Reliability Analysis of On-Chip Communication Architectures: An MPEG-2 Video Decoder Case Study

Rishad A. Shafik

Electronics Systems and Devices Group,
School of Electronics and Computer Science,
University of Southampton, SO17 1BJ, UK

Bashir M. Al-Hashimi

Electronics Systems and Devices Group,
School of Electronics and Computer Science,
University of Southampton, SO17 1BJ, UK

Abstract

In this paper, we present reliability analysis and comparison between on-chip communication architectures: dominant shared-bus AMBA and emerging network-on-chip (NoC); in the presence of single-event upsets (SEUs) using MPEG-2 video decoder as a case study. Employing SystemC-based fault simulations, reliability of the decoders is studied in terms of SEUs experienced in the computation cores and communication interconnects. We show that for a given soft error rate (SER), NoC-based decoder experiences lower SEUs than AMBA-based decoder. Using peak signal-to-noise ratio (PSNR) and frame error ratio (FER) metrics to evaluate the impact of SEUs at application-level, we show that NoC-based decoder gives up to 4dB higher PSNR, while AMBA experiences up to 3% lower FER. Furthermore, we investigate the impact of routing, application task mapping (distribution of tasks among computation cores) and architecture allocation (choice of number of computation cores) on the reliability of the decoders in the presence of SEUs.

Keywords: On-chip communication architecture, network-on-chip, soft error, reliability

1. Introduction

On-chip communication architecture is a key issue in the design of multiprocessor system-on-chip (MPSoC), since the chosen architecture influences the system performance and power

Email addresses: `ras06r@zepler.net` (Rishad A. Shafik), `bmah@ecs.soton.ac.uk` (Bashir M. Al-Hashimi)

consumption [1, 2, 3]. Shared-bus, such as advanced microprocessor bus architecture (AMBA), is a dominant, industry standard on-chip communication architecture [4]. To address the performance and scalability issues in the design of future MPSoCs, network-on-chip (NoC) has evolved as an emerging on-chip communication architecture [5]. Over the years researchers have proposed a number of flexible NoC architectures with efficient communication techniques. For example, ÆTHEREAL NoC architecture has been proposed by [6] with guaranteed communication services and NOSTRUM NoC architecture with layered communication approach has been presented in [7]. Among other developments, recently a mesh-based Intel 80-core NoC architecture with clock frequency higher than 4GHz has been proposed in [8].

An emerging challenge in MPSoC design is reliability in the presence of different faults. These faults can generally be classified in two types: permanent and transient. Permanent faults are related to irreversible physical defects in the circuit, which are produced during manufacturing process. Transient faults, also known as soft errors, take place when a single ionising radiation event produces a burst of hole-electron pairs in a transistor that is large enough to cause the circuit to change state. Single-event upset (SEU) is the most popular transient fault model used in the study of reliability [9], which is exacerbated by scaling and low power design techniques [10, 11].

To mitigate the impact of soft errors a number of studies have shown different fault tolerant on-chip communication architectures and techniques for MPSoCs. For example, in [10] an investigation into reliability of different NoC architectures has been reported. Based on the investigation, effective fault tolerance techniques have been proposed for different NoC configurations to operate in the presence of soft errors. Another reliability analysis of on-chip communication architectures from performance, reliability and energy perspective has been carried out in [12]. Using such analysis an array of different fault tolerance techniques have been introduced at architectural- and algorithmic-level to tackle the reliability issues of communication components. In [13] a fault tolerant design of interconnects in on-chip communication architectures has been considered explaining conflicting design trade-offs between reliability and performance. The impact of power minimization on reliability has been examined in [14] showing effective power-aware fault tolerance design techniques for on-chip communication architectures. Several other techniques, such as stochastic communication [15] and routing [16], have also been proposed to incorporate fault tolerance in on-chip communication architectures. Although good progress has been made in the development of fault tolerant architectures and techniques, currently there is a lack of analysis of how on-chip communication architecture affects the reliability of MPSoCs in the presence of soft errors. For the NoC methodology to gain further maturity, such insightful analysis of reliability need to be performed highlighting comparison between dominant shared-bus AMBA and NoC, which is the main aim of this paper. To the best of our knowledge, no such study has yet been reported. In this paper, using cycle-accurate SystemC-based simulations we investigate the number of SEUs experienced in computation cores and communication interconnects in shared-bus AMBA and NoC employing real application traffic of MPEG-2 video decoder. We evaluate the number of SEUs experienced for a given soft error rate (SER) and show the impact of SEUs experienced at application-level. Furthermore, we investigate the impact of routing, application task mapping (distribution of application tasks among processing cores) and architecture allocation (choice of number of processing cores) on the reliability of the AMBA- and NoC-based decoders. The rest of the paper is organized as follows. Section 2 describes application, architecture and fault injection model used in this work. Section 3 compares between AMBA- and NoC-based decoders in terms of SEUs experienced in computation cores and communication interconnects, and evaluates the impact of SEUs at application-level. Section 4 demonstrates the impact of application task mapping and architecture allocation on the reliability of decoders. Finally, Section 6 concludes the paper.

2. System Model

In this section, MPEG-2 video decoder-based application model and MPSoC architectures employing the decoder cores (with AMBA and NoC on-chip communication) are described. Also, the fault injection model used to evaluate reliability of the MPSoC decoders in the presence of soft errors is explained.

2.1. Application Model: MPEG-2 Video Decoder

MPEG-2 video decoder constitues a major component of MPSoC applications and is chosen as an application case study. Figure 1(a) shows block diagram of the MPEG-2 video decoder with four processing cores used in this work. SystemC behavioural modelling is used to design the decoder cores, while partitioning and allocation of application are performed arbitrarily to reflect MPSoC. The variable length decoder (VLD) core decompresses the input bitstream and defines the header sequence with different parameters and video sequence with coded video blocks. The

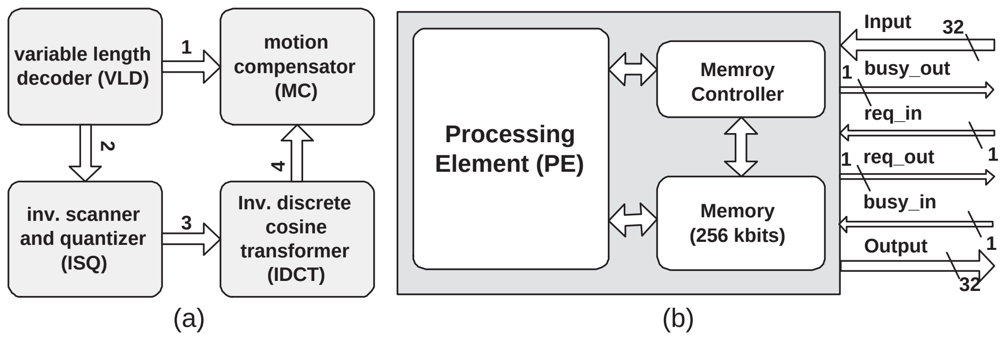


Figure 1: (a) Block diagram of MPEG-2 video decoder with four processing cores, and (b) block diagram of the processing core used in MPEG-2 video decoder

video sequence is then inverse scanned and quantized by the inverse scanner and quantizer (ISQ) core, while part of the header sequence is sent to motion compensator (MC) core. The scanned and quantized video blocks are transformed into time-domain picture-ready video blocks by the inverse discrete cosine transformer (IDCT) core. Using these picture-ready video blocks MC core forms inter- and intra-frame predictions and stores or displays the decoded frames (Figure 1(a)).

Figure 1(b) shows block diagram of a processing core used in Figure 1(a). Each core consists of a processing element (PE) for computation and a private memory (of 256 kbits) interfaced by a memory access controller. The memory size is chosen to give high availability for data processing and storage within processing cores. The processing core also lays out 32-bit input and output interfaces for transfer of data transaction units (DTUs: the unit of data transfer for an on-chip communication architecture, e.g. 32-bit payload packets in NoC or 32-bit data bus in AMBA) and handshake signals (*busy_in*, *busy_out*, *request_in* and *request_out*) for enabling communication to/from the processing core (Figure 1(b)). MPEG-2 video decoder is capable of decoding video bitstreams with different rates and sizes. Table 1 shows four video bitstreams[1] with different resolutions and sizes, which are used for comparisons in Section 3.

2.2. Shared-bus AMBA

Shared-bus AMBA employs a central multiplexor scheme, called a bus, which controls the access and direction of on-chip communication. Using such scheme all masters (e.g. processing elements) in an MPSoC are required to be granted mutually exclusive access to the bus by an arbiter to

[1]Source: ftp://ftp.tek.com/tv/test/streams/Element/

Video	Frames	Bitrate	Frame Size (pixels)
test1.m2v (tennis)	67	4 Mbps	176x120 (QCIF,NTSC)
test2.m2v (flower)	55	5.2 Mbps	352x288 (CIF,PAL)
test3.m2v (tennis)	49	7 Mbps	352x576 (2CIF,PAL)
test4.m2v (flower)	43	7 Mbps	704x480 (4CIF,NTSC)

Table 1: Video bitstreams used for comparisons in this work

be able to initiate data transfer. The slaves (e.g. memory), selected by a central decoder, cannot initiate any data transfer but can serve requested services (read or write) from master. Depending on performance and connectivity of masters or slaves, different bus architectures are defined within AMBA specification [17]. Advanced high-performance bus (AHB) is used as shared-bus AMBA in this work due to its high performance [4]. A single-layer central multiplexor configuration with

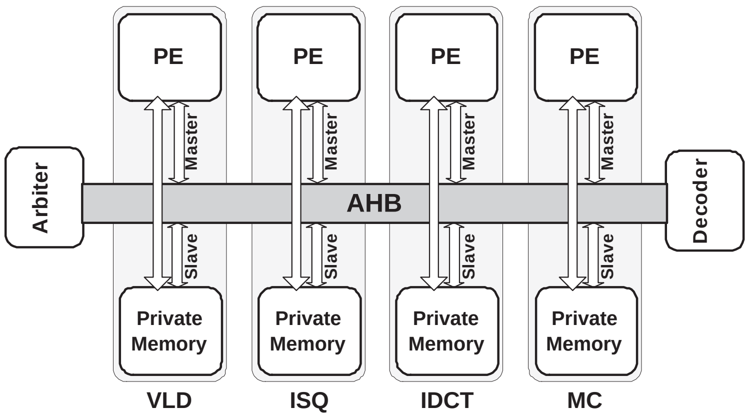


Figure 2: Shared-bus AMBA-based decoder used in this work

pipelined single-burst transfer and no waiting states are used to maximize throughput. MPEG-2 video decoder cores (Figure 1(b)) are configured by using the 32-bit input port as slave port (for memory interface) and 32-bit output port as master port (for PE interface) as shown in Figure 2. As a result each core can process data from internal memory and initiate write operation through its master interface when access to bus is available and write data to slave interface that is connected to the next communicating core. The cores share bus access in the sequence of cores VLD, ISQ and IDCT (Figure 2) with each core holding the interconnect access until the current macroblock (the basic unit in video decoding with 16×16 pixels of a video frame) is processed and stored in the memory of the next core. To facilitate AMBA-based cycle-accurate simulations, we use Synposys

Designware SystemC libraries[2].

