Alexandria Engineering Journal (2017) xxx, xxx–xxx

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

ORIGINAL ARTICLE

A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure

Rajeev Kamal *, **Juan M. Moreno Arostegui**

Electronic Engineering Department, Universitat Politecnica de Catalunya, Barcelona, Spain

Received 26 January 2017; accepted 13 February 2017

KEYWORDS

Network-on-Chip (NoC);
GALS;
Bidirectional channel;
Multi-synchronous FIFO;
On-chip communications

Abstract To enhance the performance of on-chip communications of Globally Asynchronous Locally Synchronous Systems (GALS), a dynamic reconfigurable multi-synchronous router architecture is proposed to increase network on chip (NoC) efficiency by changing the path of the communication link in the runtime traffic situation. In order to address GALS issues and bandwidth requirements, the proposed multi-synchronous bidirectional NoC's router is developed and it guarantees higher packet consumption rate, better bandwidth utilization with lower packet delivery latency. All the input/output ports of the proposed router behave as a bi-directional ports and communicate through a novel multi-synchronous first-in first-out (FIFO) buffer. The bidirectional port is controlled by a dynamic channel control module which provides a dynamic reconfigurable channel to the router itself and associated sub-modules. This proposed multi-synchronous bidirectional router architecture is synthesized using Xilinx ISE 14.7 and FPGA Virtex 6 family device XC6VLX760 is considered as target technology and its performance is evaluated in terms of power, area and delay.

1. Introduction

Networks-on-Chips (NoCs) are widely accepted as a promising approach for addressing the communication challenges associated with future Multi-Processors System-on-Chip (MPSoCs) in the face of further increases in integration density [1–4]. In the recent multi-core systems, e.g. Google Brain project, NoC routers require to transfer a large amount of communication data among processors [5,6]. Networks-on-Chips (NoCs) [7] represent a widely accepted solution to connect multiple cores, due to its scalability, flexibility and high bandwidth properties. Several regular patterns for NoC architectures have been proposed and implemented in [8]. These NoC architectures have gained popularity due to their scalability, simplicity and flexibility. In such NoC architectures, main switching component (router) behaves like a synchronous island. These routers are connected via two unidirectional links with the neighboring router [9]. A typical 2-D mesh 5-stage pipelined virtual-channel based NoC router is shown in Fig. 1.

* Corresponding author.
E-mail address: rajeevkamal062@gmail.com (R. Kamal).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

http://dx.doi.org/10.1016/j.aej.2017.02.019

Please cite this article in press as: R. Kamal, J.M.M. Arostegui, A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.02.019

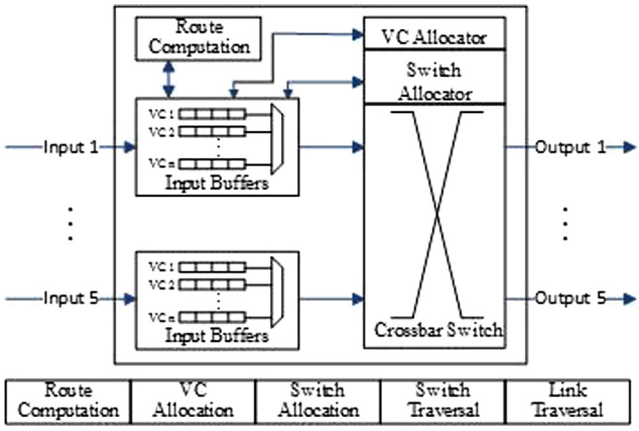

Fig. 1 A typical 2-D mesh 5-stage pipelined virtual-channel based NoC router.

A 5-stage pipelined router architecture is scalable and flexible but it has some drawbacks in terms of bandwidth and throughput, thus being not suitable for the Globally Synchronous Locally Synchronous (GALS) infrastructure [10].

Dynamic reconfigurable router architectures have been suggested to enhance NoC proficiency by changing the path of the links in the runtime traffic situation, since normal traffic pattern is commonly uneven distributed in the core of the network. To accommodate for such kind of traffic pattern, a bidirectional NoC ((BiNoC) [11] architecture was presented and further optimized as FSNoC [12] and BiLink [13]. However, the greater part of the accentuation on the current reconfigurable NoC architecture has been concentrating on upgrading the design of the switch itself. The optimization of the interconnection between two adjacent routers is hardly satisfied. On the other side, network coding was introduced in communication domain to maximize channel bandwidth to accomplish noteworthy improvement in the system throughput. Over a single physical channel, an extra coding unit was inserted between each pair of routers to enable the data transmission from both ends at a time. BiLink [13] uses an additional link stage among routers and transmits two flits in one link for every clock cycle using stage pipelining if both routers require utilizing the current link. To increase additional effective bandwidth, each link path can be arranged in every clock cycle to supply different traffic loads from both sides.

Above discussed bidirectional NoCs are not suitable for the globally asynchronous, locally synchronous (GALS) architecture and it is believed that GALS approach is best suited for the MPSoC and packet based network-on-chip. Initially Dally and Towles [1] introduced the concept of packet switching within the on-chip network communication structure and same concept followed by the Benini and Micheli in [3]. In the research articles [14–17], different types of NoC and GALS based SoC are explained.

In this paper, we contribute to the architecture and implement a bidirectional NoC router for the multi-synchronous GALS infrastructure. A better approach to design a bidirectional NoC for the multi-synchronous GALS system is shown in Fig. 2. Its consists of five bidirectional ports which support data transfer across two clock domains of arbitrary phase and frequency and it is best suited for Distributed Scalable Predictable Interconnect Networks (DSPIN) [18].

Here we propose microarchitectural modification of available bidirectional NoC's router that improves delay, area and power consumption. We contributed to the architecture and implemented a bidirectional NoC's router for the multi-synchronous GALS systems using parameterized RTL code. The contributions of this paper are summarized as follows.

- A novel multi-synchronous FIFO buffer architecture that supports valid/ready flow control mechanism at all the different input/output ports of the routers.
- A new channel directional control protocol, called Dynamic Channel Control, is proposed for the bidirectional channels. This is based on handshaking protocol.
- Implementation of the proposed NoC's router and detailed analysis in terms of area, delay, and power dissipation.

2. Proposed bidirectional NoC architecture

To enhance the performance of on-chip communication within the GALS infrastructure, we propose a Multi-Synchronous Bidirectional NoC architecture as shown in Fig. 3. This architecture is the extended version of an available BiNoC [11] architecture that is suitable for the GALS infrastructure.

In this paper, we contribute a novel multi-synchronous FIFO buffer at all the interfaces (input/output ports) of the router and a dynamic channel controller that is based on the handshaking protocol. This dynamic channel controller controls the data flow direction at each link and is implemented using a pair of ASM (Algorithmic State machine) charts namely Talker and Listener. It provides higher performance, deadlock free, and starvation free communications. Based on the motivational examples in BiNoC [11] and FSNoC [12], our proposed MBiNoC improves performance by providing more flexibility in using bandwidth. Here bidirectional channels are used to dynamically increase the bandwidth at runtime and new input buffer architecture is introduced to support intra-router data path within the GALS infrastructure. Normally in conventional NoC architecture, two adjacent routers use two unidirectional links in opposite directions for communication of data, but in our multi-synchronous bidirectional NoC, the data link between adjacent routers is able to transmit data in any direction dynamically.

2.1. Related work

Foruque et al. [19] presented a unique configurable data channel called 2X-link. This can change its maintained bandwidth at runtime on demand basis. 2X-Links use tri-state logic to support bidirectional transmission for an adaptive on-chip communication infrastructure. In a BiNoC [11], a dynamic reconfigurable architecture has been suggested to change the path of the links in the runtime traffic situation. To accommodate uneven distributed traffic in the core of the network, BiNoC architecture was initially introduced and further optimized in FSNoC [12] and BiLink [13]. Main difference between these architectures and 2X-Link is how the link direction control is implemented. In BiNoC, FSNoC and BiLink channel direction decision is made using a distributed channel direction control protocol, whereas in 2X-Link, the proposed direction decision mechanism is centralized and each link is configured

Please cite this article in press as: R. Kamal, J.M.M. Arostegui, A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.02.019

