End-to-end Bitstreams Repository Hierarchy for FPGA Partially Reconfigurable Systems

Jérémie Crenne, Pierre Bomel, Guy Gogniat and Jean-Philippe Diguet

Abstract This chapter presents an end-to-end hierarchy of bitstreams repository for FPGA-based networked and partially reconfigurable systems. This approach targets embbeded systems with very scare hardware resources taking advantage of dynamic, specific and optimized architectures. The hierarchy is based on three specific levels: FPGA local repository, local network repository and wide network repository. It allows the download of partial bitstreams depending on FPGA embedded resources and gives access to local or remote servers when a complete portfolio of bitstreams is needed. Based on real implementations and measurements, results show that the proposal is functional, use a very little of hardware and software memory, and exhibits a download and reconfiguration time faster than state of the art solutions.

1 Introduction

FPGAs (Field-Programmable Gate Array) provide reconfigurable SoCs (System-On-Chip) with a way to build systems on demand. A single reconfigurable FPGA for many applications is a right answer to current critical issues in ASIC (Application Specific Integrated Circuit) design: the exploding design and production costs due to the continuous semiconductor technology density increase and the difficulty to upgrade and fix both hardware and software firmwares. Also, FPGAs hard blocks like processors, memories, DSPs (Digital Signal Processing) and high speed communication interfaces bring extreme flexibility at hardware and software levels, as well as at fine and coarse grain.

In telecommunication industry, reconfigurable wireless Universal Terminal is now

Jérémie Crenne, Pierre Bomel, Guy Gogniat, Jean-Philippe Diguet
LAB-STICC, Université Européenne de Bretagne, Lorient, France, e-mail: {jeremie.crenne, pierre.bomel, guy.gogniat, jean-philippe.diguet}@univ-ubs.fr

a well known idea which first appears in military area and became civilly popular in the 90s. This growing topic is a direct consequence on performances of FPGAs. This technology enables massive parallelism, enough computational power to realize DFE (Digital Front End) and the capability to be reconfigured at moderate power consumption [1]. Assuming that a device should support several digital mobile telephony services, digital broadcasting services, and/or digital data transfer services, it can take advantage of the partial reconfigurability. Current devices impose severe limited services due to inflexibility of their analog technology parts but tend to be bypassed by Software Defined Radio. SDR is a set of techniques that allows reconfiguration of a communication system without the need to physically change any hardware element. The underlying goal is to produce devices capable of supporting different services (multi-standard) with an adaptation of their hardware components in function of the wireless network such as GSM (Global System Mobile), GPRS (General Packet Radio Service), UMTS (Universal Mobile Telecommunications System) and WIMAX (Worldwide Interoperability for Microwave Access). In addition, they should be able to deal with wireless LAN (Local Area Network) standards like IEEE 802.11 known as WIFI (Wireless Fidelity). Delahaye et al. [2] prove the feasibility of dynamic partial reconfiguration on a heterogeneous SDR platform which provides a flexible way to build highly reusable systems on demand. Such devices require to dynamically adapt a subset of their functions in order to take all the variations in ”real-time”. Thus these systems can take advantage of the dynamic partial reconfiguration (DPR) by swapping hardware resources on demand.

Xilinx’s Virtex FPGA reconfiguration can be exploited in different ways, partially or globally and externally (exo-reconfiguration) or internally (endo-reconfiguration). In this context Virtex’s dynamic and partial reconfiguration requires additional resources to store the numerous partial configuration bitstreams. Today, researchers exploit local FLASH memories as bitstreams repository and remote file servers accessed through standard protocols like FTP (File Transfer Protocol) of NFS (Network File System). Because memory is a scare resource in low-cost, high-volume, embedded systems, we face the migration of silicon square mm from FPGAs to memories. Although low cost memories are in favor of this migration, there are some drawbacks:

- First, their reuse rate can be extremely low, since these memories could be used just once at reset in the worst case.
- Second, the balance in terms of global silicon square mm, component number reduction, PCB (Printed-Circuit-Board) area, power consumption and MTBF (Mean Time Between Failure), is negative.
- Third, for a single function to implement, the space of possible bitstreams can be huge and becomes bigger than the local memories. There are three combinatorial factors:
  - The FPGAs families with their increasing number of devices, sizes, packages and speed grades variations.

– The number of possible configurations, unfortunately depending on spatial features like shape and location of IP (Intellectual Properties) area.
– The natural commercial life of the IPs producing regularly new versions and updates.

A bitstreams repository hierarchy becomes then necessary and must communicate through adapted physical channels and network protocols with the partially reconfigurable FPGAs. All FPGA system designers want the same: the best performances at the lowest cost to download partial bitstreams into FPGAs. The performances available today range from the ones provided by local memory where the latency is smaller to the ones of a remote file server where the latency is higher (Table 1). A bitstreams repository hierarchy delivers all versions of a single IP to all the portfolio targeted FPGAs. In a typical network topology (Figure 1), this hierarchy is composed of three levels (Figure 2):

- L1: a local bitstreams cache in memory.
- L2: a fast bitstreams server located in a dedicated LAN using a data link level protocol.
- L3: a standard global slower server located anywhere and accessed via TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) based protocols.

Level	Latency	Access Type
L1	1 ms - 10 ms	local memory
L2	10 ms - 100 ms	local server
L3	100 ms - 1 sec	remote server

Table 1 Levels latencies and accesses types.

In the following, we present and describe each level in terms of software, hardware architectures, and communication protocols. We also intend to provide a specification and an optimized implementation of a minimal software layer abstracting the access to the involved hardware resources.