2.3. Network-on-Chip

Network-on-Chip (NoC) incorporates packet-based on-chip communication with links laid out in different directions, while packet routing and communication is controlled by a switch. NoC gives large design space with different routing techniques, switch architectures and network topologies [10]. In this work, we use a mesh-based NoC topology with deterministic XY routing and single-flit-packet *wormhole* communication due to simplicity of switch design, performance and scalability [18]. The impact of using different routing algorithms in switch is investigated in Sec-

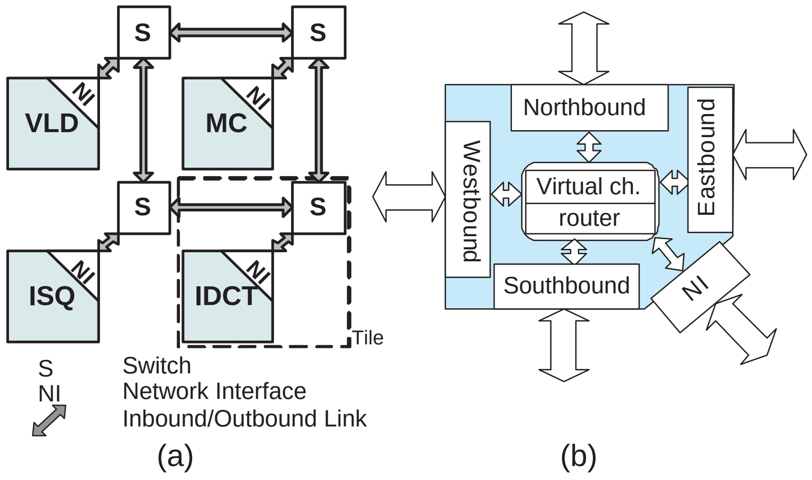


Figure 3: (a) Mesh-based (2×2) NoC employing MPEG-2 decoder cores, (b) 5-port NoC switch used in this work

tion 3. Figure 3(a) shows block diagram of a (2×2) NoC-based MPEG-2 decoder architecture with shortest path floor mapping between connecting cores. As can be seen, NoC architecture is made up of basic networking unit, called a tile, consisting of a processing core for computation, switch for communication and network interface (NI) for packets-based interface. Each packet in NoC-based communication contains 32-bit data payload and 46-bit headers as shown in Table 2. As can be seen the packet header consists of packet ID, source and destination ID, routing and virtual channel information and credit signals. The payload contains the actual computation data. For such packet structure the size of each NoC packet is (32+46)=78 bits.

Figure 3(b) shows a block diagram of switch architecture used in the NoC architecture (Figure 3(a)) with five inbound and outbound ports. Four ports connect with other switches with

[2]www.synopsys.com/Tools/SLD/VirtualPlatforms/Pages/SLLibrary.aspx

Type	Size, bits
Packet/Flit ID	20
Packet/Flit type	2
Routing type	4
Routing header	8
Source ID	4
Destination ID	4
Virtual channel ID	4
Packet/Flit payload	32

Table 2: NoC packet overheads for single-flit-packet wormhole routing

buffer for eight packets on channels and one port is laid out between PE and NI with buffer for four packets. Virtual channel (VC) provides buffering for eight incoming packets and router selects output port based on routing technique used (Figure 3(b)). The size of different buffers are chosen to give high bandwidth and less congestion on communication ports and channel. The MPEG-2 decoder cores (Figure 1(b)) are configured by connecting the 32-bit input port with network interface (NI) data input port and 32-bit output port with the NI data output port. To facilitate NoC simulations, we use SystemC-based cycle-accurate simulator NIRGAM [19].

2.4. Fault Injection Model

In this work, fault injection is carried out using SEU-based fault model employing the technique proposed in [20]. The injection of SEUs using this simulator is initiated through replacement of variable or signal types in the original design specification to equivalent fault injection enabler types. To demonstrate how such type replacements are made Figure 4(a) shows part of the original SystemC description of IDCT processing core, while Figure 4(b) shows the modified SystemC description with fault injection enabler types. As can be seen, the fault injection enabler types are incorporated into the design description through inclusion of header file *FIReg.h*. The original *int*, *short* and *sc_int* types (Figure 4(a)) are replaced by equivalent Reg<short>, Reg<short> and RegInt<..> types Figure 4(b). Such type replacement (Figure 4(b)) enables the formation of a fault locations database, which contains the target registers for SEU injection. The simulator injects SEUs based on the specified soft error rates and probability distribution to identify fault locations within the fault locations database. Figure 5 shows the fault injection setup employing the fault injection simulator used for the MPEG-2 decoder with four processing cores (Figure 1). Using type

```
.. .. ..
//Other header files
//global variables/constants
.. .. ..
int x0, x1, x2, x4, x5;
short *blk;
.. .. ..
x0 = (blk[0]<<11) + 128;
.. .. ..
sc_int<4> row, column;
.. .. ..
if(row == 0){.. .. ..}
.. .. ..
```

(a)

```
#include  "fim/FIReg.h"
//Other header files
//global variables/constants
.. .. ..
Reg<int>  x0, x1, x2, x4, x5;
Reg<short>  *blk;
.. .. ..
x0 = (blk[0]<<11) + 128;
.. .. ..
RegInt<4>  row, column;
.. .. ..
if(row == 0){.. .. ..}
.. .. ..
```

(b)

Figure 4: Example usage of fault injection enabler types to initiate fault injection

replacements for variable/signal in the original design specification, the simulator enables formation of five fault locations databases: one for each of the four processing cores and a centralized fault locations database for the interconnects. For a given soft error rate (SER, in number of SEUs per bit per cycle), the number of SEUs to be injected within each fault locations database is found and their locations are determined by Poisson distribution. The system clock is connected to the fault injection simulator to enable timing information for fault injection (Figure 5). Using simulation-specific monitor modules, total register usage and number of faults injected can be found. More details regarding fault injection can be found in [20].

3. Comparative Reliability Analysis

Reliability of an application against SEUs is related to the total number of SEUs experienced over a given time [21]. Our aim in this work is to analyze how the reliability of MPEG-2 video decoder is affected by the choice of on-chip communication architectures: AMBA and NoC. To this end, the following investigations are carried out:

- evaluate the number of SEUs experienced during computation, $\mathcal{F}_{comp}$, to show how MPEG-2 decoder computation is affected,
- evaluate the number of SEUs experienced during communication, $\mathcal{F}_{comm}$, in the MPEG-2 decoder to show how on-chip communication is affected, and
- evaluate the impact of total SEUs experienced, $\mathcal{F} = \mathcal{F}_{comp} + \mathcal{F}_{comm}$, at application-level to demonstrate how decoder reliability is affected.

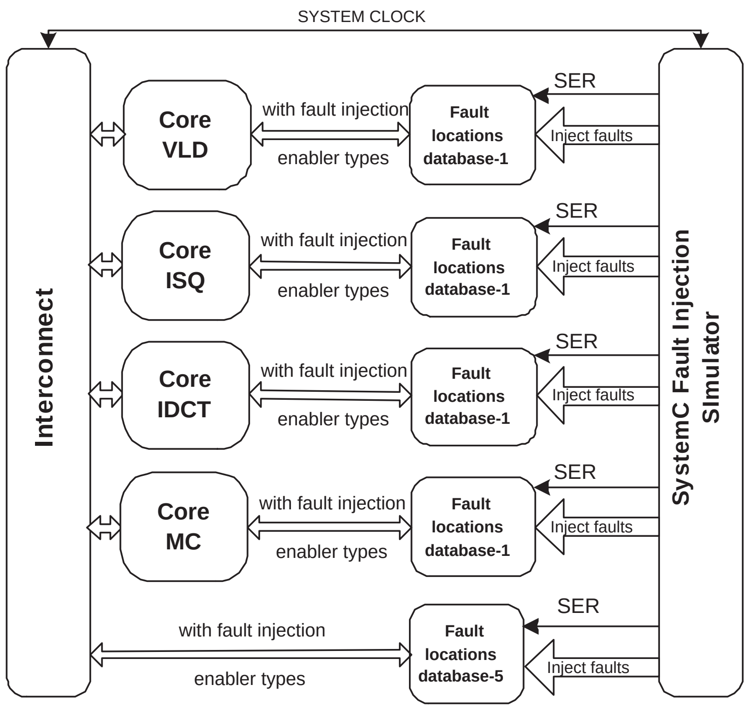


Figure 5: Fault injection setup for processing cores of an MPSoC

In the following (Sections 3.1 and 3.2), $\mathcal{F}_{comp}$ and $\mathcal{F}_{comm}$ of AMBA- and NoC-based decoders are evaluated and compared. Later (in Section 3.3), the impact of $\mathcal{F}$ is evaluated at application-level.

