An efficient realisation of FIFO buffers for NoC routers using technology dependent optimisations targeting LUT based FPGAs

Liyaqat Nazir* and Roohie Naaz Mir

Department of CSE,
National Institute of Technology Srinagar, India
Email: Liyaqat_02phd13@nitsri.net
Email: Naaz310@nitsri.net
*Corresponding author

Abstract: The communication between processing elements is facing challenges due to power, area and latency. The temporary flit storage blocks needed during communication contributes to the major power and area consumption in Network-on-Chip. Moreover, with modern FPGAs causing a rapid shift from prototype designing to low and medium volume productions, it becomes imperative to consider architectural optimisations that are specific to FPGA fabric only. This article attempts to provide novel optimised FIFO buffer realisation using technology dependent mapping strategies. This will help designers to adopt efficient design of NoC microarchitecture routers. The properties of proposed realisation are studied with a micro-architecture router for several packet flit rates given at an input port. The proposed realisation will help in the elimination of the presence of fixed inherent FIFO buffer instantiations as the proposed realisation gives us an idea to explore underlying FPGA fabric more efficiently for realisation of the FIFO than existing.

Keywords: depth; FIFO; network-on-chip; NoC; flits; traffic.

Reference to this paper should be made as follows: Nazir, L. and Mir, R.N. (2016) 'An efficient realisation of FIFO buffers for NoC routers using technology dependent optimisations targeting LUT based FPGAs', *Int. J. Circuits and Architecture Design*, Vol. 2, Nos. 3/4, pp.201–232.

Biographical notes: Liyaqat Nazir is currently a PhD Scholar in National Institute of Technology (NIT) from India. He received his BTech degree in Electronics and Communications Engineering from IUST, India, in 2011. He did his MTech degree in Communications and Information Technology from NIT Srinagar, India in 2013. Currently, he is a PhD Scholar in the Department of CSE, NIT, Srinagar. His main research interests include network-on-chip, digital VLSI design, mixed signal design. Reconfigurable architectures. He is a graduate student member of IEEE. He is also a lifetime member of IETE.

Roohie Naaz Mir received her BE (Hons) in Electrical Engineering from University of Kashmir (India), ME in Computer Science & Engineering from IISc Bangalore (India) in 1990 and PhD from University of Kashmir, (India) in 2005. She is currently a Professor in the Department of CSE 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. She is a senior member of IEEE.

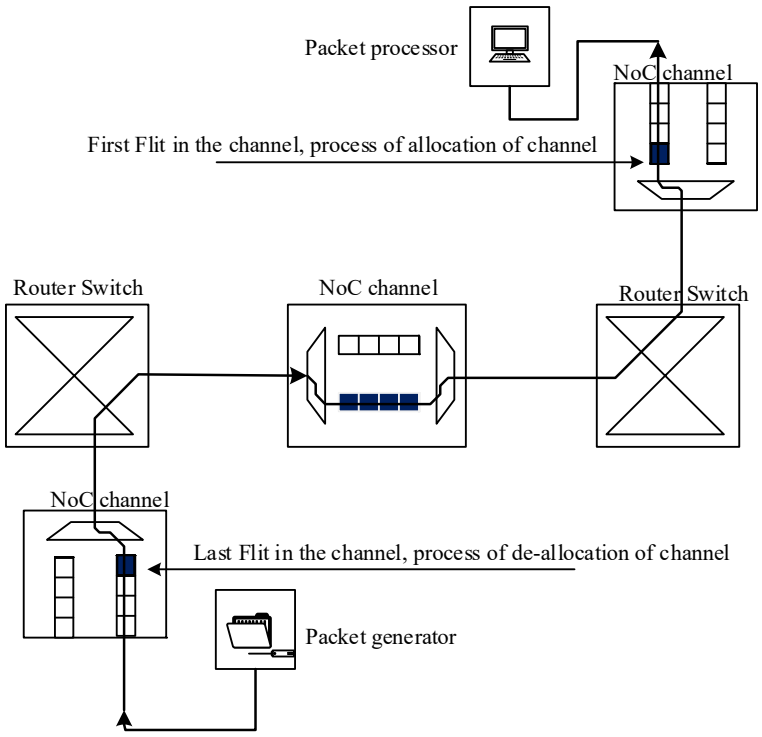
1 Introduction

In the past few years, with the concept of network-on-chip (NoC) communication architecture, NoC has attracted a lot of attention by providing higher bandwidth (BW) and higher performance architectures for communication on the chip (Marculescu et al., 2009). NoC can provide simple and scalable architectures if implemented on reconfigurable platforms (Osterloh et al., 2008). Network on chip offers a new communication paradigm for system on chip (SoC) design (Maher et al., 2013). Many processing elements of SoC are connected through 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 BW and low latency (Ahmaed et al., 2010). 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 (Anderson et al., 1993). 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 (So and Chin, 1992). As router complexity increases with BW demands, very simple routers (non-pipelined, wormhole, no virtual channels (VCs), limited buffering) can be built when high throughput is not needed, so require low area and power overhead (Buyukkoc, 1986; Kleinrock, 1975). Challenges arise when the latency and throughput demands on on-chip networks become increasingly high (Jerger and Peh, 2009). A router's architecture determines its critical path delay which affects per-hop delay and overall network latency (Serfozo, 2012; Perros and Altiok, 1986). Router micro architecture 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 (Marculescu et al. 2009). The area footprint of the router is clearly determined by the chosen router micro architecture 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 (Karol and Hluchyj, 1988; McKeown et al., 1996; Chang et al., 2013; Phanibhushana, 2011). The data paths of the on-chip router comprise of buffers, VC and switching fabric and the control paths of on-chip communication routers are largely composed of arbiters and allocators as illustrated in Figure 1 (Guo et al., 2005). Allocators are used to allocate VC and to perform matching between groups of resources on each cycle (Dally and Towels, 2013; Mckeown, 1995; Lee et al., 2003; Mello et al., 2015; Gharan and Khan, 2012). 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 (Buyukkoc, 1986; Peh and Dally, 2000). 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 (Gupta and Mckeown, 1999). In order to reduce the line of blocking, the rest of the contending flits are buffered into the VC or buffers of the router so as to service them in coming appropriate clock cycles (Dally, 1992). Buffers have simple logic and functionality as compared to the control logic, but in networks they consume most of the area resources (Saastamoinen et al., 2013). 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 (Pande et al., 2003). However, elimination of input buffers eliminates the need of VCs besides causing the reduction in area and power (Michelogiannakis and Dally, 2013). This increases the chances for head-of-line blocking and causes reduction of performance in a NoC based systems. On the other hand, NoC router architecture generally needs large amount of field programmable gate arrays (FPGA) resources (Schelle and Grunwald, 2006, 2009) 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 (Virtex-5, 2012; Virtex-6, 2014). 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 optimise the code quality (Woods et al., 2008; Peh and Dally, 2001). 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 (Bertozzi et al., 2005; Jantsch and Tenhunen, 2003). The high-level concurrent tasks can be then mapped to the underlying communication and computation resources (Sunderam, 1990). This has provided designers with sufficient impetus to look for platform oriented solutions where the underlying hardware can be utilised 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 realising various buffering policies in application specific integrated circuits (ASIC) and FPGA (Saastamoinen et al., 2003; Schelle and Grunwald, 2008; Oveis-Gharan and Khan, 2015; Kogan et al., 2012; Huang and Hwang, 2006; Chelcea and Nowick, 2000; Khan and Ansari, 2011; Zhang et al., 2011).

In this paper, we, therefore, propose a novel FPGA-based efficient realisation of FIFO buffer that will aid in the efficient implementation of NoC router micro architectures on LUT based reconfigurable platforms. We have adopted technology dependent (TD) optimisations 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 TD approach is a strong function of the nature of the target FPGA family. The optimisations 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 TD approach results not only the consumption of lesser amount of resources in designing the buffering network but also gives the possibility of realising 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 realisation 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.

Figure 1 Block diagram of NoC router communication (see online version for colours)



The rest of the paper is organised as follows. Section 2 discusses the related work. Section 3 discusses general FIFO architecture. Section 4 discusses the FIFO realisation proposed in this paper. Section 5 discusses the preliminary terminology and the architectural details. Section 6 discusses the TD optimisation of the multiply-adder unit. Synthesis, implementation and discussions are carried out in Section 7. Conclusions are drawn in Section 8 and references are listed at the end.

