2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)

Efficient Post-Silicon Validation of Network-on-Chip using Wireless Links

Sidhartha Sankar Rout[1], Kanad Basu[2], Sujay Deb[1]
[1]Indraprastha Institute of Information Technology Delhi, New Delhi, India
[2]New York University, Brooklyn, New York
[1]{sidharthas, sdeb}@iiitd.ac.in, [2]kb150@nyu.edu

***Abstract*— Modern complex interconnect systems are augmented with new features to serve the increasing number of on-chip processing elements (PE). To achieve the desired performance, power and reliability in the contemporary designs; Network-on-Chips (NoC) are reinforced with additional hardware and pipeline stages. Wireless hubs are supplemented on top of the baseline wired NoC for efficient intra-chip long distance communications. With the increasing complexity of the network, it is extremely difficult to ensure the functional correctness of the interconnect module at the pre-silicon verification stage. Hence, a robust post-silicon validation mechanism for NoCs has to be devised to guarantee the error-free functioning of the system. This paper exploits the capabilities of the wireless hubs present in wireless NoC (WNoC) to establish a novel post-silicon validation model for communication networks. The proposed method facilitates a better observability of the system in case of transient packet faults like misroute and packet-drop without any additional overhead in term of trace buffer size and trace bandwidth requirement. An overall 30% improvement in fault detection and path reconstruction is observed in comparison to the wired network using this wireless scheme. The wireless transceivers constructively use the existing network to transport the traces till the external debug analyzer, thus eliminating the need of additional trace bus while elevating the speed of trace communication.**

I. Introduction

In modern semiconductor designs, NoC is used as the medium for connecting multiple cores to achieve high performance and high throughput operations [1]. However, with increase in number of processing elements, source to destination hop count increases for long distance communication. This results in elevated power and performance overhead of the multi core system based on wired NoC. In this regard, WNoC has emerged as a potential and reliable solution [2], [3], where on-chip wireless transceivers are augmented on top of the baseline wired NoC. This enhances the performance of the network by drastically reducing the hop count during long range data transmission.

As the computation is moving towards exascale era, more number of PEs are interacting with each other at high speed on a System-on-Chip (SoC). This necessitates an efficient interconnection infrastructure, that can serve all the communication requests reliably on time. Therefore, the complexity of interconnect system has increased multi-fold with several level of parallelism and additional features. The state-of-the-art NoC routers have become extremely complex. Evaluating and maintaining the functional correctness of these modules is a real challenge. The high level of design complexity of NoC leads to the situation where a large number of functional bugs escape through the pre-silicon verification stage till the actual product on silicon. Even though the processing cores function correctly, bugs in network can very well introduce permanent faults like *Deadlocks*, *Livelocks*, and transient faults like *Dropped Data Fault (DDF)*, *Direction Fault (DF)*, *Multiple Copies in Space Fault (MCSF)*, *Multiple Copies in Time Fault (MCTF)* etc. [4]. This necessitates a strong post-silicon validation (alternatively called as post-silicon debug) framework for interconnect system to ensure minimal or no functional network fault on actual products.

A post-silicon validation method helps detecting the bugs by increasing the observability and controllability of the internal nodes of a system. Various researchers have developed such NoC monitoring platforms by introducing probes in the design of multiprocessor system on chip (MPSoC) [5], [6], [7]. A trace based post-silicon validation records the system internal states during its normal execution. Trace allows the capture of errant corner cases that functional verification phase could not cover. Determining the level of trace has the biggest impact on the cost of implementing the on-chip debug system [8]. Several works have been done on trace signal selection and trace data compression to reduce the level of trace [9], [10], [11]. An on-chip debug platform for NoC is developed in [12], which detects the type of network fault at the local IP level. This method utilizes the existing cache memories as trace buffer to store the snapshots. But the on-chip debugger increases the complexity of this design, and also this method is not suitable for transient fault detection which demands very frequent trace capture.

