Ultra-Fast Downloading of Partial Bitstreams through Ethernet

Pierre Bomel, Jeremie Crenne, Linfeng Ye, Jean-Philippe Diguet, and Guy Gogniat

Lab-STICC, Université de Bretagne Sud, UEB, CNRS UMR 3192, Lorient, France
{pierre.bomel,jeremie.crenne,linfeng.ye,jean-philippe.diguet, guy.gogniat}@univ-ubs.fr

Abstract. In this paper we present a partial bitstreams ultra-fast downloading process through a standard Ethernet network. These Virtex-based and partially reconfigurable systems use a specific data-link level protocol to communicate with remote bistreams servers. Targeted applications cover portable communicating low cost equipments, multi-standards software defined radio, automotive embedded electronics, mobile robotics or even spacecrafts where dynamic reconfiguration of FPGAs reduces the components count: hence the price, the weight, the power consumption, etc... These systems require a local network controller and a very small internal memory to support this specific protocol. Measures, based on real implementations, show that our systems can download partial bistreams with a speed twenty times faster (a sustained rate of 80 Mbits/s over Ethernet 100 Mbit/s) than best known solutions with memory requirements in the range of 10th of KB.

Keywords: partial reconfiguration, FPGA, link layer, bitstream server, ultra-fast downloading, Ethernet.

1 Introduction

Field programmable gate arrays (FPGAs) are now being integrated into applications ranging from handheld devices to space-based applications. These heterogeneous reconfigurable platforms provide a flexible way to build highly reusable systems on demand. Today, runtime partial reconfiguration can potentially reduce the number of devices or the device size, thereby reducing size, weight and power consumption. Systems requiring dynamicaly a subset of their functionnalities can take advantage of this partial reconfigurability because it allows swapping of hardware accelerators "on demand". In this context Xilinx's Virtex FPGA partial reconfiguration can be exploited in different ways, partially or globally, externally (exo-reconfiguration) or internally (endo-reconfiguration).

Virtex's dynamic and partial reconfiguration (PR) requires additional resources to store the numerous partial configuration bitstreams. Researchers

M. Berekovic et al. (Eds.): ARCS 2009, LNCS 5455, pp. 72–83, 2009.

exploit in vast majority local FLASH and RAM memories as bitstreams 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. Recently, Haiyun and Shurong [2] proposed a new adaptive LZW compression algorithm demonstrating bitstreams size gains about 43%. 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. First, their reuse rate can be extremely low, since these memories could be used just once at reset in the worst case.
2. Second, the balance in terms of global silicon square millimeters, component number reduction, PCB area, static power consumption and MTBF, is negative.
3. Third, for a single function to implement, the space of possible bitstreams can be extremely large and become bigger than local memories. This is particularly true for embedded, high-volume and low-cost systems. There are three combinatorial explosion factors namely:
   (a) the FPGAs types with their numerous devices, families, sizes, packages and speed grades variations,
   (b) the number of possible configurations depending on shape and placement of the IP on the 2D FPGA grid,
   (c) and the natural commercial life of the IP producing regularly new versions and updates.

In this paper we are presenting an evolution of a previous work [3] where we proposed a specific and quite simple protocol for partial reconfiguration over Ethernet. Through a better balanced harware/software partitionning of our hardware architecture and, with no change at protocol level, we have been able to double the sustained speed over a standard 100Mb/s (Mb = Mega bits) Ethernet local Network. The experimental results obtained prove that our systems can reach reconfiguration speeds twenty times faster than today's ones. Moreover their PR software memory footprint is small enough (a few 10th of KB, KB = Kilo Bytes) to be stored in FPGA internal memory hard blocks.

For systems having a reconfiguration latency constraint so strong that (even with a specific protocol) they need shorter downloadings we have also proposed the concept of a hierarchy of bitstreams servers [4]. It allows remote storage of bitstreams on LAN or WAN-shared servers and local storage of bitstreams copies in external RAM. A possible instance of a three level LRU caching strategy permits to cumulate advantages of distant and local storage of bitstreams.