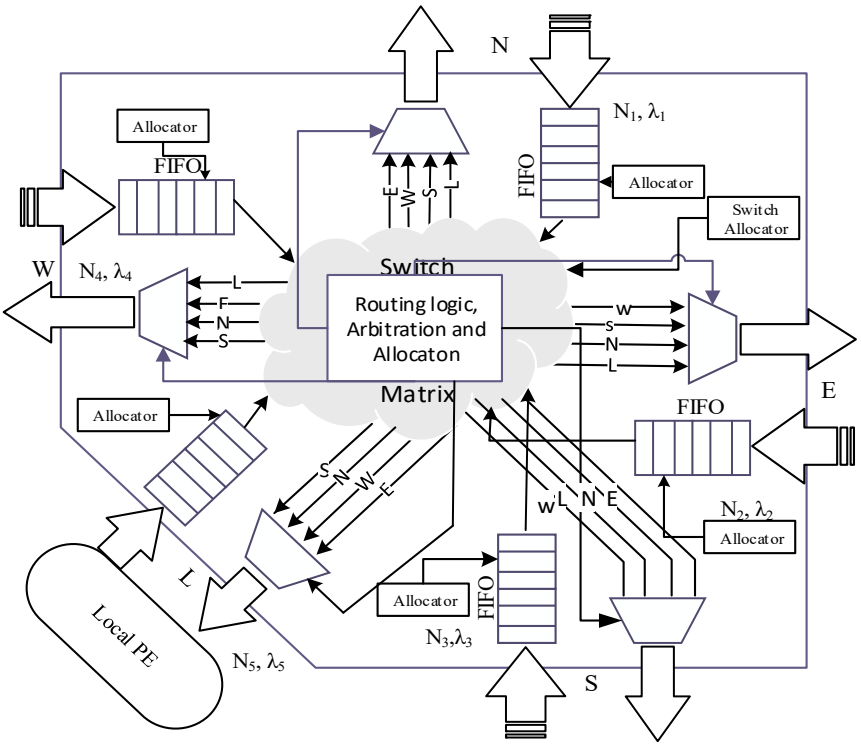
2 Related work

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 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 (Schelle and Grunwald, 2008). 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 over-all area (Ogras et al., 2005). In NoC based architectures buffering policies play a key role in determining the

throughput, latency, area utilisation and energy consumption. In order to reduce the implementation overhead in NoC, Efforts are required to minimise the overall use of buffering resources. Hence, a considerable research effort has been devoted to buffering policies that can be adopted in NoC router micro architecture 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 (Zhang et al., 2011). 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 (Khan and Ansari, 2011) have presented a novel idea of realising 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 TD optimisations 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 Liu et al. (2014). 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 optimisation of router architecture, indirectly by buffer implementation, but logic resource utilisation had not been considered as a performance parameter include the work in Peh and Dally (2000), 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 VC scheme as presented in work (Mello et al., 2005; Gharan and Khan, 2012). VC scheme multiplexes a physical channel using 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 BW, which enables support for quality of service (QoS) in applications (Saastamoinen et al., 2003). The authors in work (Lee et al., 2011) have presented low word length pipelined FIFO buffer. The proposed buffer is designed using the micro-pipeline protocol that is capable to provide relatively higher throughput. The higher throughput is achieved at the cost of the pipelined protocol which in turn requires high FPGA resource overhead hence resource inefficient. Moreover, this high throughput is also shown at smaller packet lengths, increasing the packet length or the state size beyond the considered size will definitely affect the throughput and the operating frequency. All the above-mentioned approaches use technology independent optimisations to enhance the performance of the NoC router. In this paper, we take an alternate approach and propose realisations that are based on TD optimisations. As already mentioned the performance speedup achieved using TD approach is a strong function of the nature of the target FPGA family. The optimisations 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.

Figure 2 Block diagram of NoC router

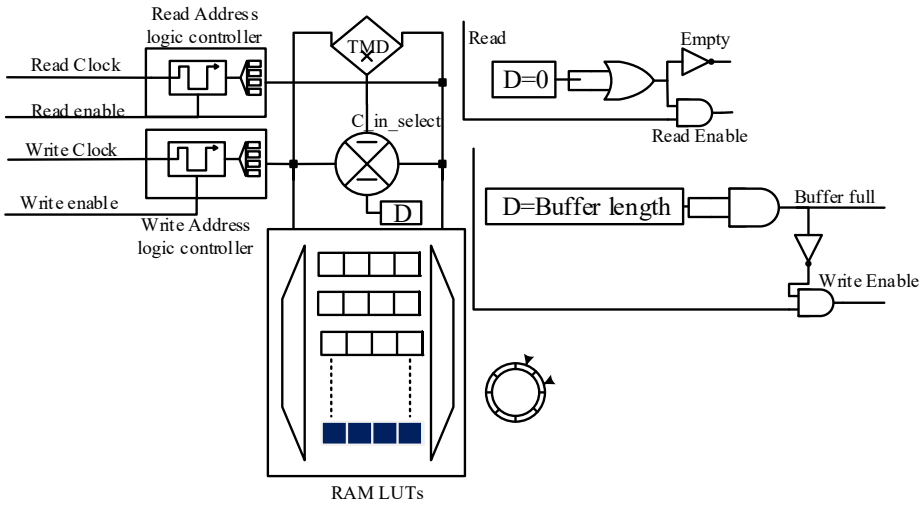


3 FIFO queuing scheme

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 and the FIFO is not full. At the read side, a pop (read) occurs when the upstream channel is ready to receive new data and the FIFO is not empty, i.e., it has valid data to send. There are two types of FIFO designs and architectural schemes: serial and parallel (Benini and Micheli, 2006; Choi and Pinkston, 2004; Donghyun et al., 2007; Yoo et al., 2008; Forstner, 1999). 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 (Ling et al., 2005). 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.

Figure 3 Block diagram of proposed circular buffer (see online version for colours)



4 Proposed FIFO realisation

We propose a novel FPGA based efficient realisation of FIFO buffer that will help in efficient implementation of NoC router micro architectures. The algorithmic interpretation of the proposed FIFO is presented in Algorithm 1. It describes the step-by-step procedure to perform the read write operations for each location of the FIFO memory. The target location for data flits to be stored temporarily is given by the loop index i that varies from 0 to R-1, where R is the depth of the FIFO. The flits to be read from the FIFO are represented by the loop index y. The algorithm is realised as a circular array of identical cells RAM LUTs from SLICEM present in the FPGA fabric. The block level illustration of the algorithm is shown in Figure 3. It mainly comprises of a pair of separate addressable controllers, each for writing (push) and pop operations. A separate full detector and 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 behaviour of the FIFO is controlled by the flow of two tokens, generated by a write address logic controller logic and a read ad-dress 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 token to the left. If it is

de-asserted, 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 neighbouring connecting node. The read address logic controller and the write address logic controller logic are 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 neighbour 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.

Algorithm 1 FIFO buffering algorithm

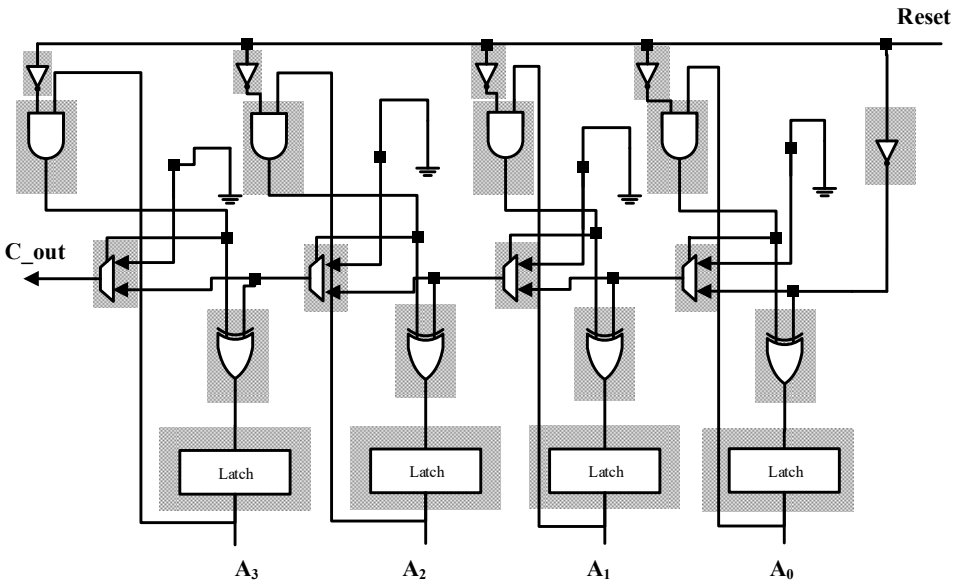
```
R = Number of rows of the FIFO
x ← 0; /* Memory write address*/
y ← 0; /* Memory Read address*/
Empty ← 1;
Full ← 0;
D = x – y;
While (en = 1) do
    While (x ≤ R – 1 and Full != 1 and Wen = 1) do
        Temp ← write(x);
        x ← x + 1;
    end while;
If (x = R) then
        Full ← 0;
End if;
    While(y ≤ R – 1 and D != 0 and Empty != 1 and Ren = 1) do
        read(y) ← Temp;
        y ← y + 1;
    end while;
If (y = R – 1) then
    Empty = 1;
End if;
end while;
```