2 Hierarchy Level L1

Level L1 is the board level where designers glue together FPGAs and FLASH or RAM (Random Access Memory) memories. Bitstreams can be stored in memories and it is very common to use 512 MB FLASH ones. This is the most popular way to store bitstreams and build prototypes because there are many evaluation boards with FLASH readers at very low costs for Universities and researchers. But the less memories there are, the cheaper the system is to produce in high volume. L1 is

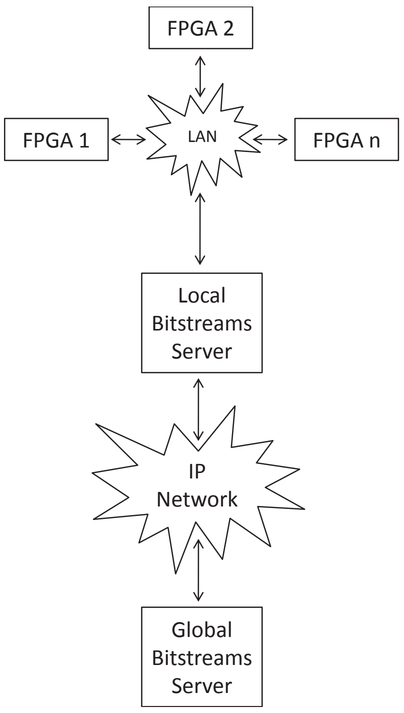

Fig. 1 LAN/WAN networking architecture. FPGAs are connected with an ad-hoc protocol to a local small bitstreams repository server. A larger global bitstreams repository server is used and can be queried with a standard IP protocol.

geographically the closest repository to the FPGA, and the one with the smallest latency. Its latency depends, of course, on the memories and bus types used. It will always be the best, when compared with networking equipments.

The PR (Partial Reconfiguration) community agrees on the fact that, in applicative domains with strong real-time constraints, PR latency is one of the most critical aspect in its implementation. If not fast enough, the PR interest to build efficient systems can be jeopardized. Reconfiguration times will be highly dependent upon the size and organization of the partially reconfigurable regions inside an 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 41x32-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 as shown in Figure 3 and have different maximum downloading speeds:

1. externally (exo-reconfiguration)

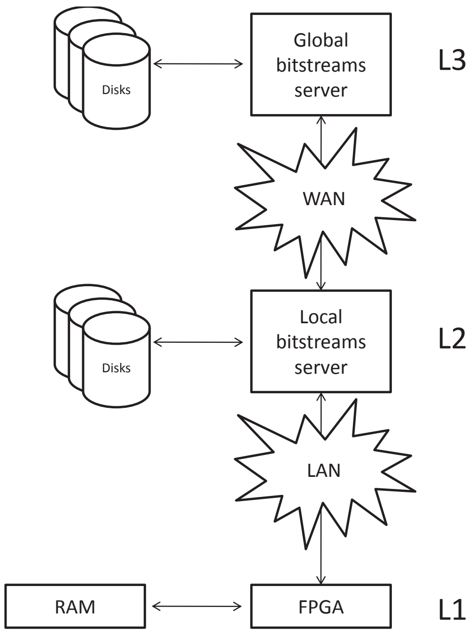

Fig. 2 Bitstreams repository hierarchy levels L1, L2 and L3. Every level can be seen as a hierarchical memory. L1 is closer to the board and acts as a cache for FPGAs. L2 server is asked when the cache doesn't contain the required bitstream. If the L2 server is not able to found the required bitstream, then the last larger L3 server is queried.

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): Internal Configuration Access Port (ICAP) [3], V2, 8 bits parallel, 100 MHz, 800 Mb/s

Of course, peak values are only objective 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 terms of system cost (added memory/peripheral 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. Virtex V2p, Virtex V4, V5 and V6 series contain ICAP port and can be interfaced with hardware IPs or hard/soft processor cores such as PowerPC, Microblaze or OpenRISC. Maximum downloading speed rate announced by the fabless in internal

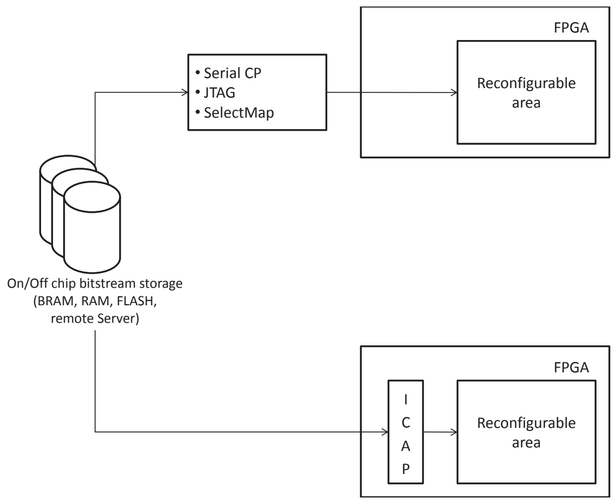

Fig. 3 Partial reconfiguration methods. The different exo-reconfiguration options are shown on the top, the endo-reconfiguration on the bottom.

reconfiguration mode is capped to a maximum of 800 Mb/s when ICAP accesses are 8 bits wide. Entire cost for hardware its implementation is 150 slices and only a single BRAM.

