TELKOMNIKA, Vol. 15, No. 2, June 2017, pp. 570 ∼ 577
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/telkomnika.v15.i2.5508

A Customized Reconfiguration Controller with Remote Direct ICAP Access for Dynamically Reconfigurable Platform

Tze Hon Tan*[1], Chia Yee Ooi[2], and Muhammad Nadzir Marsono*[3]
[1,3]Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310, Skudai, Johor, Malaysia
[2]Malaysia-Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia Kuala Lumpur, 54100 Kuala Lumpur, Malaysia
*Corresponding authors, e-mail: thtan5@live.utm.my, nadzir@fke.utm.my

Abstract

As FPGA dynamic partial reconfiguration getting into mainstream, design of reconfiguration controller becomes an active research. Most of the existing reconfiguration controllers support only the loading of partial bitstream into configuration memory without allowing user to access ICAP directly, which can provide user higher controllability over the reconfigurable device. This paper presents the architecture of a customized reconfiguration controller with remote direct ICAP access. Remote direct ICAP access allows user to configure or readback device internal registers, which offer user higher controllability over the reconfigurable device. Additionally, the proposed reconfiguration controller achieved at least 3.19 Gbps of reconfiguration throughput, which reduces the platform service downtime during dynamic partial reconfiguration. In order to reduce the latency and transmission overhead of remote functional update, partial bitstream is compressed with run-length encoding before transmission.

Keywords: *Dynamic partial reconfiguration, Self-reconfiguration, ICAP*

1. Introduction

In reconfigurable computing, dynamic partial reconfiguration is a vital feature, which enables updates in the field, improves area utilization and allows defect compensation. Dynamic partial reconfiguration provides a solution to update accelerator sub-circuits. With dynamic reconfiguration feature, FPGA becomes a viable solution for most existing hardware implementation of real-world applications that demand both processing power and flexibility, as FPGA has both performance advantages of ASIC solution and flexibility advantage of software solution. To apply changes and updates to the accelerators in the reconfigurable hardware, a new partial bitstream is loaded to the reconfigurable hardware at run time. This leads to the requirement of efficient reconfiguration controller to enable dynamic partial reconfiguration feature.

The Internal Configuration Access Port (ICAP) in Xilinx FPGA device allows the reconfiguration controller to be implemented within the chip. Hence, this provides opportunity for self-reconfiguration and single chip implementation option for designers. Maintaining distributed system for Internet-of-Things and cloud computing are big challenges for administrators if the remote update feature is absent in such systems [1]. Hence, design and implementation of a customized reconfiguration controller for remote dynamically reconfigurable platform becomes the primary focus in this research work.

There are a number of researches [2–8] that focused on developing controller to support dynamic partial reconfiguration in FPGA through ICAP. Since these controllers were customized and hardware based, high reconfiguration throughput was expected. However, utilization of shared bus architecture lowers reconfiguration throughput and increases overhead in internal transmission. Nabina et al. [7] implemented reconfiguration controller with partial bitstream compression. The compression algorithm is dictionary based with high compression ratio, where partial bistream is highly compressed.

Received December 28, 2016; Revised April 10, 2017; Accepted April 25, 2017

In this work, a customized reconfiguration controller with remote direct ICAP access is proposed to enable dynamic reconfiguration in NetFPGA for remote functional updates purposes. The developed Reconfiguration Controller achieves at least 3.19 Gbps of reconfiguration throughput, which reduces the platform service downtime during remote functional updates. In addition, the proposed controller supports remote direct ICAP access, which allows user to configure or readback device internal registers. With this feature, user has more observability and controllability on reconfigurable hardware internal operations, which include dynamic reconfiguration. In order to reduce transmission latency, partial bitstream is compressed losslessly before transmission. Even so, this proposed work prefers the run-length encoding compression scheme proposed by Liu et al. [6] after considering the hardware implementation efficiency in decompression. Instead of matching compression symbol width to ICAP bus width (32 bit) as in [6], the proposed architecture matches the compression symbol width to packet bus width (64 bit).