In the following we review in Sect. 2 the previous PR related works via a standard LAN. In Sect. 3 we present our design choices and improvements to double the past downloading speed. In Sect. 4 we describe our experiments and measures about the partial reconfiguration speeds and memory footprints. We make measures with the help of signal processing IPs representative enough of the complexity expected in such embedded systems. Finally, in Sect. 5, we conclude and explain what extensions we intend to short-term focus on.

2 Related Work

The PR community agrees on the fact that, in applicative domains with strong real-time constraints, PR latency is one of the most critical aspects in its implementation. If not brief enough, the PR interest to build efficient systems can be jeopardized. Reconfiguring times will be highly dependent upon the size and organization of the partially reconfigurable regions inside a FPGA. Virtex-2 (V2) has column-wide frames embbeded into partial bitstreams: hence V2's bitstreams are bigger than necessary. Virtex-4s (V4) and Virtex-5s (V5) have relaxed this constraint, they now allow for arbitrarily-shaped regions. They have frames composed of 41 32-bits words. The smallest V4 device, the LX15, has 3740 frames, and the largest V4 device has, the FX140, has 41152 frames. From Xilinx's datasheets, four methods of partial reconfiguration exist and have different maximum downloading speeds:

1. externally (exo-reconfiguration)
   (a) serial configuration port, 1 bit, 100 MHz, 100 Mb/s
   (b) boundary scan port (JTAG), 1 bit, 66 MHz, 66 Mb/s
   (c) SelectMap port, 8 bits parallel, 100 MHz, 800 Mb/s
2. internally (endo-reconfiguration)
   (a) internal configuration port (ICAP), V2, 8 bits parallel, 100 MHz, 800 Mb/s

Of course, peak values are only objectives and ICAP inside V4 and V5 have bigger word accesses formats (16 and 32 bits). Depending on the system designer's ability to build an efficient data pipeline from the bitstream storage (RAM, FLASH, or remote) to the ICAP, the performances will be close (or not) to the peak values. The good questions are "what latency is acceptable" for a given application and "what is its related cost" in term of system cost (added memory/peripherals components). In particular, in the field of partial reconfiguration, to be able to compare contributions, we must identify what is the average size of a partial bitstream and what is its average acceptable reconfiguration latency. Finally, because systems can run at different frequencies, we must also integrate the system frequency in the numbers. Then, we will be able to fairly compare efficiencies of the various proposals. This will be done in the following paragraphs.

Compton and Hauck [5] give a complete and global survey of the whole problematic about reconfigurable computing. Walder and Platzner [6] confirm the need for fast reconfiguration in the field of the "wearable-computing". They conclude that PR latency can be neglected if its effective latency is negligible when compared to the applications computing time. We clearly see here that the "average acceptable latency" is application dependant. Researchers investigate two strategies to reduce the PR latency. These are the systematic reduction of the bitstream files size (offline compression) and the speedup of their downloading (online decompression and optimised protocols). The first strategy relies on off-line tools and methodologies for FPGA design: "Module Based" and "Difference Based" are two design flows from Xilinx [7]. The second strategy is an

on-line approach based on embedded network services. In this paper we address the second strategy with dynamic partial endo-reconfiguration and consider the first one as necessary and complementary.

Partial and dynamic reconfiguration of Xilinx's FPGAs goes through the control of a configuration port called ICAP [8] (Internal Configuration Access Port). V2, V4 and V5 contain this port and a set of one or several PPC405 hard core processors. The ICAP port can be interfaced with "hard block" processors (PPC405) as well as synthesizable soft cores (Microblaze, PicoBlaze, etc ...) with more or less engineering work. The ICAP has been wrapped into a HWICAP component which implements around it a standard OPB interface for a cost of 150 slices and a single BRAM. With the last version of EDK (version 10.2) there is now a PLB bus version of this ICAP wrapper. The ICAP reconfiguration peak rate announced by Xilinx is exactly of one byte per clock cycle, it means 100 MB/s (MB = Mega Bytes) for systems running at 100 MHz. Because systems work at different frequencies, we'll express all measures in number of bits transmitted per second and per MHz. The reference ICAP bandwidth of 100 MB/s becomes 8 Mb/(s.MHz). From now, we will use this "figure of merit" (FOM) to compare contributions to the ICAP's peak value. Quantities will be expressed in bits, times in seconds, and frequencies in MHz.

