FPGA Implementation of a Fault-Tolerant Application-Specific NoC Design

Serif Yesil
Bilkent University
Computer Engineering
Ankara, Turkey
Email: serif.yesil@cs.bilkent.edu.tr

Suleyman Tosun
Hacettepe University
Computer Engineering
Ankara, Turkey
Email: stosun@hacettepe.edu.tr

Ozcan Ozturk
Bilkent University
Computer Engineering
Ankara, Turkey
Email: ozturk@cs.bilkent.edu.tr

***Abstract*—Today's integrated circuits are more susceptible to permanent link failures than before as a result of diminishing technology sizes. Even a single link failure can make an entire chip useless. Single link failure problem is fatal to application-specific Network-on-Chip (NoC) designs as well if they cannot tolerate such failures. One solution to this problem can be having alternative routing options on the network for each communicating pair. In this study, we present an FPGA implementation of such a method for application-specific NoCs. This method adds additional network resources to the non-fault-tolerant design in an attempt to make it fault-tolerant. We show the effects of the presented fault-tolerant method on an FPGA implementation of Mp3 encoder based on energy consumption and area increase against non-fault-tolerant case.**

I. Introduction

Network-on-Chip (NoC) has been introduced as a scalable communication method to be an alternative to classical bus-based and point-to-point communication structures [1], [2]. In this method, all communicating module pairs send the information in packets to each other via routing elements (e.g., routers and links). An NoC communication infrastructure has advantages of performance and scalability over design-specific global wiring alternative [1].

Network topology and routing mechanism of an NoC design are two important factors that affect the performance, chip area, energy consumption, and fault-tolerance. An NoC architecture can use either regular or irregular topology based on design needs. While irregular topologies give larger optimization space for performance, area, and energy consumption than regular counterparts [3], regular topologies have scalability, routing, and even fault-tolerance advantages. Mesh and torus are two examples of regular topologies that can tolerate link and router failures since every path between two communicating nodes has its alternative in case of a failure. Fig. 1.(a) demonstrates this property of the mesh topology. In this figure, nine routers are connected to nine nodes of an application. In this design, if a link on path p_1 has a permanent fault, the packet sent from node 1 can use the alternative path p_2 to reach node 5 without performance loss. In some cases, alternative routing options with some small performance degradation can be used as well.

Fault tolerant design methods and routing algorithms for the regular topologies have been well studied in previous work [4],

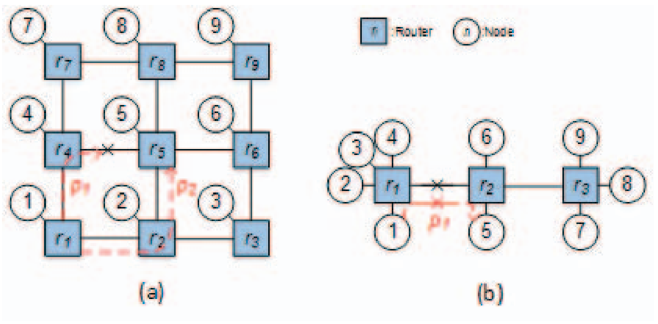

Fig. 1. Two topology examples for an application with nine nodes. (a) Regular topology (mesh) and (b) irregular topology. All the routers in both topologies have five bidirectional ports.

[5], [6], [7]. On the other hand, most of the previous studies for irregular counterparts have focused on generating topologies based on only energy consumption and performance criteria [3], [8], [9]. The generated topologies without considering fault-tolerance may not tolerate link failures since there may be only one routing path between communicating nodes. We give an example irregular topology in Fig. 1(b) as an alternative to the mesh NoC given in Fig. 1(a). This irregular topology has three five-port routers (i.e., routers have the same number of ports with the ones used in the mesh topology) connected to each other and there is only one path from each router to the others. If there is a link failure between r_1 and r_2, the communication between nodes 1 and 2 fails. These kind of permanent failures make the entire chip useless, which results in profit loss. One way to overcome link failures in this case can be doubling the links. However, this solution tremendously increases the area and energy consumption of the routers.