2.1 Cache Architecture

The use of a FLASH memory via a mass storage card or an integrated on board memory is well known and use, at least for boot time. This kind of non-volatile storage is useful for maintaining a large range of bitstreams, and when the access time is not a constraint. Without talking about writing transactions, reading is close to 500 cycles for a single 32 bits word. This value is of course dependant on the flash technology, and the associated controller. With the use of a cache, designers are able to solve this ”issue” by copying a bitstream into a faster memory which is located closer to the CPU. The problem here is that partial bitstreams are in a range of hundreths of KB when no compression is used like in [4]. Then, BRAMs memories are not the answer due to the overall available blocks which are much lower. BRAMs are a very scarce resources in FPGA. With this very short and basic analysis in mind, we propose the use of an SRAM for a cache memory. It is a tradeoff between the faster volatile memory (BRAM) and the lower non-volatile memory (FLASH). This complete software written cache, is efficient to speed up reconfiguration for some ”critical” bitstreams at a low memory cost. Of course, we could expect a significant decrease in terms of reconfiguration time if an hardware IP is doing the same job. To avoid complex logic implementation, associativity is set to be direct mapped, which means that each line of the main memory can only be registered to only one address of the cache memory. The policy about memory usage is LRU (Least Recently Used) based: the less used bitstream will be replaced by the most used ones if there is not enough memory space to store all bitstreams. The difference between a regular memory cache and L1 is that the ”line size” is rather big with L1. Xilinx’s product strategy has always been in favour of the reduction of the partial bitstreams size. Since Virtex V4 series, a real 2D partial reconfiguration can be applied and the column constraint of V2s is no more a bottleneck. Hence the average size of partial bitstreams is smaller. Anyway, the smaller the cacheline is, the less padding space will be lost when loading bitstreams which sizes are not pure multiple of this cacheline length. Defragmentation of free space in L1 can be done ”off-line” while there is no bitstream transfer ongoing. Defragmentation is not detailled here, because it has no direct impact on downloading latencies.

2.2 Hardware Architecture

All the system is built using mature tools versions when used with dynamic partial reconfiguration. Both Xilinx’s tools EDK and ISE 9.2 as well as PlanAhead 10.1 for the partial workflow are used. The hardware architecture (Figure 4) relies on a V2p 30 running at 100 Mhz on a XUP evaluation board from Xilinx. A Power PC PPC405 core executes the PR software. We consider that dynamic IPs communicate with the FPGA environment directly via some pads. Thus, the FPGA is equivalent to a set of reconfigurable components able to switch rapidly from one function to another. Communication with the PPC405 and inter-IPs communications are out of the scope of this chapter but can be implemented with Xilinx’s and Hubner’s bus macros [5] and OPB/PLB (On Chip Peripheral Bus)/(Processor Local Bus) wrappers for partially reconfigurable IPs as well as with an external crossbar like the Erlangen Slot Machine of Bodba et al. [6]. The design contains a PPC405 surrounded by its minimal devices set for PR. The PPC405 having an Harvard architecture, we add 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 communicates with its devices through two buses connected with a bridge. These are the PLB for faster devices and the OPB for slower devices. The Ethernet PHY controller is connected to the PLB and uses an integrated DMA (Direct Memory Access) to speed up transfert of incoming packets to the cache memory located in external memory. A second DMA is instanciated and managed by the PPC itself in order to copy a bitstream in cache to the ICAP removing PPC405 software copy bootlenecks. Finally the ICAP, connected to the OPB, manages the access and the downloading of bitstreams into the reconfigurable areas. The full exo-reconfiguration at reset is done using the external JTAG port while the endo-reconfiguration is dynamically done through the ICAP.

2.3 Results

Our measures are based on the repetitive endo-reconfiguration of cryptography IPs like AES (Advanced Encryption Standard), DES (Data Encryption Standard) and 3DES. To obtain the following result (Figure 5), the cache is configured to store 16 cachelines of 32 slots of 1496 bytes. With partial bitstreams of 74 KB, it is possible to store 10 bitstreams. The left side of the curve shows the sustained throughput when all bitstreams are present in cache. It exhibits an average download of bitstreams from the cache to the ICAP of about 2.1 Mb/(s.Mhz) be 210 Mb/s when the PPC405 is clocked at 100 MHz. As the cache is ”only” able to store 10 bitstreams, the throughput dramactically decreases when the number of requested bitstreams is higher than the cache capacity. On the right side of the curve, we can underline that the more the number of unique requested bitstream is, the less is the throughput be a minimum of about 50 Mb/s. When no required bitstream is found in the local cache the hierachy level L2 is automatically queried. This level is able to give access to a large number of partial bitstreams using a local server.

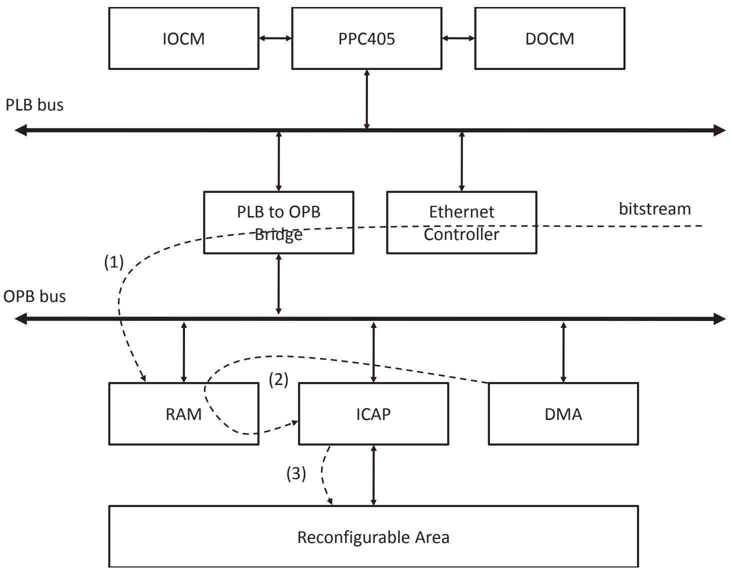

Fig. 4 L1 hardware architecure. When no bitstream in cache (RAM) is found, the incoming requested bitstream is sent by L2 repository and is copied from the Ethernet packet buffer to the cache (1). If the bitstream is present in cache, this one is copied to ICAP memory with the help of a DMA (2) and then written to reconfigurable area (3).

