Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Technology Dependent Solutions for Buffering Policies of Network on Chip Targeting FPGA Platforms

[1]Liyaqat Nazir and [2] Roohie Naaz Mir

[1]Department of Computer Science and Engineering, NIT, Srinagar, India
E-mail: [1]liyaqat_02phd13@nitsri.net, [2]Naaz310@nitsri.net

ABSTRACT

The communication between processing elements is facing challenges due to power, area, and latency. Temporary storage of flits required during communication contributes to the major power consumption out of the total power consumed by the on-chip communication. The ideal NoC interconnect should match its performance cost to that with the network channels (buffers). The majority of current NoCs consume a high amount of power and area due to router buffers resources only. Removing buffers and virtual channels (VCs) significantly simplifies router design and reduces the power dissipation by a considerable amount but it can lead to the unpredictable delay for flit flow hence can substantially degrade the overall application performance. Therefore, the buffering scheme used in NoC based router plays a significant role in determining the performance of the NoC based mesh. Moreover, with rapid development in modern FPGAs from prototype designing to low and medium volume productions, it becomes imperative to consider architectural optimizations that are specific to FPGA fabric only. In this paper, we for the first time, with the use of technology dependent mapping strategies, we attempt to provide an optimized realization of a FIFO buffer designed that will help designers to adopt the efficient design of NoC microarchitecture routers. As no such work has been reported in the past we, therefore, compare our work with technology independent optimization reported. The prime contribution of this article is that the proposed realization will helps in elimination of the presence of fixed inherent FIFO buffer instantiations as the proposed realization gives us an idea to explore underlying FPGA fabric more efficiently for the realization of the FIFO than existing. The implementation targets Virtex-5, Virtex-6 and Virtex-7 FPGA device families from Xilinx.

Keywords: *FPGA; Network-on-chip; Buffers; LUT; Reconfigurable computing*.

1. INTRODUCTION

In the past few years, with the concept of Network-on-Chip communication architecture, NoC has attracted a lot of attention by providing higher bandwidth and higher performance architectures for communication on the chip [1]. NoC can provide simple and scalable architectures if implemented on reconfigurable platforms [2]. Network on chip offers a new communication paradigm for system on chip (SoC) design [3]. Many processing elements of SoC are connected through Network-on-chip (NoC) routers which are arranged in some regular fashion such as Mesh, linear, torus, 2D, 3D type of topologies. To achieve high performance, the router should provide high bandwidth and low latency [4]. Although the performance of the NoC is normally seen by its throughput, which is defined by the network topology, router throughput and the traffic load on the network [5]. Therefore, the routers for a NoC must be designed to meet latency and throughput requirements amidst tight area and power constraints; this is a primary challenge designer are facing as many-core systems scale [6]. As router complexity increases with bandwidth demands, very simple routers (non-pipelined, wormhole, no VCs, limited buffering) can be built when high throughput is not needed, so require low area and power overhead [7], [8]. Challenges arise when the latency and throughput demands on on-chip networks become increasingly high [9]. A router's architecture determines its critical path delay which affects per-hop delay and overall network latency [10], [11]. Router microarchitecture also impacts network energy as it determines the circuit components in a router and their activity. The implementation of the routing, flow control and the actual router pipeline will affect the efficiency at which buffers and links are used and thus overall network throughput [1]. The area footprint of the router is clearly determined by the chosen router microarchitecture and underlying circuits. The critical path of the data path units in the router and the efficiency of control path units determine the router throughput [12], [13], [14], [15]. The communication between various processing elements through NoC routers require various control signal for efficient flit traversal in the communication fabric as illustrated in Fig 1 [16]. Allocators are used to allocate virtual channels (VC) and to perform matching between groups of resources on each cycle [17], [18], [19] [20], [21]. Upon the flit arrival at the input port, contention for access to the fabric with cells at both input and output occurs. The router units exchange necessary handshake signals for data/flit transfer [7], [22]. A VC allocator thus performs allocation between the input flits and allows at most one flit contending at the input port to be destined to the selected output port [23]. In order to reduce the line of blocking, the rest of the contending flits are buffered into the virtual channels or buffers of the router so as to service them in coming appropriate clock cycles [24]. Buffers have simple logic and functionality as compared to the control logic, but in networks they consume most of the area resources [25]. However, the smaller the buffers are, the bigger is the possibility that some traffic is lost during data flit transfer. As the buffering demands storage capacity i.e registers or memory, it rapidly increases area costs. Hence, the right sizing of the buffers is very important. For successful buffer design, as exact traffic characteristics as possible are also needed [26]. However, elimination of input buffers eliminates the need of virtual channels (VCs) besides causing the reduction in area and power [27]. This increases the

Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

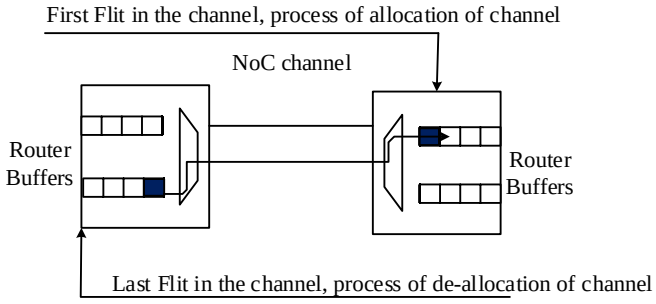


Fig 1. Block diagram of NoC router communication.

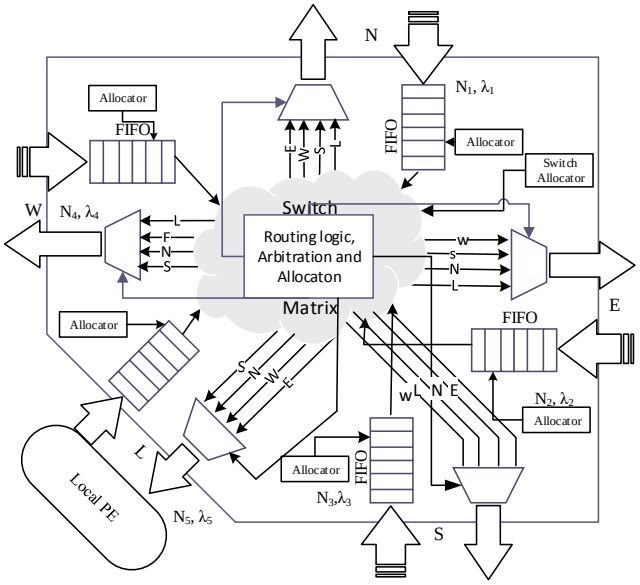


Fig 2. Block diagram of NoC router .

chances for head-of-line blocking and causes reduction of performance in a network on chip based systems. On the other hand, NoC router architecture generally needs large amount of FPGA resources [28], [29] which is the barrier to widespread adoption of NoC routers on FPGA platforms. Moreover, the limited number of in build block buffer instantiations available with a given platform increases the barrier to next higher level [30], [31]. Traditional implementations of FIFO buffering policy have been platform independence oriented, where the design process consists of developing the necessary high level code for application level with some thought given to the underlying architecture to optimize the code quality [32], [33]. However, the functional diversity and complexity can be exploited to reveal hidden parallelism helping us to formally capture concurrencies both within control logic models of computation and among multiple control logic models of general logic design [34], [35]. The high-level concurrent tasks can be then mapped to the underlying communication and computation resources [36]. This has provided designers with sufficient impetus to look for platform oriented solutions where the underlying hardware can be utilized to develop a block level solution that best matches the functional diversity and complexity in buffering policies by developing the right level of parallelism. Accordingly, attempts have been made to develop custom and reconfigurable architectures for realizing

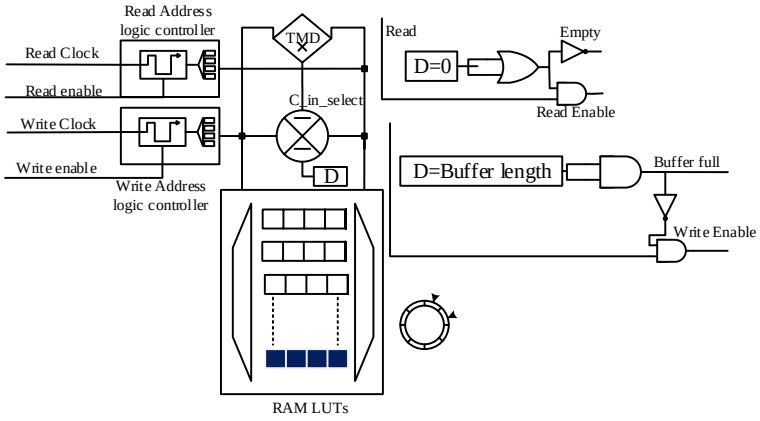


Fig 3. Block diagram of proposed circular buffer.

various buffering policies in application specific integrated circuits (ASIC) and field programmable gate arrays (FPGA) [37], [38], [39], [40], [41], [42], [43], [44].

In this paper, we, therefore, propose an efficient FPGA based realization of Circular buffer that will aid in the efficient implementation of NoC router microarchitectures on LUT based reconfigurable platforms. we have adopted technology dependent optimizations based approach in this paper. The approach is implemented successfully on Xilinx's Virtex-5, Virtex-6 and Virtex-7 FPGA devices. As the performance speed-up achieved using technology dependent approach is a strong function of the nature of the target FPGA family. The optimizations presented in this work are targeted for FPGAs with 6-input LUTs. Therefore, for comparisons, we have considered only those implementations that use FPGAs with 6-input LUTs. From experimentation; it is observed that FPGA-based implementation with the technology dependent approach results not only the consumption of lesser amount of resources in designing the Buffering network but also gives the possibility of realizing more number of FIFO buffers efficiently thus overcoming the barrier of having limited number of inherent FIFO buffers or block RAMs of FPGA device. the new realization will help the NoC design community to explore of having NoC based systems with larger mesh order with better efficiency in terms of the specific application.