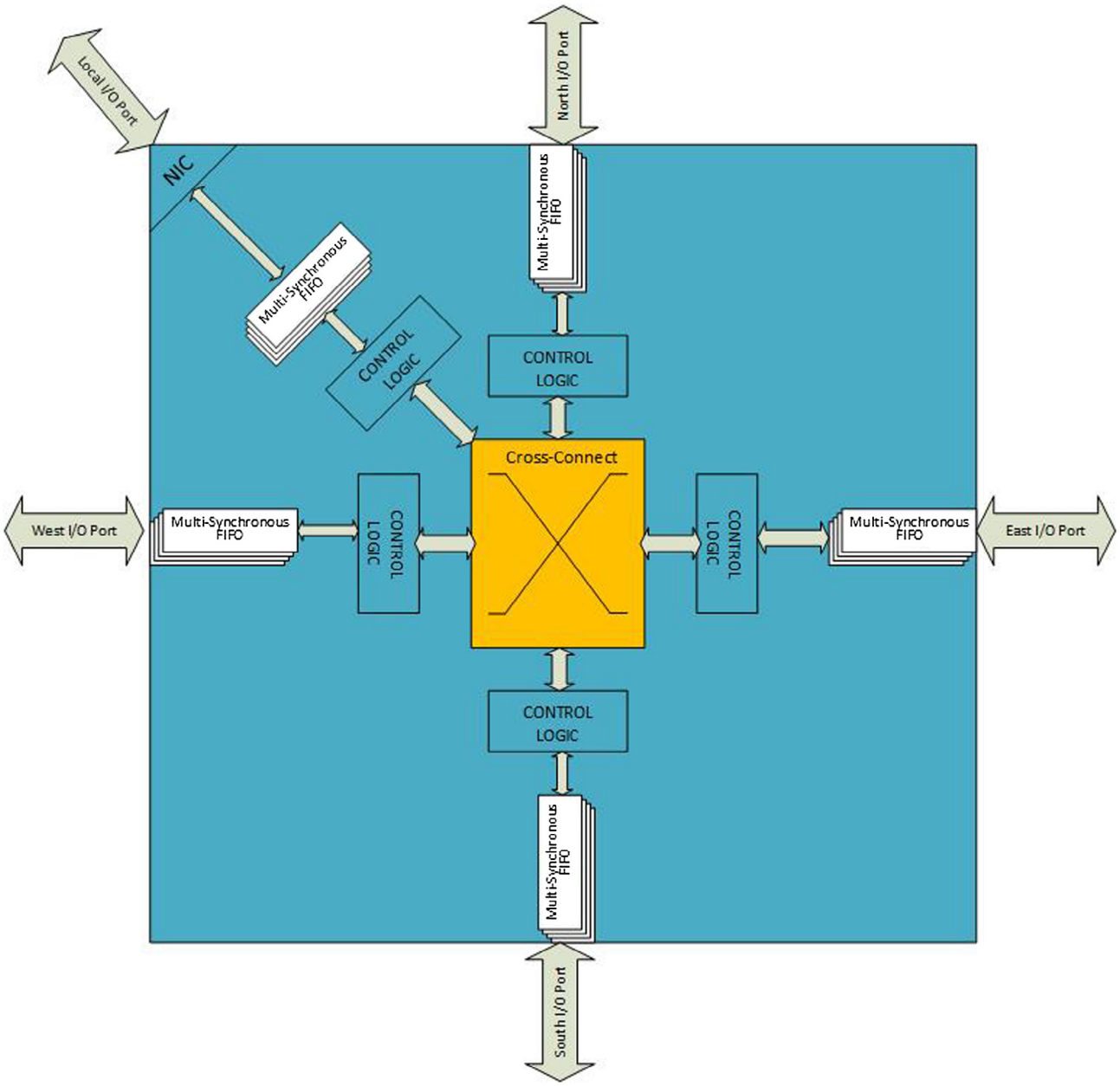

Fig. 2 Top/block level diagram of proposed multi-synchronous bidirectional NoC.

in advance based on application bandwidth requirements. Cho et al. [20] proposed a pressure based channel direction control protocol. This is different from the acquisition based channel control protocol used in BiNoC. In this paper, we implement an acquisition based dynamic channel control protocol that supports multi-synchronous GALS infrastructure. Table 1 shows the comparison of different bidirectional NoCs with proposed one.

2.2. Dynamic channel control and reconfigurable inputs/outputs

After contributing to the traditional architecture, the proposed architecture is called as Multi-synchronous BiNoC (MBiNoC) architecture. It mainly consists of three parts: multi-synchronous FIFO module at the interface of the router, dynamic channel control module and router internal architecture. Details about the multi-synchronous FIFO and its implementation will be discussed in the next section. The detailed schematic of the bidirectional ports implementation of our proposed router is shown in Fig. 4. The *direction_control* signal is generated from the dynamic channel control and assigned properly to avoid conflict. Each port within the proposed architecture can be used as either an input port or an output port. Dynamic channel control is shown in Fig. 5 and is based on handshaking protocol. Here two data channels are available called data channel #1 and data channel #2. This controller performs mainly two tasks: first it decides the link

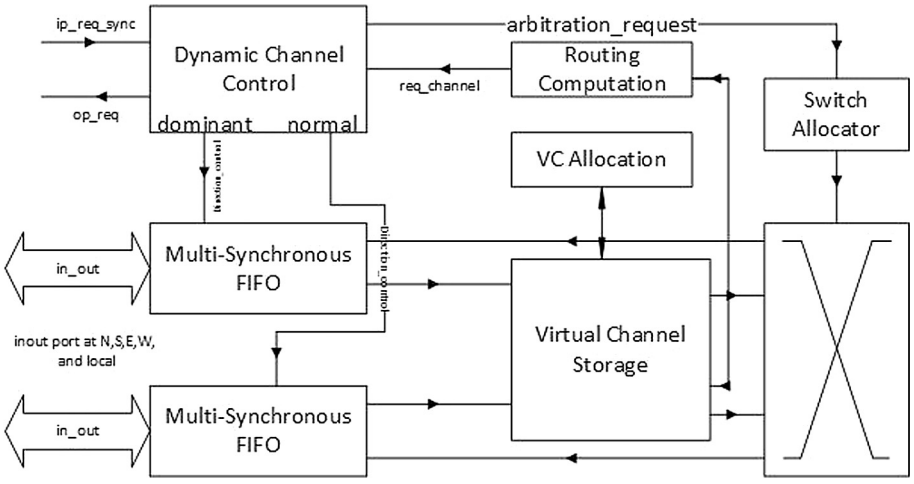

Fig. 3 Detailed architecture of proposed NoC router.

Table 1 Comparison of different bidirectional NoC with proposed one.

Name	Channel Direction Control (CDC)	Type
2X-Link	Centralized CDC	Synchronous Domain
BiNoC	Distributive CDC	Synchronous Domain
BiLink	Aggressive bidirectional Link	Synchronous Domain
FSNoC	Acquisition based CDC	Synchronous Domain
Cho et al.	Pressure based CDC	Synchronous Domain
Proposed NoC	Distributive Dynamic Channel Control	Multi-Synchronous Domain

direction dynamically and second it generates the arbitration request signal to the switch allocator.

Fig. 5 shows the operation of dynamic channel control module. Here we consider that Talker ASM (algorithmic state machine) and Listener ASM work with different frequencies. So, signal *op_req_T* is generated from router #1 and reaches router #2 as *ip_req_syncL*. Signals *clk_T* and *clk_L* are the talker clock and listener clock respectively, and communication between two neighboring routers is performed using a two flip-flop synchronizer. The detailed operation of dynamic channel control will be discussed in another section.

3. Multi-Synchronous FIFO

As have already discussed, Multi-Synchronous FIFO is used at the interface of the proposed router architecture, which supports dynamic, extendable and power efficient multi-clock architectures. It is very useful to transfer data between different clock domain modules. This projected architecture of FIFO allows the allocation of data among entirely separated clock domain modules with minimum cycles of latency between sender and receiver.

An abstract form of Valid/Ready protocol can be built around a multi-synchronous FIFO queue as shown in Fig. 6. This multi-synchronous FIFO model has two interfaces called Push and Pop, which indicates its connecting module when the FIFO is empty or full. The AND gate outside the FIFO queue controls the push and pop operation, i.e. push cannot be performed when signal full is asserted and pop cannot be performed when signal empty is asserted. The write (push) and read (pop) operations are performed on the write clock and read clock domain respectively. In both cases (write and read), it can be observed that a data transfer occurs when the corresponding handshaking signal valid or ready is asserted.

Many bi-synchronous and asynchronous FIFOs are implemented by different authors in literatures for specific applications [21]. Top-level architecture of bi-synchronous FIFO is given by Dally, Poulton and Baltch [22] but details of design and analysis are missing in the literature. In the literature fully asynchronous FIFO frequently appears as discussed by Ebergen [23] and Molnar et al. [24], but these designs do not utilize

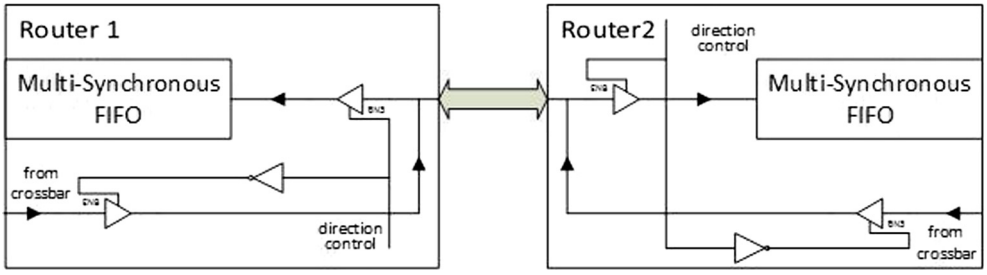

Fig. 4 Bidirectional port implementation.

Please cite this article in press as: R. Kamal, J.M.M. Arostegui, A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.02.019

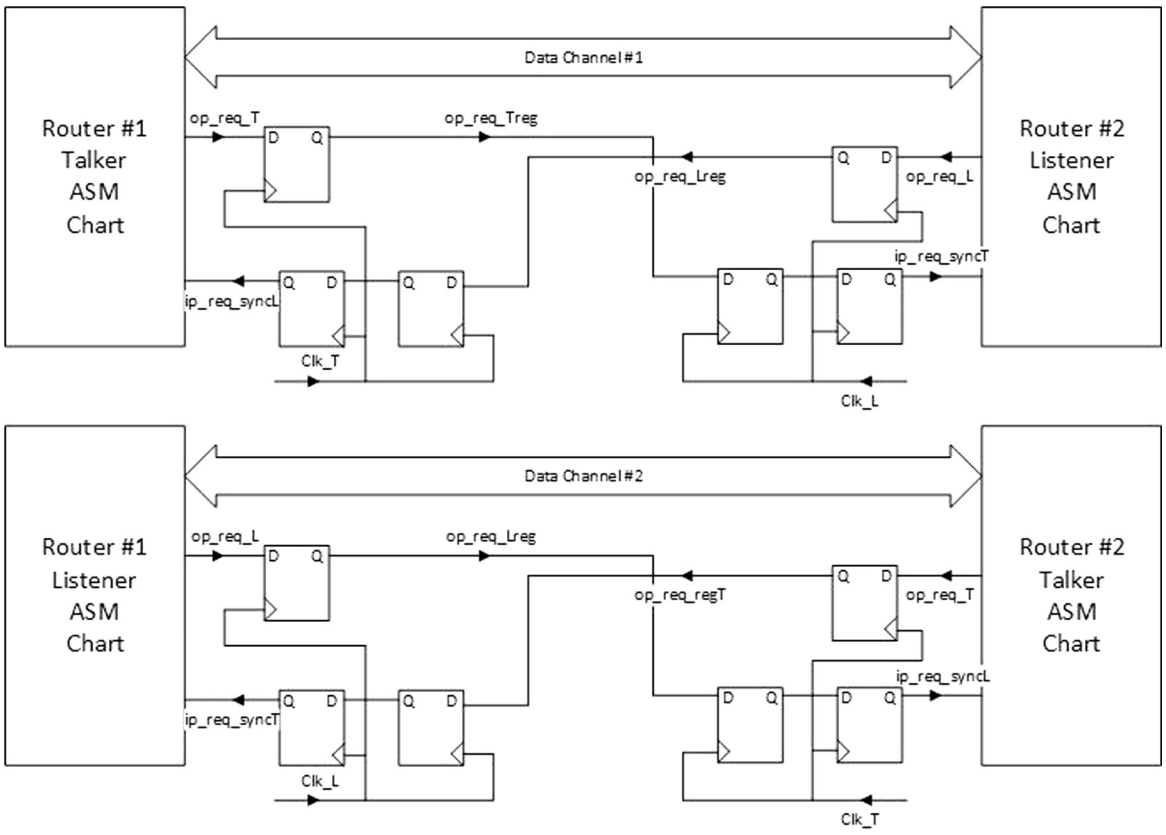

Fig. 5 Dynamic channel control.

