# A High Throughput Low Power FIFO used for GALS NoC Buffers

Mohammad Fattah, Abdurrahman Manian, Abbas Rahimi and Siamak Mohammadi
School of Electrical and Computer Engineering
University of Tehran
Tehran, Iran
(m.fattah, abd.manian, ab.rahimi and smohammadi)@ece.ut.ac.ir

*Abstract*— **In Networks-on-chip, increasing the depth of routers' buffers even by a few stages can have a significant effect on average latency and saturation threshold of the network. However, the price to pay could be high in terms of power and silicon area. In this paper, we propose a low power, high throughput asynchronous FIFO suitable for buffers of GALS NoC routers. We consistently compare the performance with regards to power, area and throughput of our FIFO with some different FIFO structures, by exploring their design trade-offs with various number of stages and for different data lengths. These structures are simulated in 90nm CMOS technology with accurate spice simulations, where results show a low power consumption and latency, with a higher throughput. Finally, a back-annotated HDL model of a 4x4 mesh network, wherein a fully asynchronous router is implemented, shows better average latency, saturation threshold and power tradeoffs, using the proposed FIFO.**

***Keywords-FIFO; buffer; networ-on-chip; NoC; Asynchronous; GALS***

## I. INTRODUCTION

Networks-on-Chip (NoCs) [1-4] are new approaches for high throughput, and scalable design of Multi-Processor Systems-on-Chip. This method is to divide an overall system into multiple independent processors that are connected through a network infrastructure; i.e. NoC paradigm is to route packets instead of interconnects [4]. An asynchronous design is a good candidate for NoC paradigm. It enables the GALS systems to operate in multi clock frequencies [5], and moves the synchronization issue only in the point of interfacing the synchronous IP with the NoC infrastructure [6].

In a NoC infrastructure, each router employs buffers in its input/output ports to overcome interconnects bottleneck of the network. The depth of these buffers has significant effect on average packet latency and network saturation threshold [6],[7]; however they take a large portion of silicon area of the NoC [8],[9]. This motivates us to design buffers that have acceptable area and power characteristics while they provide higher throughput and performance with smaller latencies.

In this paper we propose a new asynchronous self-timed FIFO which can be suitably used as NoC I/O buffers. The proposed FIFO has better metrics in comparison to FIFOs introduced so far in literature. Rest of the paper is organized as follows: First, different types of FIFOs which have been proposed will be presented in Section II. Their benefits and drawbacks will be also discussed in this section. Then the new proposed FIFO will be explored in Section III. In addition to accurate spice simulation, this structure is applied to a 4x4 mesh network to be examined as NoC buffers. Spice simulation results, besides saturation threshold and average packet latency of the mesh network are extracted and compared in Section IV. Finally, Section V concludes the paper.

## II. RELATED WORKS

A variety of 4-phase bundled-data asynchronous FIFOs have been proposed so far in literature. Muller pipeline, as a fundamental structure, is a particularly simple one. Its pipeline controller consists of a simple C element and an inverter gate. But its main drawbacks [10] are that only every other stage stores data when the pipeline is full; and each stage handshaking should tightly interact with both neighboring stages. In [10],[11] a fully-decoupled latch controller is introduced where all stages store data when the pipeline is full and each stage handshaking on the input channel completes without any interaction with the output channel. However, the throughput decreases as the controller is more complicated and therefore, slows down the handshaking.

The pipelines introduced above are designed for computational asynchronous circuits, and working with combinational logics inserted between their stages, where data move through all stages. But in a buffering FIFO data is stored with no processing, and the input data can be stored in only one stage and be read from the same stage for the output handshake. This makes the FIFO consume less dynamic power in comparison to a ripple through pipeline. Ono and Greenstreet [12] have proposed a modular FIFO which follows the asP* handshaking protocol, and is synthesizable by standard cell logics without any particular asynchronous cell. However, this FIFO does not follow the conventional 4-phase handshake protocol and requires some timing considerations: the input request must go low strictly within a time range after the acknowledge signal. And the range may vary versus different depths and word lengths. Sheibanyrad et al. [13] proposed a FIFO based on the fully-decoupled pipeline. The input data is demultiplexed on the corresponding fully-decoupled stage and the corresponding stage is multiplexed on the output channel using a proposed domino controller. Their FIFO exhibits the throughput of a fully-decoupled FIFO in addition to some multiplexing and demultiplexing penalties.

