Comparative Reliability Analysis between AMBA and Network-on-Chip: An MPEG-2 Case Study

Rishad A. Shafik, Bashir M. Al-Hashimi
School of ECS, University of Southampton, UK, SO17 1BJ
Email: {*ras06r, bmah*}*@ecs.soton.ac.uk*

***Abstract*— We present comparative reliability analysis between shared-bus AMBA and network-on-chip (NoC) in the presence of single-event upsets (SEUs) using MPEG-2 video decoder as a case study. Employing SystemC-based cycle-accurate fault simulations, we investigate how the decoder reliability is affected when SEUs are injected into the computation cores and communication interconnects of the decoder. We show that for a given soft error rate, AMBA-based decoder experiences higher SEUs during computation due to higher execution time. On the other hand, NoC-based decoder experiences higher SEUs during inter-core communication due to higher channel latency and resource usage in the interconnects. Furthermore, we evaluate the impact of total SEUs at application-level for NoC- and AMBA-based decoders.**

I. Introduction

Comparative analyses in terms of performance, latency, scalability, power, area and throughput between AMBA and network-on-Chip (NoC) have recently been reported, such as [1]. Reliability against soft errors is an emerging design objective for these systems, particularly due to exacerbation of single-event upsets (SEUs) with technology scaling [2]. To this end, a number of studies have recently been reported showing NoC fault tolerant architectures and techniques, examples [3], [4]. To our knowledge, no comparative studies between shared-bus and NoC has yet been reported considering reliability using real-application, which is the main aim of this paper. Using SEU as fault model in simulated fault injection environment, we investigate the number of SEUs experienced during computation and communication in shared-bus AMBA- and NoC-based decoder and evaluate the impact of SEUs at application-level. The rest of the paper is organized as follow. Section II describes system and fault injection setup used in this work. Section III compares between AMBA- and NoC-based decoder in terms injected SEUs and its impact on MPEG-2 video decoder. Section IV concludes the paper.

II. System Setup

In this section, AMBA- and NoC-based MPEG-2 video decoder and fault injection technique are explained briefly.

A. MPEG-2 Video Decoder Cores

MPEG-2 video decoder constitutes a major component of MPSoC applications and is chosen as an application case study. Fig. 1(a) shows block diagram of the MPEG-2 video decoder with four processing cores used in this work. Behavioral modeling was used to design the decoder cores. Partitioning and allocation of application are performed arbitrarily to reflect

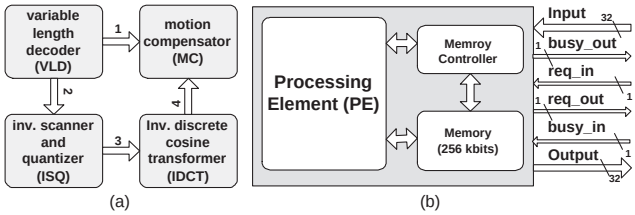

Fig. 1: Block diagram of (a) MPEG-2 video decoder MPSoC, and (b) processing core used in MPEG-2 video decoder

TABLE I: Video bitstreams used for comparisons in this work

Video	Frames	Bitrate	Frame Size (pixels)
test1.m2v (tennis)	67	4 Mbps	176×120 (QCIF, NTSC)
test2.m2v (flower)	55	5.2 Mbps	352×288 (CIF, PAL)
test3.m2v (tennis)	49	7 Mbps	352×576 (2CIF, PAL)
test4.m2v (flower)	73	7 Mbps	704×480 (4CIF, NTSC)

Source: ftp://ftp.tek.com/tv/test/streams/Element/; video split by VLC Media Player: http://www.videolan.org/vlc/

MPSoC. Fig. 1(b) shows a block diagram of a processing core used in Fig. 1(a). Each core consists of 32-bit input/output interface for transfer of data transaction units (DTUs), handshake signals and a private memory (of 256 kbits) interfaced by a memory access controller (Fig. 1(b)). The memory size is chosen to give high availability for data processing and storage within processing cores. Table I shows four video bitstreams with different resolutions and sizes, which are used for comparisons in Section III.

B. Shared-bus AMBA

Advanced high-performance bus (AHB) is used as shared-bus AMBA in this work due to its high performance [7]. A single-layer central multiplexor configuration with pipelined single-burst transfer and no waiting states are used to maximize throughput. MPEG-2 video decoder cores (Fig. 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 Fig. 2. The cores share bus access in the sequence of cores VLD, ISQ and IDCT (Fig. 2) with each core holding the interconnect access until the current macroblock is processed and stored in the memory of the next core. To facilitate AMBA-based cycle-accurate simulations, we use Synposys Designware SystemC libraries[1].

