A′ARAF: An Asynchronous Router Architecture using Four-Phase Bundled Handshake Protocol

Syed Rameez Naqvi
Department of Computer Engineering
Vienna University of Technology
Vienna, Austria
rnaqvi@ecs.tuwien.ac.at

***Abstract*—The problem of the global clock distribution on multicore systems has rightly abandoned the use of synchronous interconnection networks, but increased the design complexity of the clocked islands. In this work, the design of a completely asynchronous router, with multiple arbitration paths, is presented in detail. The data flow between all the heterogeneous components is based on four-phase handshake protocol. We simulate two 2D mesh networks of different sizes, and propose an evaluation methodology for each of the two important properties: deadlock freedom and reachability. Simulation results show that our networks satisfy both of these properties even against a reasonable flit-injection rate. The inter-router communication is also based on the same single rail, return to zero protocol.**

***Keywords*-Asynchronous; Networks-on-Chip; Router architecture; Four-phase bundled; Single rail; Return to zero (RZ);**

I. INTRODUCTION

Until recently, the multicore systems made use of the standard bus architecture to allow communication between the cores. Although such systems did not pose a great threat to performance, it is expected that in near future buses are going to become a bottleneck with a tremendous increase in the number of processing cores integrated on a single chip. The Networks-on-Chip (NoC) approach proves to be an efficient solution to communication problems, reducing wiring complexity, and thus, power consumption. However, global clock distribution in NoC, in real time bounds, may also not be possible with the billion transistor era approaching fast [1]. The Asynchronous Networks-on-Chip (ANoC) design, a special case of Globally Asynchronous Locally Synchronous (GALS) systems, has gained fame in the recent years. It not only eliminates the need of a global clock signal, but promises to provide power efficiency and higher modularity compared to its synchronous counter parts [2].

The primary job of the NoC, whether clocked or not, is to provide a communication infrastructure between numerous cores (may be processors or IP modules). Normally these cores are clocked, as a result of which, there has to be an interface between the core and the ANoC. Either this interface is made a part of the router, resulting in clocked architecture, or kept separately as a Sync-Async converter between the core and the network. While AEthereal [3] is one of the most famous NoCs that adopt the former approach, Sheibanyrad and Greiner [4] have proposed two efficient Sync-Async converters for an ANoC. In this work we assume the latter approach which seems more attractive, as it significantly reduces the design complexity of the router by eliminating the need of synchronizer circuits, and thus minimizes power consumption at the routing level.

Zeferino et al. [5] have proposed SOCIN NoC, which uses a handshake signal based flow control, where a VALID signal is sent whenever a new flit is transmitted. In this work we present the design of an asynchronous router architecture, which uses the same single rail, return to zero (RZ) handshake protocol. A 2D mesh network, with 2x2 and 4x4 structures, has been simulated. Our simulation results show deadlock freedom even with all interconnects being exercised simultaneously. Furthermore, the novelty of the design, besides being completely asynchronous, rests with the deployment of multiple arbiters per tile. While a single arbiter in a switch would allow only one input port to access the output port at a time, our scheme allows multiple pairs of tiles to communicate independently. For instance, data at the east input port being switched to the west output port, does not hinder switching between the north and south ports pair; as well as data at the west input port can proceed to the east output port simultaneously. This tradeoff between the wiring complexity and the throughput is made in order to compensate the slow nature of four-phase handshake protocol [6], which requires two transitions from each of the sender (producer) and the receiver (consumer), alternately for a successful data transfer.

The rest of the paper is organized as follows. The step-by-step design of the router and network is described in Section II. Section III presents the evaluation methodology. The simulation results are given in Section IV. Section V concludes the paper with future work.

II. ROUTER ARCHITECTURE AND NETWORK DESIGN

The ultimate aim of our work is to introduce a novel fault-tolerance/self-repairing mechanism in the ANoC (not

 ISBN: 978-1-61208-202-8

the topic of this paper though). Hence, we do not emphasize much on the performance aspects related to the throughput of our network. In fact, we intend to keep the network as simple as possible, so that the issues that the complexity of the design adjoins may be avoided; such as, livelocks which may occur even with the state-of-the-art adaptive routing in place. Having multiple arbiters on our tile also adds to the simplicity of the design by reducing the number of candidates for arbitration, eventually eliminating the possibility of a deadlock due to traffic congestion: the primary focus of this work.