5 Preliminary terminology and architectural details

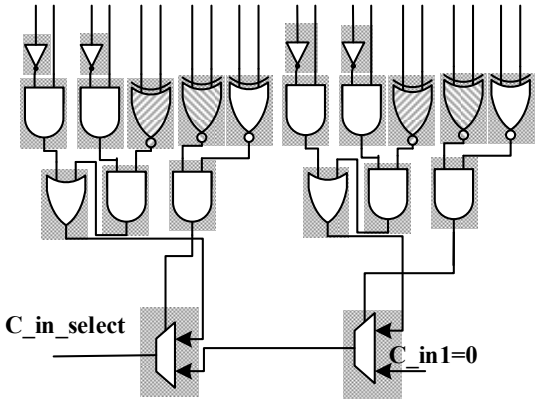
Logic synthesis FIFO buffer is concerned with realising a desired functionality with the minimum possible cost. In the con-text of digital design of a buffering policy, the cost of a circuit is a measure of its speed, area, power or any combination of these. The block level illustration shown in Figure 3 illustrates the broad architectural details. 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 ALC are realised with the help of digital synchronous counter logic network. Two separate n-bit ALC 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 realised with help of fast carry4 chain present in the FPGA target device is shown in Figure 4(a). The logic controller shown is capable of providing an address realisation of FIFO with a depth order of 16 (24 = 16) with address bits A0, A1, A2, A3. These address bits are used for physical address realisation of the RAM blocks and are used by the TDT block. The TDT block is realised with the help of a ripple carry subtraction block illustrated in Figure 5. The TDT for the FIFO is also realised with the help of fast carry4 chain logic present in the target reconfigurable platform. The TDT block takes inputs from TMD block as illustrated in Figure 3. The logic network of a TMD is shown in Figure 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 synchronisation of communication ports during the buffering of data into the actual storage cells or distributive RAM block. The distributive RAM block has been realised as 16 × 1 dual-port RAM16X1D primitive instantiation requiring two 16 × 1 LUT RAMs present within a single SLICEM slice of the underlying fabric, as illustrated in Figure 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 Figure 6. For a general depth of n-bit FIFO realisation, each 16 × 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 specialised 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 bus 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 realisation is supported by various target devices such as Spartan-3 Virtex, Virtex-E, Spartan-II, Spartan-IIE, Virtex-II, and Virtex-II Pro FPGAs.

Figure 4 Logic illustration of (a) ALC (b) TMD



(a)



(b)

Figure 5 Logic illustration of TDT block

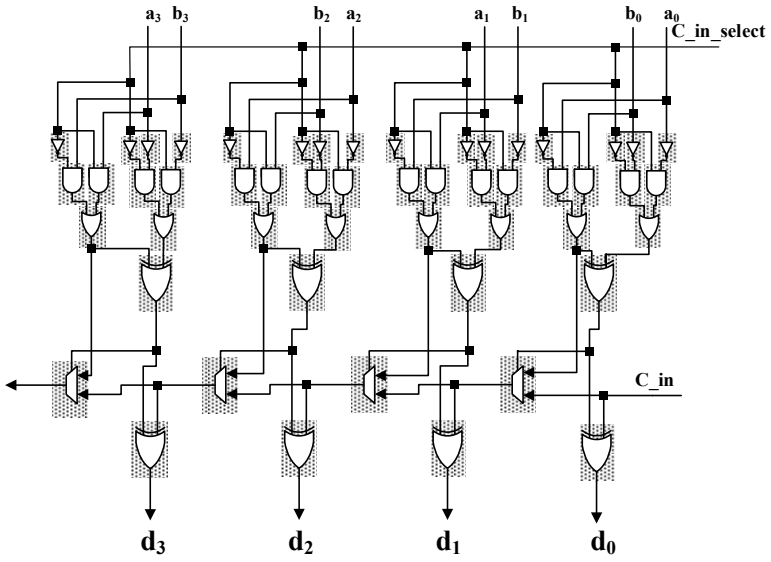
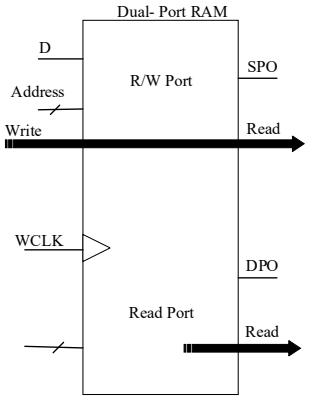


Figure 6 Block level illustration of DRB



6 TD optimisations

TD optimisations are used to transform the initial Boolean network into a circuit net list, 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 utilisation of the target device. The target element in the majority of FPGAs is k-input LUT (Ling et al., 2005; Anderson and Wang, 2011). 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 (Xilinx, 2009, 2010, 2011). An efficient utilisation 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.

TD optimisation 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 optimised sub-networks. Since the circular buffer is an assembly of ALC, TDT, TMD and DRB. An optimised realisation of these individual sub networks could be adopted to realise an optimised realisation of a circular buffering policy.

6.1 TD optimisation of ALC and TMD

Figure 4 shows the Boolean network realisation 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. The overall depth at network output is there-fore, five and four respectively in each network. The LUT count is 21 and 20 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-utilisation of the available resources in the considered network graphs. The number of required LUTs for realisation and the overall depth may be further reduced with the help of tree minimisation 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 realised within the LUT and the total number of inputs is reduced. This is shown in the circuit of Figures 8(a) and 8(b). The circuit, shown in Figure 8 is an optimised realisation 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 optimised realisation of ALC and the LUT depth count in realising TMD has been reduced to one and LUT utilisation is reduced to two. In order to ensure that the optimisation 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 various network circuits illustrated in Figure 8.

Code Instantiation: 1, instantiations 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 );
```

Code Instantiation: 2, instantiations 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));
```

