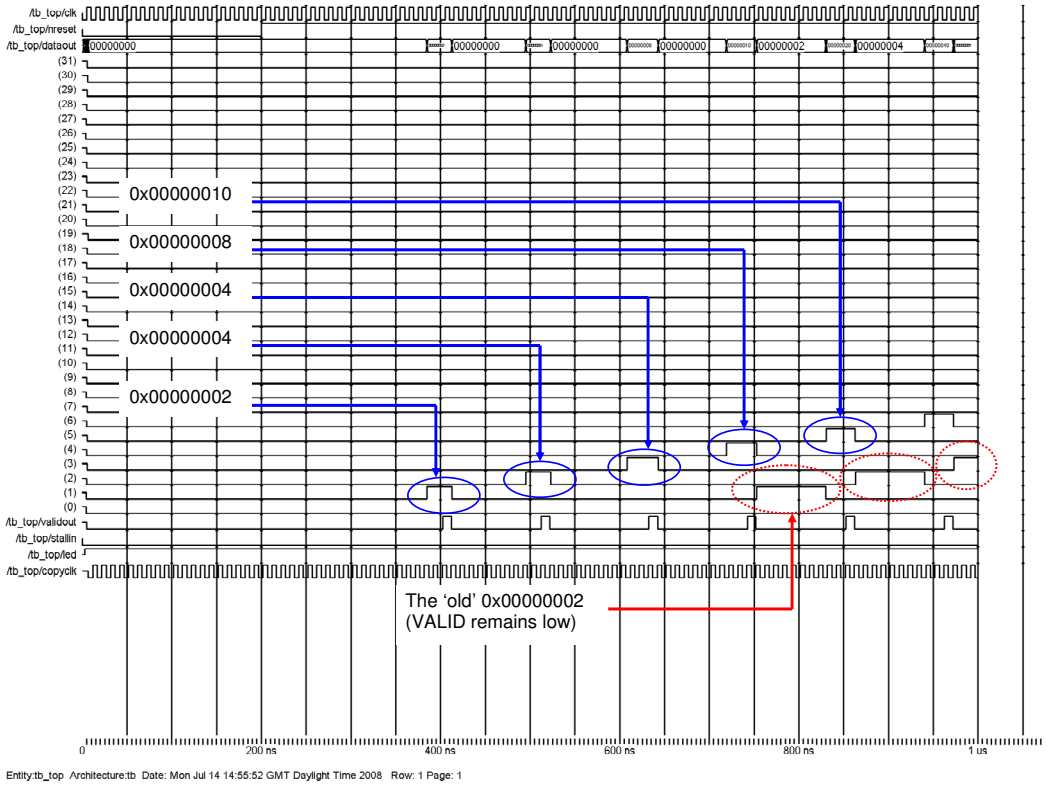


Fig. 4-39 PAR simulation of asynchronous Per Transfer Link

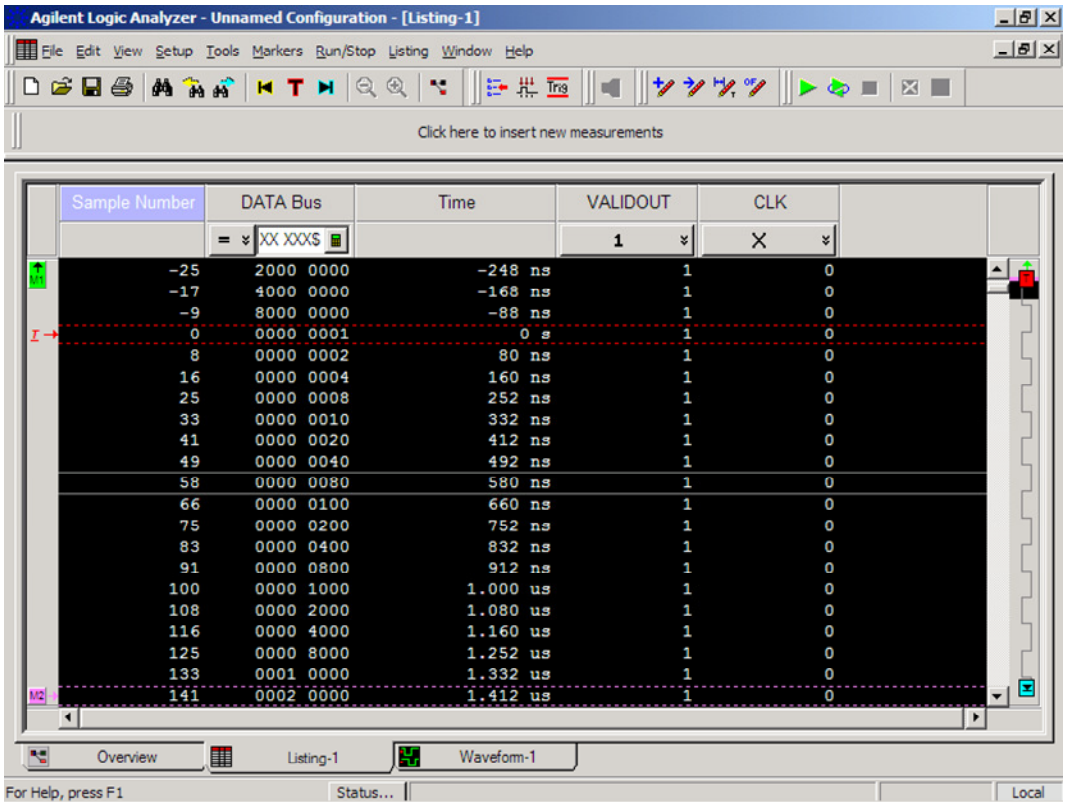


Fig. 4-40 State Listing of Asynchronous Link when VALIDOUT is high

4.8. Concluding Remarks

This chapter has proposed the use of asynchronous circuit techniques and serialization for NoC links. The proposed link uses asynchronous techniques which removes the need for global clocking along the link and also reduces power. The serialization of the data allows a reduction in the number of wires in the link and therefore a reduction in interconnect cost with respect to wiring area and the size of the wire buffers along the link. The proposed asynchronous link has been compared to a synchronous link in terms of area, power and throughput. The synchronous link is a fully synchronous 32 bit wide link, the asynchronous link goes from 32 bit wide at the start of the link to 8 bit wide along the length of the link and back to 32 bits at the end of the link, thus serializing the word into 4 slices.

The asynchronous link has demonstrated the effectiveness of serialization in reducing the number of wires without compromising the performance up to the throughput limitation caused by the asynchronous handshake cycle timing. The potential problems with synchronous design such as global clock distribution and clock skew have also been reduced. The proposed asynchronous link also reduces power by up to 65% compared to the synchronous link when 8 buffers are used. Furthermore, the area overheads of synchronous and the proposed asynchronous link have been compared and shown that although the proposed link has a 20% circuit overhead the number of wires has been reduced by up to 75%. The asynchronous link has been demonstrated with a per-transfer acknowledgement and per-word acknowledgement. The per-transfer acknowledgement scheme acknowledges every transfer of data across the NoC link and the throughput saturates at approximately 200 MFlits/s (Section 4.5.3) and is consequently slower than the per-word acknowledgement scheme which acknowledges every full word of data received and the throughput saturates at approximately 300 MFlits/s.

Functional validations of the proposed asynchronous link have been carried out using FPGA technology. The FPGA implementation confirmed correct operation but the performance was drastically reduced when compared to the cell based IC simulations. The speed of the link was 200 MHz for the cell based IC simulations and approximately 12 MHz for the FPGA.

Chapter 5. Resilient Asynchronous Links

Asynchronous links bring benefits such as simplifying the clocking and reducing power as shown in Chapter 4. The link implementation in the previous chapter used a bundle data approach where a reference signal was sent along with the data. Bundled data is less desirable as if the relationship between the reference signal and the data varies too much possible timing violations could occur. One way around this is to use delay insensitive data coding [152]. Delay Insensitive coding is used in asynchronous circuits to allow the receiver to validate the data regardless of relative timing on the wires. Delay insensitive coding could be susceptible to transient errors since a transition on a wire could lead to false data being generated at the receiver end. This chapter provides an asynchronous coding scheme that offers resilience to transient errors.

As technology scales down more IP cores are being integrated onto a single chip. Significant effort into the communication between the cores has resulted in extensive research of using Network-on-Chip (NoC) as the communication mechanism as highlighted in chapter 2. The NoC consists of switches and network interfaces connected together by links. Asynchronous methods of communication are finding their way into NoCs due to problems of power and clock distribution associated with synchronous circuits [153]. The asynchronous link can be broadly categorized into several styles such as bundled data, quasi-delay insensitive (QDI) and delay insensitive (DI). Bundled data relies on some relative timing to be kept between the data and a reference signal. DI, or self-timed, uses data encoding so the receiver knows when it receives valid data. There are numerous delay insensitive encodings such as Dual-Rail, 1 of 4, LEDR and multiple rail phase-encoding [110, 111, 113, 114].

As circuits shrink and integration increases errors will become more prominent [59]. Errors can fall into two broad categories, permanent and transient. Permanent errors are caused by the manufacturing process [154]. Transient errors can be caused by cross-talk, coupling or noise and particles. Up to 80% of errors can be caused by transient faults [60]. Dual rail, 1-or-4, Level encoded dual rail (LEDR) offer little resilience to transient errors which could lead to invalid data to be accepted at the receiver end of the link. Multiple rail phase-encoding improves on these by

offering an inherent resilience to transient errors during the idle times when data is not being transmitted but at the expense of complex receivers and transmitters as the number of wires increase. Resilience to soft-errors in NoC has been demonstrated in [155, 156] but these schemes use detection and correction at the router level. A link level detection scheme using hamming codes and interleaving has been shown in [60] at the expense of including de-interleaving and hamming distance decoding circuitry. A self correcting green joint coding scheme has been demonstrated in [116] to tolerate transient errors and reduce crosstalk through bus encoding and triplication error correction coding. Single event upset hardened pipeline interconnect has been presented in [115] but is proposed for synchronous links.

This chapter proposes the introduction of additional wires in order to use a bit symbol to represent the data bits. Using two wires per bit allows the data symbol to have four phases. Transient error resilience is achieved by exploiting the phase relationship between the data symbols and a common reference symbol. The chapter is organized as follows, the motivation for this work and examples of current asynchronous links is shown in Sections 5.1. Section 5.2 gives an overview of the proposed resilient link. Section 5.3 describes the proposed asynchronous transient resilient link architecture and the circuitry. Section 5.5 gives the experimental results and finally section 5.7 concludes the chapter.

5.1. Review of Current Asynchronous Coding and Motivation

In this section existing asynchronous links are considered and their limitations discussed. A single ended asynchronous link, such as bundled data uses a single reference signal to show when the data is valid, Fig. 5-1(a). It can be considered to have some resilience on the DATA wires as the receiver effectively ignores the data until the VALID signal is set. The relative timing between the DATA signals and the VALID signal in deep sub-micron design could diverge due to the tolerances or the wiring and the transistors in the gates. Delay insensitive codes are one of the ways to remove the dependency of timing between data and bundled reference signals since the de-coders do not care when the signals arrive [113].

Dual rail coding Fig. 5-1(b) was introduced to provide a delay insensitive solution to asynchronous data transfer. This introduced extra wires, 2 per data bit, which allows a ‘0’ or ‘1’ to be transmitted by asserting one wire or the other. In this

case asserting DATAA[x] transmits a 1 and asserting DATAB[x] transmits a 0. Fig. 5-1(c) shows level encoded dual rail (LEDR), using the same number of wires as standard dual rail it use uses less transitions per data bit. This is achieved by toggling the same wire if the data to be transmitted is the same as the previous or toggling the other wire if the data is different to the previous. Fig. 5-1(d) shows 1 of 4 encoding, where 4 wires are used to transmit 2 data bits. A single wire is asserted and then de-asserted to represent 2 bits of data, for example asserting wire A for the receiver to obtain data '00'. Fig. 5-1(e) shows an example of 4 wire multiple rail phase encoding. The information is contain in the arrival order of the edges rather than the logic level of the signals, care has to be taken to ensure that the arrival order of the signals remains the same as they propagate along the link.

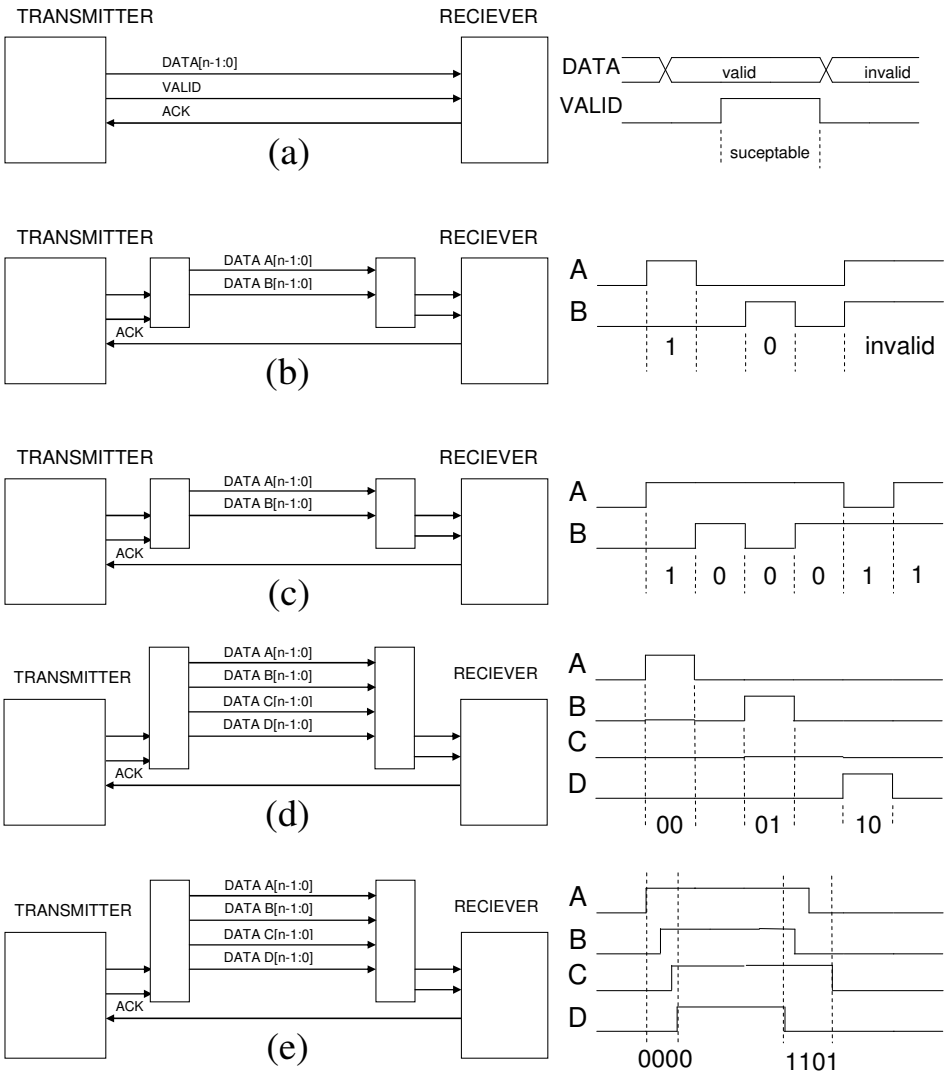


Fig. 5-1 Current asynchronous links [110, 111, 113, 114]

Recently a self correcting green join coding scheme for NoC interconnect has been proposed where the data is first encoded to minimise crosstalk and then a triplication error correction code applied to allow error correction [116]. The joint coding scheme increases the number of wires quite dramatically, for the crosstalk minimisation the increase is 1.25x and for the triplication it is 3x, meaning that the number of wires increases by 3.75x. This however is mitigated by serialization of the data before the triplication stage at the expense of higher link frequencies.

A single ended asynchronous link such as bundled data does offer some resilience to transients during the time period when valid is not asserted, but as technology scales down the issue of keeping relative timing between the bundled VALID reference signal and data may become an issue due to the tolerance variability in deep sub-micron designs. Delay insensitive techniques such as dual rail, LEDR and 1 of 4 were introduced to alleviate the relative timing problems and provide a solution in which delays do not matter. However, they are susceptible to transient faults which can corrupt the data. It may be possible to detect certain errors on dual-rail since the invalid code “11” could be detected which would suggest a single bit error at the same time data is transmitted. However, if a transient occurs on a one of the dual rail pair of wires when data is not been transmitted the receiver will see “10” or “01” instead of “00”. This could mean that the receiver detects this as a valid ‘1’ or ‘0’ data as opposed to a transient effect. The proposed encoding technique improves on this as it can cope with a single transient on a single wire of the pair of data wires even when data is not being transmitted.

A standard error detection scheme like parity checking such as adding an extra bit to the data converting to dual rail and back to single ended logic and checking the parity bit would allow the detection of a single error on one of the dual rail pairs. However, if more than one of the dual rail pairs was in error then a single parity bit cannot be used to detect both the errors. More complex schemes such as Hamming codes combine a codeword and data and send them together. For example a (7,4) Hamming code takes 4 bits of data and adds 3 code bits, resulting in a 7 bit message. Hamming codes have the advantage that a single bit error not only can be detected, but corrected too. Two bit errors can only be detected and not corrected though. The extra overhead for using Hamming codes would consist of extra logic circuitry to

code and decode the data and also 75% extra wires in the case of a (7,4) Hamming code.

The basic idea of introducing resilience is to provide some form of matching between different parts of a transmitted symbol so that the fault affecting one part can be filtered with the help of the other parts since the validity of the overall value is a 'collective responsibility' of all parts. In the phase-encoding [114] this is achieved by mutual adjudication between the wires. In the dual-rail framework, exploited in this chapter, the dual rail solution is built upon by introducing a pair of reference wires which can be compared with the pair of data wires in order to obtain the original transmitted data. The phase relationship between the reference and the data symbols provides the necessary information to obtain the original data. Both the reference and data symbols use four phases.

5.2. Proposed Resilient Link

The proposed technique uses a pair of wires per data bit plus a further pair of wires for a reference which is associated with the data bits, thus the number of wires will be (n*2)+2 for n bit wide link, Fig. 5-2. Each data bit and the reference is represented by a symbol on their pair of wires (00, 01, 11, 10). If the data symbol is in phase with the reference the data is '0'. If the data symbol is 180° out of phase with the reference the data is '1'. Should the data symbol be out of phase by ± 90° the data is can be considered invalid, Fig. 5-3. It is easy to detect invalid data with this system as an error on one of the wires of the data symbol will cause the symbol to be out of phase by ± 90° which is detected by the receiver.

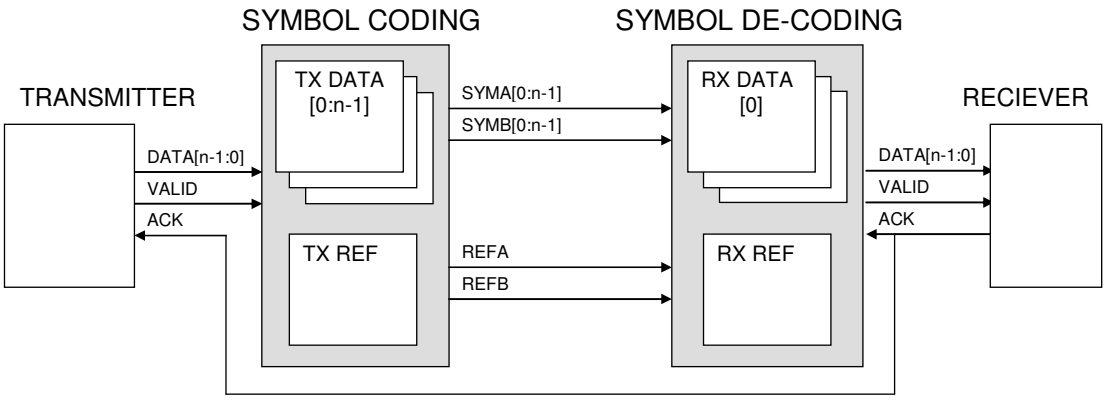


Fig. 5-2 Overview of proposed link

The reason of using a reference is that it allows the checking or validity of the symbol to see if it results in valid data or if there is an error present. A single reference pair of wires can be grouped with several pairs of data wires to support several bits transferred at a time. Each time a new piece of data is sent the reference increments around (moves around the quadrant by 90°) and the data moves either in-phase or 180° out of phase. By doing this there is only 1 transition on each pair of data wires and 1 transition on the pair of reference wires.