$$FOM = quantity/(time * frequency)$$

Claus et al. [9] consider that, for real-time video applications like driver assistance, the average bitstreams size is about 300 KB. The adaptive nature of the image flow processing implies to dynamically change algorithms without loosing a single image (640x480 pixels, VGA, black and white). Under these conditions, Claus accepts to loose one eighth of the processing time for each image. With a rate of 25 images/s, the processing time is 40 ms, and a maximum of 5 ms can be devoted to endo-reconfiguration. Transmitting 300 KB in 5 ms fixes the speed constraint at 60 MB/s, which is sustainable by the ICAP. The experimental platform is a V2 inside which a PPC405 executes the software (no RTOS specified) managing the PR. Unfortunately they did not provide speed measurements at publication time.

Not strictly dedicated to PR, the XAPP433 [10] application note from Xilinx, describes a system built around a V4 FX12 running at 100 MHz. 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 [11] 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. [12] propose the ROPES (Reconfigurable Object for Pervasive Systems) system, dedicated to the acceleration of cryptographic functions. It is build with a V2 1000 running at 27 MHz. The processor is a synthesized Microblaze executing νClinux's code. It downloads bitstreams via Ethernet with HTTP and FTP protocols on top of a TCP/IP/Ethernet stack. For bitstreams

of an average size of 70 KB, PR latencies are about 2380 ms with HTTP, and about 1200 ms with FTP. The reconfiguration speed is about 30 to 60 KB/s, be a maximum of 17 Kb/(s.MHz).

Williams and Bergmann [13] propose νClinux as a universal PR software platform. They have developed a character mode device driver on top of the ICAP. This driver enables to download the content of bitstreams coming from any location because of the full separation between the ICAP access and the file system. 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 the bitstreams located there can be naturally downloaded into the ICAP. The system is built with a V2 and the processor executing νClinux is a synthesized Microblaze. The authors agree that this ease of use has a cost in term of performances and they accept it. 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). We agree that νClinux is a universal platform but we want to pinpoint its extremely low bitstream dowloading performance. Certainly extremely usefull for fast development νClinux probably does not provide fast enough downloading facilities to be used in a highly time-constraints environment. We think that further experiments should be done with a real-time "flavor" of Linux and with better written/optimized ICAP drivers.

This first part of the state of the art establishes that "Microblaze + Linux + TCP" is widely accepted. Unfortunately, best downloading speeds are far below the ICAP and network maximum bandwidth. Moreover, memory needs are in the range of megabytes, thus requiring addition of external memories. Such a gap in speed and such a memory footprint for a PR service seem to us really excessive. First, Linux and its TCP/IP stack can't run without an external memory to store the kernel code and the communication protocols buffers. Secondly, the implementation, and probably the nature (specified in the 80s for much slower and unreliable data links) of the protocols, is such that only hundredths of KB/s can be achieved on traditional LANs.

Finally, the authors [3] propose the implementation of a specific data link level protocol over a standard 100 Mb/s local Ethernet dedicated to lightweight and partially reconfigurable systems. This work has been initially implemented on a V2 running at 100 MHz. No specific RTOS is needed (but XMK is sufficient) and a set of interrupt handlers with a background task handle the data pipeline from the remote bitstreams server to the local ICAP. Sustained speed is ranging from 375 to 400 Kb/(s.MHz) for bistreams of sizes ranging from 50 to 200 KB. This FOM is 10 times greater than the best previous works's FOMs.