Our work proposes an efficient trace based post-silicon validation scheme for NoC with augmented wireless links. The framework improves the observability of network and at the same time utilizes the wireless capability to enhance the debug process. It can be seen that the average life of packet in flight reduces notably in a WNoC in comparison to wired NoC. This creates an opportunity to capture more traces without increasing the trace buffer size which would enable the debug system to detect the network transient faults more efficiently. Existing on-chip wireless transceivers are used to accelerate the transfer of trace data. Traditional low bandwidth interface used to transfer the traces till the external debug analyzer is replaced by wireless interfaces (WIs). This significantly increases the speed of trace data transfer. The major contributions of the paper are summarized as follows:

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00082

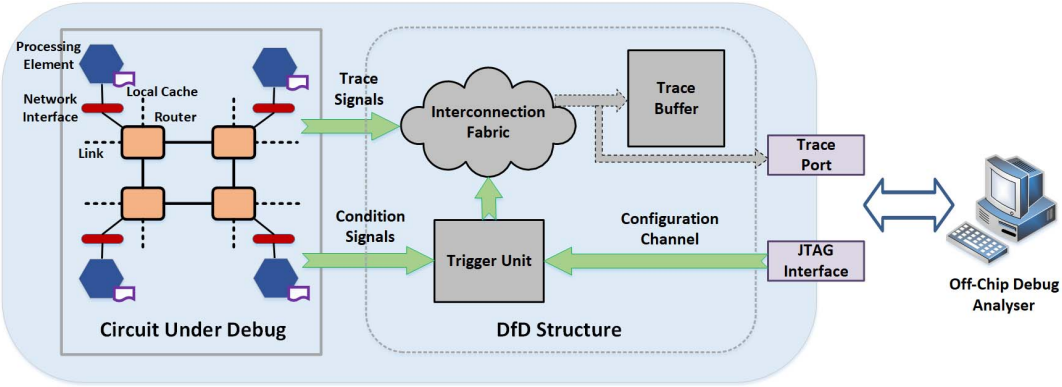


Fig. 1. **Trace Based NoC Debug Platform.** NoC is taken as the *Circuit Under Debug*. A *Design for Debug* structure has been embedded and an *Off-Chip Debug Analyzer* is instrumented for the post-silicon validation of the NoC.

1) Proposed a suitable post-silicon validation scheme for wireless NoC.
2) Observability of the network is assessed for several fault models.
3) Analyzed several wireless capabilities of the network to enhance the validation process.

II. Post-Silicon Validation Framework for NoC

This paper adopts a trace-based post-silicon debug for NoC. The process can broadly be divided into 3 parts as (a) Packet trace collection, (b) System trace data transfer, and (c) Post-silicon analysis. A trace based debug platform, inherited from [8], [10], [13] has been redrawn as a debug framework for NoC in Figure 1. As can be seen, a *Design for Debug* (DfD) structure is introduced to the *Circuit Under Debug* (CUD). The DfD unit can collect the transaction level packet information from the NoC, that is the CUD in this case. These packet information can be considered as trace signals which give a globally consistent view of the communication among multiple cores on the system.

During the debug phase, traces can be collected concurrently at all the routing nodes in every cycle, which will give the complete transaction level details of each packet traversing through the network. But this generates huge amount of trace data which is difficult to store, transfer and analyze. This will increase DfD overhead in terms of *trace buffer size* and *trace bandwidth requirement*. *Trigger unit*, shown in Figure 1 can be used to decide the *snapshot interval* between two consecutive snapshots, that is the frequency of packet tracing. The *JTAG Interface* shown in the same figure can be used as test access port (TAP) during the debug phase to configure the *trigger unit* to control the *snapshot interval*.

The collected packet traces are required to be transfered till trace buffer or trace port via the interconnection fabric shown in Figure 1. Dedicated debug buses are used for this purpose [8], which add to the routing overhead of the DfD. A NoC based system can facilitate the trace data to be transferred over the same network as is used by the normal payload, and thereby can avoid the additional hardware and routing cost caused by the debug bus [5], [7]. The on-chip WIs in case of WNoC can be used to transfer the packet traces more effectively while leveraging the burden on the wired network. This can considerably improve the network performance as is discussed in Section IV.

The real time trace data can be sent to the *Off-Chip Debug Analyzer* for bug detection and analysis through the trace port. Alternatively, On-chip *Trace Buffer* can be used to store the packet traces which can exempt the requirement of trace port and thereby can save pin area. The saved trace data can be read over the *JTAG Interface*. In this paper, we adopt the embedded trace buffer method for post-silicon debug of NoC. The size of *Trace Buffer* is usually small, normally in the range of 2 to 8 KB [14]. Therefore, the intelligent use of *Trace Buffer* is highly desired.