One less costly solution to the above stated problem is adding very small number of network components to the non-fault-tolerant topology and having alternative paths for all pairs of routers. In such a design, different routing tables must be embedded to the routers for each possible link failure. If a link failure present on a link, corresponding routing tables for the alternative routing can be powered up with a possible performance loss. Fig. 2(a) illustrates this idea. This fault-tolerant irregular topology is generated by adding additional

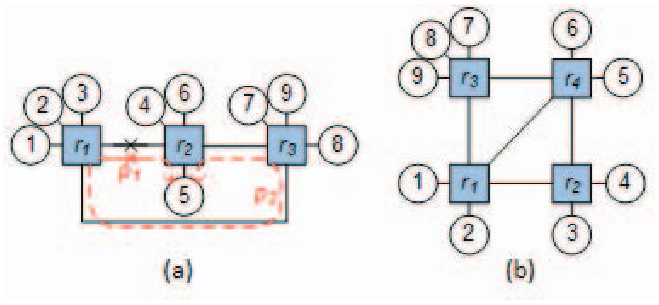

Fig. 2. Two irregular topology examples (a) with three routers and (b) with four routers for an application with nine nodes. All the routers in both topologies have five bidirectional ports.

link between routers r_1 and r_3 to the non-fault topology given in Fig. 1(b) In the fault tolerant topology, if there is a link failure on p_1 between r_1 and r_2, nodes 1 and 5 can communicate on p_2 through routers r_1, r_3, and r_2, increasing the number of hops from 1 to 2. If we add one more router to this fault tolerant topology as shown in Fig. 2(b), each communicating node can have more routing options. In this example, nodes 1 and 5 have three different routing options with different hop counts.

Fault-tolerant topology generation method for application specific NoCs was presented in [10], [11] while routing table generation to cover each link failure was presented in [12]. However, these studies demonstrate the effects of the above mentioned method using only simulation techniques. They do not test it on real hardware implementations.

In this study, we aim to fulfill this by implementing the fault-tolerant NoC design method on Field Programmable Gate Arrays (FPGAs). To do that, we first implement routers with different number of ports. The routers use routing-table-based routing strategy so that they can be configured externally for alternative routing options. We then connect them to emulate the fault tolerant topology, non-fault tolerant topology, and ring topology, which is the simplest fault-tolerant regular topology having two alternative paths for each router. We map the nodes of a well-known multimedia benchmark, Mp3 encoder, onto these topologies using simulated-annealing-based mapping and emulate them. We compare these three topology types in terms of area and energy consumption. The results show that, with a small area increase, the presented FPGA implementation of application specific NoC design achieves fault tolerance capability at least for one single link failure.

II. Fault-Tolerant Topology Generation

Fault-tolerant application-specific NoC design has three main steps: 1) Topology generation, 2) Application mapping, and 3) Routing. The input application is generally represented by a communication flow graph (CFG) $G(V,E,W)$, where V, E, and W respectively denote the nodes, edges connecting the nodes, and edge weights of the application. Fig. 3 shows the CFG of Mp3 encoder [11]. We will use this application

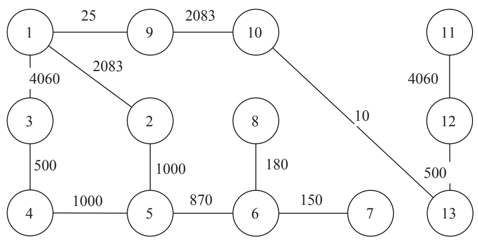

Fig. 3. CFG of Mp3 encoder [11].