Note that all the introduced FIFOs are self-timed because of their bundled data natures which require some timing engineering.

978-0-7695-4076-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ISVLSI.2010.44

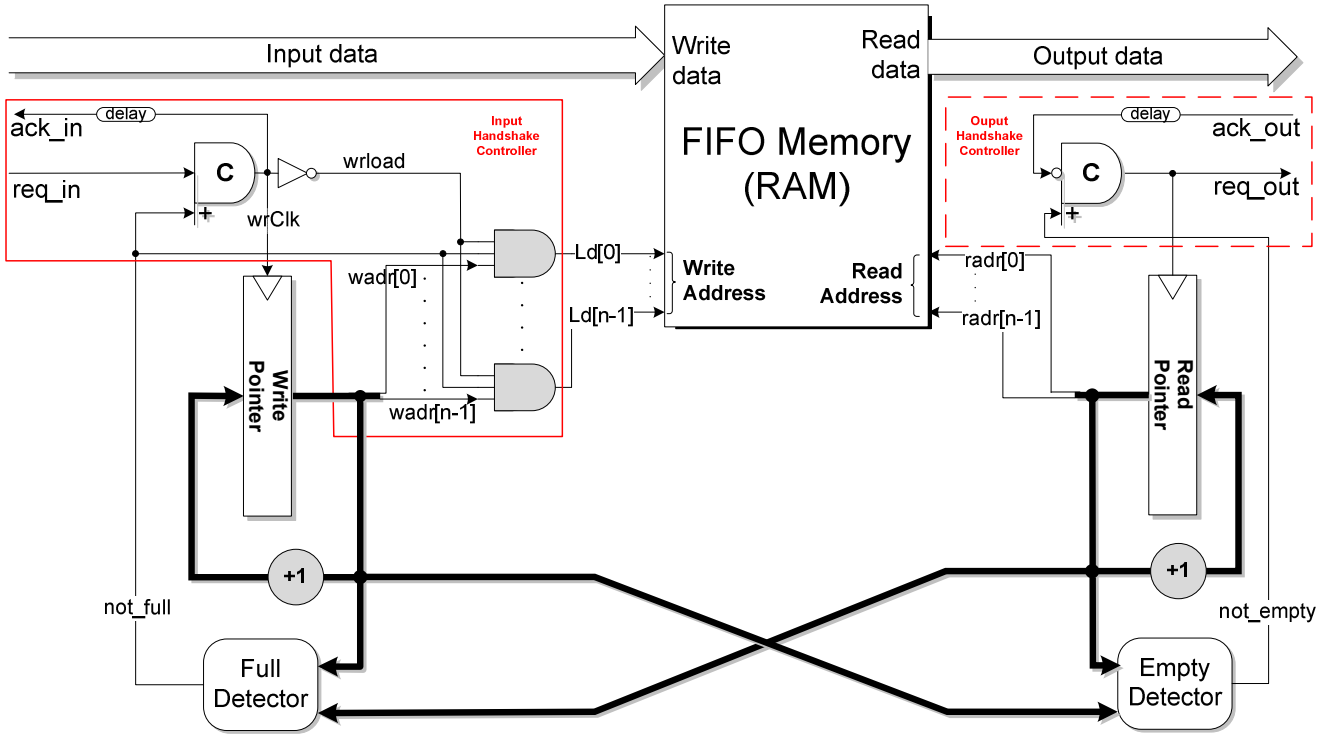


Figure 1. General architecture of the proposed FIFO

## III. FIFO ARCHITECTURE

Our proposed FIFO architecture for an *n* deep configuration FIFO is shown in Figure 1. It consists of a one-hot addressed memory (RAM), read and write pointers, full and empty detectors, and input and output handshake controllers. This FIFO produces "extended-early" outputs where data is valid while request is high, and latches are normally open (latches are open while acknowledge is low), like simple four phase bundled data latch controller [10]. This FIFO completely follows the 4-phase bundled data handshake protocol, and thus it can be easily inserted in the design as a 4-phase bundled-data pipeline stage. In the following subsections, first the general functionality of the FIFO is introduced, and then different parts of the FIFO will be explored in details.

