A Networked, Lightweight and Partially Reconfigurable Platform

Pierre Bomel, Guy Gogniat, Jean-Philippe Diguet

LESTER, Université de Bretagne Sud, CNRS FRE 2734, Lorient, France
{pierre.bomel, guy.gogniat, jean-philippe.diguet}@univ-ubs.fr

Abstract. In this paper we present a networked lightweight and partially reconfigurable platform assisted by a remote bitstreams server. We propose a software and hardware architecture as well as a new data-link level network protocol implementation dedicated to dynamic and partial reconfiguration of FPGAs. It requires a network controller and much less external memories to store reconfiguration software, bitstreams and buffer pools used by standard communication protocols. Our measures, based on a real implementation, show that our system can download remote bitstreams with a reconfiguration speed ten times faster than known solutions.

Keywords: partial reconfiguration, FPGA, link layer, bitstream server.

1 Introduction

FPGAs provide reconfigurable SoCs with a way to build systems on demand. In particular, Xilinx's Virtex FPGA reconfiguration can be exploited in different ways, partially or globally, externally (exo-reconfiguration) or internally (endo-reconfiguration). Virtex's dynamic and partial reconfiguration (DPR) requires additional resources to store the numerous partial configurations bitstreams. Today, researchers exploit in vast majority local FLASH and RAM memories as repositories. In the best case Huebner et al. [1] reduce up to 50% of the bitstream memory footprint with the help of a small hardware decompressor. Then, we face the migration of silicon square millimeters from FPGAs to memories. Although their low cost, when compared to FPGAs, is in favor of this migration, there are some drawbacks: 1) low reuse rate, 2) increase of number of components and PCB size, 3) reduction of MTBF and 4) impossibility to store all the possible bistreams (FPGA models, bistream locations, areas shapes) for a single IP.

In the following we review in Sect. 2 the previous DPR related works via a standard LAN. In Sect. 3 we present our contribution in terms of embedded hardware and software and propose a LAN-level protocol adapted to DPR constraints and objectives. In Sect. 4 we describe our experiments and measures about the partial reconfiguration speeds and memory footprints. Finally, in Sect. 5, we conclude.

2 Related works

Partial reconfiguration of Xilinx's FPGAs goes through the control of a configuration port called ICAP [2] (Internal Configuration Access Port). Virtex2 PRO, Virtex4 VFX and now Virtex5 contain this port. The reconfiguration peak rate announced is exactly of one byte per clock cycle: be 100 MB/s (100 MegaBytes) for 100 MHz systems. Because systems work at different frequencies, we'll express measures in bits transmitted per seconds and per MHz. The reference ICAP bandwidth of 100 MB/s becomes 8 Mb/s.MHz (8 Megabits).

Claus et al. [3] consider that, for automotive real-time video applications, the average bitstreams size is about 300 KB. Claus accepts to loose one eighth of the processing time to reconfigure. For 25 images/s, the processing time is 40 ms, and a maximum of 5 ms is devoted to endo-reconfiguration. The speed constraint is 60 MB/s. The experimental platform is a Virtex2 inside which a PPC405 executes the software managing the DPR. Claus's paper lets us think that no functional system was ready at publication time.

The XAPP433 [4] application note, describes a system built around a 100 MHz Virtex4 FX12. It contains a synthesized Microblaze processor executing the code of an HTTP server. The HTTP server downloads files via a 100 Mb/s Ethernet LAN. The protocol stack is Dunkel's lwIP [5] and the operating system is Xilinx' XMK. A 64 MB external memory is necessary to store lwIP buffers. The announced downloading rate is 500 KB/s, be 40 Kb/s.MHz. This rate is 200 times lesser than ICAP's one.

Lagger et al. [6] propose the ROPES system, dedicated to the acceleration of cryptographic functions. It is build with a 27 MHz Virtex2 1000. The processor is a Microblaze executing νClinux's code. It downloads bitstreams via Ethernet with HTTP and FTP protocols on top of a TCP/IP/Ethernet stack. For 70 KB bitstreams, DPR latencies are about 2380 ms with HTTP, and about 1200 ms with FTP. The max reconfiguration speed is about 60 KB/s, be 17 Kb/s.MHz.

