Design and Implementation of a Parameterized NoC Router and Its Application to Build PRDT-based NoCs

Shankar Neelkrishnan[1], Mei Yang[1], Yingtao Jiang[1], Lei Zhang[1], Yulu Yang[2], Enyue Lu[3], Xiaochun Yun[4]

[1]*Department of Electrical and Computer Engineering, University of Nevada, Las Vegas, NV 89154*
[2]*College of Information Technology and Science, Nankai University, Tianjin, China*
[3]*Department of Computer Science, Salisbury University, MD 21801*
[4]*Institute of Computing Technology, Chinese Academy, Beijing, P. R. China*

Abstract

This paper presents a parameterized router design which can be applied to build large network-on-chips (NoCs) based on a Perfect Recursive Diagonal Torus (PRDT) or mesh/torus topology. In specific, the router is designed to support two routing algorithms (conventional vector routing and a newly proposed Johnson coded vector routing) and wormhole switching. Along these lines, special considerations for the router design are given to different design options concerning scheduling, buffering strategy, and flow control. Correspondingly, the router is partitioned into four components (input channel module, output channel module, crossbar switch, and scheduler) organized to form a three-stage pipeline, and their Verilog models are designed and implemented in a modular fashion with key parameters specified upon instantiations. A 4x4 PRDT-based NoC incorporating multiple copies of the proposed router design is synthesized using TSMC 0.18μm CMOS technology.

1. Introduction

As predicted by the International Technology Roadmap for Semiconductors (ITRS) [8], for the next 5 to 10 years, System-on-Chips (SoCs), using 32*nm* transistors operating below one volt, will grow to multi-billion transistors running at a frequency of 10GHz or higher. One of the major challenges in designing such highly integrated SoCs will be to find an effective way to integrate pre-designed Intellectual Property (IP) cores for power and performance concerns [1]. As the device feature size is continuously shrinking and the bandwidth requirements are increasing, traditional bus-based SoC architectures [1] have been found creating a performance bottleneck. Network-on-Chips (NoCs) thus have emerged as a promising alternative to overcome those limitations of bus-bused architectures by employing a packet-based micro-network for inter-IP communication.

In general, a packet-based NoC consists of routers, the network interface between the routers and the IP, and the interconnection network. The major challenges faced by NoC designs include scalability, energy efficiency, and reconfigurability [3]. In designing an NoC system, the interconnection network is the key in addressing these challenges. Numerous interconnection network topologies have been considered for NoCs, including mesh [10], torus [9], fat tree [5], honeycomb [6], and a few others [3]. However, the aforementioned interconnection topologies are either not scalable or not reconfigurable. To address these two problems, a novel class of topologies named Recursive Diagonal Torus (RDT) [16] is proposed for NoCs [17]. The RDT structure is constructed by recursively overlaying 2-D diagonal torus, and it has the following features: recursive structure with constant node degree, smaller diameter and average distance, and embedded mesh/torus topology. A special type of RDT, called Perfect RDT (PRDT) [14] is considered to be a promising on-chip interconnection network topology due to its symmetric structure and simpler link connections than other RDT structures.

As the interface of an IP to the on-chip interconnection network, the router design has an important impact to the cost and performance a NoC design. In the literature, a number of NoC router designs have been proposed, such as MANGO router [18], ParIS [2]. However, these routers are tailored for a particular topology and cannot be directly applied to PRDT(2,1)-based NoCs.

This paper is focused on developing a parameterized router for PRDT-based NoCs in hardware. As PRDT can be considered as a variation of torus, the router, although designed and optimized for PRDT, is reconfigurable to be applied to build mesh/torus-based NoCs. In specific, the router implements two routing algorithms (vector routing and Johnson coded vector routing), the wormhole switching scheme, the scheduling scheme, buffering strategy, and flow control scheme. Verilog HDL codes are generated for all components and synthesized using TSMC 0.18μm CMOS technology.