3.1. SEUs Experienced During Computation

The SEUs affect computation of a processing core through perturbation of the registers. Figure 6 shows how SEUs manifest themselves in registers of the processing cores during computation. As can be seen, SEUs extending between two IDLE cycles (instance 3) do not affect computation

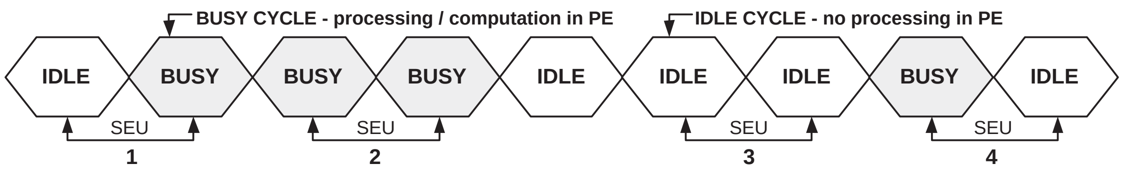

BUSY CYCLE - processing / computation in PE
IDLE CYCLE - no processing in PE
IDLE
BUSY
BUSY
BUSY
IDLE
IDLE
IDLE
BUSY
IDLE
SEU
1
SEU
2
SEU
3
SEU
4

Figure 6: Manifestation of SEUs during computation cycles of a processing core

process as no computation takes place in these cycles. On the other hand, SEUs that are injected between BUSY cycles (instance 2) or between BUSY and IDLE cycles (instances 1 and 4) are likely to affect computation process. Hence, for a given soft error rate (SER), the effective number of

SEUs experienced during computation ($\mathcal{F}_{comp}$) can be given as the number of SEUs experienced by the computation cycles (in instances 1, 2 and 4) during execution of a processing core. The $\mathcal{F}_{comp}$ of an MPSoC decoder with C processing cores can be given as

$$\mathcal{F}_{comp} = \sum_{i=1}^{C} \left(T_i - T_i^{I-I} \right) R_i \lambda \quad , \tag{1}$$

where λ is the SER (in SEUs per bit per cycle), T_i is the execution time (in clock cycles), T_i^{I-I} is the number of idle-to-idle transitions within T_i (in clock cycles) and R_i is the register usage (in bits per cycle), all for i-th processing core. The R_i gives a measure of per core register usage by the application, since SEUs in other registers have no impact [21]. The R_i is given by [20] as

$$R_i = \frac{1}{T_i} \sum_{t=1}^{T_i} R_{i,t} \quad . \tag{2}$$

where $R_{i,t}$ is the instantaneous number of registers (in bits) used by MPEG computation process at t-th clock cycle in i-th processing core. Table 3 shows execution time, T_i, idle-idle transition cycles, T_i^{I-I}, and register usage, R_i, of each processing core in AMBA- (Figure 2) and NoC-based decoders (Figure 3(a)) for decoding different video bitstreams (Table 1). The execution times (T_i and T_i^{I-I}) and the register usage (R_i) of AMBA and NoC-based decoder cores VLD, ISQ, IDCT and MC are shown in columns 3-6 (Table 3). The T_i and T_i^{I-I} values of AMBA- and NoC-based decoders are obtained from SystemC cycle-accurate simulations (Sections 2.2 and 2.3) and R_i values are found through SystemC fault simulations (Section 2.4). As can be seen, AMBA-based decoder has similar register usage, R_i, as NoC for all four cores while decoding *test1.m2v* due to same processing cores between the two decoders (row 2, columns 3-6). However, as the registers in AMBA-based decoder are also used over idle period during bus arbitration, it has up to 7% lower register usage (given by (2)) than NoC-based decoder. Due to shared-bus access among decoder cores, AMBA-based decoder has up to 2.18 times higher execution time for core MC compared to NoC-based decoder while decoding *test1.m2v*. Such time sharing of bus access also causes more idle-idle transition cycles (T_i^{I-I}) in AMBA-based decoder, resulting in up to 6.9 times higher T_i^{I-I} compared to NoC-based decoder for core MC (row 2, column 6). With increased video sizes in other video bitstreams (*test2.m2v*, *test3.m2v* and *test4.m2v*), T_i and T_i^{I-I} values increase but similar trend continues between AMBA- and NoC-based decoders for R_i, T_i and T_i^{I-I} values. Higher T_i results in higher number of SEUs experienced during computation ($\mathcal{F}_{comp}$) in AMBA-based decoder compared to NoC-based decoder for decoding video different bitstreams (Table 1),

Video	Arch.	Core VLD			Core ISQ			Core IDCT			Core MC		
		T_i, cyc. (x10^6)	T_i^{I-I}, cyc. (x10^6)	R_i, kb/c.	T_i, cyc. (x10^6)	T_i^{I-I}, cyc. (x10^6)	R_i, kb/c.	T_i, cyc. (x10^6)	T_i^{I-I}, cyc. (x10^6)	R_i, kb/c.	T_i, cyc. (x10^6)	T_i^{I-I}, cyc. (x10^6)	R_i, kb/c.
test1.m2v	NoC	6.43	0.42	23.0	3.78	0.41	19.3	6.37	0.05	19.4	6.69	0.23	25.2
	AMBA	13.4	2.4	22.5	7.48	1.9	19.0	13.7	1.4	19.1	14.6	1.6	24.7
test2.m2v	NoC	18.7	1.2	23.1	14.2	1.6	19.3	18.5	0.14	20.2	19.4	0.68	25.3
	AMBA	39.6	7.5	22.7	28.6	7.3	19.0	40.4	4.3	19.7	42.6	5.1	24.7
test3.m2v	NoC	32.3	2.2	23.4	25.0	3.0	19.4	32.0	0.25	20.5	33.6	1.2	25.5
	AMBA	69.8	14.0	22.7	51.4	13.0	19.0	70.2	7.5	19.8	74.0	9.2	24.8
test4.m2v	NoC	35.5	2.5	23.9	26.6	3.2	19.5	35.3	0.29	20.7	36.9	1.3	25.7
	AMBA	78.1	16.0	23.3	55.3	14.0	19.0	79.3	8.5	19.9	81.7	11.0	25.0

Table 3: Execution times, T_i, idle-idle transition times, T_i^{I-I}, and average register usages, R_i, of processing cores in AMBA- and NoC-based decoders

as shown in Figure 7. The $\mathcal{F}_{comp}$ values are found from simulations using an arbitrary SER of 10^{-9} SEUs/bit/cycle in simulated fault injection environment (Section 2.4). The approximate $\mathcal{F}_{comp}$ values can also be validated through (1) with T_i, T_i^{I-I} and R_i values from Table 3. As expected, the

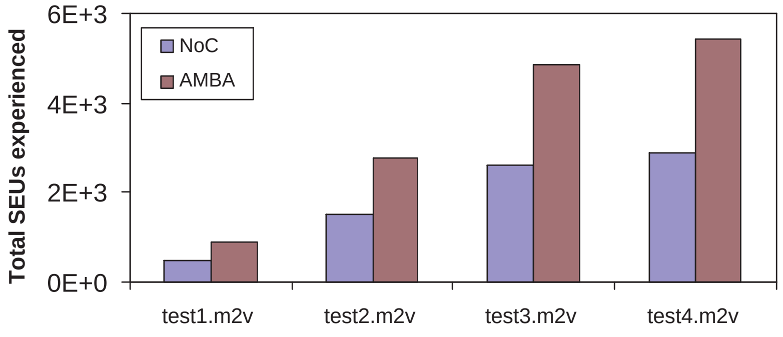


Figure 7: Comparative $\mathcal{F}_{comp}$ in AMBA- and NoC-based decoders for an SER of 10^{-9}

AMBA-based decoder experiences approximately 83% higher $\mathcal{F}_{comp}$ on average compared to NoC for decoding different video bitstreams. As a result of higher $\mathcal{F}_{comp}$, MPEG-2 decoder computation is expected to be affected more in AMBA-based decoder than NoC-based decoder. In Section 3.3 the impact of SEUs experienced is examined at application-level.