### A. General Architecture

The input data is written into the RAM row selected by the write pointer whenever the request has not yet been acknowledged and the FIFO has at least one free space to store it. When the input request goes high and the FIFO has free space, the handshake controller asserts the acknowledge signal and removes *wrload*. Subsequently the write address increases, and both full and empty detectors update their output.

Once the write phase has finished and the write address was increased, the arrival of a new data is indicated to output port, which raises the request as soon as the empty detector has indicated the existence of data and the previous handshaking has finished. After asserting the *req_out* signal, the read address is increased and causes the write and empty detectors to update their output.

### B. Detailed Architecture

As mentioned, the FIFO architecture is composed of four main sections: a one-hot addressed memory, read and write pointers, full and empty detectors, and input and output handshake controllers.

#### 1) Memory

The memory module has been implemented using latches and pass-transistors, see Figure 2, which could be replaced by a memory plane for better performance. Since addresses are one-hot, no address decoder for the memory module is needed. The increment and decrement modules will also be degraded to simple shift registers, although it implies more storage in comparison to binary address.

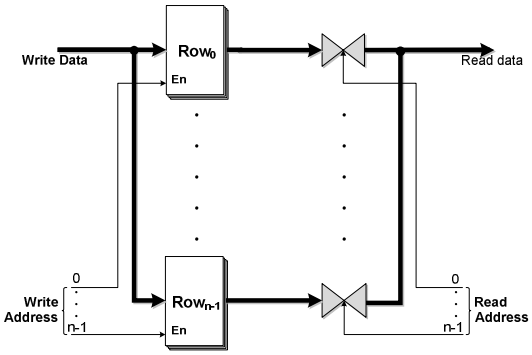


Figure 2. One-hot addressed memory structure

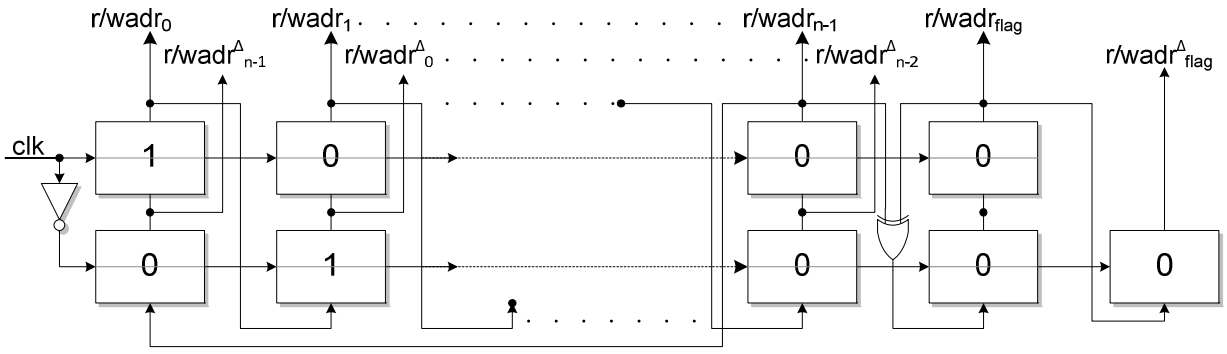


Figure 3. Address pointers and their half-period shifted signals, obtained from master latches with a circular wire shifting.

*2) Handshake Controllers*

The handshake controllers consist of an asymmetric C-element, as shown in Figure 1, and *n* AND gates for input channel. When the environment requests to enqueue a new word into the FIFO, the handshake controller blocks the request till the FIFO has a free space to store it. And when the FIFO has a free space and *ack_in* line is zero, it enables *Ld* signal regardless of a write request, which makes latches normally open like Muller pipeline. The delay element on *ack_in* line is to guarantee that *not-full* flag is valid before *wrload* signal rises.

FIFO requests to send out its data when it is not empty upon lowering of ack_out signal. Similarly the delay element on *ack_out* signal is to ensure validation of *not-empty* flag before lowering of *ack_out*.

