Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200102

Crossbar Implementation with Partial Reconfiguration for Stream Switching Applications on an FPGA

Yuichi KAWAMATA [a], Tomohiro KIDA [a], Yuichiro SHIBATA [a] and Kentaro SANO [b]
[a] *Graduate School of Engineering, Nagasaki University, Japan*
[b] *Riken Center for Computational Science, Japan*

Abstract. In this paper, we propose a network crossbar implementation using partial reconfiguration of an FPGA in a multi-FPGA cluster computing system. With a proposed framework, inter-FPGA network routing can be changed by reconfiguring the crossbar module by a partial reconfiguration mechanism. The purpose of this paper is to compare ordinary crossbar circuits and partial reconfiguration crossbar circuits, in terms of resource usage and the maximum operating frequency. As a result, by using partial reconfiguration, the maximum operating frequency is improved by 1.6 times while reducing required ALM resources by 13%, a proper bus sizes for a crossbar are selected.

Keywords. FPGA, Partial Reconfiguration, crossbar, Arria10

1. Introduction

Field Programmable Gate Arrays (FPGAs) have been attracting attention as a platform of a power-efficient custom computing because FPGAs can construct optimal data paths according to each application. Especially in recent years, with the improvement of the integration degree, FPGAs have been equipped with floating-point arithmetic cores and have devised wiring architecture to improve the operating frequency. Expectations are rising for applications in the high performance computing field [1][2].

Stencil calculation, which repeatedly applies arithmetic processing with data references of the same shape to data arranged in a grid, is a common design pattern used in various scientific calculations, and it is known that an FPGA-based system able to work efficiently in a streamwise framework [3][4][5]. Also, by connecting the operation pipelines in series and increasing the number of operations per memory access, it is able to improve the operation performance without increasing memory bandwidth required for DRAM[6]. Therefore, even in constructing a parallel system in which multiple FPGAs are interconnected, it is promising because the performance can be scaled without being restricted by the connection bandwidth between the FPGAs [7].

In order to build a multi-FPGA computing system, a communication mechanism is needed to exchange data between FPGAs. Many research projects have been carried out to extend the switch mechanism for network on chip (NoC) used for inter-module communication inside the chip, and to connect FPGAs [8] [9] [10] [11] [12]. These NoC-

derived switch mechanisms are mainly based on packet-based routing and have excellent flexibility. On the other hand, in multi-FPGA cluster computing systems that perform stream-based processing, depending on the application, it is not always necessarily required to route input packets to different destinations in a short time, as shown in the left figure of Figure 1. As shown in the right figure, it just needs to support routing as a stream for some amount of data.

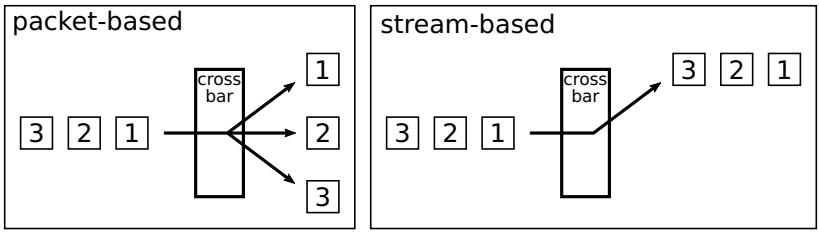

Figure 1. Comparison of routing

Since a general-purpose crossbar that can route any packets to any destinations is implemented in FPGAs as a set of multiplexers, it is thought that implementation efficiency decreases in terms of resource usage and frequency as the number of input / output bits increases. However, it is inflexible that only fixed routing can be performed for each application. That is, there is a trade-off between flexibility and performance / efficiency in crossbars for multi-FPGA systems, and there can be various design options.

In this paper, we propose a stream-based network crossbar that uses partial reconfiguration technology of FPGAs for path switching. Partial reconfiguration is a technology to change a specific circuit of FPGAs while other circuits in operation. Crossbars that use partial reconfiguration are expected to reduce resource usage and improve the maximum operating frequency, although their dynamic flexibility is limited compared to conventional general-purpose crossbars. In order to clarify these trade-off relationships and to evaluate the effect of partial reconfiguration for the crossbar, the conventional crossbar circuit is compared with the partial-reconfiguration-based crossbar in terms of resource usage and the maximum operating frequency, and the reconfiguration time required for partial reconfiguration is also evaluated. In this paper, assuming a multi-FPGA system with a two-dimensional torus as shown in Figure 2, we evaluate and verify a crossbar with a total of 5 inputs including 4 external inputs, and 1 internal input and 5 outputs as shown in Figure 3.