To start-off with, we build our interconnection network as a 2D-Mesh of sizes 2x2 and 4x4, which might be scaled later if required. Wormhole switching [7] alongside XY-Routing [8] has been adopted which allows us to: 1) build completely independent flow-paths for the header-, body- and tail- flits, saving power, 2) reduce the design complexity of the Input Controller, and 3) guarantee deadlock freedom.

Table I presents the packet- size and format for all the packet types. The two little-endian, most significant bits (MSBs) indicate the type of flit. "11" represents a header flit, "10" is reserved for a tail flit, and "0x" for the body flits. Since we keep an explicit specifier for the tail flit, we do not need an additional adder to count the number of flits that have arrived. In addition, this gives a flexibility to transmit and receive packets of variable sizes. The addressing scheme that we have adopted is influenced by MANGO [9], in which each pair of bits, starting from 31 down to 0, in the header flit, indicates the next hop; and thus each pair needs to be removed/rotated on every hop so that the next pair can indicate the next subsequent hop. For instance in Table 1(A), "00" at positions 31:30 tells the switch that the incoming data has to be directed to east. Therefore, "00" after being written into the destination latch, must be rotated, thus bringing "10" at its positions. Subsequently, at the next hop, the switch will direct the incoming data to north corresponding to the pair "10". In the same manner, "01" corresponds to the west output port, and "11" to the south. If the next hop is identical to the input port, then the packet is assumed to be directed to the core; therefore, backtracking [10] is not supported.

Fig. 1 presents the block diagram of our fully acknowledged asynchronous router. Primarily, the overall design is divided into two parts: Input Handler (IH) and Output Generator (OG). The former one is responsible for capturing the input data, reserving the appropriate output port arbiter, guiding the incoming data to the output unit, and providing the output unit with associated control signals. On the other hand, the latter guides the incoming data to the appropriate output port based on the control signals it received from the IH. In the following we describe each of these modules briefly.

Table I
PACKET FORMAT AND SIZE: (A) HEADER FLIT, (B) TAIL FLIT, (C) BODY FLIT

1	1	1	0	...	0	1	0	0	1	1
Bit-0	1	2	3	...	28	29	30	31	32	33
Dest-16		Dest-15		...	Dest-2		Dest-1		Flit-Type	
(A)										
1	0	0	0	...	0	0	1	0	0	1
Bit-0	1	2	3	...	28	29	30	31	32	33
Payload									Flit-Type	
(B)										
0	0	0	0	...	0	0	1	0	x	0
Bit-0	1	2	3	...	28	29	30	31	32	33
Payload									Flit-Type	
(C)										

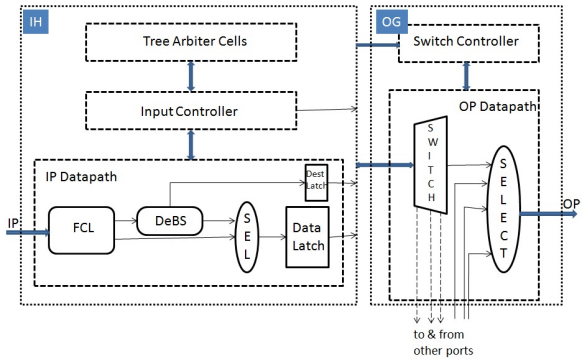

Figure 1. Block Diagram of the Async Router

A. Flit Categorization Logic (FCL)

As explained above, the two MSBs indicate the type of the incoming flit. Depending upon the type, the flit has to be directed to the appropriate unit. The Flit Categorization Logic (FCL) is responsible to: i) identify the flit, ii) report its type to the Input Controller (ICON), and iii) guide it either to DeBS in case of a header flit, or to the Select Module otherwise.