The rest of the paper is organized as follows. Section 2 introduces the PRDT structure and the two routing algorithms. Section 3 describes the design and optimization issues of the router components. Section 4 presents the experimental results, and Section 5 concludes the paper.

2. PRDT and Its Routing Algorithms

[1]This work is in part supported by NSF under grant no. ECCS-0702168.

978-0-7695-3099-4/08 $25.00 © 2008 IEEE
DOI 10.1109/ITNG.2008.172

2.1. Structure of RDT and PRDT

The RDT structure is constructed by recursively overlaying 2-D diagonal torus [16]. Consider a torus network composed of *N*x*N* nodes, where $N = nk$ and both n and k are natural numbers. The *rank-0 torus* is formed with four links between node(x, y) and its neighboring four nodes: (mod(x±1, N), y) and (x, mod(y±1, N)). On top of rank-0 torus, *rank-1 torus* is formed by adding four links between node (x, y) and nodes (mod(x±n, N), mod(y±n, N)). The direction of the rank-1torus is at an angle of 45 degrees to the rank-0 torus. On rank-1 torus, *rank-2 torus* can be formed by adding four links in the same manner. In a more general sense, a *rank-(r+1) torus* can be formed upon *rank-r torus*. *RDT(n, R, m)* is a class of RDT in which each node has links to form base torus (rank-0) and m upper tori (the maximum rank is R) with the cardinal number n.

A *perfect RDT* (*PRDT(n, R)*) is a network in which every node has links to form all possible upper rank tori (i.e. RDT(n, R, R)). Particularly, we consider PRDT(2, 1), in which each node has a constant degree of 8 except for PRDT(2, 1) with 4x4 nodes (in this case, the node degree is 5). Fig. 1 shows the structure of an 8x8 PRDT (2, 1) [14]. Due to its symmetric structure, smaller diameter and average distance, and embedded mesh/torus topology, PRDT(2,1) is shown to be promising for interconnecting tens to hundreds of nodes in an NoC system [14].

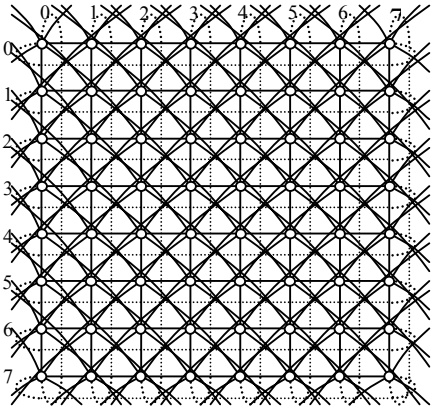

Figure 1 Structure of 8x8 PRDT(2, 1).

2.2 Routing Algorithms for PRDT

Two routing algorithms for PRDT(2, 1) are implemented in our design: the vector routing [16] and the Johnson coded vector routing [12].

2.2.1 Vector Routing Algorithm (VR)

The basic routing algorithm for PRDT (2, 1) is the vector routing algorithm [16], in which a route from a source node to a destination node is represented as a vector. The goal of the vector routing algorithm is to represent the vector with an expression that combines all unit vectors in rank-0 and rank-1 torus and the hop counts. Fig. 2 shows the directions of the unit vector for each rank torus, which rotates in a clockwise direction at a 45-degree incremental with respect to the rank increase.

The vector from a source node to its destination node is denoted as $\vec{A}$, which can be derived as $\vec{A} = vecX_1\vec{X}_1 + vecY_1\vec{Y}_1 + vecX_0\vec{X}_0 + vecY_0\vec{Y}_0$, where $\vec{X}_1$, $\vec{Y}_1$, $\vec{X}_0$, and $\vec{Y}_0$ are the unit vectors in rank-0 torus and rank-1 torus and their respective coefficients, $vecX_1$, $vecY_1$, $vecX_0$, $vecY_0$, give the number of hops on corresponding dimensions.