Finally, Williams and Bergmann [7] propose νClinux as a RDP platform. They have developed a device driver on top of the ICAP. Junction between a remote file system and the ICAP is done at the user level by a shell command or a user program. When a remote file system is mounted via NFS/UDP/IP/Ethernet the bitstreams located there can be downloaded into the ICAP. The system is built with a Virtex2 and the processor executing νClinux is a Microblaze. No measures are provided. To have an estimation of such performances we made some measures in a similar context and got transfer speeds ranging from 200 KB/s to 400 KB/s, representing a maximum of about 32 Kb/s.MHz.

3 Contribution

In this section we present our contribution in terms of hardware architecture, software architecture and data-link level protocol for DPR. We present in details the essential points improving the speed and reducing the memory footprint.

The hardware architecture we propose (Fig. 1) relies on a 100 MHz Virtex2 PRO 30. A PPC405 core executes the DPR software. We consider that IPs

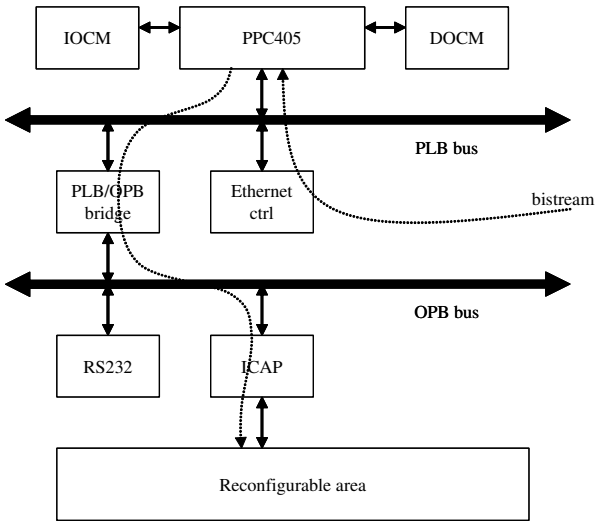

Fig. 1. Bistreams path from Ethernet to ICAP through PLB and OPB buses

communicate with the FPGA environment directly via some pads. Communication with the PPC405 and inter-IPs communication are out of the scope of this article but can be implemented with Xilinx's and Huebner's bus macros [8] and OPB/PLB wrappers as well as with an external crossbar like in the Erlangen Slot Machine of Bobda et al. [9]. We have specified with EDK, XPS and Planahead tools a system which contains a PPC405 surrounded by its minimal devices set for DPR. We have added two memories. These are respectively the IOCM (Instruction On Chip Memory) and the DOCM (Data On Chip memory). The PPC405 communicates with its devices through two buses connected through a bridge. These are the PLB bus for the faster devices and the OPB bus for the slower devices. The Ethernet PHY controller is connected to the PLB. The UART serial line, for instrumentation and trace purpose, is connected to the OPB. Finally the ICAP, connected to the OPB, manages the access and the downloading of bitstreams into the reconfigurable areas. The exo-reconfiguration is done through the JTAG port while the endo-reconfiguration is done through the ICAP.

The software architecture is a two layers one. Bottom (level 1) layer is based on the ICAP and Ethernet drivers. Top (level 2) layer handles the DPR protocol processing. They establishe a data pipeline between the remote bitstreams server and the reconfigurable areas in the FPGA. To uncouple ICAP and Ethernet we have designed a producer-consumer paradigm. The producer is the Ethernet controller and the consumer is the ICAP port. A circular buffer is asynchronously fed with packets by interrupt handlers. Packet reception occurs by bursts and the burst length is less than or equal to the half capacity of a reception packets buffer.

Each packet has a maximum size of 1518 bytes and has a maximum payload of 1500 bytes of bitstream data. The DPR protocol is executed concurrently with the interrupt handlers. It analyzes the packet content and transfers the bitstream data to the ICAP port. The bigger the burst is, the faster the protocol is.