Table 1. Identified bistreams sizes and latencies requirements from previous works

	Claus [9]	Lagger [11]	Authors [3]
Bitstream (KB)	300	70	50-200
Latency (ms)	5	1200-2380	10

On top of [3], the authors have build a hierarchy of bitstreams servers [4] to implement an LRU caching strategy for partial bitstreams. This hierarchy, being paid the cost of cache in external memory, allows to reduce the average downloading latency of partial bistreams. Hence, for systems with a known behavour, memory needs can be estimated and allocated to build the necessary cache memory. Based on [3], the worst case downloading speed has a FOM of 375 to 400 Kb/(s.MHz). But, once partial bitstreams are loaded into RAM, their downloading speed from external memory to ICAP is rising up to 4 Mb/(s.MHz), which is ten times faster. The value of the average downloading speed depends, of course, on the cache miss frequency. The smaller it is, the higher the average value is. This value of 4 Mb/(s.MHz) cannot not be considered as a contribution to ultra-fast downloading of partial bitstreams because it does not represent a pure network downloading speed. This a mix between memory accesses and network communications.

3 Contribution

In this section we present our contribution in terms of hardware and software architectures for PR. We present in details the essential points improving the speed and reducing the memory footprint. All ICAP accesses are 8 bits wide.

This contribution does not depend on the FPGA type but rather on the architecture build inside it. Tests show that, at the same frequency of 100 MHz, FOM has similar values (not strictly equals because Ethernet controllers are not the same) on a V2 or V4 implementation. We did not make a test on V5 yet, but this will be done soon. This allows us to place the discussion on the embedded architecture rather than on the latest version of FPGA available on the market. Of course, latests versions of FPGA can bring our systems to higher frequencies but should not change the FOM which is frequency independant.

3.1 Hardware Architecture

The hardware architecture we have first choosen (Fig. 1) relied on a V2 PRO 30 running at 100 MHz on a XUP evaluation board from Xilinx. A PPC405 core executed the PR software. We considered that dynamic IPs communicated with the FPGA environment directly via some pads. Thus, the FPGA was equivalent to a set of reconfigurable components able to switch rapidly from one function to another. 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 [14] and OPB/PLB wrappers for partially reconfigurable IPs as well as with an external crossbar like in the Erlangen Slot Machine of Bobda et al. [15]. We have specified with EDK, XPS and Planahead tools a system which contained a PPC405 surrounded by its minimal devices set for PR. The PPC405 having a Harward architecture, we have added two memories to store the executable code and the data. These are respectively the IOCM (Instruction On Chip Memory) and the DOCM (Data On Chip memory). The PPC405