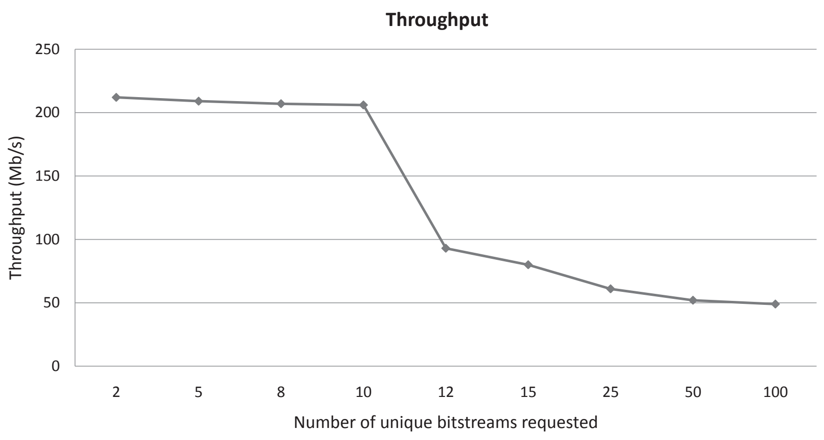

Fig. 5 Partial reconfiguration throughput. The effectiveness is related to cache capacity, replacement policy and associativity.

3 Hierarchy Level L2

Level L2 is the LAN level with a specific data link level protocol. It can provide a reconfiguration service with an average latency of 10 ms. Ethernet, in its simplest usage, is a medium sharing mechanism on top of which many protocols have been built. But it can also be seen as an excellent serial line. In terms of buying cost and ease of deployment it is a prime candidate to transfer bitstreams between close devices like our FPGA and the LAN bitstreams server.

Not strictly dedicated to DPR, the XAPP433 [7] application note from Xilinx, describes a system built around a Virtex4 FX12 running at 100 MHz. It contains a synthesized Microblaze processor executing the code of an HTTP server. The HTTP server downloads file via a 100 Mb/s Ethernet LAN. The protocol stack is Dunkel's lwIP [8] and the operating system is Xilinx's XMK (Xilinx MicroKernel). A 64 MB external memory is necessary to store lwIP buffers. The announced downloading rate is 500 KB/s, be 40 Kb/(s.MHz) performances. This rate is 200 times less than the ICAP's one. Lagger et al. [9] propose the ROPES (Reconfigurable Object for Pervasive Systems) system, dedicated to the acceleration of cryptographic functions. It is built with a Virtex2 1000 running at 27 MHz. The processor is a synthesized Microblaze executing μCLinux's code. It downloads bitstreams via Ethernet with

HTTP and FTP protocols on top of a TCP/IP stack. For bitstreams of an average size of 70 KB, DPR 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 [10] propose μCLinux as a universal DPR platform. They have developed a 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 accesses and the file system. Connection 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 naturally be downloaded into the reconfigurable area. The system is built with a Virtex2 and the processor executing the OS is a Microblaze. The authors agree that this ease of use has a cost in terms of performances and they accept it. No measures are provided. To have an estimation of such performances, some measures in a similar context have been done. A transfer speed ranging from 200 KB/s to 400 KB/s has been measured, representing a maximum performance of about 32 Kb/(s.MHz).

This state of the art establishes that ”Microblaze + Linux + TCP” dominates. 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. Linux and it’s TCP/IP stack can’t run without an external memory to store the kernel core and the communication protocols buffers. Secondly, the implementation, and probably the nature (specified in the 80s for much slower and less reliable data links) of the protocols, is such that only hundredths of Kb/s can be achieved on traditional LAN. Excessive memory footprint and maladjusted protocols are bottlenecks we intend to reduce.

3.1 Data Link Over Ethernet 100 Mb/s

Ethernet standardized IEEE 802.3 [11], created by Metcalfe and Boggs at the Xerox Parc in the 70s, is now a rich set of communication technologies to build cost effective LANs and to connect computers together. It is based on the diffusion of packets on a shared medium with collision detection (CSMA-CD). The insertion of switches and hubs (multi-ports repeaters) to simplify cabling and to improve speed and quality of services, transforms the LAN into a set of point to point links connected through LAN-level routing equipments. With this topology, two equipments connected to the same switch communicate through a quasi-private link (excepted for broadcast packets). Ethernet’s evolution is such that tenths, and even hundredths, of Mb/s are now available at very low costs with quasi-null error rates. With our repository hierarchy the bitstream server is connected to the same LAN than our system, it does not need level 3 routing toward any other LAN. Therefore we do not need IP routing and its companion protocols such as ICMP, ARP, TCP and UDP. The immediate drawback is that it does not allow the downloading of a bitstream from any other machine over Internet but remember, it will be the function of the L3.

To characterize in speed this LAN topology, a test must be done to define at which maximum speed Ethernet packets could be sent from a PC to the board. An application sends packets as fast as possible, with no specific protocol, no flow control and no error detection. Direct access to the Ethernet controller MAC level can be done easily thanks to the linux raw sockets. This test demonstrates that a speed limit of almost 100 Mb/s is reachable. It depends only on the PC and switch performances. The absence of tramsmission errors during weeks of testing proves that, in such a context, the data link quality is so high that there is probably no need to implement a complex error detection. In line with Xilinx's strategy to reduce bitstreams, an error rates estimation can be done.