The rest of the paper is organized as follows. Section II discusses the related work. Section III discusses general FIFO architecture. Section IV discusses the FIFO realization proposed in this paper. Section V discusses the preliminary terminology and the architectural details. Section VI discusses the technology dependent optimization of the multiply-adder unit. Synthesis, implementation and discussions are carried out in section VII. Conclusions are drawn in section VIII and references are listed at the end.

2. RELATED WORK

Increased advances in the NoC based communication paradigm have attracted a lot of attention from industry and academia. Being a newer field, developing a newer a design methodology for NoC based communication presents novel and exciting challenges for the EDA community. With the large requirement of hardware resources some works had been reported in the ASIC domain, but to transfer the idea efficiently

Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

and entirely on reconfigurable platforms is yet a milestone to be achieved. NoC advances on reconfigurable platforms are limited by the availability of the limited amount of logic resources and memory on FPGAs [38]. Reconfigurable platforms fail to provide the amount of logic needed for the implementation of an efficient NoC system. Buffers are critical components of a NoC router and channel buffers at each router in the NoC have a serious impact on the overall area [45]. In NoC based architectures buffering policies play a key role in determining the throughput, latency, area utilization and energy consumption. In order to reduce the implementation overhead in NoC, Efforts are required to minimize the overall use of buffering resources. Hence, a considerable research effort has been devoted to buffering policies that can be adopted in a NoC router microarchitecture in the last few years. While these policies focus mainly on energy efficiency and latency, but they also increase the complexity of the router. Throughout, the key parameter of the NoC router needs to be maintained while reducing the complexity of router design. Sophisticated input-buffered routers have been proposed for extending throughput, latency and clock speed. For instance, high-speed design of a FIFO has been proposed for extending steady data transmission between asynchronous clock domains in work [44]. The authors have exploited the instantiation of complete inbuilt RAM block available in the Xilinx based reconfigurable platforms. This has certainly provided an efficient FIFO buffering architecture as the in-build cores or instantiations are highly efficient. However, because of limited blocks available, they are unable to suffice the demands of a NoC based number. The authors in work [43] have presented a novel idea of realizing a FIFO buffer by presenting a custom cell-based design. The proposed design is aimed to provide a reliable flow of flits with reduced the latency and channel blocking overheads in a network on chip based system. The design is better than earlier reported, but the authors of the work have not given thought to the technology dependent optimizations in the work, as a result, the work inefficiently consumes large FPGA resources available with the reduction in the performance also. A similar work has been reported in [46]. The authors present a design method of asynchronous FIFO memory that primarily aims at buffer's capacity to prevent spillovers despite the fullness of data. The work is inefficient no thought is being given to the underlying architecture of the FPGA platform. Some other articles that report the work mainly aimed at throughput and latency optimization of router architecture, indirectly by buffer implementation, but logic resource utilization had not been considered as a performance parameter include the work in [47], the authors proposes a flit-reservation flow control, which sends control flits ahead of data flits, and timestamps these control flits so that buffers can be allocated just-in-time when data flits arrive. However, this still relies on input buffers. The improvement of the congestion of incoming packets can be also checked by the virtual channel (VC) scheme as presented in work [48], [49]. Virtual channel scheme multiplexes a physical channel using virtual channels (VCs), leading to the reduction in latency and increase in network throughput. The insertion of VCs also enables to implement policies for allocating the physical channel bandwidth, which enables support for quality of service (QoS) in applications [50]. All the above-mentioned approaches use technology independent optimizations to enhance the performance of the Network on chip router. In this paper, we take an alternate approach and propose realizations that are based on technology dependent optimizations. As already mentioned the performance speedup achieved using technology dependent approach is a strong function of the nature of the target FPGA family. The optimizations presented in this work are targeted for FPGAs with 6-input LUTs. Therefore, for comparisons, we have considered only those implementations that use FPGAs with 6-input LUTs.

3. FIFO ARCHITECTURE

An abstract FIFO provides a push and a pop interface and informs its connecting modules when it is full or empty. A push (write) is done when valid data are present at the input of the FIFO with FULL: signal low. At the read side, a pop (read) occurs when the upstream channel is ready to receive new data for the buffer with low empty signal, i.e., it has valid data to send. There are two types of FIFO designs and architectural schemes: serial and parallel [51], [52], [53], [54], [55]. The serial FIFO scheme such as shift registers the primitive FIFO generation that works by fall-through principle (or pipeline). However, with the advancement of architecture and circuit styling techniques the architectures of conventional FIFOs are constantly being improved. Currently, most of the FIFOs used are of parallel type, which are faster than serial FIFO [56]. This type of buffering scheme finds wide application in network on chip due to its relation to the fall through concept where the new arrival flit is stored (pushed) at the tail location of FIFO, and with each shift request, flits are shifted one location (slot) toward the head of queue. The process of pushing data into the asynchronous FIFO is done by continuously monitoring full and empty control signals from the FIFO buffer by the sender. The sender sets the request signal (push_req signal) after the data to he sent are ready. That data flits are on control basis continuously pushed into the consecutive buffer locations. The process of popping data from the asynchronous FIFO is equal to pushing process except that the data is supplied by the FIFO and obtained by the receiver. The control logic block contains control logic needed to control push pop operations on the actual memory block.

4. PROPOSED FIFO REALIZATION

We propose an efficient realization of FIFO buffer that will help in efficient implementation of NoC router microarchitectures. In our design FIFO designed as a circular array of identical cells RAM LUTs from SLICEM present in the FPGA fabric. The block level illustration of the proposed circular buffer is shown in Fig3. It mainly comprises of a pair of separate addressable controllers, each for writing (push) and pop operations. A separate full detector and an empty detector logic block, and control logic for the put operation and get operation. The full and empty detectors are required to observe the state of the FIFO and determine whether the FIFO is full or empty. The input and output behavior of the FIFO is controlled by the flow of two tokens, generated by a write address logic controller logic and a read address logic controller respectively: a put token is used to enqueue data items and a get token is used to dequeue data items. Once a data item is enqueued, it is moved only when it is dequeued. If the signal to put token generator is asserted, the FIFO enqueues one data item and rotates the put

Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

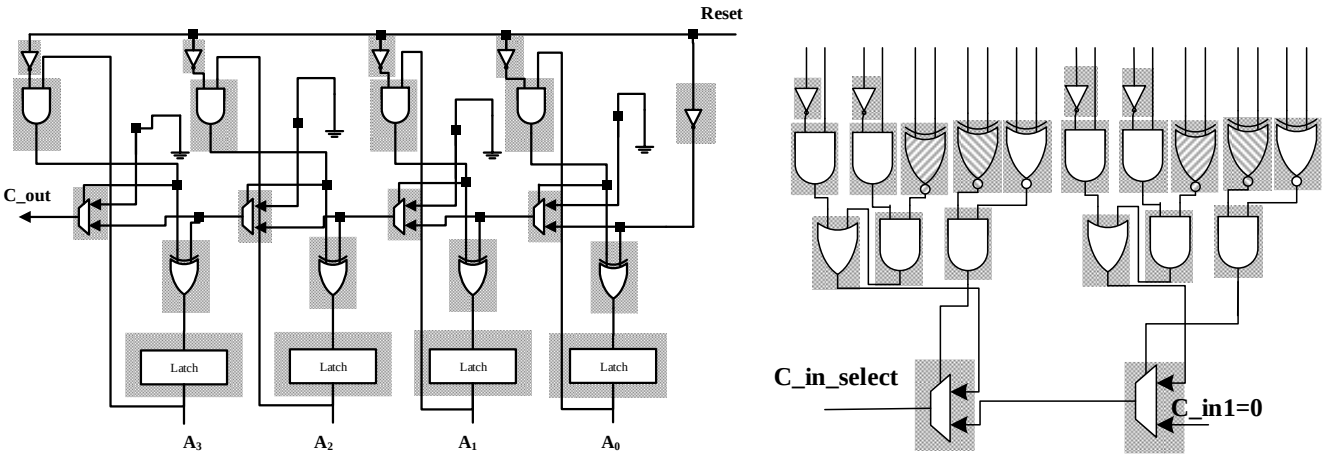


Fig 4. Logic illustration of (a) ALC (b) TMD.

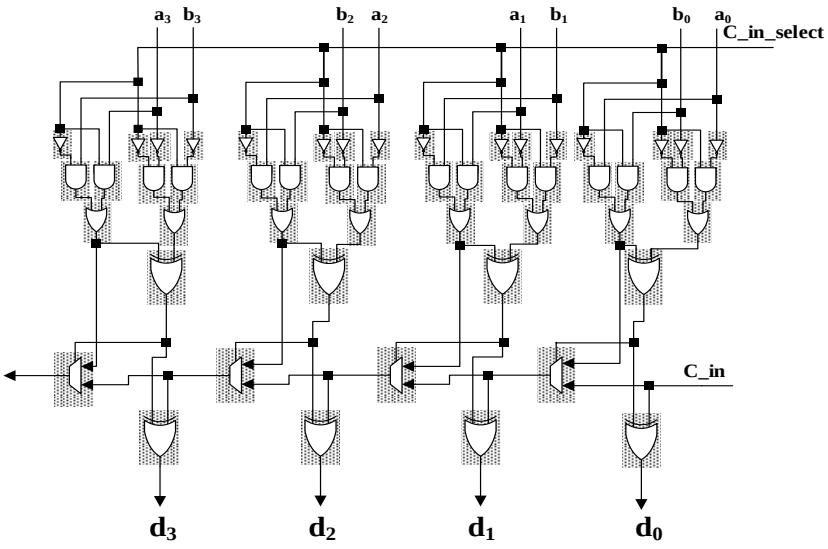


Fig 5. Logic illustration of TDT block.