*3) Address Pointers*

The write and read pointers will be zero $(00\cdots01)_{\text{one-hot}}$ to point to the first row of the memory, after resetting the FIFO. This equality of pointers shows the empty state of the FIFO. Assume a condition where the sender writes words into the FIFO, while the output port is stalled and does not respond to the *req_out* signal. The FIFO will be full exactly after *n* writes, and the write pointer is zero again, equal to read pointer. It shows the read and write pointers are equal, when the FIFO is either full or empty [14]. There are several techniques to distinguish the full and empty states, when the read and write pointers are equal. One is to consider a flag for each row of the FIFO to store its full or empty state. This is done in [12] by inserting an SR-Latch for each place. Another way to indicate the FIFO state, as used in this work, is to add one extra bit to both write and read pointers [14]. These extra bits are noted as $radr_{flag}$ and $wadr_{flag}$, and toggle whenever their pointers circulate and become zero again, see Figure 3.

Since the write pointer is trigged on *ack_in* rising edge, the *not-full* flag could be evaluated before lowering of *ack_in* signal, which enables us to have normally open latches. To have an "extended-early" output, where data is stable while *req_out* is high, a half-period shifted value of read pointer (noted as $radr^{\Delta}$) is obtained from the master latches of DFFs, see Figure 3. Thus the read pointer which is injected to memory would change its value on falling edge of the *req_out* clock. Although it stores the next address ($radr^{\Delta}+1$), thanks to one-hot address coding the correct value can be easily recovered by a circular wire shifting. The half-period shifted value of pointer's flag ($radr^{\Delta}_{flag}$) is driven by an extra latch which is fed by the actual value of the flag.

*4) Full/Empty Detectors*

In the case the FIFO is full and a read operation occurs, the full detector should not announce a *not-full* state before the read operation is finalized to avoid a write to take place on the same location at the same time. To avoid explained read/write race, as shown in Figure 4, full detector is fed by a half-period shifted value of read pointer that changes its value after having finished the current output handshake, i.e. after falling edge of *req_out*.

## IV. Results and Comparison

To evaluate this work, the proposed FIFO and some other introduced ones (Muller, Fully-Decoupled, asP*-based [12], Domino-Controlled [13]) were simulated using accurate Spice simulation. We have described some available FIFOs at the gate level in Verilog and used them as buffers in routers of a 4x4 mesh network. Spice simulation results for throughput, energy, latency and area of different FIFO models will be shown and discussed. Subsequently the simulated network, saturation threshold and average packet latency for different FIFOs will be explained.

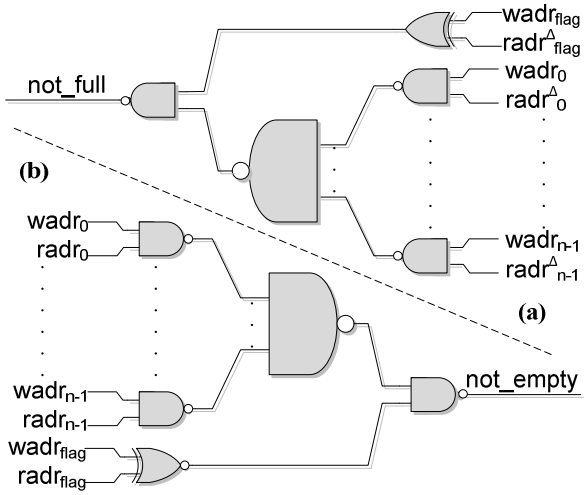


Figure 4. Full (a) and empty (b) detectors: A half-period shifted value of read pointer is injected to full detector, preventing read/write race.

### A. Spice Simulation

The proposed design, as well as all other FIFO designs, is modeled using accurate Spice simulations in 90nm CMOS PTM. A Spice library of basic logics and different C-element types are modeled and used to construct all FIFOs. As mentioned before, the designed FIFO, as all the other ones, belongs to the class of self-timed circuits and needs elaborate and engineering timing assumptions for reliable correct operation [10]. Timing engineering and evaluation of results were done without consideration of wire delays for all considered FIFOs.

To extract proper results a random Galois LFSR sequence of data words is injected to the FIFOs. Note that timing engineering of the asP*-based FIFO [12] turned out to be very difficult for large data width; thus results of this design are only discussed in 8-bit data versus different depths (3, 4, 5). Furthermore for an *n* depth Muller pipeline, twice stages are needed. In view of the fact that all FIFOs are made by the same common basic library, and the reported results do not include wiring cost for all of results; a comparison between attained results would be reasonable.