3.2. SEUs Experienced During Communication

An important aspect in the reliability of on-chip communication architectures is the number of SEUs experienced during inter-core data communication as these SEUs perturb the registers in the interconnects and affect the data transfer [22]. The number of SEUs experienced during communication, $\mathcal{F}_{comm}$, depends on how the DTUs are transferred between communicating cores in an on-chip communication architecture. For a given SER (in SEUs per bit per cycle), the total $\mathcal{F}_{comm}$ of an on-chip communication architecture can be given by the product of per data transaction unit (DTU: packet for NoC and 32-bit data for AMBA) communication time, total number of DTUs transferred among cores, the register usage of the communication components and the SER. Hence $\mathcal{F}_{comm}$ can be expressed as

$$\mathcal{F}_{comm} = \sum_{j=1}^{M} \mathcal{N}_j L_{ch_j} R_{com_j} \lambda \quad , \tag{3}$$

where M is the number of inter-core communication links in the decoder ($M = 4$, Figure 1(a)), $\mathcal{N}_j$ is the total number of DTUs between cores, L_{ch_j} is the channel latency (in clock cycles) and R_{com_j} is the average register usage in communication components during transfer of DTUs on j-th link. The channel latency, L_{ch_j} in (3) gives a measure of communication time of DTUs within the on-chip communication architecture and is given by the time (in cycles) required for a DTU to be transferred from the output port of a processing core to an input port of target processing core. The for a given link, L_{ch} can be expressed as

$$L_{ch} = \frac{1}{\mathcal{N}} \sum_{n=1}^{\mathcal{N}} \left[\tau^{S}_{c-in}(n) + \tau^{S-D}_{in-in}(n) + \tau^{D}_{in-c}(n) \right] , \tag{4}$$

where $\tau^{S}_{c-in}(n)$ is the time elapsed for DTU to travel from source output port to source interconnect port, $\tau^{S-D}_{in-in}(n)$ is the time elapsed for DTU to travel from source interconnect port to destination interconnect port and $\tau^{D}_{in-c}(n)$ is the time elapsed for DTU to travel from destination interconnect port to the destination core memory, all for n-th DTU out of total $\mathcal{N}$ DTUs. For AMBA, $\tau^{S}_{c-in}(n) =$ 1 clock cycle after bus access is granted and locked. During $\tau^{S-D}_{in-in}(n) = 1$ clock cycle the arbiter in AMBA does the necessary routing of the data and notifies the slave port. Due to direct memory interface, $\tau_{D_{in-c}}(n) = 0$ clock cycle. With these delays the minimum channel latency (without waiting states) per DTU for AMBA is $L_{ch} = 2$ clock cycles found through (4). Due to symmetric nature of NoC channels, $\tau^{D}_{c-in}(n) = \tau^{S}_{in-c}(n) = 3$ clock cycles involving intermediate NI packetizing and de-packetizing (Figure 3(a)). The delay, $\tau_{in-in}^{S-D}(n)$, in (4) involves communication over an array of switches for each DTU (packet with 32-bit payload) and depends on the number of intermediate switches travelled. The $\tau_{in-in}^{S-D}(n)$ in (4) can be given as

$$\tau_{in-in}^{S-D}(n) = \sum_{k=1}^{K-1} [\tau_{ic-r}^{s}(n) + \tau_{r}^{s}(n) + \tau_{r-oc}^{s}(n) + \tau_{oc-ic}^{k-(k+1)}(n)]. \tag{5}$$

Equation (5) is a result of multi-hop NoC packet communication through K intermediate switches and involves the following delays. The time required for the n-th packet to travel from input channel to the router of the k-th switch, $\tau_{ic-r}^{k}(n)$, is 1 clock cycle for the NoC switch design (Figure 3(b)). Also, the time required for routing decision on the k-th switch for n-th packet, $\tau_{r}^{k}(n)$, is 1 clock cycle. The n-th packet travels from router to the output channel of the k-th switch immediately in the NoC implementation and hence $\tau_{r-oc}^{k}(n) = 0$ clock cycle. Finally, the time required for the n-th packet to travel from output channel of k-th switch to input channel of the $(k+1)$-th switch, $\tau_{oc-ic}^{k-(k+1)}(n)$, is 1 clock cycle. Using (4) and (5), NoC has a minimum channel latency (L_{ch}) of 9 clock cycles (with $K = 2$ for shortest path mapping and XY routing, Figure 3(a)) compared to only 2 clock cycles in AMBA (Figure 2). Note that L_{ch} for NoC varies for different floor mapping of processing cores on NoC tiles. This is because floor mapping affects the number of intermediate switches travelled due to placement of cores on NoC tiles [23]. For example, L_{ch} increases to 15 and 20 clock cycles for floor mapping with 3 and 4 intermediate switches (3 intermediate switches correspond to one interleaved core between communicating cores and 4 intermediates switches mean two interleaved core between connecting cores), respectively. Similarly, packet routing affects the channel latency since different communication paths result in varied number of intermediate switches (in (5)) travelled [24].

The average register usage of communication components during transfer of a DTU, R_{com_j} in (3), sets up another difference between AMBA- and NoC-based decoders. The R_{com_j} can be given by dividing the total register usage during inter-core transfer of DTUs by the number of DTUs, i.e.

$$R_{com_j} = \frac{1}{(L_{ch_j}\mathcal{N}_j)} \sum_{n=1}^{\mathcal{N}_j} \sum_{l=1}^{L_{ch_j}} R_{n,l} \quad , \tag{6}$$

where $R_{n,l}$ is the instantaneous register usage on j-th link during inter-core communication of n-th DTU at l-th clock cycle (l=1:L_{ch_j}). For NoC-based decoder, $R_{n,l}$ in (6) includes registers used in packet overheads and buffers in NI interfaces, channels, VCs, and routers as packet is communicated between cores. For AMBA-based decoder, $R_{n,l}$ includes the registers used in address (HADDR), control signals (RD and WR), decoder and arbiter as DTU is communicated between cores. Using (6), R_{com_j} in NoC-based decoder (Figure 3(a)) obtained from simulation logs is approximately 212 bits per data transfer cycle (for using XY packet routing) and that in AMBA-based decoder is approximately 87 bits per transfer cycle. The higher R_{com_j} of NoC is expected as NoC incorporates packet based multi-hop routing and buffering with complex switch structure. Note that R_{com_j} of NoC is dependent on the packet routing algorithm as underlying routing algorithm determines the switch design complexity and the associated the register usage [24]. For example, using source-based routing algorithm gives R_{com_j} value of 187 bits per cycle, while using odd-even routing algorithm results in R_{com_j} value of 273 bits per cycle as opposed to 212 bits per cycle for XY routing.

Video	$\mathcal{N}_1$ (VLD→MC), $\times 10^3$	$\mathcal{N}_2$ (VLD→ISQ), $\times 10^3$	$\mathcal{N}_3$ (ISQ→IDCT), $\times 10^3$	$\mathcal{N}_4$ (IDCT→MC), $\times 10^3$
test1.m2v	66	78	108	202
test2.m2v	232	273	364	666
test3.m2v	385	454	605	1111
test4.m2v	1598	1884	2503	4580

Table 4: Inter-core data transaction units (DTUs) for decoding different video bitstreams

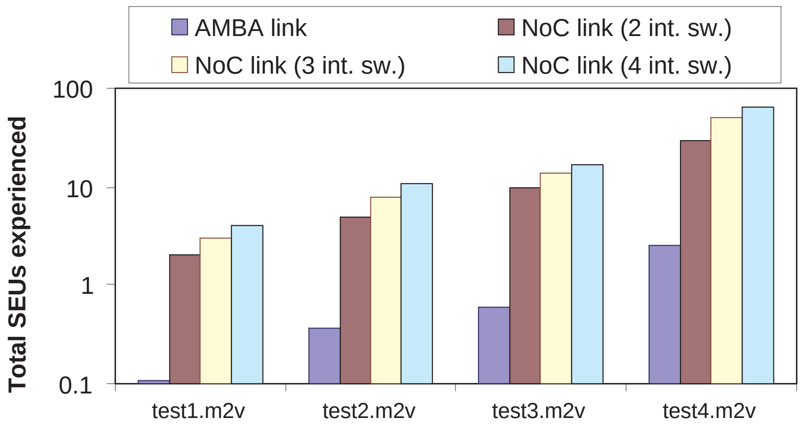


Figure 8: Comparative $\mathcal{F}_{comm}$ in AMBA and NoC links

Table 4 shows the number of DTUs, $\mathcal{N}_i$ ($\mathcal{N}_1$ for VLD-MC link, $\mathcal{N}_2$ for VLD-ISQ link, $\mathcal{N}_3$ for

ISQ-IDCT link, and $\mathcal{N}_4$ for IDCT-MC link, Figure 1(a)), recorded from simulation logs. Note that $\mathcal{N}$ values do not change between AMBA- and NoC-based decoders for a given video bitstream due to similar architecture for processing cores (Figure 1(a)). For decoding a given video bitstream, $\mathcal{N}$ is the least from core VLD to core ISQ. As the video decoding progresses with other cores, $\mathcal{N}$ between cores increases due to decompression of the original video bitstream. For example, only $\mathcal{N}$=66 $\times$ 10^3 DTUs are transferred from core VLD to core ISQ, while $\mathcal{N}$=202 $\times$ 10^3 DTUs are transferred from core IDCT to core MC for decoding *test1.m2v* (row 2, Table 4). For increased video sizes, $\mathcal{N}$ also increases for a given link. For example, 108 $\times$ 10^3 DTUs are transferred from core ISQ to core IDCT for decoding *test1.m2v* compared to 364 $\times$ 10^3 DTUs on the same link for decoding *test2.m2v* (column 4, Table 4). Figure 8 shows comparative $\mathcal{F}_{comm}$ of AMBA- and NoC-based decoders obtained from simulation logs for an arbitrary SER of 10^{-9}, while decoding different video bitstreams (Table 1). Approximate $\mathcal{F}_{comm}$ values of the decoders can be found by (3) using $\mathcal{N}_i$, R_{comm_j} and L_{ch} values discussed above. To demonstrate the impact of floor mapping, $\mathcal{F}_{comm}$ values of three different NoC configurations are shown with 2, 3 or 4 intermediate switches between cores. As expected, due to higher register usage (R_{com_j}) and channel latency (L_{ch}), NoC-based decoder links with 2 intermediate switches (Figure 3(a)) suffer from 11 times higher $\mathcal{F}_{comm}$ compared to AMBA, which worsens to 18 and 24 times higher $\mathcal{F}_{comm}$ as number of intermediate switches increase to 3 and 4, while decoding *test1.m2v* (Figure 8). Similar trends between AMBA- and NoC-based decoders in terms of $\mathcal{F}_{comm}$ are also observed with other video bitstreams (Figure 8).