C. Network-on-Chip

Network-on-Chip (NoC) incorporates packet-based on-chip communication with a large design space arising out of various

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

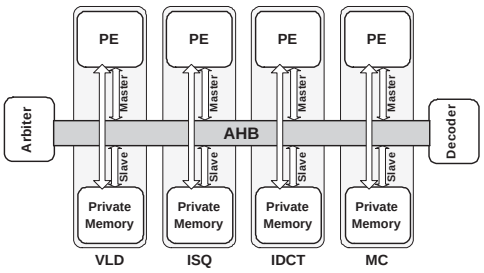

Fig. 2: Shared-bus AMBA-based decoder used in this work

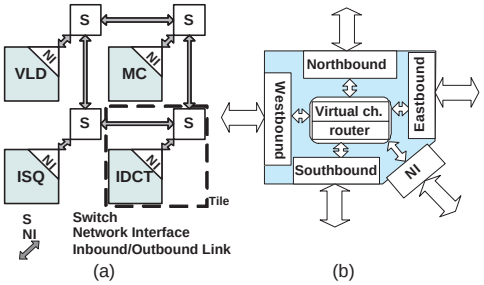

Fig. 3: (a) Mesh-based (2×2) NoC employing MPEG-2 decoder cores, (b) 5-port NoC switch used in this work

options with routing techniques and topologies [3]. In this work, we use a mesh-based NoC topology with with deterministic XY routing and single-flit-packet *wormhole* communication due to simplicity of switch design, performance and scalability [8], as shown in Fig. 3(a). As can be seen, NoC architecture is made up of tiles (Fig. 3(a)) consisting of MPEG-2 decoder cores, switch for communication and network interface (NI) for packets-based interface with 46-bit overhead (i.e. size of NoC packet is 32+46=78 bits). Fig. 1(b) shows block diagram of switch architecture used with five inbound and outbound ports. Four ports connect with other switches with buffer for eight packets on channels and one port is laid out between PE and NI with buffer for four packets (Fig. 3(b)). Virtual channel (VC) provides buffering for eight incoming packets and router selects output port based on routing technique used (Fig. 1(b)). To facilitate NoC simulations, we use SystemC-based cycle-accurate simulator NIRGAM [9].

D. Fault Injection

In this work, fault injection is carried out using SEU-based fault model (as also used in [6]) employing technique proposed in [10]. The fault injection is initiated by replacing all original data/signal types to fault injection enabler types. Such data/signal types facilitate a centralized list of register space for fault injections. For a given soft error rate (SER, in number of SEUs per bit per cycle), the number of SEUs to be injected within this register space is found and their locations are determined by Poisson distribution. Using simulation-specific monitor modules, total register usage and number of faults injected can be found. More details of fault injection can be found in [10].

III. Comparative Reliability Analysis

Reliability of an application against SEUs is related to the total number of SEUs experienced over a given time [6]. Our aim in this work is to analyze how the reliability of MPEG-2 video decoder is affected by the choice of AMBA and NoC on-chip communication architecture. To this end, we carry out the following investigations:

- SEUs experienced during *computation*, $\mathcal{F}_{comp}$, to show how computation is affected,
- SEUs experienced during inter-core *communication*, $\mathcal{F}_{comm}$, to show how inter-core communication is affected, and
- the impact of total SEUs, $\mathcal{F} = \mathcal{F}_{comp} + \mathcal{F}_{comm}$, at an application-level.

In the following (Section III-A and III-B), $\mathcal{F}_{comp}$ and $\mathcal{F}_{comm}$ of AMBA- and NoC-based decoder are investigated and compared. Later (Section III-C), the impact of $\mathcal{F}$ is evaluated at application-level.

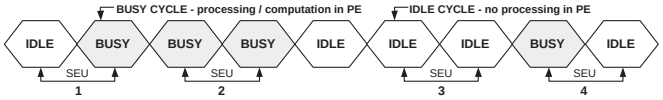

Fig. 4: Manifestation of SEUs during computation cycles

A. SEUs Experienced During Computation

The SEUs in registers of processing cores manifest themselves in different ways during computation, as shown in Fig. 4. As can be seen, SEUs extending between two *idle* cycles (instance 3) do not affect computation process, while 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 (Fig. 4). Hence, for a given SER of λ, effective $\mathcal{F}_{comp}$ in MPEG-2 video 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 T_i is the execution time (in clock cycles), T_i^{I-I} is the number of idle-to-idle transitions within T_i and R_i gives a measure of actual register usage of the application (since SEUs in other registers have no impact [6]) and is found by

