International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 4, August 2021, pp. 3222~3228
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i4.pp3222-3228

An FPGA-based network system with service-uninterrupted remote functional update

Tze Hon Tan[1], Chia Yee Ooi[2], Muhammad Nadzir Marsono[3]
[1,3]School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, Johor, Malaysia
[2]Malaysia-Japan International Institute of Technology (MJIIT), Universiti Teknologi Malaysia Kuala Lumpur, Kuala Lumpur, Malaysia

Article Info

Article history:

Received Oct 13, 2020
Revised Jan 4, 2021
Accepted Jan 19, 2021

Keywords:

Dual modular redundancy
Dynamic partial reconfiguration
NetFPGA
Service-uninterrupted remote functional update

ABSTRACT

The recent emergence of 5G network enables mass wireless sensors deployment for internet-of-things (IoT) applications. In many cases, IoT sensors in monitoring and data collection applications are required to operate continuously and active at all time (24/7) to ensure all data are sampled without loss. Field-programmable gate array (FPGA)-based systems exhibit a balanced processing throughput and datapath flexibility. Specifically, datapath flexibility is acquired from the FPGA-based system architecture that supports dynamic partial reconfiguration feature. However, device functional update can cause interruption to the application servicing, especially in an FPGA-based system. This paper presents a standalone FPGA-based system architecture that allows remote functional update without causing service interruption by adopting a redundancy mechanism in the application datapath. By utilizing dynamic partial reconfiguration, only the updating datapath is temporarily inactive while the rest of the circuitry, including the redundant datapath, remain active. Hence, there is no service interruption and downtime when a remote functional update takes place due to the existence of redundant application datapath, which is critical for network and communication systems. The proposed architecture has a significant impact for application in FPGA-based systems that have little or no tolerance in service interruption.

Corresponding Author:

Muhammad Nadzir Marsono
School of Electrical Engineering, Faculty of Engineering
Universiti Teknologi Malaysia
81310 Johor, Malaysia
Email: mnadzir@utm.my

1. INTRODUCTION

The advancement of 5G allows mass deployment of wireless sensors for IoT applications. This phenomenon directly contributes to the tremendous increase in data volume in communication networks, which translated into higher network processing throughput requirement. Meanwhile, datapath flexibility becomes a significant factor to support functional updates as some of the application requirements are unknown during the design time [1] or may change from time-to-time [2]. Hence, the functional update is a vital feature to support new emerging network applications, new network protocols [2], and to cope with the data concept drift [3, 4].

Service availability or uptime is another critical factor, especially for network monitoring and data collection applications, where service interruptions interfering applications with blackout period can negatively impact analytic [5] due to missing data samples. In network and communication applications, most

intermediary devices, including middleboxes, are required to remain active for maintaining end-to-end nodes connectivity and for processing network packets continuously to prevent data loss. For instance, the impact of service interruptions is very significant, especially when the system is deployed for mission-critical applications or deployed on the gateway between internet service providers (ISPs) [6]. Besides, service interruptions due to frequent functional updates can prevent applications from achieving five-nine availability (99.999%) requirement.

Field-programmable gate array (FPGA) systems have emerged as a feasible solution because it can provide a balanced processing throughput and datapath reconfigurability. Due to this, FPGA devices have been widely adopted in many applications, ranging from specialized time-critical applications in niche domains [7-11] to servers applications in the cloud [12, 13]. With the dynamic partial reconfiguration feature, FPGA systems exhibit a higher degree of flexibility in the datapath, where the updating sub-circuitry is unavailable for a few milliseconds, while the rest of the circuitry is not affected. Remote functional updates can be achieved by transferring a new partial bitstream and loaded it to the FPGA device. Hence, having an architecture and design that allow remote functional update by utilizing the dynamic partial reconfiguration feature is important.

