A run-time reconfigurable NoC Monitoring System for performance analysis and debugging support

Erol Koser* and Benno Stabernack**

* Insitute for Integrated Systems, Technische Universität München
** Embedded Systems Group, Fraunhofer Heinrich-Hertz-Institute Berlin

Abstract:

Recently Network-on-Chip based architectures become more and more important due to their advantages in respect to design flexibility and systems bandwidth scalability since nowadays systems consists typically of a huge number of processing elements (e.g. heterogeneous multi processor systems). In contrast to typical shared memory based systems, predicting and monitoring the runtime behaviour of the system e.g. data throughput, link utilization and contention becomes more complex and requires special architectural features. Besides the traditional approach of using simulation based approaches at design time, runtime usable features promise to have a number of advantages. In this paper we present a flexible, reusable and run-time reconfigurable NoC monitoring system for performance analysis and debugging purposes. The evaluation of the monitoring data enables the system designer to achieve better resource utilization by adjusting the system architecture and the programming model.

1 Introduction

Since traditional bus systems are the critical bottleneck if more than a few master modules are attached, Networks-on-Chip (NoC) are a promising approach to solve this issue [BDM02]. Its advantages are high scalability and massive parallel communication capabilities. Even though NoCs as a communication infrastructure are well established their usage implies a number of new problems which need to be addressed to gain their full advantage. Simulations, especially HW/SW-Co-simulations, are very time consuming. A more time efficient option is to run the complete system on a prototype and trace its activity. Since NoCs do not have a central point of communication, monitoring mechanisms have to be deployed across the system. The mechanisms can be utilized at design-time and at run-time. During design-time they can be used for rapid prototyping. In case of malicious system behaviour the monitoring data can be utilized for communication debugging. Performance analyses can be executed to adjust the programming model and the system architecture. Monitoring systems are also used at run-time to support resource management functionalities of the operating system. The main components of the proposed monitoring system are probes, that are attached to communication links, and a Central Monitoring Unit (CMU). Different modes of abstraction are offered in order to collect as much data as possible. Additionally a data compression scheme is introduced, that decreases the overall amount of monitoring data without information loss.

2 Related Work

NoC monitoring is recently getting more and more attention. The majority of the systems deal with system reliability as in [MVBT09], [PVS+10] and [ZMV+11]. Sensors are placed across the chip and measure different properties like power dissipation and soft errors. Other systems focus on the communication. In [CBR+04], [CGB+06] and [VG09] the monitoring system is used to provide transaction data for debugging purposes. In [DF14] a tree-based debugging infrastructure is introduced that debugs end-to-end-transactions and recovers the system online in case of malicious behaviour.
Little effort has been done so far on run-time management and performance analysis. In both cases similar information is required and the goal of both areas is to achieve a better system performance and resource utilization. The main difference is that for run-time management the evaluation of monitoring data is performed on chip and for performance analysis on an external host. In [FPS09] and [FPS10] the monitoring data is evaluated to adjust adaptable and reconfigurable systems. External performance analysis is executed in [ASNH10] and [HAS+08] to provide the system designer with appropriate run-time data.
The ESG (Embedded Systems Group) Monitoring System is primarily designed to deliver information for performance analyses. In opposite to [CBR+04] and [CGB+06] the ESG Monitoring System is non-intrusive and offers higher reusability. It is not utilizing the monitored NoC itself for its internal communication and therefore just the front-end of the probes are NoC specific. Furthermore it is run-time reconfigurable, which enables the system designer to change the data granularity without resynthesizing the system. The probes are monitoring one link exclusively. This property allows to monitor parallel communication on all links at the same time.
The probes in [ASNH10], [FPS09], [FPS10] and [HAS+08] mainly consist of counters, which measure different events and qualities. Even this data might be sufficient for a lot of analyses the ESG Monitoring System provides more information, which enables more precise analyses like reconstructing the paths of packets and examining delays within the system. Additionally it is the only approach that includes a data compression scheme to decrease the overall amount of monitoring data. Simulation-based approaches like [RCM14] are not considered in this section since the focus lies on monitoring real-time behaviour on-chip.