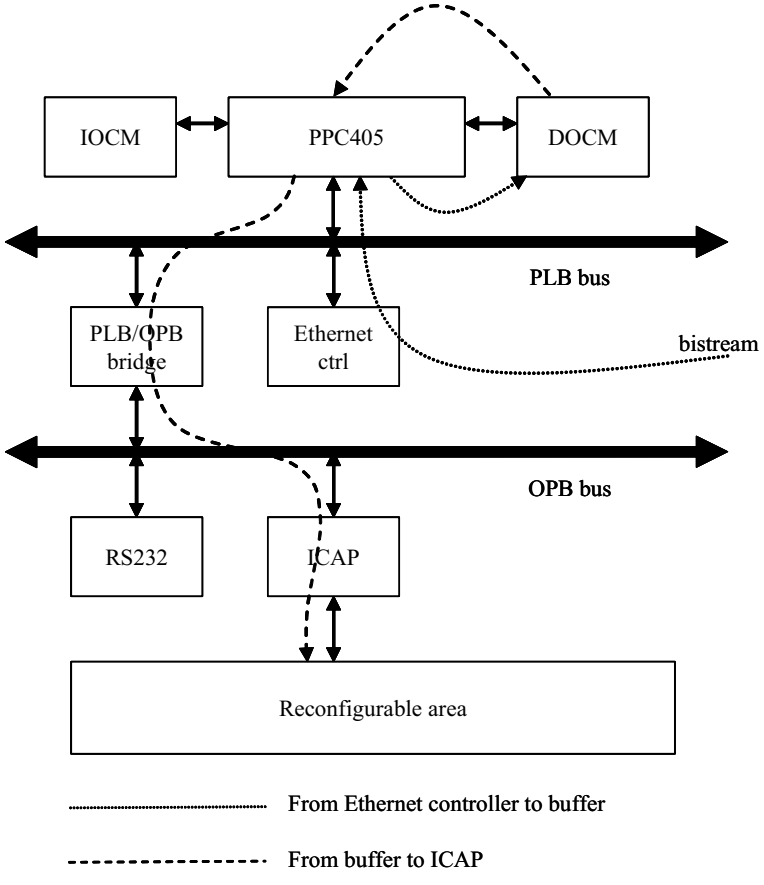

Fig. 1. Arch-1, Bistreams path from Ethernet to ICAP

communicated with its devices through two buses connected through a bridge. These were the PLB (Processor Local Bus) for the faster devices and the OPB (On Chip Peripheral Bus) for the slower devices. The Ethernet PHY controller was connected to the PLB. The UART serial line, for instrumentation and trace purpose, was connected to the OPB. Finally the ICAP, connected to the OPB, managed the access and the downloading of bitstreams into the reconfigurable areas. The full exo-reconfiguration at reset was done through the external JTAG port while the endo-reconfiguration was dynamically done through the ICAP. This architecture allowed to download partial bistreams with a FOM of 400 Kb/(s.MHz).

Measurements made later showed us that the partitionning between hardware and software was not ideal. The bottleneck coming from software. Actually, more than 90% of the processing time was spend in data transfers from Ethernet controller to circular buffer and from circular buffer to ICAP (Fig. 3).