B. Destination Bits Shifter (DeBS)

The DeBS module performs two functions: 1) rotates the bits 31:30 of the header flit to the least significant bit (LSB) places, so that the new pair at places 31:30 indicates the output port of the succeeding node, 2) forwards the rotated bits to the destination latch, fig 2. The latched data then guides the OG in switching the header and the following flits to their appropriate destination ports. In case the incoming flit is not a header flit, DeBS becomes silent (no power consumption, except for the leakage current).

C. Select Module

The Select Module is nothing but a multiplexer without an explicit selection line, whose operation is quite simple. It arbitrates the active input to the output. Therefore, the data-valid control signal associated with every incoming flit

 ISBN: 978-1-61208-202-8

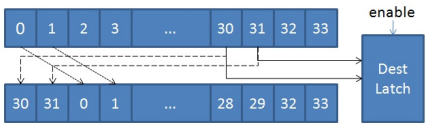

Figure 2. DeBS Module Operation Concept

acts as the selection line of the multiplexer itself. Since the incoming flit can either be a header, body or a tail flit, there is no possibility at the Select module to have a situation where two or more inputs could arrive at the same time. Hence it does not need to comprise any complicated arbitration logic at all.

D. Input CONtroller (ICON)

The Input Controller (ICON) has been modeled as an STG in Workcraft [11] and synthesized using Petrify [12]. Two important functionalities that ICON is made to perform are: 1) on-demand reservation of the mutual-exclusion (MUTEX) element associated with each output port, 2) generating the latch-enable signals both for the destination and the data latches. The destination latch needs to be enabled only with the header flit, while the data latch needs to be enabled with every incoming flit. In fig. 3 we have presented its STG along with explanation of the variables used in Table II. Realizing the level of difficulty in understanding the STG, in the following we briefly describe the operation of the ICON.

At the arrival of the header flit, a request is raised and sent to the ICON, so as indicated by "rh+" in the STG of fig 3. The arrival of the header flit must be followed by the reservation of the MUTEX. As a result, a request is sent to the arbiter associated with the target output port. This is indicated by "rm+". Once the grant from the arbiter is received "gm+", the destination bits and the data must be latched. On confirmation of the data being latched, an acknowledgement "ah" and a request "ro" are respectively sent to the previous and the next nodes simultaneously. The destination latch must not be enabled for the body and tail flits, since header flit is the only one to contain the routing information. The body and tail flits proceed similarly except for the release of the MUTEX with the tail flit. This is done by lowering the request to the MUTEX "rm-", on receiving an acknowledgement from the next node.

E. Arbiter Circuit

We have adopted the conventionally used two-input tree-arbiter-cell (TAC) [13], fig. 4, to allow sharing among all the input ports contending for the same output port. Whichever input port requests (C1req or C2req) first, wins the arbitration (C1gr or C2gr) to access the desired output port. A four-input arbitration circuit can be built by making use of two TACs and a MUTEX as shown in fig. 5. Please note that

Table II
VARIABLES USED IN FIG. 3

Signal	Input/Output	Explanation
rh	Input	request signal from the header flit
rb	Input	request signal from the body flit
rl	Input	request signal from the tail flit
gm	Input	grant/ack signal from the MUTEX
dest_a	Input	ack signal from the destination latch
data_a	Input	ack signal from the data latch
ao	Input	ack signal from the output side demux
rm	Output	req signal to the MUTEX
dest_r	Output	req signal to the destination latch
data_r	Output	req signal to the data latch
ro	Output	req signal to the output side demux
ah	Output	ack signal to the header flit
ab	Output	ack signal to the body flit
al	Output	ack signal to the last/tail flit