Code Instantiation: 3, instantiations used to map the TDT block

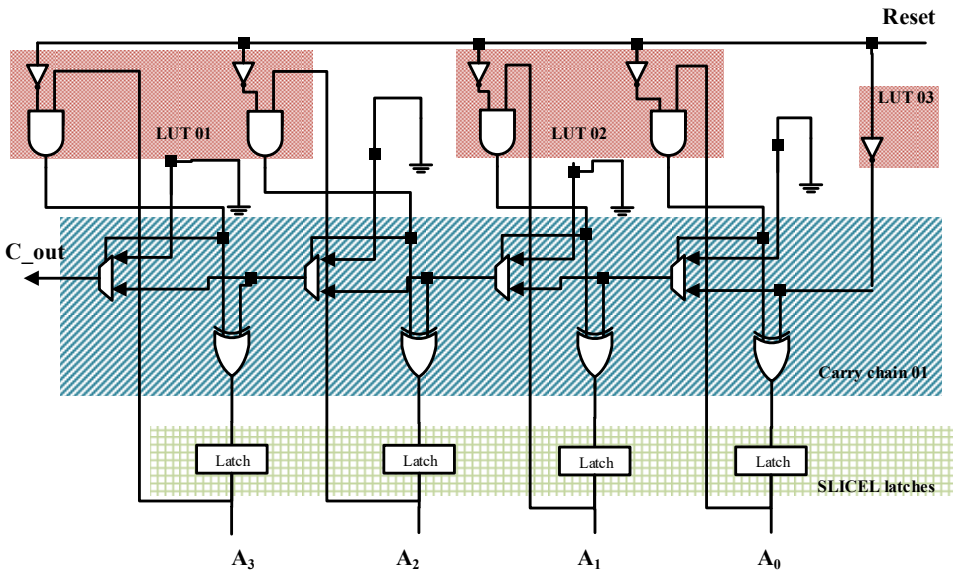
```
LUT6_2_inst0 : LUT6_2 generic map ( INIT => X"ac00000099000000") port map (p(0),
g(0), bin(0), ain(0), cout1(1), '1','1','1');
LUT6_2_inst1 : LUT6_2 generic map ( INIT => X"ac00000099000000") port map (p(1),
g(1), bin(1), ain(1), cout1(1), '1','1','1');
LUT6_2_inst2 : LUT6_2 generic map ( INIT => 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_difference_circuit : CARRY4 port map (CO => cout2, O =>
difference, 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 in case of ALC network. The complete efficient realisation of Boolean network is shown in the Figure 7(a). The LUT utilisation in realising the ALC network is reduced from

seventeen LUTs to three LUTs and the LUT depth count is reduced to one and one carry4 chain. Similarly, the TD mapping of TMD network has resulted in a LUT count of two and LUT depth of one and one carry4 chain as shown in Figure 7(b). The mapping strategy is much efficient as compared to technology independent mapping as illustrated in Figure 4(b) where the TMD network has a LUT count of 22 and LUT depth of four LUTs. The high LUT depth would increase the critical path of the Boolean network hence limits the frequency besides using high amount of hardware resources. The code instantiation 3 is responsible for realisation of TD efficient mapping of TDT Boolean network using 6-input LUTs. The strategy is able to reduce the LUT count of TDT network from 52 LUTs to four LUTs and a reduction of LUT depth from six to one and a carry4 chain is obtained as illustrated in Figure 7(c). Thus, greatly reducing the propagation delay in the network through the LUT. This further helped in reduction of the critical path of the overall FIFO hence increasing the efficiency of the FIFO in terms of speed besides consuming less FPGA resources.

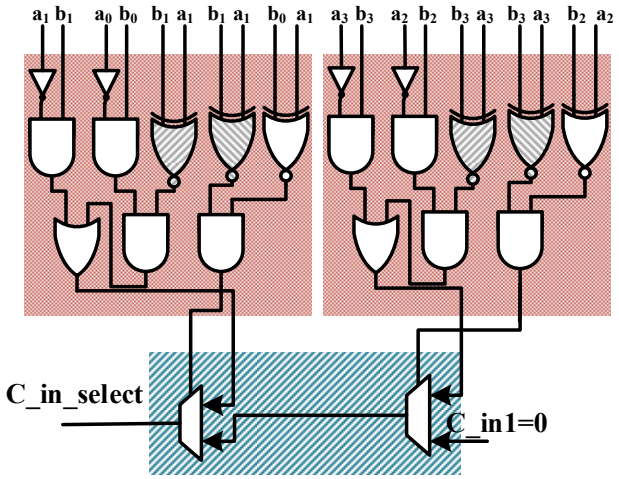
FPGAs have a well-defined design flow that starts with design entry and proceeds through phases like synthesis, translation, mapping and place and route (PAR).It was mentioned in the introductory section that the design cycle in FPGAs is simple due to the availability of the computer aided design (CAD) tools that handle the majority of the TD steps like mapping and PAR. TD optimisations 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 optimisation process is not transparent to the user (Krishnamoorthy and Tessier, 2003). Thus any optimisation done prior to the design entry may get over-ridden during the mapping and PAR phases. To counter this issue we redefine 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.

Figure 7 (a) Optimised utilisation of luts for realisation of Boolean network of (a) ALC (b) TMD (c) TDT using 6-input LUT (see online version for colours)

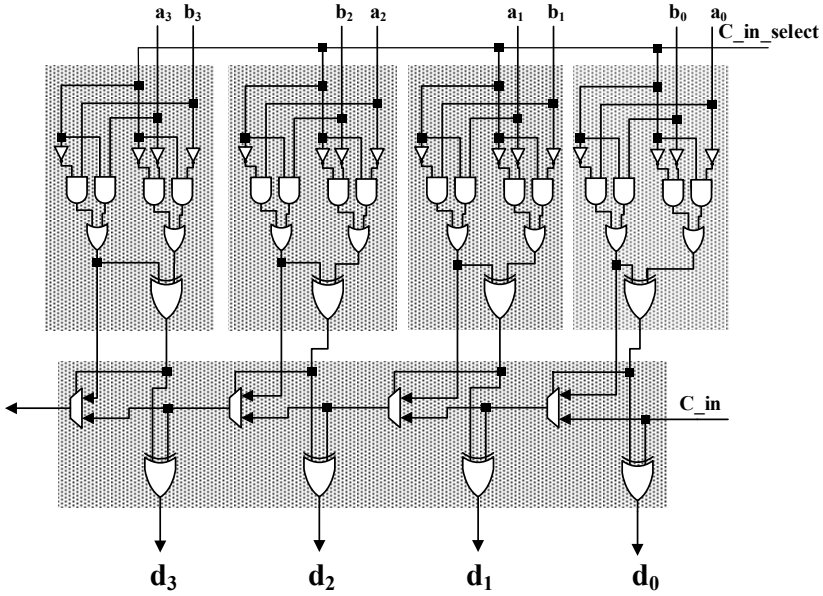


(a)

Figure 7 (a) Optimised utilisation of luts for realisation of Boolean network of (a) ALC (b) TMD (c) TDT using 6-input LUT (continued) (see online version for colours)

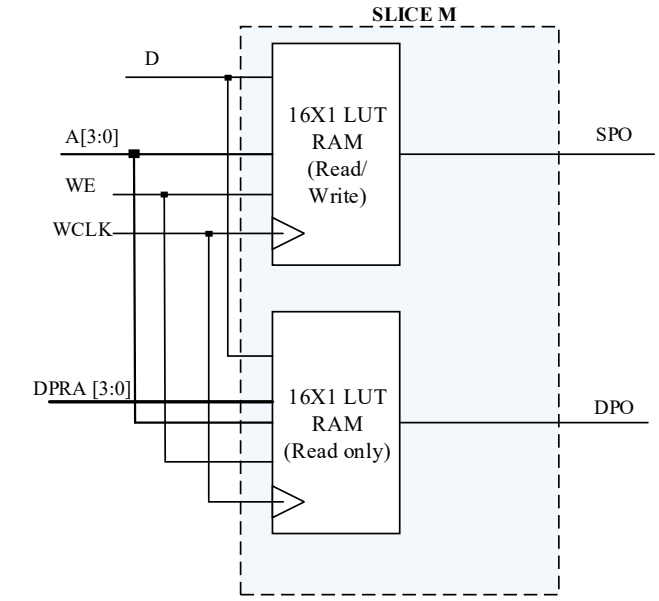


(b)

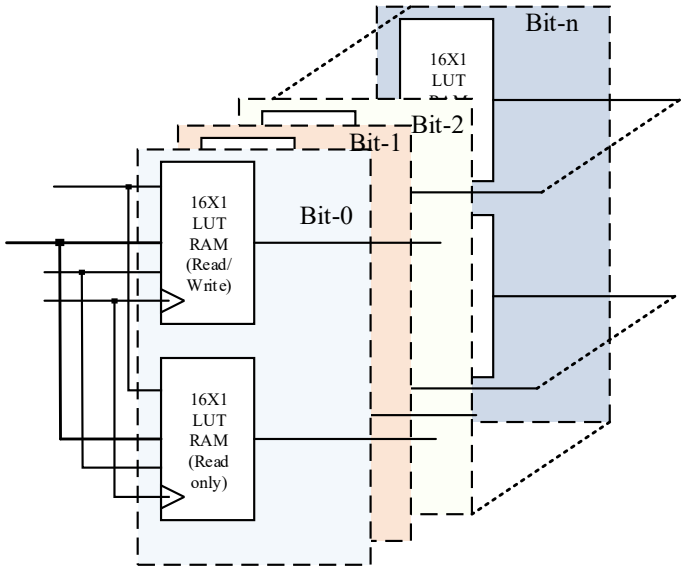


(c)

Figure 8 Dual-port DRB (16x1D) LUT realisation for single data bit (see online version for colours)

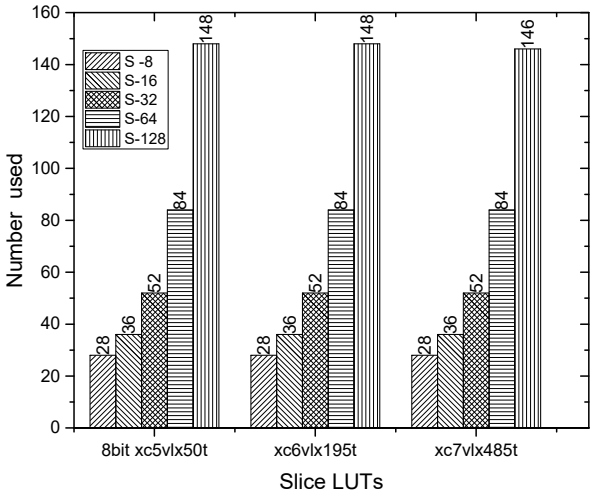


(a)

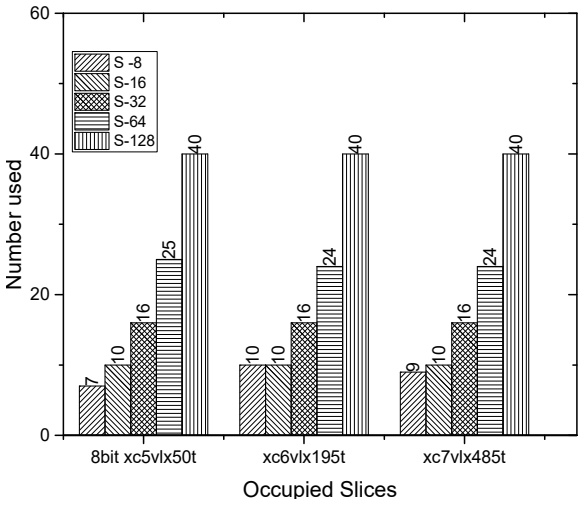


(b)

Figure 9 Resource utilisation for technology optimised for different state sizes



(a)



(b)

6.2 TD optimisation 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 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: (Xilinx, 2012) 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 realising 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 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 realised with the help 16 × 1 dual-port DRB (16X1D). The 16X1D primitive requires both 16 × 1 LUT RAMs within a single SLICEM slice, as shown in Figure 9. The first 16 × 1 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 realised by cascading the DRBs n-time for desired n-bit state size. The instantiation shown in code instantiation: four is used to map the circuit in Figure 8(a). for bit-0 of the data flit.

Code Instantiations: 4, Instantiation to map 1-bit Dual-port DRB