2. Architecture Overview

The system in FPGA consists of both Static Region and Partial Reconfigurable Region. Figure 1 illustrates the overview of the proposed high-level architecture. In the Static Region, there are Communication Manager to handle Ethernet packet transmission and Reconfiguration Controller to handle the loading of partial bitstream. Specifically, Communication Manager is responsible to perform lower layer tasks in the OSI model while Reconfiguration Controller is responsible to retrieve bitstream from Communication Manager and loading the bitstream to Reconfiguration Port. In Xilinx FPGA, the Reconfiguration Port is instantiated with the ICAP primitive, while the Reconfiguration Logic and Configuration Memory are not visible to the designer. The static region only undergoes the configuration process on the startup, while the partial reconfiguration region may undergo multiple reconfigurations during run-time. In the Partial Reconfigurable Region, the Partial Reconfigurable Module is linked to the Communication Manager for internal communication. Obviously, the performance achievement of the overall system relies on the design and implementation of Reconfiguration Controller and Communication Manager.

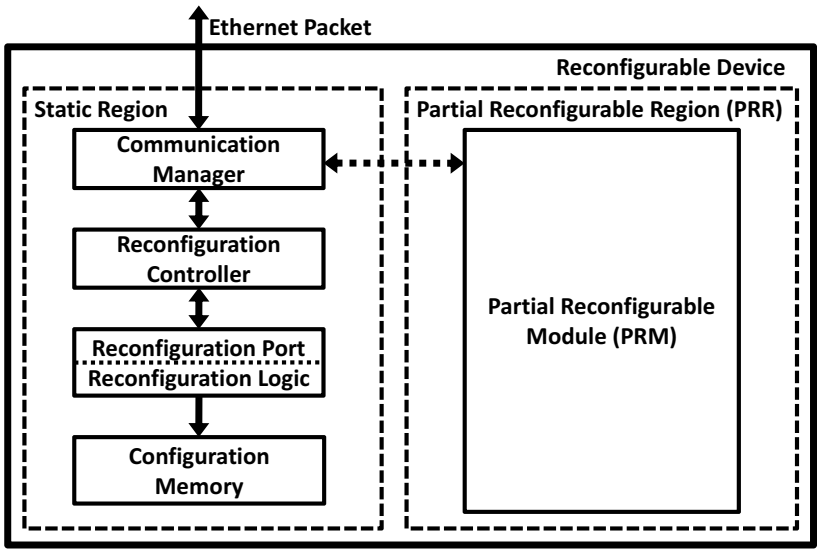

Figure 1. Proposed architecture for dynamically reconfigurable platform

3. Dynamic Partial Reconfiguration

There are several flows provided by Xilinx to support partial reconfiguration feature. Provided flows are modular method, difference-based method, small bit manipulation method, early

access method and partition-based method. This work uses partition-based method as it is the most recent consolidated flow which only available in new version of Xilinx ISE Design Suite or Xilinx Vivado Design Suite.

The major design flow includes design and synthesis on all functional modules, defining Partial Reconfigurable Module, defining design constraints and generating both bitstream and partial bitstream. The region of Partial Reconfigurable Region (PRR) is defined in a design constraint file and is used mainly during the Place and Route (PAR) process. In order to simplify the overall process, this work uses Xilinx PlanAhead to implement designs with partial reconfiguration as it provides user-friendly graphical user interface.

3.1. Reconfiguration Controller

The role of the Reconfiguration Controller is to retrieve partial bitstream from Communication Manager, which consists of Control Plane Packet Handler, Packet Type Classifiers, and Platform Manager. Figure 2 shows the implementation of customized Reconfiguration Controller. In the Reconfiguration Controller, the Bitstream Packet Handler extracts partial bitstream content in bitstream packet, acknowledges Terminal Client on the transmission and stores partial bitstream into SRAM through SRAM Interface. When the size of partial bitstreams are large, storing partial bistream in internal BRAM becomes impractical and may unnecessarily used up too much internal logic resources. Additionally, platform with on-the-fly remote dynamic partial reconfiguration poses higher risk on system failure especially when partial bistream transmission is interrupted. Moreover, such failure is not recoverable and the platform services become unavailable when the first segment of partial bitstream is loaded into the configuration memory while the other segments are in transmission. Therefore, this proposed architecture separates partial bitstream transmission from dynamic partial reconfiguration process.