3.2 Error Rates

Partial bitstreams sizes are in the range of tenth of KBytes, say a maximum of 100 KB, be 800 Kbits. Each transmitted bit has a probability p of being erroneous. With today hubs these error rates are very small, at least in the magnitude of 10^{-9}. The probability to send n bits without error is obtained with the formulae $(1-p)^n$, and the error rate is $1-(1-p)^n$ which gives the following values in (Table 2).

This shows error rates are very low for bitstreams. They are in favor of a very simple error detection and recovery strategy: a restart at bitstream level rather than at packet level. This is why the data link protocol described in [12] is a good candidate for such data transmissions. Moreover, when external pertubations occur, they will be concentrated on several adjacent packets. Their erroneous bits will have a higher correlation. Because we are not in field of RF transmissions, we do not need to decorelate error bits with Reed-Solomon like interleavers. Here it is the opposite. The more concentrated the error bits will be, the better the protocol will be because it will reject all the bitstream file and then all the error bits. Thanks to the L1 cache at FPGA level, the bitstream transmission is only required when it is not present in the local memories and the bitstreams traffic is reduced.

n	p	$(1-p)^n$	$1-(1-p)^n$
10^5	10^{-9}	0.9999	10^{-4}
10^5	10^{-10}	0.99999	10^{-5}
10^5	10^{-11}	0.999999	10^{-6}
10^6	10^{-9}	0.999	10^{-3}
10^6	10^{-10}	0.9999	10^{-4}
10^6	10^{-11}	0.99999	10^{-5}

Table 2 Estimated error rates for bitstreams downloading through network.

3.3 Hardware Achitecture

The first proposed hardware architecture (Figure 6) is similar to the one in section 3 but without DMA instanciation saving a little of FPGA logic. The design exhibits a download of partial bitstreams at 400 Kb/(s.MHz). Measurements show that the partitioning between hardware and software is not ideal. The bottleneck coming from software. Actually more than 90% of the processing time is spent in data transfers from Ethernet controller to the circular buffer and from the circular buffer to the ICAP controller.

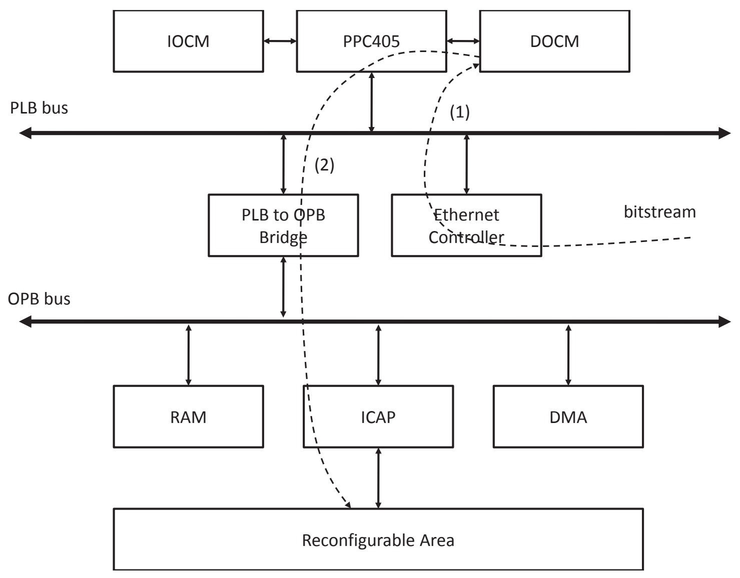

Fig. 6 L2 architecture without DMAs. The incoming bitstream is copied from Ethernet controller to DOCM memory (1) with the help of the Power PC processing time (2). The PPC writes the bitstream to the ICAP memory (2) ready to be written into the reconfigurable area.

The second hardware architecture proposed (Figure 7) relies on a V4 VFX 60 running at 100 MHz on a ML410 evaluation board from Xilinx. This FPGA has four embedded 10/100/1000 Mb/s Ethernet MAC controllers, among which only two are used on the ML410 board. Our architecture then used only one of these two, configurated to communicate at 100 Mb/s. Instead of relying on pure software data transfert loops executed by the PPC, two DMAs are running in order to:

1. Transfer the data from the Ethernet controller to the circular buffer.
2. Transfer the data from the circular buffer to the ICAP.

The packets buffer, to be accessible by both DMAs cannot stay in DOCM (private to PPC) and must migrates in BRAMs located either on PLB or OPB bus.

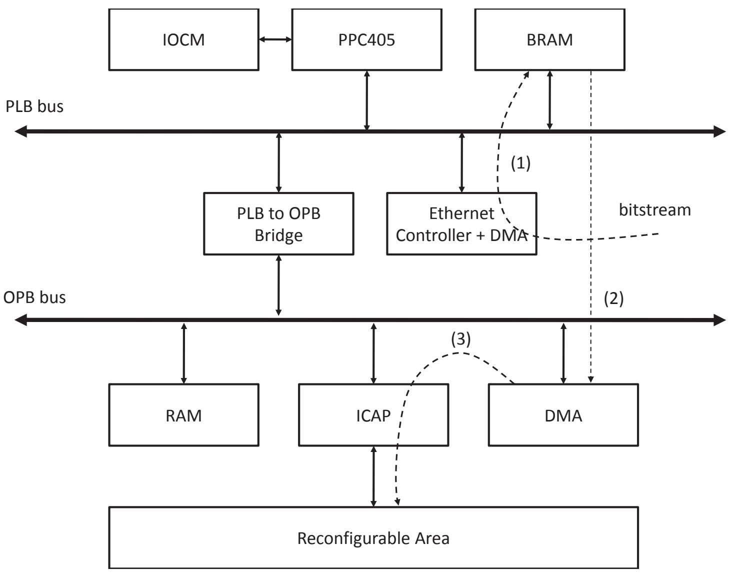

Fig. 7 L2 architecture with DMAs. The incoming bitstream is copied to a BRAM via a DMA integrated to the Ethernet controller (1). The content is then sent to the ICAP memory (2) with another DMA IP, and written into the reconfigurable area (3).