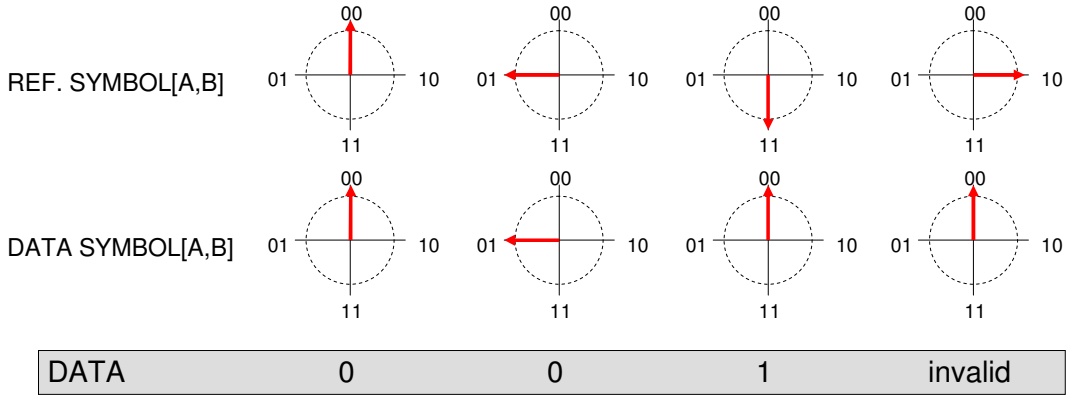
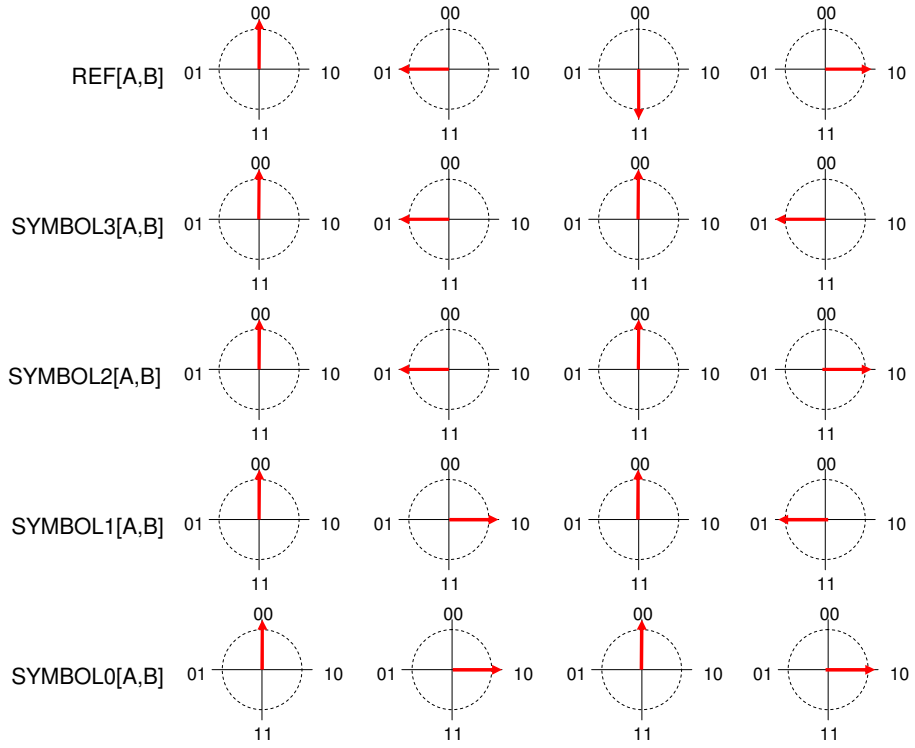


Fig. 5-3 Symbol and reference phase relationship

A further example to show how the coding for applies to multiple bits is shown in Fig. 5-4. As can be seen at first DATA[3:0] = “0000” as all the symbols are in-phase with the reference. The next piece of data SYMBOL3 and SYMBOL2 are in-phase so DATA3 and DATA2 are ‘0’ and SYMBOL1 and SYMBOL0 are 180° out of phase so DATA1 and DATA0 are ‘1’, thus DATA[3:0] = “0011”. The next piece of data all symbols are 180° out of phase so DATA[3:0] = “1111”. Finally the on the last piece of data SYMBOL3 and SYMBOL1 are out of phase and SYMBOL2 and SYMBOL0 are in-phase, so DATA[3:0] = “1010”



DATA3	0	0	1	1
DATA2	0	0	1	0
DATA1	0	1	1	1
DATA0	0	1	1	0

Fig. 5-4 Example symbol phase relationship for 4 bit wide data

Fig. 5-5 shows a state diagram of the coding technique. It can be basically considered as two cyclic planes. The low plane which cycles the symbol around 180° out of phase to the reference when the data is 1 and the higher plane which cycles the symbol around in-phase with the reference when the data is 0. The switching between the planes is when the data changes, the red arrows showing when the data is 0 and the coding switches into the in-phase plane and the blue arrows showing when the data is 1 and switching into the 180° out of phase plane.

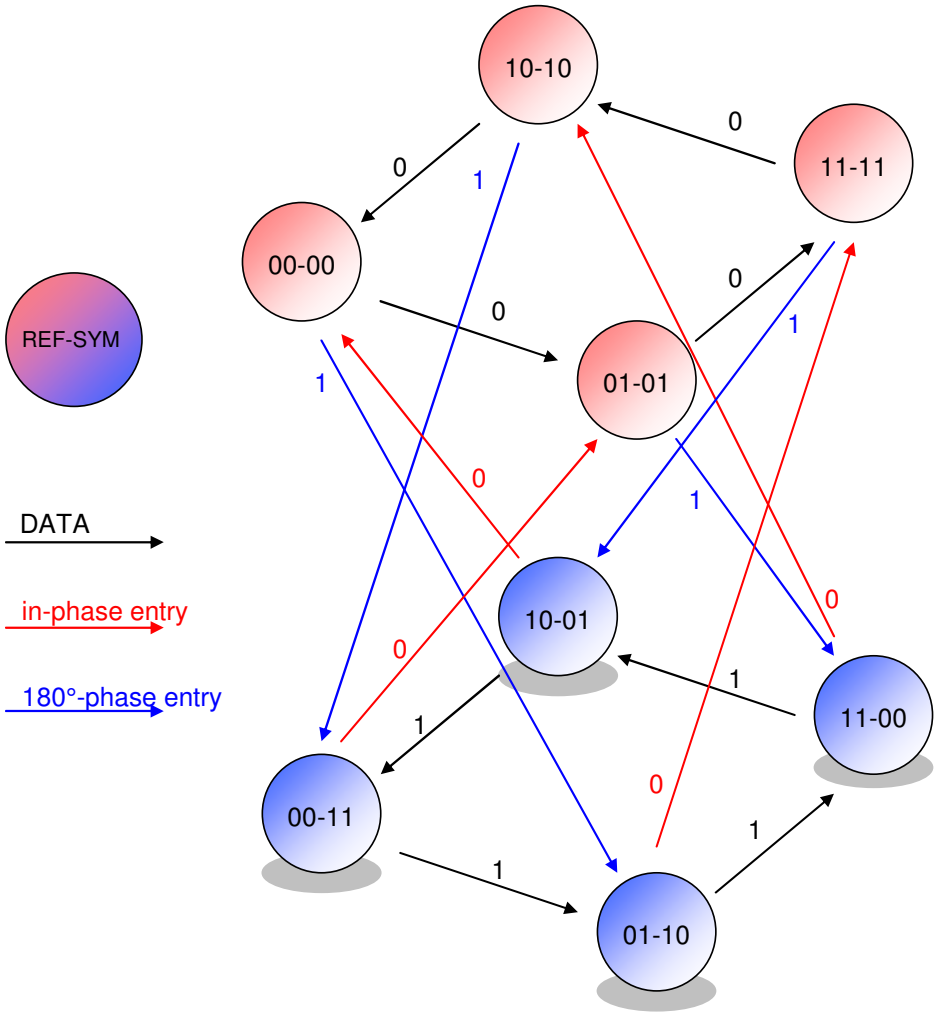


Fig. 5-5 Encoding State Diagram of proposed Link

Table 5-1 shows comparison between a number delay insensitive methods of data transfers that have been proposed for asynchronous links. LEDR, LETS and 1 of 4 encoding both improve on standard dual rail by reducing the transitions per bit but are still susceptible to transient faults. Multiple rail phase-encoding improves on these by offering resilience to transient faults and offering a reduced wire count when the number of bits is greater than four, but this comes at the expense of increasingly complex transmitter and receiver circuitry as the transmitter grows squarely and the receiver grows squarely or linearly dependent on the choice of decoding array in the receiver. The self correcting green coding scheme has high number of wires per bit so serialization can be employed to mitigate this. Our proposed approach offers a similar number of transitions per bit as LEDR and 1 of 4 as the number of bits increase from the point of view of power on the link and resilience to transients that multiple rail phase-encoding offers but with a linear growth in transmitter and receiver complexity.

Table 5-1 Comparison of proposed and existing links

Link	Wires/bit	Transitions /bit	Resilience to SEUs?	TX/RX growth
Bundled Data	1+1/n	0.5+2/n	N	-
Dual-Rail [113]	2	2	N	Linear
1 of 4 [110]	2	1	N	Linear
1 of 4 LETS[112]	2	0.5	N	Linear
LEDR [111]	2	1	N	Linear
Phase-enc [114]	w/n	w/n	Y	Square
S-C Green[116]	3.75	-	Y	Linear
Proposed	2+2/n	1+1/n	Y	Linear

where n = number of bits and (w-1)! < 2^n < w!

5.3. Proposed Link Architecture

The link consists of a transmitter for the reference symbol, a receiver for the reference symbol, transmitters for the data symbol and receiver for the data symbol. A single transmitter and receiver module pair for the reference is used with one or more transmitter and receiver pairs for the data as shown in Fig. 5-6 for an n bit wide data source. For example, an 8 bit wide data source would require 1x TX REF 1x RX REF, 8x TX DATA and 8x RX DATA modules. The SYMVALID outputs of the RX DATA modules can be ANDed together to form a single SIMVALID signal for the RX REF module. The VALIDO outputs of the RX DATA modules can also be ANDed together to provide a single VALIDO signal.

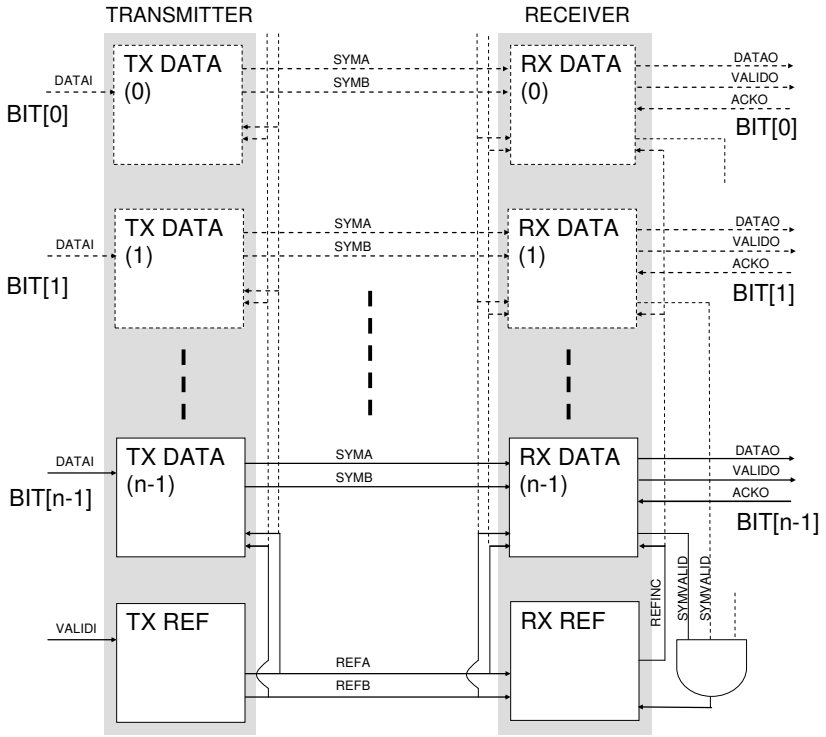


Fig. 5-6 Link showing circuit modules connectivity

5.3.1 TX DATA Circuit

The purpose of the TX DATA circuit is to provide a symbol output (SYM[A,B]) that will be in-phase or 180° out of phase with respect to the reference. The transmitter data module (TX data) is shown in Fig. 5-7. REF[A,B] is registered into the flip-flops untouched if DATA is 0 or inverted if DATA is 1 by use of the two XOR gates.

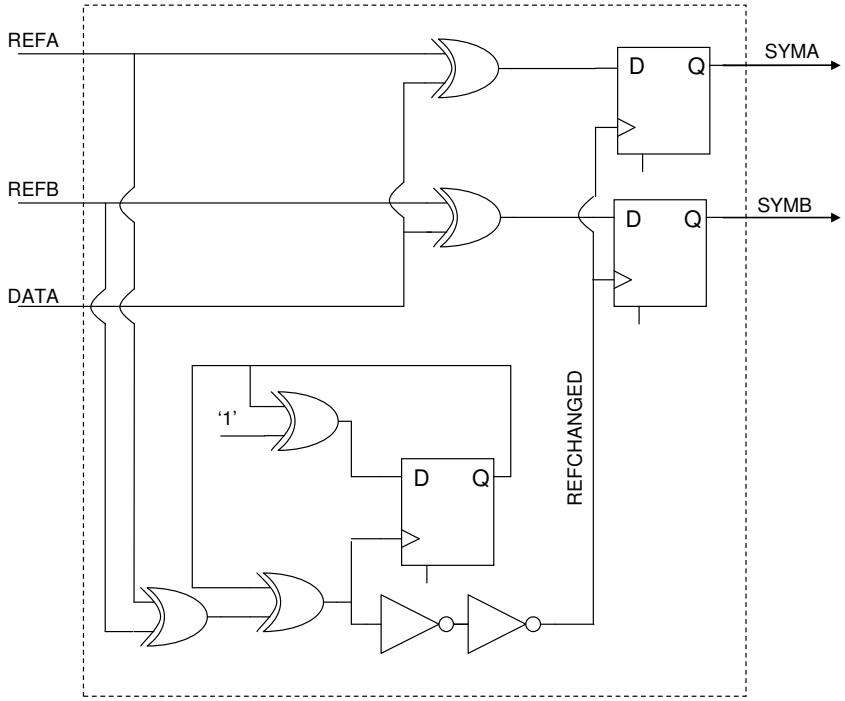


Fig. 5-7 TX DATA Circuit

The REFCHANGED signal goes high when the REF[A,B] changes and is a short pulse where the width is determined by the feedback time of the flip-flop. Consider the circuitry on it's own as shown in Fig. 5-8. The output of the XOR gate combining REFA and REFB is marked with X. The state transitions for two reference changes are shown in Fig. 5-9.

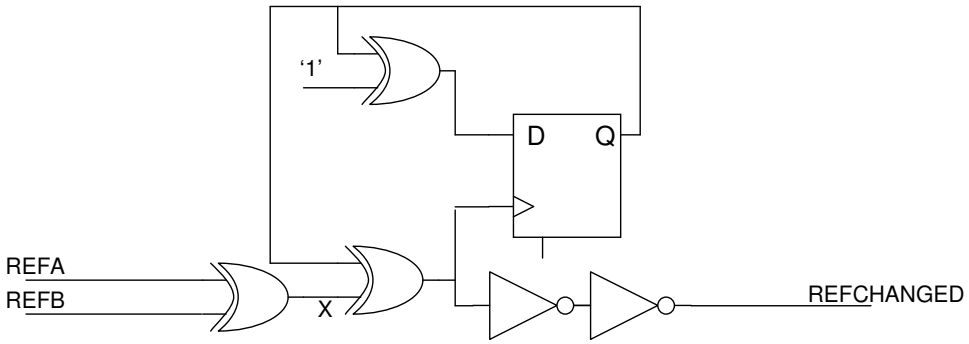


Fig. 5-8 REFCHANGED Circuitry

Examining Fig. 5-9 the stable states are highlighted. Looking when REF[A,B] = “00” it is shown that REFCHANGED is 0. When REF[A,B] changes to “01” this triggers REFCHANGED to go from 0→1, in turn Q goes 0→1 which causes REFCHANGED to go 1→0 and then remain in a stable state. The process is repeated every time REF[A,B] changes.

Fig. 5-9 Transition Table for REFCHANGED circuitry

REFA	REFB	X	D	Q	REFCHANGED
0	0	0	1	0	0
0	1	1	1	0	0→1
0	1	1	1	0→1	1
0	1	1	1→0	1	1→0
0	1	1	0	1	0
1	1	0	0	1	0→1
1	1	0	0	1→0	1
1	1	0	0→1	0	1→0
1	1	0	1	0	0
1	0	1	1	0	0→1
1	0	1	1	0→1	1
1	0	1	1→0	1	1→0
1	0	1	0	1	0
0	0	0	0	1	0→1
0	0	0	0	1→0	1
0	0	0	0→1	0	1→0
0	0	0	1	0	0

5.3.2 TX REF Circuit

The purpose of the TX REF circuit is to generate a new reference (REF[A,B]) each time VALID goes high. This can be achieved by using a gray counter. A present-next table for gray counter is shown in Fig. 5-10.

Fig. 5-10 Present-next Table for TX REF circuit

REF[ ]		REF[ ]next	
A	B	A	B
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

The next state Boolean equations for the reference are:

REFAnext = $\underline{\text{REFA}}$. REFB + REFA . REFB

REFBnext = $\underline{\text{REFA}}$. $\underline{\text{REFB}}$ + $\underline{\text{REFA}}$. REFB

A basic circuit can be realised by simply mapping these equations to the appropriate logic gates and 2x flip-flops triggered by the VALID signal. The resulting circuit is shown in Fig. 5-11. It is basically a grey code counter which increments the output REF[A,B] through the symbols 00, 01, 11, 10 each time VALID goes high.

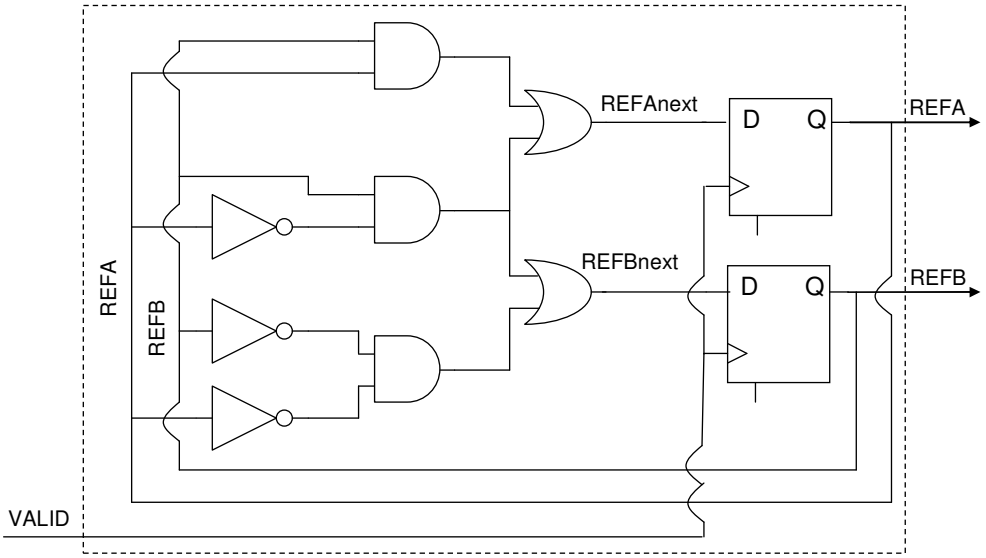


Fig. 5-11 TX REF Circuit

5.3.3 RX DATA Circuit

The purpose of the RX DATA circuit is to decode compare the symbol and reference to generated a valid data and also to tell the RX REF circuit when it has detected a valid symbol in order that the RX REF circuit can register a copy of the current reference for comparison against the next reference to check that the reference increments. The DATA decoding is simply comparing the symbol SYMB[A,B] against the reference REF[A,B]and setting data correctly. SYMVALID is also generated from comparing the reference and symbol and generating a high signal if the symbol is in-phase or 180° out of phase otherwise it should be low to signify the symbol is ±90°out of phase and therefore not valid. The truth table for DATA and SYMVALID is shown in Fig. 5-12.

Fig. 5-12 DATA and SYMVALID truth table

REFA	REFB	SYMA	SYMB		DATA		SYMVALID
0	0	0	0		0		1
0	0	0	1		X		0
0	0	1	0		X		0
0	0	1	1		1		1
0	1	0	0		X		0
0	1	0	1		0		1
0	1	1	0		1		1
0	1	1	1		X		0
1	0	0	0		X		0
1	0	0	1		1		1
1	0	1	0		0		1
1	0	1	1		X		0
1	1	0	0		1		1
1	1	0	1		X		0
1	1	1	0		X		0
1	1	1	1		0		1

The Boolean equations for DATA and SYMVALID reduce to:

DATA = SYMA ⊕ REFA

SYMVALID = $\underline{(\text{REFA} \oplus \text{SYMA}) \oplus (\text{REFB} \oplus \text{SYMB})}$

The receiver data module (RX Data) is shown in Fig. 5-13. The circuit generates a SYMVALID signal if the REF[A,B] matches SYM[A,B] or the inverse, which is out of phase by 180°. When SYMVALID is high and REFINC goes high, DATA and

VALID are registered into their respective flip-flops. ACK going high will clear the valid signal.