Upon arrival of the last segment of partial bitstream, the Bitstream Packet Handler will notify Bitstream Loader on the status and the DPR Flow Controller will issue control signal to switch both Platform Manager and Packet Type Classifier into DPR mode. In the DPR mode, the DPR Flow Controller asserts the reset signal in Partial Reconfigurable Module to stop its operation. In this moment, the Bitstream Loader retrieves partial bistream from SRAM to load it into the configuration memory through both ICAP Interface and ICAP. Since the partial bistream is compressed using Run-Length Encoding (RLE) algorithm, the ICAP Interface inline decompresses the partial bitstream before loading it into ICAP. In order to verify the outcome of dynamic partial reconfiguration, the ICAP Interface proceeds with a readback sequence to retrieve the value of internal Status Register (STAT) after the last word of partial bitstream is loaded to the configuration memory.

The DPR Flow Controller will initialize the Partial Reconfigurable Module after ICAP Interface flags a reconfiguration success status. The initialization may take several cycles depending on application components initialization requirements. Once initialization is completed, the DPR Flow Controller will switch both Platform Manager and Packet Type Classifier back to normal mode and the Partial Reconfigurable Module is activated again. In case dynamic reconfiguration is unsuccessful, the DPR Flow Controller will feedback the outcome to Terminal Client and wait for another retry.

The partial bistream compression is implemented to reduce partial bitstream transmission overhead and temporary storage usage. This is because the area of Partial Reconfigurable Region is defined as large as possible so that complex application can fit into it. However, the partial bistream file size depends on the area definition of Partial Reconfigurable Region regardless of its logic utilization. Based on observation in partial bitstreams content, Partial Reconfigurable Region with low logic utilization has higher count of repetitive content, which can be compressed losslessly to reduce its original file size. In order to reduce logic resources needed for the implementation of decompression, Run-Length Encoding is used instead of Huffman encoding. In addition, the run value size is configured to match the bus size (64bit) so that the design complexity is minimized and the architecture becomes more efficient. The counter value (7 bit) of each run is stored at parity field (8 bit) along with respective run value (64 bit) at data field (64 bit), in both SRAM and Xilinx FIFO36_72 primitive. The remaining 1bit in parity field is used to indicate

the last word of a partial bitstream content.

Other than handling dynamic partial reconfiguration, the Reconfiguration Controller provides a way for user to access the ICAP directly. This feature aims to allow user to remotely configure or readback device internal registers through direct ICAP access. However, any packet used to access ICAP directly must follow a strict format because the packet contains the internal signal assertion of ICAP Interface. These signals are chip enable (CE) and read or write request (RW) from ICAP as well as last command (L). The assertion of CE and RW are similar to ICAP, where CE is in active low while logic '1' in RW indicates a read request. Table 1 provides snippet of a sample packet to readback Status Register (STAT). In the middle part of Table 1, the control signal is toggled from write request to read request and back to write request again, which results in the sequence of "000", "011", "010", "001" and "000".

There are 3 FIFOs used in the implementation of Reconfiguration Controller. These FIFOs act as the intermediate buffer and provide clock domain crossing between functional blocks. By splitting clock domain of ICAP from the platform, the ICAP can be clocked up to 100MHz, which is the maximum frequency provided by Xilinx [9]. The FIFOs with label "A" and "B" are implemented with Xilinx FIFO18_36 primitive and are used for direct user access to ICAP while FIFO with label "C" is implemented with Xilinx FIFO36_72 primitive and is used for dynamic partial reconfiguration.