To demonstrate the impact of choice of NoC packet routing algorithms on the $\mathcal{F}_{comm}$, Figure 9 shows the $\mathcal{F}_{comm}$ values for different packet routing algorithms: source-based, XY and odd-even routing algorithm implemented on NIRGAM [19]. The $\mathcal{F}_{comm}$ values are found with SER of 10^{-9}, while decoding the video bitstream *test4.m2v* (Table 1). The approximate values of $\mathcal{F}_{comm}$ can be found through (3) using the L_{ch} and R_{com_j} values of AMBA- and NoC-based decoders. As can be seen, using source-based packet routing in NoC switches gives the least SEUs experienced during communication ($\mathcal{F}_{comm}$), while odd-even routing algorithm gives the highest $\mathcal{F}_{comm}$. This is because, due to source initiated routing information inserted in the packets, source-based routing gives the least register usage of 187 bits per cycle and simpler switch design. On the other hand, odd-even routing implements adaptive strategy of packet routing with a control mechanism to

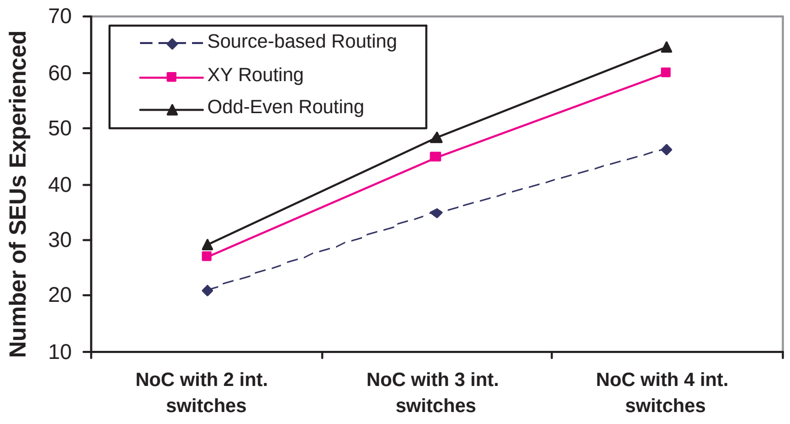


Figure 9: Impact of choice of routing algorithm on $\mathcal{F}_{comm}$ in NoC interconnects, while decoding *test4.m2v*

avoid deadlock and intermediate packet buffering, resulting in complex switch design [24] and higher register usage of 273 bits per cycle. The XY routing has lower register usage (212 bits per cycle) than odd-even due to its deterministic nature of choice of routing directions [25]. As expected, as more number of switches are travelled by NoC packets using these routing algorithms, the $\mathcal{F}_{comm}$ values also increase linearly.

Comparing between $\mathcal{F}_{comm}$ (Figure 7) and $\mathcal{F}_{comp}$ values (Figure 8) of AMBA- and NoC-based decoders while decoding a given video bitstream, it can be seen that $\mathcal{F}_{comm} \ll \mathcal{F}_{comp}$. Nevertheless, $\mathcal{F}_{comm}$ affects the reliability on-chip communication as it leads to faults resulting in misrouting or loss of DTUs [22]. The loss of DTUs or misrouting causes the decoding process to be terminated or skip a number of video blocks or frames while decoding [26]. Next, the impact of overall SEUs experienced ($\mathcal{F}$) is evaluated at application-level.

3.3. Impact of SEUs at Application-Level

In Sections 3.1 and 3.2, the reliability of AMBA- and NoC-based decoders were investigated in terms of the SEUs experienced during computation ($\mathcal{F}_{comp}$) and communication ($\mathcal{F}_{comm}$). With the $\mathcal{F}_{comp}$ and $\mathcal{F}_{comm}$ values from (1) and (3), the total number of SEUs experienced, $\mathcal{F}$, is given as

$$\begin{aligned} \mathcal{F} &= \mathcal{F}_{comp} + \mathcal{F}_{comm} \quad , \\ &= \left[\sum_{i=1}^{C} \left(T_i - T_i^{I-I} \right) R_i \lambda \right] + \left[\sum_{j=1}^{M} \mathcal{N}_j L_{ch_j} R_{com_j} \lambda \right] . \end{aligned} \tag{7}$$

In this section, the impact of injected SEUs, $\mathcal{F}$, given by (7), is evaluated at application-level. Such evaluation has also been used in [21] showing that the faults at architectural-level do not always lead to faults at application-level enabling low-cost fault tolerance mechanisms. We evaluate the impact of $\mathcal{F}$ on decoder reliability using peak signal-to-noise ratio (PSNR) metric (as also used by [21]). PSNR is defined as

$$PSNR = 10\log_{10}\frac{1}{PQ}\sum_{p=1}^{P}\sum_{q=1}^{Q}\frac{255^2}{(x_{p,q}-y_{p,q})^2} \quad , \tag{8}$$

where P is the number of frames, each with Q pixels, $x_{p,q}$ and $y_{p,q}$ are the q-th pixels in p-th reference and decoded frames. Note that in the presence of SEUs, PSNR (given by (8)) is degraded due to alterations in computation registers containing $y_{p,q}$ values. As a result, the SEUs experienced during computation ($\mathcal{F}_{comp}$) has a direct impact on the PSNR. However, due to normalization with decoded frames and pixels PSNR does not reflect temporal fidelity in the event of loss of frames [26]. To evaluate fidelity in the event of frame losses, we use frame error ratio (FER) metric, defined as

$$FER = \frac{x}{P} \quad , \tag{9}$$

where x is the number of lost frames out of P frames. Frame losses during video decoding take place mostly due to misrouting of DTUs between communicating cores. Hence, SEUs experienced during communication ($\mathcal{F}_{comm}$) have a direct impact on the FER [26]. The SEUs experienced during computation ($\mathcal{F}_{comp}$) has a direct impact on PSNR but an indirect impact on FER as computation of video parameters are affected by $\mathcal{F}_{comp}$.

Figure 10(a) and (b) show the PSNR (in dB) and FER (in %) values of decoded video frames found through (8) and (9), while decoding video bitstream *test4.m2v* in AMBA- and NoC-based decoders. The PSNR and FER values of NoC-based decoder are observed for three different NoC configurations: with 2, 3 and 4 intermediate switches between communicating cores. An arbitrary SER of 10^{-9} SEUs per bit per cycle is used in simulated fault injection environment (Section 2.4). As expected, NoC-based decoder outperforms AMBA-based decoder with up to 4dB higher PSNR (Figure 10(a)). This is because NoC-based decoder experiences lower $\mathcal{F}_{comp}$ than AMBA-based decoder (Section 3.1). However, since PSNR does not reflect the fidelity of video blocks due to perturbation of registers by $\mathcal{F}_{comm}$ (and also since the number of intermediate switches does not affect $\mathcal{F}_{comp}$, given by (1)), NoC-based decoder shows similar PSNRs for all configurations. Comparing the FER values in Figure 10, it can be seen that AMBA-based decoder gives 3% lower

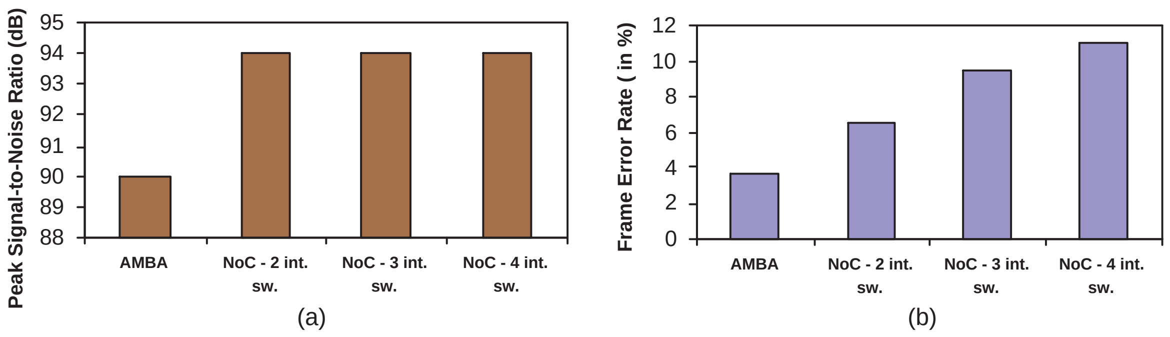


Figure 10: Comparative (a) PSNRs and (b) FERs of AMBA- (Figure 2) and NoC-based decoders (Figure 3(a)) while decoding *test4.m2v*

FER compared to NoC-based decoder configuration with 2 intermediate switches due to higher number of SEUs experienced during communication, $\mathcal{F}_{comm}$ (Section 3.2). As expected, with increased number of intermediate switches, NoC-based decoder experiences higher FER due to increased $\mathcal{F}_{comm}$ given by (3) (Section 3.2). For example, FER of NoC-based decoder increases to from 6.5% to 9.5% and 11% as number of intermediate switches increases from 2 to 3 and 4 (Figure 10).

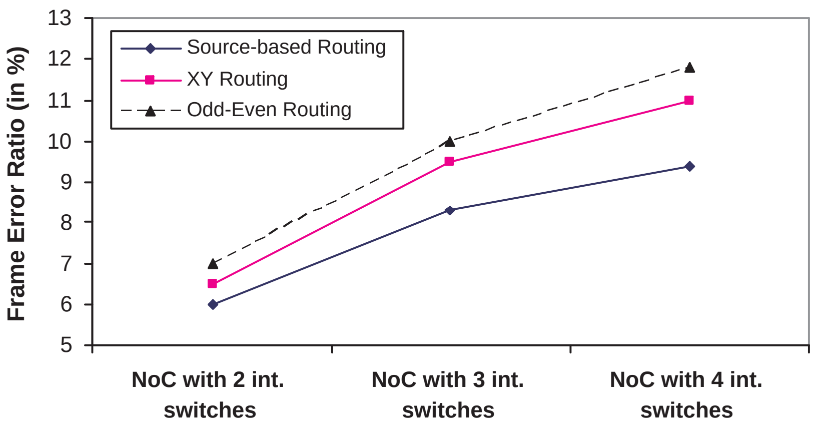


Figure 11: Impact of choice of routing algorithm on the FER of NoC-based decoder, while decoding *test4.m2v*

The FER values of NoC-based decoder in Figure 10(b) are obtained using XY packet routing algorithm. Figure 11 demonstrates the impact of choice of routing algorithm on the FER of the NoC-based decoder (Figure 3(a)), while decoding the video bitstream *test4.m2v* (Table 1). Three different packet routing algorithms are used: source-based, XY and odd-even. FER values are obtained through (9) from decoded video frames in SystemC fault injection environment with an SER of 10^{-9}. As expected, using the source-based packet routing algorithm gives the lowest FER among the routing algorithms due to the lowest $\mathcal{F}_{comm}$ in NoC-based decoder (Section 3.2). Employing XY or odd-even routing algorithm gives higher FER in the decoder due to the higher $\mathcal{F}_{comm}$ (Section 3.2). It can be seen that with increasing number of intermediate switches between communicating cores, the FER of the NoC-based decoder increases almost linearly due to increased $\mathcal{F}_{comm}$, given by (3).