2. Partial Reconfiguration under Intel Environment

In this work, Partial Reconfiguration (PR) is performed with Intel FPGA Arria 10. In the experiment, JTAG is used to transfer configuration data and partial reconfiguration data. We use Intel's Quartus Prime Version 18.1 Pro Edition as a development environment.

2.1. Design procedure for partial reconfiguration design

The design procedure of the partial reconfiguration circuit in the Intel FPGA is as follows [13].

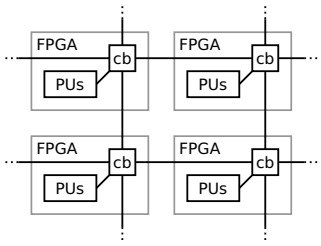

Figure 2. 2D torus consisting of FPGA

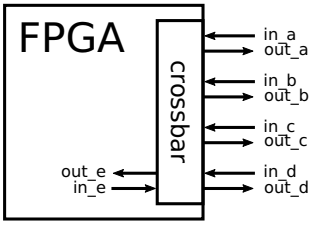

Figure 3. Example of stream connection

1. Circuit design and description
2. Creating Design Partition and LogicLock region
3. Allocating Placement region and Routing region
4. Adding the Partial Reconfiguration Controller IP Core
5. Creating Revisions
6. Compiling the Base Revision
7. Generating a qdb file
8. Compiling each persona

Details of the above procedures are described in the following.

2.1.1. Creating Design Partition and LogicLock region

Design Partition is created from partially reconfigured modules. From the created Design Partition, Quartus's LogicLock function [14] is used to fix the placement of the partially reconfigured modules. This fixed area is called LogicLock region. Since the designated modules are placed and routed only at the designated locations, the degree of freedom of placement and routing is reduced in the partial reconfiguration module, so the maximum operating frequency may be reduced compared to the case where the placement is not fixed. In this paper, we create LogicLock region and fix the crossbar module. And this design is compared with the ordinarily designed circuit and the partial reconfiguration circuit.

2.1.2. Allocating Placement region and Routing region

The LogicLock region has Placement region and Routing region. So we fix the location and size of the Placement region and the Routing region. The Placement region is a region for modules to be partially reconfigured, and the Routing region is a region for arranging paths connecting to the Placement region.

2.1.3. Adding the Partial Reconfiguration Controller IP Core

PR control mechanism creates and uses IP Core in Quartus. When performing partial reconfiguration, only one PR Controller IP is required on the FPGA [15]. The interface of PR Controller IP is shown in Figure 4.

In Figure 4 *nreset* is the asynchronous reset signal input for the PR Controller IP Core. *clk* is a clock for PR Controller IP Core, supporting up to 100 MHz. It begins a partial reconfiguration event when the *pr_start* signal changes from 0 to 1. It receives the next *pr_start* signal only when the *freeze* signal is low (0). It inputs configuration data

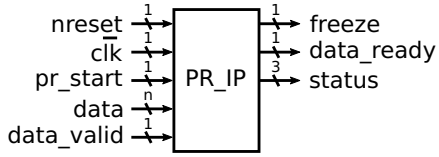

Figure 4. PR Controller IP

for partial reconfiguration to the *data* port. Data width can be selected from 1, 8, 16, and 32 bits. *data_valid* indicates that valid data has been input to the *data* port. *freeze* outputs a high (1) signal during partial reconfiguration. *data_ready* indicates that the *data* port is ready to receive data. *status* is a 3-bit error output indicating the status of partial reconfiguration event. When performs partial reconfiguration via the JTAG interface, the PR Controller IP exchanges signals with the JTAG interface, so insertion of *clk*, *pr_start*, *data*, *data_valid*, and *data_ready* values into these is ignored.

2.1.4. Compiling Revisions and Personas

In the partial reconfiguration design flow, Quartus uses project revision format. There are two versions, Base and Persona Implementation. The Base revision is designed for the entire circuit, and the Persona Implementation revision is designed for the partially reconfigured module. Usually there are only one Base revision and multiple Persona Implementation revisions. The Base revision is compiled first. This compilation operation includes logic synthesis, placement and routing, timing analysis, configuration data generation, etc. The compiled base revision is written out into a qdb database file. The functional module to be partially reconfigured is called persona. Partial reconfiguration data using the qdb file created in the previous step is generated.