3.2. Terminal Client

The Terminal Client reads and compressed the partial bitstream generated from Bitgen before transmits it to proposed platform through UDP/IP. The partial bistream is transmitted to the platform in multiple segments depending on user-specified packet size and the Terminal Client proceeds with packet retransmission whenever acknowledgment is not received within a specified timeframe. The bitstream packets are transmitted in bulk excluding the last segment, which is only transmitted to the platform after the acknowledgements of all other segments have been received. Figure 3 shows the bitstream packet format for dynamic partial reconfiguration. The Compression Header consists of pairs of run location (location of repeated content) and run length (repeated count) of the partial bitstream. For instance, the pair {0x02,0x0A} in Compression Header indicates that the third word of the Bitstream Content is a 11 times repeated word, where this word will be loaded 11 times into configuration memory. By default, each partial bitstream word without repetition (run length with value 0) will be loaded once into configuration memory, where this type of word is not compressed and tracked in Compression Header. Based on the packet definition, each word of Compression Header can store up to 4 pairs of run location and run length. For optimal performance, the value of segment size is configured to either 64 or 128, which are the values from power of two and are smaller than maximum transmission unit (MTU). Figure 4 illustrates the flow chart of implemented Terminal Client.

4. Platform Evaluation

Table 2 summarizes the logic resources required to implement proposed platform in NetFPGA 10G board that comes with Xilinx Virtex 5 (XC5VTX240T) FPGA, which provides 37440 slices and 324 BRAM. The proposed platform utilized 7297 slices (as reported by Xilinx ISE DS) out of 37440 available slices (less than 20% logic utilization), which left more than 80% of slices for Partial Reconfigurable Module implementation. However, the BRAM utilization of Static Region Module is almost 33.6% due to extensive use of FIFO in the design to hold packets and to buffer partial bitstream.

In order to verify the implemented platform experimentally, the partial reconfigurable module is dynamically reconfigured with various types of packet forwarding designs while network packets are injected to the platform. These network packets are captured and analyzed using Wireshark packet analyzer to verify the behavior of implemented platform.

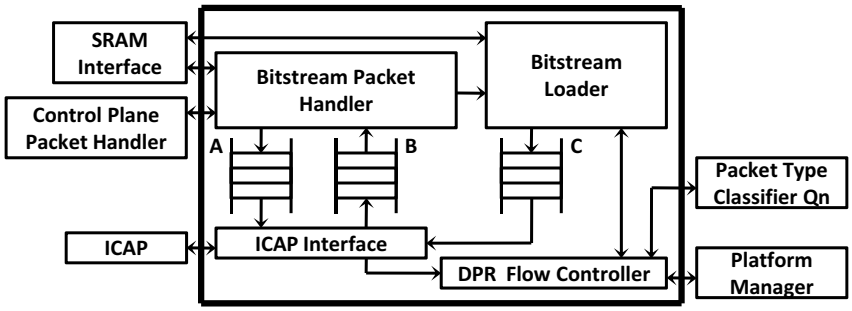

Figure 2. Implementation of the Reconfiguration Controller.

Table 1. Snippet of a sample packet to readback Status Register.

Packet content	Control signal {L,RW,CE}
FFFFFFFF_00000000	000
000000BB_00000000	000
11220044_00000000	000
FFFFFFFF_00000000	000
AA995566_00000000	000
20000000_00000000	000
2800E001_00000000	000
20000000_00000000	000

Packet content	Control signal {L,RW,CE}
20000000_00000000	000
00000000_03000000	011
00000000_02000000	010
00000000_01000000	001
30008001_00000000	000
0000000D_00000000	000
20000000_00000000	000
20000000_04000000	100