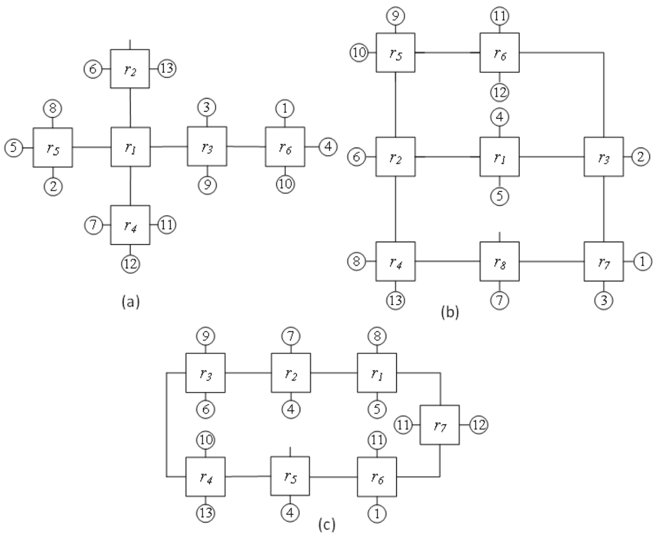

Fig. 4. Three example topologies (a) non-fault-tolerant, (b) fault-tolerant, and (c) ring. The CFG in Fig. 3 is mapped on each topology using simulated-annealing-based [11] mapping algorithm.

throughout this paper for the topology generation and FPGA implementation.

Inputs of the topology generation step are the CFG of the given application and a set of routers. In our method, we used identical routers with the same number of ports to better deal with design complexity. Fault-tolerant topology generation (FTTG) starts with non-fault-tolerant topology generation (N-FTTG) step. N-FTTG determines the minimum number of routers, r_{min}, based on the number of application nodes, router ports p, and $r_{min}-1$ links, which is the least link number to have fully connected network topology. It then randomly connects the routers one by one to each other until all routers are fully connected. N-FTTG iteratively generates new topologies and compares the new ones with the previously generated topologies if they are the same (i.e., isomorphic). Using this random procedure, N-FTTG generates several non-isomorphic topologies. Fig. 4(a) gives an example N-FTTG generated topology for the CFG given in Fig. 3.

The example in Fig. 4(a) demonstrates that N-FTTG topology has only one path between any router pairs. In order to make them fault-tolerant for single permanent link failures, all router pairs must have at least two routing paths. To guarantee this, we add additional network resources (i.e., links or routers-links). In some cases, there can be some empty ports in N-FTTG topology that can be used to add some more links to the network. In such a case, the topology can be made fault-tolerant. The topologies in Fig. 2 illustrate this situation. In this figure, the non-fault-tolerant topology on the left is made fault-tolerant by only adding a link between r_1 and r_2. However, adding only links to a N-FTTG topology may not convert it to a FTTG topology since there may not be enough empty ports. To increase the number of empty ports for the additional links, we should add more routers to the N-FTTG topology. Thus, we can have different numbers of routers for different FTTG topologies.

In [11], the number of routers for FTTG topologies are selected between r_{min} and $r_{max} = r_{min} + \log_2 r_{min}$. FTTG algorithm starts randomly generating non-fault-tolerant topologies with r_{min} routers. Then, it adds additional links to make the topology fault-tolerant. All the network components of the network must be on at least one cycle to have at least two alternative paths. When FTTG adds a link to the topology at hand, it selects two empty ports that minimizes the average path length (APL) of the topology. The reason for doing so is to increase the possibility of minimizing the energy consumption. If the data from one node to the other travels on a short path on the network, the circuit switching becomes small, resulting in low dynamic energy consumption.

FTTG adds links to all empty ports to increase the cycles on the topology and to minimize the APL. Similar to N-FTTG algorithm, FTTG also checks each generated topology with the ones in the topology library. Each topology added to the library must be non-isomorphic. Finally, the algorithm returns a topology library with different numbers of routers and APL values. The designer can select the one that fulfills the design needs. In Fig. 4(b), we give our selected FTTG generated topology for our FPGA implementation.