3. Evaluated Implementation

In this paper, in addition to the usual design, the LogicLock (hereinafter LL) circuit with fixed area for crossbar module and Partial Reconfiguration (hereinafter PR) circuit which partially reconfigures the crossbar module were created, as shown in Figure 5.

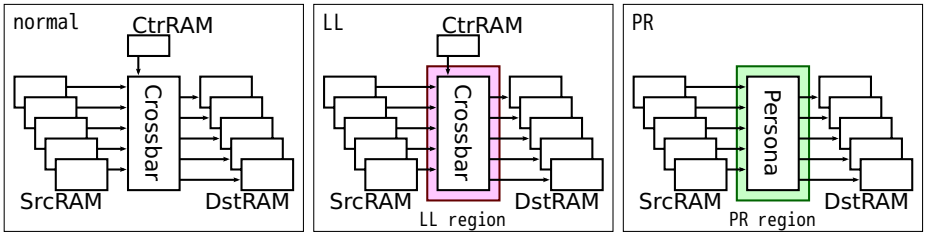

Figure 5. Outline of each circuit

In the FPGA design created this time, stream data is generated from SrcRAM, which is M20K Embedded Memory, and stream data is stored in DstRAM through the crossbar. Also, the crossbar is controlled from CtrRAM.

All crossbar modules other than PR circuit mount full crossbars. The bit widths for stream data of 8 bits, 32 bits, 64 bits, 256 bits, 1024 bits and 4096 bits are evaluated.

3.1. Implementation of full crossbar

The implementation of the full crossbar is shown in Figure 6. It has 5 inputs and 5

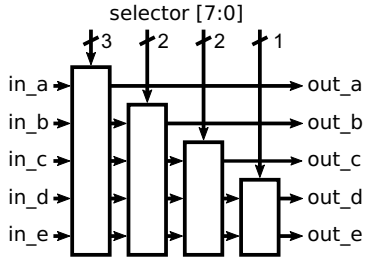

Figure 6. Implementation of full crossbar

outputs, and connection of the inputs and outputs is changed by the 8 bit selection line. The upper 3 bits select 1 output from the leftmost crossbar, and the other outputs are connected to the crossbar on the right without changing the order. In this way, 8-bit connection lines are divided into 3 bits, 2 bits, 2 bits and 1 bit, used as selection lines for each crossbar. In this structure, multiple inputs cannot be connected to a same output.

3.2. persona

Three personas (ST, RT, and X) shown in Figure 7 are evaluated. Since the PR crossbar

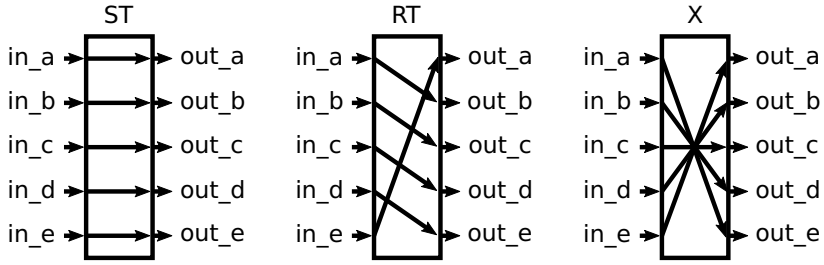

Figure 7. Implemented personas

circuit changes the routing by exchanging the persona, the selection line for crossbar control is not necessary. Therefore, the selection line is not implemented in the PR circuit.

In persona ST, inputs and outputs are connected in the same order. Applying this persona in all FPGAs will make each FPGA independent. In persona RT, the inputs are connected to the output next to that in persona ST. The bottom input is connected to the top output. Considering Figure 2, a system as shown in Figure 8 can be configured. In persona X, the inputs and outputs are connected in reverse order.

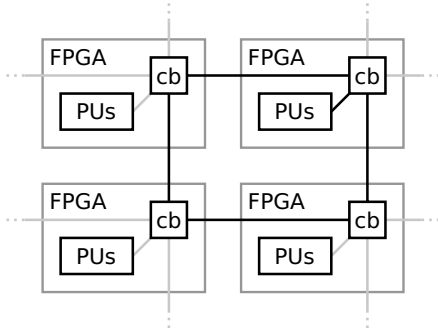

Figure 8. Example of Persona RT routing

3.3. Embedded Memory

We used M20K to store bit stream data. By using M20K as single port RAM, In-System Memory Content Editor can be used to read from / write to the RAM. In the evaluation experiments, the circuit operation was verified by writing data to SrcRAM and reading data from DstRAM via In-System Memory Content Editor. The number of words is set to 8 for all the crossbar designs.