In this work, the architecture of a service-uninterrupted FPGA-based network and communication processing system with remote functional update capability is proposed, and it is implemented in the NetFPGA CML development board for experimental testing. In order to achieve functional update without service interruption, the application datapath is duplicated for redundancy, as the reconfiguring (updating) datapath is temporarily disabled during the dynamic partial reconfiguration process. The reconfiguration throughput for dynamic partial reconfiguration is >3.19 Gbps, and the size of the largest partial bitstream is 3.49 MB, which could cause approximately 9.17 ms of service downtime for each functional update when not adopting the proposed architecture with redundancy. For an 8 Gbps processing module, 9.17 ms service downtime can cause up to 8.74 MB of data loss on full bandwidth utilization. Hence, the proposed architecture in this paper is targeted for application in FPGA-based systems that have little or no tolerance in service interruption.

2. RELATED WORKS

Application-specific integrated circuit (ASIC) has been commonly used in the implementation of packet processing function to achieve high throughput. Thus, any functional update would involve the swapping of the electronic circuit board. This would introduce problems in terms of maintenance, where physical access to the facilities is required that will consume a significant amount of service downtime for the maintenance [6]. Software approach is preferred to improve system flexibility, but a full software system has limited processing throughput, i.e. less than several hundred Mbps [6]. Furthermore, applying functional updates to either system would result in service interruption and downtime to critical computation infrastructures.

In general, there are two major approaches for service-uninterrupted functional update in FPGA-based systems [6]: switching-based and buffering-based. The switching-based approach adopted a redundancy mechanism with context switching to select the updating datapath and the operating datapath. Hence, this approach doubles the logic resources required for implementation. On the other hand, the buffering-based approach relies on using buffers to store the packets during the functional update. Although the required logic resources in this approach are halved compared to the switching-based approach, most FPGA devices have insufficient amount of internal random-access memory (RAM) for such packets buffering in gigabit rate. Hence, this approach can result in packet drops when the packet buffer is full.

Katayama *et al.* [6] proposed the buffering-based approach in a functional update to avoid service interruption. In order to prevent the packet drop during a functional update, a multi-context type FPGA is used in the implementation. Zhou *et al.* [14] included a mechanism to realize a functional update without causing service interruption in their Openflow Switch on SoC platform. This mechanism [14] utilized the switching-based approach on the flow table, where a dual-port RAM is used to allow concurrent read and write for context switching during a functional update. Apparently, such mechanism as in [14] is applicable to a single flow table in dual-port RAM rather than customized datapath in FPGA. Hence, the architecture and mechanism utilizing dynamic partial reconfiguration in FPGA to enable service-uninterrupted remote functional update on the datapath is the primary focus of this proposed work.

NetFPGA [15, 16] is an FPGA-based development board that are widely used for prototyping networking devices. To date, there are four variants of NetFPGA development boards, which are the NetFPGA 1G, NetFPGA 10G, NetFPGA CML, and NetFPGA SUME. NetFPGA CML and NetFPGA SUME are high-end development boards, which comes with Xilinx 7 Series FPGA. Besides a large number of logic resources, Xilinx 7 Series FPGA includes the support for dynamic partial reconfiguration. Except for NetFPGA 1G, the other NetFPGA development boards support stand-alone mode, where attachment to a host PC can be avoided as

power can be supplied externally. Essentially, NetFPGA provides a well-established framework [16] for network application development and basic open-source reference designs [15] for packet forwarding.

With dynamic partial reconfiguration feature, a pre-defined region of circuitry can be dynamically reconfigured without impacting the other functional blocks. This feature is managed by the reconfiguration controller, which is either implemented internally within the FPGA device or implemented with an external device. A single-chip solution is made possible by implementing a reconfiguration controller internally with logic resources available within the FPGA device. Fundamentally, dynamic partial reconfiguration enables update-in-the-field [17, 18] to deal with dynamic requirements in application accelerators, where the datapath can be updated without impacting the other functional modules. In order to utilize this feature, architecture with a mechanism capable of handling the dynamic partial reconfiguration process is needed.