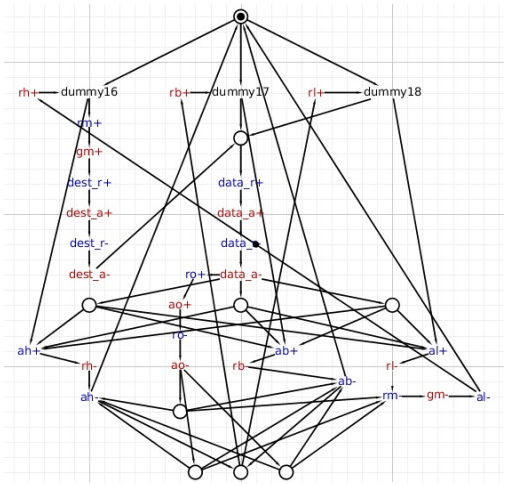

Figure 3. STG of the Input Controller

"r1/r2", "g1/g2" signals in fig. 4 correspond to respective request and grant signals to and from each MUTEX element. An input port keeps hold of the MUTEX until the message is completely transmitted. Although the structure of the tree does not guarantee round robin arbitration, the latter can be achieved by simply replacing a module with priority arbiter if felt necessary. The tree arbiter however, utilizes fewer resources.

The XY-routing algorithm naturally limits the number of permissible switching turns, fig. 6. It is forbidden to switch the data from the north or the south input port to east and/or west output ports. This allows us to keep a different sized arbiter for each output port. For instance, in case of east output port, the number of contending input ports is only two, i.e. west and the core. On the other hand, the number of contending input ports for north and south output ports would be four. However, keeping the generality of the router alive, it is possible to have the four input arbiter module on a reconfigurable partition of the FPGA which allows dynamic

 ISBN: 978-1-61208-202-8

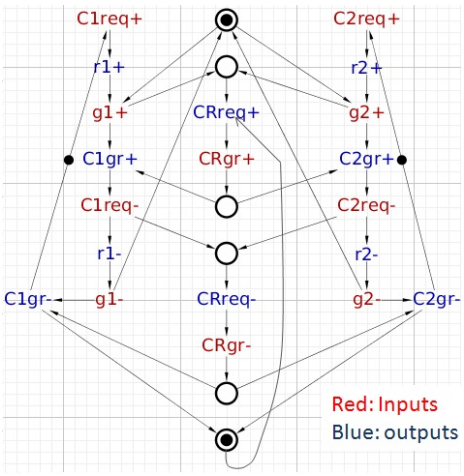

Figure 4. STG of a Two-Input Tree Arbiter Cell

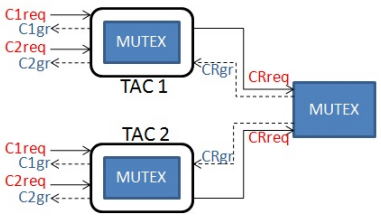

Figure 5. A Four-input Arbiter made from two TACs

exchange of the arbiter circuits as per the routing algorithm (not covered in this paper).

In fig. 7 we have presented the complete circuit diagram of our router with all the important control signals. Some of the wires (req/ack to other input ports) have been deliberately removed from the figure keeping in view the limitation of space. The input demultiplexer together with the three C-gates make up the unit FCL (please refer to Sec. II-A). A C-gate [14] is the most fundamental element for any asynchronous circuit. It works as an AND-gate if the two inputs share the same logic state, and maintains its previous state otherwise. In our router, a number of C-gates are used to ensure the speed independence (SI) [6] property of the circuit; for instance a C-gate placed between DeBS and ICON (please refer to Sec. II-B and Sec. II-D respectively) forces the demultiplexer to keep its output data stable until

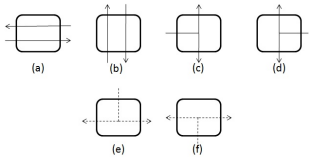

Figure 6. Permissible (a, b, c, d) and Forbidden (e, f) Switching

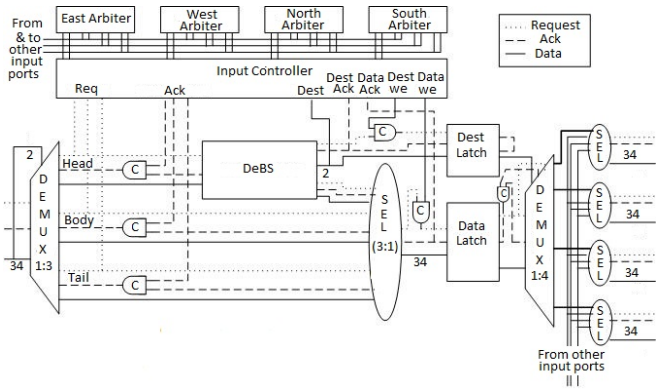

Figure 7. Complete Async Router Architecture

they are acknowledged by both of the receivers. The output demultiplexer, which is controlled by the destination latch, acts as a switch, and together with the four select modules make up the OG module shown in fig. 1. The four select modules and the arbiters are shared between all the input ports, whereas, the rest of the circuit needs to be replicated for each input port to allow parallel execution.