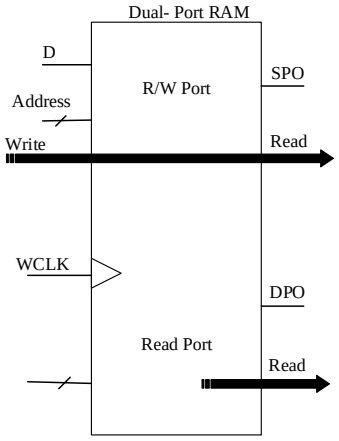


Fig 6. Block level illustration of DRB

token to the left. If it is deasserted, the put token is stalled with no enqueue operation in the FIFO. Similarly, the get controller enables and disables the get operations. Tokens move counter clockwise through the array of LUT based RAM cells. The LUT RAM cell having the corresponding put token (tail of the queue) has permission to store the enqueued data item, and the cell having the corresponding get token (head of the queue) has the permission to dequeue its data to the neighboring connecting node. The read address logic controller and the write address logic controller logic is designed in such a way that the get token is never ahead of the put token. After the token has been consumed by the LUT-based RAM cell, it will be passed to its left neighbor at the beginning of the next clock cycle, after the respective operation is completed. The movement of tokens across the LUT RAM cells is controlled both by interface requests as well as the state of the FIFO (full or empty), which are combined into the global signals Write and Read.

5. PRELIMINARY TERMINOLOGY AND ARCHITECTURAL DETAILS

Logic synthesis FIFO buffer is concerned with realizing a desired functionality with the minimum possible cost. In the context of digital design of a buffering policy, the cost of a circuit is a measure of its speed, area, power or any combination of these. The block level illustration of the proposed realization of the FIFO shown in the Fig 3. The primary blocks required to design the FIFO are the address logic controllers (ALC), token distance tracking (TDT), token magnitude detection (TMD), control signal generation blocks and a distributed RAM block (DRB). Distributed RAM is crucial to many high-performance applications that require relatively small embedded RAM blocks, such as FIFOs or small register files. The address logic controllers are realized with the help of digital synchronous counter logic network. So two separate n-bit address logic controllers are required for separate write and read operations of the FIFO into 2n RAM block location of each LUT based RAM block. The separate use of address logic controller block is required for the separate address generation in respective ports. As we are targeting a ring FIFO buffer therefore a synchronous counter is required for the desired operation. The logic level diagram of an address logic controllers realized with help of fast carry4 chain present in the FPGA target device is shown in Fig4(a). The logic controller shown is capable of providing an address realization of FIFO with a depth order of

Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

16 (24=16) with address bitsA0, A1, A2,A3. These address bits are used for physical address realization of the RAM blocks and are used by the token distance tracking block. The token distance tracking (TDT) block is realized with the help of a ripple carry subtraction block illustrated in Fig 5. The TDT for the FIFO is also realized with the help of fast carry4 chain logic present in the target reconfigurable platform. The TDT block takes inputs from token magnitude detection (TMD) block as illustrated in Figure 3. The logic network of a TMD is shown in Fig 4(b). TMD calculates the absolute distance between the tokens generated by ALC by providing a signal C_in_select input to the TDT block. TDT logic provides output to simple logic networks needed for both read and write ports and are called as the signal generation blocks. The signal generation blocks upon suitable receiving suitable inputs from TDT block generate Empty and Full signals that are needed for synchronization of communication ports during the buffering of data into the actual storage cells or distributive RAM block. The distributive RAM block has been realized as 16x1 dual-port RAM16X1D primitive instantiation requiring two 16x1 LUT RAMs present within a single SLICEM slice of the underlying fabric, as illustrated in Fig 6. Data is provided simultaneously to both LUT RAMs and controlled by address A[3:0], WE, and WCLK. The dual port RAM (DPR) has two access ports D and DPO as illustrated in Fig 6. For a general depth of n-bit FIFO realization, each 16 x 1-bit RAM is cascaded for n-occurrences for deeper and/or wider memory applications in the form of an array of memory to store the data, with a minimal timing penalty incurred through specialized logic resources. Distributed RAM writes synchronously and reads asynchronously by two separate sets of control signal, address and data busses. However, if required by the application, use the register associated with each LUT to implement a synchronous read function. For dual-port RAM16X1D, the first LUT out of two is required for the implementation of the A[3:0] port, i.e. the write and read address, and the second LUT is required to implement an independent read-only address i.e. DPRA[3:0] port. The port A address buss is an address bus takes its address values from write ALC, data bus output from the memory is DPO. Port D is the actual data buss that provides data to be stored in data memory. The control signal blocks act as an arbitration circuit used to determine which port has the right to write the memory, when to read and when ports are trying to update the data in the same address at the same time. Such kind of RAM realization is supported by various target devices such as Spartan-3 Virtex, Virtex-E, Spartan-II, Spartan-IIE, Virtex-II, and Virtex-II Pro FPGAs.

6. TECHNOLOGY DEPENDENT OPTIMIZATIONS

Technology dependent optimizations are used to transform the initial Boolean network into a circuit netlist, efficiently compatible with the target logic elements. The transformation is carried out optimally in accordance with the logic distribution among the targeted elements so ensure minimum possible LUT depth and minimum resource utilization of the target device. The target element in the majority of FPGAs is k-input LUT [56], [57]. It is a block RAM function generator that can implement any Boolean function of k variables by directly storing its truth table. State-of-art FPGAs support 6-input, dual output LUTs with the capacity of implementing a single 6-input Boolean function or two 5-input Boolean functions that share inputs [58], [59], [60]. An efficient utilization of this circuit element could lead to implementation of higher logic densities resulting in a reduced fan-out of the logic nets and thus a minimal-depth circuit.

Technology dependent optimization using LUTs is carried out in two steps. Firstly, the entire digital network is partitioned into suitable sub-networks or blocks. Individual nodes within each sub-block are then covered with suitable cones that maps a local Boolean function or a local truth table onto a separate LUT. Secondly a reverse process of the above step is carried, i.e. the entire network is then reconstructed by assembling the individually optimized sub-networks. Since the circular buffer is an assembly of ALC, TDT, TMD and DRB. An optimized realization of these individual sub networks could be adopted to realize an optimized realization of a circular buffering policy.

A. Technology dependent optimization of ALC and TMD

Figure 4 shows the Boolean network realization of ALC block and TMD block respectively. The network is traversed beginning at the primary inputs and proceeding toward the primary outputs. At each node in the network a best circuit is constructed that implements the sub-network extending from the node to the primary inputs. Next, we try to find an optimal covering for the nodes within each sub-network. A straight forward approach would be to cover each node with a separate cone and then map the local function implemented by each cone onto a separate LUT as shown in figure 4. The overall depth at network output is therefore, five respectively in each network. The LUT count is twenty-one and twenty respectively, the shaded blocks in the figure represents the LUTs consumed. Since we are targeting 6-input LUTs the implementation in figure 4 leads to severe under-utilization of the available resources in the considered network graphs. The number of required LUTs for realization and the overall depth may be further reduced with the help of tree minimization in the sub-networks. A further saving in resources is possible by exploiting the reconvergent PI nodes in the carry sub-network. A node in the network with a fan-out greater than one that terminates at other nodes within the same network is a source of reconvergent path. Reconvergent paths can be realized within the LUT and the total number of inputs is reduced. This is shown in the circuit of Fig 7 and Fig 8. The circuit, shown in Fig 7 is an optimized realization of ALC and TMD using 6 input LUTs. The depth of the circuit is now reduced to one and the total LUT count is also reduced to three in the optimized realization of ALC and the LUT depth count in realizing TMD has been reduced to one and LUT utilization is reduced to two. In order to ensure that the optimization done prior to the design entry should not get over-ridden during the mapping and PAR phases. We have re-defined the coding strategy at the design entry phase. Instead of writing conventional inferential codes, we adopt an instantiation based coding strategy, wherein a target element is directly called and the desired functionality is assigned to it. This ensures a controlled mapping.

The following instantiations were used to map TMD circuit illustrated in Fig 7.b.