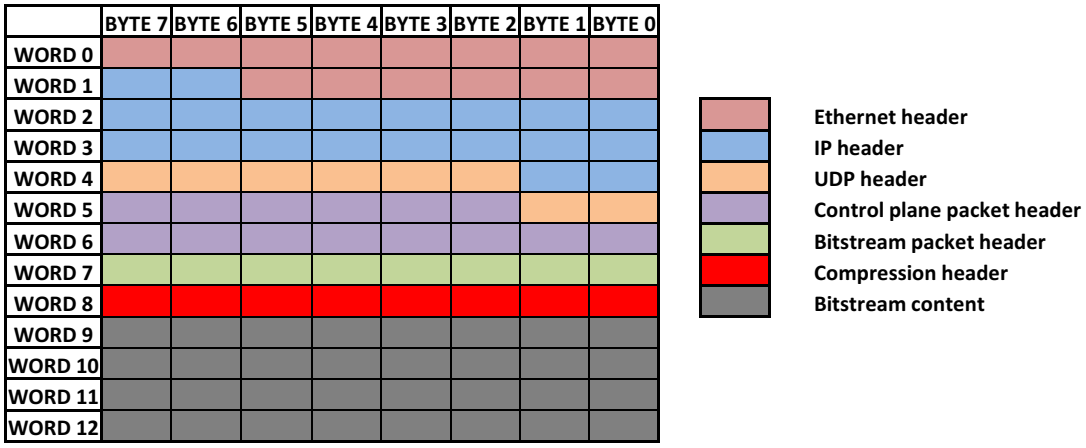

Figure 3. Bitstream packet format.

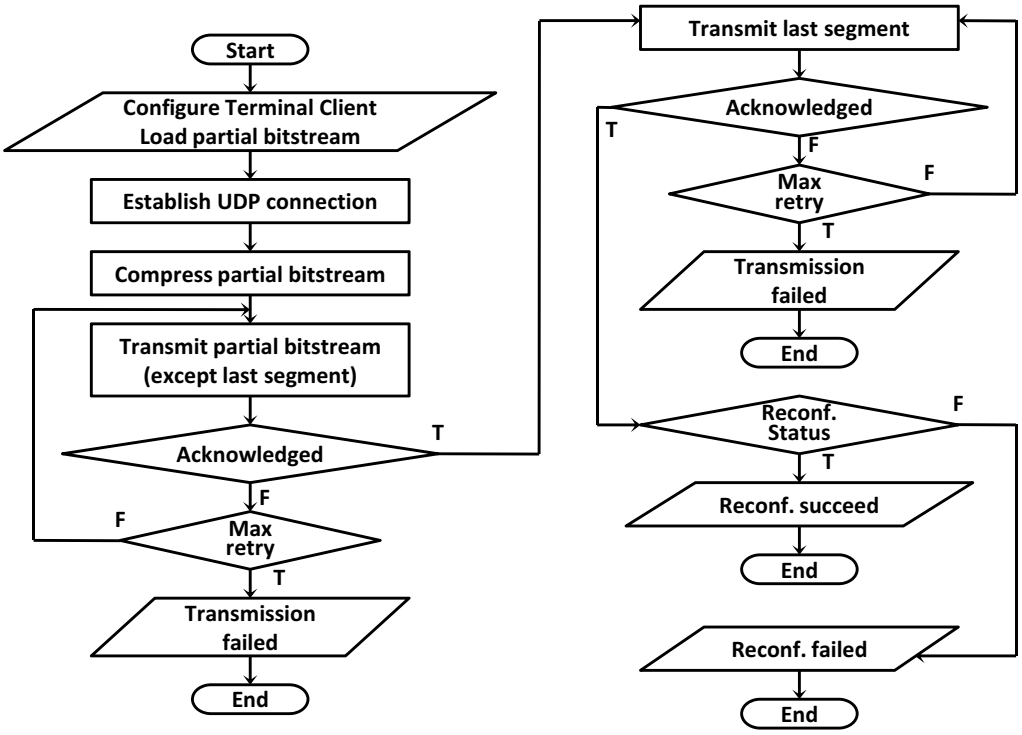

Figure 4. Flow chart of Terminal Client.

Table 2. Logic utilization of proposed architecture.