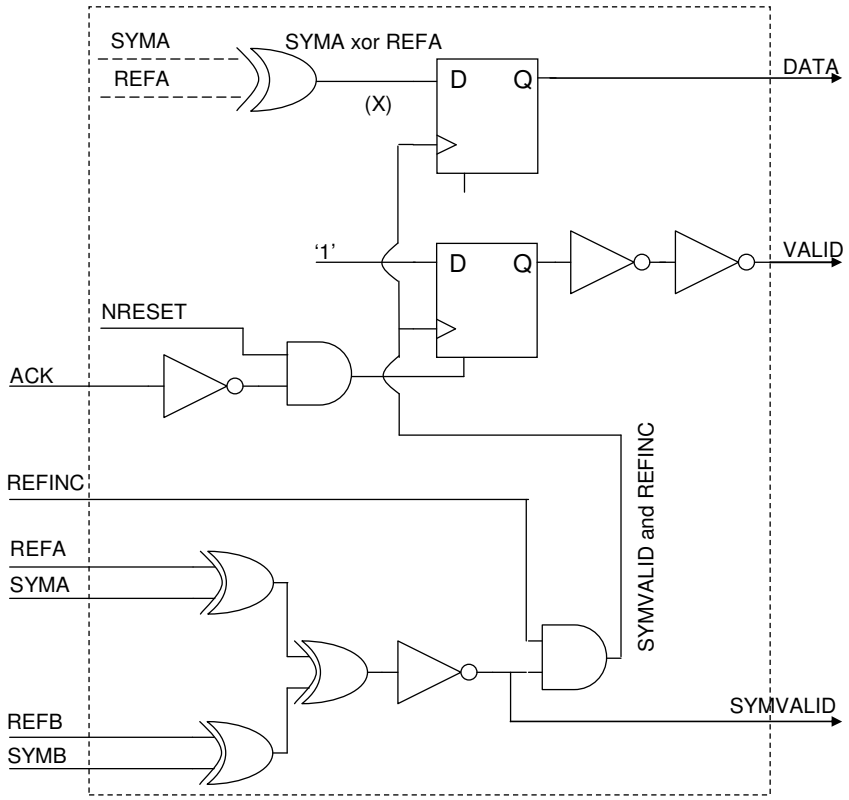


Fig. 5-13 RX DATA Circuit

5.3.4 RX REF Circuit

The purpose of the RX REF circuit is to generate a signal which signifies that the reference has incremented around 90° (same as incrementing in gray code). The instinctive way of doing this is to compare the current reference with the old reference and then generate a signal if it is incremented. If we define REFINC as the signal which signifies that the reference has incremented [A,B] as the current reference and [OA,OB] as the old reference then the truth table for REFINC is shown in Fig. 5-14.

Fig. 5-14 Truth Table for REFINC

A	B		OA	OB		REFINC
0	0		0	0		0
0	0		0	1		0
0	**0**		**1**	**0**		**1**
0	0		1	1		0
0	**1**		**0**	**0**		**1**
0	1		0	1		0
0	1		1	0		0
0	1		1	1		0
1	0		0	0		0
1	0		0	1		0
1	0		1	0		0
1	**0**		**1**	**1**		**1**
1	1		0	0		0
1	**1**		**0**	**1**		**1**
1	1		1	0		0
1	1		1	1		0

From the truth table we obtain the sum of products Boolean equation for REFINC:

REFINC = $\underline{A}.\underline{B}.OA.\underline{OB} + \underline{A}.B.\underline{OA}.\underline{OB} + A.\underline{B}.OA.OB + A.B.\underline{OA}.OB$

Which further reduces to:

REFINC = $\underline{A}.\underline{OB}.(B \oplus OA) + A.OB.(B \oplus OA)$

REFINC = $\underline{A \oplus OB} \,.\, B \oplus OA$

The circuitry consists of 2x flip-flops, 2x XOR gates, 1x Inverter and 1x AND gate. The flip-flops functions are to hold the old reference and the remaining gates perform the function of detecting the current reference is incremented. The complete RX REF circuit is shown in Fig. 5-15. A further signal is introduced called SYMVALID. This

signal goes high when the RX DATA circuit has found that the symbol it has received is valid. When SYMVALID goes high the current reference, REF[A,B], is registered into OLD[A,B] for comparison on the next transfer and REFINC therefore goes low.

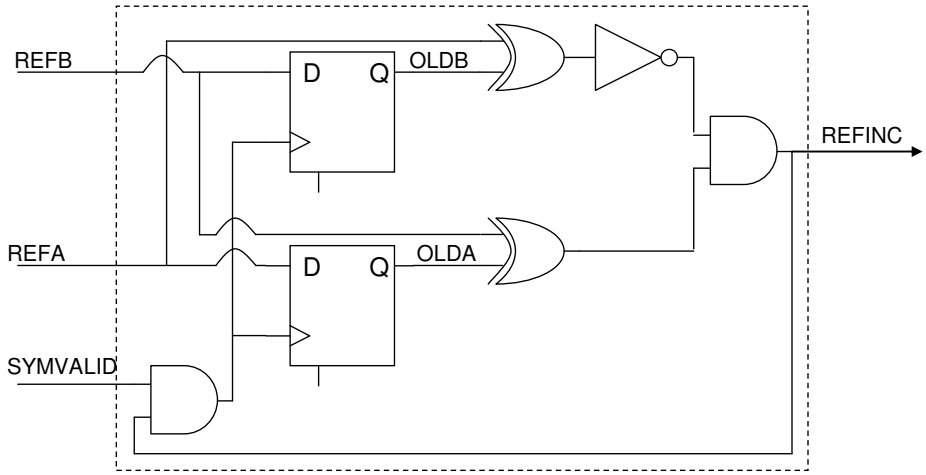


Fig. 5-15 RX REF Circuit

5.4. Resilience

Although the link is resilient to transient faults it is not totally immune from them. An example is given in Fig. 5-16 which shows that under certain conditions invalid data could be latched out on the receiver end. Examining Fig. 5-13 and Fig. 5-15 we can show the normal sequence of events when a valid reference and data symbol arrives at the receiver end, Fig. 5-16(a). The valid data symbol (SYM[A,B]) and reference (REF[A,B]) generates the SYMVALID which combined with REFINC through an 'and' gate generates the signal to latch SYMA xor REFA onto the DATA output. Provided that no transients affect or corrupt SYMA xor REFA before it is latched into the flip-flop then valid data is obtained. Thus the input to the flip-flop must remain stable during its setup time period. Fig. 5-16(b) shows what happens if transients occur within the setup time of the flip-flop which latches DATA. The transient ripples through and corrupts SYMA xor REFA causing invalid data to be latched onto the DATA output.

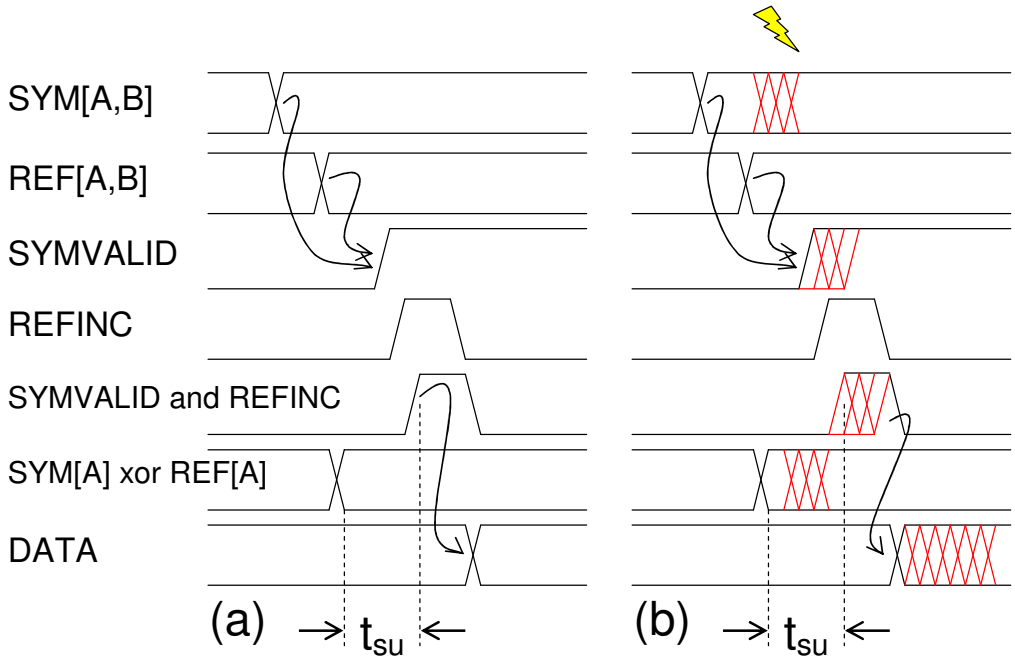


Fig. 5-16 RX DATA timing

The setup time of the flip-flop used is approximately 125 ps nominal which can be obtained from the data sheet (ST CORE9GPHS HCMOS9 data book [157]). With this it is possible to find out the probability if the data being corrupted if a transient fault does occur. Define T_w as the transient width, t_{period} as the inverse of the operating frequency and FF_{su} as the flip-flop setup time. Assuming that the transient is

caused by the environment, such as particles and that the time it affects the circuit is completely random and can assume a uniform distribution of the transient occurring in t_{period} it is possible to use the following formula to predict the probability that data will be corrupted.

$$P(corruption) = \frac{T_w + FF_{su}}{t_{period}}$$

Using a FF_{su} of 125 ps the probability of the data being corrupted can be shown if a single transient occurs on one of the wire pairs and while data is being transmitted. From this equation Fig. 5-17 was generated which shows the probability of corruption versus the operating frequency for transients widths of 100, 200, 300 and 400 ps if a transient occurs. As can be seen for a given transient width the probability of corruption increases with frequency. For a given frequency it can be seen that the transient width also affect the probability of corruption. For example at 1000 MHz the probability of corruption is approximately 0.23 if a 100 ps transient affects the symbol and rises to around 0.58 when the transient width increases to 400 ps.

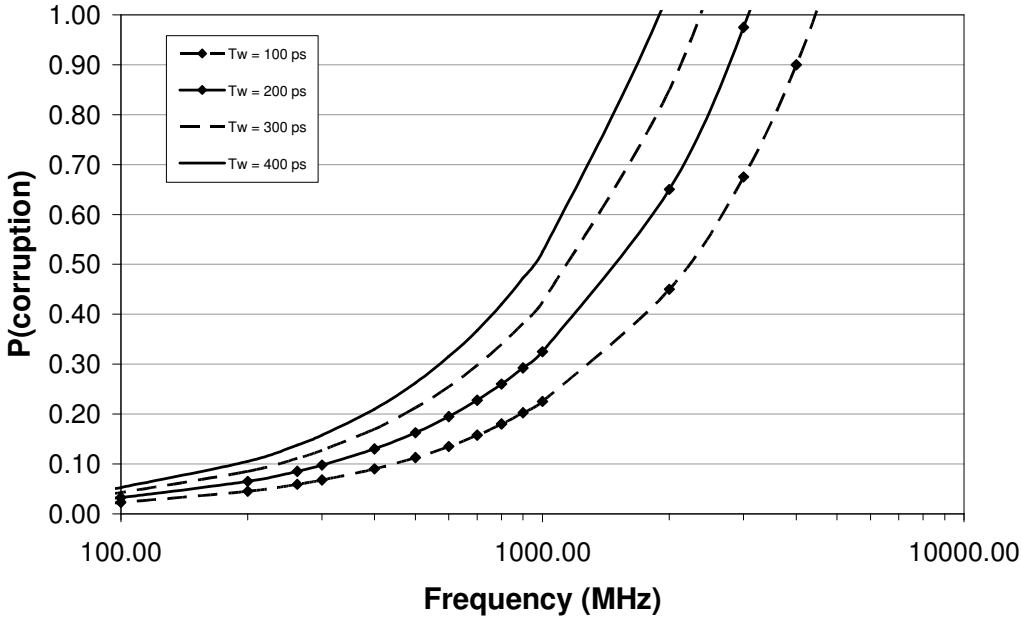


Fig. 5-17 Probability of corruption for a single transient

5.5. Experimental Results

To evaluate the resilience of the proposed link SpectreVerilog simulations were performed. The test bench scenario is shown in Fig. 5-18. The receiver and transmitter are circuit level designs. The driver and receiver (TB driver.v and TB receiver.v) are verilog modules which generate the appropriate handshaking for the asynchronous interfaces. The driver and receiver verilog modules use the handshake timings shown in Table 5-2 for Fig. 5-19 to Fig. 5-21.

Table 5-2 Test bench handshake parameters

Cycle	Time	Module
$DATA_{VALID}$ to $VALID_{HIGH}$	1 ns	TB driver.v
ACK_{HIGH} to $VALID_{LOW}$	1 ns	TB driver.v
$VALID_{HIGH}$ to ACK_{HIGH}	1 ns	TB receiver.v
$VALID_{LOW}$ to ACK_{LOW}	1 ns	TB receiver.v

The pulse stream generator (TB pulses.v) uses a single bit in a 10 bit LFSR operating at a certain frequency (~115 MHz in the case of Fig. 5-19 to Fig. 5-21) to generate several 300 ps wide pulses which are then XORed into the chosen signal. The output of the XOR will then have sporadic transients present within it. This can be spliced into any of the wires to provide a noisy or transient infected signal as required during simulation. If more than one wire is needed to have transients present then further pulse stream generators can be used with different LFSR starting seeds and frequencies as required.

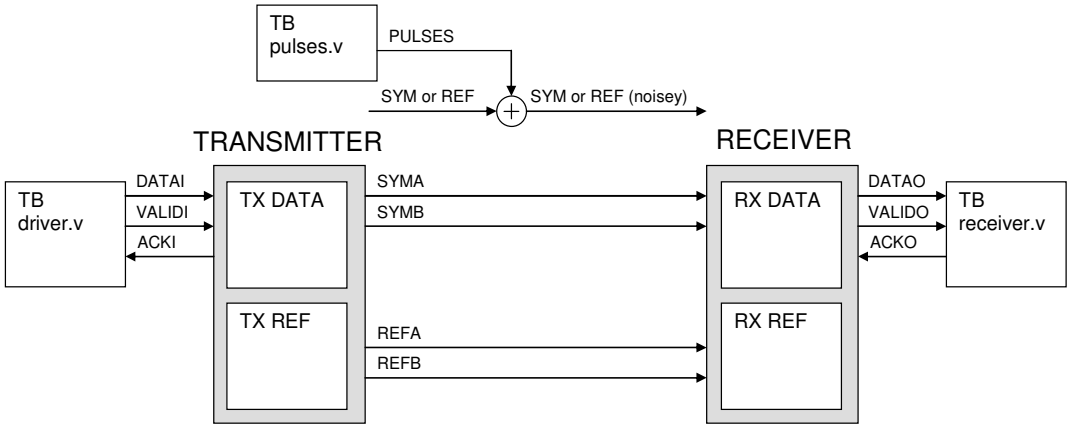


Fig. 5-18 Test bench setup

Fig. 5-19 shows the signal waveforms from simulation. As can be seen the transmitted data (TX DATA) is the same as the received data (RX DATA). Also note that REF[A,B] can clearly be seen incrementing through 00, 01, 11, 10, 00, …

SYM[A,B] can also be seen to be in-phase for DATAI=0 or 180°-phase for DATAI=1. Fig. 5-20 and Fig. 5-21 show transients on a SYM and REF wire respectively. Note that the received data (DATAO) is received correctly even though transients are corrupting one of the symbol or reference wires.

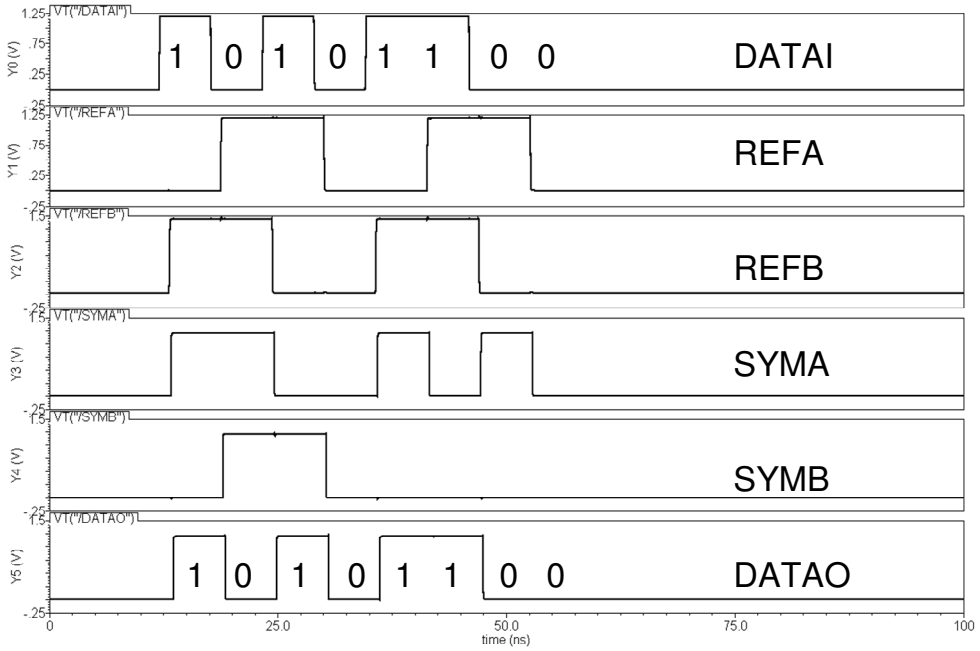


Fig. 5-19 Reference and symbol signalling

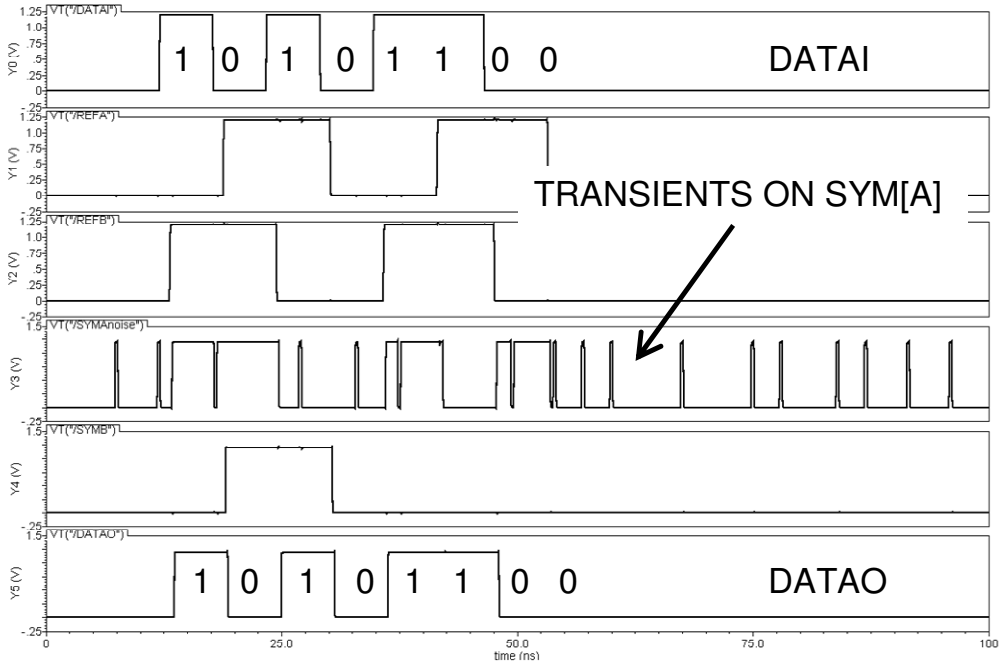


Fig. 5-20 Transients on a symbol wire

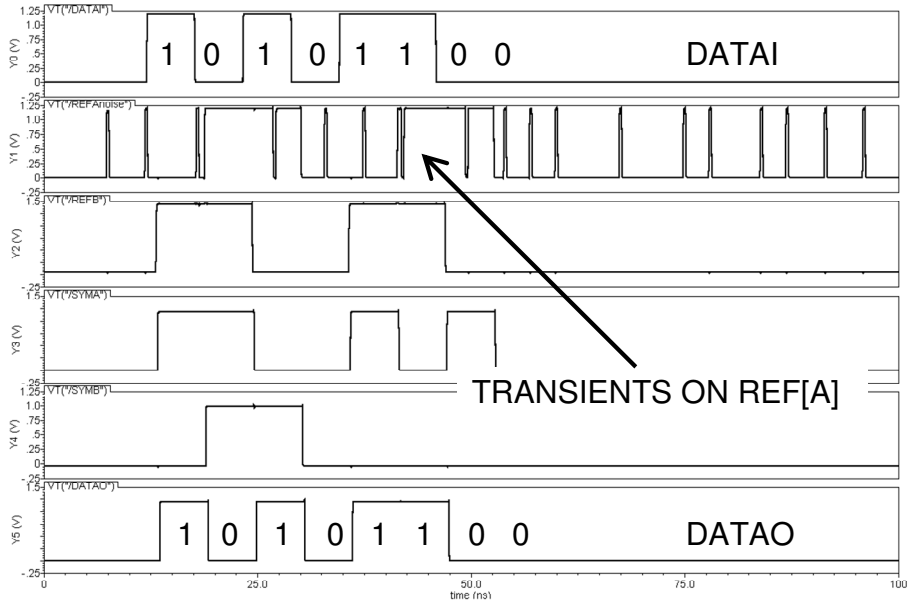


Fig. 5-21 Transients on a reference wire

Fig. 5-22 shows transients on both symbols wires during the data transfer period. The handshaking timing was shortened to 300 ps for each of the parameters in Table 5-2 for this simulation and data pattern 0xFF00AAAAAA00FF was used. Again it can be seen that the output data matches the input data and is not corrupted.