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X


```
equalblock2:LUT6_2 generic map ( INIT => X"9009000022b20000") port map ( AlessB(1),AeqB(1),bin(3),ain(3),bin(2),
ain(2),'1','1');
equalblock1:LUT6_2 generic map ( INIT => X"9009000022b20000") port map ( AlessB(0),AeqB(0),bin(1),ain(1),bin(0),
ain(0),'1','1');
CARRY4_inst : CARRY4 port map (CO => cout1, O => dif1, CI => '0',CYINIT => '0', DI => AlessB, S => AeqB );
```

```
The following instantiations were used to map the ALC network.
LUT2_L_inst0 : LUT2_L  generic map (INIT => X"2") port map (Sinrd(0), q1rd(0), sr(0));
LUT2_L_inst1 : LUT2_L  generic map (INIT => X"2") port map (Sinrd(1), q1rd(1), sr(1));
LUT2_L_inst2 : LUT2_L  generic map (INIT => X"2") port map (Sinrd(2), q1rd(2), sr(2));
LUT2_L_inst3 : LUT2_L  generic map (INIT => X"2") port map (Sinrd(3), q1rd(3), sr(3));
CARRY4_inst_read : CARRY4   port map (COrd,Ord,'0','1',DIrd,Sinrd);
FDSE_inst0 : FDRE generic map (INIT => '0') port map (Q => q1rd(0),C => clk,CE => Rd_ce,R => S,D => Ord(0));
FDSE_inst1 : FDRE generic map (INIT => '0') port map (Q => q1rd(1),C => clk,CE => Rd_ce,R => s,D => Ord(1));
FDSE_inst2 : FDRE generic map (INIT => '0') port map (Q => q1rd(2),C => clk,CE => Rd_ce,R => S,D => Ord(2));
FDSE_inst3 : FDRE generic map (INIT => '0') port map (Q => q1rd(3),C => clk,CE => Rd_ce,R => S,D => Ord(3));

The following instantiations were used to map the TMD network.
equalblock2:LUT6_2 generic map ( INIT => X"9009000022b20000") port map ( AlessB(1),AeqB(1),bin(3),ain(3),bin(2),
ain(2),'1','1');
equalblock1:LUT6_2    generic map ( INIT => X"9009000022b20000") port map ( AlessB(0),AeqB(0),bin(1),ain(1),
bin(0),ain(0),'1','1');
CARRY4_inst : CARRY4 port map (CO => cout1, O => dif1, CI => '0',CYINIT => '0', DI => AlessB, S => AeqB );
```

```
The following instantiations were used to map the TDT block.
LUT6_2_inst0 : LUT6_2    generic map ( INIT => X"ac00000099000000") port map ( p(0),g(0),bin(0),ain(0),cout1(1),
'1','1','1');
LUT6_2_inst1 : LUT6_2    generic map ( INIT => X"ac00000099000000") port map ( p(1),g(1),bin(1),ain(1),cout1(1),
'1','1','1');
LUT6_2_inst2 : LUT6_2    generic map ( INIT => X"ac00000099000000") port map ( p(2),g(2),bin(2),ain(2),cout1(1),
'1','1','1');
LUT6_2_inst3 : LUT6_2    generic map ( INIT => X"ac00000099000000") port map ( p(3),g(3),bin(3),ain(3),cout1(1),
'1','1','1');
CARRY4_inst_absolute_diffrence_circuit : CARRY4 port map (CO => cout2, O => diffrence, CI => '1',CYINIT => '1', DI
=> g, S => p );
```

The Boolean network now has an LUT count of only three and a depth of only one LUT. The complete efficient realization is shown in the Fig 7. The Network is realized with help of three 6-input dual output LUTs and two six input LUT with a total LUT depth of one respectively for ALC and TMD network.

FPGAs have a well-defined design flow that starts with design entry and proceeds through phases like synthesis, translation, mapping and place and route (PAR).It was mentioned in the introductory section that the design cycle in FPGAs is simple due to the availability of the computer aided design (CAD) tools that handle the majority of the technology dependent steps like mapping and PAR. Technology dependent optimizations mainly focus on improving the mapping of Boolean networks onto target LUTs. However, with modern CAD tools, both technology mapping and PAR are automated and the optimization process is not transparent to the user [64]. Thus any optimization done prior to the design entry may get over-ridden during the mapping and PAR phases. To counter this issue we re-define the coding strategy at the design entry phase. Instead of writing conventional inferential codes, we adopt an instantiation based coding strategy, wherein a target element is directly called and the desired functionality is assigned to it. This ensures a controlled mapping. The following instantiations were used to map the various networks in the proposed FIFO buffer.

B. Technology dependent optimization of RAM block

In every topology of a NoC based communication network, there is an exchange of data flits between various IPs at a very rapid rate. Intermediate storage or buffering is always required when data arrive at routing nodes at a high

Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

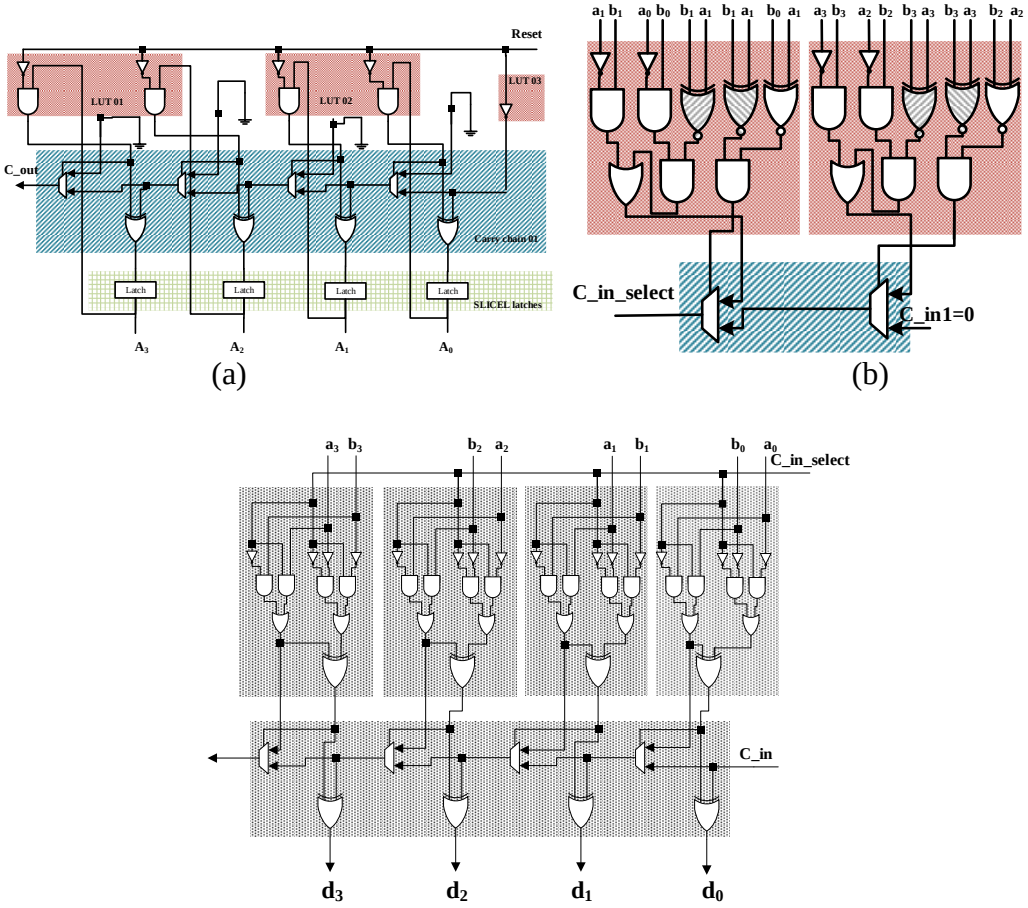
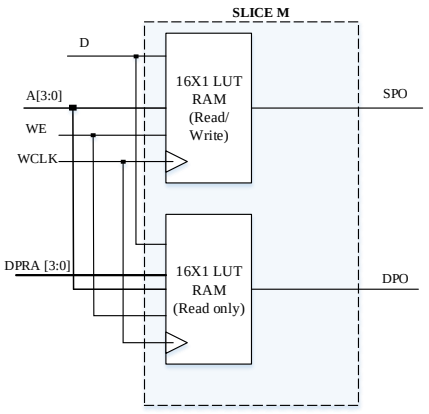


Figure 7 (a) Optimized utilization of LUTs for realisation of Boolean network of (a) ALC (b) TMD (c) TDT using 6-input LUT



(a)

Figure 8 Dual-port distributed RAM block (16X1D) LUT realization for single data bit

rate or in batches, but are processed slowly or irregularly. Modern FPGAs provides a variety of slice elements to support logic, arithmetic, and ROM functions. In addition to this, FPGAs is equipped with some slices to provide additional functions such as storing data using distributed RAM and shifting data with 32-bit registers. Slices that support these additional functions are called SLICEM. Such basic memory capabilities are embedded within the CLBs of various Xilinx FPGA families. Multiple LUTs in a SLICEM can be combined in various ways to store large amount of data. The function generators (LUTs) in SLICEMs can be implemented as an asynchronous RAM resource called a distributed RAM element. RAM elements are configurable within a SLICEM to implement various configurations of RAM [61]: Distributed RAM modules are synchronous (write) resources. A synchronous read can be implemented along with a storage element or a flip-flop in the same slice. The use of flip-flop for realizing the distributed RAM, improves the performance by decreasing the delay into the clock required to operate the flip-flop. However, an additional clock latency is added. The

Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

distributed elements share the same clock input. For a write operation, the Write Enable (WE) input, driven by either the CE or WE pin of a SLICEM, must be set High. The memory structure of FIFO in this work is realized with the help 16x1 dual-port distributed RAM block (16X1D). The 16X1D primitive requires both 16x1 LUT RAMs within a single SLICEM slice, as shown in Fig 8. The first 16x1 LUT RAM, with output on single-port RAM (SPO), implements the read/write port controlled by address A[3:0] to read and write. The second LUT RAM implements the independent read-only port controlled by dual port read only address (DPRA), i.e. DPRA [3:0]. Data is presented simultaneously to both LUT RAMs, again controlled by address A[3:0], WE, and WCLK. The entire RAM block is realized by cascading the distributed RAM blocks n-time for desired n-bit flit width.