3. PROPOSED ARCHITECTURE

In our previous work [19], a standalone remote dynamically reconfigurable middlebox has been developed. However, the application services in this middlebox are interrupted during every remote functional update. By leveraging on the architecture from [19] as shown in Figure 1, this work presents the proposed high-level architecture in Figure 2 with major additional updates to enable remote functional update without causing service interruptions. The major updates over [19] are:

- Duplicating the reconfigurable partition (RP) and its internal modules
- The arbiter is replaced by allocator so that the processing modules in both reconfigurable partitions are capable of functioning together in usual operating mode

In Figures 1 and 2, the dotted boxes denote region that can be dynamically reconfigured. Meanwhile, the shaded regions represent the application plane modules. In general, the proposed architecture consists of application plane and management plane. The modules in the management plane process management related tasks, including dynamic partial reconfiguration and modules coordination. On the other hand, the application plane is mainly focused on network packets and applications processing. By adopting dual modular redundancy (DMR) approach, the application modules consisting application datapath residing in the respective reconfigurable partition can cover for each other when the dynamic partial reconfiguration is in progress, thus allowing uninterrupted services.

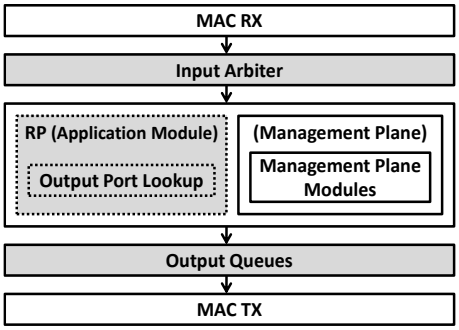

Figure 1. Referenced network middlebox architecture [19]

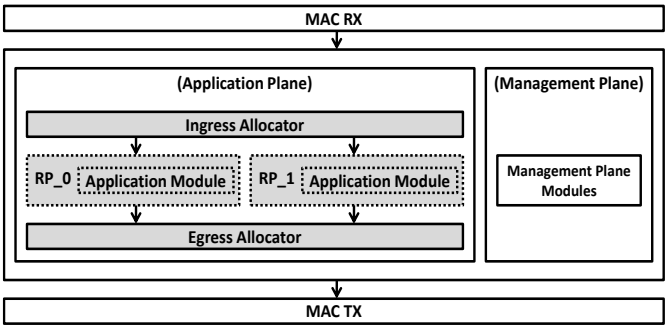

Figure 2. Referenced network middlebox architecture [19]

4. IMPLEMENTATION

The proposed architecture is implemented on NetFPGA CML board according to the functional block diagram shown in Figure 3. The FPGA implementation flow includes: behaviorally describes the function blocks and modules with Verilog HDL, verifies their functionality with ModelSim waveform simulation, and test the implementation experimentally in NetFPGA CML board. The NetFPGA CML board is operating in the standalone mode, where the power is supplied externally so that attachment to PC through PCIe can be avoided. This is useful for the embedded system deployment and can improve the overall scalability.

Based on NetFPGA CML development environment setup, the modelled Verilog HDL sources are synthesized with Xilinx ISE 14.6. All synthesized netlists are then used in Xilinx PlanAhead for subsequent flow includes: map, place and route, timing analysis and bitstream generation. Another reason for using the Xilinx PlanAhead is to simplify the dynamic partial reconfiguration flow, where the scripts for execution are automatically generated, and its execution can be managed internally. Additionally, the Xilinx PlanAhead contains a GUI to ease the definition of location and size of the dynamically reconfigurable area.