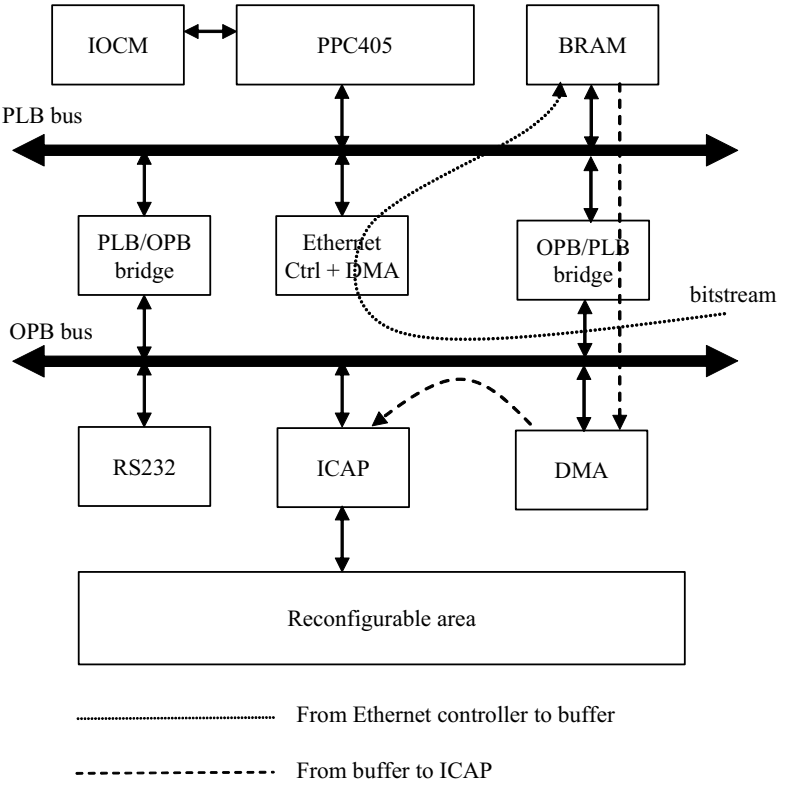

Fig. 2. Arch-2, Bistreams path from Ethernet to ICAP

The second hardware architecture we propose (Fig. 2) relies on a V4 VFX 60 running also at 100 MHz (which simplifies comparisons with first hardware harchitecture) on a ML410 evaluation board from Xilinx. This FPGA has four embbeded 10/100/1000 Mb/s Ethernet MAC controllers, among which only two are used on the ML410 board. Our architecture then uses only one of these two, configured to communicate at 100 Mb/s. Instead of relying on pure software data transfert loops executed by the PPC, we decided to use two DMAs in order to 1) transfer the data from the Ethernet controller to the packets circular buffer and 2) to transfer the data from the circular buffer to the ICAP. The first DMA is easily activated thanks to the configuration of the Ethernet controller IP available in EDK. The second DMA has to be instanciated (223 slices) and managed by the PPC itself when needed. The packets buffer, to be accessible by both DMAs cannot stay in DOCM (private to PPC) and must migrate in BRAMs located either on PLB or OPB bus. Because master accesses must be allowed for both DMA, two bus bridges (PLB/OPB and OPB/PLB bridges) must be added to allow for such data transfers. After testing on V2/XUP and V4/ML410, we obtained similar results: be a FOM about 800 Kb/(s.MHz).

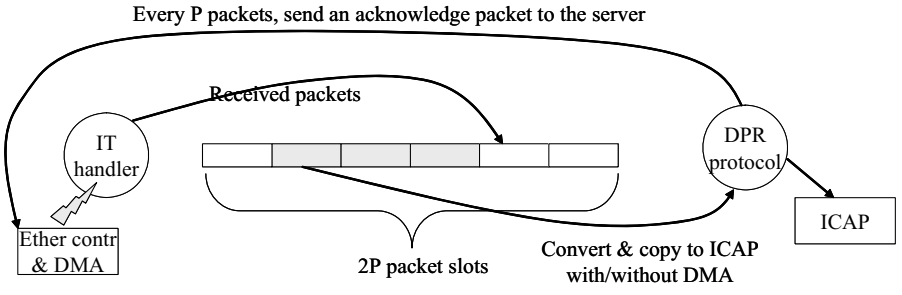

Fig. 3. Producer-consumer packets reception buffer

3.2 Software Architecture

The software architecture (Fig. 3) is based on three modules: the ICAP driver, the Ethernet driver and the PR protocol processing. Main objective is to reduce the number of software layers to cross when bitstreams are flowing from the Ethernet controller to the ICAP port. A time measurement module based on the internal hardware timer of the PPC405 and the access to the serial line via Xilinx's libc are also used and will not be commented as their use is marginal. This software establishes a data pipeline between the remote bitstreams server and the reconfigurable areas in the FPGA. We planned to reach the Ethernet maximum bandwidth of 100 Mb/s and today, with our 800 Kb/(s.MHz) FOM, we have reached a sustained rate of 80 Mb/s.

To uncouple the ICAP downloading from the Ethernet packet reception we have designed in software a producer-consumer paradigm: the producer being the Ethernet controller and the consumer being the ICAP port. A circular buffer is asynchronously fed with Ethernet packets by the Ethernet controller private DMA. Packets reception occurs by bursts: several packets are received without any data flow control feedback. The packet burst length (P) is less than or equal to the half capacity of the packets buffer. Each Ethernet packet has a maximum size of 1518 bytes and has a maximum payload of 1500 bytes of bitstream data. The PR protocol is executed concurrently with the Ethernet interrupt handlers. It analyzes the packets content and transfers the bitstream data from the buffer to the ICAP port via the second DMA programmation. The intermediate buffer sizing is a critical point in terms of performances. The bigger the burst is, the faster the protocol is. The buffer size depends on the available memory at the reconfiguration time and this scare resource can change in time. The protocol has been tailored to dynamicaly adapt its burst sizes to the buffer size, [3] gives its detailed specification.