The following instantiations were used to map the circuit in Fig 8. for bit-0 of the data flit.

```
RAM32X1D_inst_bit_0: RAM32X1D generic map (INIT => X"00000000") -- Initial contents of RAM port map (DPO(0), SPO(0), WrAd(0), WrAd(1), WrAd(2), WrAd(3), WrAd(4), Din(0), Rdad(0), Rdad(1), Rdad(2), Rdad(3), Rdad(4),WCLK, wr_CE );
```

7. SYNTHESIS AND IMPLEMENTATION

The implementation in this work targets FPGAs that have 6-input LUTs as the basic logic element. In particular, we have considered devices from Virtex-5, Virtex-6 and Virtex-7 FPGA families from Xilinx. The implementation is carried for different word lengths of the data flits needed to be stored. The parameters considered are area, timing and power dissipation. The area is measured in terms of LUTs, flip-flops and slices utilized. Timing analysis may be static or dynamic. Static timing analysis gives information about the Minimum period and operating frequency of the design. Static timing analysis is done post synthesis and post PAR. However, the metrics obtained after synthesis are often not accurate enough due to the programmability of the FPGA which allows for interconnect delays to change significantly between iterations. Therefore, the metrics presented in this paper are post PAR. Dynamic timing analysis verifies the functionality of the design by applying test vectors and checking for correct output vectors. An important result from the dynamic timing analysis is the switching activity information captured in the value charge dump (VCD) file. Apart from post PAR timing analysis the functionality of the design is also verified by dumping the design on the Virtex-5, Virtex-7 platform. Power dissipation is given by the sum of static power dissipation and dynamic power dissipation. Static power dissipation is device specific and is mainly determined by the specific FPGA family. Dynamic power dissipation is related to the charging and discharging of capacitances along different logic nodes and interconnects. Dynamic power dissipation mainly consists of the logic power, clock power and signal power [62]. Logic power depends on the amount of on-chip resources being utilized by the design. Clock power is proportional to the operating frequency. Signal power depends on the switching activity and the density of the interconnects. For simulation and metrics generation similar test benches have been used and are typically designed to represent the worst case scenario (in terms of switching activity) for data entering into the FIFO buffer. Design entry is done using VHDL. As mentioned earlier instantiation based coding strategy is used. The constraints relating to synthesis and implementation are duly provided and a complete timing closure is ensured. Synthesis and implementation is carried out in Xilinx ISE 12.1 [63]. Power analysis is done using the Xpower analyzer tool.

There has been no work regarding the implementation of FIFO buffering policies using the technology dependent optimizations. Since such optimizations are a strong function of the type and nature of the underlying fabric, we have considered some technology independent FIFO buffer realizations that utilize the same FPGA devices. The idea is to provide a comparative analysis of the performance speed up that is achievable using the technology dependent (TD) approaches. However, our initial comparisons focus on the performance improvement achieved over the buffer realizations based on programmable logic unit cells implemented in [43] targeted for Xilinx FPGAs.

A. Area Analysis

Table 2 provides a comparison of the different FPGA resources utilized by the realization of the circular FIFO buffering policy based on the technology optimized sub blocks. The depth of FIFO buffer is 16 bits and the flit order is varied as 23, 24, and 25. Target devices are xc5vlx50, xc6vlx195t and xc7vx485t from Virtex-5, Virtex-6 and Virtex-7. Further analysis is carried out by plotting the various resources utilized as a function of the flit with. The results are shown in Fig 10. The number of Flip flops instantiated from the slice for realizing ALC block remains fixed upto 25 bits this is illustrated in figure 10(b), further more increase in flit size demands more FPGA resources in terms of LUT's and slices this increasing demand is illustrated in figure 10(c) and 10(d) respectively. Since the proposed FIFO buffer is realized with the help of SLICEM BRAM LUT's, therefore in general for an n-bit flit size of FIFO it requires n-memories and n-dual port RAMs. This is illustrated in figure 10 (e),(f) respectively.

Next we compare our implementation against FIFO buffer implementation presented in the [43]. The work presents a FIFO design using flip flop and memory cell design, proposed by the authors. The authors have considered the direct Xilinx ISE based realizations of the Buffer. Two sets of results have been reported giving details about the device utilization summary and timing parameters of the proposed design however, the power dissipation of the proposed design is not reported. The devices considered are Vertex-7 device. We implemented our realization with the same target device and same package. The performance parameters recorded PAR are mentioned in table 1 and table 3 and illustrated in figure 9. It is observed that the FIFO buffer based on the technology optimized mapping realization of the network on LUTs uses the underlying fabric efficiently hence relatively lesser FPGA resources are consumed. But the efficient realization of the underlying fabric puts a prominent critical path delay in the design thus lowering the clock frequency. This has resulted because the instantiation based coding strategy used can ensure constraints related to synthesis and implementation while as mapping constraints are taken care by Xilinx ISE itself. Sometimes the Boolean network may be complex to map, providing poor timing closure while optimizing other goal of

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

parameters of interest. Hence, as a result of the optimization the area, latency and power a tradeoff in the speed is observed. The results of the work [44] are also compared with the proposed realization. The authors have used BRAM Block configurable memory module that is generated by the EDK design tools based on the configuration of the BRAM interface controller IP. The resource utilization summary is not completely mentioned. However, the work has a poor clock speed as mentioned in table 1.

B. Timing Analysis

Technology optimized structures are implemented with minimum possible depth; therefore, the critical path delays are quite low. Since clock frequency is also a strong function of the propagation and routing delays associated with the critical path, a minimum depth circuit also ensures higher operating frequencies. Table 3 provides a comparison of the critical path delay and maximum clock frequency for the FIFO buffer realization based on the technology optimized mapping and the one based on the memory cell based design. The data flit length is 8bits and the depth of the buffer is 16. Target devices are xc5vlx50, xc6vlx195t and xc7vx485t from Virtex-5, Virtex-6 and Virtex-7. Further analysis is carried out by plotting the maximum clock frequency as a function of data flit width and target devices. The results are shown in figure 11. We can observe that the clock speed decreases with the increase in flit size this is due to the increase in the size of TDT block and due to the increase in memory logic needed to make larger flit size FIFO buffers. As the BRAM LUT blocks used for this purpose need a write clock for push operations therefore increasing FIFO memory size affects the clock speed. More over the mapping constraints set by Xilinx ISE itself are not part of an optimization coding strategy hence this also impacts the timing closure while optimizing other parameters of interest thus affecting the clock speed of the FIFO buffer.

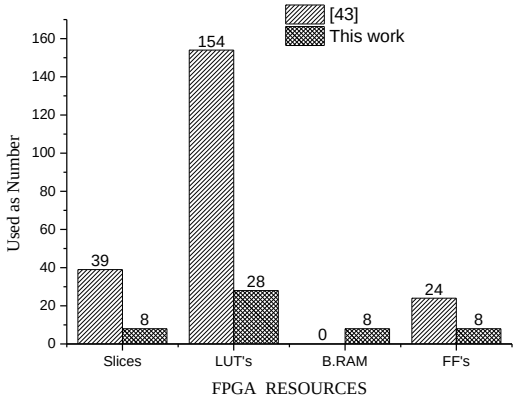


Figure 9 Resource utilization for technology optimized and memory cell based FIFO

TABLE 1 RESOURCE UTILIZATION FOR TECHNOLOGY OPTIMIZED V/s REPORTED WORK.

FIFO Buffer Design	LUTs	Flip-flops	Slices	BRAM	Clock frequency (MHz)
Technology dependent [this work]	28	8	8	8	366
Logic cell unit based [43]	154	24	39	0	293
RAMB_S8_S8 [44]	N.R	N.R	N.R	8	100

TABLE 2 RESOURCE UTILIZATION FOR DIFFERENT FIFO BUFFER REALIZATIONS WITH VARIOUS FLIT SIZES

Device	xc5vlx50t			xc6vlx195t			xc7vx485t		
Package	ff1136			2ff784			-		
Word size	2^3	2^4	2^5	2^3	2^4	2^5	2^3	2^4	2^5
Slice registers	8	8	8	8	8	8	8	8	8
Flip-flops used as	8	8	8	8	8	8	8	8	8
Slice LUT's	24	34	50	26	32	48	28	36	52
Occupied slices	6	10	13	8	10	14	8	11	18
Fully used LUT-FF pairs	8	8	8	8	8	8	8	8	8
No. used as memory	8	16	32	8	16	32	8	16	32
Dual port RAM	8	16	32	8	16	32	8	16	32

TABLE 3 TIMING ANALYSES FOR TECHNOLOGY OPTIMIZED AND LOGIC CELL BASED FIFO

FIFO Buffer Design	**Critical path (ns)**	**Max. clock frequency (MHz)**
Technology dependent [this work]	6.715	293
Logic cell unit based [43]	8.751	366
RAMB_S8_S8 [44]	N.A	100

Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

TABLE 4 CRITICAL PATH DELAY AND MAXIMUM CLOCK FREQUENCY FOR DIFFERENT FLIT SIZES OF BUFFERS REALISED ON VARIOUS DEVICES

Device	**xc5vlx50t**	**xc6vlx195t**	**xc7vx485t**
Max. clock frequency (MHz) 23-bit	263.644	304.822	293.637
Max. clock frequency (MHz) 24-bit	201.288	252.755	200.16
Max. clock frequency (MHz) 25-bit	147.471	198.177	89.314
Critical path delay (ns) 23-bit	4.23	4.86	4.92
Critical path delay (ns) 24-bit	4.91	5.038	5.055
Critical path delay (ns) 25-bit	5.022	5.26	5.33

TABLE 5 POWER DISSIPATION FOR TECHNOLOGY OPTIMIZED FIFO BUFFERS WITH VARIABLE FLIT SIZES

FPGA Resource	**Power Dissipation (mW)**								
	xc5vlx50t			**xc6vlx195t**			**xc7vx485t**		
Flit size (bits)	2^3	2^4	2^5	2^3	2^4	2^5	2^3	2^4	2^5
Clocks	1.03	1	0.9	1.43	1.21	1.15	1.64	1.26	1.21
Logic	0.08	0.32	0.45	0.06	0.29	0.37	0.047	0.27	0.33
Signals	0.74	0.8	1.09	0.7	0.77	1.01	0.55	0.74	0.595
IOs	24.78	27.6	37.75	18.2	21.4	29.12	17.1	19.1	25.33
Dynamic	26.63	29.72	40.19	20.39	23.67	31.65	19.337	21.37	27.465
Quiescent	529.37	560.54	529.38	529.37	712.12	711.95	379.38	206.51	206.44
Total	582.63	619.98	609.76	1.43	1.21	1.15	418.054	249.25	261.37

TABLE 6 POSSIBLE NUMBER OF FIFO BUFFERS THAN CAN BE REALISED USING TECHNOLOGY OPTIMIZED MAPPING.