3.4. Reconfiguration area

In the evaluation, the LL circuit and PR circuit with the same bus bits were implemented in the same area position, with setting the same area size. The LogicLock region and the partial reconfiguration area were set as shown in Table 1. The area size was set to

Table 1. Comparison of each area

	Width	Height	Origin
8bit LL circuit	10	10	X88_Y8
8bit PR circuit	10	10	X88_Y8
32bit LL circuit	10	10	X88_Y8
32bit PR circuit	10	10	X88_Y8
64bit LL circuit	10	10	X88_Y8
64bit PR circuit	10	10	X88_Y8
256bit LL circuit	10	30	X88_Y8
256bit PR circuit	10	30	X88_Y8
1024bit LL circuit	10	110	X88_Y8
1024bit PR circuit	10	110	X88_Y8
4096bit LL circuit	28	210	X35_Y11
4096bit PR circuit	28	210	X35_Y11

10×10 for circuits up to 64 bits. For 256 bits and 1024 bits, since it was not possible to implement in 10×10 area, the area was expanded in the Y direction. For 4096 bits, the area was expanded in both X and Y directions, and the position of the reference point (Origin) was also changed so that the area could be set.

4. Evaluation and Consideration

We evaluate and discuss the above mentioned circuits. The 1024 bit RT, 4096 bit RT and 4096 bit X can not be implemented because they can not be placed and routed. The evaluation environment is shown below.

- FPGA : Intel Arria10 10AX115N2F45E1SG FPGA
- CPU : Intel Core(TM) i7-8700K
- MEM : DDR4 16GB
- OS : CentOS 7.5.

4.1. Maximum operating frequency

Figure 9 and Table 2 show the maximum operating frequency of the circuit evaluated this time.

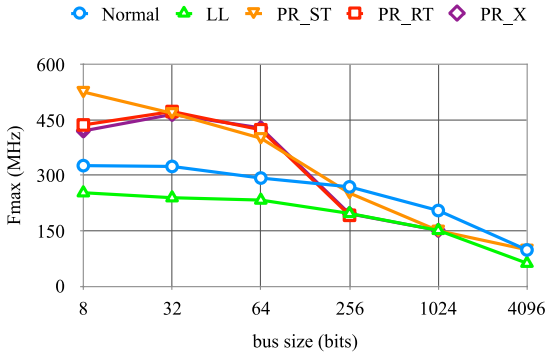

Figure 9. Maximum operating frequency comparison

Table 2. Maximum operating frequency (MHz)

bus size (bits)	8	32	64	256	1024	4096
normal circuit	326.26	324.04	292.65	268.89	204.79	98.12
LL circuit	252.91	239.41	233.05	196.89	151.01	62.25
PR circuit ST	525.49	467.63	400.48	251.95	150.26	98.73
PR circuit RT	436.30	472.59	423.19	191.86	-	-
PR circuit X	420.17	464.68	428.27	196.00	150.99	-

For 64 bits or less, the crossbar implemented by PR results in a higher maximum operating frequency than the normal circuit. On the other hand, for 256 bits or more, the frequency is slower than the normal circuit. The LL circuits are slower than the normal circuit for all the bits. A different PR circuit persona achieves a different maximum operating frequency, the order of the achieved frequency is also different depending on bit numbers. This is because a sufficient area can be allocated to the PR region for the designs up to 64 bits PR, and a high degree of freedom for routing is kept. For larger size designs, the freedom for routing is limited and thus the maximum frequency is degraded. Moreover, since the place where the data streams can access to the RAM, is fixed, the design without RAM can result in higher maximum operating frequency.