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

In (2), $R_{i,t}$ is the instantaneous number of registers (in bits) used or occupied by MPEG computation process at t-th clock cycle in i-th processing core. Table II shows execution time, T_i, idle-idle transition cycles, T_i^{I-I} and register usage, R_i of each processing core in AMBA- (Fig. 2) and NoC-based decoder (Fig. 3(a)) obtained through simulations. As can be seen from Table II (row 2), AMBA-based decoder has similar register usage, R_i, as NoC for all four cores but 2.08, 1.98, 2.15 and 2.18 times higher execution time, T_i, in processing cores VLD, ISQ, IDCT and MC, respectively,

TABLE II: 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 decoder

Video	Arch.	Core VLD			Core ISQ			Core IDCT			Core MC		
		T_i, cyc. ($\times 10^6$)	T_i^{I-I}, cyc. ($\times 10^6$)	R_i, kb/c.	T_i, cyc. ($\times 10^6$)	T_i^{I-I}, cyc. ($\times 10^6$)	R_i, kb/c.	T_i, cyc. ($\times 10^6$)	T_i^{I-I}, cyc. ($\times 10^6$)	R_i, kb/c.	T_i, cyc. ($\times 10^6$)	T_i^{I-I}, cyc. ($\times 10^6$)	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

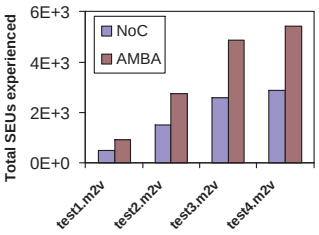

Fig. 5: Comparative $\mathcal{F}_{comp}$ in AMBA- and NoC-based decoder for an arbitrary SER of 10^{-9}

due to time-sharing of bus access among processing cores while decoding $test1.m2v$. The time sharing of bus access causes more idle cycles in AMBA-based decoder, resulting in 5.7, 4.6, 2.8 and 6.9 times higher T_i^{I-I} compared to NoC-based decoder for processing cores VLD, ISQ, IDCT and MC, respectively (row 2, Table II). With increased video sizes ($test2.m2v$, $test3.m2v$ and $test4.m2v$, Table II), T_i and T_i^{I-I} increase but similar trend continues between AMBA- and NoC-based decoder for R_i, T_i and T_i^{I-I} values. This results in higher $\mathcal{F}_{comp}$ in AMBA compared to NoC for decoding video different bitstreams (Table I), as shown in Fig. 5 (found using (1) with T_i, T_i^{I-I} and R_i values from Table II and for an arbitrary SER of 10^{-9} SEUs/bit/cycle). As expected, AMBA-based decoder experiences approximately 83% higher $\mathcal{F}_{comp}$ compared to NoC, while decoding video bitstream $test3.m2v$ due to higher execution time (Fig. 5). Higher $\mathcal{F}_{comp}$ in AMBA-based decoder is expected to perturb MPEG-2 computation more than NoC-based decoder, which will be examined at application-level in Section III-C.

B. SEUs Experienced During Communication

The SEUs experienced during communication, $\mathcal{F}_{comm}$, depends on how data transfer is carried out between communicating cores and is given by

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

where M is the number of inter-core communication links within the MPSoC ($M = 4$, Fig. 1(a)), N_j is the total number of data transaction units (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. Channel latency, L_{ch_j} in (3) gives a measure of contention of DTUs within the channels 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 as

$$L_{ch} = \tau_{c-in}^{S} + \tau_{in-in}^{S-D} + \tau_{in-c}^{D} + \tau_{wait} \quad , \tag{4}$$

where τ_{c-in}^{S} is the DTU traveling time from source core output port to source interconnect (AHB for AMBA and switch in NoC) port, τ_{in-in}^{S-D} is DTU traveling time from source interconnect port to destination interconnect port and τ_{in-c}^{D} is the DTU traveling time from destination interconnect port to the destination core and τ_{wait} is optional waiting time (τ_{wait}=0 in this work, Section II-D). From simulation logs, L_{ch} can be found by (4) as 2 cycles for AMBA and 9 cycles for NoC using shortest path mapping of core with 2 intermediate switches between cores (Fig. 3). Note that for NoC, L_{ch} increases as more switches are traveled by DTUs due to choice of routing or mapping technique. For example, L_{ch} increases to 15 and 20 clock cycles with 3 and 4 intermediate switches, respectively. The average register usage of communication components during DTU transfer, R_{com_j} in (3) sets up another difference between AMBA- and NoC-based decoder, given by

$$R_{com_j} = \frac{1}{\left(L_{ch_j} N_j\right)} \sum_{n=1}^{N_j} \sum_{l=1}^{L_{ch_j}} R_{n,l} \quad , \tag{5}$$

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 (5) 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 (5), R_{com_j} in NoC-based decoder obtained from simulation logs is approximately 212 bits per data transfer cycle and that in AMBA-based decoder is approximately 87 bits per transfer cycle (details not shown due to space constraint).