The Off-Chip Debug Analyzer processes the packet traces to identify the type of functional fault and reconstructs the path used by the packets to traverse through the network. The trace collection, transfer and analysis phases are iterated multiple times till all the functional bugs are identified.

III. WI-Based NoC Post-Silicon Validation

This paper establishes an efficient post-silicon validation framework for NoC by adding wireless links to the platform discussed in the previous section. This framework differs from the one depicted in Figure 1 on the basis of three modifications as shown in Figure 2.

1) Reuse of local cache memory of each core as distributed *Trace Buffer*.
2) Reuse of NoC Infrastructure (CUD in this case) for packet trace transfer.
3) Use of Wireless links for off-loading the packet traces till the *External Debug Analyzer*.

A. Packet Trace Collection

Trace Buffer, an on-chip memory space meant for storing the trace data is used only during the post silicon validation phase and has no use thereafter. Hence, the proposed method utilizes the local L2 cache of each core as distributed trace buffer like [12], rather than having a dedicated one. This saves on-chip area and routing overhead. A specified portion

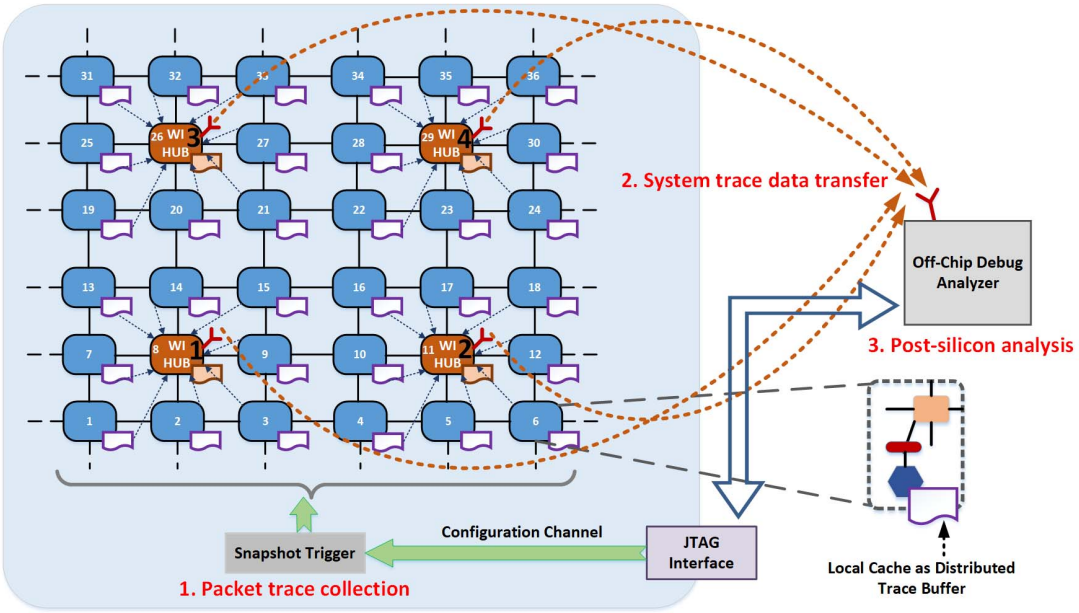


Fig. 2. **NoC Debug Platform using Wireless Links.** Framework is using local cache as distributed trace buffer. WI HUB represents the network node with Wireless Interface. Wireless communication is used for packet trace transfer.

of the local caches are temporarily used to store packet traces during debug phase and are released for normal operations after the post-silicon validation.

During trace collection phase, each router takes the snapshot of its own state in a periodic manner and store the traces in the reserved space of attached cache. The interval between snapshots can be controlled by the *Snapshot Trigger* unit depending on the type of suspected fault and the type of network node. For transient faults the interval is kept low. The average life span of a transient fault is pretty low and can be detected if at-least a single snapshot is available between the occurrence of the fault and before it dies. Whereas for permanent faults, the snapshot interval can be set with a high value. A permanent fault can be observed on the network all the time after its occurrence and can be detected even if with very large snapshot interval. The number of nodes with wireless interface (shown as *WI HUB* in Figure 2) in the network is less and the total wireless utilization percentage is small with respect to the whole communication. This facilitates the *Trigger* unit to fix the snapshot interval of one cycle for *WI HUBs* present in the network.