3 ESG Monitoring System

NoC traffic analyses require concurrent transaction tracing. In order to achieve concurrency, probes are attached to communication links. The **probe placement** is an important design time choice. It determines both, the obtainable information and its granularity. The scheme offers high flexibility and maximum design freedom. One option is to place the probes at the links between the network interfaces (NI) and the routers. This option is only capable of end-to-end analyses (total delay, communication between entities, etc.) but also results in the lowest complexity. The internal run-time behaviour of the NoC (contention, traversed paths, etc.) can be monitored, if the probes are placed at links between routers.

Using exclusive probes for individual links enables concurrent monitoring. The application of the ESG Monitoring System requires (1) packet based transactions, (2) flit based packet composition and (3) wormhole switching.

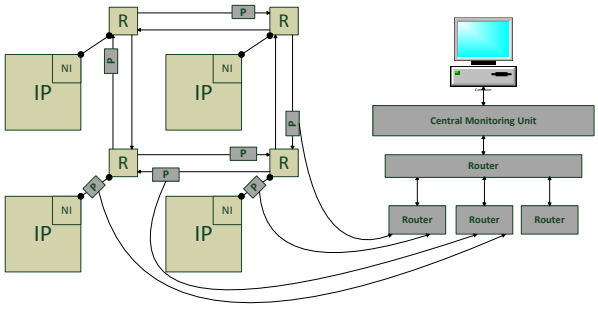

Figure 1: In the ESG Monitoring System the probes are placed directly at the NoC links. The communication infrastructure is organised in a tree topology where routers forward the monitoring packets from the probes (P) to the CMU. The IP cores (IP), NoC routers (R) and network interfaces (NI) are part of the monitored NoC and do not belong to the monitoring system.

Communication Infrastructure. A common approach is to use NoCs as the internal communication medium of monitoring systems. The first option is to use the monitored NoC itself. Since extra load would distort performance analyses it is not suitable for our purposes. The second option is to use a dedicated NoC. Even though it is non-intrusive it is questionable if NoCs are the best choice. The reason is that in monitoring systems all communication is concentrated on a central module. Therefore the parallel communication abilities of NoCs are not required. If the mostly unidirectional communication behaviour of monitoring systems is taken into account, a tree based topology (see Fig. 1) appears to be a good option [PVS+10]. Packets are routed from the probes (leaves) to the Central Monitoring Unit (the root) by routers. Functions like address translation, complex routing algorithms or ID management are not required which means that low cost routers are employed. The main advantage is high scalability. If the probe number increases, new routers and levels in the tree architecture can be easily introduced. Another advantage is that the buffers in the routers are logically shared between the probes.

Probe Architecture. The probes monitor the transactions on the NoC links, process the information and send it to the Central Monitoring Unit (CMU). Additional statistics are taken over a specified time window similar to [FPS09]. The monitored NoC (ESG NoC) has a larger bus width (144 bit) than the ESG Monitoring System (32 bit). Additionally extra information (header, timestamp) has to be attached to the raw data. This properties make it impossible to process a link utilization of 100 %. Usually not all information of a transaction are relevant for an analysis. Depending on the required information and to provide as much data as possible different modes are available. The modes mainly differ in their abstraction level similar to the modes presented in [CGB+06]. The probe architecture is illustrated in Fig. 2. It consists of 4 main parts. The timer is used to generate timestamps and to trigger time-outs whereas the configuration register stores configuration bits that can be adapted during run-time. The configuration options are: (1) switching the

probe on or off and (2) programming its mode. The FIFO is used to buffer the monitoring packets until they are read by a router.