FIFO Buffer	**FPGA Resource**	**Buffering realization**								
		xc5vlx50t			**xc6vlx195t**			**xc7vx485t**		
	Maximum Memory LUTs available	7680			16720			16720		
	Flit size (bits)	2^3	2^4	2^5	2^3	2^4	2^5	2^3	2^4	2^5
Proposed realization	Number of Buffers possible	960	480	240	2090	1045	522	2090	1045	522
FIFO18 (Xilinx based)	Number of Buffers Present	60	60	N.S	344	344	N.S	1030	1030	N.S

Tables 4 mentions the PAR values of the critical path delay recorded for the proposed FIFO with various flit sizes for the technology optimized realization. The devices considered are xc5vlx50, xc6vlx195t and xc7vx485t from Virtex-5, Virtex-6 and Virtex-7 respectively. The various Flit sizes taken are 8-bit, 16-bit and 32-bits word length. The depth of the buffer is taken as 16.

C. Power Analysis

Technology dependent optimization reduces the power dissipation in two ways. First, the high activity switching nodes within a network are hidden within the LUTs in the final circuit netlist. This reduces the overall switching activity associated with the logic nodes. Second, technology dependent optimization results in a minimal depth circuit with a high logic density. This reduces the length of interconnects. Since interconnects in FPGAs are reconfigurable switches, there is a further reduction in the switching activity and thus the power dissipated. The analysis is done for a constant supply voltage and maximum operating frequency in each case. Test benches were designed for worst-case switching activity and the buffer functionality was verified for more than data flits. The design node activity from the simulator database along with the power constraint file (PCF) was used for power analysis in the Xpower analyzer tool. Table 5 gives the detailed power dissipation for proposed FIFO structure generated using technology optimized mapping. The target device is Virtex-5, Virtex-6 and Virtex-7 and the flit sizes taken are 8-bit, 16-bit and 32-bits word length. Since the power dissipation in the existing work is not reported therefore this paper shows no comparison of the power dissipation with the existing designs or reported work. Furthermore, the power dissipated in clocking resources varies with the clock frequency. Since technology optimized design operates at slightly higher frequency in general but operating frequency decreases with the increase in flit size as explained above, the power dissipated by clocking resources also decreases as illustrated in Figure 12. (a). From figure 12. (b) it can be seen the logic power increases as the flit size increases, this is due to the increase in the logic with increase in the flit size. As the number of inputs, outputs and the respective signals also increase with flit size, thus leading to the increased IO's and signal power as illustrated in figure 12. (c), (d) respectively. Finally, growth of logic with the flit size leads to increased logic activity, thus increased switching activity, hence increased dynamic power dissipation as illustrated in 13. (e). In general, power dissipated by on-chip resources is lesser for technology-optimized design because of the efficient utilization of the underlying resources. Finally, a reduction in switching activity due to hiding of nodes and reduction of interconnects results in lower power dissipation in the signals. Figure 12 gives the variation in power dissipation in different FPGA resources as the flit size for buffering varies. Table 6 gives the possibility of realizing efficient FIFO buffers based on technology dependent optimizations and compare it with the inherent FIFO (FIFO 18) resource present in the FPGA device. Table 6 gives the possibility of realizing efficient FIFO buffers based on

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

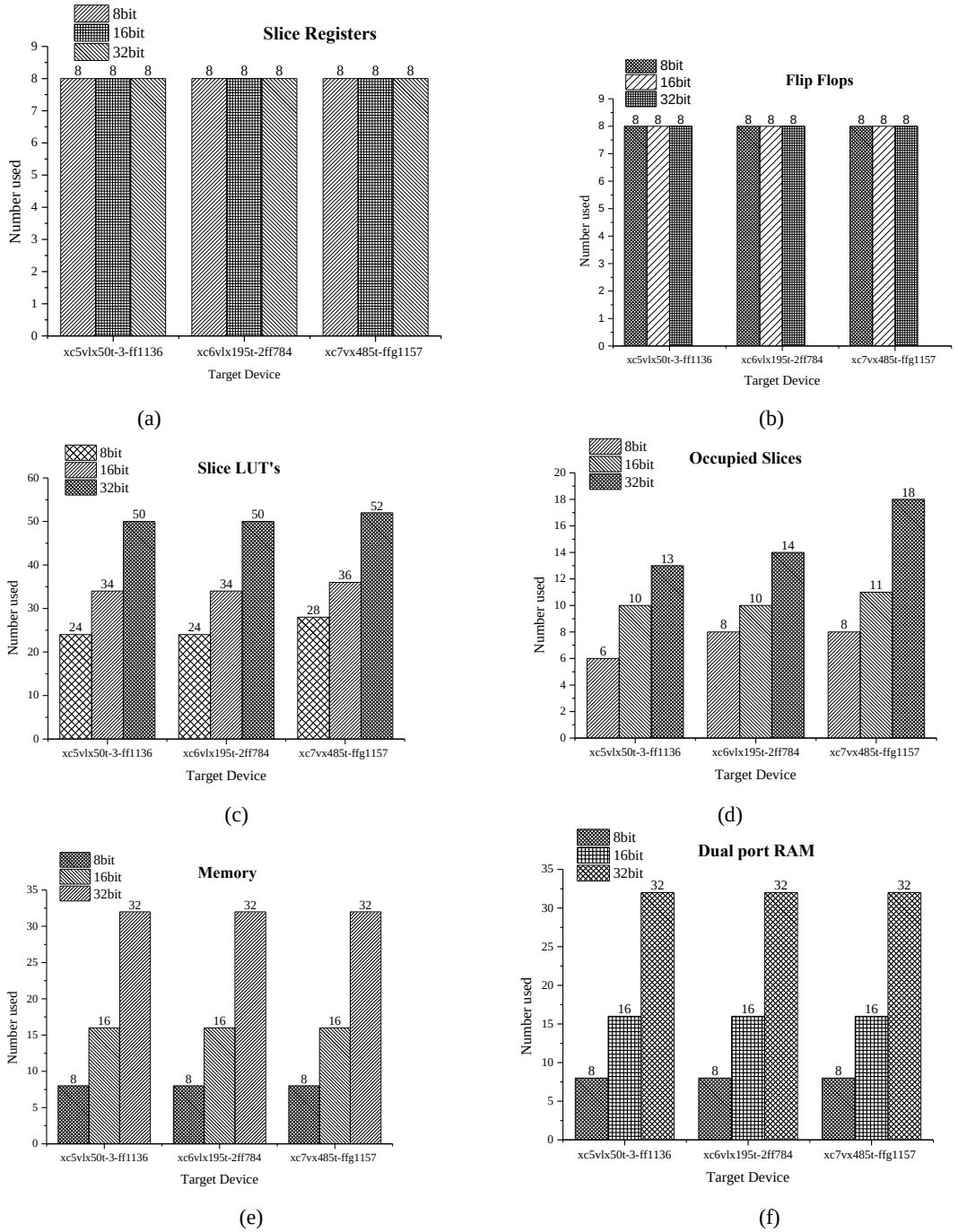


Figure 10 Resource utilization for technology optimized for different flit sizes.

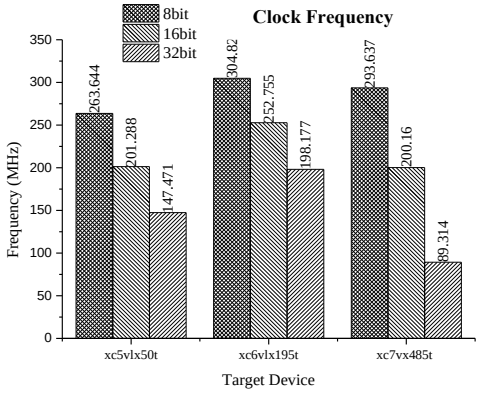
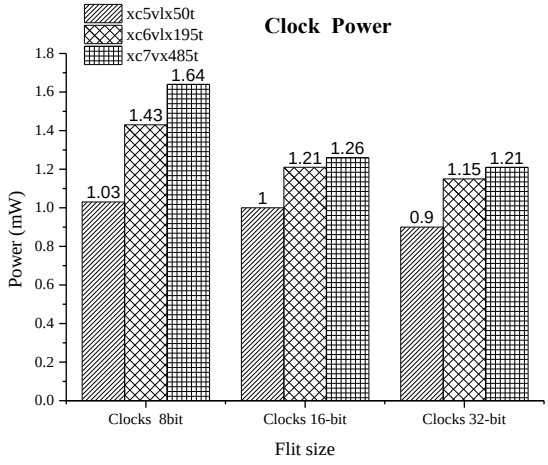


Figure 11 Timing analyses for technology optimized FIFO realizations with different flit sizes.