```
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, implementation and results

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 utilised. 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 out-put 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 (Deng et al., 2011). Logic power depends on the amount of on-chip resources being utilised 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 (http://www.xilinx.com). Power analysis is done using the Xpower analyser tool.

There has been no work regarding the implementation of FIFO buffering policies using the TD optimisations. Since such optimisations are a strong function of the type and nature of the underlying fabric, we have considered some technology independent FIFO buffer realisations that utilise the same FPGA devices. The idea is to provide a comparative analysis of the performance speed up that is achievable using the TD approaches. However, our initial comparisons focus on the performance improvement achieved over the buffer realisations based on programmable logic unit cells implemented in (Khan and Ansari, 2011) and micro-pipeline-based implemented in Lee et al., 2011 that are targeted for Xilinx FPGAs.

7.1 Area analysis

Area refers to the embedded resource utilisation, the resource utilisation in FPGA is a vector, with coordinates specific to the given FPGA family. The resource utilisation is an important measure predicting design or algorithm flexibility (Homsirikamol et al., 2011). Table 1 provides a comparison of the different FPGA resources utilised by the realisation of the proposed TD based FIFO buffering policy based on the technology optimised sub blocks. The depth of FIFO buffer (denoted by d) is 16 and the flit order is varied as 23, 24, 25, 26 and 27. 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 utilised as a function of the flit size = state size (denoted by s). since for a constant d up to 24 the logic for ALC, TDT and TMD remains fixed hence the number of flip-flops and slice registers instantiated from the slice for realising these Boolean network blocks should remain fixed and this is listed in Table 1. Varying the s parameter of the buffering policy requires more storage logic, i.e., RAM block and LUTs as a result the resource usage in terms of LUTs, slices and SLICEM BRAM LUT's is likely to go up. The results are shown in Figure 9. Figure 9(a) illustrates the utilisation of slice LUTs and Figure 9(a) illustrates the utilisation of occupied slices. The increased resource utilisation is listed in Table 1. Since the proposed FIFO buffer is realised with the help of SLICEM BRAM LUT's, therefore in general for an n-bit state size of FIFO it requires n-memories and n-dual port RAMs. The area comparison of our TD based implementation against FIFO buffer implementation presented in the Khan and Ansari (2011) is mentioned in Table 2. 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 realisations of the

Buffer. Two sets of results have been reported giving details about the device utilisation 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 realisation with the same target device and same package. The performance parameters recorded PAR are mentioned in Table 1. It is observed that the FIFO buffer based on the technology optimised mapping realisation of the network on LUTs uses the underlying fabric efficiently hence relatively lesser FPGA resources are consumed and results in lesser area delay product (ADP). The TD based realisation has also proved to be efficient in terms of timing parameters as illustrated in Table 2. The results of the work (Chelcea and Nowick, 2000) are also compared with the proposed realisation. 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 utilisation summary is not completely mentioned. However, the work has a poor clock speed as mentioned in Table 1.

Table 1 Resource utilisation for different FIFO buffer realisations with various state sizes

Device	*xc5vlx50t*					*xc6vlx195t*					*xc7vx485t*				
Package	*ff1136*					*2ff784*					-				
State size	2^3	2^4	2^5	2^6	2^7	2^3	2^4	2^5	2^6	2^7	2^3	2^4	2^5	2^6	2^7
Slice registers	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
Flip-flops used as	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
Slice LUT's	28	36	52	84	148	28	36	52	84	148	28	36	52	84	146
Occupied slices	7	10	16	25	40	7	10	14	24	40	9	10	14	24	40
Fully used LUT-FF pairs	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
No. used as memory	8	16	32	64	128	8	16	32	64	128	8	16	32	64	128
Dual port RAM	8	16	32	64	128	8	16	32	64	128	8	16	32	64	128

Table 2 Resource utilisation for technology optimised vs. reported work

FIFO buffer design	*LUTs*	*Flip-flops*	*Slices*	*Clock frequency (MHz)*	*ADP*
TD based	28	12	7	429	12012
Logic cell unit-based (Khan and Ansari, 2011)	154	24	39	366	56364
RAMB_S8_S8 (Zhang et al., 2011)	NA	NA	NA	100	NA

7.2 Timing analysis

Timing analysis attempts to capture the effect of interconnect on the delay within the realised architecture. The interconnect delay varies with the logic block depth of Boolean

networks. Technology optimised 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 realisation based on the technology optimised mapping and the one based on the memory cell based design. Further analysis is carried out by plot-ting the maximum clock frequency as a function of s and target devices. The results are shown in Figure 10. We can observe that the clock speed decreases with the increase in state size this is due to the increase in the SLICEM BRAM LUT's blocks that needs to be clocked simultaneously for realisation of high s value 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 optimisation coding strategy hence this also impacts the timing closure while optimising other parameters of interest thus affecting the clock speed of the FIFO buffer. Tables 4 mentions the PAR values of the critical path delay recorded for the proposed FIFO with various s values for the technology optimised realisation. The devices considered are xc5vlx50, xc6vlx195t and xc7vx485t from Virtex-5, Virtex-6 and Virtex-7 respectively. The various state sizes taken are 23, 24, 25, 26 and 27. The depth of the buffer is taken as 16.

The devices considered are xc5vlx50, xc6vlx195t and xc7vx485t from Virtex-5, Virtex-6 and Virtex-7 respectively. The various s values taken are 2^3, 2^4, 2^5, 2^6 and 2^7. The depth of the buffer is taken as 16.

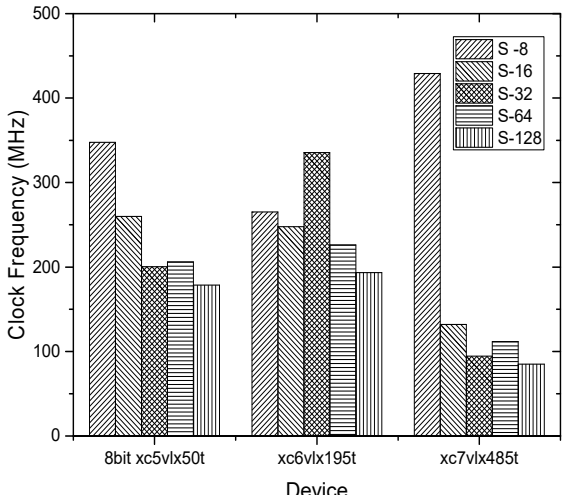
Table 3 Timing analyses for technology optimised and logic cell based FIFO

FIFO buffer design	*D, S*	*Critical path (ns)*	*Max. clock frequency (MHz)*	*BW*
Logic cell unit based Khan and Ansari, 2011)	(16, 8)	2.73	366	2928
RAMB_S8_S8 (Zhang et al., 2011)	(NA, 8)	10	100	
TDM based [this work]	(16, 8)	2.33	429	3432

Table 4 Critical path delay and maximum clock frequency for different sate sizes of buffers realised on various devices

Device	*xc5vlx50t*	*xc6vlx195t*	*xc7vx485t*
Max. clock frequency (MHz) 2^3-bit	347.58	265.11	429
Max. clock frequency (MHz) 2^4-bit	259.87	247.893	132.084
Max. clock frequency (MHz) 2^5-bit	200.441	335.345	94.411
Max. clock frequency (MHz) 2^6-bit	206.058	226.04	111.732
Max. clock frequency (MHz) 2^7-bit	178.66	193.461	85.07
Critical path delay (ns) 2^3-bit	2.877	3.772	2.33
Critical path delay (ns) 2^4-bit	3.848	4.034	8.843
Critical path delay (ns) 2^5-bit	4.989	2.982	10.59
Critical path delay (ns) 2^6-bit	4.853	4.424	8.95
Critical path delay (ns) 2^7-bit	5.597	5.169	11.755