4. Impact of Application Task Mapping and Architecture Allocation

The impact of application task mapping and architecture allocation on system performance in the context of HW/SW co-design has been studied extensively [27]. In this section, the impact of application task mapping and architecture allocation on the reliability of on-chip communication architectures is investigated.

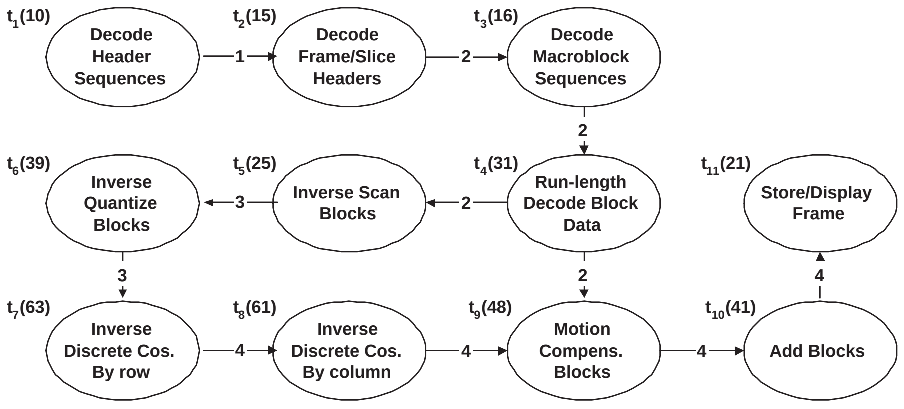


Figure 12: Task graph of MPEG-2 video decoder

4.1. Application Task Mapping

Application task mapping is a crucial design step of MPSoC applications, which involves distribution of the computation and communication tasks among the processing cores and interconnects of an MPSoC architecture. Figure 12 shows the MPEG-2 video decoder task graph showing eleven tasks. Each node represents a computational task weighted by number in parenthesis, indicating the cost in terms of execution time. The edge between nodes represents the communication task shown with cost that describes the time required to transfer data between the tasks with shown directions. All costs are multiples of 5.5×10^6 clock cycles and are obtained through SystemC cycle-accurate simulations assuming 32-bit transfer width. The computational tasks are modelled as separate task processes, while the communication between tasks is modelled as message passing queues. The communication time between tasks is found by dividing the size of inter-task queue by the bandwidth of the channel (in bits per cycle). The effect of mapping the tasks on processing cores on communication costs is not modelled explicitly rather an worst-case approximation is assumed. Similar assumptions have also been used in [28].

Mapping	Core	Mapped Tasks
M1 (Figure 1(a))	Core 1	t_1, t_2, t_3, t_4
	Core 2	t_5, t_6
	Core 3	t_7, t_8
	Core 4	t_9, t_{10}, t_{11}
M2 (optimized for reduced register usage)	Core 1	t_1, t_2, t_3
	Core 2	t_4, t_5
	Core 3	t_6, t_7, t_8, t_9, t_{10}
	Core 4	t_{11}
M3 (optimized for parallelism)	Core 1	t_1, t_2, t_3, t_4, t_9
	Core 2	t_5, t_6, t_7
	Core 3	t_8
	Core 4	t_{10}, t_{11}
M4 (optimized for reduced register usage & parallelism)	Core 1	t_1, t_2, t_3, t_4, t_5, t_6
	Core 2	t_7, t_8
	Core 3	t_9
	Core 4	t_9, t_{10}, t_{11}

Table 5: Four application task mappings of MPSoC decoder using four processing cores (Figure 1)

Numerous mapping combinations are possible for decoder design using the task graph (Figure 12). Table 5 shows four different task mappings of the decoder with the mapped tasks on each processing core. Mapping M1 (row 2) is the mapping employed in Figure 1(a), mapping M2 (row 3) is optimized for reduced register usage, mapping M3 (row 4) is optimized for high parallelism and finally, mapping M4 (row 5) is jointly optimized for reduced register usage and high parallelism. The task mappings M2, M3 and M4 in Table 5 are found through simulated annealing using group-migration based task movement proposed in [28]. As can be seen, mapping M2 localizes most of the the tasks (for example, tasks t_1-t_8 are mapped in core 1) to achieve low overall register usage ($R = \sum_i R_i$), while mapping M3 distributes the tasks among processing cores to optimize for high parallelism. Mapping M4 achieves reduced register usage and high parallelism by carefully distributing the tasks among cores (for example, related tasks t_7 and t_8, which share IDCT parameters and video blocks between them, are mapped in core 2). Figure 13(a) and (b) show the register usages (R) and multiprocessor execution times (T_M) obtained from SystemC cycle-accurate simulations for the AMBA- and NoC-based decoder designs with the tasks mappings (Table 5. As expected, mapping M2 gives the lowest R for AMBA- and NoC-based decoders

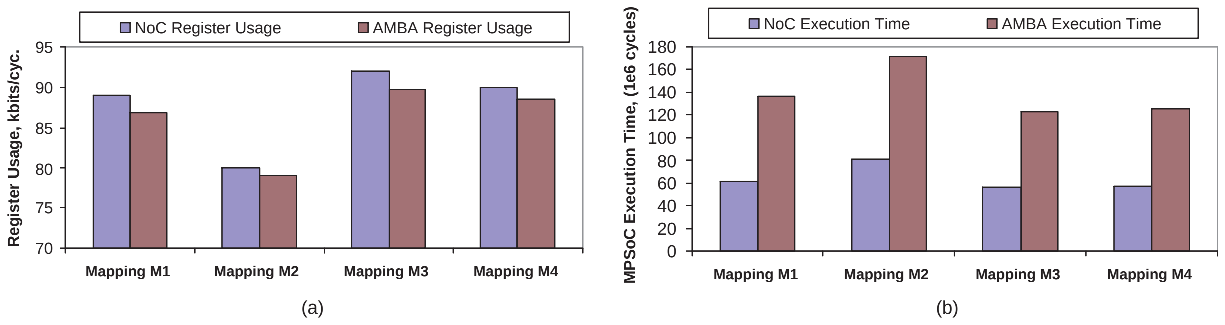


Figure 13: Impact of application task mapping on (b) register usage and (b) multiprocessor execution time of AMBA- and NoC-based decoders, while decoding *test4.m2v*

due to optimization with reduced register usage (Figure 13(a)). However, low R in mapping M2 is obtained at the expense of the highest T_M caused by localization of application tasks (Figure 13(b)). Mapping M3 gives the lowest T_M due to high parallelism among the processing cores for both decoders. Since such low T_M is achieved through distribution of tasks among cores to give higher parallelism, shared register resources among these tasks are duplicated in processing cores. As a result, mapping M3 gives the highest R. Mapping M4 offers a good trade-off between R and T_M. It can be seen that AMBA-based decoder has lower R compared to NoC-based decoder due to contention of registers over idle period during bus arbitration (Section 3.1). As expected, T_M is high for AMBA-based decoder due to shared-access of bus and hence lower concurrency among processing cores [3] (Figure 13(b)).

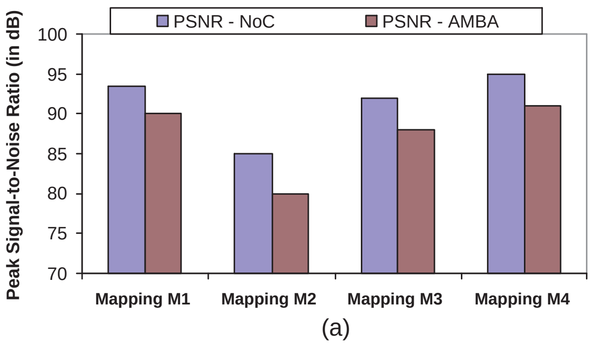
To demonstrate the impact of application task mapping on reliability, Table 6 shows the SEUs experienced during computation ($\mathcal{F}_{comp}$) and communication ($\mathcal{F}_{comm}$) of AMBA- and NoC-based decoders (with 2 intermediate switches in NoC-based decoder), while decoding different video bitstreams (Table 1). The $\mathcal{F}_{comp}$ and $\mathcal{F}_{comm}$ values are obtained through SystemC fault simulation

(Section 2.4) using an SER of 10^{-9}. Mapping M2, M3 and M4 results are shown in column 3-5 (Table 6) and mapping M1 results are shown in Figure 7. As can be seen, mapping M2 experiences

Video	Arch.	Mapping M2		Mapping M3		Mapping M4	
		$\mathcal{F}_{comp}$	$\mathcal{F}_{comm}$	$\mathcal{F}_{comp}$	$\mathcal{F}_{comm}$	$\mathcal{F}_{comp}$	$\mathcal{F}_{comm}$
test1.m2v	NoC	5.21E+02	1	4.44E+02	1	4.12E+02	1
	AMBA	9.71E+02	0	8.14E+02	0	7.43E+02	0
test1.m2v	NoC	1.83E+03	3	1.51E+03	3	1.40E+03	3
	AMBA	3.43E+03	0	2.76E+03	0	2.51E+03	0
test1.m2v	NoC	3.15E+03	5	2.67E+03	5	2.50E+03	5
	AMBA	5.84E+03	2	4.78E+03	2	4.45E+03	2
test1.m2v	NoC	3.85E+04	23	3.76E+03	22	3.71E+03	23
	AMBA	7.26E+04	3	6.97E+03	3	6.73E+03	3