III. EVALUATION METHODOLOGY

Any NoC is expected to guarantee three things: i) deadlock freedom, ii) livelock freedom, iii) reachability to every other node. While any deterministic routing protocol would naturally handle livelocks, we define a methodology to test A′ARAF NoC on both of the other dimensions. The latter is rigorously tested for a 4x4 2D mesh, whereas the former is done for 2x2.

A. Deadlock Freedom

Although a network of size 2x2 does not seem to be an impressive test case, the analysis, however, can be very thorough. The point is to ensure that all the 16 nets (34 bit each) are exercised simultaneously, loading the network with maximum possible traffic. Our methodology is adopted from [15] in which Cota et al. have tested the interconnects for possible faults. Fig. 8, reprinted from the same paper nicely describes the methodology, where “core 0” communicates with “core 3” and vice versa, and “core 1” does with “core 2” and vice versa. We maintain two considerations: 1) The communication pattern remains XY-routing, for example “core 0” sends its packets to “core 3” via “router00”, “router01”, and “router11”, and “core 3” sends its packets to “core 0” via “router11”, “router10”, and “router00”. Please note that each type of an arrow represents communication only in one direction, and maintains XY-routing pattern. For instance, bold arrows represent the packet transfer from “core 0” to “core 3”, and so on for the rest. 2) The order of the flits must be header flit to tail flit.

 ISBN: 978-1-61208-202-8

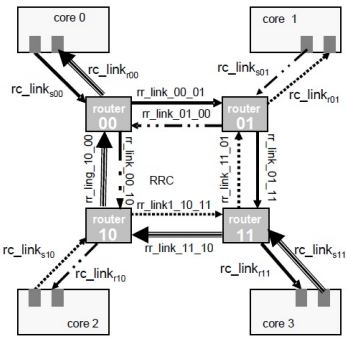

Figure 8. Deadlock Freedom Evaluation Methodology, reprinted from [15]

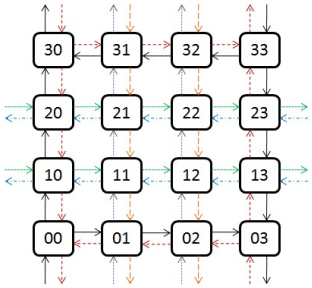

Figure 9. Reachability Evaluation Methodology

B. Reachability

In order to verify that all nodes can access each other, we propose to forward two test packets header, body, tail from each corner node to the nodes at the far ends. For example, “router00” is supposed to forward one packet each to routers “03” and “30” along x+ and y+ axes respectively. In the same manner, “router33” forwards packets to routers “30” and “03” along x- and y- axes respectively. For the two sandwiched rows and columns, each node forwards one packet to the far end. For example, nodes “01” and “31” forward a packet to each other simultaneously, and so on for the remaining pairs. The overall scheme is presented in fig. 9.

IV. SIMULATION RESULTS

According to the methodology described above, we perform four different simulations to test our NoC. All of the simulations are done in Modelsim using a test bench and macro files. We inject 150 packets, one after the other without halting, on each input of a 2x2 network to verify deadlock freedom. All of the packets reach their respective destinations, and are received correctly. Fig. 10 shows the complete propagation of a packet from the south input port of “router00” to the north output port of “router33”. Please note that the header flit changes on every hop, since the two destination bits keep rotating on every node. For example, in the header flit “000000053” (hexadecimal representation),