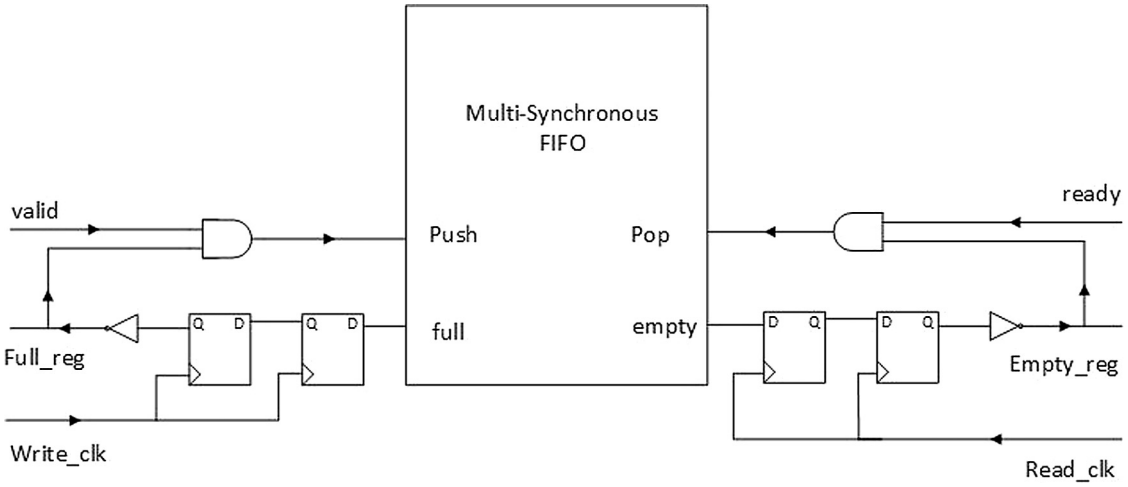

Fig. 6 Valid/ready protocol can be built around a multi-synchronous FIFO.

the clocks, that's why it is difficult to apply synchronization between different clock domains. In the work presented by Greenstreet [25] the clocks are derived from the same global clock (mesochronous). Chakraborty et al. [26] presented a FIFO design that first calculates time to develop an estimate for frequency difference, before transferring the data. A linear FIFO architecture is presented by Seizovic [27] for data synchronization; it has some limitations in terms of initial latency. Apperson et al. [28] introduced a scalable and robust bi-synchronous FIFO architecture, but its memory size is fixed, so that it does not provide high throughput and therefore is cost ineffective. Similarly, Panades and Greinear proposed a FIFO architecture that is well-suited for GALS system, but is not appropriate for mesochronous systems [29]. Chelcea and Nowick proposed an alternative FIFO architecture for the application of GALS system [30]. This design is based on a Register File and each register has its own full and empty flags. This style is suitable for small FIFO only.

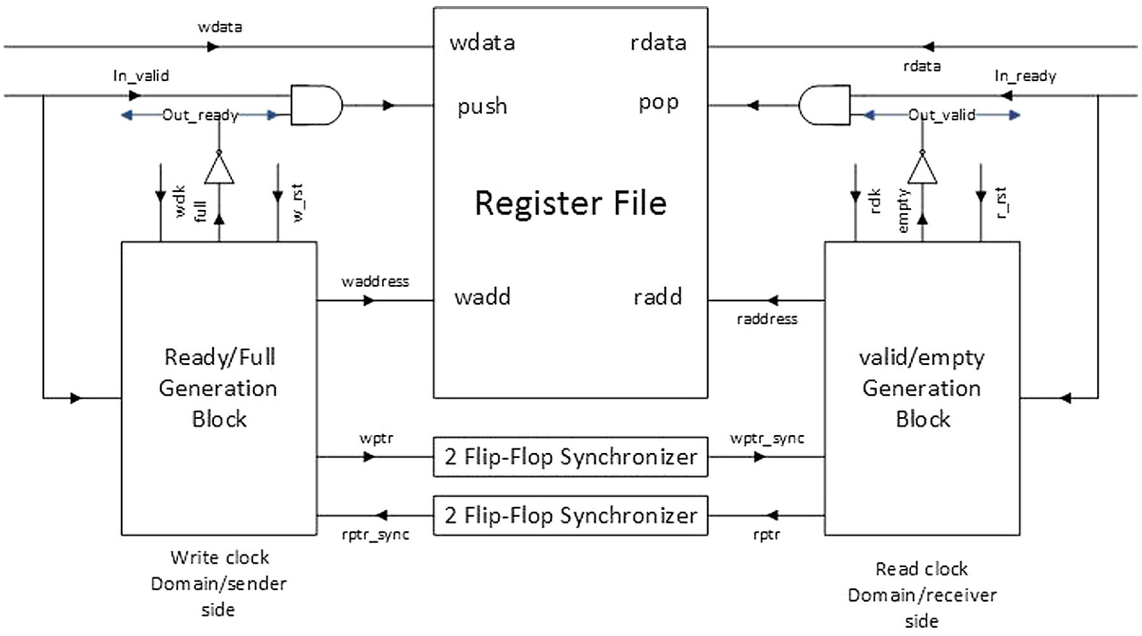

Fig. 7 Proposed architecture of Multi-Synchronous FIFO buffer.

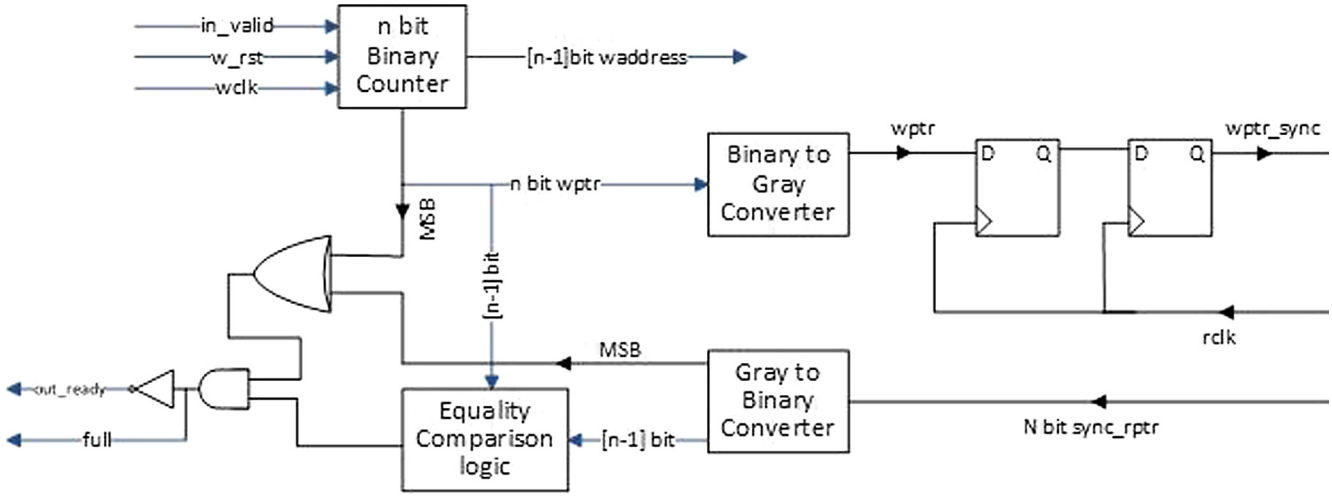

Fig. 8 Detailed architecture of Full/Ready generation module.

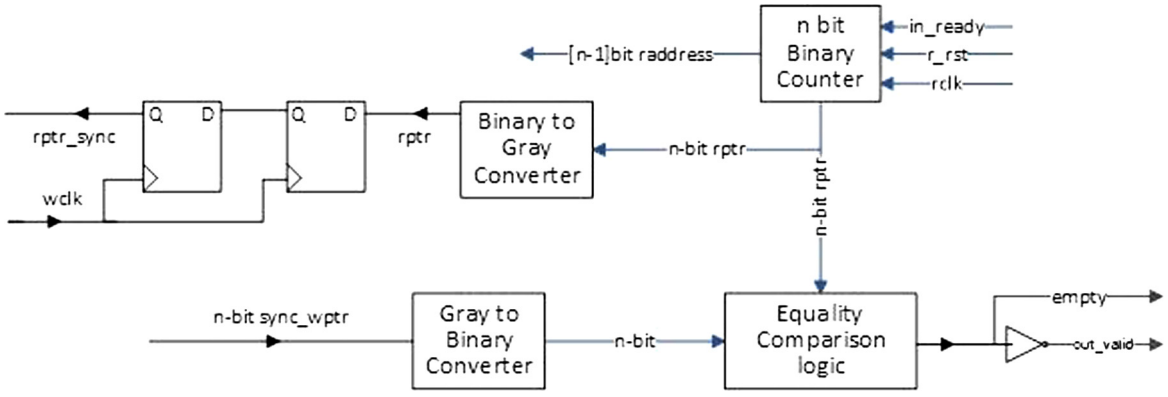

Fig. 9 Internal architecture of Valid/Empty generation module.