TABLE III: Inter-core data transaction units (DTUs) for decoding different video bitstreams

Video	N_1, $\times 10^3$	N_2, $\times 10^3$	N_3, $\times 10^3$	N_4, $\times 10^3$
test1.m2v	66	78	108	202
test2.m2v	232	273	364	666
test3.m2v	385	454	605	1111
test4.m2v	1598	1884	2503	4580

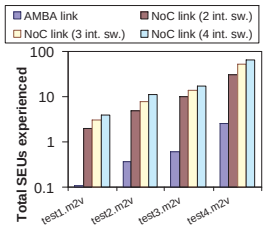

Fig. 6: Comparative $\mathcal{F}_{comm}$ in AMBA and NoC links

Table III shows the number of DTUs, N_i (N_1 for VLD-MC link, N_2 for VLD-ISQ link, N_3 for ISQ-IDCT link, and N_4 for IDCT-MC link, Fig. 1(a)), recorded from simulation logs. Note that N values do not change for AMBA- and NoC-based decoder for a given video bitstream due to similar architecture for processing cores (Fig. 1(a)). As video sizes increase, N also increases for a given link. For example, 108×10^3 DTUs are transferred from core ISQ to IDCT, while decoding $test1.m2v$ compared to 364×10^3 DTUs on the same

link, while decoding $test2.m2v$ (column 4, Table III). Fig. 6 shows comparative $\mathcal{F}_{comm}$ for an arbitrary SER of 10^{-9}, while decoding different video bitstreams (Table I) using above L_{ch} and R_{com} values for AMBA- and NoC-based decoder (with links having 2, 3 or 4 intermediate switches). As expected, due to higher register usage (R_{com_j}) and channel latency (L_{ch}), NoC-based decoder links with 2 intermediate switches (Fig. 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 $test1.m2v$ (Fig. 6). Similar trends between AMBA- and NoC-based decoder in terms of $\mathcal{F}_{comm}$ are also observed with other video bitstreams (Fig. 6). Although, $\mathcal{F}_{comm} \ll \mathcal{F}_{comp}$ (Fig. 5 and 6), $\mathcal{F}_{comm}$ leads to high-level faults including misrouting or loss of DTUs and are of higher significance [5]. Next, the impact of injected SEUs is evaluated at application-level.

C. Impact of SEUs at Application-Level

In Section III-A and III-B, NoC- and AMBA-based decoders have been compared in terms of total SEUs experienced at architectural-level during computation ($\mathcal{F}_{comp}$) and communication ($\mathcal{F}_{comm}$). In this section, we evaluate the impact of total SEUs ($\mathcal{F} = \mathcal{F}_{comp} + \mathcal{F}_{comm}$) at application-level using peak signal-to-noise ratio (PSNR) as the application fidelity metric. PSNR is used to characterize subjective quality of video in the presence of SEUs [6] (also used in [6]) and is defined as

$$PSNR = 10 \log_{10} \frac{1}{PQ} \sum_{p=1}^{P} \sum_{q=1}^{Q} \frac{255^2}{(x_{p,q} - y_{p,q})^2} \quad , \quad (6)$$

where P is the number of frames, Q is number of pixels per frame, $x_{p,q}$ and $y_{p,q}$ are the q-th pixels in p-th reference and decoded frames. PSNR describes fidelity of video decoder but it does not reflect temporal fidelity [6]. Hence, to evaluate temporal fidelity, we use frame error ratio (FER), defined as