Resources type	Partial Reconfigurable Module	Static Regions Module	Total utilization	Available
Slice registers	4818	15150	19968	149760
Slice LUTs	3962	16185	20147	149760
Occupied slices	1830	7297	9127	37440
BRAM	32	109	141	324

Based on experimental evaluation, the Reconfiguration Controller achieves at least 3.19 Gbps of reconfiguration throughput. The partial bitstreams used have the same file size, which are 1,524,139 Bytes (~1.489 MB). In order to capture the time taken in dynamic partial reconfiguration, a timer is implemented and used in the Reconfiguration Controller. The loading of each partial bistream to configuration memory through Reconfiguration Controller takes 381,052 clock cycles with ICAP clocked at 100MHz, which result in 3.199855 Gbps of reconfiguration throughput. Ideally, the maximum achievable reconfiguration throughput is 3.2 Gbps, where the ICAP is used with 32 bit bus wide and is clocked at 100MHz. However, due to minor overhead results from the internal state machine design and the SRAM storage handling, the Reconfiguration Controller managed to achieve reconfiguration throughput that close to the ideal maximum performance. Since partial bitstream transmission is independent from the dynamic partial reconfiguration process, the platform service only becomes unavailable for 3.81052 millisecond.

Although partial bitstream transmission does not impact the platform service availability, the partial bitstream transmission can still impact the latency performance for remote update. One of the approaches to improve this is through partial bitstream compression. Figure 5 shows the compression ratio of partial bitstream in various logic utilization. The compressed partial bitstream is always smaller than the uncompressed partial bitstream. As the logic utilization increases, the compression performance becomes lower. This mainly due to partial bitstream with low logic utilization has significant amount of repetitive content, where unused logic is filled with zero. This

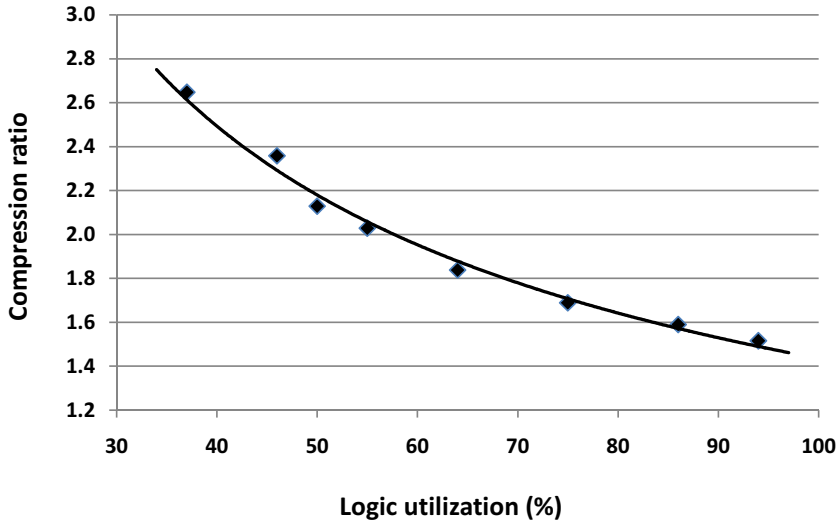

Figure 5. Compression ratio of partial bitstream with various logic utilization.

Table 3. Comparison with previous work.

Publication	Reconf. throughput (Gbps)	Storage	Additional detail
AC_ICAP [2]	3.04824	BRAM	-
DPR Manager [3]	3.07432	SD Flash	-
MST_HWICAP [4]	1.88160	DDR SDRAM	-
BRAM_HWICAP [4]	2.97120	BRAM	-
ICAP Controller [5]	3.19840	DDR SDRAM	UART transmission
Intelligent ICAP Controller [6]	3.19832	SRAM	32bit RLE compression
FlashCAP [7]	3.08000	BRAM	X-MatchPRO compression
ZyCAP [8]	3.05600	DRAM	Xilinx Zynq FPGA
Proposed	3.19985	SRAM	64bit RLE compression, Gigabit Ethernet transmission

repetitive content can be effectively compressed using run-length encoding, where compression of a partial bitstream with 37% logic utilization results roughly in 2.6 compression ratio. Once the logic resources are utilized, the content no longer filled with zero, which increases the entropy in partial bitstream. Even so, part of the partial bitstream can still be compressed, where this part consist of a series of No Operation (NOOP) commands. Additionally, utilized BRAM resources without initialization are filled with zero as well, which can be effectively compressed.