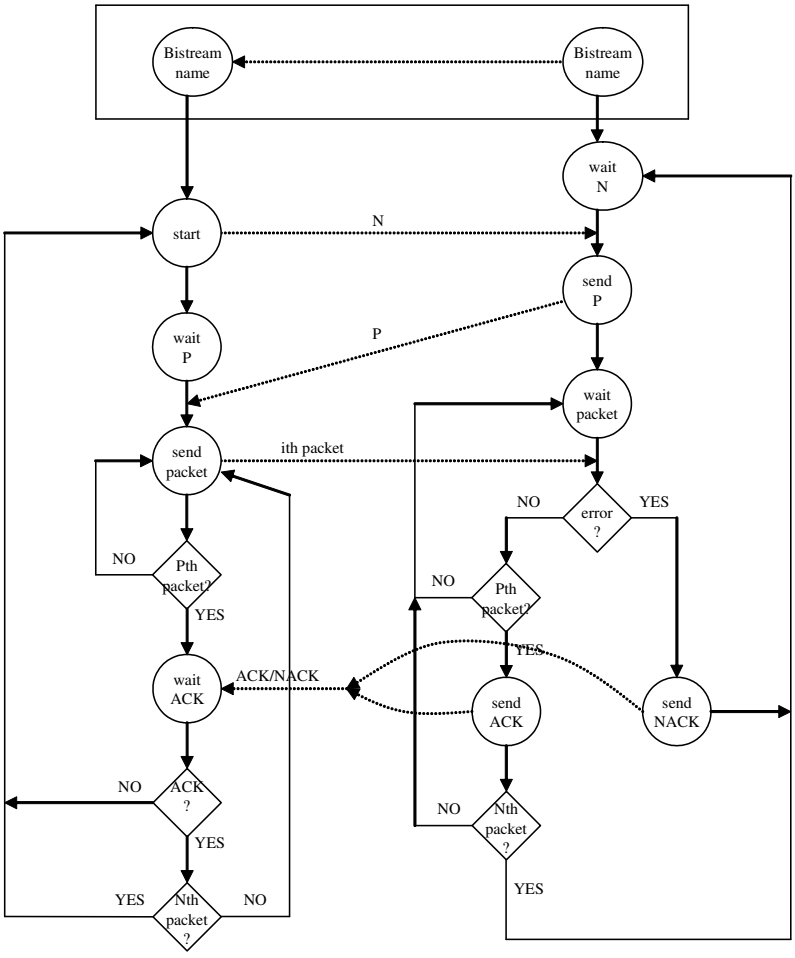

Fig. 2. RDP Protocol state machines for server (left) and target (right)

Our protocol (Fig. 2) implements a data link with error detection and flow control. It is qualified as "adaptive" because it is able to adapt itself to the memory resources available in the lightweight system at endo-reconfiguration time. Would an error be detected, the DPR is instantaneously stopped after signalization of the error to the remote bitstreams server and the reconfiguration

restarted from the beginning. The Virtex tendancy (Virtex4 and Virtex5) being to systematically reduce bitstreams sizes for partial reconfiguration, we think a bitstream level restart strategy is better than a packet level restart one. This choice is, of course, only valid for small amount of data transmitted over a very low error rate LAN, which is our context. So the Ethernet controller detects all transmission errors and the sequential numbering of packets allows detection of missing, duplicated and moved packets in the packets flow. To implement the necessary data flow regulation, we have chosen a positive acknowledge scheme every P packets. P is determined by the DPR module from the available memory space at the reconfiguration time. The DPR protocol can be used in two different modes. In "optional master" mode (top of Fig. 2) the lightweight system specifies the identity (a file name relative to the bitstream directory managed by the server) of the bitstream to send. In "slave" mode it receives the bitstream without knowledge of its function and location in the FPGA. When a transmission starts, the server sends the total number of packets, N, that will be transmitted for the full bitstream. The target answers with the number P which specifies the burst size. Immediately after this parameters negotiation, and after every positive acknowledge, the server sends a burst of P packets and waits for the next acknowledge. The finite state machine on the left describes the server's behavior and the one on the right the target's behavior. The downloading is constituted by $\lceil N/P \rceil$ bursts of maximum P packets, until the Nth packet ends the session. In case of hardware reset, both state machines come back to their waiting state. Timers on both sides help in the detection of unexpected death of an extremity of the pipeline and restart state machines if necessary.