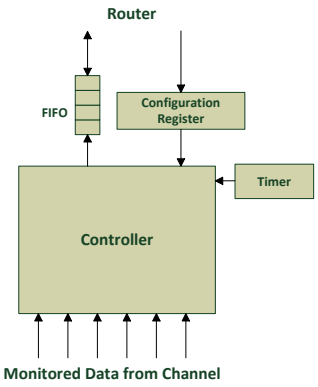

Figure 2: Probe Architecture

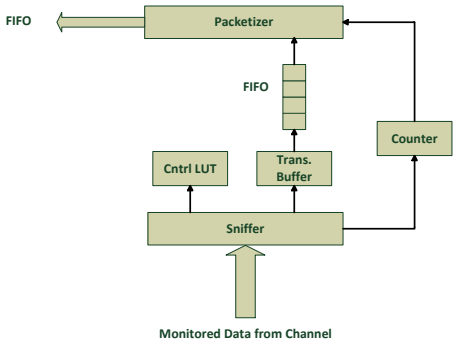

Figure 3: Probe Controller

The controller consists of two modules (see Fig. 3). The **Sniffer** analyses the data streams on the link utilizing a look-up-table (LUT). The **Control-LUT** is used to check if a packet completely passed the link. If a header flit is monitored, the packet-ID and the packet length are stored in the Control-LUT. For each subsequent payload flit the length will be decreased by one. Once the packet length reached zero the packet fully passed the link. The **Transaction Buffer** stores the characteristical parameters of the packet. If a packet passed the link, the parameters are taken from it and saved together with the timestamps and delays in the **Packetizer-FIFO**. The **Packetizer** takes the transaction information from the Packetizer-FIFO and stores it together with some static information (probe number, transaction type, etc.) in the required packet format in the FIFO on Fig. 2.

In **raw mode** the monitored raw data will be forwarded to a host. No abstraction or analysis is performed. The monitoring packets consist of 6 flits (32 bit each). The first flit is the header, which contains general and routing information. Since one NoC flit contains 4 data words (32 bits), 4 payload flits are added. The packet is completed by a timestamp. For each **NoC flit** a subsequent **monitoring packet** is sent. The header and the timestamp cause additional 33% overhead to the monitored payload. Since 6 clock cycles are required to send a monitoring packet it is not able to process a higher link utilization than 16.6 %.

The **packet mode** introduces the first abstraction level. Transactions are analysed from the packet perspective. The monitoring packet in this mode consists of 4 to 5 flits (see Fig. 5). The first flit is the header similar to the raw mode. The three subsequent flits contain the source address of the monitored NoC packet, the destination address and a timestamp. Depending on the burst length of the NoC packet, a fifth flit (the delay flit) will be attached. The delay flit contains the delays of all NoC flits to their predecessor[1]. For every **NoC packet** (not flit) a subsequent **monitoring packet** is sent. Compared to a NoC

[1]The ESG NoC operates with a guaranteed throughput and the maximal packet length is 8 flits. Therefore one monitoring flit is able to carry all delays, which might not be sufficient for other NoCs or other routing algorithms.

packet and depending on its burst length the data is comprised by 11 % to 86 % since the payload is removed. In this mode the probes are able to process a link utilization between 25 % an 100 %. 25 % in case the monitored NoC operates with single flit packets only, which can be theoretically monitored at each clock cycle. The subsequent monitoring packet would would consist of four flits and require four clock cycles to be transferred. Once the NoC packet consists at least of five flits the subsequent monitoring packet would have the same or a lower number of flits and the processable link utilization would increase to 100%.

The **transaction mode** provides the highest abstraction level. Instead of flit delays the whole transaction is analysed from the channel view. The header flit has an additional field. The field contains the delay between the header flit and the last payload flit (packet delay) in clock cycles. The monitoring packet length is four flits. The subsequent three flits (source address, destination address, timestamp) have the same function as in the packet mode. Just as in the packet mode for each NoC packet a subsequent monitoring packet is generated and the probe is able to process a link utilization of 25 % up to 100 % without information loss. The difference is that the probe is already able to process a link utilization of 100 % when the NoC packets consist at least of four flits. Table 1 presents an overview of the characteristical values of the modes.

Table 1: Comparison of the available Modes

	Raw Mode	Packet Mode	Transaction Mode
Abstraction Level	raw data	flit delays	packet delay
Monitored entities	flits	packets	packets
Packet Length	6 flits	4-5 flits	4 flits
Data reduction	+33.3 %	-11 % to -86 %	-11 % to -89 %
Max. link utilization	16.6 %	25 % to 100 %	25 % to 100 %

Central Monitoring Unit. The CMU collects the monitoring packets and manages the communication with the host. Since the CMU needs to forward concurrently collected data (monitoring packets) in a sequential manner it presents the critical bottleneck in the ESG Monitoring System. In addition the on-chip data rate usually exceeds the external data rate which worsens the problem. To reduce the overall amount of monitoring data a compression scheme is introduced. Fig. 4 illustrates the CMU architecture. The CMU consists of three main modules: the **Packet Analyzer** which analyses the incoming stream of monitoring packets, the **Packetizer** which performs the data compression and data packing and the **Sender** which transmits the packets to the host. The Packet Management RAM (**PM-RAM**) saves the characteristics of a NoC transaction as a parameter set. The parameters are static information of the monitoring packets that belong to the same monitored NoC transaction. The **Payload-RAM** stores the non-redundant information.