Figure 5 shows Spice simulation waveform of a 3-stage 8-bit data word FIFO, in which the handshaking is done with 1GW/S speed at both sides. As could be seen, the "extended-early" output is satisfied and the new output data changes before asserting *req_out* and is valid while it is high.

The maximum handshaking speed that could be achieved simultaneously at both sides is called throughput and is shown for 8-bit data words in Figure 6 for different FIFO depths. The maximum achieved throughput for 32-bit data word FIFOs is also reported in Table I. It could be seen that the FIFOs' throughputs are sorted according to their controllers' complexity. Since the handshake controllers of our proposed FIFO are made of simple asymmetric C-elements, it exhibits the highest throughput, as it could be expected. The Muller and fully-decoupled pipelines' throughputs are constant versus different depths because their respective controllers do not change when the FIFO depth increases. And due to serious timing engineering of the asP*-based FIFO its throughput falls down considerably for a 5-stage design. As predicted, throughput of the FIFO proposed in [13] is less than fully-decoupled one, due to additional overheads.

To have a comparison between power consumption of different FIFO types, the amount of the energy which would be dissipated to enqueue and send a data word is reported in Table I. It could be seen that as said because of ripple through nature of Muller and fully-decouple pipelines the dissipated energy for a single data word increases sharply as a function of FIFO depth. Energy consumptions in this work and asP*-based are considerably higher than in domino, due to usage of Flip-Flops for each stage; as nearly 45% of energy is used by address pointers of this work in an 8-bit 5-stage FIFO.

As latency of FIFOs, delta times between *req_in* and *req_out* for a data word is measured in two situations: injecting data to an empty FIFO and to a FIFO in a stable condition where data words traverse at the highest possible throughput. These two latency numbers are reported in Table I respectively, separated with a comma. It could be seen that the proposed FIFO's latency is less than others in a stable state, and is near to domino's and less than that of the others in an empty situation.

As mentioned, all FIFOs have been modeled using a library of basic elements. As a consequence, number of transistors used in each FIFO would be a fair metric to express silicon area which will be occupied by each FIFO, as listed in Table I.

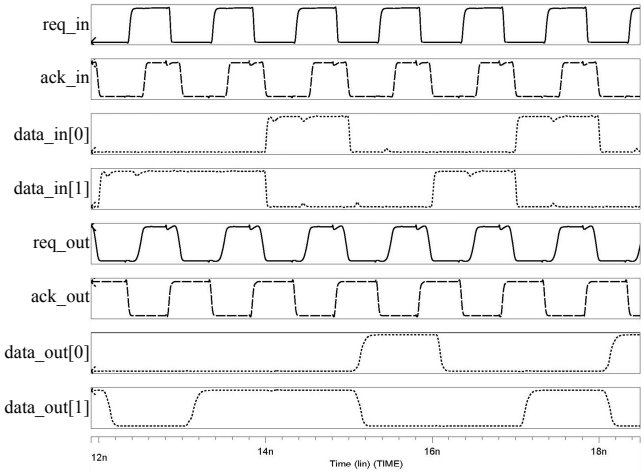

Figure 5. Waveform of the proposed FIFO in 1GW/S data transfer.

Table I. Spice simulation results for different configurations of the modeled asynchronous FIFOs.