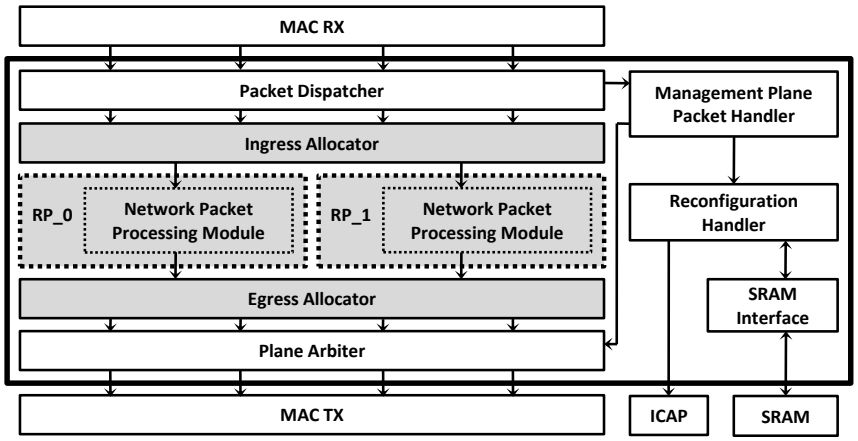

Figure 3. The implemented functional block diagram with respective application module in RP_0 & RP_1

4.1. Management plane

The packet dispatcher dispatches packets to either management plane or application plane based on the packet header identifier. Meanwhile, the plane Arbiter is responsible for packets arbitration between the management plane and application plane. For management type packets, the payload is extracted by the management plane packet Handler.

The reconfiguration Handler stores the extracted partial bitstream to SRAM through SRAM Interface. Upon the arrival of all partial bitstream to the FPGA device, the Reconfiguration Handler asserts flag to stop the packet flow to respective partition modules. Once the modules transit into the idle state, the Reconfiguration Handler loads the partial bitstream to configuration memory through an internal configuration access port (i.e., ICAP, a Xilinx FPGA primitive). After the final piece of the partial bitstream is loaded to the configuration memory, a readback sequence is used to retrieve the status of dynamic partial reconfiguration. Subsequently, the modules are initialized, and it will assert the flag to allow packets to flow through.

4.2. Application plane

The application plane consists of ingress allocator, network packet processing module (application module consisting application datapath), and Egress Allocator. The Ingress Allocator redirects packets to the network packet processing module in the idle state for processing. Similarly, the Egress Allocator redirects packets to Plane Arbiter whenever the network packet processing modules flags its request.

There are two network packet processing modules located at respective reconfigurable partition, namely RP_0 and RP_1. This implementation allows the network packet processing modules to cover for each other, especially when either one is dynamically reconfigured. When both network packet processing modules are activated, the Ingress Allocator redirects packets based on their availability, where both modules can service packets in parallel. The network packet processing module is deactivated for reconfiguration based on the flag from the reconfiguration packet header. As the application plane datapath depends on the network algorithmics, its architecture is not discussed in this paper.

5. EVALUATION

In the NetFPGA CML Kintex 7 device (XC7K325T-1FFG676), there are 50,950 slices and 445 BRAMs available for implementation. Table 1 lists the required logic resources for implementation. Modules in each reconfigurable partition (RP) utilized approximately 1,006 slices and 17 BRAMs for a learning content addressable memory (CAM) [20] switch application.

For functional verification and experimental testing, the reconfigurable modules are dynamically reconfigured for expansion with deep packet inspection (DPI) blocks. Table 2 lists the required amount of logic resources for implementation after DPI blocks expansion. Network packets are injected from another PC to the NetFPGA for analysis using Wireshark packet analyzer, where the behaviour from the captured packets is used for verification of successful reconfiguration process.

Based on the experimental evaluation, the reconfiguration throughput is higher than 3.19874 Gbps, where the maximum reconfiguration throughput at 100 MHz clock frequency and 32-bit bus width is 3.20000 Gbps. The size of partial bitstreams is 3,666,884 bytes and 2,241,540 bytes for RP_0 and RP_1 respectively. Loading these partial bitstreams to the configuration memory through ICAP and modules initialization

requires 1,146,179 clock cycles and 700,758 clock cycles respectively, where the exact reconfiguration throughput for each partition is 3.19922 Gbps and 3.19874 Gbps respectively. Table 3 shows the comparison of acquired reconfiguration throughput with other similar works.