Router state represents the packet information of all the packets present in the router at an instant. Packet information can be extracted as trace data from the header flit of the packet. Figure 3 shows the format of packet and header flit transmitted over WNoC. In WNoC each WI is assigned with a unique *WI Address*. The packet header holds the *WI Address* of both source and destination WI along with the actual *Source* and *Destination node Addresses* of the respective packet. *Packet Number* is another information carried by the header flit which is explicitly added to be used during the *Analysis* phase. An incremental packet number is assigned to each packet generated at individual core level. This can be implemented using a sequential counter in every core. The *Packet Number* along with the *Source* and *Destination node Address* combined can act as a unique *Packet ID* for every packet traverses through the network.

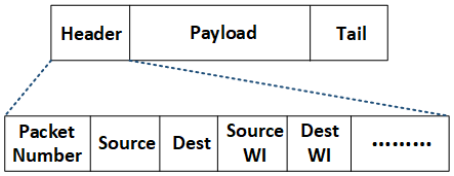


Fig. 3. **Format of packet and header flit content transmitted over WNoC.** Unused space in header flit can be used to hold *Packet Number*, *Source WI* and *Dest WI*.

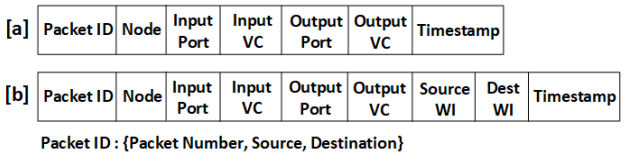
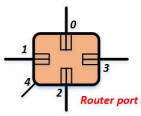


Fig. 4. **Packet trace pattern.** [a] trace pattern for wired node, [b] trace pattern for WI HUB

Whenever a snapshot is taken, header flits of all the packets present in a router are detected. Packet information from the header flit along with the packet status inside the router are extracted to build the packet trace. Pattern of packet trace at both wired node and *WI HUB* is shown in Figure 4. For a wired router node, *Packet ID* is collected from the header flit whereas *Node*, *Port* and *VC* information are collected from the packet status in the router. Physical



Router port numbering and Virtual Channels

A packet generated from node '1' is intended to be consumed at node '22'.

The expected correct path: **1 → 2 → 8 ⇢ 29 → 28 → 22**

Wireless Communication

Table – 1 : Complete packet traces captured with 1 cycle snapshot interval (Ideal Case) – No functional fault

Packet ID	Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	-	-	-	-	1002
{2,1,22}	1	4	2	3	2	-	-	1004
{2,1,22}	2	1	2	0	-	-	-	1007
{2,1,22}	2	1	2	0	1	-	-	1009
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	-	-	1	4	1014
{2,1,22}	29	-	-	-	-	1	4	1017
{2,1,22}	29	-	-	1	2	1	4	1020
{2,1,22}	28	3	1	2	-	-	-	1022
{2,1,22}	28	3	1	2	2	-	-	1023
{2,1,22}	28	3	1	2	2	-	-	1025
{2,1,22}	22	0	2	4	2	-	-	1029

Note: A filtered packet trace pattern is shown for demonstration purpose

Table – 2 : Complete packet traces captured with 1 cycle snapshot interval - Direction fault at node 29

Packet ID	Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	-	-	-	-	1002
{2,1,22}	1	4	2	3	2	-	-	1004
{2,1,22}	2	1	2	0	-	-	-	1007
{2,1,22}	2	1	2	0	1	-	-	1009
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	-	-	1	4	1014
{2,1,22}	29	-	-	-	-	1	4	1017
{2,1,22}	29	-	-	0	1	1	4	1020
{2,1,22}	35	2	1	1	1	-	-	1023
{2,1,22}	34	3	1	-	-	-	-	1025
{2,1,22}	34	3	1	2	1	-	-	1027
{2,1,22}	28	0	2	2	-	-	-	1030
{2,1,22}	28	0	2	2	2	-	-	1032
{2,1,22}	22	0	2	4	2	-	-	1035