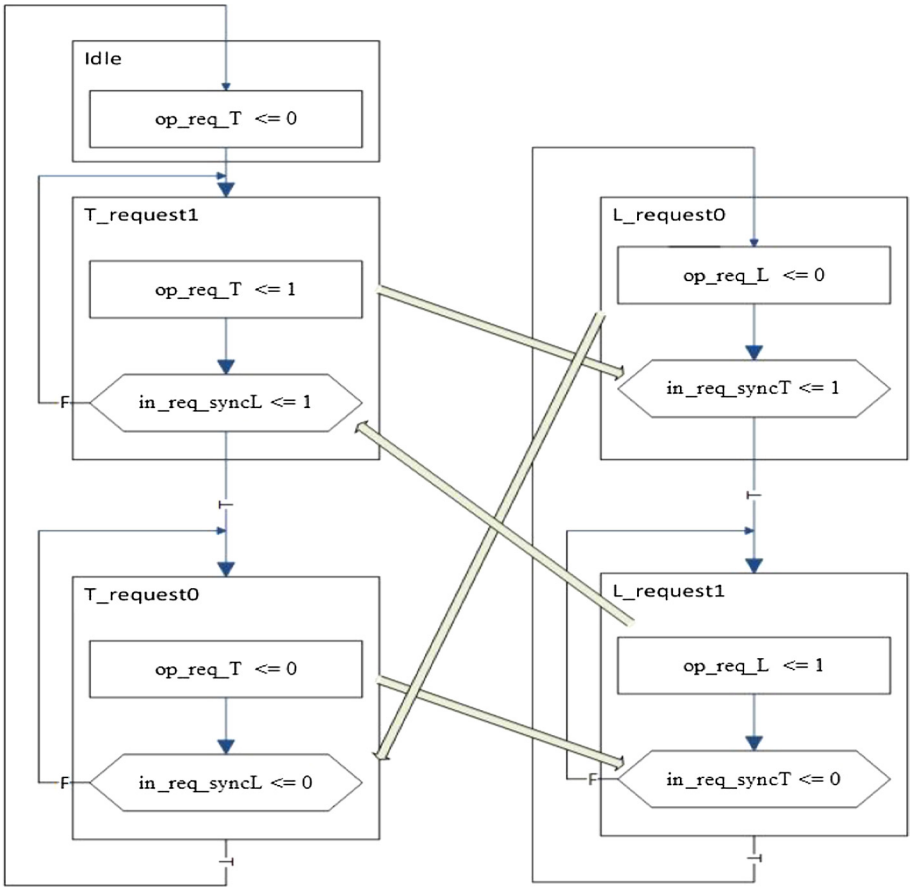

Fig. 10 ASM chart for the inter-router transmission.

3.1. Multi-clock FIFO buffer architecture

Fig. 7 describes the proposed architecture of Multi-Synchronous FIFO buffer architecture model, which supports data transfer between two different arbitrary clock domains as well as ready/valid protocol. The sender clock (wclk) and receiver clock (rclk) are not related to each other in terms of their phase and frequency. This block diagram consists of mainly five sub-modules: Register File, Ready/Full generation block, Valid/Empty generation block, and two flip-flop synchronizers. On the left hand side of Fig. 7 *write* logic is presented and similarly in the right hand side of Fig. 7 *read* logic is presented.

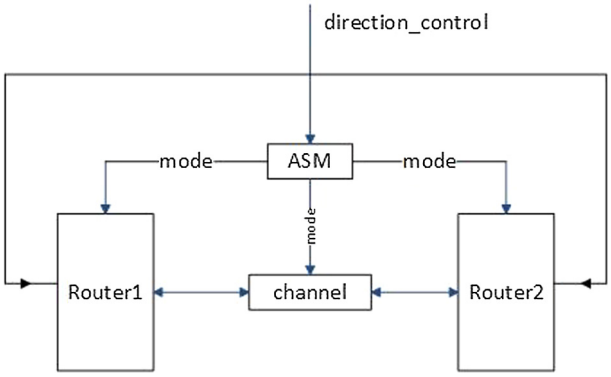

Fig. 11 Mode controller of ASM.

3.2. Gray code pointers

The suggested architecture uses two pointers called *read* and *write* address pointers to track utilization of the FIFO memory. The size of the pointer is $log_2(n) + 1$, where n is the memory word size. The pointer requires one more bit for the generation of *full*//*ready* flag and *empty*/*valid* flag. There are two types of pointer available, called *write* pointer and *read* pointer, where write pointer always points to next word to be written and similarly read pointer always points to the current FIFO word to be read.

Due to the metastability issues, the pointers are transferred to a Gray code format before the clock domain crossing. The implementation of *Binary-to-Gray* conversion and *Gray-to-Binary* conversion requires special circuit that is based on *xoring* operations. In the case of *Binary-to-Gray*, an *n-bit* binary vector (B_{n-1}, $B_{n-2}\ldots B_2$, B_1, B_0) can be used to convert to n-bit gray coded vector (G_{n-1}, $G_{n-2}\ldots G_2$, G_1, G_0) as shown in given Eq. (1), where $\oplus$ indicate the XOR function.

$$G_{n-1} = B_{n-1}, G_{n-2} = B_{n-1} \oplus B_{n-2}, G_{n-3} = B_{n-2} \oplus B_{n-3}, \ldots, G_1 = B_2 \oplus B_1, G_0 = B_1 \oplus B_0 \quad (1)$$

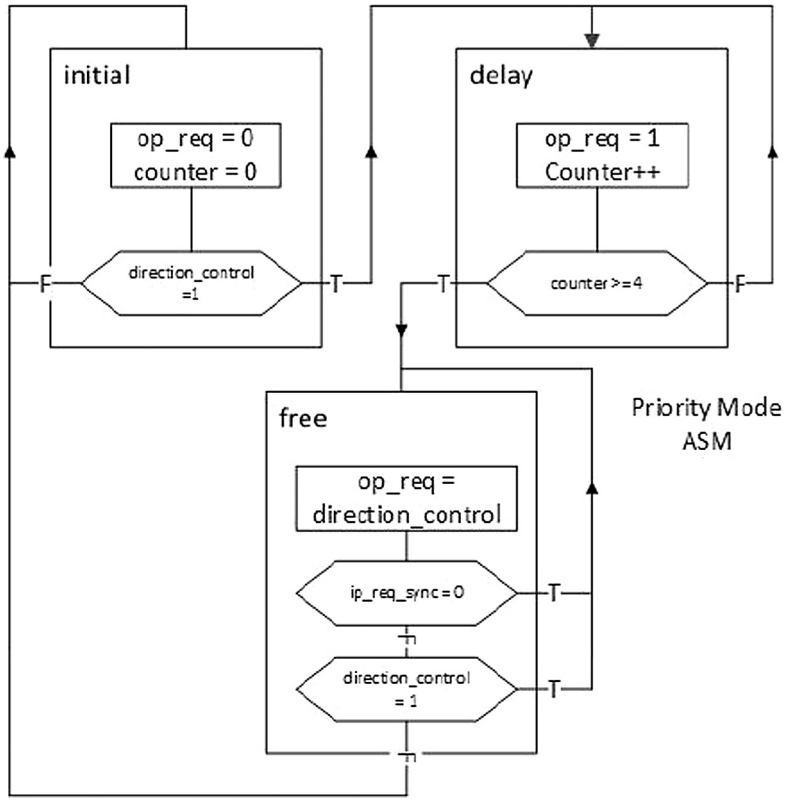

Fig. 12 Dynamic channel control ASM in priority mode.

Similarly in the case of *Binary-to-Gray*, an *n-bit* gray coded vector (G_{n-1}, G_{n-2}, … G_2, G_1, G_0) can be used to convert to *n-bit* binary coded vector (B_{n-1}, B_{n-2}, …, B_2, B_1, B_0) as shown in given Eq. (2), where $\oplus$ indicate the same *xor function*.

$$B_{n-1} = G_{n-1}, B_{n-2} = B_{n-1} \oplus G_{n-2}, B_{n-3} = B_{n-2} \oplus G_{n-3}, \ldots, B_1 = B_2 \oplus G_1, G_0 = B_1 \oplus G_0 \quad (2)$$

3.3. Ready/Full generation block

Fig. 8 shows the details about the left hand side of the main architecture of the multi-synchronous FIFO. On the *write* side, the FIFO indicates whether it is *full* or not. The sender should only send data when the FIFO is not full and asserting *in_valid* signal. From Fig. 6, when synchronized read pointer is equal to the write pointer except MSBs, the FIFO is full. The condition for the full flag generation in the write clock domain is

if$((wprt[n] == \sim rptr\_sync[n])$ ***and*** $(wptr[n-1] == rptr\_sync[n-1]))$ ***then***
$full = 1$ ***and*** $out\_ready = 0$
else
full = 0 and out_ready = 1

3.4. Valid/Empty generation block

Fig. 9 shows the details about the right hand side of the main architecture of the multi-synchronous FIFO. On the *read* side, the FIFO indicates whether it is *empty* or not, on the basis of *in_ready* signal, the receiver can consume all the data available within the FIFO memory. The condition for the empty flag generation in the read clock domain is

if $(rprt == wptr\_sync)$ ***then***
$empty = 1$ ***and*** $out\_valid = 0$
else
$empty = 0$ ***and*** $out\_valid = 1$

4. Dynamic channel control protocol

Our proposed dynamic channel control protocol is similar to channel direction control (CDC) protocol used in BiNOC [11], FSNoC [12], and double data rate protocol used in BiLink [13] except that all previously implemented protocols support only synchronous design whereas our proposed protocols support GALS infrastructure. Fig. 5 shows the inter-router arrangement of dynamic channel control where a pair of ASMs namely Talker ASM and Listener ASM is supporting two unrelated clock domains and communicated on the principle of handshaking protocol.

4.1. Inter-router transmission

The basic operation sequence of four-phase handshaking protocol [31] is illustrated in Fig. 10. Initially the Talker ASM activates the *op_req_T* signal and this signal is passed through a register for unwanted glitches and then passed through two

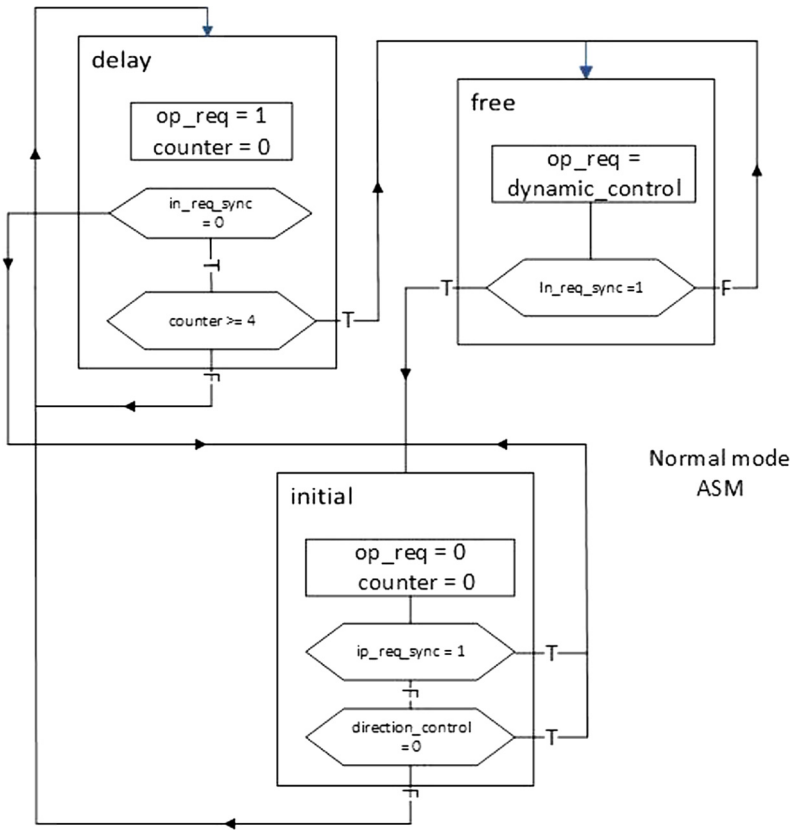

Fig. 13 Dynamic channel control ASM in normal mode.