The clock cycles used for dynamic reconfiguration and module initialization implies service downtime period as well. For a fully utilized bandwidth datapath, there will be a data loss of 9,169,432 bytes (1,146,179 * 64 bits) if the proposed architecture and mechanism is not adopted, where the data bus width is 64 bits. For maximum transfer unit (MTU) of 1,500 bytes, the amount of full-sized packet loss is more than 6112 packets. This impact is significant for applications with little to no tolerance to such service interruption or when the deployment is near to the network core. Figure 4 shows the floor plan for RP_0 and RP_1 in Xilinx PlanAhead.

Table 1. Logic resources used for implementation with learning CAM switch

Resources type	RP_0 Partial Reconfigurable Module	RP_1 Partial Reconfigurable Module	Static Regions Module	Total utilization	Available
Slice registers	1,232	1,232	19,256	21,720	407,600
Slice LUTs	1,411	1,331	18,694	22,139	203,800
Occupied slices	1,006	981	7,251	9,238	50,950
BRAM	17	17	115	149	445

Table 2. Logic resources used for implementation with learning CAM switch and DPI blocks

Resources type	RP_0 Partial Reconfigurable Module	RP_1 Partial Reconfigurable Module	Static Regions Module	Total utilization	Available
Slice registers	1,496	1,496	19,256	22,248	407,600
Slice LUTs	1,886	1,806	18,694	22,937	203,800
Occupied slices	957	812	7,251	9,020	50,950
BRAM	17	17	115	149	445

Table 3. Reconfiguration throughput comparison with other similar works

Publication	Reconf. throughput (Gbps)	Storage
ZyCAP [21]	3.05600	DRAM
DPR Manager [22]	3.07432	SD Flash
FlashCAP [23]	3.08000	BRAM
Intelligent ICAP Controller [24]	3.19832	SRAM
ICAP Controller [25]	3.19840	DDR SDRAM
BRAM_HWICAP [26]	2.97120	BRAM
AC_ICAP [27]	3.04824	BRAM
Proposed	3.19874	SRAM