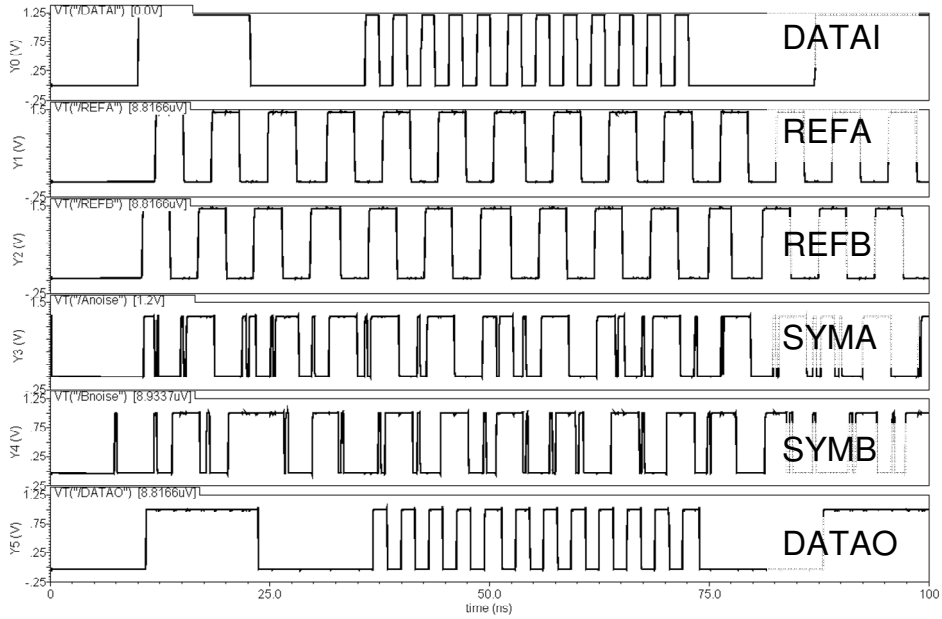


Fig. 5-22 Transients on a SYM[A,B] pair

5.5.1 Throughput and Latency

To give an indication of the maximum operating frequency of the circuits handshaking parameters shown in Table 5-2 were all set to 0 with the exception of $DATA_{VALID}$ to $VALID_{HIGH}$ which was set to 300 ps to ensure that DATA was valid

before the VALID signal goes high. Running a simulation showed that 32 back to back transfers occurred in a time period of 30.3 ns. Thus a single transfer happens in 0.95 ns, giving a theoretical operating frequency of 1.056 GHz. It is important to note that the actual operating frequency of a complete link would be lower than this as the handshaking timing would need to be based on the speed of the handshaking of the asynchronous circuitry interfacing to the link. To give insight of the extra latency introduced by the encoding and decoding the time from DATAI to DATAO in the simulated waveforms was obtained. The latency introduced by the circuitry is 0.8 ns. However, it should be noted that the latency in a physical implementation will be more than this as wire delays are introduced and need to be taken into consideration.

5.5.2 Area Overhead

Table 5-3 shows the area overhead of the circuits in the 0.12µm technology used to simulate the circuits. (ST core9gphs). The TX DATA and RX DATA modules are significantly larger then for the TX REF and RX REF modules. This is due to increased complexity to the data modules which need to compared against the reference in order to generate their outputs. The extra overhead per bit is 409.49 μm^2 and the overhead of the reference is 262.24 μm^2. For example, for an 8 bit wide link the total area overhead would be 3538.16 μm^2 (262.24 μm^2 + 8*409.49 μm^2). Also shown is estimated area of the multiple rail phase encoding scheme [114] which also offers transient resilience but the complexity of the circuit does not have linear growth as the number of bits increase, the estimation of the circuitry is shown in more detail in Appendix E. For a 1 bit link multiple rail phase encoding does offer an advantage in terms of area, but for 8 bits the area has ballooned considerably to 55435.99 μm^2 due to some of the circuitry growing squarely in size.

Table 5-3 Area Overhead of Links (μm^2)

Circuit	1 bit	8 bit	16 bit
TX DATA	215.84	1726.72	3453.44
RX DATA	193.65	1549.20	3098.40
TX REF	137.17	137.17	137.17
RX REF	125.07	125.07	125.07
Proposed Total	671.73	3538.16	6814.08
Multiple Rail Phase Encoding[114]	195.67	55435.99	-

5.5.3 Power Consumption

To give insight into the average power consumption within the various circuit modules the data pattern 0xFF00AAAAAA00FF was sent bit serially from the test bench driver to the circuit. The breakdown of the power used in the four circuit modules is shown in Table 5-4. The power was obtained from a SpectreVerilog simulation by taking the average of the supply voltage multiplied by the current into each circuit module over the simulation run time. The total dynamic power for the DATA modules is 199.47 µW. The simulation run time was 100 ns, by multiplying the power by the simulation run time the energy used to transmit the data can be obtained. Multiplying the power by the time (199.47 µW × 100 ns) gives an approximate energy usage of 0.0199 pJ to transfer the whole 56 bit data pattern, dividing this by 56 gives an energy per bit of 356 fJ/bit. The REF modules power is 85.66 µW giving an energy usage of 0.0086 pJ to transfer the data pattern. The energy per data transfer is 153 fJ. For example, for an 8 bit wide link the energy for each data transfer is 3001 fJ (153 fJ + 8*356 fJ).

Table 5-4 Dynamic and Static Average Power (µW)

Circuit	Power (dynamic)	Power (static)	Power (total)
TX DATA	91.72	0.72	92.47
TX REF	41.87	0.47	42.34
RX DATA	106.26	0.74	107.00
RX REF	43.00	0.32	43.32

Examining Table 5-4 it is shown that the RX DATA circuit (Section 5.3.3) consumes the most power of 107 µW. This is because the RX DATA circuit is the most complex relative to the three other circuit modules with the largest gate count. Examining TX REF and RX REF circuit modules it is shown that the RX REF consumes more dynamic power than TX REF (43 µW for RX REF and 41.87 µW for TX REF). However, static power for RX REF is lower than TX REF (0.32 µW for RX REF and 0.47 µW for TX REF) suggesting that the switching activity for TX REF is lower than RX REF but the static power is slightly more for TX REF due to higher number of gates used in the circuit module.

5.5.4 Limits of Resilience

In order to show how corruption of the data can occur the test bench simulation was run with a transient pulse T that was XOR'd with one of the symbols wires S. The frequency or repetition rate of the transients was slightly slower that that of the symbol rate in order that the transient would sweep through and corrupt the symbols at different points. Fig. 5-23 and Fig. 5-24 shows how the transient affected symbol wire (S XOR T) leads to a corrupted DATAOUT. Examining Fig. 5-23 we can see that on the corrupted symbol wire (S XOR T) the transient slowly bleeds into the rising edge of the symbol causing it to go to logic 0. At a certain point shown in the waveforms the symbol is low and gets caught in setup and hold time of the flip-flip which provides DATAOUT based on the corrupted symbol wire. If the setup and hold time is violated the DATAOUT is inversed as the decoding is basically SYMA ⊕ REFA (Section 5.3.2) hence DATAOUT is logic 0 when it should be logic 1. Fig. 5-24 shows the situation where the transient bleeds into the falling edge of the symbol wire, in this situation the DATAOUT is now at logic 1 instead of being at logic 0.

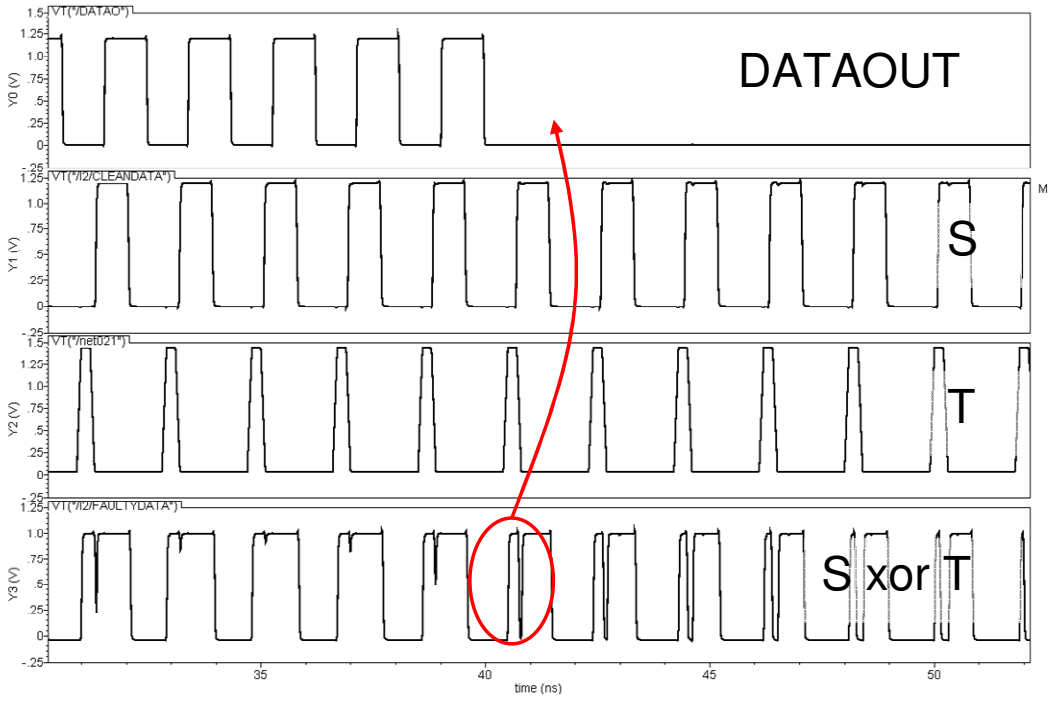


Fig. 5-23 Corruption of the Data on positive symbol edge

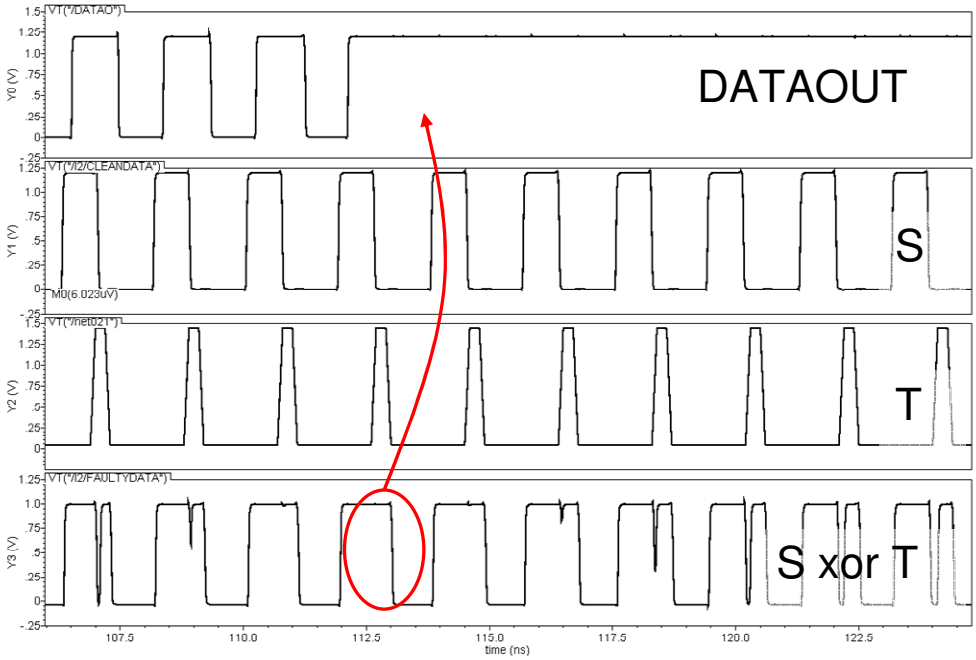


Fig. 5-24 Corruption of the Data on negative symbol edge

To show that the transient pulse width has an effect on the corruption the transient width (Tw) was varied between 100 and 400 ps to show the effect on DATAOUT. Fig. 5-25 shows the effect on DATAOUT, as can be seen when the transient width is small (Tw = 100 ps) the amount of times DATAOUT out is corrupted and does not change is small. When the transient width is increased the amount of times DATAOUT is corrupted becomes larger, the gap or period of time that DATAOUT stays at logic 0 or logic 1 is increased.

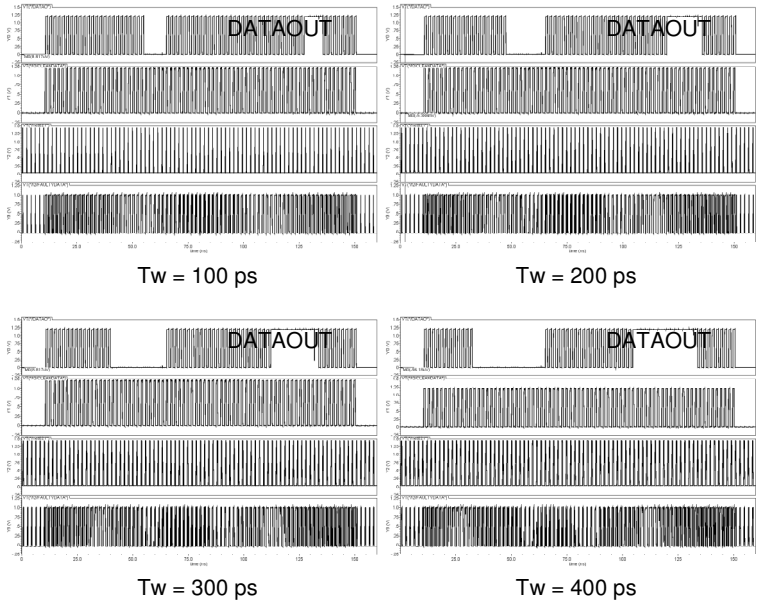


Fig. 5-25 Example corruption of data for various transient widths

To further verify the probability of a single event transient corrupting the data and to provide evidence that the estimated probability of corruption in Fig. 5-17 is reasonable, further simulation was performed which sweeps the a transient through the symbol in order to corrupt the data. A series of 300 bits of alternating 1's and 0's were sent across the link and one of the symbol wires (SYMA) had a transient pulse superimposed on it by use of an XOR gate to corrupt the data seen at the input to the flip-flop in the RX DATA circuit (marked X in Fig. 5-16). The repetition rate of the transient was 555 MHz, this was chosen to be of a similar frequency to the rate at which the symbol changes (526 Msym/s) in order that the transient affects each symbol at a different position over the period it takes to transmit the 300 bits. Fig. 5-26 shows the calculated and simulated probability of a bit error occurring with transient widths of 100 to 600 ps. The calculated values were obtained by using the equation from Section 5.4:

$$P(corruption) = \frac{T_w + FF_{su}}{t_{period}}$$

The flip-flop setup time (FF_{su}) was 125 ps and the time period (t_{period}) was 1.901 ns ($1/526\times10^6$), the probability of corruption was obtained for transient widths (T_w) of 100 to 600 ps. The simulated values were obtained from by sweeping a transient through the symbols and counting the number of bits in error. The transient was injected into the circuit by use of an XOR gate. One of the symbol wires from the transmitter was fed into one XOR input, the other input was connected to a pulse generator. The output of the XOR gate was then connected to the input symbol wire on the receiver. As can be seen the trend of the curve is similar to calculated, but slightly lower offset. This could be due to the fact that in simulation even if the transient affects the symbol enough to encroach into the setup time it may still not be enough to cause a violation within the analogue simulation. For a transient width of 100 ps the simulation shows that approximately 6% of the time the received data was corrupted. Increasing the transient width to 600 ps increased the rate of data corruption to approximately 34%. This level of corruption is to be expected as a transient width of 600 ps is around 1/3rd of the total time period of the symbol which is 1.901 ns and will therefore lead to corrupt data 1/3rd of the time.

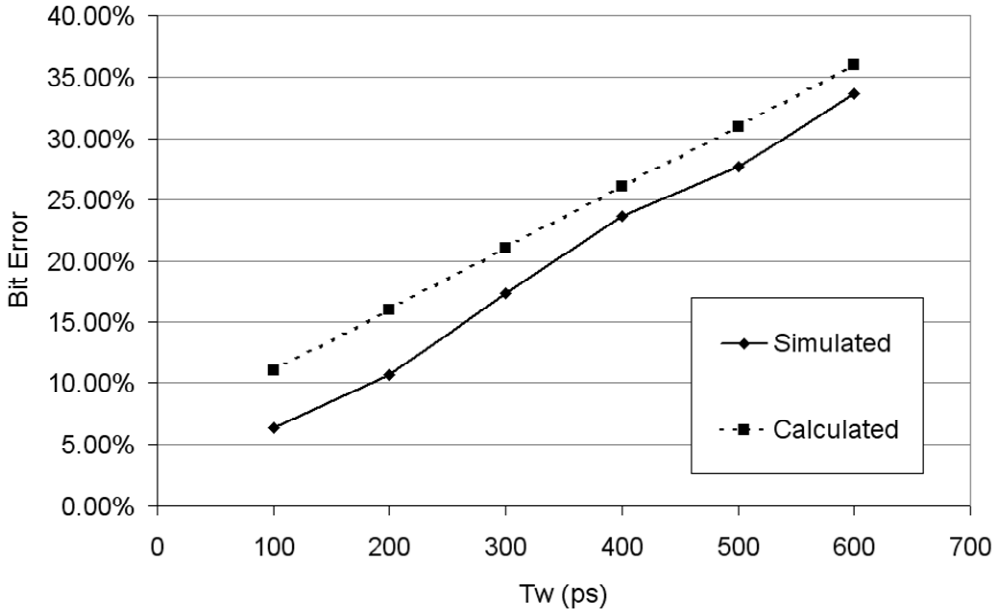


Fig. 5-26 Transient width vs Bit Error for 300 data bits

5.6. Wire Buffering

The work in this chapter has concentrated on the coding method and the associated circuitry. For long links wire buffers may be needed to segment the long wire into shorter sections and allow a pipelined structure to be used. To give an indication of how the wire buffers could be implemented this section describes two possible implementations of wire buffers and discusses the complexity. The choice of structure for wire buffers could range from simple to more complex depending on the needs of the link. For example several buffers could be placed along the length of the wire but this would mean the return acknowledgement signal would have to traverse the whole length of the link back before the next piece of data is transferred, Fig. 5-27.

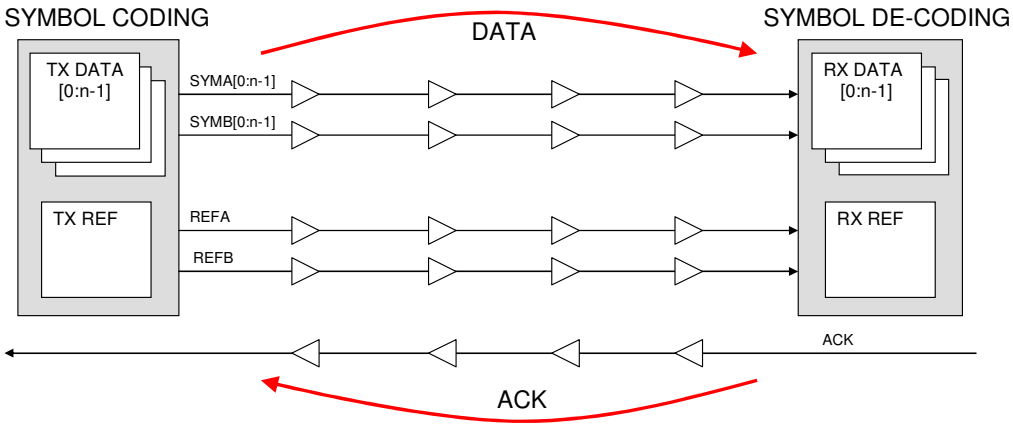


Fig. 5-27 Simple Wire Buffers

A better approach would be to register or latch the signals along the length of the wire, Fig. 5-28. This would require some sort of memory element or flip flop with some associated handshaking circuitry.

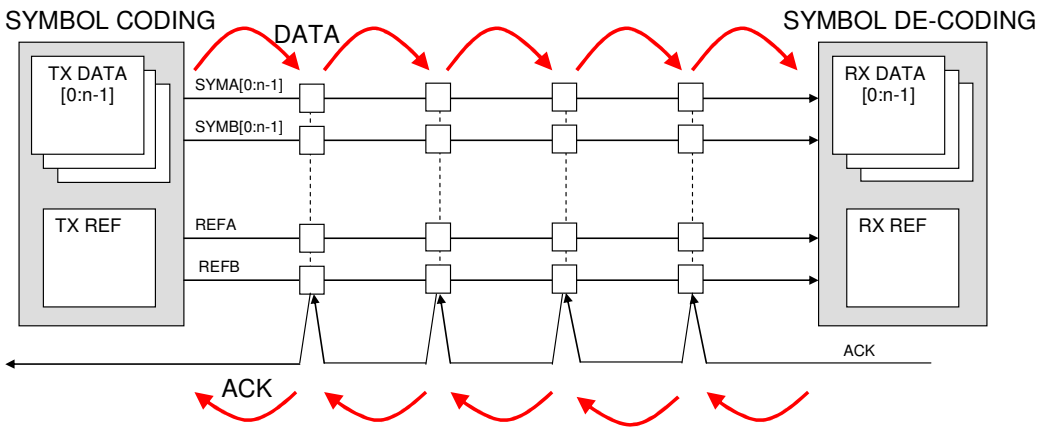


Fig. 5-28 Latched or Registered Wire buffers

If the relative timing between SYM[A,B] and REF[A,B] is known to be close at the transmitter we could in theory obtaining the acknowledgement handshaking by observing that REF[A,B] has changed and just registering the SYM[A,B] independently. This would cause the link not to be truly delay insensitive anymore since if the SYM[A,B] were generated faster or slower than their associated REF[A,B] signal you could potentially have a situation where the wrong SYM was trying to be matched with the REF at the receiver end. An example of such a wire buffer is shown in Fig. 5-29. Both the SYM and REF signals are registered into a flip-flop by XORing the signal with a delayed version of itself (the two invertors) to generate the flip-flop clock or latch signal. The ACK out is generated by ORing the two clock or latch signals of the REF[A,B]. This sets the S-R flip-flop which is fed into a C-Element (Fig. 4-5). The C-Element is then set and the ACK out signal set high. The ACK out also resets the S-R flip-flop and the C-Element now waits for an acknowledgement from the next wire buffer in order to set ACK out low again and repeat the process.