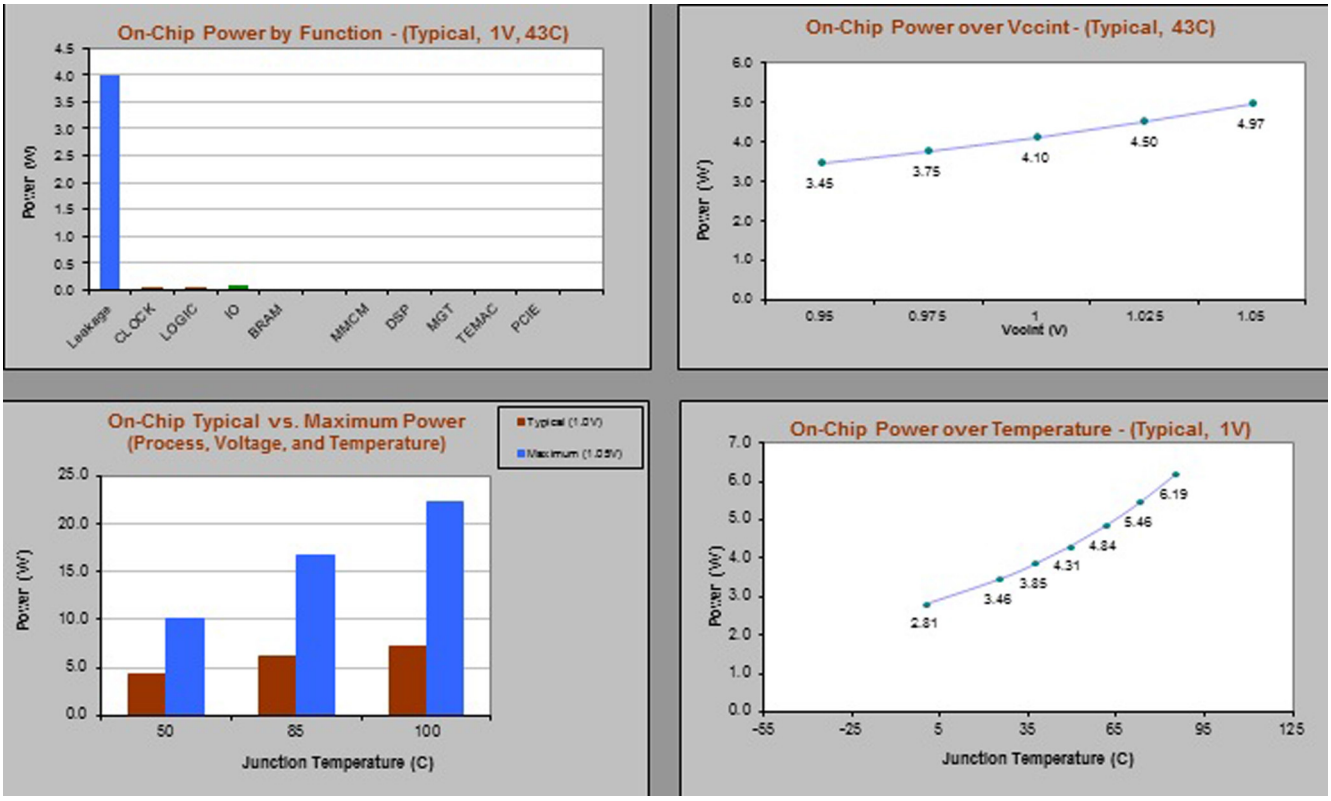

Fig. 14 Power analysis report of dynamic channel control module.

Table 2 Synthesis report of dynamic channel control module.

Device macro statistics	Timing parameter	Power dissipation
4×4 bit RAMs: 1 2-bit subtractors: 4 3-bit subtractor s: 4 1-bit registers: 2 2-bit registers: 4 3-bit registers:1 64-bit registers: 1 2-bit 2-to-1mux: 8	1.955 ns 0.442 ns logic, 1.513 ns route 22.6% logic, 77.4% route	4.782 W

Table 3 Synthesis report of multi-synchronous FIFO module.

Device macro statistics	Timing parameter	Power dissipation
16×16 bit RAM: 1 Dual-port block RAM: 1 2-bit adder: 12 2-bit subtractor: 4 Flip-Flops: 64 4-bit comparator equal: 12 1-bit 2-to-1 mux: 4 4-bit 2-to-1 mux: 13	2.126 ns 1.787 ns logic, 0.339 ns route 84% logic, 16% route	4.682 W

Table 4 Synthesis report input port controller as a whole module.

Device macro statistics	Timing parameter	Power dissipation
32×64 bit dual-port RAM: 2 4×4 bit single-port RAM: 1 8×5 bit single-port RAM: 8 8×8 bit single-port RAM: 8 2-bit adder: 8 3-bit subtractor: 4 3-bit adder: 12 3-bit subtractor: 36 1-bit register: 28 2-bit register: 9 20-bit register: 1 3-bit register: 1 4-bit register: 5 5-bit register: 13 8-bit register: 8 64-bit register: 1 5-bit comparator equal: 12 1-bit 2-to-1 mux: 20 2-bit 2-to-1 mux: 20 4-bit 2-to-1 mux: 14 5-bit 2-to-1 mux: 17 8-bit 2-to-1 mux: 16 1-bit xor2: 4	6.641 ns 1.533 ns logic, 5.108 ns route 23.1% logic, 76.9% route	7.782 W

Table 5 Synthesis report of virtual channel allocator using separable input-first allocation.

Device macro statistics	Timing parameter	Power dissipation
8×5 bit single-port RAM: 20 1-bit register: 100 3-bit register: 20 1-bit 2-to-1 mux: 100 2-bit 2-to-1 mux: 100 3-bit 2-to-1 mux: 20 5-bit 2-to-1 mux: 20	6.894 ns 0.555 ns logic, 6.339 ns route 8.1% logic, 91.9% route	6.182 W

Table 6 Synthesis report of virtual channel allocator using separable output-first allocation.

Device macro statistics	Timing parameter	Power dissipation
8×5 bit single-port RAM: 20 1-bit register: 160 2-bit 2-to-1 mux: 100 3-bit 2-to-1 mux: 20 5-bit 2-to-1 mux: 20	5.063 ns 0.494 ns logic, 4.569 ns route 9.8% logic, 90.2% route	7.73 W

Table 7 Synthesis report of virtual channel allocator using wavefront allocation.

Device macro statistics	Timing parameter	Power dissipation
16×10 bit single-port RAM: 20 4×4-bit multiplier: 2 8×4-bit multiplier: 2 4-bit register: 2 4-bit 2-to-1 mux: 2 1000-bit shifter logical left: 2	10.814 ns 1.131 ns logic, 9.683 ns route 10.5% logic, 89.5% route	9.49 W

flip-flop synchronizers. In the *L_request1* state, Listener ASM senses activation of *op_req_T*, and *ip_req_sycT* signals, and then it sends ack *op_req_L* signal toward Talker. In the *T_request1* Talker detects activation of *op_req_L* as *ip_req_synL*, and it sets the value *op_req_T* signal as zero.

Now listener deactivates the *op_req_L* signal after it notices deactivation of the *op_req_T* signal. Finally Talker comes back to the idle state once it detects deactivation of the *op_req_L* signal. Initially in *idle* state, the *op_req_T* signal is not

Please cite this article in press as: R. Kamal, J.M.M. Arostegui, A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.02.019

Table 8 Synthesis report of output port controller.

Device macro statistics	Timing parameter	Power dissipation
4×4 bit single-port RAM	2.844 ns	6.96 W
1	0.503 ns logic, 2.341 ns route	
4-bit adder: 4	17.7% logic, 82.3% route	
4-bit subtractor: 4		
1-bit register: 29		
10-bit register: 1		
2-bit register: 1		
3-bit register: 1		
4-bit register: 8		
5-bit register: 4		
64-bit register: 1		
1-bit 2-to-1 mux: 38		
4-bit 2-to-1 mux: 12		
5-bit 2-to-1 mux: 4		
4-bit xor2: 1		

activated and it moves to the *T_request1* state at the edge of *clk_T*, in this state, the *op_req_T* signal is set to be active. The finite state machine (FSM) then stays in the *T_request1* state until activation of Listener signal, *in_req_syncL*. Then it transfers to the *T_request0* state and *op_req_T* signal becomes zero. The FSM returns to the *idle* state after it senses deactivation of the *in_req_syncL* signal. The Talker FSM is similar to the Listener FSM except that it has one more *idle* state, and thus can only respond to the Talker FSM. The *op_req_T* and *in_req_syncL* signals must be glitch-free because both signals are synchronized by a different clock domain.

4.2. Intra-router transmission

In our proposed router architecture, each bidirectional channel request to be achieved wisely to confirm that each transmission is valid unalike transmission channel used in conventional routers. To make sure that on each bidirectional link, only one direction is effective at one time, and an inter-router communication channel control scheme is implemented for each channel in an algorithmic state machine as shown in Fig. 12. Within the Dynamic channel control module, algorithmic state machine dynamically reconfigures the route of each link. As shown in Fig. 12, our ASM design is quite simple but very efficient and it contains mainly three states: initial, free, and delay. Here communication between two routers from router1 to router2 (R1 → R2) and from router2 to router1 (R2 → R1) is shown in Fig. 12. Here we developed a mode controller, which can set the mode of operation of ASM as priority mode or normal mode as shown in Fig. 11.