Because master accesses must be allowed for 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 obtain similar results be a download of bitstreams at 800 Kb/(s.MHz).

3.4 Software Achitecture

The software architecture is based on three modules : the ICAP driver, the Ethernet driver and the PR protocol processing (Figure 8). Thein 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 can plan to reach the Ethernet maximum bandwidth of 100 Mb/s and today, with our 800 Kb/(s.MHz), we reach a sustained rate of 80 Mb/s. To uncouple the ICAP downloading from the Ethernet packet reception we can design 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. The intermediate buffer sizing is a critical point in terms of performances. The bigger the burst is, the faster the protocol. 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 dynamically adapt its burst sizes to the buffer size, [12] gives its detailed specification.

3.5 Results

Results obtained (Figure 9) depend also, as we could expect, on the producer-consumer packets buffer size allocated to the PR protocol. The curves at the top represent respectively from the bottom to the top, measured speeds for the first architecture for the second one. 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 x $1.5KB$) 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. Flat lines curves at the bottom, represent the average speeds reached by Xilinx, Lagger and probably Williams. Our PR protocol exhibits a reconfiguration speed of 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 fast downloading is 20 times more efficient and needs less memory space.

4 Hierarchy Level L3

An ad-hoc specific data link is useful when no IP routing is required and only a little amount of hardware and software resources is available. However it is necessary to be able to download a bitstream from any machine. The use of the standard network architecture TCP/IP fits perfectly when a remote reconfiguration is necessary. Level L3 is the WAN (Wide Area Network) level where the latency is about 100 ms because of its geographic position which is the farthest.

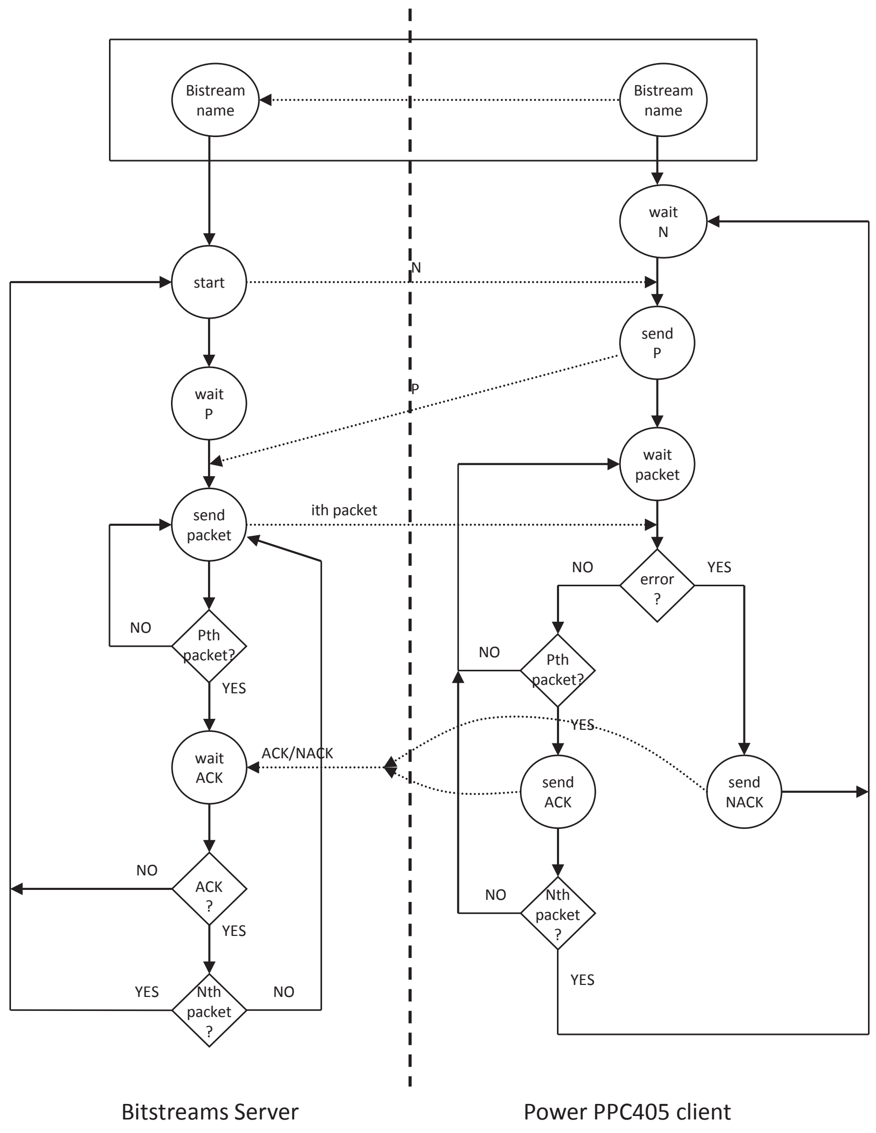

Fig. 8 Software dynamic partial reconfiguration protocol. Server and client software DPR protocol are respectivily represented on the left and right side of the figure.