| Metric | Depth: | 8-bit data word | | | | | 32-bit data word | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | *This work* | *Muller* | *asP*-based* | *Fully-decoupled* | *Domino* | *This work* | *Muller* | *Fully-decoupled* | *Domino* |
| Throughput (GW/S) | 3 | 2.8 | 2.4 | 2.4 | 1.55 | 1.32 | 2.8 | 2.6 | 1.65 | 1 |
| | 4 | 2.6 | 2.4 | 2.28 | 1.55 | 1.27 | 2.6 | 2.6 | 1.65 | 1 |
| | 5 | 2.5 | 2.4 | 1.9 | 1.55 | 1.27 | 2.4 | 2.6 | 1.65 | 1 |
| Energy(pJ) | 3 | 0.208 | 0.438 | 0.217 | 0.217 | 0.173 | 0.427 | 1.481 | 0.768 | 0.379 |
| | 4 | 0.211 | 0.585 | 0.238 | 0.366 | 0.173 | 0.438 | 1.940 | 1.031 | 0.375 |
| | 5 | 0.214 | 0.732 | 0.280 | 0.456 | 0.177 | 0.447 | 2.425 | 1.282 | 0.385 |
| Latency(pS) | 3 | 386, 556 | 520, 596 | 228, 1180 | 638, 824 | 392, 1870 | 386, 390 | 505, 550 | 598, 770 | 380, 2600 |
| | 4 | 426, 637 | 704, 785 | 238, 1240 | 855, 1050 | 392, 2770 | 435, 440 | 679, 728 | 798, 982 | 380, 3730 |
| | 5 | 482, 691 | 886, 974 | 317, 2060 | 1069, 1270 | 390, 3590 | 491, 475 | 852, 903 | 1000, 1190 | 378, 4750 |
| #Transistors | 3 | 980 | 840 | 832 | 561 | 940 | 2324 | 3192 | 1737 | 2260 |
| | 4 | 1236 | 1120 | 1108 | 748 | 1252 | 3015 | 4256 | 2316 | 3019 |
| | 5 | 1504 | 1400 | 1452 | 935 | 1568 | 3740 | 5320 | 2895 | 3778 |

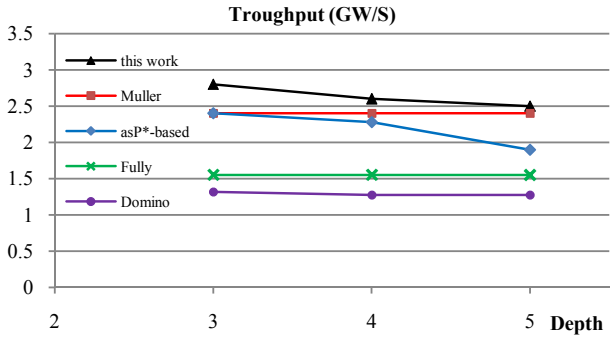


Figure 6. Maximum achievable throughput for different FIFO structures.

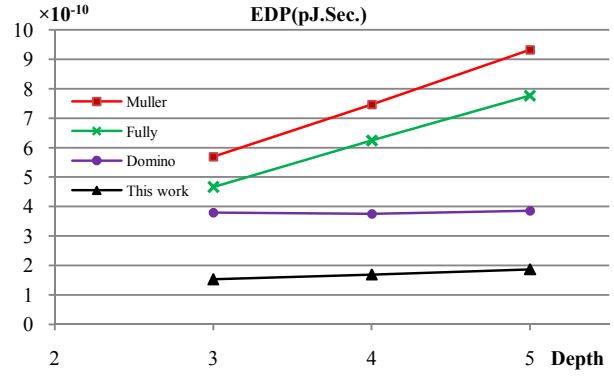


Figure 7.EDP for different FIFOs with 32-bit data word.

Neither the throughput nor energy dissipation is a good candidate to express benefits and costs of a design. Thus we rather use EDP which is obtained by dividing energy by throughput of a FIFO. This is shown in Figure 7Figure 7.EDP for different FIFOs with 32-bit data word. for different 32-bit length designs.

As shown, domino FIFO dissipates the least energy compared to all simulated FIFOs. Compared to domino, the proposed FIFO in its worst case has about 100% gain in throughput and latency with only 20% penalty in consumed energy with a nearly equal but better area in its 8-bit 5stage configuration. However Muller pipeline exhibits 4% higher throughput in 32-bit 5stage state, at the expense of 4 times more energy dissipation and 1.5 times larger transistor count. Accordingly, the comparing results would be better for other FIFOs configurations.

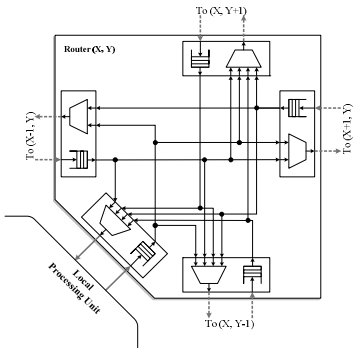


Figure 8. Overall router architecture of the simulated network.

## B. Network Simulation

In the previous subsection, we explored transistor-level simulation results of our FIFO compared with some other ones. To evaluate the performance of the proposed FIFO, we have employed it as buffers in routers of a 4x4 mesh network. Hereunder we explain routers' structure and network architecture. After that, extracted saturation threshold and average packet latency of the network over different network loads will be shown and compared.