Table – 3 : Actual packet traces captured with 3 cycle snapshot interval (Test Case) – No functional fault

Packet ID	Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	-	-	-	-	1002
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	-	-	1	4	1014
{2,1,22}	29	-	-	-	-	1	4	1017
{2,1,22}	29	-	-	1	2	1	4	1020
{2,1,22}	28	3	1	2	2	-	-	1023
{2,1,22}	22	0	2	4	2	-	-	1029

Table – 4 : Actual packet traces captured with 3 cycle snapshot interval - Direction fault at node 29

Packet ID	Node	Input Port	Input VC	Output Port	Output VC	Source WI	Dest WI	Timestamp
{2,1,22}	1	4	2	-	-	-	-	1002
{2,1,22}	8	2	1	-	-	1	4	1012
{2,1,22}	8	2	1	-	-	1	4	1014
{2,1,22}	29	-	-	-	-	1	4	1017
{2,1,22}	29	-	-	0	1	1	4	1020
{2,1,22}	35	2	1	1	1	-	-	1023
{2,1,22}	28	0	2	2	2	-	-	1032
{2,1,22}	22	0	2	4	2	-	-	1035

Fig. 5. **Walk-through example.** Direction fault detection and packet path reconstruction from the available packet traces. Router port numbering and Virtual Channels are shown for reference.

timestamp at which the snapshot is taken is added to form a complete packet trace. For a packet present in *WI HUB* at the time of snapshot capture, the packet trace contains WI source and destination addresses fetched from header flit in addition to all other entries present in case of wired node.

Traces are captured and stored in the local buffer of each router with a frequency of pre-determined snapshot interval. Once the trace buffer space is filled with traces, the network operation is halted and *Trace Data Transfer* phase is started.

B. Trace Data Transfer

Trace data need to be communicated to the *Off-chip Debug Analyzer* for fault identification and path reconstruction in case of any fault detection. Reconstructed path helps in debugging the location of fault and gives inputs for corrective action. In traditional method, a dedicated trace bus is used for trace communication. In the proposed method, we are reusing the existing network bandwidth for the same purpose.

During the *Trace Data Transfer* phase, the normal execution of the network is stopped. Trace data stored in local cache of each router gets transferred to the nearest *WI HUB* using the wired path. Then, the accumulated traces at each *WI HUB* are sent to the *Off-chip Debug Analyzer* for post processing. This communication is traditionally accomplished using a low bandwidth JTAG interface [15]. In the proposed approach, we have used inter-chip wireless communication which uses high bandwidth wireless links to transport the trace data till the *Analyzer* as shown in Figure 2.

C. Post-Silicon Analysis

Upon the collection of packet traces, the *Debug Analyzer* processes the data and detects the packet level functional faults, if any. All the traces related to a particular packet can be extracted using the unique *Packet ID* assigned to it and stitched together. Now, the timestamps in the traces corresponds to a packet help reconstructing the path taken by the packet. The detection algorithm in the *Analyzer* can take this path as input to make a decision on the occurrence of any functional fault to the respective packet. The constructed path also indicates the location of fault. Figure 5 demonstrates a *Direction Fault* case and its analysis for a particular packet by a walk-through example.

The second packet generated from node 1 is intended to be consumed at node 22 as shown in Figure 5. The correct path of the packet is shown for XY routing. The node and the *WI HUB* numbers can be referred from Figure 2 for this example, whereas the router port number can be viewed at the top left corner of Figure 5. This example considers the router with two VCs at each port. *Table 1* and *Table 3* in the figure show the fault-free packet traces collected with snapshot interval of 1 cycle and 3 cycles respectively. We can observe that a snapshot interval of 3 reduced the amount of traces to almost 50%. The individual trace shows the exact status of the packet at the time of snapshot. *Source WI* and *Dest WI* for the packet communication are captured only at the *WI HUB* to save some buffer space. We can notice that Table 3 holds all the traces related to wireless communication as Table 1. This is because *WI HUBs* are triggered to take the snapshot at every cycle to ensure the detection of wireless path in a packet communication if it is taken. This is still affordable in terms of trace buffer size limitation as wireless is only used during long range packet communication.

