Reducing FPGA Reconfiguration Time Overhead using Virtual Configurations

Ming Liu[‡†], *Zhonghai Lu*[†], *Wolfgang Kuehn*[‡], *Axel Jantsch*[†]

‡ II. Physics Institute
Justus-Liebig-University Giessen (JLU), Germany
{ming.liu, wolfgang.kuehn}@physik.uni-giessen.de

† Dept. of Electronic Systems
Royal Institute of Technology (KTH), Sweden
{mingliu, zhonghai, axel}@kth.se

***Abstract*—Reconfiguration time overhead is a critical factor in determining the system performance of FPGA dynamically reconfigurable designs. To reduce the reconfiguration overhead, the most straightforward way is to increase the reconfiguration throughput, as many previous contributions did. In addition to shortening FPGA reconfiguration time, we introduce a new concept of Virtual ConFigurations (VCF) in this paper, hiding dynamic reconfiguration time in the background to reduce the overhead. Experimental results demonstrate up to 29.9% throughput enhancement by adopting two VCFs in a consumer-reconfigurable design. The packet latency performance is also largely improved by extending the channel saturation to a higher packet injection rate.**

I. Introduction

Partial Reconfiguration (PR) enables the process of dynamically reconfiguring a particular section of an FPGA design while the remaining part is still operating. This vendor-dependent technology provides common benefits in adapting hardware modules during system run-time, sharing hardware resources to reduce device count and power consumption, shortening reconfiguration time, etc. [1] [2] [3]. Typically partial reconfiguration is achieved by loading the partial bitstream of a new design into the FPGA configuration memory and overwriting the current one. Thus the reconfigurable portion will change its behavior according to the newly loaded configuration. Despite the flexibility of changing part of the design at system run-time, overhead exists in the reconfiguration process since the reconfigurable portion cannot work at that time due to the incompleteness of the configuration data. It has to wait to resume working until the complete configuration data have been successfully loaded in the FPGA configuration memory. Therefore in performance-critical applications which require fast or frequent switching of IP cores, the reconfiguration time is significant and should be minimized to reduce the overhead.

To reduce the dynamic reconfiguration overhead, the most straightforward way is to increase the data write-in throughput of the configuration interface on FPGAs, specifically the Internal Configuration Access Port (ICAP) on Xilinx FPGAs [4]. As an additional approach, we address the challenge of reducing the reconfiguration overhead by employing the concept of virtualization on FPGA configuration contexts. The remainder of the paper will be organized as follows: In Section II, related work of reducing dynamic reconfiguration overhead will be discussed. In Section III, we introduce the concept of Virtual ConFigurations (VCF), with which the dynamic reconfiguration time may be fully or partly hidden in the background. Experimental results will demonstrate the performance benefits of using VCFs in Section IV, in terms of data delivery throughput and latency. Finally we conclude the paper and propose our future work in Section V.

II. Related Work

FPGA dynamic reconfiguration overhead refers to the time spent on the module reconfiguration process. At that time, the reconfigurable region on the FPGA cannot effectively work due to the lack of a complete bitstream, and consequently it has negative effects on the system performance. Reconfiguration overhead may be minimized either with a reasonable scheduling policy which decreases the context switching times of hardware modules, or by reducing the required time span for each configuration. There is related discussion on the former approach in our previous publication of [5]. We observe from the experimental results that only less than 0.3% time is spent on the configuration switching with a throughput-aware scheduling policy, not exacerbating much the overall processing throughput of the under-test system; With regard to the latter approach, design optimization approaches have been previously adopted to increase the configuration throughput. For instance in [6], [7] and [8], authors explore the design space of various ICAP designs and enhance the reconfiguration throughput to the order of magnitude of Megabytes per second. Unfortunately the reconfiguration time is still constrained by the physical bandwidth of the reconfiguration port on FPGAs. Other approaches of compressing the partial bitstreams are discussed in [8] and [9] for shrinking the reconfiguration time under the precondition of a fixed configuration throughput. In addition to all the above described contributions, in this paper we will address the challenge of reducing the reconfiguration overhead, employing the concept of virtualization on FPGA configuration contexts.