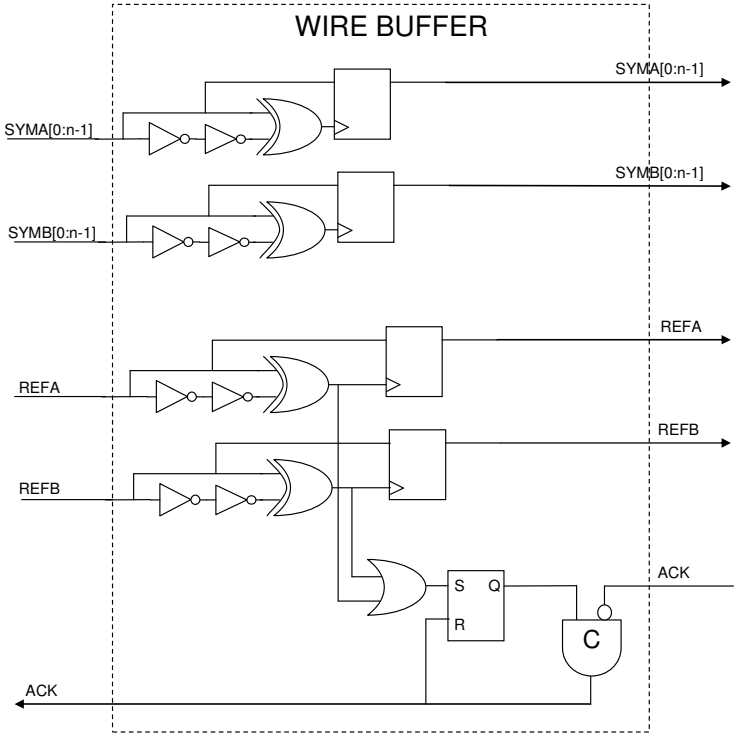


Fig. 5-29 Wire Buffer

This wire buffer would allow pipelining of the signals but it could also propagate transient faults along the link since the clocking or latching of the signals is done when a change is detected, it is effectively a 'dumb' latching mechanism. In order to get around this and also make the link back to a true delay insensitive scheme we need

to introduce the circuitry which matches the SYM and REF signals and produces a valid output when the match is detected. Fig. 5-30 shows an improved wire buffer which generates the clocking or latching signal when there is a valid match between the SYM and REF signal shown in the shaded area. It is important to note however that this shaded circuitry will need to be replicated n times for an n bit wide link and then gated together to form a single latching signal. The clocking or latching signal is converted to a pulse to set the flip-flip which triggers the C-Element and handshaking the same as the previous wire buffer shown. This improved wire buffer should be truly delay insensitive since the signals are only latched when a valid match between the SYM and REF signals exists. The latency of the handshaking could increase as the number of bits n increases since the matching circuitry would have to be gated together to form a single valid signal, this could introduce another layer or two of gates into the handshaking feedback timing.

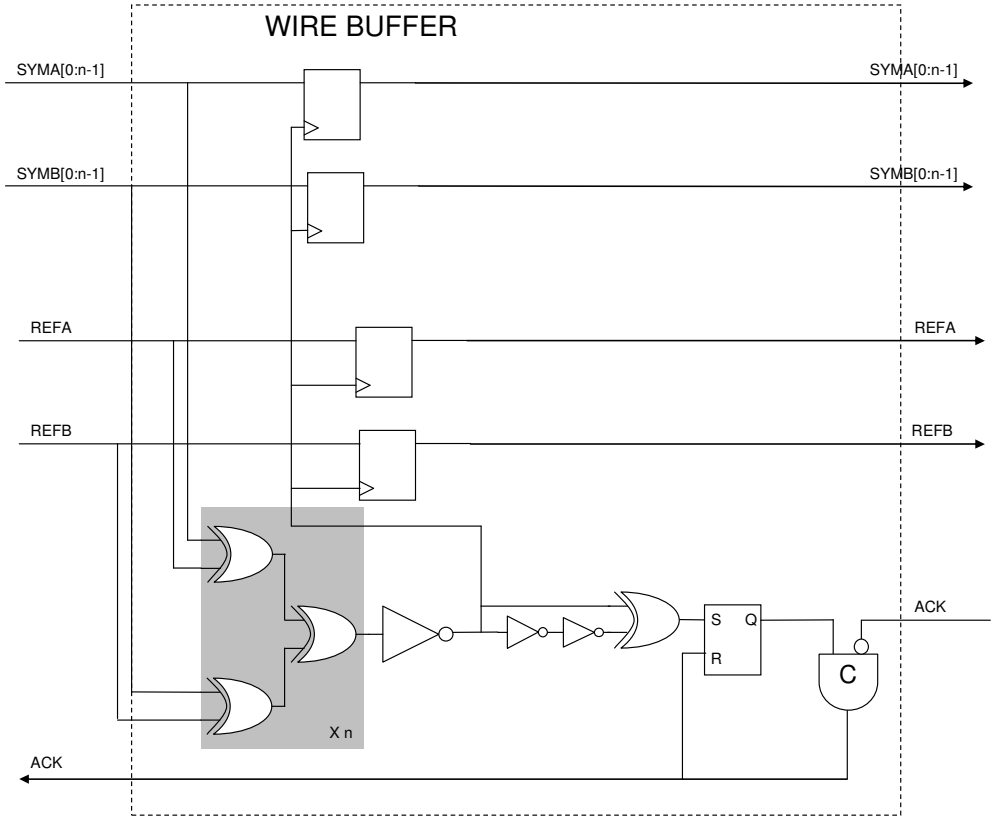


Fig. 5-30 Improved Wire Buffer

A selection of possible wire buffers has been presented, careful consideration as to the needs of delay insensitivity, throughput and complexity of the wire buffer needs to be performed in order that an appropriate choice can be made.

5.7. Concluding Remarks

This chapter has proposed and demonstrated an asynchronous link technique that has resilience to soft-errors which is achieved by the use of a novel coding technique. The proposed coding technique has been applied to an asynchronous NoC link. The coding technique is based on transmitting symbols for each bit and a common reference signal. The relationship between the data symbols and a reference is compared in order to decode the data. A single error on one of the symbols can be detected if there is a mismatch between the symbol and the reference. When compared to dual-rail the proposed coding increases the resilience of the link to soft-errors at the expense of extra circuitry required to encoded and decode the symbols. The proposed link uses a similar number of wires as dual-rail, ($\Rightarrow$ 2) and offers a similar number of transitions per bit as 1-of-4 encoding ($\Rightarrow$ 1). The link was designed and simulated in 0.12µm technology.

The proposed link has an area overhead of 409.49 μm^2 per bit and an approximate energy usage of 356 fJ/bit. Possible implementations of two wire buffers for the link have been discussed and examples given, one type of wire buffer could be used if the relative timing between the reference and the symbols could be kept close or the alternative wire buffer which checks the validity of the symbol and reference before buffering the symbol and reference. Simulation of the link using the proposed coding was carried out using gate level simulation in the analog environment in SpectreVerilog. The maximum operating frequency of the coding and decoding circuitry was found to be 1.056 GHz although in practice this would be slower due handshake timing of the surrounding asynchronous circuitry.

It is hoped that the proposed link makes a valuable contribution to the area of efficient and reliable NoC architecture for multi-processor SoC.

Chapter 6. Conclusions and Future Work

It is likely that the demand for multiprocessor system on-chip will continue to increase as more IP cores are integrated onto a single chip. With the increasing number of cores new communication mechanisms will be needed to address the scalability of the on-chip communication architecture. Network-on-chip is emerging as a potential communication structure that addresses the scalability issue using a packet based approach to communication. The network-on-chip provides IP cores the means to communicate with each other via a number of on-chip routers and point to point links. The work presented in this thesis focuses on potential techniques to improve the point to point links for network-on-chip. The four main areas investigated in this thesis have been:

(a) *Bit-Serial Compression using Unused Significant Bit Removal* has been explored for NoC links. The compression makes use of exploiting data which does not have rapidly changing bits in the most significant bit positions. Various data examples have been applied to the compression scheme and the reduction in data size given.

(b) *Asynchronous Serialized NoC Links* have been shown that wire reduction can be achieved through serialization and power reduced and clocking simplified through asynchronous techniques. Although there is extra overhead for the synchronous to asynchronous conversion circuitry and serializers the wire area savings are greater than the extra overhead, especially for longer wires.

(c) *Resilient Asynchronous Links* have been explored and a new data coding scheme has been demonstrated which has resilience to single event transient faults. The coding scheme uses a similar amount of wires as dual-rail techniques that are commonly used for asynchronous circuits but in addition give resilience to transient errors.

(d) Gate level design and experimental validation of the developed techniques have been carried out in order to verify the functionality and give indications of power, area and performance.

Section 6.1 summarises the main contributions made by the presented work and section 6.2 outlines possible future research directions which could further the work presented in this thesis.

6.1. Conclusions

Network-on-Chip is a current area of research that is gaining much attention in the system-on-chip world. The scalability offered by NoC for multi-core system means that NoC is rapidly gaining momentum as an alternative to traditional bus based communication mechanisms. The work presented in this thesis has focused on efficient NoC links using compression, serialization and asynchronous techniques as well as introducing resilience to transient errors.

Chapter 3 has shown a simple but effective compression scheme for bit-serial communication which can be applied to bit-serial NoC links. The compression technique exploits similarities in the most significant bits in consecutive pieces of data. The compression scheme has been shown to reduce the amount of data that is transmitted across the link down to 49% of the original uncompressed size for picture data. While there is also a reduction in the number of transitions the compression scheme can be combined with a transition reduction scheme [103] to reduce the number of transitions further. The compression scheme is suitable for data where the most significant bits change more than the least significant bits. In situations where the LSBs change more than the MSBs a transformation of the data could be performed to swap the LSBs and MSBs around so that the compression scheme could still be used. For random data compression does not work and can result in expansion of the amount of data. Furthermore an algorithm was developed to allow dynamic block sizing of the amount of data to be compressed if a fixed block size solution was not required.

Chapter 4 has proposed the use of asynchronous serialization of the NoC links as a means of reducing the number of wires and simplifying clocking. In our implementations the extra area overhead of the asynchronous circuitry compared to a typical synchronous link was 20%. However the number of wires was reduced by 75% from 32 wires down to 8. The reduction in number of wires also allowed smaller wires buffers to be used allowing a reduction in power of up to 65% when 8 wire buffers are used on the NoC link. There are further benefits of reducing the number of

wires in addition to reducing the wire area, the number of metal to metal layer vias and cross talk could also be reduced. Validation of the circuits were carried out on FPGA to allow functional testing on hardware. The FPGA implementation was slower that the full custom gate simulations but it did allow functional validation of the circuit proposed on hardware and works as expected in theory. Performance of the link should be improved if full custom IC implementation is used. This chapter has shown potential use of serialization in the asynchronous domain and circuit implementations. A general idea from this chapter is the use of synchronous routers operating in the synchronous domain interfacing to asynchronous links.

Chapter 5 has proposed a new data coding scheme for asynchronous NoC links. The coding scheme uses a similar number of wires per bit to existing schemes such as dual-rail but comes with the additional benefit of offering resilience to transient errors. The coding scheme has been shown to be resilient to transients although it can still be susceptible to a transient fault at the critical time when data is latched in the receiver and has been described and validated in simulation. Example circuits have been given to implement a link and could be further optimized to reduce the circuit size as the circuits presented were a first cut implementation to prove the functionality of the coding. It is hoped that this chapter has contributed a novel coding scheme that sits along side dual rail as a possible delay insensitive coding scheme with resilience to transient errors.

The developed techniques and experimental validations contributed towards the current efforts undertaken by academia and industry worldwide targeting the development of efficient and reliable on-chip communication for NoC infrastructure expected to be employed in future multiprocessor SoC. Such systems will be needed to offer the required performance in modern multimedia applications that have limited battery life and require reliability.

6.2. Future Research

Chapter 4 discussed serialized asynchronous NoC links and validation on FPGA. Chapter 5 provided a promising technique which increases the reliability of NoC links in the presence of soft-errors. As the coding was a new idea towards the end of the work there is further possibilities to build on this. Further research to build on the research in this thesis could consists of the following areas:

6.2.1 Custom ASIC Validation

The FPGA validation in Chapter 4 of the serialized asynchronous links provided functional validation of the link but the performance on FPGA was much slower. A full IC design using standard gates and a number of custom gates for cells such as C-Elements would provide functional validation and confidence of higher performance throughput. The IC could also incorporate a second copy of the NoC link which uses the transient resilient technique discussed in chapter 5 in order to validate the reliability on-chip.

6.2.2 Symbol Exploration

In chapter 5, the transient resilient links, the coding uses 2 wires per bit and 2 wires for the reference. The symbols can be 00, 01, 11 and 10. This allows detection of a single bit error on one of the pairs of wires but it cannot be corrected it since the hamming distance to a valid symbol is equally the same. For example if the reference is 00 and a symbol 00 is sent and a single bit error or transient affects one of the symbol wires 10 or 01 could be received. The symbol is invalid when comparing it to the reference as it is 90° out of phase, but is impossible to tell if it should be 00 or 11 since both of these are valid symbols, Fig. 6-1. In the proposed scheme the receiver circuit will stall the acknowledgement until the transient error goes away or if the transient hits at a certain point in time it could latch in invalid data.

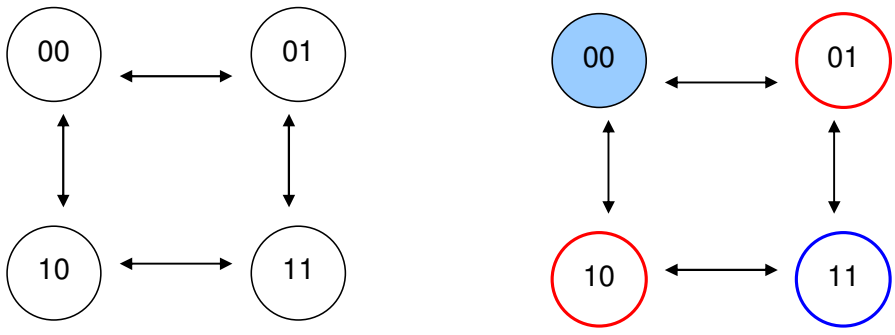


Fig. 6-1 Possible states for 2 wire symbols

One possible way around this would be to increase the number of wires for each symbol and increasing the hamming distance such that if a single bit error occurs the symbol is 'closer' in terms of hamming distance to one of the possible valid symbols. This would allow valid data to be received even if there are single bit errors present in the symbols. The negatives aspects of this would be the increased amount of wires and also the amount of transitions could increase as going from symbol to symbol could require more than 1 wire to change state to maintain hamming distance greater than 1. The resulting coding would effectively go from 4 possible symbols to 8. Consider a scheme using 3 wires per symbol which would give us 8 possible symbols to use. The scheme can be easily visualized as a cube with the states at each corner, Fig. 6-2. If symbol is in phase to the reference it will be the same. If the symbol is out of phase by 180° then the valid symbol will be on the opposite corner. If a reference of 111 is used and a symbol 111 is sent but there is a single bit error the received symbol will be 110, 101 or 011. As these three possible symbols in error are closer in hamming distance to 111 as opposed to 000 we can reasonably assume that the original uncorrupted symbol is likely to be 111.

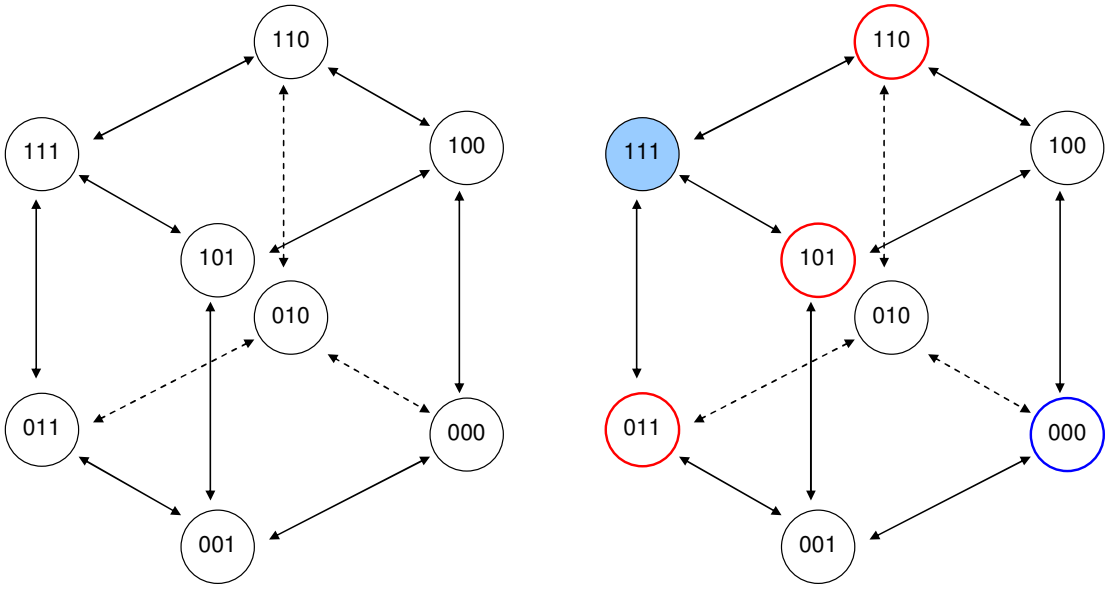


Fig. 6-2 Coding using 3 wires per Symbol

It is difficult to say how the complexity of the scheme would increase and the problems associated with increasing the number of symbol but this could be one possible future research direction for this particular topic. Certainly it can be seen that going from one symbol to the opposite side of the cube to represent a change in data would result in all 3 bits of the symbol changing.

6.2.3 Pair Wise Data and Reference

In the original scheme for the transient resilient links a single reference was partnered with several symbols. It may be advantageous to not have an exclusive reference signal but instead have two pair-wise symbols and use themselves to detect validity. Consider Fig. 6-3. A master symbol and a slave symbol (prefixed with M and S respectively in the figure) is used to represent two bits of data. The master symbol would rotate 90° anti-clockwise to signify a 1 and 90° clockwise to signify a 0 for the first data bit. The slave would either be in-phase to signify the second bit is the same or 180° out of phase to signify the second bit is the opposite. In the example shown in Fig. 6-3 the master symbol (M_SYMBOL) rotates anti-clockwise from 00 to 01 signifying the first bit (DATA[0])is a 1 and the slave symbol (S_SYMBOL) is in-phase with it so the second data bit (DATA[1]) is the same and therefore is also a 1. So the resulting data is 00. Next the master symbol rotates anti-clockwise from 01 to 11 and the slave symbol is 180° out of phase so the data is 01. Finally the master symbol rotates clockwise so the first data bit (DATA[0]) is now a 0, and the slave is in-phase, so the data is 00.

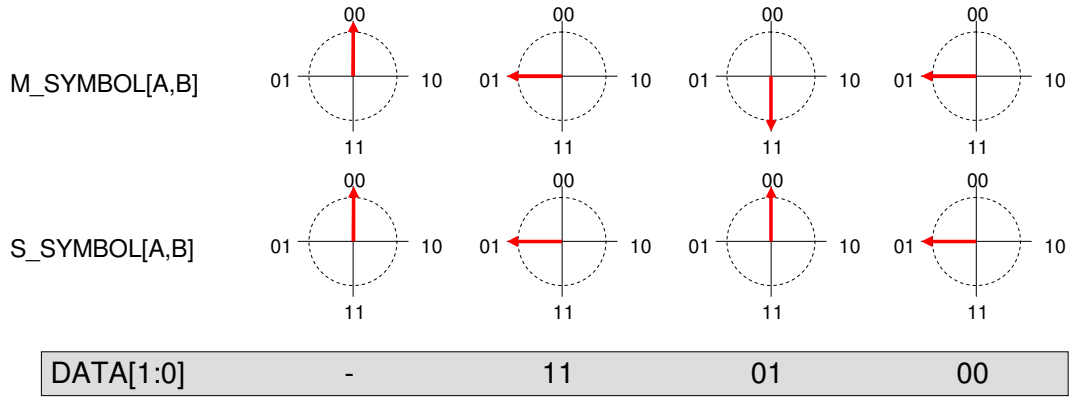


Fig. 6-3 Pair-Wise Symbols

This pair-wise scheme removes the need for an explicit reference and several of these pair-wise groups can be used together for data several bits wide However it may increase the complexity to the circuitry since the direction of rotation needs to be monitored to extract the data. The validity is still kept intact as both symbols are still in-phase or 180° out of phase with each other and as such single bit errors can still be detected. However, the single bit error can only occur on one of the wires in the pair-wise group. Further research is necessary to assess the impact on circuit complexity and robustness of using such a scheme.