Table 2 and *Table 4* exhibits the traces for the same above packet with a *Direction Fault* at node 29 for snapshot period of 1 cycle and 3 cycles respectively. Rather than moving to node 28 from 29, the packet moves to node 35 due to the fault. The fault detection algorithm in the *Debug Analyzer* can detect this anomaly from the fifth trace in Table 4 indicated by a different *Output Port* than the intended one.

A rough estimation of the percentage of path reconstruction for the above example can be as follows. If we observe both *Table 2* and *Table 4* carefully, we can find that trace data in *Table 4* do not have any information regarding the intermediate nodes 2 and 34 actually covered by the packet. In its complete traversal, the packet covered 8 nodes whereas

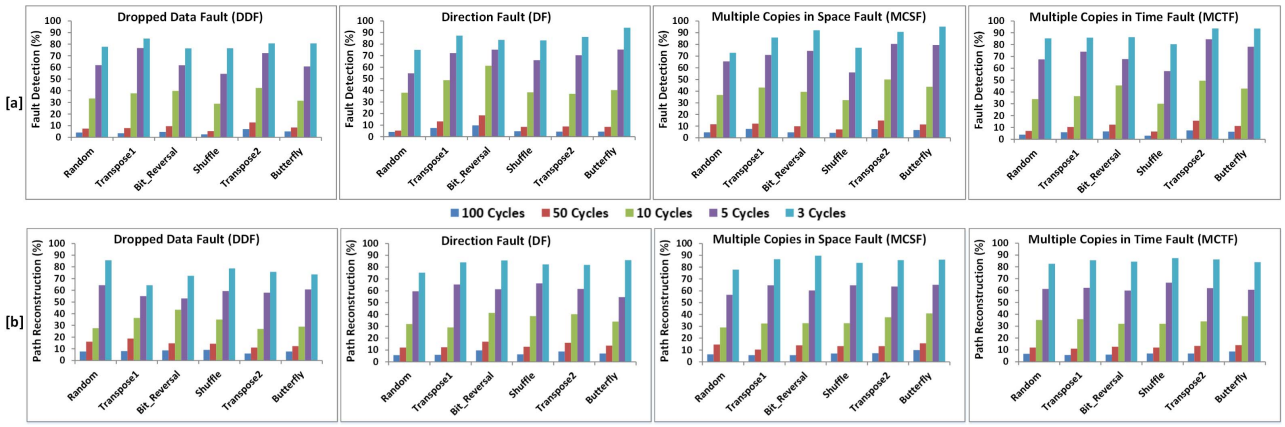


Fig. 6. **[a] Percentage fault detection, [b] Percentage path reconstruction.** Percentage fault detection and percentage path reconstruction for different network transient faults are shown with varying snapshot interval. Six different *Benchmarks* are considered as shown in the x-axis of each functional fault.

the collected trace gives the information about 6 nodes. This returns a rough estimation of 75% path reconstruction for the corresponding packet. In real scenario, this percentage can be increased as the input port '0' for node 28 indicates the packet must have traversed through 34.

D. Wireless Interface

In the proposed validation framework, WIs are used for trace communication. This WNoC topology is comprised of Base Routers (BRs) and few Hybrid Routers (HRs) shown as *WI HUBs* in Figure 2. HR integrates WI with BR components and can convert digital packet data to RF domain and vice versa. This is composed of serializer/deserializer, modulator/demodulator, power amplifier (PA) and LNA components. An antenna is used to both transmit and receive data whereas the wireless channel is shared between all the WIs [16]. A variable gain PA [17] is used which can provide different amplification level depending on whether the WI is used for normal on-chip packet communication or is used for off-chip trace communication. The amplifier pumps more power to transfer traces till the *Off-chip Debug Analyzer*. Power gated WIs are used to reduce the energy consumption [18].

IV. Results and Analysis

This section discusses the experimental setup for WI based debug platform. The multi-core interconnect fabric is modeled on cycle accurate Noxim simulator [19]. The details of the WNoC and the simulation setup are shown in Table I. All cores and wired network modules are operated at a clock frequency of 1 GHz. Fault detection and path reconstruction capability of the proposed framework is evaluated by injecting several faults to the network. Performance of the wireless platform is compared with the wired platform.