### 1) Router Architecture

The NoC used in this paper is a fully asynchronous NoC, based on QNoC [5] and ASPIN [6], using two cascaded Flip-Flops to connect synchronous IP cores to the asynchronous network. The four-phase bundled data handshake protocol is employed for implementing the routers.

As shown in Figure 8, each router addressable with a two dimensional number (X, Y) contains five (input/output) ports that connect it to its neighbors and to the local IP core. To route packets between these five different sides, from input ports to output ports, asynchronous NoC uses the distributed X-First algorithm guaranteeing the in-order-delivery property for the network [6], which has been implemented in the input ports of the router.

Wormhole data flow control is used where each packet is divided into some flits. Input ports of each router buffers incoming flit and when a header flit of a packet is received, the destination address field is analyzed and the flit as well as its following ones is forwarded to the corresponding output port. The output port arbitrates between simultaneous packets and sends out packets to the neighbor's input port without any precedence. The Round-Robin scheduler [15] which implements the Token-Ring arbiter [16] is used in output ports for arbitrating different incoming packets.

### 2) Simulation Environment and Results

Delay and energy parameters from the Spice library are extracted as a function of gates' fanout, and are back-annotated in a Verilog HDL library. The described router is modeled in gate level, using back-annotated library, and a 4x4 mesh network is constructed by behavioral model local IP cores as network traffic generators and analyzers. As interconnect links between two neighboring nodes, delay parameters of our delay-insensitive current-mode link proposed in [17] are used.

The proposed FIFO as well as domino is modeled using back-annotated library and used as buffers of routers' input ports. FIFOs are selected to be 5-stages deep with 34bit data word length, since each flit is 32 bits of data next to 2 control bits.

A uniform [18] type of traffic is generated by local IPs, where each node generates a constant rate of traffic and sends it to all other nodes with a constant probability. Packets are 8flit length in the injected traffic. And the network load varies from 1M packets per second for each router till network saturates. As shown in Figure 9 the proposed FIFO results in less packet latency and more network saturation threshold compared to domino FIFO. The power consumption of the routers (excluding links and synchronizing Flip-Flops) for different network loads is shown in Figure 10 for both employed FIFOs.

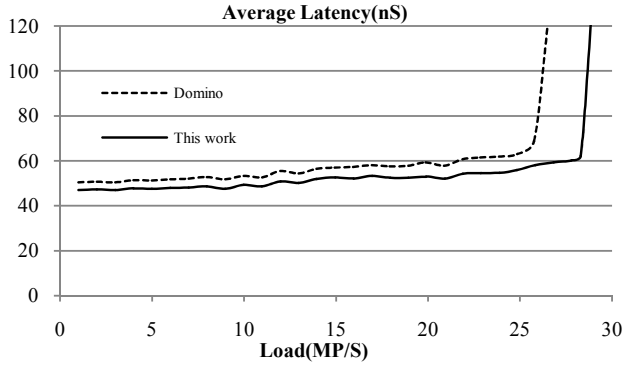

Figure 9. Average packet latency of the network over different loads.

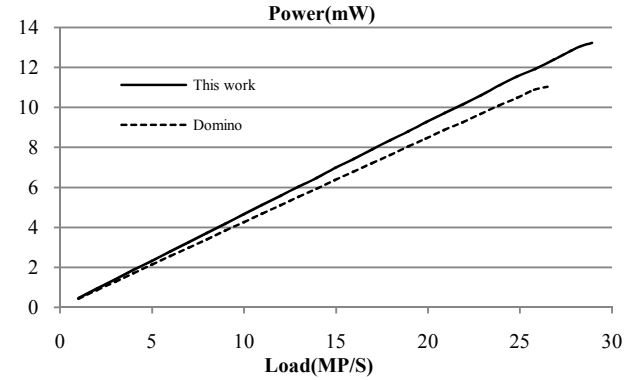

Figure 10. Routers power consumtion versus different loads.

Table II shows more details of results of the simulated networks for two employed buffers. It could be seen that nearly half of the consumed power is dissipated and 65% of the area is occupied by input buffers.