4.2. Amount of resource used

A comparison of the ALM usage of the crossbar modules evaluated this time is shown in Figure 10 and Table 3. Figure 10 plots the relative usage which is normalized to the

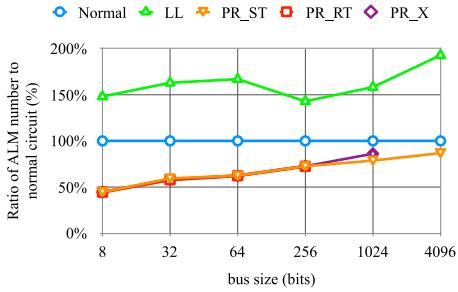

Figure 10. Comparison of ALM useage

Table 3. Resource usage of crossbar module (ALMs)

bus size(bits)	8	32	64	256	1024	4096
normal circuit	101	276	514	1911	6511	26686
LL circuit	150	450	858	2729	10300	51421
PR circuit ST	45	164	321	1386	5122	23168
PR circuit RT	45	160	321	1386	-	-
PR circuit X	45	160	320	1386	5605	-

amount of ALMs used in the normal circuit. The PR circuit was smaller than the normal circuit for all bit numbers, and the LL circuit was larger than the normal circuit. In the ALM usage for the PR crossbar module is about 45% of that for the normal circuit when the bus size is 8 bits. This ratio becomes higher as the bus size increases, and reaches approximately 87% at 4096 bits. On the other hand, differences in ALM usages due to change of personas for PR circuits are relative low, since there is only a difference in connection between inputs and outputs in the PR circuit.

Table 4 shows the amount of resources used for the entire FPGA design for evaluated designs. Even though the PR circuit includes a PR controller IP circuit in addition to

Table 4. Amount to Total resource use

	ALM	Register	RAM
32bit normal circuit	1138	1138	21
32bit LL circuit	1307	1149	21
32bit PR circuit ST	988	1109	20
4096bit normal circuit	47950	42961	2051
4096bit LL circuit	73038	42361	2051
4096bit PR circuit ST	41706	42912	2050

the normal circuit, the PR circuit is smaller than the normal circuit in terms of every resource. The usage of ALM is about 87% of the normal circuit when the bus size is 32 bits and 4096 bits.

4.3. Generated file size

Table 5 compares bit stream file sizes fort the evaluated designs. The sof (sram object

Table 5. File size (MB) and reconfiguration time (sec)

	sof	rbf : ST	rbf : RT	rbf : X	Configuration	PR :ST	PR : RT	PR : X
8bit normal circuit	36	-	-	-	21.24	-	-	-
8bit LL circuit	36	-	-	-	21.02	-	-	-
8bit PR circuit	36	5.9	5.9	5.9	21.12	7.68	7.69	7.62
32bit normal circuit	36	-	-	-	20.99	-	-	-
32bit LL circuit	36	-	-	-	21.08	-	-	-
32bit PR circuit	36	6.1	6.1	6.1	21.12	7.93	8.23	7.70
64bit normal circuit	36	-	-	-	21.05	-	-	-
64bit LL circuit	36	-	-	-	21.21	-	-	-
64bit PR circuit	36	6.1	6.1	6.1	21.30	7.95	7.86	7.84
256bit normal circuit	36	-	-	-	21.18	-	-	-
256bit LL circuit	36	-	-	-	21.28	-	-	-
256bit PR circuit	36	17	16	17	21.26	18.20	19.00	19.06
1024bit normal circuit	36	-	-	-	21.25	-	-	-
1024bit LL circuit	36	-	-	-	21.28	-	-	-
1024bit PR circuit	36	54	-	54	21.37	56.65	-	56.46
4096bit normal circuit	36	-	-	-	21.36	-	-	-
4096bit LL circuit	36	-	-	-	21.36	-	-	-
4096bit PR circuit	36	114	-	-	21.36	163.34	-	-

file) is used for entire configuration, and the rbf (raw binary file) is used for partial reconfiguration. Therefore, rbf is not generated for normal circuits and LL circuits. The rbf file sizes for designs of 256 bits or more are larger than those for the designs up to 64 bits, probably due to a larger PR region. Even with the same area, the file size is larger for 32 bits and 64 bits compared to the design with 8 bits. Moreover, difference in rbf file size was shown due to change of personas even for the same bus size. On the other hand, all designs have the same size of sof. This evaluation results mean that crossbars with a wide bus size require a large area and large on-chip memory capacity to store the bit stream data.

4.4. Reconfiguration time

A comparison of the reconfiguration time via JTAG interface is shown in Table 5. The data in the table were obtained as an average of 5 measurement results. The measured values include not only circuit reconfiguration time but also a startup overhead of the Quartust tool. There is no significant difference in the configuration time as well as sof size. On the other hand, in partial reconfiguration, the increase in the reconfiguration time was observed after 256 bits, where the rbf size is large. When the bus size is 64 bits, the partial reconfiguration time is about 37% of the normal configuration time, and it increases to 86%, 265%, and 765% when it is 256 bits, 1024 bits, and 4096 bits, respectively.