Figure 6 shows the fault detection and path reconstruction ability of proposed platform respectively with different snapshot intervals and for several synthetic traffic patterns. The results are shown for transient network faults. Faults like deadlock and livelock can be 100% detected even if the snapshot interval is very high since these faults are

TABLE I
Network Topology and Simulation Setup

Component	Configuration
Topology	8x8 WNoC (Mesh augmented with wireless links), 4 wireless hubs
Router	5 I/O wired ports, 2 virtual channels, 8 flit buffers, 8 flit packets, 32 bit flits
Wireless Link	60 GHz carrier, 16 Gbps bandwidth, single cycle latency
Workload	Synthetic - *bitrevarsal*, *butterfly*, *random*, *shuffle*, *transpose1*, and *transpose2*

permanent in nature, as already shown in [12]. Hence, we do not consider those fault models in our discussion. From Figure 6, it can be seen with increasing snapshot interval, the percentage of transient fault detection degrades significantly. This is because these faults are short lived on the network and snapshots after large intervals are unable to capture most of the traversal information of faulty packets. Hence, the fault detection as well as path reconstruction for transient faults demand very frequent trace collection. However, collecting more traces incur area overhead due to large trace buffer size, as well as high trace bandwidth requirements.

A. Trace Buffer Size

Smaller trace buffer size is always desired as it saves silicon area as well as cost. Though in this work we have reused a small portion of each cache memory as trace buffer, it is required to assign a large portion of the cache for normal operation even during the debug phase. In our experiment a single snapshot of a wired only network is of 52 bits. This includes 22 bits of *Packet ID*, 14 bits of *Packet Status* information inside a router and 16 bits of *timestamp* value. Snapshot of a wireless communication adds 4 more bits of WI address values on top of the wired snapshot and consumes a total of 56 bits.

We have considered a 8x8 mesh network with 4 wireless hubs. For a random traffic pattern, the wireless network results average hop count of 4 in comparison to 6 per packet

communication with a wired only network. If packet snapshot is taken once at each hop during a packet transmission, in case of a wired network, total of 3.81 KB of trace buffer size is needed to accommodate traces for 100 packets. In case of a wireless network, two hops involved in wireless transmission would require 56 bits each to store the snapshot while the remaining hops require 52 bits only. So, with around 10% of wireless communication, the network requires a 2.54 KB of trace buffer which is around 67% of wired case.

Traces collected in every 5 cycles requires 60% of buffer space in comparison to the trace collection in every 3 cycles. This requirement is further increased as snapshots are taken in every clock cycle at the wireless hubs as explained in Section III. Therefore, We have fairly assumed that the trace buffer requirement with 3 cycles snapshot in case of a wireless network is almost equivalent to the 5 cycles snapshots in case of a wired network. Hence, the proposed wireless post-silicon validation platform can show a considerable amount of improvement both in case of transient fault detection and path reconstruction in comparison to a wired post-silicon validation platform. This can be observed in Table II for a random traffic pattern for all the different transient faults.

TABLE II
WIRELESS AND WIRED POST-SILICON VALIDATION COMPARISON

% Improvement	***DDF***	***DF***	***MCSF***	***MCTF***
Fault Detection	25	37	11	26
Path Reconstruction	33	26	38	35

B. Efficient Trace Data Transfer

In the proposed post-silicon validation framework, trace data is transferred using the wireless links from the trace buffer till the external *Analyzer*, replacing the on-chip as well as off-chip wired medium used traditionally. It has been observed that for a 20 *mm* distance, propagation delay of on-chip WIs is 0.15 *nsec* whereas delay with unbuffered and buffered wire are 3.86 *nsec* and 1.16 *nsec* respectively [20]. The off-chip wireless communication gives even better result than off-chip wired interconnect in terms of propagation delay. Experiments in [20] also shows WIs consume 9.7% and 43.6% of total energy dissipated by unbuffered wires and buffered wires respectively to transmit one bit of data over 20 *mm* distance. As the distance increases, WIs provide improved energy saving compared to wired links.