4.2.1. Priority mode ASM

Dynamic channel control ASM in priority mode is originated at the *free* state as shown in Fig. 12. In this state if control signal *direction_control* = 1 or input signal *in_req_sync* = 0, it will continue in this state. In other words, the link path will remain outward path as long as there are data flits within the current router to be sent via this link. The channel path should still remain unaffected even when there are no data to transmit, and if there is no request to send data from the

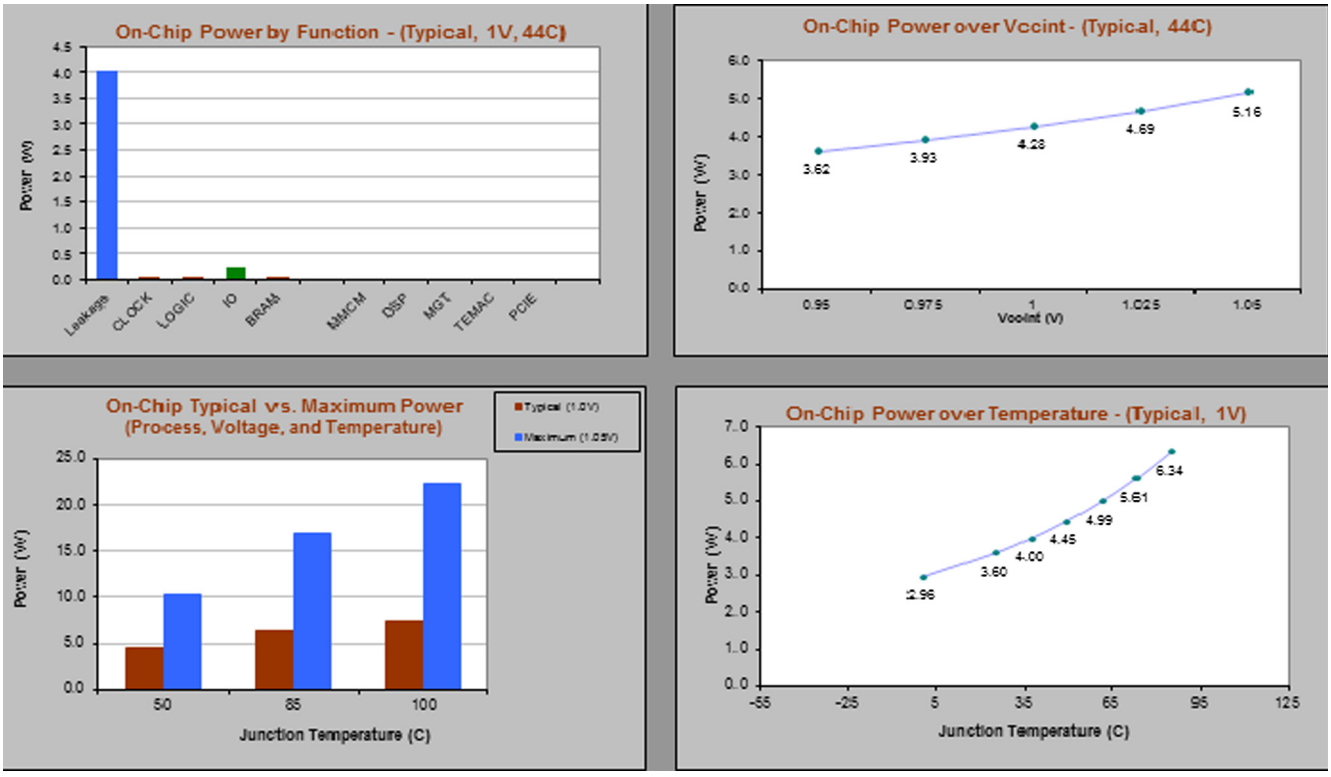

Fig. 15 Power analysis report of multi-synchronous FIFO module.

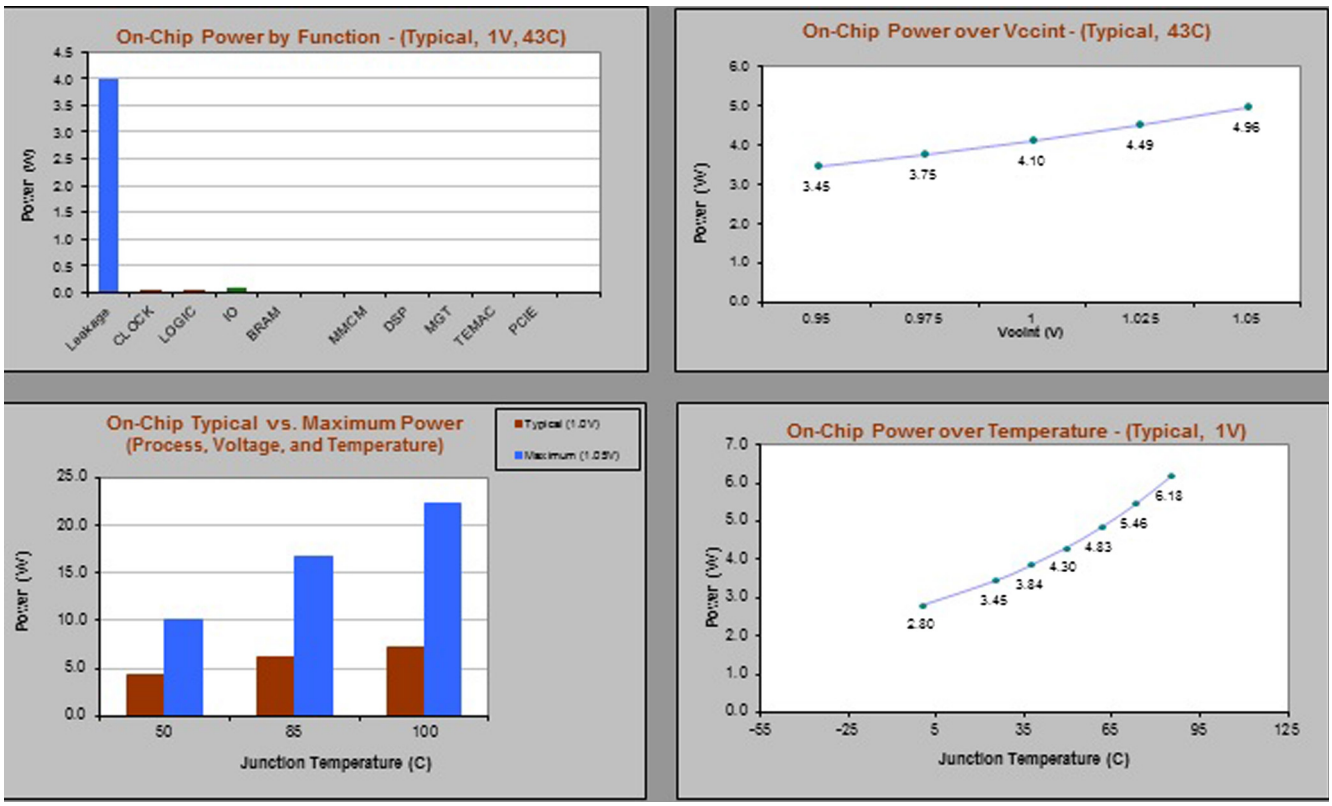

Fig. 16 Power analysis report of input port controller as a whole module.

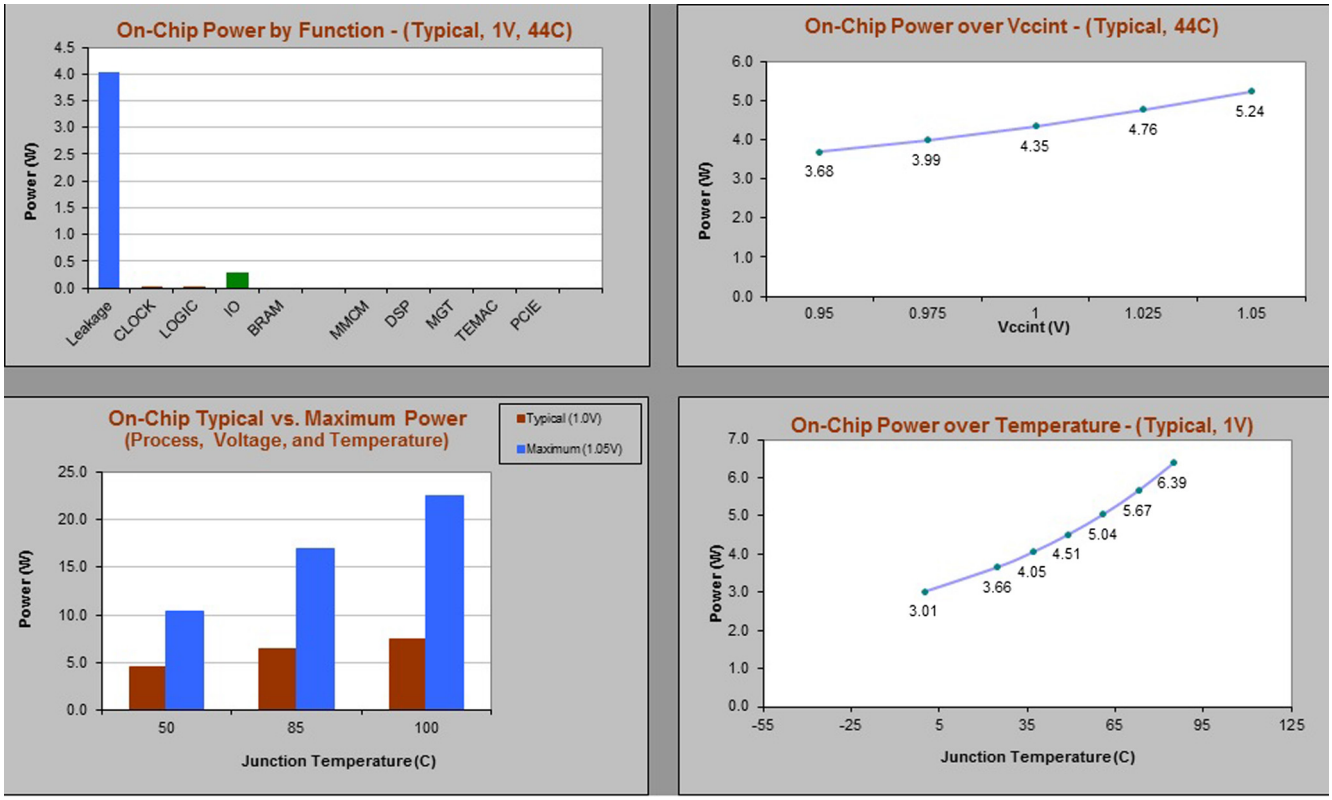

Fig. 17 Power report of virtual channel allocator using separable input-first allocation.