Faster partial reconfiguration is possible by using internal memory. If partial reconfiguration is performed at 3.2 Gbps, which is the theoretical maximum performance of PR controller IP, it can be estimated that the time required to change a 5.9 MB of an 8-bit crossbar module persona is about 15 milliseconds. In addition, the 4096-bit crossbar persona, which requires the largest rbf size of 115 MB among the evaluated designs in this time, can be theoretically reconfigured in about 285 milliseconds.

5. Conclusion

In this paper, we described a crossbar for stream data using partial reconfiguration. The number of required ALMs can be reduced by 13% when the bus size is 32 bits and 4096 bits, compared to a usual full crossbar. This means that partial reconfiguration is effective for reducing the resources used in crossbar design. So it can be said that it is effective when you want to reduce the resources used in the design. The partial reconfiguration also improves the maximum operating frequency when a sufficient partial reconfiguration area is allocated for routing. For example, the maximum operating frequency is improved by 1.6 times for an 8-bit crossbar design. However, for wider bus size such as 256 bits, the maximum operating frequency degraded compared to the normal circuit. Partial reconfiguration with JTAG interface took several seconds in this experiment. Therefore, this approach is not suitable for applications that frequently change the routing. One of our important future work is to verify high-speed partial reconfiguration from internal memory.

References

[1] Kentaro Sano and Satoru Yamamoto. FPGA-Based Scalable and Power-Efficient Fluid Simulation using Floating-Point DSP Blocks. *IEEE Transactions on Parallel and Distributed Systems*, 28:2823–2837, 2017.

[2] Czajkowski, Tomasz and Aydonat, Utku and Denisenko, Dmitry and Freeman, John and Kinsner, Michael and Neto, David and Wong, Jason and Yiannacouras, Peter and P. Singh, Deshanand. From OpenCL to high-performance hardware on FPGAs. *Proceedings - 22nd International Conference on Field Programmable Logic and Applications, FPL 2012*, pages 531–534, 08 2012.

[3] Kentaro Sano. FPGA-Based Systolic Computational-Memory Array for Scalable Stencil Computations. In *High-Performance Computing Using FPGAs*, pages 279–303, 2013.

[4] Yukinori Sato, Yasushi Inoguchi, Wayne Luk, and Tadao Nakamura. Evaluating reconfigurable dataflow computing using the Himeno benchmark. In *Proc. ReConFig*, pages 1–7, 2012.

[5] Heiner Giefers, Christian Plessl, and Jens Förstner. Accelerating Finite Difference Time Domain Simulations with Reconfigurable Dataflow Computers. In *Proc. HEART*, pages 33–38, 2013.

[6] Keisuke Dohi, Koji Okina, Rie Soejima, Yuichiro Shibata, and Kiyoshi Oguri. Performance modeling of stencil computing on a stream-based FPGA accelerator for efficient d esign space exploration. *IEICE Transactions on Information and Systems*, 98–D(2):298–308, 2015.

[7] Kentaro Sano, Yoshiaki Hatsuda, and Satoru Yamamoto. Multi-fpga accelerator for scalable stencil computation with constant memory bandwidth. *Parallel and Distributed Systems, IEEE Transactions on*, 25:695–705, 03 2014.

[8] Sagar Latti. FPGA Implementation of Four Port Router for Network on Chip. *International Research Journal of Engineering and Technology (IRJET)*, 03:887–880, 2016.

[9] Andreas Ehliar, Dake Liu. *A Network on Chip based gigabit Ethernet router implemented on an FPGA*, volume 03. SSoCC, 2006.

[10] Andreas Ehliar, Dake Liu. An FPGA based open source Network-on-Chip architecture. *FPL*, 03:800–803, 2007.

[11] Roman Gindin, Israel Cidon, Idit Keidar. NoC-based FPGA: Architecture and routing. *NOCS*, pages 253–264, 2007.

[12] David Bafumba-Lokilo, Yvon Savaria, Jean-Pierre David. Generic Crossbar Network on Chip for FPGA MPSoCs. *IEEE*, pages 269–272, 2008.

[13] Intel. *AN 797: Partially Reconfiguring a Design on Intel Arria 10 GX FPGA Development Board*. Intel, 2018.

[14] Altera. *15. Analyzing and Optimizing the Design Floorplan with the Chip Planner*. Altera, 2013.

[15] Intel. *Intel Quartus Prime Pro Edition User Guide Partial Reconfiguration*. Intel, 2018.