4.1 Common Used Transport Protocols

Rind et al. [13] describe choices for TCP (Transport Communication Protocol) over UDP (User Datagram Protocol) and vice-versa related to speed, numbers of mo-

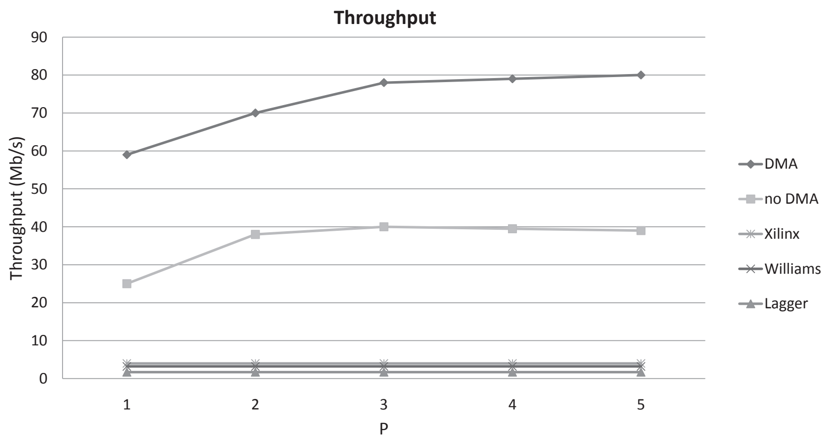

Fig. 9 Endo-reconfiguration throughput VS burst size P.

bile devices and link capacity (bandwidth) metrics. Results are given in terms of throughput via a network simulator. It shows that TCP is giving better performance when minimum number of mobile devices are connected to a WLAN (Wireless LAN) and clearly setup that faster moving nodes are highly disturbing packets transmissions. UDP is found better if it is possible to bear little loss of packets. Consequently it is a first choice protocol for fast delivery of data. Uchida [14] presents an hardware-based TCP processor for Gigabit Ethernet which requires only one Ethernet PHY device. The circuit is small enough to be implemented on a single FPGA, with an announced 949 Mb/s throughput in both emission and reception directions. It has to be carefully compared with other systems. With this approach, no high traffic over the network is considered and TCP congestion control is well known to be designed and optimized for wired networks. International IEEE 802.11 [15] describes characteristics liable to a wireless LAN (WLAN). It makes possible to build broadband wireless local networks and in practice allows to link computers, laptops, PDAs, communicating devices and other peripherals, indoor or outdoor with ranges, speeds and modes depending on numerous revisions of the standard from 802.11a to 802.11s. Wireless is sometimes the only possibility in applications where it is required to have a great mobility. In industry, reduction of wiring proves its pertinence: costly to install, to repair, imposing strict placements limitation. Compared to Bluetooth, the WIFI main strength stands in its higher throughput and range. As the system we target will be using a WIFI link and thus is limited to a much lower throughput, very high Gigabit transfert rate is oversized. Ploplys [16] et al. perform a study where ”wireless” UDP is used for real-time performance in control. Loss of data is well defined, explained and evaluated based on many factors such as range, environmental obstacles, computational loads and increased network traffic. Existing work establishes that TCP is vastly employed in LAN topology and UDP in WLAN. The use of UDP is natural when targeting wireless handled devices. UDP is also the most suitable standard for systems with a high latency and needs by nature, a shorter communication time.

4.2 TCP/IP Architecture Model

TCP/IP [17] (Transmission Control Protocol/Internet Protocol) is a networking reference framework used for developing networking applications. This model is usually described as a four-layered architecture as shown in Figure 10. For a transmission, data are sent from layer 4 to layer 1, and vice-versa for a reception. In the following lines, we place the discussion on layer 1 and 3.

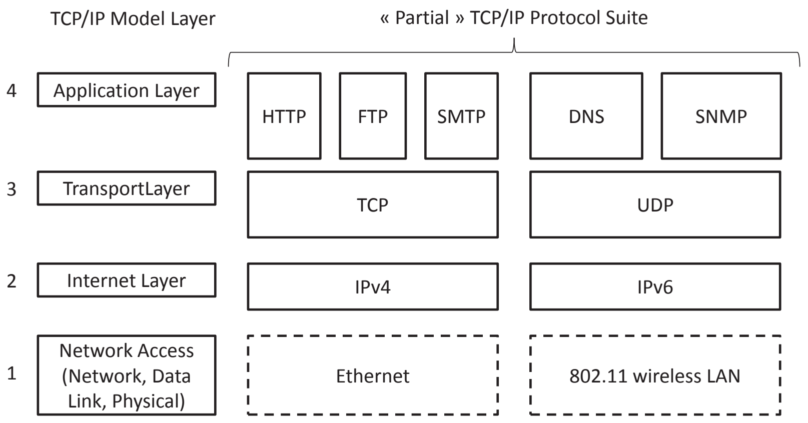

Fig. 10 The five-layered TCP/IP architecture model.

Today, IP protocols are adapted to low latency and high bandwidth data transfers. Therefore, this second point leds us to adapt our protocol to one of the most common transport protocol (layer 3). TCP [18] is used in many non-critical applications such as HTTP, FTP and SMTP. It is connection-based and guarantees that the receiver will get exactly what the sender sent without any errors and in correct order. TCP resends the packet if it doesn't arrive correctly to the destination. To avoid congestion, TCP is cutting down speed whenever a packet is lost and re-increasing it slowly when packets are successfully transmitted. As for the most appropriate transmission protocol, we pinpoint that packets looses in TCP are attributed to congestion i.e a high traffic. Hence for a wireless environment where bit error rate is high, TCP performances are highly degraded due to its window based congestion control mechanism. UDP [19] is similar to TCP and stands in the same TCP/IP layer. Known UDP applications are DNS and SNMP. Connectionless, its difference is located in the relationship between two parties. In other words, one can send data to another and that is all. UDP doesn't provide any reception reliability, thus, there is no guarantee that a packet will arrive. However if the transmission is correct, the packet will be received without any data corruption. UDP is faster than TCP as there is no extra overhead for error-checking above the packet level. A comparison between TCP and UDP is given in Table 3.