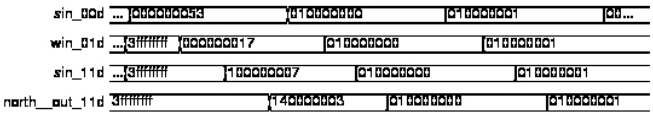

Figure 10. Propagation Path of a Packet

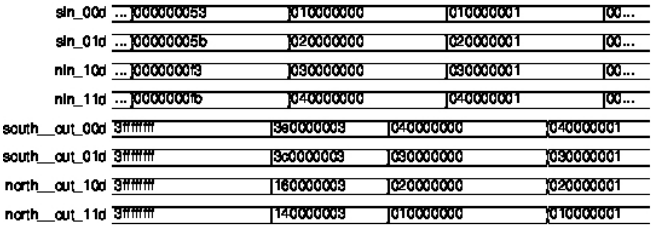

Figure 11. Deadlock Freedom Verification

equivalent to “000...01010011” (34 bits), the last two bits “11” indicate that it is a header flit. The next two bits “00” have to be rotated after the first hop, bringing “10” at their places. So the flit changes to “000...00010111” which is equivalent to “000000017”. Similarly, the flit keeps changing on the rest of the hops. However, the body and tail flits remain the same until they reach the destination. On the other hand, fig. 11 shows a snapshot of the simulation for a few of the initial packets during the deadlock freedom test simulation. It can be seen how packets are correctly transferred between the routers connected on diagonals.

For the reachability test, once again we inject 150 packets (3 flits each) on every input. This time however, the network is 4x4 2D mesh. Two different simulations are performed. Fig. 12 shows the case where we follow the methodology described in the last section. All packets correctly reach their respective outputs. The final simulation, fig. 13, shows the correct working of the arbitration circuit. We deliberately force all the nodes to forward packets to one destination, north output of “router33” (“north__out_33d”). Once again, each node transmits 150 packets to the same target node. The arbiters give access to all the paths one at a time, and lead to correct and complete reception of all of them.

V. CONCLUSION AND FUTURE WORK

In this paper we have presented the design and implementation of A′ARAF, a router for asynchronous NoCs. Our async router supports wormhole switching, and it has been made generic to support any deterministic and adaptive routing algorithm. We have verified two important properties: deadlock freedom and reachability for XY-routing, by heavily loading the network of two different sizes. Simulation results have been presented and discussed in detail.

Although all the deterministic routing algorithms guarantee deadlock freedom, their major drawback is their inability