III. Virtual Configuration

In canonical PR designs, one Partially Reconfigurable Region (PRR) has to stop from working, when a new module is to be loaded to replace the existing one by run-time reconfiguration. This is the overhead of switching hardware processes, which restricts the overall system performance. As a

solution, we propose the concept of Virtual ConFiguration (VCF) to hide the configuration overhead of a PR design. As shown in Figure 1, two copies of configuration contexts, each of which represents a VCF, are altogether dedicated to a single PRR on a multi-context FPGA [10] [11] [12]. The active VCF may still keep working in the foreground when module switching is expected. The run-time reconfiguration only happens invisibly in the background, and the new partial bitstream is loaded into configuration context 2. After the reconfiguration is finished, the newly loaded module can start working by being swapped with the foreground context, migrating from the background to the foreground. The previously active configuration will be deactivated into the background and wait for the next time reconfiguration. The configuration context swapping between the background and the foreground is logically realized by changing the control on the PRR among different VCFs. It does not need to really swap the configuration data in the FPGA configuration memory, but instead switches control outputs taking effect on the PRR using multiplexer (MUX) devices. Hence the configuration context swapping takes only very short time (normally some clock cycles), and is tiny enough to be negligible compared to the processing time of the system design.

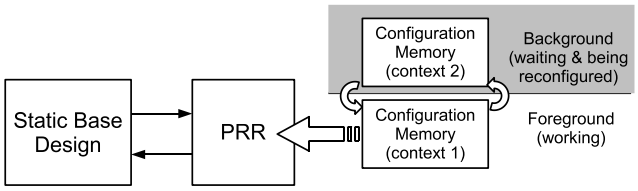

Figure 1. Virtual reconfigurations on multi-context FPGAs

With the approach of adopting VCFs, the reconfiguration overhead can be fully or partly removed with the duplicated configuration contexts. The timing advantage is illustrated in Figure 2, comparing to the canonical PR designs without VCFs. We see in Figure 2a, the effective work time and the reconfiguration overhead have to be arranged in sequence on the time axis, in the canonical PR design without VCFs. By contrast in Figure 2b, the reconfiguration process only happens in the background and the time overhead is therefore hidden by the working VCF in the foreground.

In normal FPGAs with only single-context configuration memories, VCFs may be implemented by reserving duplicated PRRs of the same size (see Figure 3). At each time, only one PRR is allowed to be activated in the foreground and selected to communicate with the rest static design by MUXes. The other PRR waits in the background for reconfiguration and will be swapped to the foreground to work after the module is successfully loaded. Taking into account the resource utilization overhead of reserving duplicated PRRs, usually we do not adopt more than 2 VCFs.