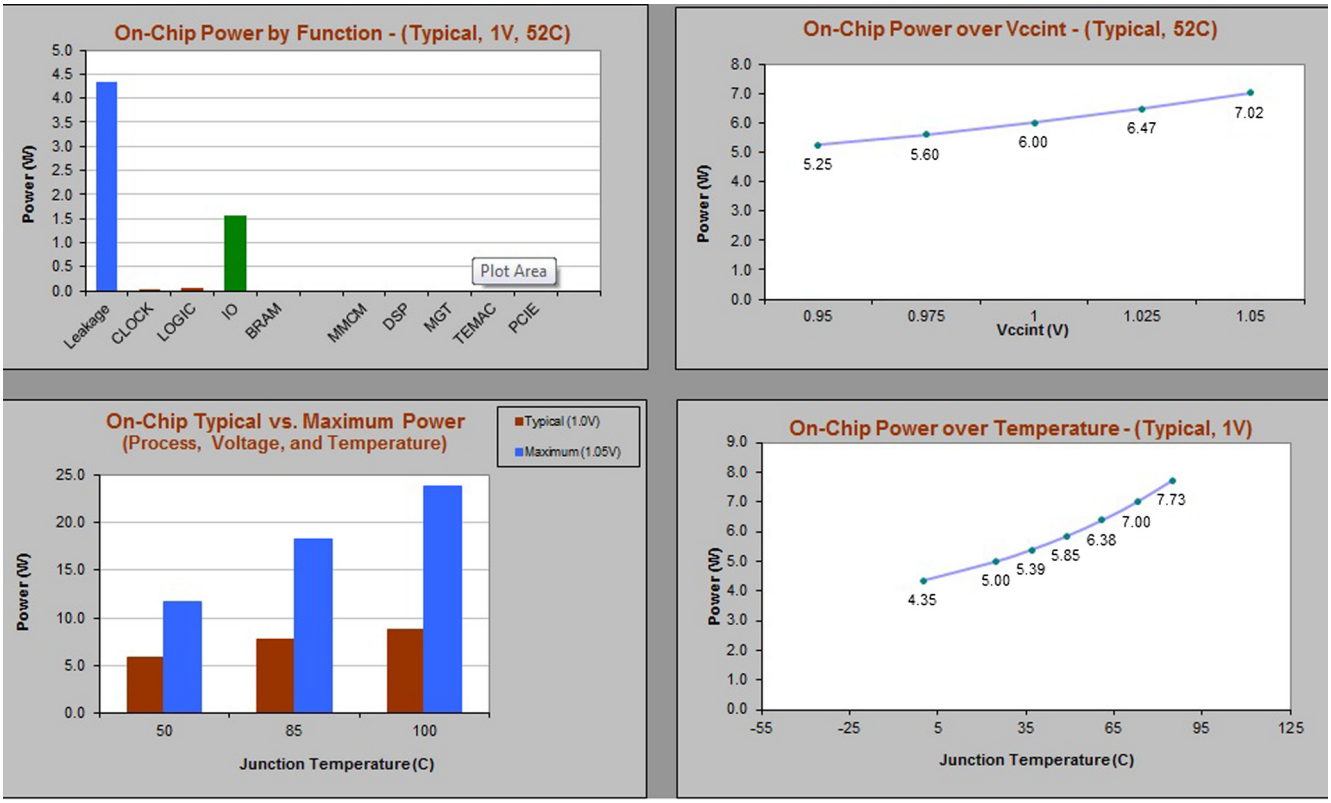

Fig. 18 Power report of virtual channel allocator using separable output-first allocation.

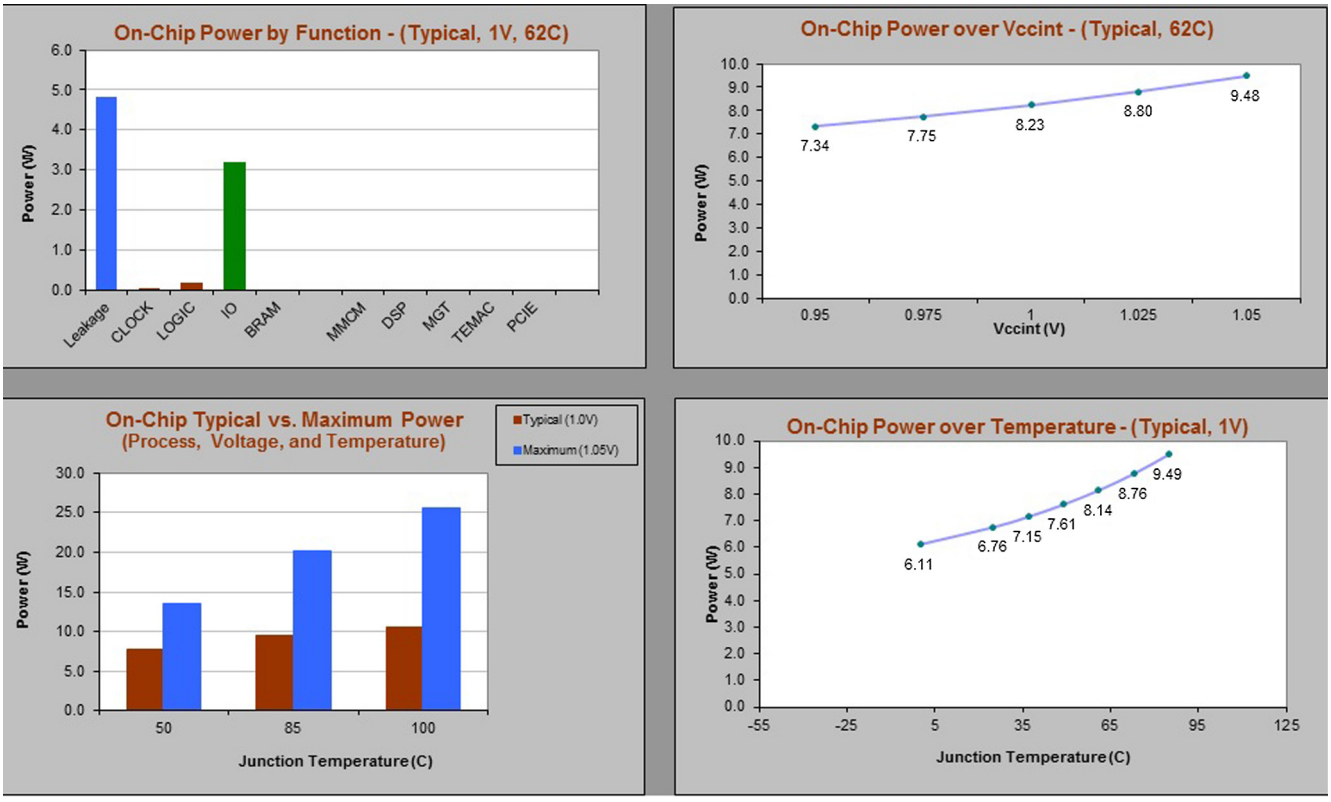

Fig. 19 Power report of virtual channel allocator using wavefront allocation.

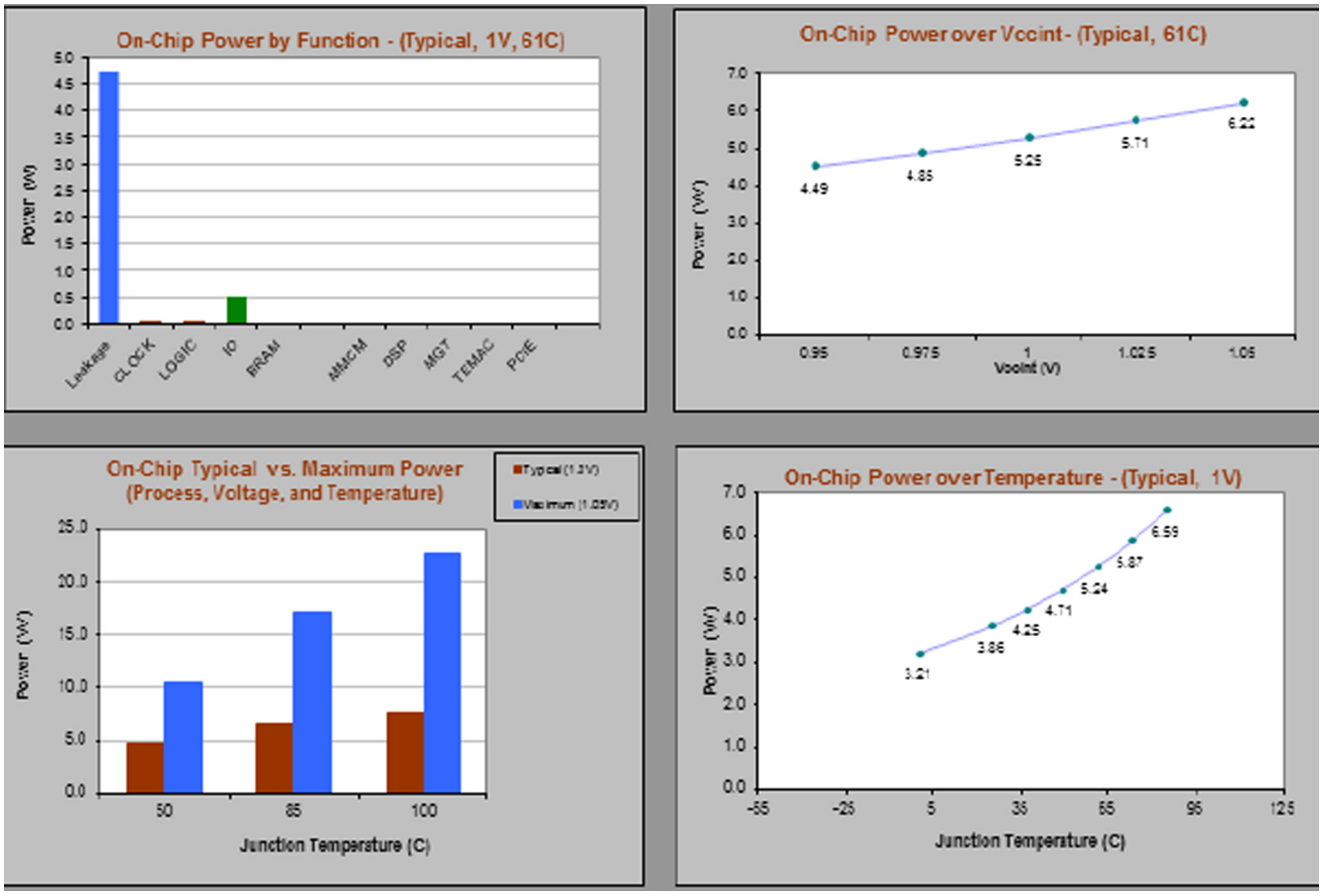

Fig. 20 Power report of output port controller.

neighboring router. When there are data to be sent from the neighboring router and if there are no data to transfer from the current router, on this condition the priority ASM will move from *free* state to *initial* state. The output signal was *op_reg* = *direction_control* in *free* state; subsequently, the output may get to be 0, permitting the neighboring router to ask for the channel. After entering in initial state, priority ASM will remain in this state as long as there are no data to transmit outward path, i.e. *direction_control* = 0. Meanwhile, in *initial* state, *op_req* = 0 and also ASM initializes the counter as *counter* = 0. Whenever the value of signal *direction_control* changes from 0 to 1 the priority ASM will move into *delay* state waiting to regain the channel control to transmit data. Within the *delay* state, the counter will increment its value during each clock cycle and *op_req* = 1. Whenever the value to *counter* = 4, the priority ASM returns to the *free* state and starts data transmission. Within *delay* state the output signal is *op_reg* = 1 and has to reach the neighboring router in such a manner that the normal ASM will capture the link by initializing the *initial* state. This the main reason for the *delay* state.