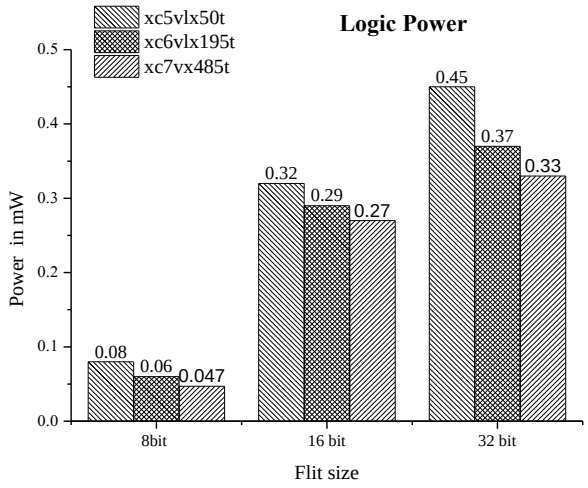
Volume 5, Issue 5
October 2016

ITEE Journal
Information Technology & Electrical Engineering

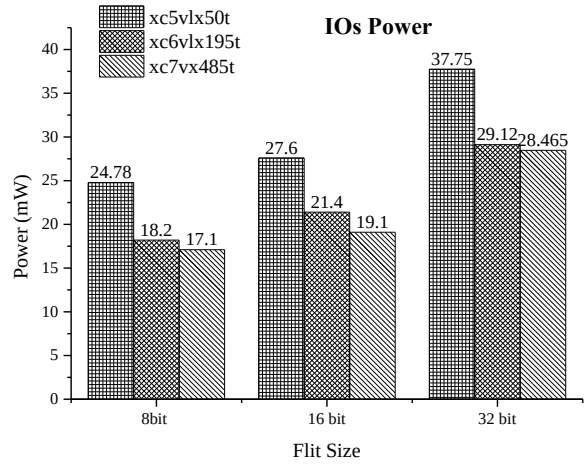
ISSN: - 2306-708X



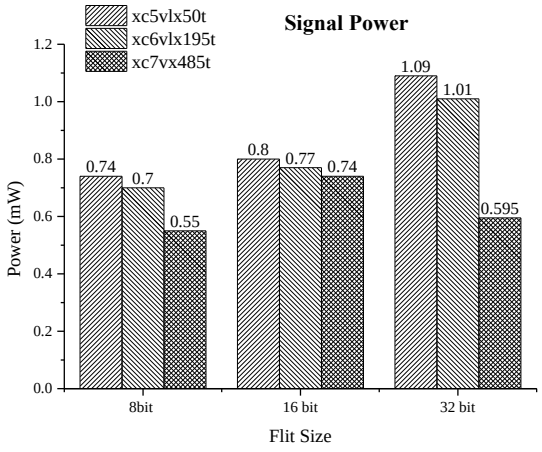
(a)



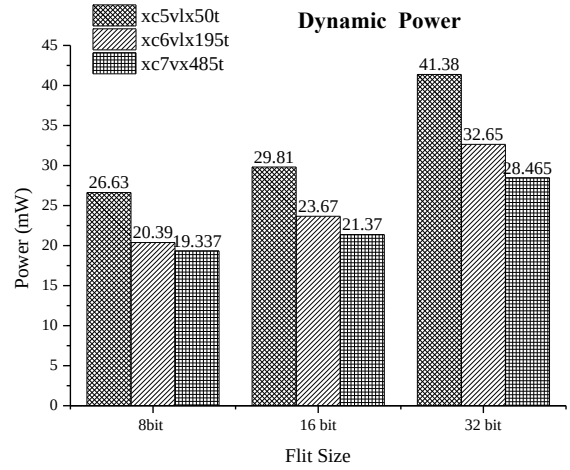
(b)



(c)

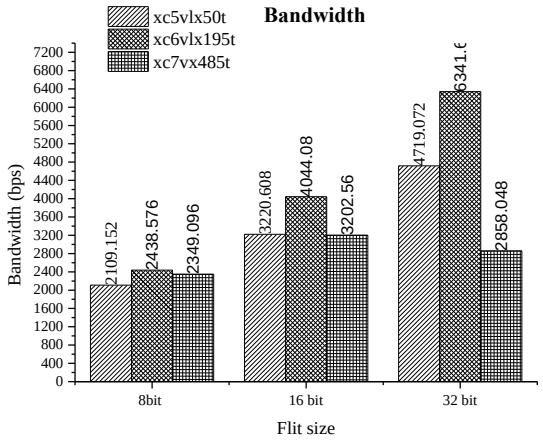


(d)



(e)

Figure 12 Variation in power dissipation in different FPGA resources for technology optimized FIFO realizations with



different flit sizes

Figure 13 Bandwidth supported by the proposed technology optimized FIFO realization with different flit sizes and on different target devices.

technology dependent optimizations and compare it with the inherent FIFO (FIFO 18) resource present in the FPGA device. FIFO 18 can support a flit width up to 18-bits at most and the

flit width of 25 is not supported (N.S). As it can be seen that there are a limited number of FIFO buffers in Xilinx FPGA devices and their number varies as the target device and package varies. The proposed realization helps in eliminating the barrier of having a fixed number of buffers as shown in the table 6. Figure 13 illustrates the bandwidth supported by the proposed realization of the FIFO buffer. The bandwidth of the NoC router is important in determining the latency through the channels and area cost. In this paper, we assume we (ch) = W. Then the bandwidth (BW) of the NoC channel is given by

$$BW=f\_ch\times W \quad (1)$$

Where *f_ch* is the FIFO buffer operating frequency. Increasing in W reduces the contention-free message latency. From figure 13. It can be seen that realizing FIFO with a flit size of 32-bits provides a best bandwidth of 6341.7 bps.

8. CONCLUSIONS

This paper presents a novel idea of realizing circular FIFO buffer using technology dependent optimizations. The results presented in this work showed that technology dependent optimizations have a direct impact on area, delay and power dissipation of the design. FIFO buffers capable of storing NoC traffic with various flit sizes and a fixed depth were implemented and it was shown that for a depth of buffers, the technology optimized realizations will always have an improved performance in terms of various parameters with reduction in the judicious trade-off between area, power and throughput parameters. A key feature of the technology dependent optimization is that the same optimization results in the improvement of all the performance parameters (area and power) and sometimes speed also depending upon the type of circuit network and mapping strategy. This is generally not the case with technology independent optimization where there is always an application driven trade-off that drives the design process. However, performance speed-up through technology dependent optimization strongly relies on the amount of control the designer has over the mapping process. In this paper, we tackled this issue by modifying the coding strategy and writing instantiation based codes to map the behavior of the optimized Boolean networks. This has complicated the design entry and although an efficient mapping is achieved, a complete control over the mapping process still remains a bottleneck in technology dependent optimizations. Another key contribution of this paper is that it has eliminated the bottleneck of having a limited number of FIFO buffer instantiations (limited number of FIFO resources) on FPGA platform which is a major bottleneck for NoC designers to adopt FPGA platforms. The idea of this realization of the buffer will help NoC communication architecture design community to implement NoC based systems easily on the reconfigurable platforms

REFERENCES

[1] Marculescu, Radu, et al. "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives." Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 28.1 (2009): 3-21.

[2] B. Osterloh, H. Michalik, B. Fiethe, K.Kotarowski, "SoC Wire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications," in proc of NASA/ESA Conference on Adaptive Hardware and Systems, pp 51-56, June 2008.

[3] Abdelrasul Maher, R. Mohhamed, G. Victor., "Evaluation of The Scalability of Round Robin Arbiters for NoC Routers on FPGA,"7th International symposium on Embedded Multicore/Manycore System-on-chip, pp61-66,2013.

[4] Akram Ben Ahmaed, Abderazek Ben Abdallah, Kenichi, Kuroda, "Architecture and Design of Efficient 3D Network-on-Chip (3D NoC) for custom multicore SoC," in International confrence on Broadband, Wireless Computing, communication and Application, FIT,Fukuoka, Japan, Nov 2010.

[5] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," ACM Trans. Compute. Syst., vol. 11, no. 4, pp. 319–352, Nov. 1993.

[6] So, Kut C., and E. Chin Ke-tsai. "Performance bounds on multi server exponential tandem queues with finite buffers. "European journal of operational research 63.3 (1992): 463-477.

[7] Buyukkoc, C., "An approximation method for feed forward queueing networks with finite buffers a manufacturing perspective," in Robotics and Automation. Proceedings. 1986 IEEE International Conference on, vol.3, no., pp.965-972, Apr 1986.

[8] Kleinrock, Leonard. Theory, volume 1, Queueing systems. Wiley-interscience, 1975.

[9] Jerger, Natalie Enright, and Li-ShiuanPeh. "On-chip networks." Synthesis Lectures on Computer Architecture 4.1 (2009): 1-141.

[10] Serfozo, Richard. Introduction to stochastic networks. Vol. 44. Springer Science & Business Media, 2012.

[11] Perros, H.G.; Altiok, Tayfur, "Approximate analysis of open networks of queues with blocking: Tandem configurations," in Software Engineering, IEEE Transactions on, vol.SE-12, no.3, pp.450-461, March 1986

[12] M. Karol and M. Hluchyj, "Queueing in high-performance packet switching," IEEE J. Select. Areas Commun., vol. 6, pp. 1587–1597, Dec. 1988.

[13] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in Proc. IEEE INFOCOM '96, San Francisco, CA, pp. 296–302.

[14] Yuan-Ying Chang, Huang, Y.S.-C., Poremba, M. Narayanan, V.Yuan, Xie King, C, "Title TS-Router: On maximizing the Quality-of-Allocation in the On-Chip Network," in IEEE 19th International Symposium on High Performance Computer Architecture (HPCA2013), pp 390-399, Feb 2013.

[15] B. Phanibhushana, K. Ganeshpure, S. Kundu, "Task model for on-chip communication infrastructure design for multicore systems," in proc of IEEE 29th International Conference on Computer Design (ICCD), pp 360-365, oct 2011.

[16] J. Guo, J. Yao, LaxmiBhuyan, "An efficient packet scheduling algorithm in network processors," in proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, pp 807- 818, march 2005.

[17] William John Dally, Brain Towels, Principles and Practices of Interconnection Networks, Ist ed. Morgan Kaufmann publications, 2003.

[18] N.Mckeown, "Scheduling algorithms for input buffered cell switches," Ph.D thesis, University of California at Berkeley, 1995.