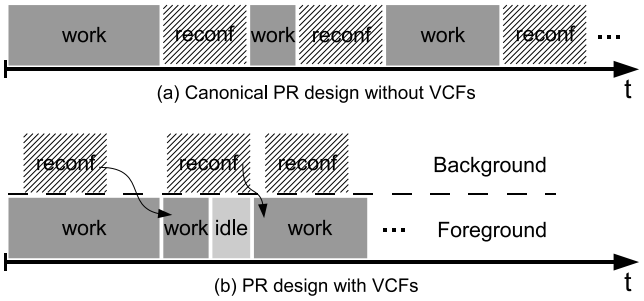

Figure 2. Timing diagrams of PR designs without or with VCFs

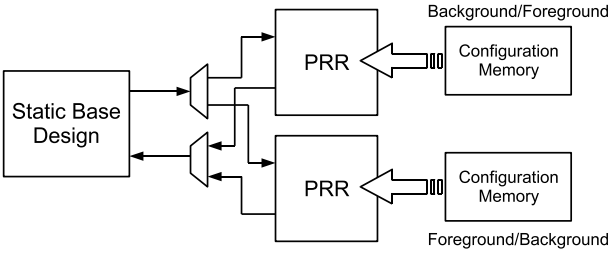

Figure 3. Virtual reconfigurations on single-context FPGAs

IV. Experiments

A. Experimental Setup

To investigate the impact of VCFs on performance, we set up a producer-consumer design with run-time reconfiguration capability. As illustrated in Figure 4, the producer periodically generates randomly-destined packets to 4 consumers and buffers them in 4 FIFOs. Each FIFO is dedicated to a corresponding consumer algorithm, which can be dynamically loaded into the reserved consumer PRR. The scheduler program monitors the "almost_full" signals from all FIFOs and arbitrate the to-be-loaded consumer module using a Round-Robin policy. Afterwards, the loaded consumer will consume its buffered data in a burst mode, until it has to be replaced by the winner of the next-round reconfiguration arbitration. The baseline canonical PR design has only one configuration context and must stop the working module before the reconfiguration starts. In the PR design with VCFs, we adopt only two configuration contexts since the on-chip area overhead of multiple configuration contexts should be minimized. Experimental measurements have been carried out in cycle-accurate simulation using synthesizable VHDL codes. Simulation provides much convenience for observing all the signals in the waveform and debugging the design. It will have the same results when implementing the design on any dynamically reconfigurable FPGA. Both the baseline and the VCF designs run at a system clock of 100 MHz. The overall on-chip buffering capability is parameterized in the order of KiloBytes. For the reconfiguration time of each module, we select 10

μs which is a reasonable value when using the practical Xilinx ICAP controller for partial reconfiguration [7]. The generated packets are 256-bit wide. The FIFO width is 32 bits. Before packets go into the FIFO, they are fragmented into flits.

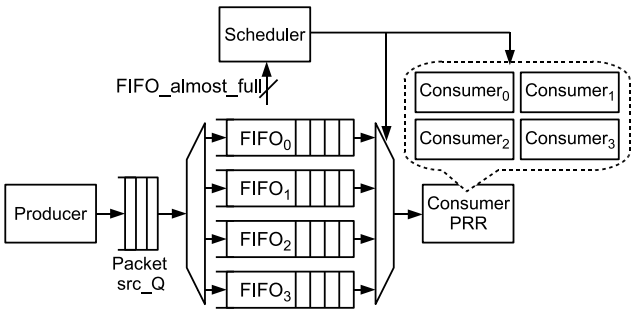

Figure 4. Experimental setup of the consumer-reconfigurable design

B. Results

We did measurements on the received packet throughput in the unit of packets per cycle per consumer node, with the FIFO depth of 512, 1K and 2K respectively. Measurement results are demonstrated in Figure 5. We observe from the figure that:

1) As the packet injection rate increases, the on-chip communication becomes saturated progressively due to the limitation of the packet consuming capability;
2) For both types of PR designs (red or light curves for with 2 VCFs and blue or dark curves for without), larger FIFO depths lead to higher saturated throughput, since the data read-out burst size can be increased by larger buffering capability, and the reconfiguration time overhead is comparatively reduced;
3) Introducing VCFs can further reduce the reconfiguration overhead by hiding the reconfiguration time in the background. In the most obvious case of 1K FIFO depth, two VCFs increase the throughput from 0.0127 packets/cycle/node to 0.0165, achieving a performance enhancement of 29.9%. Other two cases of 512 and 2K FIFO depth have a performance enhancement of 26.4% and 17.9% respectively.