Ring topology is the least router consuming regular fault-tolerant topology alternative to our irregular topologies. Thus, we included it in our comparisons. Given the number of application nodes and the number of ports for the routers, we can have only one ring topology. In Fig. 4(c), we give the ring topology for our application.

After generating a topology, we map the application nodes onto it under bandwidth constraints. Our objective function is dynamic energy minimization of the design. We calculate the energy consumption by simply adding the energy consumption of the traveling bits over the routers and links. Fig. 4 shows the mappings we use in our FPGA implementation for N-FTTG, FTTG, and ring topologies, respectively.

Routing mechanism is the last phase in our method. We use the shortest path routing for three topology types. The routing information for the packets are stored in the routing tables of the routers on their paths. However, N-FTTG and ring topologies must use more than one routing table since they use the alternative routing table in case of a link failure. A routing table is powered up externally from the pins of the chip. If the routing table options increase, both the area of the router and the external pin count increase. Thus, we try to minimize the number of routing tables that cover all the single link failures.

TABLE I
PATH TABLE (PT0) FOR THE DEFAULT ROUTING OF FTTG IN FIG. 4(B).

Source Node	Destination Node	Routers in the Path
1	2	$7 \rightarrow 3$
1	3	7
1	9	$7 \rightarrow 3 \rightarrow 6 \rightarrow 5$
2	5	$3 \rightarrow 1$
3	4	$7 \rightarrow 3 \rightarrow 1$
4	5	1
5	6	$1 \rightarrow 2$
6	7	$2 \rightarrow 4 \rightarrow 8$
6	8	$2 \rightarrow 4$
9	10	5
10	13	$5 \rightarrow 2 \rightarrow 4$
11	12	6
12	13	$6 \rightarrow 5 \rightarrow 2 \rightarrow 4$

TABLE II
PTS FOR FOUR ALTERNATIVE ROUTINGS. WE ONLY GIVE THE ROWS OF PTS THAT ARE DIFFERENT THAN THE DEFAULT PT GIVEN IN TABLE I.

PTi	Links Covered	Source Node	Dest. Node	Routers in the Path
PT1	$l_{3,6}$	1	9	$7 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 5$
	$l_{4,8}$	6	7	$2 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 8$
PT2		6	7	$2 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 8$
	$l_{2,4}$	6	8	$2 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 4$
	$l_{2,5}$	10	13	$5 \rightarrow 6 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 4$
		12	13	$6 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 4$
PT3	$l_{1,2}$	1	9	$7 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 5$
	$l_{5,6}$	5	6	$1 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 4 \rightarrow 2$
		12	13	$6 \rightarrow 3 \rightarrow 7 \rightarrow 8 \rightarrow 4$
PT4		1	2	$7 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 6 \rightarrow 3$
	$l_{1,3}$	1	9	$7 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 5$
	$l_{3,7}$	2	5	$3 \rightarrow 6 \rightarrow 5 \rightarrow 2 \rightarrow 1$
		3	4	$7 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

In Table I, we show the path table for the default routing for the topology in Fig. 4(b). We use the shortest path routing as the routing algorithm to generate path tables. When we generate the alternative routing tables, we try to cover as many link failures as possible to minimize the number of routing tables. To achieve this goal, we remove one uncovered link from each cycle in the topology as long as the topology is fully connected. Obviously, the energy consumption of the alternative routings will be higher than the default one since some of the packets will use longer paths than the default case. Thus, when we select the links to remove from the default topology, we select the ones with the minimum utilization to keep the energy consumption increase as small as possible.

Table II shows the four alternative routings, each covering two link failures of our FTTG topology. In this table, we only show the changes made on the default path table. The table also shows the covered links by this routing (i.e., the routing can be used even if these links are faulty.) This routing table methodology decreases the number of routing tables from ten (default routing plus the number of links on the topology) to five, which results in a significant area savings on the chip. Additionally, the external routing table selection pins decreases from four to three.