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

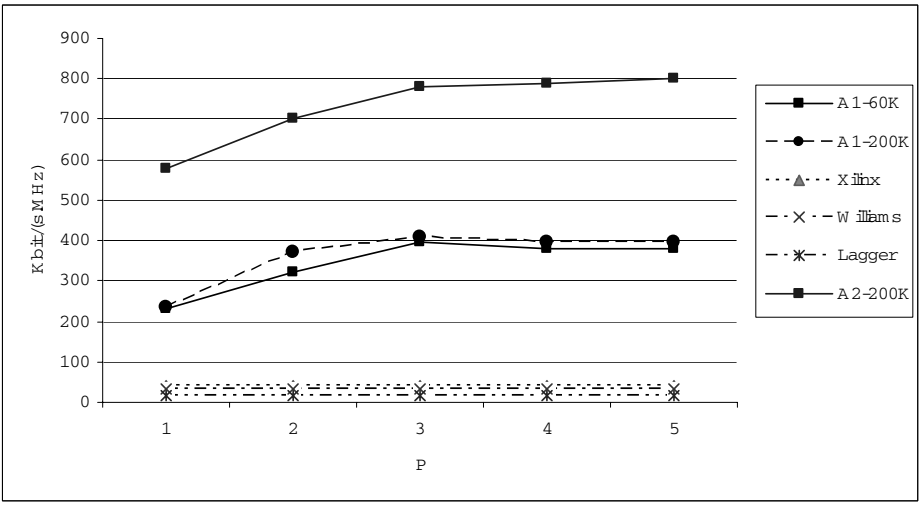

Fig. 4. Endo-reconfiguration speed vs P

KB. Results obtained (Fig. 4) depend also, as we could expect, on the producer-consumer packets buffer size allocated to the PR protocol. So the FOMs depend on P. The curves at the top (plain and dashed curves) represent respectively measured speeds for 60 KB and 200 KB bitstreams for Arch-1 and 200 KB bitstreams for Arch-2. One can establish that, in all cases, when the packets burst has a size greater or equal to three packets (P = 3), maximum speeds of 400 Mb/(s@MHz) (first architecture) and 800 Mb/(s.MHz) (second architecture) are reached and are stabilized. The size of the circular buffer being $2P$, it needs room for exactely six packets, be 9 KB ($6 * 1.5KB$) 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 PR service.

Dotted curves at the bottom represent the average speeds reached by Xilinx, Lagger and probably Williams. Our PR protocol exhibits a reconfiguration speed 80 Mb/s closer to our local 100 Mb/s Ethernet LAN limit. The gap between the reconfiguration speed and the ICAP speed is now about one order of magnitude instead of three orders of magnitude as previously. Finally, our PR software fits into 32 KB of data memory and 40 KB of executable code memory.

When compared to related works, the endo-reconfiguration speed we have reached with our ultra-fast downloading is 20 times more efficient and needs less

Table 2. Comparative endo-reconfiguration speeds and memory footprints

	Lagger [11]	Williams [12]	Xilinx [9]	Arch-1 [authors]	Arch-2 [authors]
FOM (Mb/s@MHz)	17	32	40	400	800
Memory (bytes)	$> 1M$	$> 1M$	$> 1M$	$< 100K$	$< 100K$

memory space. Table 2 sums up the respective speeds expressed in Mb/(s.MHz) and memory footprints in bytes.

5 Conclusion and Future Extensions

Our PR platform and experiments show there are still opportunities to improve LAN-level, and probably IP-level, protocols in order to provide an efficient and remote reconfiguration service standard networks. Our implementation exhibits an order of magnitude gain (X 20) in speed when compared to related works.

From here, we target implementations and protocol optimizations for future low-latency, high-bandwidth and network-reconfigurable sets of partially reconfigurable embedded systems. Would another FPGA maker provide a new configuration port, the protocol presented here could be reused "as is". Integration of our specific PR protocol on top of UDP into Dunkel's lwIP stack will be a way to promote its usage as well as the customization of UDP/IP stacks inside a real-time version of Linux or MicroC/OS-II.