We enlarged the time span of each configuration from 10 μs to 50 μs and did further throughput measurements with a middle-size FIFO depth of 1K. Results are demonstrated in Figure 6, comparing the PR design using 2 VCFs with the one without VCF. We observe that the overall system throughput is worsened by the increased reconfiguration time overhead, specifically from a saturated value of 0.0127 (see Figure 5) into 0.00492 packets/cycle/node for the non-VCF design. The increased reconfiguration time also easily results in the channel saturation at an even lower packet injection rate of about 1 packet per 50 cycles. In this test, we can still see the performance improvement of 27.6%

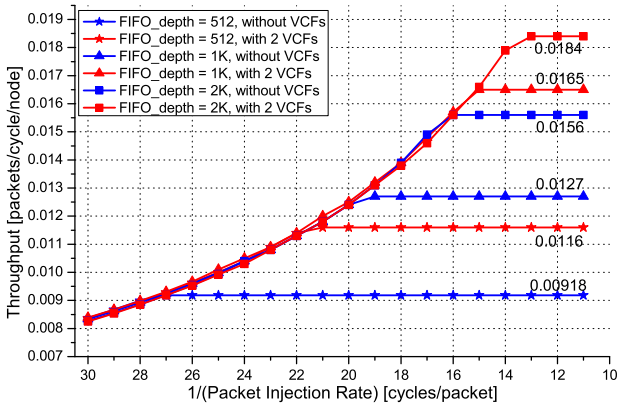

Figure 5. Throughput measurement results (reconfiguration time = 10 μs)

(0.00628 vs. 0.00492 packets/cycle/node), using 2 VCFs to partly counteract the reconfiguration overhead. The channel saturation point is extended to about 1 packet per 35 cycles by duplicated VCFs

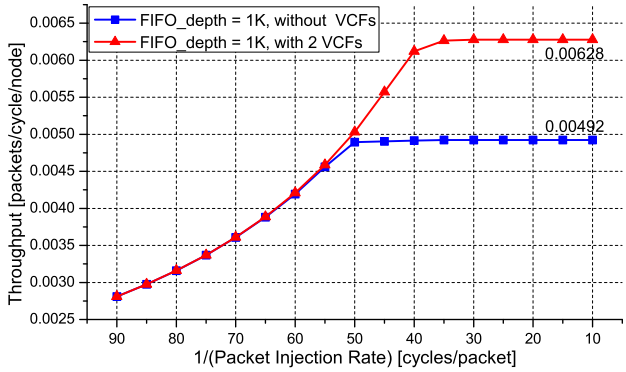

Figure 6. Throughput measurement results (reconfiguration time = 50 μs)

Except for the throughput comparison, we collected also statistics on packet latency performance to demonstrate the effect of using VCFs. We discuss the average latency of a certain amount of packets, and exclude the system warm-up and cool-down cycles out of measurements, only taking into account steady communications. The latency is calculated from the instant when the packet is injected into the source queue to that when the packet is received by the destination node. It consists of two components: the queuing time in the source queue and the network delivery time in flit FIFOs. Measurements were conducted in the experimental setup with the smaller reconfiguration time of 10 μs and the middle-size FIFO depth of 1K. Results are illustrated in Figure 7. We observe that 2 VCFs have a slight reduction effect on the packet latency before the channel saturation. In this curve segment, packets do not stay in the source queue for too long time, but they must wait in flit FIFOs until

their specific destination node is configured to read them out in a burst mode. Therefore we see two comparatively flat curve segments before the channel saturation, because of the steady switching frequency of consumer nodes. Nevertheless after the channel's packet delivery capability is saturated, packets have to spend much time waiting in the source queue to enter the flit FIFOs. Thus the average latency of packets deteriorates significantly and generates rising curve segments in the figure. By contrast, using 2 VCFs may reduce the reconfiguration overhead and extends the channel saturation to a higher packet injection rate. It reduces the packet wait time in the source queue and introduce them into the flit FIFOs at an early time, leading to a large improvement on the packet latency performance.