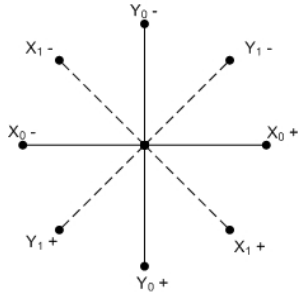

Figure 2 Directions of four dimensions.

The vector routing algorithm for PRDT(2, 1) is listed below.

Vector Routing Algorithm for PRDT(2, 1):

begin

$vecX_1 = (a + b) / 2n$

$vecY_1 = (b – a) / 2n$

$vecX_0 = a – (n * vecX_1 – n * vecY_1)$

$vecY_0 = b – (n * vecX_1 + n * vecY_1)$

end

The vectors are computed at the source node and encapsulated into the packet. At each intermediate node, the routing vectors will be checked in the order of $vecX_1$, $vecY_1$, $vecX_0$, $vecY_0$. The packet will be routed to the direction determined by the first unit vector with non-zero coefficient value (hop count) and this vector's coefficient, depending on the routing direction, will be incremented/decremented before the data packets are actually forwarded to the next hop [16].

2.2.2 Johnson Coded Vector Routing (JCVR)

In the Johnson coded vector routing algorithm [12], the X and Y coordinates of each node of an *N*x*N* PRDT(2, 1) are represented with two m-bit binary Johnson codes, where $m=N/2$. The binary code for a node thus is a 2m-bit binary number by concatenating the Johnson codes of its X and Y coordinates.

Similar to VR, in JCVR, the routing vectors ($vecX_1$, $vecY_1$, $vecX_0$, $vecY_0$) are calculated at the source node. First, the f_r function is applied to determine the direction and the distance of routing on X and Y coordinates. Given two m bits shift-code: A and B, where $A = a_0a_1a_2\ldots a_{m-1}$ and $B = b_0b_1b_2\ldots b_{m-1}$, the distance p is calculated as

$$p = \sum_{i=0}^{m-1} (a_i \oplus b_i),$$

and the direction c is calculated as

$$\text{if } a_0 = b_0 = 0, \text{ then } \quad c = \begin{cases} 1 & if \ \sum_{i=0}^{m-1} b_i > \sum_{i=0}^{m-1} a_i \\ -1 & if \ \sum_{i=0}^{m-1} b_i < \sum_{i=0}^{m-1} a_i \end{cases}$$

$$\text{if } a_0 = b_0 = 1 \text{ then } \quad c = \begin{cases} 1 & if \ \sum_{i=0}^{m-1} b_i < \sum_{i=0}^{m-1} a_i \\ -1 & if \ \sum_{i=0}^{m-1} b_i > \sum_{i=0}^{m-1} a_i \end{cases}$$

$$\text{if } a_0 = 0, b_0 = 1, \text{ then } \quad c = \begin{cases} 1 & if \ \sum_{i=0}^{m-1} \overline{b_i} <= \sum_{i=0}^{m-1} a_i \\ -1 & if \ \sum_{i=0}^{m-1} \overline{b_i} > \sum_{i=0}^{m-1} a_i \end{cases}$$

$$\text{if } a_0 = 1, b_0 = 0, \text{ then } \quad c = \begin{cases} 1 & if \ \sum_{i=0}^{m-1} \overline{b_i} >= \sum_{i=0}^{m-1} a_i \\ -1 & if \ \sum_{i=0}^{m-1} \overline{b_i} < \sum_{i=0}^{m-1} a_i \end{cases}$$

Given the source and destination addresses S=S_x,S_y, D=D_x,D_y, where S_x/D_x and S_y/D_y represent the X coordinate and Y coordinate of the source/destination node in m-bit shift code, respectively, the direction and distance values on both coordinates, (c_x, p_x) and (c_y, p_y), are calculated according to the above formula.