Figure 10 Timing analyses for technology optimised FIFO realisations with different state sizes



7.3 Power analysis

Power consumption has been one of the primary technical figure of merit for selecting an FPGA for the performance of architecture targeted for FPGA platforms. The two primary types of power consumption in FPGA are static (consumed due to transistor leakage) and dynamic power consumed by toggling nodes as a function of voltage, frequency and node capacitance that are switching. TD optimisation 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 net list due to tight logic packing into LUTs. This reduces the overall switching activity associated with the logic nodes. Second, TD optimisation 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 analyser tool. Table 5 gives the detailed power dissipation for proposed FIFO structure generated using technology optimised mapping. The values are recorded for target devices Virtex-5, Virtex-6 and Virtex-7 against the s values 2^3, 2^4, 2^5, 2^6 and 2^7. The dynamic power dissipation is a function of the toggling frequency of the nodes hence with increase in the frequency the dynamic power dissipation should go up this is accurately followed by our design as illustrated in Figure 11. Dynamic and I/O dominates the total power consumption. The I/O buffers that charge and discharge the loads become the main consumer of power. And with the increase in s value I/O power and dynamic power is expected to go up. The trend is listed in Table 5 and is illustrated in Figure 12. As the number of inputs, outputs and the respective signals also increase with s value,

thus leading to the increased I/O's and signal power. The growth of logic with the s value leads to increased logic activity, thus increased switching activity, hence increased dynamic power dissipation as illustrated in Table 5. In general, power dissipated by on-chip resources is lesser for technology-optimised design because of the efficient utilisation 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. Furthermore, the power dissipated in clocking resources varies with the clock frequency. Since technology optimised design operates at slightly higher frequency in general but operating frequency decreases with the increase in state size as explained above, the power dissipated by clocking resources is expected also to decrease from Figure 13. 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. Figure 13 give the relative comparison of the proposed TDM based FIFO realisation with the existing cell unit based FIFO as illustrated in Khan and Ansari (2011). Table 6 shows the comparison of TDM based realisation and micro pipeline based realisation mentioned in Lee et al. (2011), it can be seen that the TDM based is efficient than micro pipeline. The max frequency in micro pipeline seems to be more than TDM based but since the micro pipeline based realisation has a s value of only four as compared to TDM based that has s values of eight, hence for a s value of eight the max frequency of the micro pipeline is expected to decrease and efficiency will further decrease. The band-width of the NoC router is important in determining the latency through the channels and area cost. In this paper, we assume w(ch) = S. Then the BW of the NoC channel is given by

$$BW = f_{ch} \times S \tag{1}$$

where f_{ch} is the FIFO buffer operating frequency. Increasing in S reduces the contention-free message latency. To remove the ambiguity, we have considered band with BW of the design as a figure of merit for comparison. In terms of BW, the TDM based realisation is more efficient. The BW supported by the TDM based realisation against various s values is illustrated in Figure 14. As mentioned above high s value FIFO will provide better BW but the area requirement will also grow affecting critical path delay of the architecture at cost of more clock logic used. At the area-optimised and the delay-optimised extremes, the trade-off between area and delay may become severely unbalanced (Kuon and Rose, 2008). The area delay trade-off is illustrated in Figure 15. Based on the suitable application from Figure 15, we can select a region of elasticity of the buffer where the trade-off is arbitrary, i.e., neither too small nor too large. The region for the TDM based for the target FPGAs is between 60 LUTs to 100 LUTs as illustrated from the plot. Table 7 gives the possibility of realising efficient FIFO buffers based on TD optimisations and compare it with the inherent FIFO (FIFO 18) resource present in the FPGA device. FIFO 18 can support a state size up to 18-bits at most and the state size of 25 is not supported (NS). 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 realisation helps in eliminating the barrier of having a fixed number of buffers as shown in Table 7.

Table 5 Power dissipation for technology optimised FIFO buffers with variable state sizes

FPGA resource	*Power dissipation (mW)*														
	xc5vlx50t					*xc6vlx195t*					*xc7vx485t*				
State size	2^3	2^4	2^5	2^6	2^7	2^3	2^4	2^5	2^6	2^7	2^3	2^4	2^5	2^6	2^7
Clocks	6.5	6.89	8.3	11.5	29.83	6.14	6.28	11.01	19.69	15.23	1.29	1.34	1.65	1.921	2.6
Logic	0.31	0.54	0.62	0.77	0.82	0.14	0.37	0.52	0.69	1.17	0.07	0.08	0.1	0.11	0.13
Signals	0.71	2.27	2.85	5.41	10.9	0.49	2.89	3.9	4.51	6.29	0.43	0.85	1.44	1.5	1.96
IOs	11.8	15.11	29.03	36.51	28	7.37	17.94	22.1	26.5	30.73	8.49	12.54	16.9	17.7	22.64
Dynamic	19.32	24.81	40.8	54.19	69.55	14.14	27.48	37.53	51.39	53.42	10.28	14.81	20.09	21.231	27.33
Quiescent	560.61	560.61	560.61	560.61	560.61	1629	1633	1631	1630	1630	205.7	205.8	205.7	205.8	205.8

Table 6 Timing analyses for technology optimised and logic cell based FIFO

FIFO Buffer Design	*D, S*	*Critical path (ns)*	*Max. clock frequency (MHz)*	*BW*
Micropipeline based (Lee et al., 2011)	(6, 4)	2.11	472	1888
TDM based (this work)	(16, 8)	2.87	348	2784

Figure 11 Dynamic power dissipation vs. toggling frequency in different FPGAs against various s values (see online version for colours)

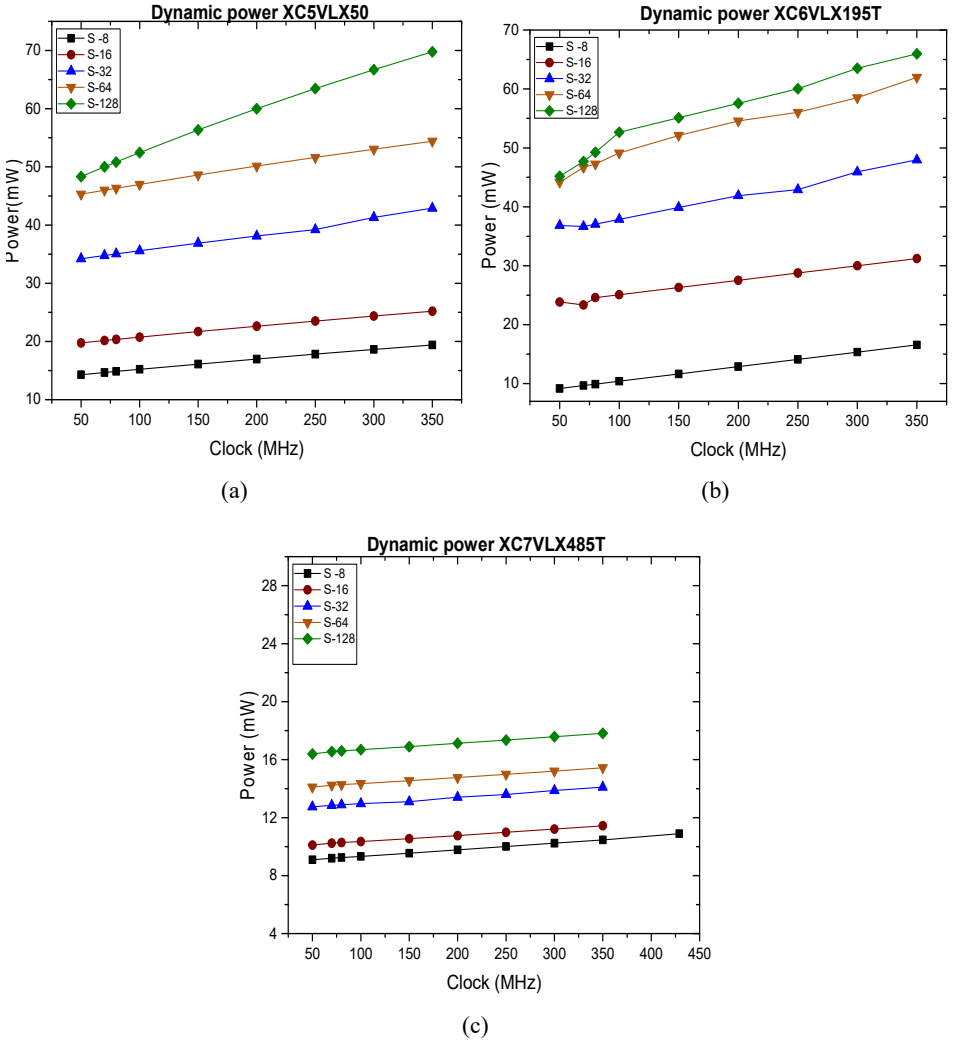


Table 7 Possible number of FIFO buffers than can be realised using technology optimised mapping