Appendix A VHDL Modules for Compression

An overview of the VHDL modules and there connectivity to the test bench for the standard bit-serial link and the bit serial link using the proposed compression scheme is given in Fig. A-1 and Fig. A-2 respectively.

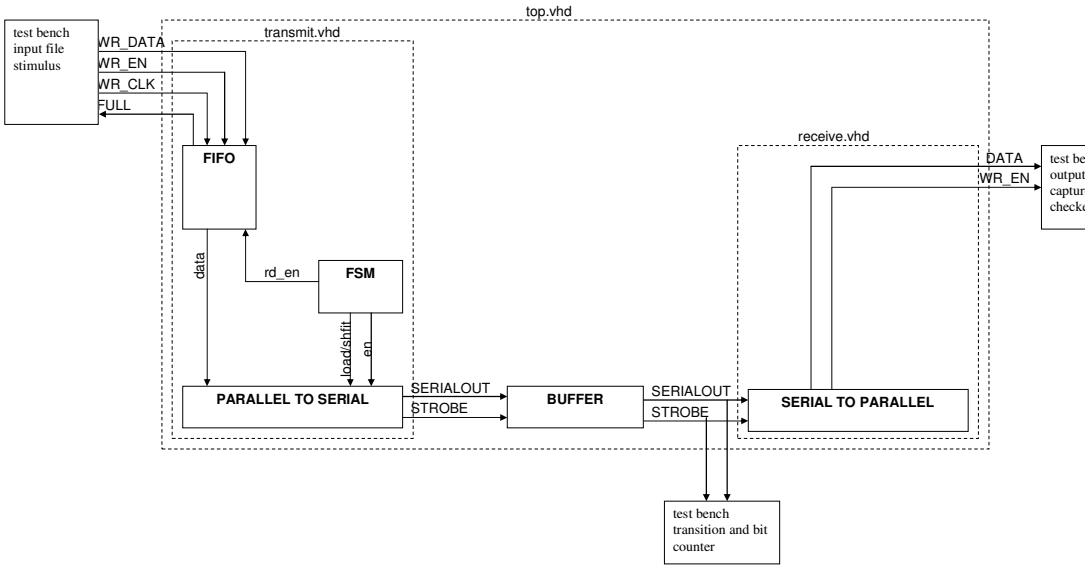


Fig. A-1 Top level RTL Serial Link

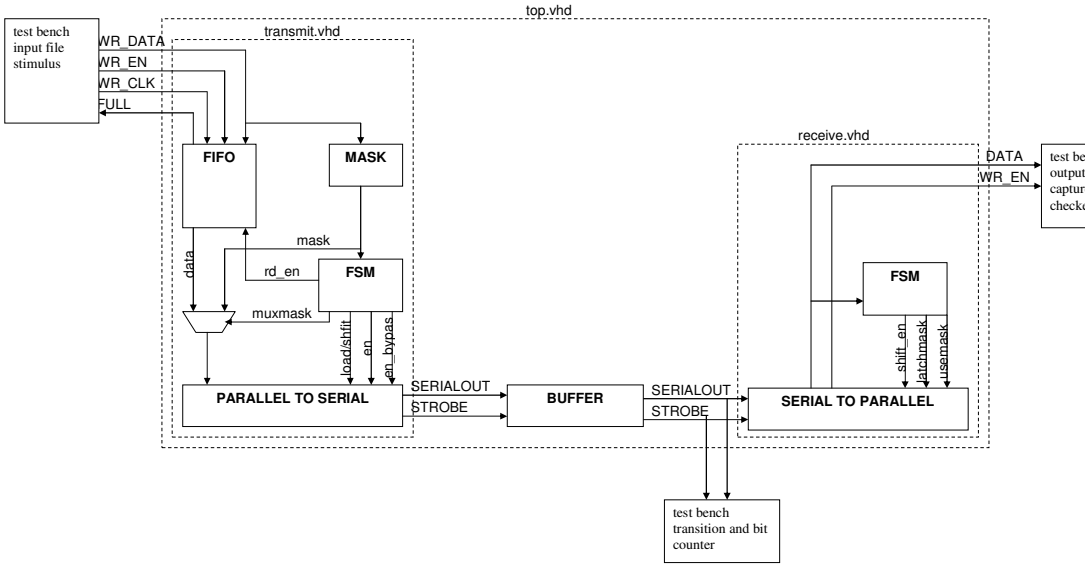


Fig. A-2 Top Level RTL partitioning, USBR

Appendix B MPEG Background Information

This section provides background details on MPEG decoding which is used as the case study for the results in Chapter 3. It is not intended as a complete description of MPEG but just enough to justify the concept of frame or picture buffering in a MPEG decoder. Further information on MPEG video decoding can be found in ISO 11172-2 and ISO 13818-2.

MPEG compression has become the current de-facto standard in digital video storage. MPEG was developed to allow compressed video to be stored on digital formats such as compact discs, digital versatile disc (DVDs) and hard drives. MPEG utilises a block based compression scheme. The picture is divided up into a number of macro-blocks horizontally and vertically. The information which makes up these macro-blocks are then pushed through cosine transformation to turn the data into a frequency representation of the image, just like a Fourier transform turns a signal in to its frequency components. The transformed data is then quantised to try and zero out a lot of the high frequency information which our eyes do not really notice. The quantized data is then compressed using Variable length coding (Huffman) since the quantised data will probably contain long runs of 0's in the high frequency areas.

The MPEG video stream has a layered structure consisting of several layers. The highest layer is a video sequence and sequence header. The next layer down is a group of pictures followed by picture, slice, macro-block and finally the block layer. The block layer is the lowest layer which is effectively an 8x8 data matrix which represents a small portion of the visual image. Fig. B-1 shows an overview of the layered bit-stream in more detail. Each layer effectively wraps around the next lower level layer and contains information and data applicable to the current layer.

Complex system on chip devices are integrated more and more cores onto a single die. These multi-core solutions in System on Chip and Network on Chip the data that cores work with is often non-random. Certain application such as video decoding will decode the picture data from the run length encoding bit-stream and then need to store the picture data in memory to allow out of order picture decoding. It is preferable to make sure that data stored in memory should be byte aligned. Byte alignment helps keep coherency between other devices which share the same memory and access the same data. In MPEG applications after the Huffman decoding and

IDCT the picture data will be buffered in memory since the decoded picture order will be different to the displayed order and certain picture types require past and future references.

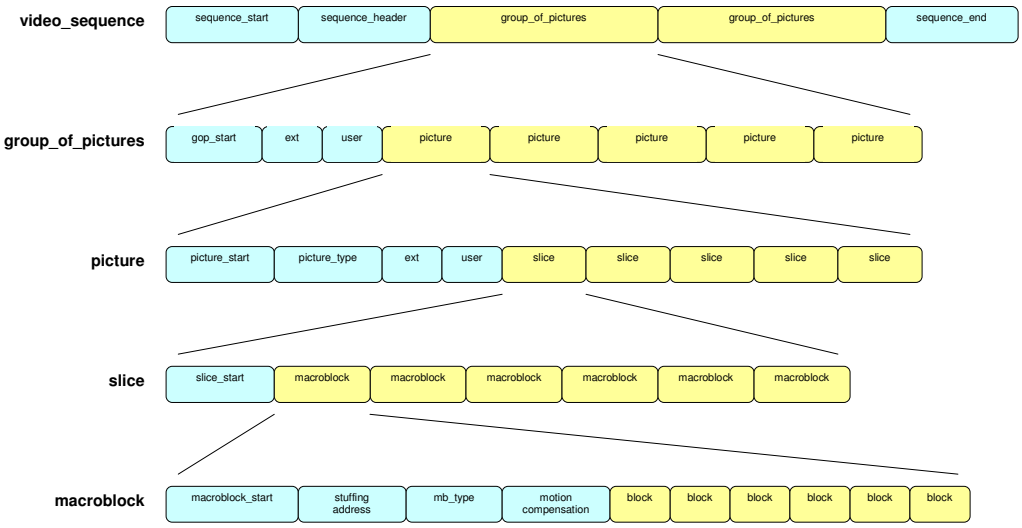


Fig. B-1 Layered Hierarchy of the Video Sequence

Typically in a MPEG video stream you will have Intra (I), Predicted (P), and Bi-directional (B) coded pictures.

Decoded order: I – P – B – B – B – P – B – B – B

Displayed order: I – B – B – B – P – B – B – B – P

Assume the data is stored in the YUV colour space format[158] and the data are values from 0 to 255. Using 10 bits to store fixed point data with this magnitude should give ±1 LSB after the colour space transformation to 8 bit RGB colour space. For decoding and subsequent displaying there should be enough memory for at least 3 pictures worth of data. Memory would be needed for Past, Future and Current. If we also assume we need at least 3 pictures worth of buffer space then we can calculate how much memory we need to have for the decoding process. For each macro-block we require 6 * 64 * 16 = 6144 bits. A typical MPEG1 picture is 330 macro-blocks. MPEG1 therefore needs 330 * 6144 bits ≈ 2 Mbit per picture, so to buffer 3 pictures we need approximately 6 Mbit for MPEG1. As can be seen a large amount of MPEG decoding is the buffering of data into memory. If 10 bit signed data is stored in a byte aligned memory space then each data value will occupy 16 bits. Clearly the MSBs will not be used and are somewhat redundant. Also because visual pictures do not have rapidly changing colour information the data can often be similar within a block.

Fig. B-2 shows the data flow, in an Intra-coded picture the data will come from the IDCT and get written directly to the buffer since all the macro-blocks will be intra-coded. In predicted and bi-directional pictures the data could directly get written into memory if the block is intra-coded or get passed to a motion compensation unit where motion vectors are used to obtain a prediction macro-block which results in the reading of data from the past or future picture buffers which get added and the result stored back into a buffer. From this it can be seen that the buffering of pictures represents the most substantial bandwidth usage in a video decoding system. Anything that can be done to reduce power or improve bandwidth in this application would be a great benefit. To reduce the number of bits or transitions some sort of encoding at the source and decoding at the destination must be done. Provided that the power used to encode and decode the data is less than the power saved in transferring the encoded data through the communication path then an overall power saving is achieved. The encoding could be profiled for different applications. For instance, the data that is transferred from an IDCT core to a memory, the four luminance blocks in a macro-block would be very similar if there is not much change within that macro-block.

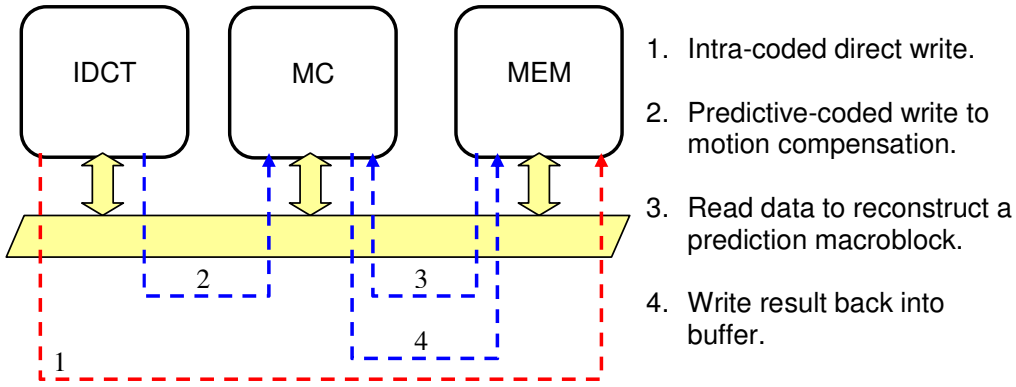


Fig. B-2 Part of the MPEG Video Decoding Structure

It can be seen in Fig. B-3 that the blocks with the luminance information have a low of changing data since the outline of the bottom of the tyre can be seen clearly and is in contrast to the background. The lower luminance blocks also show the top of the wall which is in contrast too. Fig. B-4 shows the macro-block for part of the wall. Since there are no sharply contrasting features in this macro-block the data values do not change that much either within the block or from block to block. It can be seen that in an application such as MPEG there are a reasonable amount of redundant bits in macro-blocks, infact that is what MPEG coding is used for, to compress video

information. However, when the variable length decoding and IDCT operations have been performed and the data is being moved around to different cores and memory the data is now longer in a compressed format. More likely it will be stored as values representing the YUV colour information. Usually this reading and writing to memory is over some form of communication link. If a serial link is used then the compression technique that removes redundant bits in a block could be used.

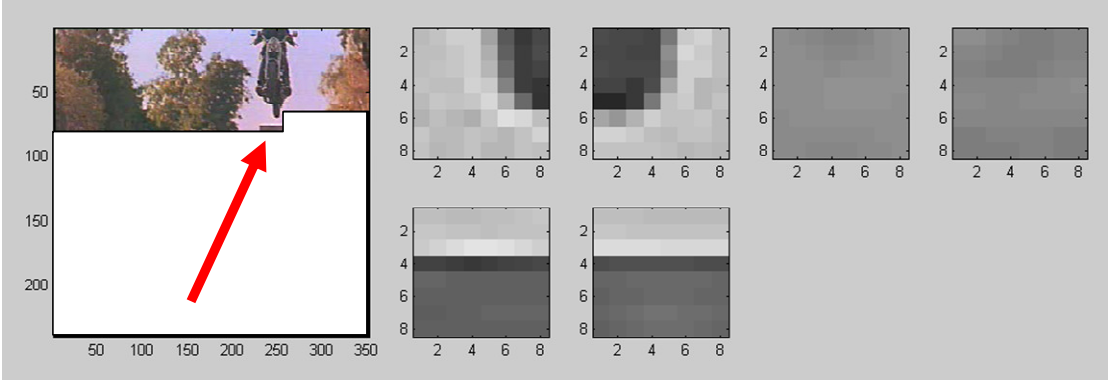


Fig. B-3 Luminance and Chrominance Blocks of Macro-block, tyre

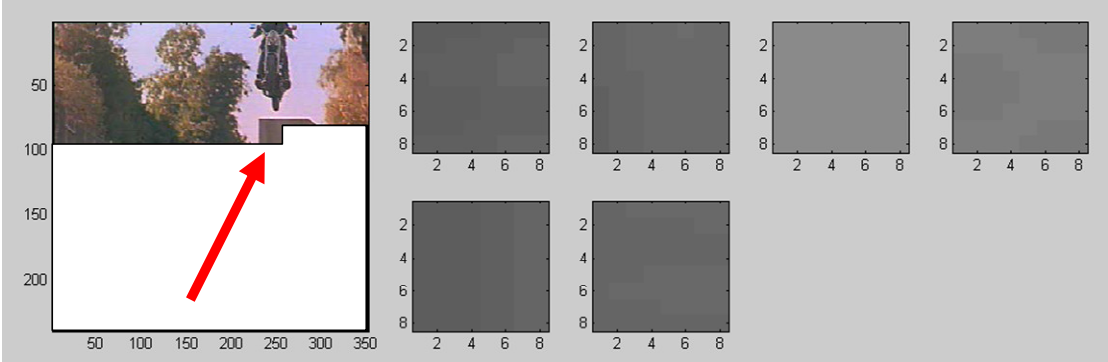


Fig. B-4 Luminance and Chrominance Blocks of Macro-block, Wall

Since the data in MPEG blocks is often similar for macro-blocks that do not contain a lot of detail then there is some argument for the use of a simple compression technique when data is being read and written to memory. Furthermore if the data precision is less than the byte alignment used in memory then gains can be had for all blocks since there will be redundant MSBs anyway.

An example of byte alignment is shown in Fig. B-5. Assume the MPEG block data is 10 bit unsigned data. This means the data range will be 0x000 to 0x3FF. However, data should always be aligned to the nearest byte boundary. One could argue that the data could be packed together bitwise as shown in Fig. B-5(B), but this makes data transfer more difficult since processors and DMAs usually operate on byte addressing. The high cost of unpacking and repacking the data bitwise in software

would be prohibitive. A software engineer would simply want to read data then use it. If bit packing was used the software would have to start performing shifts and logical bitwise operations in order to get the data into a format it could be used in which burns MIPS and is therefore generally unacceptable.

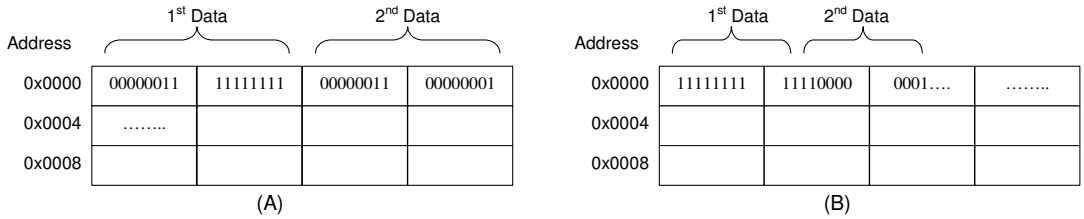


Fig. B-5 Byte aligned and bit-packed for 10 bit numbers stored in memory

Appendix C Reducing Wire Delay

This section explains the reasoning why buffered pipelined wires are used in NoC type applications where point-to-point links may travel relatively long distances across chip. Wire delay does not grow linearly with length, it grows squarely as shown in Fig. C-1. This means the wire delay for long global interconnect may be considerable.

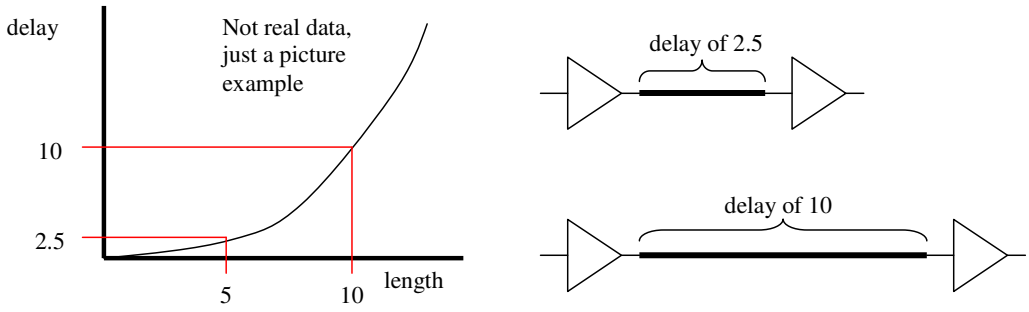


Fig. C-1 Wire Delay

Long wires present a problem, a solution to this is to segment the long wire with intermediate buffers if the a gate delay of a buffer is relatively small compared to the wire delay. Lets say a buffer has a delay of 1. Then by inserting the buffer into a wire of length 10 we split the length of wire into two 5 length segments separated by a buffer as shown in Fig. C-2. The segmentation of the wire helps linearise the delay with respect to the length.

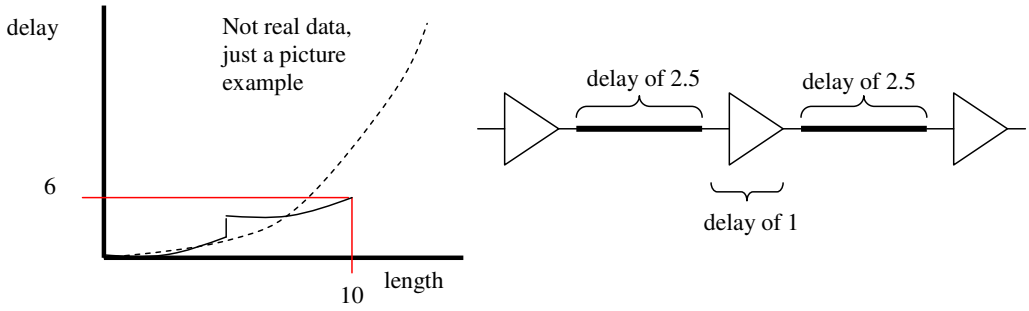


Fig. C-2 Wire Delay (Buffered)

It is perhaps useful to see why segmented or buffered wires are used in the point-to-point links of network-on-chip in order to see a direction on how to further improve. If we take a simple request acknowledge cycle, ignoring things like effects of crosstalk on delay for the moment, we can see a very basic but typical asynchronous cycle, Fig. C-3. At the bare minimum the cycle time in order for an asynchronous transmitter is based on the time it takes for data (and possibly bundled

control) to arrive at the receiver. The receiver then has to say when this is valid and finally send an acknowledgement back to the transmitter to say it is okay to go onto the next transfer. The transfer cycle needs one complete transfer sent and acknowledgement received. If the wire length is long then the wire delay would dominate the cycle time.