Then the address of each intermediate node M=(M_x M_y) on the routing path from S to D is calculated as follows,

$$M_x = f_s(S_x, c_x, N) = \begin{cases} \text{mod}(S_x + n, N) & when \quad c_x \geq 0 \\ \text{mod}(S_x - n, N) & when \quad c_x < 0 \end{cases}$$

$$M_y = f_s(S_y, c_y, N) = \begin{cases} \text{mod}(S_y + n, N) & when \quad c_y \geq 0 \\ \text{mod}(S_y - n, N) & when \quad c_y < 0 \end{cases}$$

Below lists the JCVR algorithm for PRDT(2, 1).

Johnson Coded Routing Algorithm for PRDT(2, 1):

```
begin
 initialize vecX0 , vecY0 , vecX1 , vecY1 to zero
 let Mx=Sx, My=Sy
 while (Dx ≠ Mx and Dy ≠ My) do
    calculate (cx, px) and (cy, py) using fr function
    if (cx = 0 and cy = 0) then
      set all vectors to zero
    elseif (px + py ≤ 2) then        //do rank 0 routing
      if (cx≥0) then vecX0 = vecX0 + px
      elseif (cx<0) then vecX0 = vecX0 – px
      if (cy≥0) then vecY0 = vecY0 + py
      elseif (cy<0) then vecY0 = vecY0 – py
      goto end
    elseif (px + py > 2) then        //do rank 1 routing
      if (cx≥ 0 and cy≥ 0) then vecX1 = vecX1 + 1
      if (cx≥ 0 and cy< 0) then vecY1 = vecY1 – 1
      if (cx< 0 and cy≥ 0) then vecY1 = vecY1 + 1
      if (cx< 0 and cy< 0) then vecX1 = vecX1 – 1
      calculate Mx and My using fs function
    endif
 endwhile
end
```

For PRDT(2, 1), at most the while loop needs be executed twice to get the routing vector calculated. The routing vectors will be stored in the packet header of each packet sending by the source node. At each intermediate node, the routing vectors will be checked and updated following the same way as in VR.

Apparently, the JCVR algorithm involves more calculations and variables than the VR algorithm. However, JCVR can achieve minimal routing (i.e., the number of hops traversed on each path is minimized), which is not guaranteed in VR.

In our implementation, both algorithms are extended to provide fault-tolerance to account for single link/node failure. Following the fault-tolerance routing scheme proposed in [15], in the case of link failure on X (or Y) dimension, the routing will be detoured to Y (or X) dimension, respectively.

3. Router Design

3.1 PRDT Router Design

The router designed for PRDT(2,1)-based NoC's needs to support up to 9 communication ports to match with the number of links in a PRDT network node, and these ports are named as N, NE, E, SE, S, SW, W, NW and L. The Local (L) is reserved to attach the router to an IP, and all other eight ports (whose names follow the eight directions) connect the router to all its neighbors in eight directions. Each port has both input and output channels.

The main function of the router includes buffering, routing, scheduling, switching, and flow control. In our design, wormhole switching is implemented as it is considered most suitable for NoCs due to its advantage of low data latency with small buffer requirement. The basic flow control unit is flit. The flit size is equal to the phit size, which is fixed as 24 bits. Each data packet is decomposed into flits with the header flit carrying the destination address and the routing vectors.

The data flow of the router is described as follows. When a packet is sent out in flits by an IP to a router, it is stored at the local input port temporarily and waited to be forwarded. The header flit is then read and the routing algorithm is applied to decide the output port. Then the data flits are sent from the input port to the destined output port through an internal switch fabric.

Note that since there may exist flits from more than one input port destined to the same output port, scheduling is needed to resolve this conflict. In addition, flow control is needed to synchronize the data transmission inside the router and between routers.

To implement these functions, the following components are designed: input channel module, output channel module, crossbar switch, and scheduler. Figure 3 shows the top level block diagram of the PRDT router. The router is designed as a Verilog HDL soft-core incorporating these parameterized components. In the following, we will describe the design of each component in detail.