CMU Work Flow. To manage incoming monitoring packets and to determine which monitoring packets belong together the PM-RAM and Payload-RAM are used. The PM-RAM is managed by the PM-LUT (Packet Management LUT). The PM-LUT is a register file that keeps track about the status (occupied, available) of PM-RAM rows. Timers are used

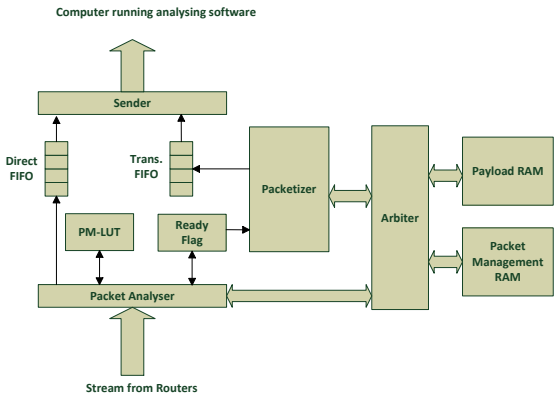

Figure 4: CMU Architecture

to determine if all monitoring packets of a particular NoC transaction are received. If for a specific time window no monitoring packets belonging to a particular NoC transaction are received, it is assumed that all information is received. With this scheme it is not necessary that probes are placed over the entire path. The Packetizer takes the parameters of the NoC transaction from the PM-RAM and the payload information from the Payload-RAM. Afterwards the information is packed into flits and saved in the Transaction-FIFO. The Sender takes the flits from the Transaction-FIFO and sends them to the host. The direct FIFO is used for statistical packets.

Data Compression. As mentioned earlier one of the main tasks of the CMU is to reduce the amount of monitoring data. With a close look on the monitoring packets in packet mode and transaction mode it becomes obvious that the monitoring packets contain a lot of redundant data. This redundancies are exploited in order to relax the time pressure at the interface between CMU and external host. The redundant parts of the packets are highlighted on Fig. 5.

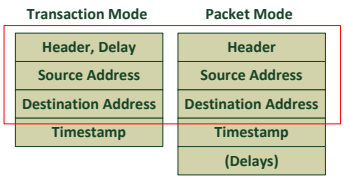

Figure 5: Redundancies within monitoring packets that belong to the same NoC transaction

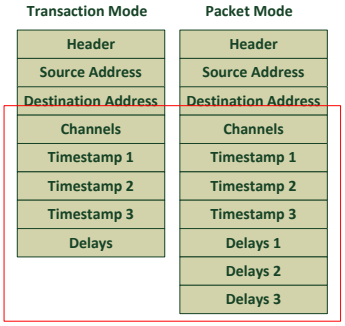

Figure 6: CMU Packet Structure

The header information, the source address and the destination address are directly taken from the original NoC packet. The only non-redundant information within the monitoring

packets (the parts which differ between the monitoring packets that belong to the same NoC transaction) is the probe number, the timestamps and the delays. The probe number (a field in the header flit) identifies on which link the packet was monitored and the timestamp shows the time the header flit passed the channel. The delay flit contains the delays between the NoC flits (packet mode) or the delay field in the header flit the overall packet delay (transaction mode). In order to save bandwidth the redundant information is removed. The same principle is utilized by the Stream Analyzer. The redundant information of monitoring packets is stored once in the PM-RAM and the variable information in the Payload-RAM. In order to reduce the amount of data a new packet format for the communication between the CMU and a host is defined (see Fig. 6). The first three flits are similar to the monitoring packets and are only transmitted once. For both modes a new flit type is introduced. The channel flit contains one to three probe numbers (depending on the number of links the NoC packets passed), which were initially part of the variable information of the monitoring packets. The next flits are one to three timestamps (depending on how many probe numbers the channel flit contains). The timestamps are followed by delay flits. The number of delay flits differs between the modes. In the packet mode for each channel in the channel flit a corresponding delay flit has to be sent. In the transaction mode the delays for each channel in the channel flit can be sent within one flit (one delay flit belongs to one channel flit). If a NoC packet passed more than 3 channels, channel flits, timestamp flits and delay flits are added in the same order.

