CHAPTER 1

INTRODUCTION

1.1 Reconfigurable Computing as a Paradigm Shift in Computing

The inefficiency [1, 2] of the von Neumann machine paradigm [3] leads to integration of both von Neumann central processing unit (instruction driven) and non-von Neumann accelerators (data driven) implementation [2]. This is because instruction stream execution of von Neumann machine requires a lot of memory cycles. Currently, the focus of implementation has shifted towards replacing hardwired accelerators with reconfigurable devices. This approach offers better flexibility [4] and contributes to the emergence of reconfigurable computing techniques and methodologies.

Systems-on-Chip (SoC) becomes a common implementation solution in electronic systems but it lacks the flexibility to cope with the rapid changes of functionality requirement. A reconfigurable device with large number of logic elements becomes a good alternative to SoC by offering flexibility at the cost of performance. This results in the emergence of System-on-Programmable-Chip (SoPC) [5], which implement a system-on-chip consisting intellectual property (IP) cores, general-purpose processor (GPP) and custom hardware in a single reconfigurable device. To apply changes and updates to the custom hardware in the reconfigurable device, a new partial bitstream is loaded to the reconfigurable device at run time. This leads to the requirement of a good framework for remote dynamically reconfigurable platform.
1.2 Dynamic Reconfiguration in Network Processing Unit

Telecommunication bandwidth is expected to grow at a rate three times higher than computation processing capability [6]. This trend clearly shows that high performance requirement, especially throughput requirement is critical in network processing units to cope with the growing demands for bandwidth [7]. Application-specific integrated circuit (ASIC) design is a possible implementation solution as it enables low-level parallelism and processing of packets in deep pipeline [8]. Unfortunately, ASIC designs are inflexible and do not allow functional updates. Thus, ASICs usually have limited lifetime in the market due to their incapability to adapt to changes. ASIC implementation also requires longer time to be designed, fabricated and tested, compared to other implementation alternatives.

The flexibility to do functional update is an important requirement in network processing [9] as it would allow network applications to remain updated from time to time and prolongs its lifetime in the market. Flexibility requirements come when some of the execution requirements are not known during the design time or may change over time in unforeseen ways. For instance, firewall implemented in ASIC could only protect the networks from known threats during its design time [8]. Network processing units implemented in software can have functional updates but the execution is very low in performance, especially the throughput. For example, throughput of software-based router in packet processing is slower than hardware-based router in several orders of magnitude [7]. This limitation is very significant, as network application requires throughput to cope with the growing demands for bandwidth [7]. In short, both performance and flexibility are important for network application devices.

Field-programmable gate array (FPGA) is a good option to implement network application devices because FPGA implementation can offer desirable balance between flexibility and performance. FPGA has both performance advantages of ASIC solution and flexibility advantage of software solution [7]. For instance, firewall implemented in reconfigurable device is able to achieve significant improvements in performance and security [10]. The system flexibility comes from the reconfigurability feature in the FPGA devices and is used to update the system when new security threats is found [10]. With the performance and flexibility offered by reconfigurable device, NetFPGA [11] has emerged as a network application prototyping platform that utilizes FPGA devices.
Most network application especially middleboxes are required to operate in high throughput and are distributed. Additionally, middleboxes are required to remain active and operate continuously so that their connectivity with end nodes is maintained and it can be updated remotely. Thus, the cost to apply updates to such system is very high. This problem can be solved by utilizing dynamic reconfiguration feature found in reconfigurable devices. However, utilization of dynamic reconfiguration feature is not straightforward and requires proper methodology in the design process. Therefore, a good framework that is able to efficiently perform dynamic reconfiguration in reconfigurable devices is required in network applications.

### 1.3 Problem Statement

Extant works have shown that reconfigurable devices is a good solution for implementation that requires both performance and flexibility [7–9, 12, 13]. There are many works that have been proposed using reconfigurable device to achieve performance advantage. However, works on exploiting the flexibility in reconfigurable device are still limited. The reason lies on the fact that dynamic reconfiguration is of recent interest and is only supported by limited family of reconfigurable devices. Additionally, proper methodology in the design process is required to utilize dynamic reconfiguration.