III. FPGA IMPLEMENTATION

The main network component of a NoC architecture is the router. Thus, we first implemented a simple router architecture using Verilog language. Since our goal in FPGA implementation is to compare the fault-tolerant topology with a non-fault-tolerant version and a regular fault-tolerant alternative, we tried to design the router as simple as possible with a minimum FPGA components. In our FPGA evaluation, we use the topologies given in Fig. 4 and all these topologies use only four-port routers. Therefore, we designed a simple four-port router. We give the logical design schema of our router in Fig. 5.

The router is composed of three main components: (1) FIFOs, (2) routing logic, and (3) a look-up table (LUT). A FIFO writes the incoming packet into its queue whenever a new data is available in its input port. We fixed the size of the FIFO queues to eight packets. When we generate the topology with router modules, the unused FIFOs are removed from the design in an attempt to minimize the chip area. FIFOs are connected to a multiplexer (MUX) and each FIFO is selected in a round robin (RR) fashion. The data taken from the MUX outputs is fed to the LUT and the routing logic.

LUT behaves as a ROM and stores the path table information as the routing table. Each line of the LUT stores the ID of the packet and its output destination. Since each router has its own packets and routing information, the LUTs are hard-coded (i.e., LUTs cannot be changed after the chip fabrication.) For simplicity, we decided to make the length of the LUTs to be the number of edges in the CFG of given application, which is the maximum possible packet size for a router. For example, our Mp3 benchmark has 13 edges and the LUTs of the routers in our implementation has 13 lines. The size of the LUTs can be optimized to have smaller area and energy consumption, however; this optimization problem is not in the scope of this work and it can be a topic of another study. The ID bits in a packet is directly related to the number of edges (i.e., the number of packets). In our case, we will need four bits to represent each packet ID since $log_2\lceil 13 \rceil = 4$. We can represent four output ports with two bits. Then, the total bits in each line of the LUT becomes six bits. As a summary, in our implementation, the size of the LUT for a router is fixed and it is 13×6.

After the output port bits are selected from the LUT table, the routing logic simply enables the corresponding output port (i.e., the tristate buffers) as shown in Fig. 5 and the data is sent to its next destination from the selected output port.

The packet size of a router is generic and we can select any reasonable size. In our Mp3 encoder application, we decided to make it 32-bits. Although this small packet size leads to a high data transfer between nodes of Mp3 encoder application, it does not affect the evaluation of different topologies. Since we use four bits for representing packet IDs, each packet carries 28 bits of data. Thus, we can calculate the number of packets between communicating nodes. Table III summarizes the numbers of packets that are sent between different nodes

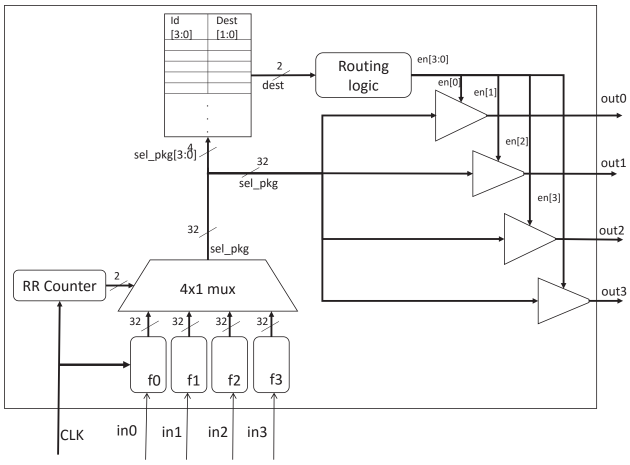

Fig. 5. Single router schema.