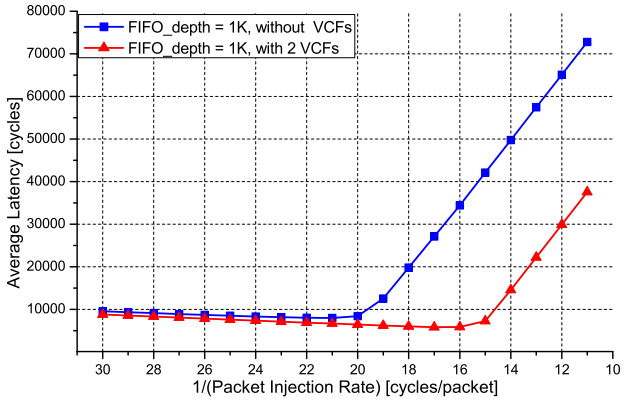

Figure 7. Latency measurement results (reconfiguration time = 10 μs)

V. Conclusion and Future Work

In this paper, we introduce the concept of virtual configurations to hide the FPGA dynamic reconfiguration time in the background and reduce the reconfiguration overhead. Experimental results on a consumer-reconfigurable design demonstrate up to 29.9% throughput improvement of received packets by each consumer node. The packet latency performance is largely improved as well, by extending the channel saturation to a higher packet injection rate. This approach is well suited for PR designs on multi-context FPGAs. For single-context FPGAs, performance improvement is accompanied by resource utilization overhead of reserving duplicated PR regions.

In the future work, we will take advantage of VCFs in practical PR designs for specific applications. Research and engineering work on multi-context FPGAs will also be useful to popularize this technology in dynamically reconfigurable designs with high performance requirements.

Acknowledgment

This work was supported in part by BMBF under contract Nos. 06GI9107I and 06GI9108I, FZ-Juelich under contract No. COSY-099 41821475, HIC for FAIR, and WTZ: CHN 06/20.

References

[1] C. Kao, "Benefits of Partial Reconfiguration", *Xcell Journal*, Fourth Quarter 2005, pp. 65 - 67.

[2] E. J. Mcdonald, "Runtime FPGA Partial Reconfiguration", *In Proc. of 2008 IEEE Aerospace Conference*, pp. 1 - 7, Mar. 2008.

[3] C. Choi and H. Lee, "An Reconfigurable FIR Filter Design on a Partial Reconfiguration Platform", *In Proc. of First International Conference on Communications and Electronics*, pp. 352 - 355, Oct. 2006.

[4] Xilinx Inc., "Virtex-4 FPGA Configuration User Guide", UG071, Jun. 2009.

[5] M. Liu, Z. Lu, W. Kuehn, and A. Jantsch, "FPGA-based Adaptive Computing for Correlated Multi-stream Processing", *In Proc. of the Design, Automation & Test in Europe conference*, Mar. 2010.

[6] J. Delorme, A. Nafkha, P. Leray and C. Moy, "New OPBHW-ICAP Interface for Realtime Partial Reconfiguration of FPGA", *In Proc. of the International Conference on Reconfigurable Computing and FPGAs*, Dec. 2009.

[7] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, "Run-time Partial Reconfiguration Speed Investigation and Architectural Design Space Exploration", *In Proc. of the International Conference on Field Programmable Logic and Applications*, Aug. 2009.

[8] S. Liu, R. N. Pittman, and A. Forin, "Minimizing Partial Reconfiguration Overhead with Fully Streaming DMA Engines and Intelligent ICAP Controller", *In Proc. of the International Symposium on Field-Programmable Gate Arrays*, Feb. 2010.

[9] J. H. Pan, T. Mitra, and W. Wong, "Configuration Bitstream Compression for Dynamically Reconfigurable FPGAs", *In Proc. of the International Conference on Computer-Aided Design*, Nov. 2004.

[10] Y. Birk and E. Fiksman, "Dynamic Reconfiguration Architectures for Multi-context FPGAs", *International Journal of Computers and Electrical Engineering*, Volume 35, Issue 6, Nov. 2009.

[11] M. Hariyama, S. Ishihara, N. Idobata and M. Kameyama, "Non-volatile Multi-Context FPGAs using Hybrid Multiple-Valued/Binary Context Switching Signals", *In Proc. of International Conference Reconfigurable systems and Algorithms*, Aug. 2008.

[12] K. Nambaand H. Ito, "Proposal of Testable Multi-Context FPGA Architecture", *IEICE Transactions on Information and Systems*, Volume E89-D, Issue 5, May. 2006.