3.2. Components of PRDT Router

3.2.1 Input Channel Module (ICM)

As shown in Fig. 3, each ICM has three inputs ("Packetin", "val", "grant") and three outputs ("ret",

"req", "Packet to crossbar"). The ICM performs the following functions:
(i) It buffers the incoming packets using a FIFO buffer and it analyzes the header.
(ii) It performs the routing algorithm.
(iii) It generates a request ("req") to the scheduler requesting the destined output channel.

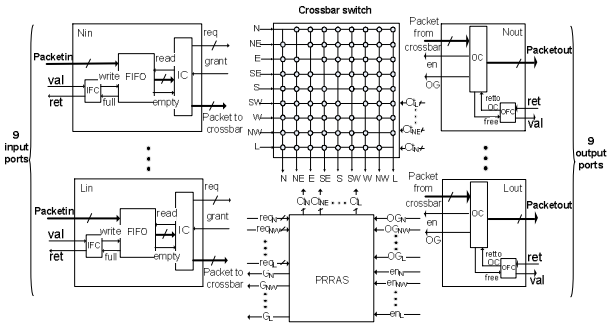

Figure 3 Block diagram of the router.

Each ICM is composed of Input Flow Controller (IFC), FIFO, and Input Controller (IC), which are described as follows.

IFC: It synchronizes the receiving of incoming flits by communicating with the OFC of the adjacent router. A handshake flow control protocol is adopted.

FIFO: The FIFO buffers are used to store flits of packets temporarily blocked inside the router and waiting to be forwarded. The buffer size is parameterized.

IC: The IC at local port embeds the routing algorithm module. Once the header flit reaches the IC, the routing vectors are calculated and updated in the header flit. The IC also generates the request ("req") to the scheduler. The IC at other ports only implements the function of updating the routing vectors as described in Section 2.

3.2.2 Output Channel Module (OCM)

The OCM is responsible for flow control inside the router and outside the router. As shown in Fig. 3, each OCM has two inputs ("Packet from crossbar", "ret") and three outputs ("OG", "val", "Packetout"). It is composed of two architectural blocks named Output Flow Controller (OFC), and Output Controller (OC):

OFC: It synchronizes the transmission of outgoing flits by communicating with the IFC of the adjacent router.

OC: Two main operations are performed at the OC. 1) It sends a grant ("OG") to the scheduler to tell whenever the OC is free. 2) When a tail flit arrives at the OC, it resets the crossbar switch by controlling the "Ct" output of the scheduler using "en" signal.

3.2.3 Crossbar Switch

The 9x9 crossbar switch has 9 data inputs, each connected by the "packet to crossbar" output of an ICM, 9 control inputs ("Ct") from the scheduler, and 9 data outputs, each connected to the "packet from crossbar" input of an OCM.

Instead of using crosspoint-based (81 crosspoints are needed) design [11], for Verilog soft-core implementation, the multiplexer-based design is adopted to build the crossbar switch. In both VR and JCVR, it is illegal for an ICM to request the OCM on the same direction. Therefore, only 72 out of 81 connections are allowed for a PRDT router. Hence, 9 1-to-8 splitters and 9 8-to-1 multiplexers are actually needed (as shown in Fig. 4).

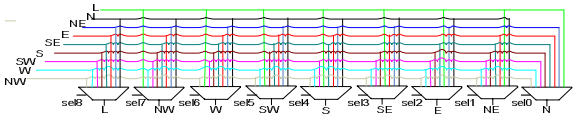

Figure 4 Crossbar switch.

3.2.4 Parameterized Round Robin Arbiter based Scheduler (PRRAS)

The PRRAS receives the requests ("req_i") from ICMs and the OG signals ("OG_i") from OCMs, and determines if there is a matching between the input ports and the output ports. If there is, the grant signals ("G_i") will be sent back to the input ports. The scheduler will also generate control signals ("Ct") to set up the crossbar switch.