TABLE III
NUMBER OF KILO PACKETS SENT IN A SECOND FOR MP3 ENCODER APPLICATION.

Source Node	Destination Node	Packets (Kilo)
1	9	1
1	3	145
1	2	74
3	4	17
4	5	35
2	5	35
5	6	31
6	8	6
6	7	5
9	10	74
10	13	1
12	13	17
11	12	140

of Mp3 encoder application.

After implementing the routers, we connect them together to generate our target topology. In our implementation, we generated three topologies given in Fig. 4. However, we do not implement the nodes of the Mp3 encoder. We connect the corresponding inputs to the routers to emulate the Mp3 encoder application. By doing so, we only calculate the area and energy consumption of the NoC topology.

IV. EXPERIMENTAL RESULTS

As we stated in the previous sections, in the experiments, we compare three different topology types, namely; N-FTTG, FTTG, and ring, based on area and energy consumptions. N-FTTG topology is optimized for area and energy consumption and does not consider fault-tolerance. We give this topology in Fig. 4(a). FTTG brings fault-tolerance by adding extra network components to N-FFTG topology as explained in [11]. FTTG also considers area and energy minimization. Among the generated FTTG topologies, we have selected the one with minimum energy consumption, given in Fig. 4(b). In this FTTG topology, there are two more routers and four more links than N-FTTG topology. The third topology type is the ring topology, which is the least area consuming regular alternative to FTTG topology. As our ring topology, we use the energy optimized ring topology given in Fig. 4(c). In the next subsections, we first give the experimental setup for our evaluations. We then show the area and energy consumption comparison results, respectively.

A. Experimental Setup

For our FPGA evaluations, we used Xilinx 14.7 [13] environment with WebPack license. For the simulations, we used Isim simulator, which is provided by Xilinx WebPack. We used Xilinx's XPower Analysis Tool to get energy consumption values of our designs. Simulation is done for one second and the signal activity is dumped to a VCD file. Map file and VCD file are fed to the XPower Analysis Tool to get the energy results of our design. Fig. 6 summarizes energy analysis steps. As the FPGA generation, we selected Xilinx Virtex-6 FPGAs, which is built on 40nm technology. The design critical paths are also verified to fit in 50MHz and 100Mhz clock frequencies, which are provided by the selected FPGA family.

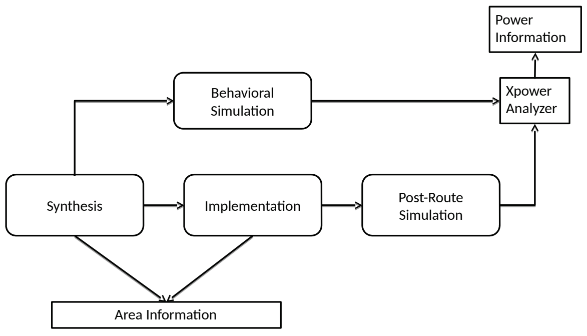

Fig. 6. FPGA test methodology flow chart.

B. Area Evaluations

Before we give the total area consumption of each topology, we first give the used logic components of the generic router module on an FPGA in Table IV. When we generate the topology, some of the input FIFOs are not used. Therefore, in the optimization step during synthesis, implementation steps of FPGA design would effect the final area of the architecture. Even if our router design has four ports, unused FIFOs are removed from the design. Based on the utilized components, we compare the area consumption of three topologies in Fig. 7. In this figure, we give the normalized area values with respect to the area of ring topology since the ring topology consumes the biggest area.

Area values are driven by the number of FIFOs in the NoC system. When we consider the number of ports and FIFOs used for NoC system, we see that area values increase proportionally. Even if the number of routers used in ring topology is less than FTTG topologies, ring has more FIFOs than FTTG. For instance, our FTTG topology has 26 FIFOs

TABLE IV
LOGIC COMPONENTS OF THE SINGLE ROUTER AFTER SYNTHESIS.