FIFO Buffer	*FPGA Resource*	*Buffering realisation*																	
		xc5vlx50t					*xc6vlx195t*						*xc7vx485t*						
	Maximum memory LUTs available	*7680*					*16720*						*16720*						
	State size	2^3	2^4	2^5	2^6	2^7	2^3	2^4	2^5	2^6	2^7		2^3	2^4	2^5	2^6	2^7		
TDM based	Number of buffers possible	960	480	240	120	60	2090	1045	522	261	130		2090	1045	522	261	130		
FIFO18 (Xilinx based)	Number of buffers Present	60	60	NS	NS	NS	344	344	NS	NS	NS		1030	1030	NS	NS	NS		

Figure 12 I/O power of the FIFO targeted in different FPGAs against various s values

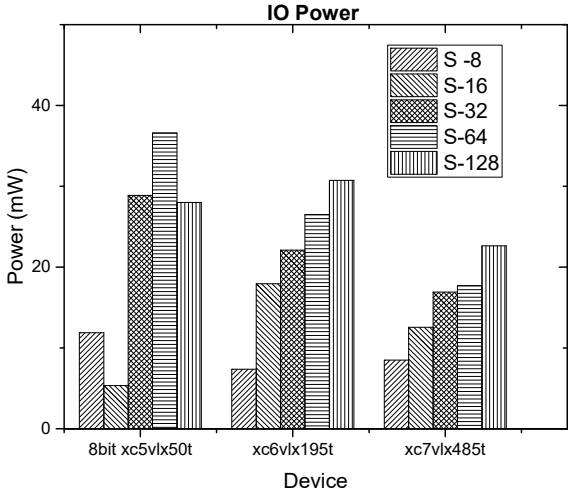


Figure 13 Comparison of TDM based realisation with cell unit based FIFO

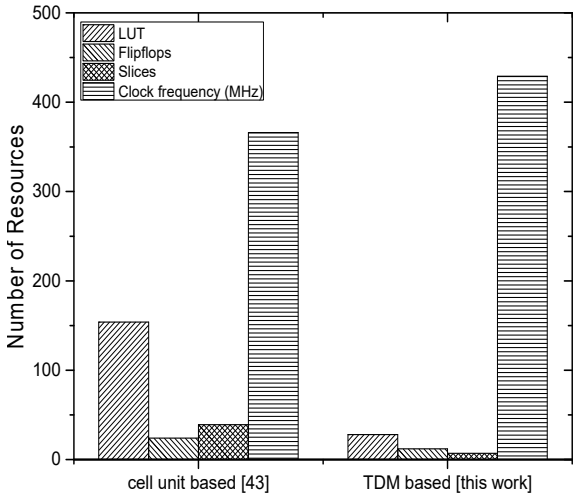


Figure 14 BW supported by the proposed technology optimised FIFO realisation with s values and on different target devices.

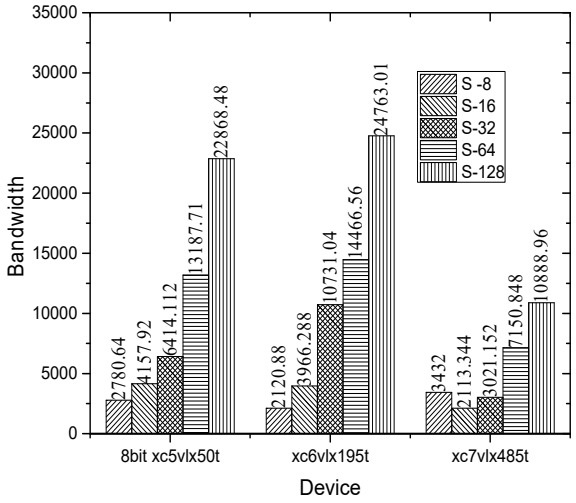
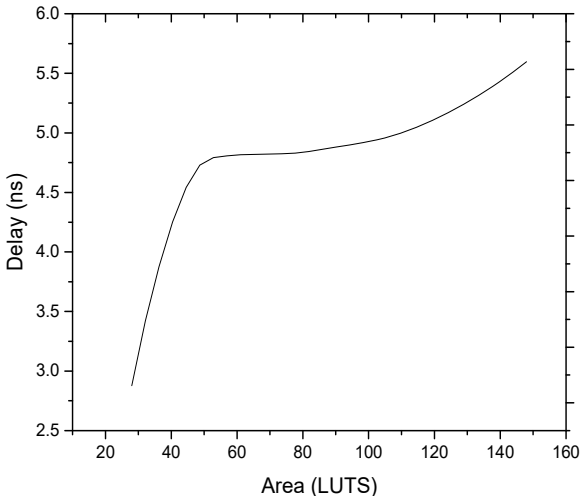


Figure 15 Plot of area vs. delay for TDM based realisation



8 Conclusions

This paper presents a novel idea of sloving the buffering problems for the NoC routers using technology dependent optimisations. The results presented in this work showed that TD optimisations have a direct impact on area, delay and power dissipation of the design. FIFO buffers capable of storing NoC traffic with various state sizes and a fixed depth were implemented and it was shown that for a depth of buffers, the technology optimised realisations 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 TD optimisation is that the same optimisation results in the improvement of all the performance parameters (area, speed and power). This is generally not the case with technology independent optimisation where there is always an application driven trade-off that drives the design process. However, performance speedup through TD optimisation 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 behaviour of the optimised 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 TD optimisations. 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 realisation of the buffer will help NoC communication architecture design community to implement NoC based systems easily on the reconfigurable platforms.

References

Abusaidi, P., Klein, M. and Philofsky, B. (2008) 'Virtex-5 FPGA system power design considerations', *Xilinx WP285 (v1. 0)* 14 February.

Ahmaed, A.B., Abdallah, A.B. and Kuroda, K. (2010) '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*, November, FIT, Fukuoka, Japan.

Anderson, J.H. and Wang, Q. (2011) 'Area-efficient FPGA logic elements: architecture and synthesis', *16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, January.

Anderson, T., Owicki, S., Saxe, J. and Thacker, C. (1993) 'High speed switch scheduling for local area networks', *ACM Trans. Compute. Syst.*, November, Vol. 11, No. 4, pp.319–352.

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

Bertozzi, D. et al. (2005) 'NoC synthesis flow for customized domain specific multiprocessor systems-on-chip', *Parallel and Distributed Systems, IEEE Transactions on*, Vol. 16, No. 2, pp.113–129.

Buyukkoc, C. (1986) 'An approximation method for feed forward queueing networks with finite buffers a manufacturing perspective', in *Robotics and Automation, Proceedings 1986 IEEE International Conference*, April, Vol. 3, No. 1, pp.965–972.

Chang, Y-Y. Huang, Y.S-C., Poremba, M., Narayanan, V., Yuan, X. and King, C. (2013) '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)*, February, pp.390–399.

Chelcea, T. and Nowick, S.M. (2000) 'A low-latency FIFO for mixed-clock systems', *Proceedings IEEE Computer Society Workshop on VLSI 2000. System Design for a System-on-Chip Era*, pp.119–126, Orlando, FL.

Choi, Y. and Pinkston, T.M. (2004) 'Evaluation of queue designs for true fully adaptive routers', *J. Parall. Distrib. Comput.*, Vol. 64, No. 5, pp.606–616, Orlando, FL.

Dally, W.J. (1992) 'Virtual-channel flow control', in *Parallel and Distributed Systems, IEEE Transactions on*, March, Vol.3, No.2, pp.194–205.

Dally, W.J. and Towels, B. (2003) *Principles and Practices of Interconnection Networks*, 1st ed., Morgan Kaufmann Publications, San Francisco, CA.

Deng, L., Sobti, K., Zhang, Y. and Chakarbarti, C. (2011) 'Accurate area, time and power models for FPGA based implementations', *Journal of Signal Processing Systems*, Springer, Springer-Verlag, Heidelberger Platz 3, 14197 Berlin, Germany.

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

Forstner, P. (1999) *FIFO Architecture, Functions, and Applications* [online] http://927www.ti.com/lit/an/scaa042a/scaa042a.pdf (accessed 2 April 2014).