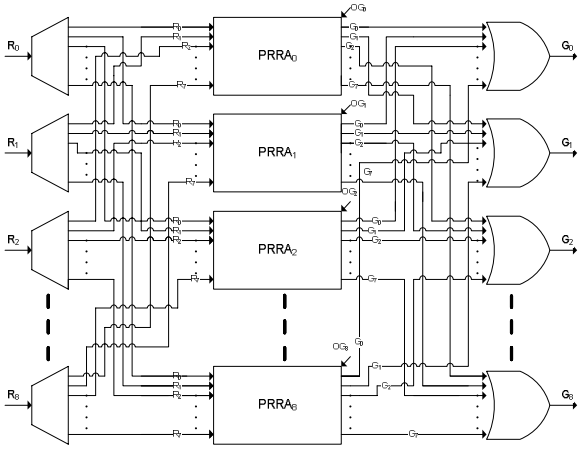

Figure 5 Design of PRRAS.

The PRRAS design is parameterized with the router size. Fig. 8 shows the block diagram of PRRAS for a 9x9 PRDT router, which is composed of 9 decoders, 9 Parallel Round Robin Arbiters (PRRAs) [19], and 9 OR gates. Each PRRA is associated with one output port and responsible for arbitrating the requests to the output port from up to 9 input ports. It is shown in [19] that the PRRA outperforms other round-robin arbiters with a simple binary tree structure.

However, the number of inputs to a PRRA must be a 2's power which is equal to or greater than the input size of the PRRAS. Then, the PRRAS for the PRDT router would have 9 16-input PRRAs, leaving a waste of 7 inputs to each PRRA. To solve this problem, the

scheduler is actually optimized in a slightly different way. For both VR and JCVR, an ICM will not issue a request to the OCM on the same direction (e.g., the ICM on the NE direction will not request for the OCM on the NE direction). That is, each input port will only request the other 8 output ports. Hence, each PRRA only needs to receive the requests from up to 8 ICMs. As such, PRRAS the scheduler is reduced to include just 9 8-input PRRAs.

4. Simulation Results

In this section, the simulation results of the components and whole router design are presented. For all designs, Verilog HDL codes [4] are generated and synthesized using Synopsys's design analyzer [12] targeting TSMC 0.18μm CMOS technology. The performance of each component and the whole router in terms of area (given as the number of 2-input NAND gates equivalent), timing, and power consumption (both dynamic power and leakage power) is reported.

4.1. Results of Components of the Router

4.1.1 Input Channel Module

The ICM contains IC, FIFO and IFC. The FIFO is parameterized, currently designed to contain 4 flits. Two types of ICs are designed: IC for the local port (IC_L), the port that connects the local node to the router, and IC at other ports (IC_O). Two types of IC_L are designed based on the VR and JCVR algorithms with some degree of fault-tolerance capability. Tab. 1 lists the area, timing delay, and power of the IC_L for two routing algorithms. Consistent with our intuition, the results show that IC_L (VR) for VR has less area, power consumption, and timing than IC_L (JCVR).

The results of ICM with IC_L and ICM with IC_O are shown in Tabs. 2 and 3. The results of ICM with IC_o are nearly same for both routing algorithms as the operations involved at those intermediate nodes are same. As the function to be performed at IC_o is much simpler than that at IC_L, ICM with IC_o has significantly smaller power and area footprints than ICM with IC_L.

Table 1: Results of two types of IC_Ls.

		IC_L (VR)	IC_L (JCVR)
Total Area (2-input NAND gate)		117	316.22
Power	Dynamic Power (*mW*)	4.8	11.166
	Leakage Power (*nW*)	14.1	31.37
Timing Delay (*ns*)		5.53	7.23

Table 2: Results of ICMs with IC_L.

		ICM with IC_L (VR)	ICM with IC_L (JCVR)
Total Area (2-input NAND gate)		385.37	584.59
Power	Dynamic Power (*mW*)	5.9	6.55
	Leakage Power (*nW*)	68.3	85.53
Timing Delay (*ns*)		6.07	12.27

Table 3: Results of ICMs with IC_o.