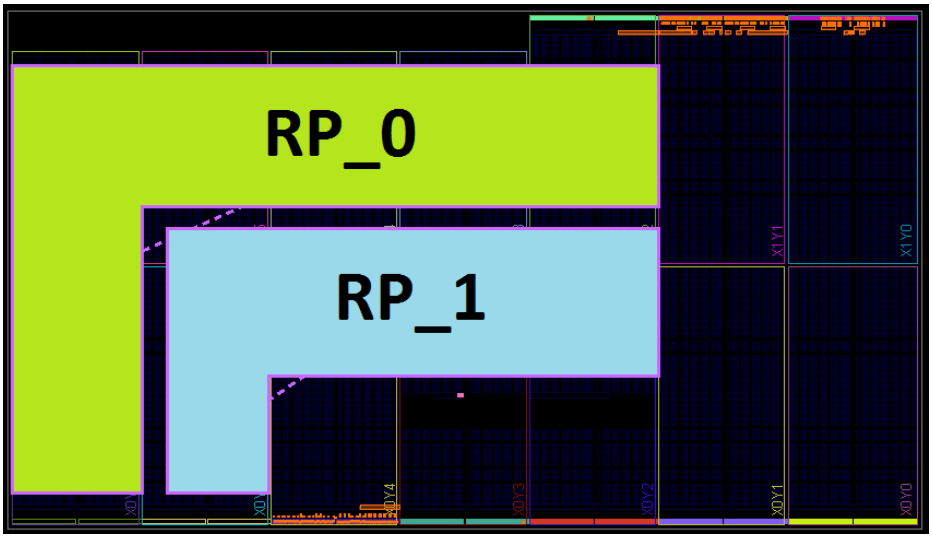

Figure 4. The floor plan consisting RP_0 & RP_1 in Xilinx PlanAhead

6. CONCLUSION

This paper presented a standalone FPGA-based system architecture that allows remote functional update without causing service interruption by adopting a redundancy mechanism in the application datapath and utilizing the dynamic partial reconfiguration feature. Significantly, service interruption is no longer triggered by remote functional updates, and the processing throughput is doubled except during the dynamic partial reconfiguration (9.17 ms). Hence, the proposed architecture in this paper is well-suited for applications of the FPGA-based system that has limited tolerance in service interruption. Such applications include, IoT sensors in monitoring and data collection applications are operating continuously at all time (24/7) to avoid data loss. Wireless sensors can be deployed prevalently with the emergence of the 5G network, which would further strengthen the significance of having a service-uninterrupted remote functional update in FPGA-based system. Future works will focus on architecture exploration for flexibility improvement and data analytics integration for deployment.

ACKNOWLEDGEMENTS

This work is supported in part by the Ministry of Higher Education Malaysia Fundamental Research Grant FRGS/1/2019/TK04/UTM/02/30 (UTM Vote No. 5F159).

REFERENCES

[1] F. Hategekimana, T. J. Whitaker, M. J. H. Pantho, and C. Bobda, "IoT device security through dynamic hardware isolation with cloud-based update," *Journal of Systems Architecture*, vol. 109, p. 101827, 2020.

[2] ARISTA, "Four key trends in the networked use of FPGAs," Dec 2018, White Paper. [Online]. Available: https://www.arista.com/assets/data/pdf/Whitepapers/Trends-in-FPGA-WP.pdf

[3] P. Mulinka and P. Casas, "Stream-based machine learning for network security and anomaly detection," *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, Budapest, Hungary, 2018, pp. 1-7.

[4] X. Tian, Q. Sun, X. Huang, and Y. Ma, "A dynamic online traffic classification methodology based on data stream mining," *2009 WRI World Congress on Computer Science and Information Engineering*, Los Angeles, CA, USA, 2009, pp. 298-302.

[5] M. L. Brown and J. F. Kros, "Data mining and the impact of missing data," *Industrial Management & Data Systems*, vol. 103, no. 8, pp. 611-521, 2003.

[6] M. Katayama, H. Kai, J. Yoshida, M. Inami, H. Yamada, K. Shiomoto, and N. Yamanaka, "A 10 Gb/s firewall system for network security in photonic era," *IEICE transactions on communications*, vol. 88, no. 5, pp. 1914-1920, 2005.

[7] G. P. Seu, G. N. Angotzi, G. Tuveri, L. Raffo, L. Berdondini, A. Maccione, and P. Meloni, "On-FPGA real-time processing of biological signals from high-density meas: A design space exploration," *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, Lake Buena Vista, FL, 2017, pp. 175-183.

[8] B. Johnson., and Sheeba Rani J., "A high throughput fully parallel-pipelined FPGA accelerator for dense cloud motion analysis," *2016 IEEE Region 10 Conference (TENCON)*, Singapore, 2016, pp. 2589-2592.

[9] R. Pottathuparambil, J. Coyne, J. Allred, W. Lynch, and V. Natoli, "Low-latency FPGA based financial data feed handler," *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*, Salt Lake City, UT, USA, 2011, pp. 93-96.

[10] C. O. Sereati, A. D. W. Sumari, T. Adiono, and A. S. Ahmad, "Towards cognitive artificial intelligence device: an intelligent processor based on human thinking emulation," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, no. 3, pp. 1475-1482, 2020.

[11] B. M. Khammas, I. Ismail, and M. Marsono, "Pre-filters in-transit malware packets detection in the network," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 17, no. 4, pp. 1706-1714, 2019.

[12] D. Diamantopoulos and C. Kachris, "High-level synthesizable dataflow mapreduce accelerator for FPGAcoupled data centers," *2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, Samos, Greece, 2015, pp. 26-33.

[13] J. Weerasinghe, R. Polig, F. Abel, and C. Hagleitner, "Network-attached FPGAs for data center applications," 2016 *International Conference on Field-Programmable Technology (FPT)*, Xi'an, 2016, pp. 36-43

[14] S. Zhou, W. Jiang, and V. K. Prasanna, "A flexible and scalable high-performance openflow switch on heterogeneous SoC platforms," *2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, Austin, TX, USA, 2014, pp. 1-8.