Also, a software implementation running on a Microblaze soft core and a full hardware implementation of the PR protocol might be welcome when targeting systems without PPC405 hard cores. Ferivatives will be necessary when PR-based systems will be connected to other LANs like Wifi or CAN. At last, for performance purpose, 1 Gbit/s Ethernet and wider (16 bits and 32 bits) ICAP word accesses should be tested together to raise the FOM.

References

1. Huebner, M., et al.: Real-time Configuration Code Decompression for Dynamic FPGA Self-Reconfiguration. In: Proc. of the 11th Reconfigurable Architectures Workshop (RAW/IPDPS 2004), Santa Fe, New Mexico, USA, April 26-30 (2004)
2. Haiyun, G., Shurong, C.: Partial Reconfiguration Bitstream Compression for Virtex FPGAs. In: Proc. of International Congress on Image and Signal Processing (CISP 2008), Sanya, Hainan, China, May 27-30 (2008)
3. Bomel, P., Gogniat, G., Diguet, J.-P.: A Networked, Lightweight and Partially Reconfigurable Platform. In: Woods, R., Compton, K., Bouganis, C., Diniz, P.C. (eds.) ARC 2008. LNCS, vol. 4943, pp. 318–323. Springer, Heidelberg (2008)
4. Bomel, P., Gogniat, G., Diguet, J.-P., Crenne, J.: Bitstreams Repository Hierarchy for FPGA Partially Reconfigurable Systems. In: Proc. of 7th Intl. Symposium on Parallel and Distributed Computing (ISPDC 2008), Krakow, Poland, July 1-5 (2008)
5. Compton, K., Hauck, S.: Reconfigurable Computing: A Survey of Systems and Software. ACM Computing Surveys 34(2), 171–210 (2002)
6. Walder, H., Platzner, M.: Online Scheduling for Block-partitioned Reconfigurable Devices. In: Proc. of Design, Automation and Test in Europe Conference and Exposition (DATE 2003), Munich, Germany, March 3-7 2003. IEEE Computer Society, Los Alamitos (2003)
7. Xilinx XAPP290. Two Flows for Partial Reconfiguration: Module Based or Difference Based (September 2004)

8. Blodget, B., McMillan, S., Lysaght, P.: A lightweight approach for embedded reconfiguration of fpgas. In: Proc. of Design, Automation and Test in Europe Conference and Exposition (DATE 2003), Munich, Germany, March 3-7, 2003. IEEE Computer Society, Los Alamitos (2003)
9. Claus, C., Zeppenfeld, J., Muller, F., Stechele, W.: Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System. In: Proc. of Design, Automation and Test in Europe Conference and Exposition (DATE 2007), Nice, France, April 20-24 (2007)
10. Xilinx, XAPP433. Web Server design using MicroBlaze Soft Processor (October 2006)
11. Adam Dunkels. lwIP. Computer and Networks Architectures (CNA), Swedish Institute of Computer Science, `http://www.sics.se/~adam/lwip/`
12. Lagger, A., Upegui, A., Sanchez, E.: Self-Reconfigurable Pervasive Platform For Cryptographic Application. In: Proc. of the 16th Intl. Conference on Field Programmable Logic and Applications (FPL 2006), Madrid, SPAIN, August 28-30 (2006)
13. Williams, J., Bergmann, N.: Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip. In: Proc. of the Intl. Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA 2004), Las Vegas, Nevada, USA, June 21-24 (2004)
14. Huebner, M., Becker, T., Becker, J.: Real-Time LUT-based Network Topologies for Dynamic and Partial FPGA Self-Reconfiguration. In: Proc. of the 17th Symposium on Integrated Circuits and Systems design (SBCCI 2004), Ipojuca, Brazil, September 7-11 (2004)
15. Bobda, C., Majer, M., Ahmadinia, A., Haller, T., Linarth, A., Teich, J.: The Erlangen Slot Machine: Increasing Flexibility in FPGA-Based Reconfigurable Platforms. Journal of VLSI Signal Processing Systems 47(1), 15–31 (2007)