The number of required channel flits (cfn) for the number of received monitoring packets (x) can be calculated as follows:

$$cfn(x) = \begin{cases} cfn(x+1), & \text{if x mod 3 } > 0 \\ x/3, & \text{if x mod 3 = 0} \end{cases} \quad (1)$$

The CMU packet length (cpl) in the transaction mode:

$$cpl(x) = 3 + 2 * cfn(x) + x \quad (2)$$

and in the packet mode:

$$cpl(x) = 3 + cfn(x) + 2 * x \quad (3)$$

Example: To demonstrate the efficiency of the compression scheme we take an example in which a NoC packet consisting of 8 flits passes 9 channels. The system is assumed to run in the transaction mode. Initially: $9 * 4 = 36$ flits are received. After data compression $3 + 2 * (\frac{9}{3}) + 9 = 18$ flits are left. In this small example the compression scheme reduces the amount of data by 50%. The gain increases the more channels are passed and decreases if the distances in the NoC are relatively small.

4 Results

All modules were tested and evaluated individually to prove their functionality. Afterwards a simulation of a 2x2 NoC, based on ESG NoC components, has been established to test the ESG Monitoring System in a realistic configuration by connecting four processor models with four memory instances. To evaluate the NoC functionality port models of processing elements are used instead of full functional processor cores. To verify the functional and non-functional parameters of the monitoring system the architecture has been implemented on the basis of an FPGA validation system.

Resource Utilization. The whole design has been implemented in synthesiseable VHDL code. The resource costs have been verified by synthesizing the modules individually with corresponding EDA tools for the used FPGA technology. Results are shown in Tab 2. The results for combinatorial ALUTS are given as absolute numbers. The numbers in brackets show the size in relation to the monitored NoC.

Table 2: Resource utilization and clock frequency

	Monitoring Router	Probe	CMU
Combinatorial ALUTS	433 (<1%)	2515 (<4%)	4940(<8%)
Total registers	228	1433	2955
Total block memory bits	0	5632	92928
FMAX	330 MHz	254 MHz	159 MHz

Tracing Performance. The first investigation considers the ratio between the real and the theoretical probe performance. A channel workload of 100 % with different NoC packet lengths has been simulated. The results show a discrepancy between the simulation results and the theoretical values described in Sec. 3. Table 3 shows the overview of the results.

Table 3: Probe Performance Analysis

Mode	Raw	Packet	Transaction
Required NoC packet length to monitor 100% link utilization (theo./real)	/	5 flits/ 7 flits	4 flits/ 5 flits
Max. monitorable channel load (theo/real.)	16.6%/ 14.3%	25% to 100%/ 20% to 100%	25% to 100%/ 20% to 100%

The differences are based on delays, which are caused by implementation details. An example is that the finite state machine (FSM) of the packetizer needs to reach its idle state between the processing of subsequent monitoring packets. Therefore it takes 5 clock cycles to send a transaction packet instead of 4. Also each router in the monitoring system introduces an additional delay of one clock cycle. The simulated workload of 100 % presents the worst case. For real applications and NoCs with smaller bus widths the probes

would achieve better results. The analysis of the CMU shows that the compression scheme introduces a noticeable delay. The data compression algorithm eliminates redundant information. Therefore the delay is balanced and a positive yield is achieved if a specific amount of monitoring packets belonging to the same NoC transaction are received. The following formula shows when the break even point is reached, considering the monitoring packet length (mpl) and the number of received monitoring packets (x):

$$\text{Break Even Point} = 100 - \frac{x * (\text{mpl+1})}{7 * x} \tag{4}$$

If we take the example from section 3 the break even point is reached if the external data rate is $(100 - \frac{9*(4+1)}{7*9}) = 28\,\%$ lower than the internal one. Since the CMU represents the critical bottleneck in the monitoring system it is creating backpressure for the probes. Based on simulations and calculations it is estimated that a monitoring system with 20 probes would be able to process 5.5% average link utilization. The assumptions that are made are that the compression scheme is enabled and the NoC monitoring systems operates with the same clock frequency as the monitored NoC. The external data rate is neglected.