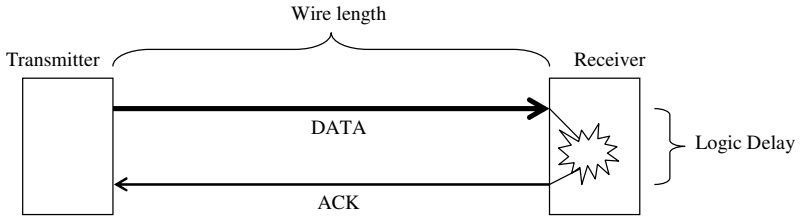


Fig. C-3 Basic Asynchronous Cycle

The wire delay can be alleviated by adding simple buffers along the length of the wire, Fig. C-4. Since the shorter wire segments with intermediate buffers can reduce wiring delay dramatically the request / acknowledge cycle time would be reduced.

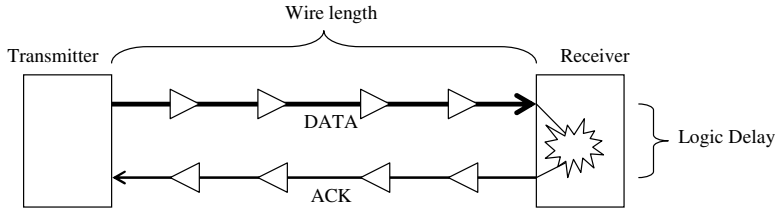


Fig. C-4 Buffered Wires

It is possible to use latched buffers along the length of the wire, Fig. C-5. This is what we have done in our asynchronous solution in Chapter 4. With this approach the cycle time is based on the data sent to the first registered buffer and the acknowledgement back. The wire is also pipelined, meaning that different data items can occupy different stages along the wire which should give an increase in throughput compared to non-registered wire buffers.

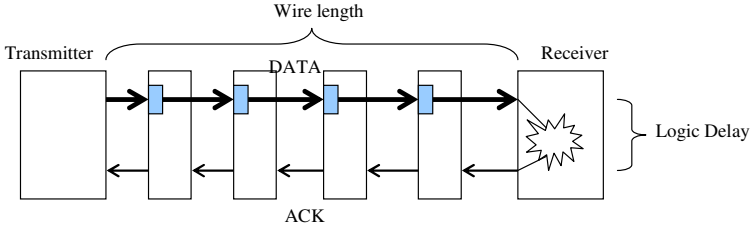


Fig. C-5 Registered Buffered Wire

Appendix D FPGA Design Flow

Synchronous

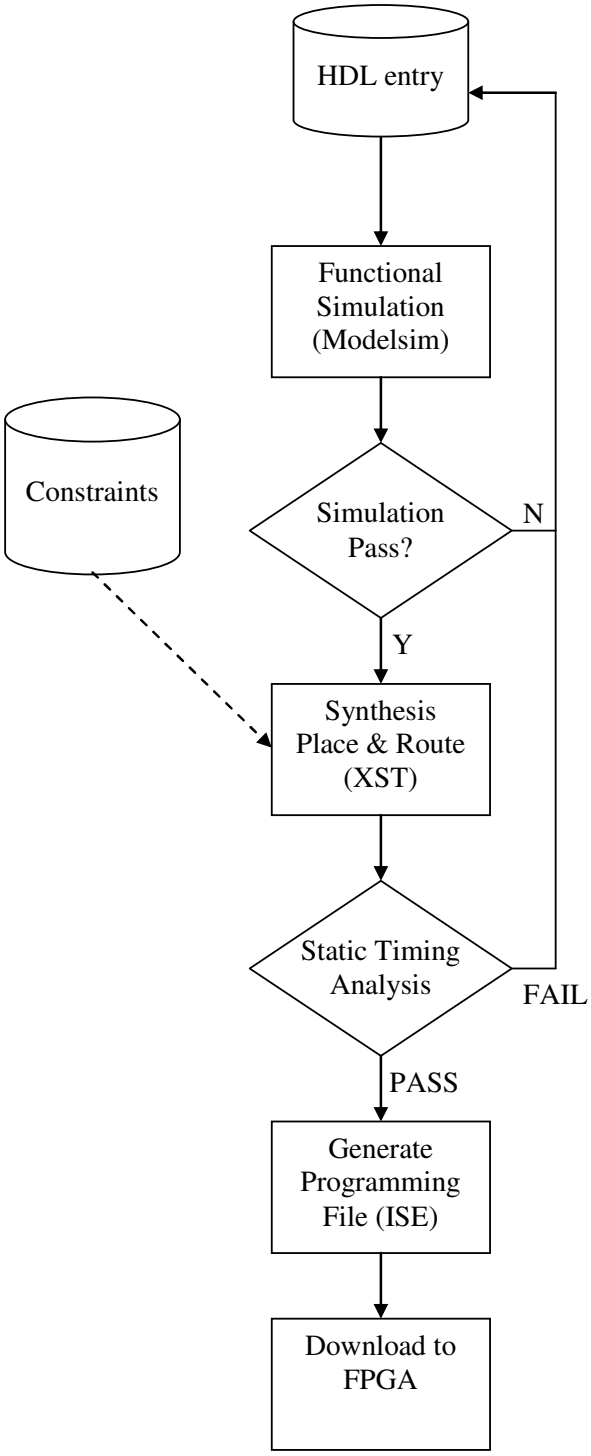


Fig. D-1 Synchronous FPGA Design Flow

Asynchronous

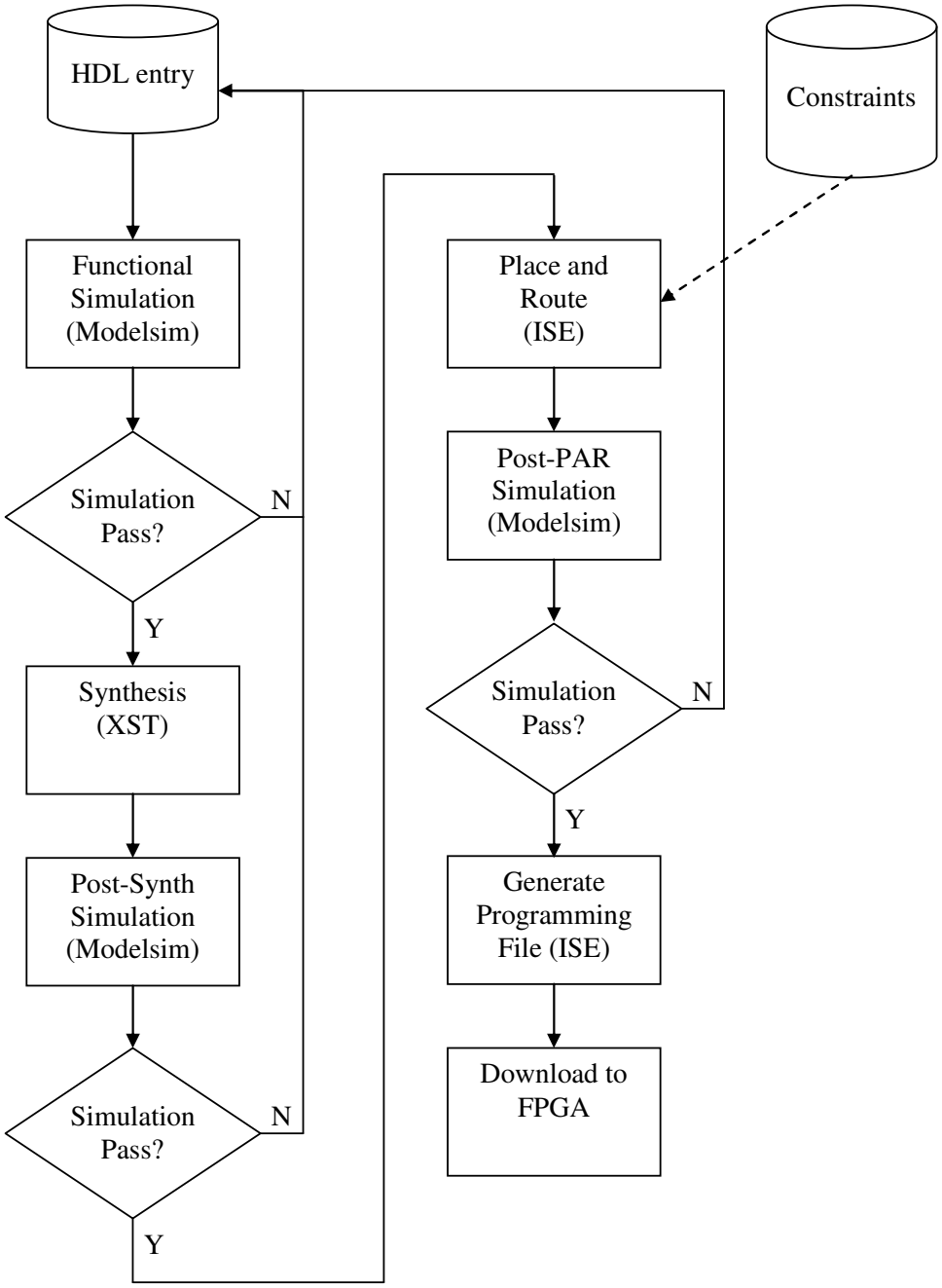


Fig. D-2 Asynchronous FPGA Design Flow

Appendix E Area Estimation of Phase Encoding

Estimation of the area for Multiple-Rail phase encoding is not easily done. This section estimate the gates for 1 bit, 8 bit and 16 bit wide implementations of a multiple rail phase encoded link and should be read in conjunction with section VII of the work in [114]. The link is divided into four main parts; matrix encoder, delay array, mutex array and decoding array, Fig. E-1. The matrix encoder takes the input data and then generates outputs which correspond to which tri-state buffer should be enable on each wire to drive it such that the correct delay is introduced. The delay array consist of an array of tri-state buffers and delay elements (which we assume are two invertors) which provides the phase delay. The mutex array arbitrates the order of the received signals on the wires by comparing the arrival time of the edges of the signals. The decoding array take the arrival time information and provides the necessary data output. The m bit wide data is sent over a link with n wires. The relationship between the number of bits and the number of wires needed to support the necessary amount of symbols is:

$(n-1)! < 2^m < n!$, where m is the data bit width.

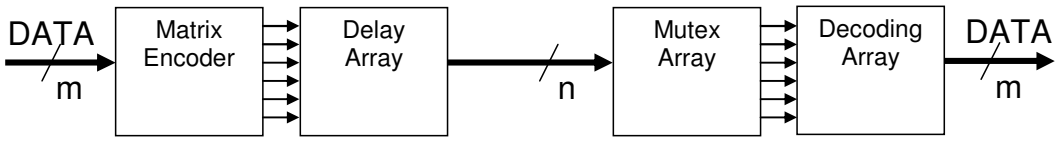


Fig. E-1 Multiple Rail Phase Encoding Link

For 1 bit wide data the link will use 2 wires and can be implemented with the gates shown in Fig. E-2.

Matrix Encoder

Matrix Encoder Outputs

DATA	*11 12*	*21 22*
0	1 0	0 1
1	0 1	1 0

2x INV

Mutex Array

2x C-Element
2x AND2
3x OR2
2x NAND2
2x INV

Delay Array

4x INV
4x TBUF

Decoding Array

nil

Fig. E-2 Gate Count for 1 Bit Wide M-Rail phase encoding

For 8 bit wide data the number of wires is 6 and the gate estimate is more difficult, some discussion about estimation for the link is as follows:

Matrix Encoder

Only a small section of the matrix encoder is shown in Fig. E-3 since there are far too many entries to feasibly show.

	Matrix Encoder Outputs															
DATA	*11*	*12*	*13*	*14*	*15*	*16*	*21*	*22*	*23*	*24*	*25*	*26*	*31*	*32*	…	*66*
00000000	1	0	0	0	0	0	0	1	0	0	0	0	0	0	…	
00000001	1	0	0	0	0	0	0	0	1	0	0	0	0	0	…	
00000010	1	0	0	0	0	0	0	0	0	1	0	0	0	0	…	
00000011	0	1	0	0	0	0	1	0	0	0	0	0	0	0	…	
…	…						…						…			

Fig. E-3 Partial Matrix Encoder

We can make some estimates based on the fact that we know that each group of outputs (11-16, 21-26, … 61-66) will only have one output active which controls it's associated tri-state buffer. Lets us assume that each output is active for 1/6 of the amount DATA inputs, so 1/6 of the time wire 1 will switch first, another 1/6 of the time wire 2 will switch first etc… Each matrix encoder output can be realised as a sum of products.

Each product term would be the 8 data inputs ANDed together with four of the DATA inputs inverted on average. If each matrix encoder output is active for 1/6 of the 256 possible DATA inputs then we have 256/6 = 43 sums of 8 product terms per matrix encoder output. The sum of products could be optimized by logic reduction to get the smallest solution, however without knowing the actual Boolean equations it is difficult to say the smallest solution size. Let us assume we can on average shrink the sum of products to a more manageable 8 sums of 4 products, with each product using 2 invertors and 6x6 = 36 matrix encoder outputs the cost of the matrix encoder would be:

36x (8x AND4)
36x (16x INV)
36x (2x OR4)
36x (1x OR2)

Delay Array

Looking at Fig. 9 in [114] it is clear that for a 6 wire solution there would need to be 36 tri-state buffers and 5x5 = 25 delay elements, assuming we use 2x INV for the delay elements the cost is:

36x TBUF
50x INV

Mutex Array

The mutex array is now just a 6x2 array of enhanced mutex elements, from section VII(B) of [114] the cost is:

12x (2x C-Element)
12x (2x AND2)
12x (3x OR2)
12x (2x NAND2)
12x (2x INV)

Decoding Array

Looking at the structure in Fig. 10(b) in [114] we can need a 7 layer logic circuit for each DATA word output. The 7 layer logic circuit can be realise with 8x AND4 and 6x AND2 gates. However, this has to be replicated 256 times, one for each DATA word output. The cost is:

256x (8x AND4)
256x (6x AND2)

Totals

The area totals are shown in Fig. E-4. A 16 bit wide link is not estimated since the size will spiral out of control for the decoding array.

Impl.	Matrix Encoder	Delay Array	Mutex Array	Decoding Array
1 bit	2x INV	4x INV 4x TBUF	2x C-Element 2x AND2 3x OR2 2x NAND2 2x INV	-
Area	2x 6.052	4x 6.052 4x 10.086	2x 22.189 2x 10.086 3x 10.086 2x 6.052 2x 6.052	-
Area Totals	12.104	64.552	119.016	-
8 bit	36x (8x AND4) 36x (16x INV) 36x (2x OR4) 36x (1x OR2)	50x INV 36x TBUF	12x (2x C-Element) 12x (2x AND2) 12x (3x OR2) 12x (2x NAND2) 12x (2x INV)	256x (8x AND4) 256x (6x AND2)
Area	36x (8x 14.120) 36x (16x 6.052) 36x (2x 14.120) 36x (1x 10.086)	50x 6.052 36x 10.086	12x(2x 22.189) 12x (2x 10.086) 12x (3x 10.086) 12x (2x 6.052) 12x (2x 6.052)	256x (8x 14.120) 256x (6x 10.086)
Area Totals	8932.248	665.696	1428.192	44409.856
16 bit	-	-	-	-
Area	-	-	-	-

Fig. E-4 Gate and Area Cost for 1 bit and 8 bit M-Rail Phase Encoding

References

[1] System on Chip, http://en.wikipedia.org/wiki/System-on-a-chip

[2] A. Deshpande, "Verification of IP-Core Based SoC's," in *Quality Electronic Design, 2008. ISQED 2008. 9th International Symposium on*, 2008, pp. 433-436.

[3] Fujitsu HDTV SoC, http://www.fujitsu.com/global/services/microelectronics/product/assp/video/index_p2.html

[4] C. C. P. Chen and E. Cheng, "Future SoC design challenges and solutions," in *Quality Electronic Design, 2002. Proceedings. International Symposium on*, 2002, pp. 534-537.

[5] ITRS, "International Technology Roadmap for Semiconductors," vol. Design, 2007.

[6] H. Ron, M. Ken, and M. Horowitz, "Managing wire scaling: a circuit perspective," in *Interconnect Technology Conference, 2003. Proceedings of the IEEE 2003 International*, 2003, pp. 177-179.

[7] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *Micro, IEEE,* vol. 23, pp. 14-19, 2003.

[8] AMBA System Architecture, http://www.arm.com/products/solutions/AMBAHomePage.html

[9] OCP IP Specification 2.2, http://www.ocpip.org/home

[10] CoreConnect Bus Architecture Product Brief, http://www.ibm.com/chips/products/coreconnect

[11] Sonics Inc., http://www.sonicsinc.com

[12] Wishbone, Rev B.3 Specification, 2002, http://www.opencores.org/

[13] Buses Presentation, 2004, www.imit.kth.se/courses/2B1447/Lectures/2B1447_L4_Buses.pdf

[14] Bus Contention, http://en.wikipedia.org/wiki/Bus_contention

[15] C. A. Zeferino and A. A. Susin, "SoCIN: a parametric and scalable network-on-chip," Sao Paulo, Brazil, 2003, pp. 169-74.

[16] C. Sangik and K. Shinwook, "Implementation of an on-chip bus bridge between heterogeneous buses with different clock frequencies," in *System-on-Chip for Real-Time Applications, 2005. Proceedings. Fifth International Workshop on*, 2005, pp. 530-534.

[17] Advanced Microprocessor Bus Archiecture (AMBA) Bus System, http://www.elecdesign.com/Articles/Index.cfm?AD=1&ArticleID=4165

[18] First Details of AMBA Presentation, http://www.jp.arm.com/kk/arm_forum2003/ppt/first_details_of_amba.ppt

[19] P. J. Aldworth, "System-on-a-chip bus architecture for embedded applications," *Proceedings - IEEE International Conference on Computer Design: VLSI in Computers and Processors,* pp. 297-298, 1999.

[20] On-Chip Split Bus Versus Switching Interconnects 2006, http://www.elecdesign.com/Articles/ArticleID/13983/13983.html

[21] P. Wijetunga, "High-performance crossbar design for system-on-chip," Calgary, Alta., Canada, 2003, pp. 138-43.

[22] K. Donghyun, L. Kangmin, L. Se-joong, and Y. Hoi-Jun, "A reconfigurable crossbar switch with adaptive bandwidth control for networks-on-chip," Kobe, Japan, 2005, pp. 2369-72.

[23] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *Computer,* vol. 35, pp. 70-8, 2002.

[24] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," Las Vegas, NV, USA, 2001, pp. 684-9.

[25] S. Murali and G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," San Diego, CA, USA, 2004, pp. 914-19.

[26] L. Se-Joong, S. Seong-Jun, L. Kangmin, W. Jeong-Ho, K. Sung-Eun, N. Byeong-Gyu, and Y. Hoi-Jun, "An 800MHz star-connected on-chip network for application to systems on a chip," San Francisco, CA, USA, 2003, pp. 468-9.

[27] A. Radulescu, J. Dielissen, K. Goossens, E. Rijpkema, and P. Wielage, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, pp. 878-883 Vol.2.

[28] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine,* vol. 4, pp. 18-31, 2004.

[29] D. A. P. John L. Hennessy, David Goldberg, Krste Asanovic, "Interconnect Networks and Clusters," in *Computer Architecture* 3rd ed: Morgan Kaufmann, p. 811.

[30] X. Zhu, Y. Cao, and L. Wang, "A Novel Routing Algorithm for Network-on-Chip," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 1877-1879.

[31] N. Kavaldjiev, G. J. M. Smit, and P. G. Jansen, "A virtual channel router for on-chip networks," in *SOC Conference, 2004. Proceedings. IEEE International*, 2004, pp. 289-293.

[32] C. A. Zeferino, M. E. Kreutz, and A. A. Susin, "RASoC: a router soft-core for networks-on-chip," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, pp. 198-203 Vol.3.

[33] A. Hosseini, T. Ragheb, and Y. Massoud, "A fault-aware dynamic routing algorithm for on-chip networks," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 2008, pp. 2653-2656.

[34] M. Dehyadgari, M. Nickray, A. Afzali-kusha, and Z. Navabi, "Evaluation of pseudo adaptive XY routing using an object oriented model for NOC," in *Microelectronics, 2005. ICM 2005. The 17th International Conference on*, 2005, p. 5 pp.

[35] H. Jingcao and R. Marculescu, "DyAD - smart routing for networks-on-chip," in *Design Automation Conference, 2004. Proceedings. 41st*, 2004, pp. 260-263.

[36] Routing Table Minimization for Irregular Mesh NoCs, http://www.ee.technion.ac.il/bolotin/papers/routing_date2007.pdf

[37] D. Andreasson and S. Kumar, "Slack-time aware routing in NoC systems," Kobe, Japan, 2005, pp. 2353-6.

[38] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Improving wormhole adaptive routing in networks on chip," *WSEAS Transactions on Computers,* vol. 5, pp. 544-51, 2006.

[39] M. Dehyadgari, M. Nickray, A. Afzali-kusha, and Z. Navabi, "Evaluation of pseudo adaptive XY routing using an object oriented model for NOC," Islamabad, Pakistan, 2006, p. 5 pp.

[40] R. Holsmark, M. Palesi, and S. Kumar, "Deadlock free routing algorithms for mesh topology NoC systems with regions," Dubrovnik, Croatia, 2006, p. 8 pp.

[41] N. Huy-Nam, N. Vu-Duc, and C. Hae-Wook, "Assessing routing behavior on on-chip-network," Cairo, Egypt, 2006, pp. 62-5.