		ICM with IC_o (VR)	ICM with IC_o (JCVR)
Total Area (2-input NAND gate)		339.95	339.19
Power	Dynamic Power (*mW*)	5.67	5.68
	Leakage Power (*nW*)	60.03	60.51
Timing Delay (*ns*)		2.74	2.74

4.1.2 Output Channel Module

The OCM has a very simple functionality as explained in Section 3 and the OCM design is independent of the routing algorithm. The results of the OCM are shown in Tab. 4.

Table 4: Results of OCM.

Total Area (2-input NAND gate)		37.09
Power	Dynamic Power (*mW*)	1.05
	Leakage Power (*nW*)	7.58
Timing Delay (fs)		0.33

4.1.3 Crossbar Switch

Tab. 5 summarizes the results of MUX-based 9x9 crossbar switch design.

Table 5: Results of MUX-based 9x9 crossbar switch design.

Total Area (2-input NAND gate)		1631.73
Power	Dynamic Power (*mW*)	19.77
	Leakage Power (*nW*)	42.75
Timing Delay (*ns*)		1.64

A comparative study of the crosspoint-based and MUX-based crossbar designs is performed using Magic Layout editor, Synopsys CosmosSE and Awaves, and it has revealed that for the given technology, the crosspoint-based design only holds its edge in terms of area, timing, and power consumption for larger N ($N \geq 8$). Thus in this study, we only consider the MUX-based crossbar switch design.

4.1.4 PRRAS

As the main component of PRRAS, PRRA is implemented in a completely parameterized way. The results of the optimized PRRAS are shown in Tab. 6.

Table 6: Results of PRRAS (9 requests).

Total Area (2-input NAND gate)		1151.5
Power	Dynamic Power (*mW*)	22.4
	Leakage Power (*nW*)	82.3
Timing Delay (*ns*)		4.2

4.2 Results of Complete Router

Tab. 7 summarizes the results of the VR-based and the JCVR-based PRDT router designs. Consistent with the results in Tabs. 1 and 2, VR-based router has less area, power consumption, and timing than the JCVR-based router.

Table 7: Results of PRDT routers.

	VR-based Router	JCVR-based Router
Area (2-input NAND gates)	6067.2	6266.4
Dynamic Power (*mW*)	61.2	62.2
Leakage Power (*nW*)	741.8	759.0
Timing Delay (*ns*)	6.34	8.1

4.3 Results of 4x4 PRDT (2, 1)

The router design is used to build the PRDT network. The PRDT(2, 1) network with 4x4 routers has been built. The 4x4 PRDT (2, 1) is a special network that does not need all 9 ports. Three variations of the network design are considered, 1) using 6-port routers instead of 9-port routers, 2) using 9-port routers but with minimum connections, 3) using 9-port routers and with complete connections. Tab. 8 and Tab. 9 tabulate the area, power, and timing results for two sets of 4x4 PRDT networks incorporating the routers based on VR and JCVR algorithms, respectively.

Table 8 Results of 4x4 PRDT(2,1) using VR-based routers.

	Area (2-input nand gates)	Timing Delay (*ns*)	Dynamic Power (*mW*)	Leakage Power (*uW*)
6-port routers	55657.34	6.97	344.35	7.679
9-port routers (minimum)	95455.17	7.06	526.08	11.84
9-port routers (complete)	95972.86	7.06	526.20	11.84

Table 9: Results of 4x4 PRDT(2,1) using JCVR-based routers.

	Area (2-input nand gates)	Timing Delay (*ns*)	Dynamic Power (*mW*)	Leakage Power (*uW*)
6-port routers	58266.83	8	344.44	7.92
9-port routers (minimum)	98667.88	8.1	529.49	12.14
9-port routers (complete)	99185.57	8.1	530.84	12.14

In both tables, the design using 6-port routers reduces the area and power significantly while the design using 9-port routers with minimum connections does not differ much from the design using 9-port routers with complete connections. Comparing the corresponding results of Tables 8 and 9, the VR-based routers have less area and circuit delay, compared to their comparative JCVR-based routers. The power consumption for the two sets of designs is close.