FPGA Logic Units	#
13x2-bit dual-port distributed ROM	2
8x32-bit dual-port distributed RAM	4
2-bit adder	1
3-bit up counter	8
Flip-Flops	6
3-bit comparator not equal	4
1-bit 4-to-1 multiplexer	37

while ring and N-FTTG have 27 and 23 FIFOs, respectively. Additionally, number of look-up tables, comparators, and multiplexers also vary and contributes to the area consumption of the design.

As a result, adding a fault-tolerance to the N-FTTG topology, the area of the topology increases by 11%. This area increase is mainly caused by the additional routing tables (LUTs) and FIFOs of the FTTG design. The area results of FPGA implementation shows a small differences with simulation results given in [11]. In this previous work, the FIFO optimizations and area increase of routing tables are ignored and resulted in only 2% area increase. This difference shows that the additional routing tables and FIFOs may increase the area around 9%. Additionally, simulation results given in [11] shows a 0.85% area increase against ring topology. However, FPGA implementation brings around 2% area reduction against ring due to FIFO optimization. In summary, FPGA-based area evaluation considers several aspect of the design and gives more reliable results than simulation-based evaluations.

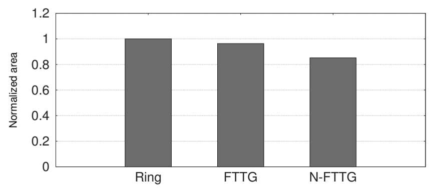

Fig. 7. Area comparison of ring, FTTG, and N-FTTG topologies. Areas are normalized with respect to area of ring topology.

C. Energy Evaluations

We have examined energy consumption of three topologies in two different aspects: Without router shot down and with router shot down. In the first set of experiments, we use the energy values taken from XPower Analyzer without considering a router shot down when the router is idle. We give the total energy consumption of each topology in Fig. 8. In this bar chart, we also show the energy consumption of each router separately. In this figure, FTTG(0) is the topology that uses default routing given in Table I (i.e., the routing without any link failures.) and each FTTG(i) uses the PT(i) given in Table II. As expected, when an alternative routing is used, the energy consumption value increases since the packets use longer routing path than the default one. When we compare the energy consumption of FTTG against ring and N-FTTG, we see small energy overhead. One reason for this increase is using more routers than the other two. Additionally, each router consumes energy even though they are idle. Thus, we conducted second set of experiments to show the effects of power shot down of routers.

For this analysis, we first calculated per packet energy consumption using XPower Analyzer which is found as 9.152 *nJ*. By multiplying the number of packets that travels through a single router with the unit energy per packet, we calculated the total energy consumption. Second set of experiments also shows that dynamic energy results are proportional to the switching activity. Table II shows the details of communication between nodes. Except FTTG(3), generated FTTG topologies provides better energy values ranging from 4% to 16%. However, FTTG(3) is (43%) and N-FTTG topologies are (24%) worse than ring topology. Since the topologies other than FTTG(0) are only used for faulty chips, some area and energy increase can be tolerable to recover the faulty chip rather than being disposed.

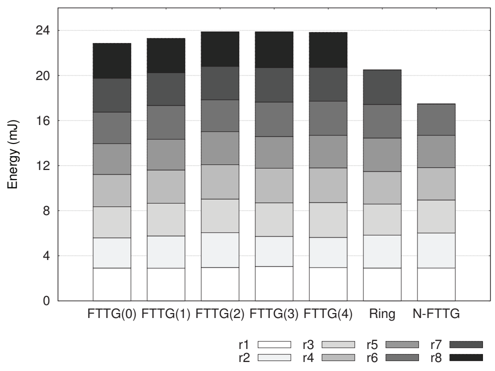

Fig. 8. Energy comparison of FTTG, ring, and N-FTTG topologies without router shut down.