[15] "Netfpga," Published online, 2014. [Online]. Available: http://netfpga.org/

[16] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "NetFPGA: reusable router architecture for experimental research," *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, Seattle, WA, USA, 2008, pp. 1-7.

[17] J. Becker and R. Hartenstein, "Configware and morphware going mainstream," *Journal of Systems Architecture*, vol. 49, no. 4-6, pp. 127–142, Sep 2003.

[18] A. Schallenberg, "Dynamic partial self-reconfiguration: Quick modeling, simulation, and synthesis," Germany: Suedwestdeutscher Verlag fuer Hochschulschriften, 2010.

[19] T. H. Tan, C. Y. Ooi, and M. N. Marsono, "rrBox: A remote dynamically reconfigurable network processing middlebox," *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, London, UK, 2015, pp. 1-4.
[20] N. S. Kay and M. Marsono, "Ternary content addressable memory for longest prefix matching based on random access memory on field programmable gate array," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 17, no. 4, pp. 1882-1889, 2019.
[21] 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, 2014.
[22] J. Tarrillo, F. A. Escobar, F. L. Kastensmidt, and C. Valderrama, "Dynamic partial reconfiguration manager," *2014 IEEE 5th Latin American Symposium on Circuits and Systems*, Santiago, Chile, 2014, pp. 1-4.
[23] A. Nabina and J. L. Nunez-Yanez, "Dynamic reconfiguration optimisation with streaming data decompression," *2010 International Conference on Field Programmable Logic and Applications*, Milan, Italy, 2010, pp. 602-607.
[24] 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, 2012.
[25] K. Vipin and S. A. Fahmy, "A high speed open source controller for FPGA partial reconfiguration," *2012 International Conference on Field-Programmable Technology*, Seoul, Korea (South), 2012, pp. 61-66.
[26] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time partial reconfiguration speed investigation and architectural design space exploration," *2009 International Conference on Field Programmable Logic and Applications*, Prague, Czech Republic, 2009, pp. 498-502.
[27] L. A. Cardona and C. Ferrer, "AC_ICAP: A flexible high speed ICAP controller," *International Journal of Reconfigurable Computing*, vol. 2015, no. 79, pp. 1-15, 2015.

BIOGRAPHIES OF AUTHORS

Tze Hon Tan is a Ph.D. student at Faculty of Engineering from Universiti Teknologi Malaysia. He obtained his Master Degree in Electrical Engineering from Universiti Teknologi Malaysia in 2015 and Bachelor Degree in Computer Engineering from Universiti Teknologi Malaysia in 2012. His research areas are: digital systems, reconfigurable computing, fog computing, and data analytics. He is working on an FPGA-based middlebox and fog computing platform to embrace the upcoming IoT wave.

Chia Yee Ooi, is an associate professor of Malaysia-Japan International Institute of Technology at Universiti Teknologi Malaysia. She received her B.E. and M.E degrees in electrical engineering from Universiti Teknologi Malaysia in 2001 and 2003 respectively. She obtained her PhD degree in Information Science from Nara Institute of Science and Technology in 2006. She joined Universiti Teknologi Malaysia in 2002. Her research interests include design-for-test, verification, and digital system design. She is a member of IEICE and a senior member of IEEE.

Muhammad Nadzir Marsono, is an associate professor in Electronic and Computer Engineering, School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia. He obtained his PhD in Electrical and Computer Engineering from the University of Victoria BC Canada in 2007, MEng in Electrical Engineering from Universiti Teknologi Malaysia in 2001, and BEng in Computer Engineering from Universiti Teknologi Malaysia in 1999. His research focuses are in digital system design, computer architecture, embedded systems, domainspecific reconfigurable computing, multicore/manycore system-on-chip, network-on-chip, network algorithmics, and network processing architectures. Further info on his homepage: http://www.fke.utm.my/nadzir/