Gharan, M.O. and Khan, G.N. (2012) 'A novel virtual channel implementation technique for multi-core on-chip communication', in *Applications for Multi-Core Architectures (WAMCA), 2012, Third Workshop on*, 24–25 October, pp.36–41.

Guo, J., Yao, J. and Bhuyan, L. (2005) 'An efficient packet scheduling algorithm in network processors', in *Proceedings of 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, March, pp.807–818.

Gupta, P. and Mckeown, N. (1999) 'Designing and implementing a fast crossbar scheduler', in *Proc. of Micro., IEEE*, February, Vol. 19, No. 1, pp.20–28.

Homsirikamol, E., Rogawski, M. and Gaj, K. (2011) 'Throughput vs. area trade-offs in high-speed architectures of five round 3 SHA-3 candidates implemented using Xilinx and Altera FPGAs', *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin Heidelberg.

Huang, P-T. and Hwang, W. (2006) '2-level FIFO architecture design for switch fabrics in network-on-chip', *2006 IEEE International Symposium on Circuits and Systems*, p.4, p.4866, Island of Kos.

Jantsch, A. and Tenhunen, H. (Eds.) (2003) *Networks on Chip*, Vol. 396, Kluwer Academic Publishers, Dordrecht.

Jerger, N.E. and Peh, L-S. (2009) 'On-chip networks', *Synthesis Lectures on Computer Architecture*, Vol. 4, No. 1, pp.1–141.

Karol, M. and Hluchyj, M. (1988) 'Queueing in high-performance packet switching', *IEEE J. Select. Areas Commun.*, December, Vol. 6, No. 9, pp.1587–1597.

Khan, M.A. and Ansari, A.Q. (2011) 'n-Bit multiple read and write FIFO memory model for network-on-chip', *2011 World Congress on Information and Communication Technologies*, pp.1322–1327, Mumbai.

Kleinrock, L. (1975) 'Theory', *Queueing Systems*, Vol. 1, Wiley-Interscience, Radha offset, Delhi.

Kogan, K. et al. (2012) 'FIFO queueing policies for packets with heterogeneous processing', *Design and Analysis of Algorithms*, pp.248–260, Springer, Berlin Heidelberg.

Krishnamoorthy, S. and Tessier, R. (2003) 'Technology mapping algorithms for hybrid FPGAs containing lookup tables and PLAs', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 22, No. 5, pp.545–559.

Kuon, I. and Rose, J. (2008) 'Area and delay trade-offs in the circuit and architecture design of FPGAs', *Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays*, pp.149–158, ACM, New York, NY, USA.

Lee, J-G. et al. (2011) '472MHz throughput asynchronous FIFO design on a Virtex-5 FPGA device', *IEICE Electronics Express*, Vol. 8, No. 9 pp.676–683.

Lee, K., Lee, S-J. and Yoo, H-J. (2003) 'A distributed crossbar switch scheduler for on-chip networks', in *Custom Integrated Circuits Conference, 2003, Proceedings of the IEEE*, September, pp.671–674.

Ling, A., Singh, D.P. and Brown, S.D. (2005) 'FPGA technology mapping: a study of optimality', *IEEE Proceedings Design Automation Conference*, June, pp.427–432.

Liu, B.Q., Liu, M.Z., Yang, G., Mao, X.B. and Li, H.L. (2014) 'Research and design of asynchronous FIFO based on FPGA', *In Applied Mechanics and Materials*, Vol. 644, pp.3440–3444, Trans Tech Publications, Switzerland.

Maher, A., Mohhamed, R. and Victor, G. (2013) 'Evaluation of the scalability of round robin arbiters for NoC routers on FPGA', *7th International Symposium on Embedded Multicore/Manycore System-on-Chip*, pp.61–66.

Marculescu, R. et al. (2009) 'Outstanding research problems in NoC design: system, microarchitecture, and circuit perspec-tives', *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Vol. 28, No. 1, pp.3–21.

Mckeown, N. (1995) *Scheduling Algorithms for Input Buffered Cell Switches*, PhD thesis, University of California, Berkeley.

McKeown, N., Anantharam, V. and Walrand, J.(1996) 'Achieving 100% throughput in an input-queued switch', in *Proc. IEEE INFOCOM '96*, San Francisco, CA, pp. 296–302.

Mello, A. et al. (2005) '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, ACM New York, NY, USA.

Michelogiannakis, G. and Dally, W.J. (2013) 'Elastic buffer flow control for on-chip networks', in *Computers, IEEE Transactions on*, February, Vol.62, No.2, pp.295–309.

Ogras, U.Y., Hu, J. and Marculescu, R. (2005) '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, Jersey City, NJ, USA, USA.

Osterloh, B., Michalik, H., Fiethe, B. and Kotarowski, K. (2008) 'SoC wire: a network-on-chip approach for reconfigurable sys-tem-on-chip designs in space applications,' in *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems*, June, pp.51–56.

Oveis-Gharan, M. and Khan, G.N. (2015) 'Statically adaptive multi FIFO buffer architecture for network on chip', *Microprocessors and Microsystems*, Vol. 39, No. 1, pp.11–26.

Pande, P.P. et al. (2003) 'High-throughput switch-based interconnect for future SoCs', *System-on-Chip for Real-Time Applications, 2003, Proceedings, The 3rd IEEE International Workshop on*, IEEE, Calgary, Alberta, Canada.

Peh, L-S. and Dally, W.J. (2000) 'Flit-reservation flow control', *Proceedings Sixth International Symposium on High-Performance Computer Architecture*, pp.73–84, HPCA-6 (Cat. No. PR00550), Touluse.

Peh, L.S. and Dally, W.J. (2001) 'A delay model and speculative architecture for pipelined routers', *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, pp.255–266, Monterrey.

Perros, H.G. and Altiok, T. (1986) 'Approximate analysis of open networks of queues with blocking: tandem configurations', in *Software Engineering, IEEE Transactions on*, March, Vol. SE-12, No. 3, pp.450–461.

Phanibhushana, B., Ganeshpure, K. and Kundu, S. (2011) 'Task model for on-chip communication infrastructure design for multi-core systems', in *Proc. of IEEE 29th International Conference on Computer Design (ICCD)*, October, pp.360–365.

Saastamoinen, I., Alho, M. and Nurmi, J. (2003) 'Buffer implementation for Proteo network-on-chip', *Proceedings of the International Symposium on Circuits and Systems*, Vol. 2, pp.113–116, ISCAS, Bangkok.

Schelle, G. and Grunwald, D. (2006) 'Onchip interconnect exploration for multicore processors utilizing FPGAs', *In 2nd Workshop on Architecture Research using FPGA Platforms*, 12th February, pp.1–4, Austin.

Schelle, G. and Grunwald, D. (2008) 'Exploring FPGA network on chip implementations across various application and network loads', *2008 International Conference on Field Programmable Logic and Applications*, pp.41–46, Heidelberg.

Serfozo, R. (2012) *Introduction to Stochastic Networks*, Vol. 44, Springer Science & Business Media, Berlin.

So, K.C. and Chin K-T.E. (1992) 'Performance bounds on multi server exponential tandem queues with finite buffers', *European Journal of Operational Research*, Vol. 63, No. 3, pp.463–477.

Sunderam, V.S. (1990) 'PVM: a framework for parallel distributed computing', *Concurrency: Practice and Experience*, Vol. 2, No. 4, pp.315–339.

Virtex-5 (2012) *FPGA User Guide UG190 (v5.4)*, 16 March [online] http://www.xilinx.com [accessed 14June 2014].

Virtex-6 (2014) *FPGA Memory Resources, User Guide, UG363 (v1.8)*, 5 February [online] http://www.xilinx.com (accessed 14 June 2014).

Woods, R., McAllister, J., Lightbody, G. and Yi, Y. (2008) *FPGA-based Implementation of Signal Processing Systems*, Wiley, Chichester, UK.

Xilinx (2009) *Virtex-5 Family Overview*, DS100 (v 5.0) 6 February [online] http://www.xilinx.com (accessed 14 June 2014).

Xilinx (2010) *Virtex-6 Libraries Guide for HDL Designs*, UG623 (v 12.3) 21 September [online] http://www.xilinx.com (accessed 25 June 2014).

Xilinx (2011) *Spartan-6 Family Overview*, DS160 (v 2.0) 25 October [online] http://www.xilinx.com (accessed 7 July 2014).

Xilinx (2012) *Virtex-6 FPGA Configurable Logic Block*, UG364 (v1.2) 3 February.

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

Zhang, Y., Yi, C., Wang, J. and Zhang, J. (2011) 'Asynchronous FIFO implementation using FPGA', *Proceedings of 2011 International Conference on Electronics and Optoelectronics*, , pp.V3-207–V3-209, Dalian.