4.2.2. Normal mode ASM

Dynamic channel control ASM in normal mode is originated at the *initial* state with *op_req* = 0 as shown in Fig. 13. If a local route compute module requests to use the link (*direction_control* = 1) and the priority ASM of neighboring routers yields the channel (*ip_req_sync* = 0), normal ASM will move from the *initial* state to the *delay* state. For eight clock cycles, the normal ASM continues in the *delay* state and it will return to an *initial* state whenever priority ASM requests the channel as *in_req_sync* = 0. Whenever *in_erq_sync* = 0 and after eight consecutive clock cycles, the normal ASM will move into the *free* state. If priority ASM does not have any data to communicate, i.e. *in_req_sync* = 0, the normal ASM could continue in the *free* state. Normal ASM stops immediately data transmission after *in_req_sync* = 1 and returns back to the *initial* state.

5. Results

We have developed a parameterized RTL model using Verilog HDL and synthesized it in a commercial FPGA design flow. Synthesis is performed using Xilinx ISE 14.7 and FPGA Virtex 6 family device XC6VLX760 is considered as target technology. Our proposed Multi-synchronous bidirectional NoC architecture is divided into five sub-modules such as *input port controller*, *Virtual channel allocator*, *Switch allocator*, *Crossbar switch* and *output port controller*. Here we represent synthesis report of each individual sub-module which includes macro statistics, power report and delay information. Power analysis is performed by the Xpower analyzer and graphs are plotted by the Xpower estimator.

Please cite this article in press as: R. Kamal, J.M.M. Arostegui, A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.02.019

Input port controller mainly contains channel interface at receive side, Multi-synchronous FIFO as a flit-buffer, Route Compute, Flow control interface at the sender side, Error reporting module, and dynamic channel control module. Fig. 14 and Table 2 show the device macro statistics, timing parameter and power dissipation of Dynamic channel control module. Similarly we represent here synthesis reports and power analysis reports of some of the important modules belonging to Input port controller (see Tables 3 and 4 and Figs. 15 and 16).

In the virtual channel flow control, when a packet arrives at a router interface and before it can proceed further, its head flit must secure one of the virtual channels linked with the physical channel associated with its endpoint. Here we have implemented virtual channel controller in three ways as using separable input-first allocation, using separable output-first allocation and using wavefront allocation. A series of figures and tables show the synthesis reports as well as power analysis reports (see Tables 5–7 and Figs. 17–19).

Once a packet has completed Virtual Channel allocation, its flits can be forwarded to the selected destination port subject to buffer space availability. The switch allocator is responsible for scheduling, for each flit to be transferred, a crossbar connection between the corresponding input and output ports. As input port controller, output port controller contains some important modules, e.g. flow control interface at receive side, error reporting module and it's tracks state of the buffer in downstream buffer. Synthesis and power analysis report of output port controller as a whole is shown in Table 8 and Fig. 20 respectively.

6. Conclusions

In this paper, Dynamic reconfigurable multi-synchronous router architecture is proposed and implemented an RTL model using parameterized Verilog HDL. A novel multi-synchronous first-in first-out (FIFO) buffer is implemented at the input/output ports to resolve GALS issue. A new distributed dynamics channel controller is implemented to increase existing NoC proficiency using change of the path of the communication link at the runtime traffic situation. The Synthesis and power analysis report of each individual block shows the implemented multi-synchronous bidirectional NoC's router provides higher packet consumption rate, better bandwidth utilization with lower packet delivery latency and also fulfills the GALS issues and bandwidth requirements.

Acknowledgment

The authors are grateful to their respective university and tutors for their help and support.

References

[1] W. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2003.
[2] W. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, in: Proceeding of the 38th Design Automation Conference, 2001, pp. 684–689.
[3] L. Benini, G. De Micheli, Networks on chips: a new SoC paradigm, Computer 35 (1) (2002) 70–78.
[4] N.E. Jerger, L.S. Peh, On-Chip Networks, Morgan Kaufmann, 2009.
[5] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q.V. Le, Large scale distributed deep networks, Adv. Neural Inform. Process. Syst. 25 (2012) 1223–1231.
[6] Q.V. Le, Building high-level features using large scale unsupervised learning, in: IEEE conference Acoustics, Speech and Signal Processing (ICASSP), 2013, pp. 8595–8598.
[7] A. Agarwal, C. Iskander, R. Shankar, Survey of Network on chip (NoC) Architecture & Contribution, J. Engg. Comput. Architect. 3 (1) (2009) 21–27.
[8] T. Bjerregaard, S. Mahadevan, A survey of research and practices of network-on-chip, ACM Comput. Surv. 38 (1) (2006) 1–51.
[9] G. Dimitrakopoulos, A. Psarras, I. Seitanidis, Microarchitecture of Network-on-Chip Routers, A Designer's Perspective, Springer, 2015.
[10] E. Kasapaki, M. Schoeberl, R.B. Sørensen, C. Müller, K. Goossens, J. Sparsø, Argo: A real-time network-on-chip architecture with an efficient GALS implementation, IEEE Trans. Very Large Scale Integration Syst. 24 (2) (2016) 479–492.
[11] Y.C. Lan, H.A. Lin, S.H. Lo, Y.H. Hu, S.J. Chen, A Bidirectional NoC (BiNoC) architecture with dynamic self-reconfigurable channel, IEEE Trans. Computer-Aided Design Integrat. Circ. Syst. 30 (3) (2011) 427–440.
[12] Z. Qian, S.M. Abbas, C.Y. Tsui, FSNoC: a flit-level speedup scheme for network-on-chips using self-reconfigurable bidirectional channels, IEEE Trans. Very Large Scale Integr. VLSI Syst. 23 (9) (2015) 1854–1867.
[13] J. Zhu, Z. Qian, C.Y. Tsui, BiLink: A high performance NoC router architecture using bi-directional link with double data rate, Integrat. VLSI J. 55 (2016) 30–42.
[14] A. Chakraborty, M.R. Greenstreet, Efficient self-timed interfaces for crossing clock domains, in: Proc. ASYNC, 2003, pp. 78–88.
[15] J. Mekie, S. Chakraborty, D.K. Sharma, G. Venkataramani, P. S. Thiagarajan, Interface design for rationally clocked GALS systems, in: Proc. ASYNC, 2006, pp. 160–171.
[16] A. Sheibanyrad, A. Greiner, Two efficient synchronous ↔ asynchronous converters well-suited for networks-on-chip in GALS architectures, Integrat. VLSI J. 41 (2008) 17–26.
[17] E. Beigne, P. Vivet, Design of on-chip and off-chip interfaces for a GALS NoC architecture, in: Proc. ASYNC, 2006, pp. 172–183.
[18] A. Chattopadhyay, Z. Zilic, GALDS: A complete framework for designing multi clock ASICs and SoCs, IEEE Trans. VLSI Syst. 13 (6) (2005) 641–654.
[19] M.A.A. Faruque, T. Ebi, J. Henkel, Configurable links for runtime adaptive on-chip communication, in: IEEE Conf. Design, Automation & Test in Europe Conference & Exhibition, 2009, pp. 20–24.
[20] M.H. Cho, M. Lis, M. Kinsy, K.S. Shim, T. Wen, S. Devadas, Oblivious routing in on-chip bandwidth-adaptive networks, in: Proc. PACT, 2009, pp. 181–190.
[21] W.J. Dally, Virtual-channel flow control, IEEE Trans. Parallel Distribut. Syst. 3 (2) (1992) 194–205.
[22] W.J. Dally, J.W. Poulton, Digital Systems Engineering, Cambridge University Press, 1998.
[23] J. Ebergen, Squaring the FIFO in GasP, in: Proc., Asynchronous. Circuits and Systems, 2001, pp. 194–205.
[24] C.E. Molnar, I.W. Jones, W.S. Coates, J.K. Lexau, A FIFO ring performance experiment, in: Proc. Adv. Res. in. Asynchronous. Circuits and Systems, 2001, pp. 194–205.

[25] M.R. Greenstreet, Implementing a STARI chip, in: Proc. Int. conf. on VLSI in Computers and Processors, 1995, pp. 38–43.
[26] A. Chakraborty, M.R. Greenstreet, Efficient self-timed interfaces for crossing clock domains, in: Proc. Asynchronous Circuits and Systems, 2003, pp. 78–88.
[27] J.N. Siezovic, Pipeline synchronization, in: Proc. Advanced Research in Asynchronous Circuits and Systems, 1994, pp. 87–96.
[28] I.M. Panades, A. Greiner, Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in GALS architectures, in: First Int. Symposium on NoC, 2007, pp. 83–94.
[29] T. Chelcea, S.M. Nowick, A low-latency FIFO for mixed-clock systems, in: Proc. IEEE Computer Society Workshop on VLSI, 2000, pp. 119–126.
[30] C. Cummings, Simulation and synthesis techniques for asynchronous FIFO design, Synopsys Users Group, 2002.
[31] P.P. Chu, RTL Hardware Design Using VHDL, Coding for Efficiency, Portability, and Scalability, John Wiley Sons, 2006.

Please cite this article in press as: R. Kamal, J.M.M. Arostegui, A Multi-Synchronous Bi-Directional NoC (MBiNoC) architecture with dynamic self-reconfigurable channel for the GALS infrastructure, Alexandria Eng. J. (2017), http://dx.doi.org/10.1016/j.aej.2017.02.019