Table 3 summarizes the comparison of implemented Reconfiguration Controller with several related works. The proposed Reconfiguration Controller has slightly higher reconfiguration throughput compared to [5] and [6] mainly due to the use of dedicated bus and SRAM interface for dynamic partial reconfiguration, which result in lower overhead. Dedicated bus used in the proposed architecture offers advantages such as lower processing overhead, higher reliability (due to being independent from other components) and higher consistency in reconfiguration throughput. Additionally, the partial bitstream used in evaluation of proposed work is considerable large in file size, which is 1,524,139 Bytes (~1.489 MB).

5. Conclusion

In this paper, a customized Reconfiguration Controller with remote access to ICAP is proposed. The customized Reconfiguration Controller can achieve at least 3.19 Gbps of recon-

figuration throughput, which significantly reduces the platform service downtime during dynamic partial reconfiguration. Besides that, the latency of partial bitstream transmission is reduced with partial bitstream compression. In addition, the customized Reconfiguration Controller allows user to remotely access to the ICAP for device internal registers readback and configuration. With remote dynamic partial reconfiguration, the accelerator sub-circuits can be updated remotely at run-time after deployment. In general, functional update is important to patch existing design flaws and bugs, to optimize design performance and to cope with the changing of execution unit's functional requirement. Future work will focus on augmenting platform security through end-to-end packet encryption so that the platform can be deployed on non-secured network.

Acknowledgment

This work is supported in part by the CREST grant (UTM Vote No. 4B176) and Universiti Teknologi Malaysia matching grant (UTM Vote No. 00M75).

References

[1] A. Schallenberg, *Dynamic partial self-reconfiguration: Quick modeling, simulation, and synthesis*. Germany: Suedwestdeutscher Verlag fuer Hochschulschriften, 2010.
[2] L. A. Cardona and C. Ferrer, "AC_ICAP: A flexible high speed ICAP controller," *International Journal of Reconfigurable Computing*, vol. 2015, 2015.
[3] J. Tarrillo, F. A. Escobar, F. L. Kastensmidt, and C. Valderrama, "Dynamic partial reconfiguration manager," in *2014 IEEE 5th Latin American Symposium on Circuits and Systems (LASCAS)*, Santiago, Chile, Feb 2014, pp. 1–4.
[4] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time partial reconfiguration speed investigation and architectural design space exploration," in *2009 International Conference on Field Programmable Logic and Applications*, Prague, Czech Republic, Sep 2009, pp. 498–502.
[5] K. Vipin and S. A. Fahmy, "A high speed open source controller for FPGA partial reconfiguration," in *2012 International Conference on Field-Programmable Technology (FPT)*, Seoul, Korea, Dec 2012, pp. 61–66.
[6] S. Liu, R. N. Pittman, A. Forin, and J.-L. Gaudiot, "Minimizing the runtime partial reconfiguration overheads in reconfigurable systems," *The Journal of Supercomputing*, vol. 61, no. 3, pp. 894–911, Sep 2012.
[7] A. Nabina and J. L. Nunez-Yanez, "Dynamic reconfiguration optimisation with streaming data decompression," in *2010 International Conference on Field Programmable Logic and Applications*, Milan, Italy, Sep 2010, pp. 602–607.
[8] K. Vipin and S. A. Fahmy, "ZyCAP: Efficient partial reconfiguration management on the Xilinx Zynq," *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 41–44, Sep 2014.
[9] Xilinx, "UG702 (v13.4) partial reconfiguration user guide," 2012.