5 Conclusion and Future Work

In this paper we presented a universal and run-time reconfigurable monitoring system that enables performance analysis and debugging support for SoCs. Probes are tracing communication activities within the SoC and provide different abstraction modes to provide as much and as accurate data as possible. The probes are placed at links that potentially carry relevant information. The probe placement offers high flexibility which enables a good trade off between resource cost and obtainable information. The probes send packets to a central monitoring unit which is responsible for the communication to an external host. Since it is not always known beforehand which information is required during an analysis, all components within the ESG Monitoring System can be reconfigured at run-time. This enables to regulate the amount of data and the data granularity without resynthesizing the system. To offer high reusability and to not interfere the monitored system, a dedicated tree based communication infrastructure is established. Also a compression scheme, which removes redundant data, is presented. The evaluation showed that the system benefits from the compression scheme in case the NoC packets pass a certain average number of links and if the external bandwidth is slower than the internal one. The provided data can be used by the system designer to align the SoC architecture or the programming model to achieve a better system performance and resource utilization. Communication debugging in case of malicious behaviour is also supported. The resource utilization of the monitoring system shows reasonable costs in comparison to the cost of the monitored NoC itself. The future work will include the adaptation and usage of the monitoring system for run-time resource management. As mentioned earlier run-time management and performance analysis require similar information. In this sense the CMU will be upgraded with run-time management capabilities to adjust reconfigurable and adaptive parts of an SoC.

References

[ASNH10] A. Alhonen, E. Salminen, J. Nieminen, and T. D. Hamalainen. A scalable, non-interfering, synthesizable Network-on-chip monitor. In *Proc. NORCHIP*, pages 1–6, 2010.

[BDM02] L. Benini and G. De Micheli. Networks on chip: a new paradigm for systems on chip design. In *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*, pages 418–419, 2002.

[CBR+04] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen. An event-based network-on-chip monitoring service. In *Proc. Ninth IEEE Int. High-Level Design Validation and Test Workshop*, pages 149–154, 2004.

[CGB+06] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon. Transaction Monitoring in Networks on Chip: The On-Chip Run-Time Perspective. In *Proc. Int. Symp. Industrial Embedded Systems IES '06*, pages 1–10, 2006.

[DF14] M. Dehbashi and G. Fey. Transaction-Based Online Debug for NoC-Based Multiprocessor SoCs. In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pages 400–404, 2014.

[FPS09] L. Fiorin, G. Palermo, and C. Silvano. MPSoCs run-time monitoring through Networks-on-Chip. In *Proc. DATE '09. Design, Automation & Test in Europe Conf. & Exhibition*, pages 558–561, 2009.

[FPS10] L. Fiorin, G. Palermo, and C. Silvano. A Monitoring System for NoCs. In *Third International Workshop on Network on Chip Architectures. New York, USA*, 2010.

[HAS+08] K. Holma, T. Arpinen, E. Salminen, M. Hannikainen, and T. D. Hamalainen. Real-time execution monitoring on multi-processor system-on-chip. In *Proc. Int. Symp. System-on-Chip SOC 2008*, pages 1–6, 2008.

[MVBT09] S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier. A monitor interconnect and support subsystem for multicore processors. In *Proc. DATE '09. Design, Automation & Test in Europe Conf. & Exhibition*, pages 761–766, 2009.

[PVS+10] B. Phanibhushana, P. Vijayakumar, P. Shabadi, G. Prabhu, and S. Kundu. Towards efficient on-chip sensor interconnect architecture for multi-core processors. In *Proc. Int. SoC Design Conf. (ISOCC)*, pages 307–310, 2010.

[RCM14] M. Ruaro, E.A. Carara, and F.G. Moraes. Tool-set for NoC-based MPSoC debugging — A protocol view perspective. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pages 2531–2534, 2014.

[VG09] B. Vermeulen and K. Goossens. A Network-on-Chip monitoring infrastructure for communication-centric debug of embedded multi-processor SoCs. In *Proc. Int. Symp. VLSI Design, Automation and Test VLSI-DAT '09*, pages 183–186, 2009.

[ZMV+11] Jia Zhao, S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier. A Dedicated Monitoring Infrastructure for Multicore Processors. 19(6):1011–1022, 2011.