V. CONCLUSION

In this paper, we have proposed a post-silicon validation framework for NoC using wireless links. The method reuses the local cache as distributed trace buffer and utilizes the existing wireless transceivers for efficient transfer of trace data. A variable gain power amplifier provides the required amplification gain for off-chip communication. The scheme is proved to be more useful for detecting network transient faults by providing the flexibility of frequent trace collection without adding any extra overhead. This also improves the path reconstruction for packets during Analysis phase. Experiments exhibit that the wireless post-silicon validation framework shows around 30% of improvement in fault detection and path reconstruction in comparison to traditional wired-based method.

REFERENCES

[1] L. Benini and G. De Micheli, "Networks on chips: A new soc paradigm," *computer*, vol. 35, no. 1, pp. 70–78, 2002.

[2] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless noc as interconnection backbone for multicore chips: Promises and challenges," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, pp. 228–239, 2012.

[3] C. Wang, W.-H. Hu, and N. Bagherzadeh, "A wireless network-on-chip design for multicore platforms," in *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*. IEEE, 2011, pp. 409–416.

[4] A. Alaghi, N. Karimi, M. Sedghi, and Z. Navabi, "Online noc switch fault detection and diagnosis using a high level fault model," in *Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on*. IEEE, 2007, pp. 21–29.

[5] S. Tang and Q. Xu, "A multi-core debug platform for noc-based systems," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*. IEEE, 2007, pp. 1–6.

[6] B. Vermeulen and K. Goossens, "A network-on-chip monitoring infrastructure for communication-centric debug of embedded multiprocessor socs," in *VLSI Design, Automation and Test, 2009. VLSI-DAT'09. International Symposium on*. IEEE, 2009, pp. 183–186.

[7] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, "Transaction monitoring in networks on chip: The on-chip run-time perspective," in *Industrial Embedded Systems, 2006. IES'06. International Symposium on*. IEEE, 2006, pp. 1–10.

[8] W. Orme, "Debug and trace for multicore socs," *ARM White Paper*, 2008.

[9] K. Basu and P. Mishra, "Rats: Restoration-aware trace signal selection for post-silicon validation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 605–613, 2013.

[10] H. F. Ko and N. Nicolici, "Algorithms for state restoration and trace-signal selection for data acquisition in silicon debug," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 2, pp. 285–297, 2009.

[11] K. Basu and P. Mishra, "Test data compression using efficient bitmask and dictionary selection methods," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 9, pp. 1277–1286, 2010.

[12] R. Abdel-Khalek and V. Bertacco, "Functional post-silicon diagnosis and debug for networks-on-chip," in *Proceedings of the International Conference on Computer-Aided Design*. ACM, 2012, pp. 557–563.

[13] Q. Xu and X. Liu, "On signal tracing in post-silicon validation," in *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*. IEEE Press, 2010, pp. 262–267.

[14] A. Martin, "Debugging with arm coresight," *LAUTERBACH NEWS*, 2009.

[15] K. Morris, "On-chip debugging - built-in logic analyzers on your fpga," *J. FPGA Structured ASIC*, vol. 2, no. 3, 2004.

[16] S. H. Gade, S. S. Rout, M. Sinha, H. K. Mondal, W. Singh, and S. Deb, "A utilization aware robust channel access mechanism for wireless nocs," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018.

[17] S. K. Kumar, M. Agrawal, H. K. Mondal, S. H. Gade, and S. Deb, "Path loss-aware adaptive transmission power control scheme for energy-efficient wireless noc," in *Circuits and Systems (MWSCAS), 2017 IEEE 60th International Midwest Symposium on*, pp. 132–135.

[18] H. K. Mondal, S. Kaushik, S. H. Gade, and S. Deb, "Energy-efficient transceiver for wireless noc," in *VLSI Design and 2017 16th International Conference on Embedded Systems (VLSID), 2017 30th International Conference on*. IEEE, 2017, pp. 87–92.

[19] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, "Cycle-accurate network on chip simulation with noxim," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 27, no. 1, p. 4, 2016.

[20] S. H. Gade and S. Deb, "Achievable performance enhancements with mm-wave wireless interconnects in noc," in *Proceedings of the 9th International Symposium on Networks-on-Chip*. ACM, 2015, p. 29.