[42] L. Il-Gu, L. Jin, and P. Sin-Chong, "Adaptive routing scheme for NoC communication architecture," Phoenix Park, South Korea, 2005, pp. 1180-4.

[43] H. Jingcao and R. Marculescu, "DyAD-smart routing for networks-on-chip," San Diego, CA, USA, 2004, pp. 260-3.

[44] O. Tayan and D. Harle, "A Manhattan street network implementation for networks on chip," Damascus, Syrian Arab Republic, 2004, pp. 683-684.

[45] T. Marescaux, B. Bricke, P. Debacker, V. Nollet, and H. Corporaal, "Dynamic time-slot allocation for QoS enabled networks on chip," in *Embedded Systems for Real-Time Multimedia, 2005. 3rd Workshop on*, 2005, pp. 47-52.

[46] Interconnect-Centric Design for Advanced NoC and SoC, 2004, www.tkt.cs.tut.fi/kurssit/8404941/S04/chapter9.ppt

[47] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal network on chip: concepts, architectures, and implementations," *IEEE Design & Test of Computers,* vol. 22, pp. 414-21, 2005.

[48] S. Mitra, P. Sanda, and N. Seifert, "Soft Errors: Technology Trends, System Effects, and Protection Techniques," in *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International*, 2007, pp. 4-4.

[49] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on,* vol. 5, pp. 305-316, 2005.

[50] S. S. Mukherjee, J. Emer, and S. K. Reinhardt, "The soft error problem: an architectural perspective," in *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, 2005, pp. 243-247.

[51] S. Mitra, Z. Ming, T. M. Mak, N. Seifert, V. Zia, and K. Kee Sup, "Logic soft errors: a major barrier to robust platform design," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*, 2005, p. 10 pp.

[52] Soft Errors, http://en.wikipedia.org/wiki/Soft_error

[53] Teraflops Research Chip Overview, http://techresearch.intel.com/articles/Tera-Scale/1449.htm

[54] Arteris, http://www.arteris.com/

[55] L. Benini, D. Bruni, A. Macii, and E. Macii, "Memory energy minimization by data compression: algorithms, architectures and implementation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 12, pp. 255-68, 2004.

[56] H. Lekatsas, J. Henkel, V. Jakkula, and S. Chakradhar, "A unified architecture for adaptive compression of data and code on embedded systems," Kolkata, India, 2005, pp. 117-123.

[57] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "SPIN: a scalable, packet switched, on-chip micro-network," in *DATE 2003*, Munich, Germany, 2003, pp. 70-3.

[58] D. Wiklund and L. Dake, "SoCBUS: switched network on chip for hard real time embedded systems," in *IPDPS 2003*, Nice, France, 2003, p. 8 pp.

[59] E. Dupont, E. Dupont, M. Nicolaidis, and P. Rohr, "Embedded robustness IPs for transient-error-free ICs," *Design & Test of Computers, IEEE,* vol. 19, pp. 54-68, 2002.

[60] P. L. Teijo Lehtonen, and Juha Plosila, "Online Reconfigurable Self-Timed Links for Fault Tolerant NoC," *VLSI Design,* vol. 2007, 2007.

[61] K. Lahiri, A. Raghunathan, and S. Dey, "Efficient exploration of the SoC communication architecture design space," San Jose, CA, USA, 2000, pp. 424-430.

[62] K. Lahiri, A. Raghunathan, and S. Dey, "System-level performance analysis for designing on-chip communication architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 20, pp. 768-83, 2001.

[63] K. K. Ryu and V. J. Mooney Iii, "Automated bus generation for multiprocessor SoC design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, pp. 1531-1549, 2004.

[64] K. Lahiri, A. Raghunathan, G. Lakshminarayana, and S. Dey, "Design of high-performance system-on-chips using communication architecture tuners," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, pp. 620-36, 2004.

[65] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 3, pp. 49-58, 1995.

[66] M. R. Stan and W. P. Burleson, "Low-power encodings for global communication in CMOS VLSI," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 5, pp. 444-55, 1997.

[67] E. Musoll, T. Lang, and J. Cortadella, "Working-zone encoding for reducing the energy in microprocessor address buses," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 6, pp. 568-572, 1998.

[68] S. Osborne, A. T. Erdogan, T. Arslan, and D. Robinson, "Bus encoding architecture for low-power implementation of an AMBA-based SoC platform," *IEE Proceedings: Computers and Digital Techniques,* vol. 149, pp. 152-156, 2002.

[69] M. Lampropoulos, B. M. Al-Hashimi, and P. Rosinger, "Minimization of crosstalk noise, delay and power using a modified bus invert technique," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 2004, pp. 1372-1373 Vol.2.

[70] Y. Aghaghiri, F. Fallah, and M. Pedram, "Transition reduction in memory buses using sector-based encoding techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 23, pp. 1164-1174, 2004.

[71] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: The energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, pp. 818-831, 2005.

[72] T. Lang, E. Musoll, and J. Cortadella, "Extension of the working-zone-encoding method to reduce the energy on the microprocessor data bus," in *Computer Design: VLSI in Computers and Processors, 1998. ICCD '98. Proceedings. International Conference on*, 1998, pp. 414-419.

[73] H. Jiun-Sheng, T. Shang-Wei, and J. Jing-Yang, "On-chip bus encoding for LC cross-talk reduction," in *VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT). 2005 IEEE VLSI-TSA International Symposium on*, 2005, pp. 233-236.

[74] T. Lv, J. Henkel, H. Lekatsas, and W. Wolf, "A Dictionary-Based En/Decoding Scheme for Low-Power Data Buses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 11, pp. 943-951, 2003.

[75] T. A. Bartic, J. Y. Mignolet, V. Nollet, T. Marescaux, D. Verkest, S. Vernalde, and R. Lauwereins, "Highly scalable network on chip for reconfigurable systems," Tampere, Finland, 2003, pp. 79-82.

[76] L. Benini and D. Bertozzi, "Network-on-chip architectures and design methods," *IEE Proceedings-Computers and Digital Techniques,* vol. 152, pp. 261-72, 2005.

[77] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: A novel on-chip communication network," Tampere, Finland, 2004, p. 15.

[78] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "xpipes: A latency insensitive parameterized network-on-chip architecture for multi-processor SoCs," San Jose, CA, United States, 2003, pp. 536-539.

[79] D. Siguenza-Tortosa, T. Ahonen, and J. Nurmi, "Issues in the development of a practical NoC: The Proteo concept," *Integration, the VLSI Journal,* vol. 38, pp. 95-105, 2004.

[80] D. Siguenza-Tortosa and J. Nurmi, "Proteo: a new approach to network-on-chip," in *IASTED Conference on Communication Systems and Networks*, Malaga, Spain, 2002, pp. 355-9.

[81] R. Mullins, A. West, and S. Moore, "The design and implementation of a low-latency on-chip network," in *Design Automation, 2006. Asia and South Pacific Conference on*, 2006, p. 6 pp.

[82] W. Dong, B. M. Al-Hashimi, and M. T. Schmitz, "Improving routing efficiency for network-on-chip through contention-aware input selection," Yokohama, Japan, 2005, p. 6 pp.

[83] M. K. F. Schafer, T. Hollstein, H. Zimmer, and M. Glesner, "Deadlock-free routing and component placement for irregular mesh-based networks-on-chip," San Jose, CA, United States, 2005, pp. 238-245.

[84] A. Ejlali, B. M. Al-Hashimi, P. Rosinger, and S. G. Miremadi, "Joint Consideration of Fault-Tolerance, Energy-Efficiency and Performance in On-Chip Networks," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, 2007, pp. 1-6.

[85] G. Campobello, M. Castano, C. Ciofi, and D. Mangano, "GALS networks on chip: a new solution for asynchronous delay-insensitive links," Munich, Germany, 2006, p. 6 pp.

[86] J. Quartana, L. Fesquet, and M. Renaudin, "Modular asynchronous network-on-chip: application to GALS systems rapid prototyping," Perth, WA, Australia, 2005, pp. 397-402.

[87] A. Sheibanyrad and A. Greiner, "Two efficient synchronous ⇔ asynchronous converters well-suited for network on chip in GALS architectures," Montpellier, France, 2006, pp. 191-202.

[88] M. Greenstreet and J. Ren, "Surfing Interconnect," in *12th ASYNC*, 2006.

[89] M. Amde, T. Felicijan, A. Efthymiou, D. Edwards, and L. Lavagno, "Asynchronous on-chip networks," *IEE Proceedings-Computers and Digital Techniques,* vol. 152, pp. 273-83, 2005.

[90] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework," in *11th IEEE International Symposium on Asynchronous Circuits and Systems*, New York City, NY, USA, 2005, pp. 54-63.

[91] S. Moore, G. Taylor, R. Mullins, and P. Robinson, "Point to point GALS interconnect," in *International Symposium on Asynchronous Circuits and Systems*, Manchester, UK, 2002, pp. 69-75.

[92] J. Bainbridge and S. Furber, "Chain: a delay-insensitive chip area interconnect," *Micro, IEEE,* vol. 22, pp. 16-23, 2002.

[93] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proceedings of the IEEE,* vol. 89, pp. 490-504, 2001.

[94] L. Kangmin, L. Se-Joong, and Y. Hoi-Jun, "Low-power network-on-chip for high-performance SoC design," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 14, pp. 148-160, 2006.

[95] Pulsed Based On Chip Interconnect, 2007, http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-698.pdf

[96] P. P. Sotiriadis and A. Chandrakasan, "Bus energy minimization by transition pattern coding (TPC) in deep sub-micron technologies," in *Computer Aided*

Design, 2000. ICCAD-2000. IEEE/ACM International Conference on, 2000, pp. 322-327.

[97] S. R. Sridhara, A. Ahmed, and N. R. Shanbhag, "Area and energy-efficient crosstalk avoidance codes for on-chip buses," in *Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings. IEEE International Conference on*, 2004, pp. 12-17.

[98] L. Macchiarulo, E. Macii, and M. Poncino, "Wire placement for crosstalk energy minimization in address buses," in *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, 2002, pp. 158-162.

[99] A. Morgenshtein, I. Cidon, A. Kolodny, and R. Ginosar, "Comparative analysis of serial vs parallel links in NoC," in *International Symposium on System-on-Chip* Tampere, Finland, 2004, pp. 185-8.

[100] S. Kimura, T. Hayakawa, T. Horiyama, M. Nakanishi, and K. Watanabe, "An on-chip high speed serial communication method based on independent ring oscillators," United States, 2003, pp. 385-390.

[101] K. Lee, S.-J. Lee, S.-E. Kim, H.-M. Choi, D. Kim, S. Kim, M.-W. Lee, and H.-J. Yoo, "A 51mW 1.6GHz on-chip network for low-power heterogeneous SoC platform," San Francisco, CA., United States, 2003, pp. 152-153.

[102] S.-J. Lee, K. Lee, S.-J. Song, and H.-J. Yoo, "Packet-switched on-chip interconnection network for system-on-chip applications," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 52, pp. 308-312, 2005.

[103] K. Lee, S.-J. Lee, and H.-J. Yoo, "SILENT: Serialized low energy transmission coding for on-chip interconnection networks," in *ICCAD*, San Jose, CA, United States, 2004, pp. 448-451.

[104] R. Dobkin, R. Ginosar, and A. Kolodny, "Fast Asynchronous Shift Register for Bit-Serial Communication," in *12th ASYNC 2006*, 2006, pp. 117-126.

[105] S. Lee, K. Kim, H. Kim, N. Cho, and H. Yoo, "Adaptive network-on-chip with wave-front train serialization scheme," in *Symposium on VLSI Circuits,*, 2005.

[106] J. Xu and W. Wayne, "A wave-pipelined on-chip interconnect structure for networks-on-chips," in *11th Symposium on High Performance Interconnects*, 2003.

[107] H. Ito, J. Inoue, S. Gomi, H. Sugita, K. Okada, and K. Masu, "On-chip transmission line for long global interconnects," San Francisco, CA, USA, 2005, pp. 677-80.

[108] K. Masu, K. Okada, and H. Ito, "On-chip transmission line interconnect for Si CMOS LSI," San Diego, CA, USA, 2006, p. 4 pp.

[109] T. Mak, C. D'Alessandro, P. Sedcole, P. Y. K. Cheung, A. Yakovlev, and W. Luk, "Implementation of Wave-Pipelined Interconnects in FPGAs," in

Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on, 2008, pp. 213-214.

[110] J. Bainbridge and S. Furber, "Delay insensitive system-on-chip interconnect using 1-of-4 data encoding," in *7th ASYNC*, 2001, pp. 118-126.

[111] M. E. Dean, T. E. Williams, and D. L. Dill, "Efficient self-timing with level-encoded 2-phase dual-rail (LEDR)," in *1991 University of California/Santa Cruz conference on Advanced research in VLSI*, 1991, pp. 55-70.

[112] P. B. McGee, M. Y. Agyekum, M. A. Mohamed, and S. M. Nowick, "A Level-Encoded Transition Signaling Protocol for High-Throughput Asynchronous Global Communication," in *Asynchronous Circuits and Systems, 2008. ASYNC '08. 14th IEEE International Symposium on*, 2008, pp. 116-127.

[113] T. Verhoeff, "Delay-insensitive codes — an overview " *Distributed Computing,* vol. 3, Number 1, 1988 1998.

[114] C. D'Alessandro, D. Shang, A. Bystrov, A. Yakovlev, and O. Maevsky, "Multiple-Rail Phase-Encoding for NoC," in *12th ASYNC*, 2006, pp. 107-116.

[115] A. Ejlali and B. M. Al-Hashimi, "SEU-Hardened Energy Recovery Pipelined Interconnects for On-Chip Networks," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, 2008, pp. 67-76.

[116] H. Po-Tsang, F. Wei-Li, W. Yin-Ling, and H. Wei, "Low Power and Reliable Interconnection with Self-Corrected Green Coding Scheme for Network-on-Chip," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, 2008, pp. 77-83.

[117] M. Koibuchi, H. Matsutani, H. Amano, and T. Mark Pinkston, "A Lightweight Fault-Tolerant Mechanism for Network-on-Chip," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, 2008, pp. 13-22.

[118] J. Flich, A. Mejia, P. Lopez, and J. Duato, "Region-Based Routing: An Efficient Routing Mechanism to Tackle Unreliable Hardware in Network on Chips," in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, 2007, pp. 183-194.

[119] S. Murali, D. Atienza, L. Benini, and G. De Micheli, "A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip," in *Design Automation Conference, 2006 43rd ACM/IEEE*, 2006, pp. 845-848.

[120] M. Mondal, W. Xiang, A. Aziz, and Y. Massoud, "Reliability Analysis for On-chip Networks under RC Interconnect Delay Variation," in *Nano-Networks and Workshops, 2006. NanoNet '06. 1st International Conference on*, 2006, pp. 1-5.

[121] L. Chunsheng, Z. Link, and D. K. Pradhan, "Reuse-based test access and integrated test scheduling for network-on-chip," in *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, 2006, p. 6 pp.

[122] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, "Power-aware noc reuse on the testing of core-based systems," in *Test Conference, 2003. Proceedings. ITC 2003. International*, 2003, pp. 612-621.

[123] E. Cota, M. Kreutz, C. A. Zeferino, L. Carro, M. Lubaszewski, and A. Susin, "The impact of NoC reuse on the testing of core-based systems," in *VLSI Test Symposium, 2003. Proceedings. 21st*, 2003, pp. 128-133.

[124] N. John Mark and M. Rabi, "A TDM Test Scheduling Method for Network-on-Chip Systems," in *Microprocessor Test and Verification, 2005. MTV '05. Sixth International Workshop on*, 2005, pp. 90-98.

[125] T. Xuan-Tu, Y. Thonnart, J. Durupt, V. Beroulle, and C. Robach, "A Design-for-Test Implementation of an Asynchronous Network-on-Chip Architecture and its Associated Test Pattern Generation and Application," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, 2008, pp. 149-158.

[126] R. Dobkin, A. Morgenshtein, A. Kolodny, and R. Ginosar, "Parallel vs. Serial On-Chip Communication," in *SLIP* Newcastle, UK, 2008.

[127] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proceedings of the IEEE,* vol. 91, pp. 305-327, 2003.

[128] J. M. Rabaey, *Digital Integrated Circuits - A Design Perspective*, 1996.

[129] A. Narasimhan, M. Kasotiya, and R. Sridhar, "A low-swing differential signalling scheme for on-chip global interconnects," in *VLSI Design, 2005. 18th International Conference on*, 2005, pp. 634-639.

[130] Digital Video Interface Rev 1.0, 1999, http://www.ddwg.org/lib/dvi_10.pdf

[131] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proc IRE,* vol. 40, pp. 1098-1101, 1952.

[132] JPEG Standard, ISO/IEC IS 10918-1 | ITU-T, http://www.jpeg.org/jpeg/index.html

[133] S. Bunton and G. Borriello, "Practical dictionary management for hardware data compression," *Communications of the ACM,* vol. 35, pp. 95-104, 1992.

[134] A. V. Flores, "Hardware-based data compression technique," *IBM Technical Disclosure Bulletin,* vol. 27, pp. 2177-80, 1984.

[135] J. L. Nunez and S. Jones, "Gbit/s lossless data compression hardware," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 11, pp. 499-510, 2003.

[136] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory,* vol. IT-23, pp. 337-43, 1977.

[137] T. Lv, J. Henkel, H. Lekatsas, and W. Wolf, "An adaptive dictionary encoding scheme for SOC data buses," in *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, 2002, pp. 1059-1064.

[138] N. Banerjee, P. Vellanki, and K. S. Chatha, "A power and performance model for network-on-chip architectures," Paris, France, 2004, pp. 1250-1255.

[139] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," New Orleans, LA, United States, 2002, pp. 524-529.

[140] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The Nostrum backbone-a communication protocol stack for Networks on Chip," Mumbai, India, 2004, pp. 693-6.

[141] M. Amde, T. Felicijan, A. Efthymiou, D. Edwards, and L. Lavagno, "Asynchronous on-chip networks," in *System-on-Chip: Next Generation Electronics*, B. M. Al-Hashimi, Ed.: IEE, 2006, pp. 625-52.

[142] A. Iyer and D. Marculescu, "Power efficiency of voltage scaling in multiple clock multiple voltage cores," in *Computer Aided Design, 2002. ICCAD 2002. IEEE/ACM International Conference on*, 2002, pp. 379-386.

[143] S. Rama, H. Kim, and A. K. Somani, "Low-power high-performance reconfigurable computing cache architectures," *Computers, IEEE Transactions on,* vol. 53, pp. 1274-1290, 2004.

[144] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor," *Micro, IEEE,* vol. 27, pp. 51-61, 2007.

[145] D. Kearney and N. W. Bergmann, "Bundled data asynchronous multipliers with data dependent computation times," in *Advanced Research in Asynchronous Circuits and Systems, 1997. Proceedings., Third International Symposium on*, 1997, pp. 186-197.

[146] C. D'Alessandro, S. Delong, A. Bystrov, and A. Yakovlev, "PSK signalling on NoC buses," in *PATMOS* Leuven, Belgium, 2005, pp. 286-96.

[147] D. E. Muller and W. S. Bartky, "A Theory of Asynchronous Circuits," in *Proceedings of an International Symposium on the Theory of Switching*, 1959, pp. 204-243.

[148] R. David, "MODULAR DESIGN OF ASYNCHRONOUS CIRCUITS DEFINED BY GRAPHS," *IEEE Transactions on Computers,* vol. C-26, pp. 727-737, 1977.

[149] L. Morin and H. F. Li, "Design of synchronisers: a review," *IEE Proceedings E (Computers and Digital Techniques),* vol. 136, pp. 557-64, 1989.

[150] S. B. Furber and P. Day, "Four-phase micropipeline latch control circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 4, pp. 247-253, 1996.

[151] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit, "An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, pp. 155a-155a.

[152] R. Mariani, R. Roncella, R. Saletti, and P. Terreni, "On the realisation of delay-insensitive asynchronous circuits with CMOS ternary logic," in *Advanced Research in Asynchronous Circuits and Systems, 1997. Proceedings., Third International Symposium on*, 1997, pp. 54-62.

[153] S. Ogg, E. Valli, B. Al-Hashimi, A. Yakovlev, C. A. D'Alessandro, and L. A. Benini, "Serialized Asynchronous Links for NoC," in *Design, Automation and Test in Europe, 2008. DATE '08*, 2008, pp. 1003-1008.

[154] K. Chakrabarty, "Efficient modular testing and test resource partioning for core-based SoCs," in *System-on-Chip: Next Generation Electronics*, B. M. Al-Hashimi, Ed.: IEE, 2006, pp. 751-789.

[155] Arthur Pereira, F. Kastensmidt, Fernanda Lima, Luigi Carro, and A. Erika Cota, "Dependable Network-on-Chip Router Able to Simultaneously Tolerate Soft Errors and Crosstalk," in *Test Conference, 2006. ITC '06. IEEE International*, 2006, pp. 1-9.

[156] D. Rossi, P. Angelini, and C. Metra, "Configurable Error Control Scheme for NoC Signal Integrity," in *On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International*, 2007, pp. 43-48.

[157] ST-Microelectronics, *CORE9GPHS HCMOS9 TEC 3.2.a* vol. UNICAD2.4 / December 14, 2001, 2001.

[158] YUV Formats, http://www.fourcc.org/