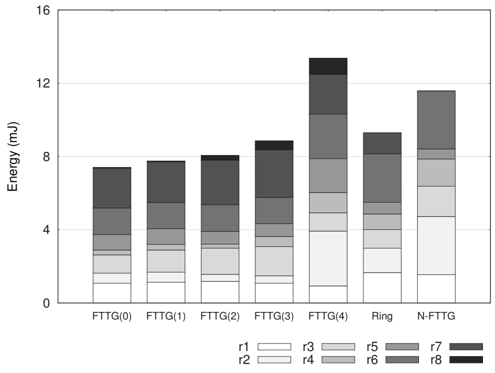

Fig. 9. Energy comparison of FTTG, ring, and N-FTTG topologies with router shut down.

V. Conclusion

Previous studies evaluates the fault-tolerant application specific NoC designs using only simulation techniques. In this study, we show the FPGA implementation of fault-tolerant irregular topology based application specific NoC design. As a case study, we implemented Mp3 encoder benchmark on non-fault-tolerant, fault-tolerant, and ring topologies. We compare three topology types based on chip area and energy consumption. The fault-tolerant topology gains the capability of fault-tolerance with 11% overhead against non-fault-tolerant topology. It also has better area and energy values than ring and better energy values than non-fault-tolerant topologies. The paper also shows that FPGA implementation shows very small differences from simulation outputs, showing that both evaluations gives similar results.

Acknowledgment

This work is supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under grant number 112E360 and EU COST Action IC1204 - TRUDEVICE.

References

[1] W. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks,” in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684–689.

[2] L. Benini and G. De Micheli, “Networks on chips: a new soc paradigm,” *Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[3] K. Srinivasan, K. Chatha, and G. Konjevod, “Linear programming based techniques for synthesis of network-on-chip architectures,” in *Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings. IEEE International Conference on*, Oct 2004, pp. 422–429.

[4] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, “A highly resilient routing algorithm for fault-tolerant nocs,” in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE ’09.*, April 2009, pp. 21–26.

[5] M. Gomez, J. Duato, J. Flich, P. Lopez, A. Robles, N. Nordbotten, O. Lysne, and T. Skeie, “An efficient fault-tolerant routing methodology for meshes and tori,” *Computer Architecture Letters*, vol. 3, no. 1, pp. 3–3, January 2004.

[6] V. Puente, J. Gregorio, F. Vallejo, and R. Beivide, “Immunet: a cheap and robust fault-tolerant packet routing mechanism,” in *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, June 2004, pp. 198–209.

[7] S. Rodrigo, J. Flich, J. Duato, and M. Hummel, “Efficient unicast and multicast support for cmps,” in *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*, Nov 2008, pp. 364–375.

[8] K.-C. Chang and T. Chen, “Low-power algorithm for automatic topology generation for application-specific networks on chips,” *Computers Digital Techniques, IET*, vol. 2, no. 3, pp. 239–249, May 2008.

[9] S. Tosun, Y. Ar, and S. Ozdemir, “Application-specific topology generation algorithms for network-on-chip design,” *Computers Digital Techniques, IET*, vol. 6, no. 5, pp. 318–333, September 2012.

[10] S. Tosun, V. Ajabshir, O. Mercanoglu, and O. Ozturk, “Fault-tolerant irregular topology design method for network-on-chips,” in *Digital System Design (DSD), 2014 17th Euromicro Conference on*, Aug 2014, pp. 631–634.

[11] ——, “Fault-tolerant topology generation method for application-specific network-on-chips,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, no. 9, pp. 1495–1508, Sept 2015.

[12] V. Ajabshir and S. Tosun, “Fault-tolerant routing for irregular-topology-based network-on-chips,” in *Computing and Networking (CANDAR), 2014 Second International Symposium on*, Dec 2014, pp. 123–129.

[13] “Xilinx ise design suite,” http://www.xilinx.com/products/design-tools/ise-design-suite.html, accessed: 2015-09-30.