 ISBN: 978-1-61208-202-8

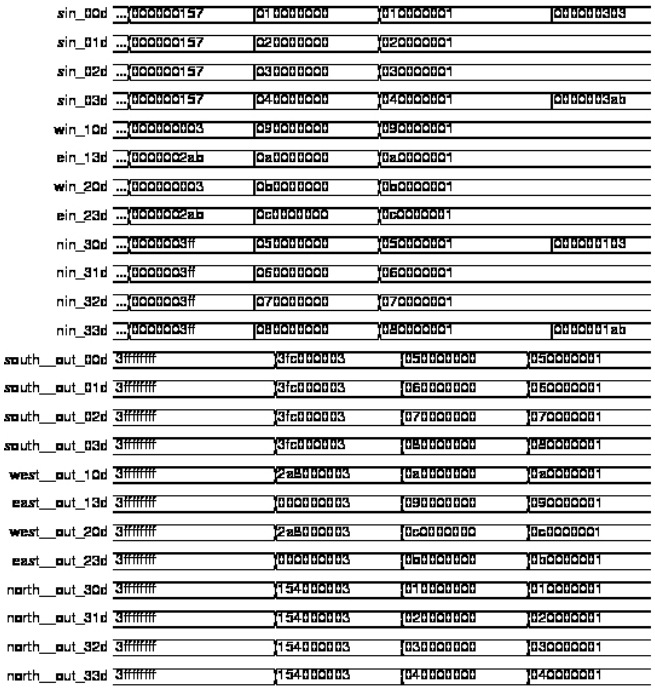

Figure 12. Reachability Verification

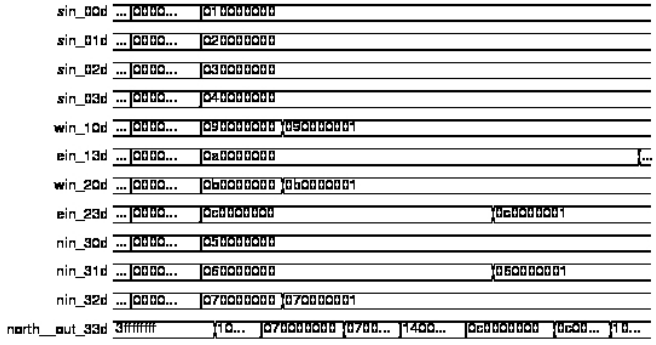

Figure 13. All-to-one Arbitration

to tolerate faults. XY-routing for instance, would immediately result in a complete deadlock once a node fails to forward a packet in the desired direction, or the channel itself becomes permanently fault due to electromigration effect. In future, we aim to address both transient and permanent faults within our routers architecture and the interconnects as well, keeping in view the outstanding problems with the state-of-the-art fault-tolerant NoC designs.

REFERENCES

[1] A. Agarwal, C. Iskander, and R. Shankar, "Survey of NoC Architectures and Contributions," *Engineering, Computing and Architecture*, vol. 3, no. 1, 2009.

[2] Y. Shi, S. B. Furber, J. Garside, and L. A. Plana, "Fault tolerant delay insensitive inter-chip communication," in *Proc. 15th IEEE Symp. on Asynchronous Circuits and Systems (async 2009)*, ser. ASYNC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 77–84.

[3] K. Goossens, J. Dielissen, and A. Radulescu, "Aethereal network on chip: concepts, architectures, and implementations," *Design Test of Computers, IEEE*, vol. 22, no. 5, pp. 414 – 421, sept.-oct. 2005.

[4] A. Sheibanyrad and A. Greiner, "Two efficient synchronous <–> asynchronous converters well-suited for networks-on-chip in gals architectures," *Integr. VLSI J.*, vol. 41, no. 1, pp. 17–26, Jan. 2008.

[5] C. A. Zeferino and A. A. Susin, "SoCIN: A Parametric and Scalable Network-on-Chip," in *Proc. 16th Symp. on Integrated Circuits and Systems Design*, 2003, pp. 169–175.

[6] J. Sparso and S. B. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*. Springer, 2001.

[7] K. M. Al-Tawil, M. Abd-El-Barr, and F. Ashraf, "A Survey and Comparison of Wormhole Routing Techniques in Mesh Networks," *IEEE Network*, vol. 11, pp. 38–45, 1997.

[8] C. Neeb, M. Thul, and N. Andwehn, "Network On-Chip-Centric Approach to Interleaving in High Throughput Channel Decoders," in *Proc. IEEE Int. Symp. on Circuits and Systems*, 2005, pp. 1766–1769.

[9] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the mango clockless network-on-chip," in *Proc. the conf. on Design, Automation and Test in Europe - Vol. 2*, ser. DATE '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1226–1231.

[10] M. Koibuchi, H. Matsutani, H. Amano, and T. Mark Pinkston, "A lightweight fault-tolerant mechanism for network-on-chip," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE Int. Symp. on*, april 2008, pp. 13 –22.

[11] I. Poliakov, V. Khomenko, and A. Yakovlev, "Workcraft — A Framework for Interpreted Graph Models," in *Proc. 30th Int. Conf. on Applications and Theory of Petri Nets*, ser. PETRI NETS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 333–342.

[12] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers," 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.8484

[13] D. L. Dill, "Trace theory for automatic hierarchical verification of speed-independent circuits," in *Proc. 5th MIT conf. on Advanced research in VLSI*. Cambridge, MA, USA: MIT Press, 1988, pp. 51–65.

[14] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.

[15] E. Cota, F. Kastensmidt, M. Cassel, P. Meirelles, A. Amory, and M. Lubaszewski, "Redefining and testing interconnect faults in mesh nocs," in *Test Conf., 2007. ITC 2007. IEEE Int.*, oct. 2007, pp. 1 –10.

 ISBN: 978-1-61208-202-8