4 Results

Our measures are based on the repetitive endo-reconfiguration of cryptography IPs like DES and triple DES producing bitstreams file sizes about 60 KB and 200 KB. Results obtained depend, as we could expect, on the producer-consumer

	Lagger [6]	Williams [7]	Xilinx [4]	RDP [authors]
Speed (Mb/s@MHz)	17	32	40	375-400
Memory (bytes)	$> 1M$	$> 1M$	$> 1M$	$< 100K$

Table 1. Comparative endo-reconfiguration speeds and memory footprints

packets buffer size allocated to the DPR protocol. So the speed depends on P. Measures establish that in both cases (60 KB and 200 KB bistreams), when the packets burst has a size greater or equal to three packets (P = 3), a maximum speed ranging from 375 to 400 Mb/s.MHz is reached and is stabilized. The size of the circular buffer being $2P + 1$, it needs room for exactely seven packets,

be 10.5 KB ($7 * 1.5KB$) only. Compared to usual buffer pools of hundredths of KB for standard protocol stacks, this is a very small amount of memory to provide a continuous DPR service. In this context our DPR protocol exhibits a sustained reconfiguration speed about 40 Mb/s. Finally, our DPR software fits into 32 KB of data memory and 40 KB of executable code memory. This memory footprint and the reconfiguration speed enable us to qualify this system as being a "lightweight DPR system". Table 1 sums up the respective speeds expressed in Mb/s.MHz and memory footprints in bytes.

5 Conclusion and future extensions

Our DPR platform shows there is still opportunities to improve LAN-level, and probably IP-level, protocols in order to provide an efficient and remote reconfiguration service (or communication service as well) over a standard network. Our implementation exhibits an order of magnitude gain in speed when compared to related works.

References

1. 'Real-time Configuration Code Decompression for Dynamic FPGA Self-Reconfiguration', Michael Hubner, Michael Ullmann, Florian Weissel, Jurgen Becker, Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04), 2004.
2. 'A lightweight approach for embedded reconfiguration of fpgas'. B. Blodget, S. McMillan, P. Lysaght, Proceedings of Design, Automation and Test in Europe (DATE'03), 2003.
3. 'Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System', Christopher Claus, Johannes Zeppenfeld, Florian Muller, Walter Stechele, DATE 2007.
4. 'Web Server design using MicroBlaze Soft Processor', Xilinx, XAPP433, October 2006.
5. 'lwIP', Adam Dunkels, Computer and Networks Architectures (CNA), Swedish Institute of Computer Science, http://www.sics.se/ adam/lwip/.
6. 'Self-Reconfigurable Pervasive Platform For Cryptographic Application', Arnaud Lagger, Andres Upegui, Eduardo Sanchez, Proceedings of International Conference on Field Programmable Logic and Applications (FPL'06), 2006.
7. 'Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip', John Williams, Neil Bergmann, Proceedings of the 2004 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'04), ISBN 1-932415-42-4, 2004.
8. 'Real-Time LUT-based Network Topologies for Dynamic and Partial FPGA Self-Reconfiguration', M. Huebner, T. Becker, J. Becker, 17th Symposium on Integrated Circuits and Systems Design (SBCCI'04), September 2004.
9. 'The Erlangen Slot Machine: Increasing Flexibility in FPGA-Based Reconfigurable Platforms', C. Bobda, M. Majer, A. Ahmadinia, T. Haller, A. Linarth, J. Teich, Journal of VLSI Signal Processing Systems, Volume 47, Issue 1 (April 2007), Pages: 15-31, ISSN:0922-5773.