Remote dynamic reconfiguration is capable to cope with rapid functional changes for system implemented in reconfigurable devices. To achieve this, partial bitstream should be loaded into these reconfigurable devices in the most generic and efficient way. However, some recent works [14–20] achieved partial reconfiguration by using General Purpose Processor embedded in the design. This requires additional logic resources and longer time to perform partial reconfiguration. Additionally, some of the works in [15, 19–24] used shared bus structure, which may restrict other components from using it during reconfiguration process [25].

NetFPGA development board is a network application FPGA development board. Previous works [10, 26] show that the NetFPGA development board has the potential to be developed into a remote dynamically reconfigurable platform. This is because NetFPGA uses reconfigurable device that support dynamic reconfiguration feature and also provide well-established communication framework. Recently, [26] implemented a network application that uses the partial reconfiguration in NetFPGA
development board. However, JTAG interface is used to load the partial bitstream into the FPGA device, which requires longer reconfiguration time [15]. Zhang et al. [10] have designed a remote dynamically reconfigurable security system using NetFPGA development board. However, the remote dynamic reconfiguration rely on host PC for bitstream transmission and translation, which is inefficient.

1.4 Research Objectives

Based on the background studies and existing issues, the aim of this thesis is to design and implement a remote dynamically reconfigurable platform. The main objectives of this research work are:

1. To design and implement a remote dynamically reconfigurable platform using NetFPGA development board. The developed platform does not rely on GPP or host computer to handle the dynamic reconfiguration process. Instead, the reconfiguration controller has been implemented using existing logic resources in the reconfigurable device itself. The application implemented using the developed platform should be able to be updated remotely through the Ethernet connection.

2. To design and implement a remote dynamically reconfigurable middlebox for network protection scheme. The packet-forwarding algorithm in the developed middlebox should be able to be updated remotely through the Ethernet connection. The network protection application implemented using the developed middlebox should be able to be updated remotely through the UDP/IP connection.

1.5 Scope of Work

Based on the research objectives and available resources, the scope of this research are as follow:

1. The design of remote dynamically reconfigurable platform excludes authentication mechanism. Authentication mechanism is not in the scope of work
because it is an optional feature and this feature can be extended to the developed platform when the application requires it.

2. The partial bitstream is not encrypted for dynamic reconfiguration. Partial bitstream encryption is not in the focus of this work as it is an optional feature as this feature can be included afterward depending on application requirements.

3. The case studies of network protection are targeted for stateless Network Intrusion Prevention System (NIPS) and port based firewall. However, other applications can still be implemented using the developed platform as the developed platform is functionally extensible.

4. The developed platform supports packets size up to 2048 Bytes, which is larger than the maximum transmission unit of Ethernet V2. However, the packets size can be increased by adjusting the depth of FIFO used.

5. The size of each bistream packets are limited to 1016 Bytes. Even so, the size of the bitstream packet can be increased by adjusting the depth of FIFO.

1.6 Research Contributions

This thesis contributes to two research contributions. The first contribution is the architecture for remote dynamically reconfigurable platform on NetFPGA 10G development boards. The proposed architecture supports remote update on the packet-forwarding mechanism, which allows high degree of customization and optimization after system deployment. With the proposed architectures, most existing network applications from NetFPGA repository [27] can be updated remotely when integrated into the implemented platform.

The second contribution is the architecture of a customized reconfiguration controller using available logic resources in the FPGA device. The design effort results in higher efficiency in the dynamic reconfiguration process and utilization in logic resources. Additionally, the dynamic reconfiguration process can be handled internally by the implemented reconfiguration controller, therefore reduces external component dependency. Combination of both contributions result in a better design and implementation alternative for remote dynamically reconfigurable platform, which is resource-efficient and processing efficient.
1.7 Thesis Organization

The rest of this thesis is organized based on the following structure.

Chapter 2 covers literature review of this research, which are related theoretical background and related works. Discussion on literatures mainly focus on dynamic reconfiguration and network applications.

Chapter 3 describes methodology to achieve the research objectives. This includes explanation on the architecture components, implementation flow, development environment and verification techniques.

Chapter 4 presents details on design and implementation of the proposed platform using NetFPGA development board. This chapter also includes evaluation of the implemented platform for verification and benchmark purposes.

Chapter 5 provides a case study of network protection application using the implemented remote dynamically reconfigurable platform.

Chapter 6 summarizes this thesis, stating contributions and limitations of this research and provides suggestions for future research.