Thanks to one-hot address coding, it is easy to find out portion of the FIFO which is occupied, and thus power consumption of the network could be reduced by applying our history based DVS policies [19] as future work.

## V. CONCLUSION

In this paper, we proposed an asynchronous one-hot memory addressed FIFO for NoC buffers. The Spice simulation results exhibit better energy, throughput and area tradeoffs compared to some previous works. The proposed architecture is also suitable for being used in different asynchronous designs, as well as in NoCs for interfacing IP cores to the network which can be done by any type of synchronizer. The proposed FIFO structure was examined in a 4x4 mesh network and a 12% gain in network saturation threshold reached. With the proposed buffer we have found a trade-off between throughput and power metrics, where DVS can be applied for better power results as a future work. Double cascaded Flip-Flops, used as a synchronizer, decrease the throughput and therefore as a future work, the design of the FIFO should be modified to make it more suitable for GALS interface and reach higher throughput.

Table II. Detailed results of network simulation

| Buffer | Sat. threshold | Power ratio | Area ratio |
|---|---|---|---|
| This work | 28.8 MP/S | 48% | 65% |
| Domino | 25.7 MP/S | 45% | 67% |

## REFERENCES


[1] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," 2000, pp. 250-256.

[2] A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, D. Lindqvist, and A. Hemani, "Network on chip: An architecture for billion transistor era," *Proceeding of the IEEE NorChip Conference*, 2000, pp. 166–173.

[3] G. De Micheli and Benini, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, pp. 70-78.

[4] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," 2001, pp. 684-689.

[5] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar, "An asynchronous router for multiple service levels networks on chip," 2005, pp. 44-53.

[6] A. Sheibanyrad, A. Greiner, and I. Miro-Panades, "Multisynchronous and Fully Asynchronous NoCs for GALS Architectures," *Design & Test of Computers, IEEE*, vol. 25, 2008, pp. 572-580.

[7] Jingcao Hu, Umit Y. Ogras, and Radu Marculescu, "System-Level Buffer Allocation for Application-Specific Networks-on-Chip Router Design," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, 2006, pp. 2919-2933.

[8] Jingcao Hu and R. Marculescu, "DyAD - smart routing for networks-on-chip," *Design Automation Conference, 2004. Proceedings. 41st*, 2004, pp. 260-263.

[9] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo network-on-chip," *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, 2003, pp. II-113-II-116 vol.2.

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

[11] S. Furber and P. Day, "Four-phase micropipeline latch control circuits," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 4, 1996, pp. 247-253.

[12] T. Ono and M. Greenstreet, "A modular synchronizing FIFO for NoCs," *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, IEEE Computer Society, 2009, pp. 224-233.

[13] A. Sheibanyrad and A. Greiner, "Hybrid-Timing FIFOs to use on Networks-on-Chip in GALS Architectures," *International Conference on Embedded Systems and Applications (ESA'2007)*, Las Vegas, Nevada, USA: 2007.

[14] E.C. Clifford, "Simulation and Synthesis Techniques for Asynchronous FIFO Design," *Synopsys Users Group Conference, SNUG 2002, also available at. www.sunburst-design.com/papers.*

[15] A. Sheibanyrad, "Asynchronous Implementation of a Distributed Network-on-Chip," Pierre et Marie Curie (UPMC), 2008.

[16] A.J. Martin, "The Design of a Self-timed Circuit for Distributed Mutual Exclusion," Jan. 1983.

[17] M. Fattah, S.A. Moghaddam, and S. Mohammadi, "A Hazard-Free Delay-Insensitive 4-phase On-Chip Link Using MVCM Signaling," *Proceedings of the 2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, IEEE Computer Society, 2009, pp. 61-66.

[18] S. Koohi, M. Mirza-Aghatabar, S. Hessabi, and M. Pedram, "High-Level Modeling Approach for Analyzing the Effects of Traffic Models on Power and Throughput in Mesh-Based NoCs," *Proceedings of the 21st International Conference on VLSI Design*, IEEE Computer Society, 2008, pp. 415-420.

[19] A. Rahimi, M. E. Salehi, M. Fattah, and S. Mohammadi, "History-Based Dynamic Voltage Scaling with Few Number of Voltage Modes for GALS NoC," *5th IEEE International Conference on Future Information Technology (DATICS-FutureTech'10)*, Busan, Korea: .