[19] K. Lee, Se-jee Lee, hoi-jun Yoo, "A Distributed Crossbar Switch Scheduler for On-Chip Networks," in Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE, pp 671-674, Sept. 2003.

[20] Mello, Aline, et al. "Virtual channels in networks on chip: implementation and evaluation on hermes NoC."Proceedings of the 18th annual symposium on Integrated circuits and system design. ACM, 2005.

[21] Gharan, M.O.; Khan, G.N., "A Novel Virtual Channel Implementation Technique for Multi-core On-chip Communication," in Applications for Multi-Core Architectures (WAMCA), 2012 Third Workshop on, vol., no., pp.36-41, 24-25 Oct. 2012

[22] Peh, Li-Shiuan, and William J. Dally. "Flit-reservation flow control." High-Performance Computer Architecture, 2000. HPCA-6. Proceedings. Sixth International Symposium on. IEEE, 2000.

[23] P. Gupta, N. Mckeown, "Designing and implementing a Fast Crossbar Scheduler," in proc of Micro, IEEE, vol 19, no 1, pp 20-28, Feb 1999

[24] Dally, W.J., "Virtual-channel flow control," in Parallel and Distributed Systems, IEEE Transactions on, vol.3, no.2, pp.194-205, Mar 1992

[25] Saastamoinen, Ilkka, MikkoAlho, and JariNurmi. "Buffer implementation for Proteo network-on-chip." Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on. Vol. 2. IEEE, 2003.

[26] Pande, ParthaPratim, et al. "High-throughput switch-based interconnect for future SoCs." System-on-Chip for Real-Time Applications, 2003. Proceedings. The 3rd IEEE International Workshop on. IEEE, 2003.

[27] Michelogiannakis, G.; Dally, W.J., "Elastic Buffer Flow Control for On-Chip Networks," in Computers, IEEE Transactions on, vol.62, no.2, pp.295-309, Feb. 2013

[28] Schelle, Graham, and Dirk Grunwald. "Onchip interconnect exploration for multicore processors utilizing FPGAs." 2nd Workshop on Architecture Research using FPGA Platforms. 2006.

[29] Schelle, Graham, and Dirk Grunwald. "Exploring FPGA network on chip implementations across various application and network loads." Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on. IEEE, 2008.

[30] Virtex-5 FPGA User Guide, UG190 (v5.4) March 16, 2012

[31] Virtex-6 FPGA Memory Resources, user Guide, UG363 (v1.8) February 5, 2014.

[32] R. Woods, J. McAllister, G. Lightbody and Y. Yi, "FPGA-based Implementation of Signal Processing Systems," Wiley, 2008

[33] Peh, Li-Shiuan, and William J. Dally. "A delay model and speculative architecture for pipelined routers." High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on. IEEE, 2001.

[34] Bertozzi, Davide, et al. "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip." Parallel and Distributed Systems, IEEE Transactions on 16.2 (2005): 113-129.

[35] Jantsch, Axel, and HannuTenhunen, eds. Networks on chip. Vol. 396. Dordrecht: Kluwer Academic Publishers, 2003.

[36] Sunderam, Vaidy S. "PVM: A framework for parallel distributed computing. "Concurrency: practice and experience 2.4 (1990): 315-339.

[37] Saastamoinen, Ilkka, MikkoAlho, and JariNurmi. "Buffer implementation for Proteo network-on-chip." Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on. Vol. 2. IEEE, 2003.

[38] Schelle, Graham, and Dirk Grunwald. "Exploring FPGA network on chip implementations across various application and network loads." Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on. IEEE, 2008.

[39] Oveis-Gharan, Masoud, and Gul N. Khan. "Statically adaptive multi FIFO buffer architecture for network on chip." Microprocessors and Microsystems39.1 (2015): 11-26.

[40] Kogan, Kirill, et al. "FIFO queueing policies for packets with heterogeneous processing." Design and analysis of algorithms. Springer Berlin Heidelberg, 2012. 248-260.

[41] Huang, Po-Tsang, and Wei Hwang. "2-level FIFO architecture design for switch fabrics in network-on-chip." Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on. IEEE, 2006.

[42] Chelcea, Tiberiu, and Steven M. Nowick. "A low-latency FIFO for mixed-clock systems."wvlsi. IEEE, 2000.

[43] Khan, Mohammad Ayoub, and Abdul Quaiyum Ansari. "n-Bit multiple read and write FIFO memory model for network-on-chip."Information and Communication Technologies (WICT), 2011 World Congress on. IEEE, 2011.

[44] Zhang, Yanjun, et al. "Asynchronous FIFO implementation using FPGA."Electronics and Optoelectronics (ICEOE), 2011 International Conference on. Vol. 3. IEEE, 2011.

[45] Ogras, Umit Y., Jingcao Hu, and RaduMarculescu. "Key research problems in NoC design: a holistic perspective." Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis. ACM, 2005.

[46] Liu, Bing Qi, et al. "Research and Design of Asynchronous FIFO Based on FPGA." *Applied Mechanics and Materials*. Vol. 644. Trans Tech Publications, 2014.

[47] Peh, Li-Shiuan, and William J. Dally. "Flit-reservation flow control." High-Performance Computer Architecture, 2000. HPCA-6. Proceedings. Sixth International Symposium on. IEEE, 2000.

[48] Mello, Aline, et al. "Virtual channels in networks on chip: implementation and evaluation on hermesNoC."Proceedings of the 18th annual symposium on Integrated circuits and system design. ACM, 2005.

[49] Gharan, M.O.; Khan, G.N., "A Novel Virtual Channel Implementation Technique for Multi-core On-chip Communication," in Applications for Multi-Core Architectures (WAMCA), 2012 Third Workshop on, vol., no., pp.36-41, 24-25 Oct. 2012

[50] Saastamoinen, I.; Alho, M.; Nurmi, J., "Buffer implementation for Proteo network-on-chip," in Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on, vol.2, no., pp.II-113-II-116 vol.2, 25-28 May 2003

[51] L. Benini, G.D. Micheli, Register designs for queuing buffer, in: Networks on Chips: Technology and Tools, Morgan Kaufmann Publishers, San Francisco, 2006, pp. 65–66.

[52] Y. Choi, T.M. Pinkston, Evaluation of queue designs for true fully adaptive routers, J. Parall. Distrib. Comput. 64 (5) (2004) 606–616 (Orlando, FL).

[53] K. Donghyun, K. Kwanho, K. Joo-Young, L. Seung-Jin, Y. Hoi-Jim, Solutions for real chip implementation issues of NoC and their application to memory- centric NoC, in: First International Symposium on Networks-on-Chip, Princeton, New Jersey, May 2007, pp. 30–39.

[54] By H.J. Yoo, K. Lee, J.K. Kim, Network on chip based SoC, in: Low-Power NoC for High-Performance SoC Design, CRC Press, Boca Raton, 2008, pp. 142–145 Thearchitechture of the FIFO buffers is braoadly classified aS serial and parallel [7–10].

[55] P. Forstner, FIFO Architecture, Functions, and Applications, 1999. <http:// 927 www.ti.com/lit/an/scaa042a/scaa042a.pdf> (accessed 02.04.14).

[56] A. Ling, D. P. Singh, and S. D. Brown, "FPGA Technology Mapping: A Study of Optimality," IEEE Proceedings Design Automation Conference, pp. 427-432, June 2005.

[57] J. H. Anderson and Q. Wang, "Area-Efficient FPGA Logic Elements: Architecture and Synthesis," 16th Asia and South Pacific Design Automation Conference (ASP-DAC), January 2011

[58] Xilinx, "Virtex-5 Family Overview," DS100 (v 5.0) Feb. 6, 2009. www.xilinx.com.

[59] Xilinx, "Virtex-6 Libraries Guide for HDL Designs," UG623 (v 12.3) September 21, 2010. www.xilinx.com.

[60] Xilinx, "Spartan-6 Family Overview," DS160 (v 2.0) October 25, 2011. www.xilinx.com.

[61] Xilinx, "Virtex-6 FPGA Configurable Logic Block," UG364 (v1.2) February 3, 2012.

[62] L. Deng, K. Sobti, Y. Zhang and C. Chakarbarti, "Accurate Area, Time and Power models for FPGA based Implementations," Journal of Signal Processing Systems, Springer, 2011.

[63] http://www.xilinx.com.

AUTHOR PROFILES

Liyaqat Nazir, is presently a Ph.D. scholar in National Institute of Technology from India. He has received the B.Tech degree in Electronics and Communications Engineering from the Islamic University of Science and Technology, India, in 2011,

He did his M.Tech degree in Communications and Information Technology from National Institute of Technology Srinagar, India in 2013. Currently he is a Ph.D. scholar in the department of Computer Science and Engineering, NIT, Srinagar. His main research interests include Network on chip, Digital VLSI design, Mixed signal design. Reconfigurable architectures, Architectural and technology dependent optimizations targeted for FPGA platforms, etc. He has many publications in the related field and is a Graduate student member of IEEE. He is also a lifetime member of IETE.

In recent years, Network on chip has been actively researched. Buffer management and buffering policies are the elementary problem in the applications.

Roohie Naaz received B.E. (Hons) in Electrical Engineering from University of Kashmir (India), M.E. in Computer Science & Engineering from IISc Bangalore (India) in 1990 and Ph.D from University of Kashmir, (India) in 2005. She is currently a Professor in the department of Computer Science & Engineering at NIT Srinagar, India. She is the co-author of many scientific publications in international journals and conferences. Her current research interests include Reconfigurable computing, security and routing in wireless ad-hoc networks, Sensor networks, High level computer architecture design Network on chip, Digital VLSI design, Mixed signal design., Reconfigurable architectures, Architectural and technology dependent optimizations targeted for FPGA platforms, etc. she has authored many research articles in the related field and is a member of IEEE.