$$FER = \frac{x}{P} \quad , \quad (7)$$

where x is the number of lost frames out of P frames. Fig. 7 shows PSNR (in dB) and FER (in %) of decoded video frames found by using (6) and (7), while decoding video bitstream $test4.m2v$ in AMBA- (Fig. 2) and NoC-based decoder (with links having 2 intermediate switches, Fig. 3(a)). The PSNR and FER results were obtained by varying SER from 10^{-10} to 10^{-6} in simulated fault injection environment (Section II-D). As expected, NoC outperforms with upto 5dB higher PSNR (Fig. 7) due to lower $\mathcal{F}_{comp}$ (Section III-A). However, due to higher $\mathcal{F}_{comm}$ (Section III-B), NoC-based decoder has upto 3% higher FER compared to AMBA-based decoder. To investigate and compare the impact of NoC routing or floor mapping (as demonstrated in Section III-B), Fig. 8 shows the FERs for AMBA- and NoC-based decoder with routing of 2 (Fig. 3(a)), 3 and 4 intermediate switches for a given SER of 10^{-9}, while decoding $test4.m2v$. As expected, AMBA-based decoder has the lowest FER of 3.6% (Fig. 8). Due to higher $\mathcal{F}_{comm}$, NoC with 2 intermediate switches (Fig. 3(a)) has higher FER of 6.5%, which increases to 9.5% and 11% as number of intermediate switches increases to 3 and 4 (Fig. 8). The comparative FER (Fig. 8) indicates that NoC-based decoder is more susceptible to SEUs during communication compared to AMBA-based decoder. We expect that the observation made about AMBA- and NoC-based decoders with respect to SEUs experienced to follow similar trend if the number of computation cores increase.

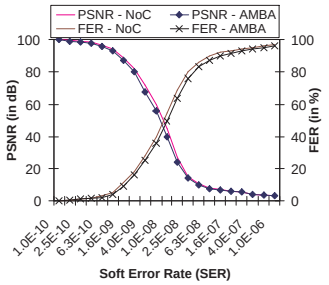

Fig. 7: PSNR and FER of AMBA (Fig. 2) and NoC (Fig. 3(a)) with varying SERs

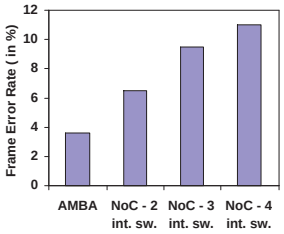

Fig. 8: Comparative FER of AMBA- (Fig. 2) and NoC-based decoder (with 2, 3 and 4 int. switches) at SER of 10^{-9}

IV. Conclusions

Using MPEG-2 video decoder traffic in simulated fault injection environment, we presented a comparative reliability analysis between shared-bus AMBA and NoC using four processing cores. We showed that for a given soft error rate, AMBA-based decoder experiences higher single-event upsets (SEUs) during computation due to higher execution time (Section III-A). On the other hand, NoC-based decoder experiences higher SEUs during inter-core communication (Section III-B). Considering the impact of total SEUs at application-level using peak signal-to-noise ratio (PSNR) as a reliability metric, we showed that NoC-based decoder gives higher PSNR than AMBA-based decoder. However, due to higher SEUs experienced by NoC-based decoder interconnects, it suffers from higher frame error ratio than AMBA-based decoder.

References

[1] H. Lee *et al*, "On-Chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-Point, Bus, and Network-on-Chip Approaches," *ACM TODAES*, vol. 12, no. 3, pp. 1–20, August 2007.

[2] A. Ejlali *et al*, "Joint Consideration of Fault-Tolerance, Energy-Efficiency and Performance in On-Chip Network," in *Proceedings of the DATE*, 2007, pp. 1647–1652.

[3] D. Park *et al*, "Exploring Fault-Tolerant Network-on-Chip Architectures," in *Proceedings of DSN*, June 2006, pp. 94–104.

[4] J. Kim *et al*, "Design and Analysis of an NoC Architecture from Performance, Reliability and Energy Perspective," in *Proceedings of the ACM ANCS*. NY, 2005, pp. 173–182.

[5] A. Dalirsani *et al*, "An Analytical Model for Reliability Evaluation of NoC Architectures," in *Proceedings of IOLTS*. USA, 2007, pp. 49–56.

[6] X. Li and D. Yeung, "Exploiting Application-Level Correctness for Low-Cost Fault Tolerance," *Journal of Instruction-level Parallelism*, vol. 10, pp. 1–28, 2008.

[7] D. Flynn, "AMBA: Enabling Reusable On-Chip Design," *IEEE Micro*, vol. 17, no. 4, pp. 20–27, Jul/Aug 1997.

[8] S. Kumar *et al*, "A Network on Chip Architecture and Design Methodology," in *Proceedings of Annual Symposium on VLSI*, 2002, p. 117.

[9] NIRGAM, www.nirgam.ecs.soton.ac.uk.

[10] R. A. Shafik *et al*, "SystemC-based Minimum Intrusive Fault Injection Technique with Improved Fault Representation," in *Proceedings of IOLTS*, July 2008, pp. 99–104.