Table 6: Impact of application task mapping on the reliability of AMBA- and NoC-based decoders in terms of $\mathcal{F}_{comp}$ and $\mathcal{F}_{comm}$

the highest $\mathcal{F}_{comp}$ among all task mappings. For example, mapping M2 experiences 18% and 27% higher $\mathcal{F}_{comp}$ compared to mappings M3 and M4, while decoding video bitstream *test1.m2v*. Similar trend is also observed while decoding other video bitstreams (rows 3-5, Table 6). The higher $\mathcal{F}_{comp}$ in mapping M2 is due to reduced register usage (R) through localization of the tasks on a processing core. Such localization causes high multiprocessor execution time (T_M) and leads to high $\mathcal{F}_{comp}$, given by (1). Mapping M3 also experiences higher $\mathcal{F}_{comp}$ than mapping M4 due to increased register usage (R) through duplication of shared registers. Due to joint optimization with reduced register usage (R) and high parallelism, mapping M4 provides the lowest $\mathcal{F}_{comp}$. Note that $\mathcal{F}_{comm}$ does not vary for different task mappings as the total number of DTUs communicated among processing cores, $\mathcal{N} = \sum_i \mathcal{N}_i$, does not vary significantly while decoding a given video bitstream. Figure 14 shows the impact of $\mathcal{F}$ (given by (7)) at application-level in terms of PSNRs and FERs of AMBA- and NoC-based decoders, while decoding *test4.m2v*. The PSNR and FER values were obtained using (8) and (9) from the decoded videos using SER of 10^{-9} in SystemC fault injection environment (Section 2.4). As expected, mapping M2 gives the lowest PSNR (79dB and 85dB for AMBA- and NoC-based decoders) due to the highest $\mathcal{F}_{comp}$ (Figure 14(a)). Due to lower $\mathcal{F}_{comp}$, mapping M3 gives up to 7dB higher PSNR compared to mapping M2. Mapping M4 gives the best PSNR (91dB for AMBA-based decoder and 95dB for NoC-based decoder) when compared to the other three mappings due to the lowest $\mathcal{F}_{comp}$. From Figure 14(b) it can be seen



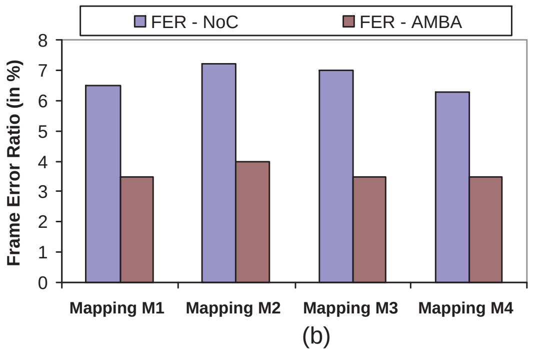


Figure 14: Impact of task mapping on decoder reliability in terms of (a) PSNR and (b) FER, while decoding *test4.m2v*

that, despite similar $\mathcal{F}_{comp}$ values for different mappings (Table 6), the FER is higher (7.5%) for mapping M2 due to incorrect computation of video parameters with high $\mathcal{F}_{comm}$. With low $\mathcal{F}_{comm}$, mapping M4 gives the least FER (6.2%) among all task mappings.

4.2. Architecture Allocation

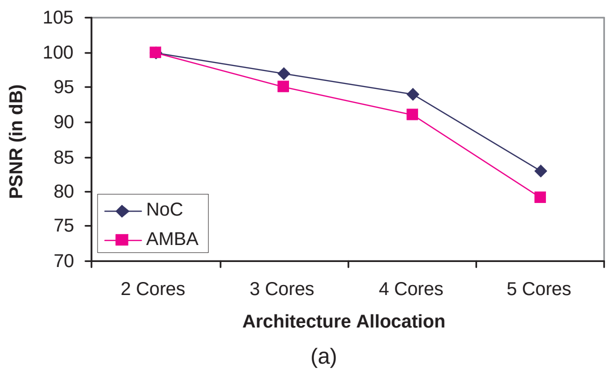
Architecture allocation is a system-level design step for MPSoCs that deals with allocation of processing elements and their interconnects into the architecture [29]. In this section, we refer to architecture allocation as allocation of number of computation cores in the MPSoC decoder (Figure 1). To investigate the impact of architecture allocation on reliability, different number of allocated cores were simulated using mapping M4 (Section 4.1). Table 7 shows the number of SEUs experienced during computation ($\mathcal{F}_{comp}$) and communication ($\mathcal{F}_{comm}$) by the AMBA- and NoC-based decoders for architecture allocations of 2, 3 and 5 cores (simulation results for architecture allocation of 4 cores are shown in Tables 3 and 4). The number of SEUs experienced during computation ($\mathcal{F}_{comp}$) and the number of SEUs experienced during communication ($\mathcal{F}_{comm}$) for architecture with 2 allocated cores is shown in column 3, while that of 3 and 5 allocated cores are shown in columns 4 and 5 (Table 7). As expected, NoC-based decoder experiences less number of SEUs during computation ($\mathcal{F}_{comp}$) than AMBA-based decoder, while AMBA-based decoder experiences less SEUs during communication ($\mathcal{F}_{comm}$) for all architecture allocations (Sections 3.1 and 3.2). It can be seen that both AMBA- and NoC-based decoders experience higher $\mathcal{F}_{comp}$ as the number of allocated cores increases in the architectures. This is because with higher number of allocated cores, the overall register usage ($R = \sum_i R_i$) increases due to duplication of shared resources, resulting in higher $\mathcal{F}_{comp}$ given by (1). Also, with increased architecture allocation

Video	Arch.	2 Cores		3 Cores		5 Cores	
		$\mathcal{F}_{comp}$	$\mathcal{F}_{comm}$	$\mathcal{F}_{comp}$	$\mathcal{F}_{comm}$	$\mathcal{F}_{comp}$	$\mathcal{F}_{comm}$
test1.m2v	NoC	3.94E+2	1	4.44E+2	1	5.45E+2	1
	AMBA	7.02E+2	0	7.97E+2	0	9.78E+2	0
test2.m2v	NoC	1.20E+3	3	1.36E+3	4	1.66E+3	4
	AMBA	2.16E+3	0	2.43E+3	0	2.93E+3	1
test3.m2v	NoC	2.03E+3	5	2.34E+3	6	2.83E+3	8
	AMBA	3.65E+3	1	4.28E+3	2	5.19E+3	3
test4.m2v	NoC	3.06E+4	22	3.48E+3	25	4.21E+3	31
	AMBA	5.51E+3	2	6.24E+3	3	7.44E+3	5

Table 7: Impact of architecture allocation on the reliability in terms of number of SEUs experienced

with more processing cores $\mathcal{F}_{comm}$ increases as the number of inter-core communication links (M) increases (due to (3)). For example, NoC-based decoder with 5 processing cores experiences up to 20% higher $\mathcal{F}_{comp}$ and 9% higher $\mathcal{F}_{comm}$ than NoC-based decoder with 3 processing cores for decoding *test4.m2v*.

To observe the impact of total number of SEUs experienced at application-level, Figure 15 shows the corresponding PSNRs and FERs of different architecture allocations of AMBA- and NoC-based decoders for decoding *test4.m2v*. The PSNR and FER values were found at SER of 10^{-9} while decoding the video bitstream *test4.m2v* in simulated fault injection environment (Section 2.4). As can be seen, NoC-based decoder gives better PSNR for all architecture allocations



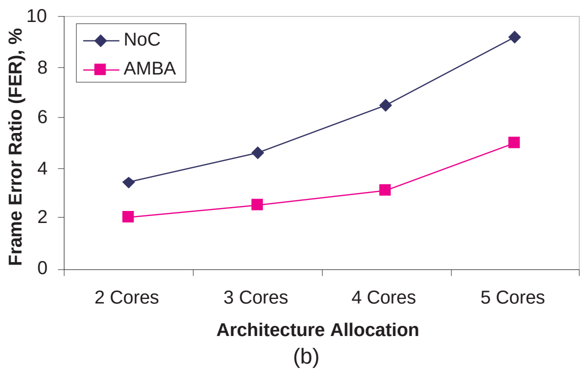


Figure 15: (a) Comparative PSNRs of AMBA- and NoC-based decoders for different architecture allocations while decoding *test4.m2v*, (b) Comparative FERs of AMBA- and NoC-based decoders for different architecture allocations while decoding *test4.m2v*

(Figure 15(a)) due to lower number of SEUs experienced during computation, $\mathcal{F}_{comp}$ (Section 3.1). Due to increased $\mathcal{F}_{comp}$ for increasing number of allocated cores, architecture with higher number of cores give poorer PSNRs for AMBA- and NoC-based decoders. For example, PSNR decreases from 99dB for architecture with 2 cores to 84dB for architecture with 5 cores for AMBA-based decoder (Figure 15(a)). As expected, decoder architecture with higher number of cores gives higher FER due to increased $\mathcal{F}_{comm}$ (Table 7). For example, FER increases from 2% for architecture with 2 cores to 4% in the case of NoC size 2×2 (4 cores) and 4.5% for architecture with 5 cores for AMBA-based decoder (Figure 15(b)). Note that AMBA-based decoder gives lower FER when compared with NoC-based decoder due to lower number of SEUs experienced during communication, $\mathcal{F}_{comm}$ (Section 3.2).