5. Conclusion

In this paper, a few router designs were presented for the PRDT-based NoCs. The routers were designed based on parameterized and synthesizable components following a Verilog HDL-based design methodology. These components include input controller module, output controller module, crossbar switch, and the scheduler PRRAS, and they are synthesized using TSMC 0.18μm CMOS technology. An important feature of this router design is that it can be reconfigured for mesh/torus-based NoCs as the PRDT network naturally embeds the mesh/torus.

Future work includes the implementation of adaptive routing algorithms.

References

[1] L. Benini and G. De Micheli, "Networks on chip: a new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70-78, Jan. 2002.

[2] T. Bjerregaard, J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. DATE*, vol. 2, 2005, pp. 1226-1231.

[3] T. Bjerregaard, and S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Comput. Surv.*, vol. 38, no.1, Jun. 2006.

[4] T. Felicijan and S. B. Furber. "An asynchronous on-chip network router with quality-of-service (QoS) support," in *Proc. IEEE Int'l SoC Conf.*, 2004, pp. 274-277.

[5] P. Guerrier and A. Greiner, "A generic architecture for on chip packet-switched interconnections," in *Proc. Design, Automation and Test in Europe*, 2000, pp. 250-256.

[6] A. Hemani, *et al.*, "Network on a chip: an architecture for billion transistor era," in *Proc. IEEE NorChip Conf.*, 2000.

[7] IEEE Standards Board, *IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language,* New York: IEEE Inc., 1996.

[8] International Technology Roadmap for Semiconductors Roadmap, Available: http://public.itrs.net/.

[9] T. Marescaux, *et al.*, "Interconnection networks enable fine-grain dynamic multi-tasking on FPGAs," in *Proc. 12th Conf. FPGA*, 2002, pp. 795-805.

[10] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The Nostrum backbone - a communication protocol stack for networks on chip," in *Proc. Intl. Conf. VLSI design*, 2004, pp. 693-396.

[11] Pedler, "Approaching crosspoint switches," Available: http://www.commsdesign.com/main/2000/08/0008top.htm.

[12] X. Tan, L. Zhang, S. Neelkrishnan, M. Yang, Y. Jiang, and Y. Yang, "Architecture and routing algorithm for QRDT-based NoC designs," submitted to ISCA 2008.

[13] Synopsys Synthesis Tutorial, Available : http://www.facweb.iitkgp.ernet.in/~apal/lpcs2007/webpages/271_Syn_tut.pdf.

[14] G. Yang, M. Yang, Y. Yang, and Y. Jiang, "On the physical layout of PRDT-based NoCs," in *Proc. ITNG,* 2007, pp. 729-33.

[15] M. Yang, T. Li, Y. Jiang, and Yulu Yang, "Fault-tolerant routing schemes in RDT(2,2,1)/α-based interconnection network for networks-on-chip designs," in *Proc. Parallel Architectures, Algorithms and Networks (ISPAN)*, 2005, pp. 52-57.

[16] Y. Yang, H. Amano, H. Shibamura, and T. Sueyoshi, "Recursive diagonal torus: an interconnection network for massively parallel computers," in *Proc. 5th IEEE Symp. Parallel and Distrib. Processing*, 1993, pp. 591-594.

[17] Y. Yu, M. Yang, Y. Yang, and Y. Jiang, "A RDT-based interconnection network for scalable NoC designs," in *Proc. ITCC*, 2005, pp. 723-728.

[18] C.A. Zeferino, F.G.M.E. Santo, and A.A. Susin, "ParIS: a parameterizable interconnect switch for networks-on-chip," in *Proc. Integrated Circuits and Systems Design,* 2004, pp. 204-209.

[19] S. Q. Zheng and M. Yang, "Algorithm-hardware codesign of fast parallel round-robin arbiters," *IEEE Trans. Parallel and Distrib. Syst.*, vol. 18, no. 1, Jan. 2007.