The comparative analysis carried out so far is based on architecture allocation of 2, 3, 4 and 5 cores. It was observed that architecture allocation affects the analysis parameters: per core execution time (T_i), register usage (R_i) and inter-core communication links ($\mathcal{N}$). As a result for architecture allocation with higher number of allocated cores (for example 4×4 NoC size) the SEUs experienced during computation ($\mathcal{F}_{comp}$, given by (1)) will increase for AMBA- and NoC-based decoders. Due to higher execution times AMBA-based decoder will experience higher $\mathcal{F}_{comp}$ compared to NoC-based decoder. On the other hand the SEUs experienced during communication ($\mathcal{F}_{comm}$, given by (3)) will increase for NoC-based decoder due to increased number of inter-core links ($\mathcal{N}$). Also, for NoC-based decoder choice of routing algorithm is expected to affect $\mathcal{F}_{comm}$ (Section 3.2). The impact of total SEUs experienced during computation and communication ($\mathcal{F} = \mathcal{F}_{comp} + \mathcal{F}_{comm}$) at application-level will be reflected as higher PSNR and FER for NoC-based decoder compared to AMBA-based decoder as shown in Figures 15(a) and (b).

5. Summary of Comparisons

From the comparative analysis (Sections 3 and 4) the following observations are made:

1. For a given architecture allocation and soft error rate (SER) AMBA-based decoder experiences higher number of SEUs during computation than NoC-based decoder. This is because AMBA-based decoder has higher execution time than NoC-based decoder due to shared bus access in AMBA (Section 3.1).

2. NoC-based decoder experiences higher SEUs during inter-core communication than AMBA-based decoder. This is because NoC-based decoder has higher channel latency and register usage in communication interconnects. The source-based routing with shortest path mapping between cores (2 intermediate switches between cores) gives the minimum number of SEUs experienced during communication in NoC-based decoder (Section 3.2).
3. Considering the impact of total number of SEUs experienced at application-level, NoC-based decoder exhibits higher error resilience in terms of peak signal-to-noise ratio (PSNR) compared to AMBA-based decoder. However, it suffers from higher frame error ratio (FER) due to higher SEUs experienced during communication (Section 3.3).
4. For a given architecture allocation, application task mapping (the distribution of application tasks among cores) affects the total number of SEUs experienced by AMBA- and NoC-based decoders. To minimize the number of SEUs experienced careful choice of application task mapping is needed (Section 4.1).
5. With increased number of allocated cores in architecture allocation of the decoders, the number of SEUs experienced during computation and communication increases for AMBA- and NoC-based decoders (Section 4.2).

6. Conclusions

Using MPEG-2 video decoder as a case study in simulated fault injection environment, we have presented a comparative reliability analysis between shared-bus AMBA and NoC. We have shown that AMBA-based decoder experiences higher SEUs during computation than NoC-based decoder due to higher execution time than NoC-based decoder (Section chap4:results:computation). We have also shown that NoC-based decoder experiences higher SEUs during inter-core communication than AMBA-based decoder due to higher channel latency and register usage in communication interconnects (Section 3.2). Considering the impact of SEUs at application-level, we have shown that NoC-based decoder is more error resilient (in terms of peak signal-to-noise ratio) compared to AMBA-based decoder but it suffers from higher frame error ratio due to higher SEUs experienced during communication (Section 3.3). Furthermore, we have investigated the impact of routing, application task mapping and architecture allocation on the reliability of the decoders in the presence of SEUs (Section 4). It is hoped that the findings in this work would contribute towards the current research efforts in identifying appropriate on-chip communication architecture for emerging multimedia applications.

References

[1] L. Benini, G.D. Micheli, Networks on Chips: A New SoC Paradigm, Computer 35 (1) (2002) 70–78.

[2] H. Lee, N. Chang, U. Ogras, R. Marculescu, On-chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-point, Bus, and Network-on-chip Approaches, ACM Transactions on Design Automation of Electronic Systems 12 (3) (2007) 1–20.

[3] R. Shafik, P. Rosinger, B. Al-Hashimi, MPEG-based Performance Comparison between Network-on-Chip and AMBA MPSoC, in: Proceedings of Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2008, pp. 98–103.

[4] D. Flynn, AMBA: Enabling Reusable On-chip Design, IEEE Micro 17 (4) (1997) 20–27.

[5] W. J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers, 2004, San Francisco, CA, USA.

[6] K. Goossens, J. Dielissen, A. Radulescu, ÆTHEREAL Network on Chip: Concepts, Architectures, and Implementations, IEEE Design & Test of Computers 22 (5) (2005) 414– 421.

[7] M. Millberg, E. Nilsson, R. Thid, A. Jantsch, Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks Within the NOSTRUM Network-on-Chip, in: Proceedings of Design, Automation and Testing in Europe Conference (DATE), IEEE, 2004, pp. 890–895.

[8] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Schanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jairr, S. Venkataramarr, Y. Hoskote, N. Borkar, An 80 Tile 1.28 TFLOPs Network-on-Chip in 65 nm CMOS, in: Proceedings of International Solid State Circuit Conference (ISSCC), 2007, pp. 98-100.

[9] A. Ejlali, B. M. Al-Hashimi, P. Rosinger, S. G. Miremadi, Joint Consideration of Fault-Tolerance, Energy-Efficiency and Performance in On-Chip Network, in: Proceedings of the Conference on Design, Automation and Test in Europe (DATE), 2007, pp. 1647–1652.

[10] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, C. R. Das, Exploring Fault-Tolerant Network-on-Chip Architectures, in: International Conference on Dependable Systems and Networks (DSN), 2006, pp. 94–104.

[11] R. Melhem, D. Mosse, and E. Elnozahy. The interplay of power management and fault recovery in real-time systems. *IEEE Transactions on Computers*, 53(2):217–231, February, 2004 2004.

[12] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, C. R. Das, Design and Analysis of an NoC Architecture from Performance, Reliability and Energy Perspective, in: Proceedings of the ACM Symposium on Architecture for Networking and Communications Systems, 2005, pp. 173–182.

[13] R. R. Tamhankar, S. Murali, G. D. Micheli, Performance Driven Reliable Link Design for Networks on Chips, in: Proceedings of the Conference on Asia South Pacific Design Automation, 2005, pp. 749–754.

[14] K. Mihic, T. Simunic, G.D. Micheli, Reliability and Power Management of Integrated Systems, in: Proceedings of EUROMICRO Digital System Design (DSD), August, 2004, pp. 5–11.

[15] T. Dumitraş, S. Kerner, R. Mărculescu, Towards On-chip Fault-tolerant Communication, in: Proceedings of the 2003 Asia and South Pacific Design Automation Conference (ASPDAC), Kitakyushu, Japan, 2003, pp. 225–232.

[16] A. Hosseini, T. Ragheb, Y. Massoud, A Fault-aware Dynamic Routing Algorithm for On-chip Networks, in: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Washington, USA, 2008, pp. 2653–2656.

[17] AMBA: Advanced Microprocessor Bus Architecture Specification, v2.0, 1999, http://www.arm.com.

[18] S. Kumar, A. Jantsch, M. Millberg, J. Oberg, J. Soininen, M. Forsell, K. Tiensyrja, A. Hemani, A Network on Chip Architecture and Design Methodology, in: Proceedings of IEEE Computer Society Annual Symposium on VLSI, 2002, p. 117.

[19] NIRGAM: A Dynamic SystemC Simulator for NoC, http://nirgam.ecs.soton.ac.uk.

[20] R. A. Shafik, P. Rosinger, B. M. Al-Hashimi, SystemC-based Minimum Intrusive Fault Injection Technique with Improved Fault Representation, in: Proceedings of International On-Line Testing Symposium (IOLTS), Rhodes, Greece, 2008, pp. 99–104.

[21] X. Li, D. Yeung, Exploiting Application-level Correctness for Low-Cost Fault Tolerance, Journal of Instruction-level Parallelism 10 (2008) 1–28.

[22] A. Dalirsani, M. Hosseinabady, Z. Navabi, An Analytical Model for Reliability Evaluation of NoC Architectures, in: IOLTS '07: Proceedings of the 13th IEEE International On-Line Testing Symposium, IEEE Computer Society, Washington, DC, USA, 2007, pp. 49–56.

[23] F. Angiolini, P. Meloni, S. M. Carta, L. Raffo, L. Benini, A Layout-aware Analysis of Networks-on-Chip and Traditional Interconnects for MPSoCs, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 26 (3) (2007) 421–434.

[24] P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, Performance Evaluation and Design Trade-offs for Network-on-Chip Interconnect Architectures, Transactions on Computers 54 (8) (2005) 1025–1040.

[25] A. Jalabert, S. Murali, L. Benini, G. D. Micheli, xpipesCompiler: A Tool for Instantiating Application Specific Networks on Chip, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'04), Vol. 2, 2004, pp. 884–889.

[26] P. Cherriman, L. Hanzo, Error-Rate-based Power-Controlled Multimode H.263-assisted Videotelephony, IEEE Transactions on Vehicular Technology 48 (5) (1999) 1726–1738.

[27] A. Mohsen, R. Hofmann, Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation, Vassilis Paliouras, Johan Vounckx, Diederik Verkest (Ed.s), Birkhuser, 2005, Ch. Power-Aware Scheduling for Hard Real-Time Embedded Systems using Voltage Scaling Enabled Architectures, pp. 127–136.

[28] H. Orsilla, T. Kangas, E. Salminen, T. Hmlinen, M. Hnnikinen, Automated Memory-aware Application Distribution for Multi-processor System-on-Chips, Journal of Systems Architecture: the EUROMICRO Journal 53 (11) (2007) 795–815.

[29] M. T. Schmitz, B. M. Al-Hashimi, P. Eles, System-Level Design Techniques for Energy-Efficient Embedded Systems, Kluwer Academic Publisher, 2004.