Protocol	Complexity	Speed	Architecture	Caveats
UDP	Low	High	Broadcast Client/Server	Unreliable, String data
TCP	Average	Low	Client/Server	String data

Table 3 Comparison of TCP and UDP protocols [20].

With this table and previous studied arguments, UDP has been found to be the most acceptable protocol for WLAN bitstream diffusion. From this point, we assume the use of UDP over the network.

4.3 Software Architecture

The architecture workflow differentiates the software running onto the server and the software onto the FPGA. On the left hand, the server executes a console application eventually hooked to a front-end executable. On the right hand, onto the client FPGA, the processor runs an executable built based on a TCP/IP stack.

4.3.1 lwIP as a TCP/IP networking stack

Instead of developping a networking library from scratch and in order not to reinventing the wheel, we choose an open source TCP/IP stack designed for embedded systems: lwIP. Directly available in EDK, lwIP [8] is an implementation under BSD licence of the TCP/IP stack with RAM usage friendly in mind. It was initially written by Adam Dunkels of the Swedish Institute of Computer Science and is now maintained by several developers headed by Leon Woestendberg and hosted by Savannah. The porting proposed by Xilinx in EDK is robust (both Microblaze and PPC can be used without any problems), with a lot of parameters configurable at compile time that can be tuned to tailor an architecture according to the requirements. lwIP is also featuring a quite exhaustive characteristics list and can be run with an underlying OS or not.

Our first approach was to tailor lwIP to use UDP only as we don't need another protocol. To ensure packets producer-consumer paradigm, lwIP stack uses a pool of buffers. This pool is a critical point in terms of performances and has to be well scaled. Default setting value for this segment is close to 800 KB large be 512 packets of 1528 bytes. With that consideration we found native lwIP parameters to be oversized in EDK. The absence of transmission errors during days of testing, which consists of sending and receiving data as fast as possible and checking right packet order, proves that reducing it to 100 KB is pertinent without interfering overall performances. Next, we tried to unload the processor which is running the software from heavy computations. To achieve this goal, an option called UDP checksum offload could be set. It enables the network adapter to compute the UDP checksum on transmission and reception, saving the CPU time for doing it. The gain in terms of throughput is significant (x2) when the PPC is clocked at 100 MHz and can't handle too much computation.

4.3.2 Software DPR Protocol

We are now describing our protocol by first introducing the chosen frame's structure. It is always constituted of two fields. The first one is reserved for the frame number and is 4 bytes long. The second is the data, i.e. the bitstream. The allowable packet size relies directly to the frame format we use: DIX Ethernet Version II frame format. The total minimum size of one Ethernet frame is 1528 bytes. It includes 12 bytes of inter-frame spacing and 8 bytes of preamble. From these 1518 bytes, 4 are for Frame Check Sequence (FCS). Another 6 are for destination Ethernet address, 6 are source Ethernet address, and 2 are the type, leaving 1500 bytes for user data in each frame. Lastly, 20 bytes go to an IP header, and 8 to the UDP header, leaving a maximum of 1472 bytes for data in each frame. Bitstreams are generally bigger than this maximum so it has to be fragmented before emission. Packet's transmissions are synchronized using a classic end-to-end handshake for ensuring correct data transmission. The protocol is able to work in two modes: slave and master (Figure 11). In master mode, the FPGA is responsible for asking the LAN server a partial bitstream. In slave mode, it reacts to the server requests and is forced to update itself. Obviously, obtaining a maximal reconfiguration throughput must be considered with care. Safety concerning the write of a partial bitstream to the reconfigurable area is necessary in partial reconfiguration context. A loss of a packet will result in an incomplete form of data reception, so on an impossibility of writing the complete partial bitstream into the reconfigurable area. Manifestly, it will lead to an unpredictable behavior. To avoid this, a frame number helps to know if a packet is missing or a wrong received order reception occurs. In addition, before writing to ICAP, a CRC (Cyclic Redundancy Check) is done to be sure that everything was fine during the transmission and every packets was received correctly any error.

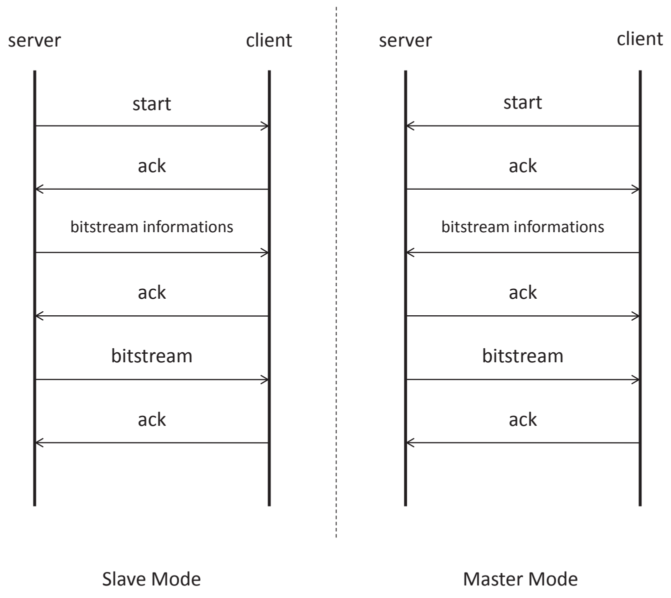

Fig. 11 Slave and master mode protocol description.

4.4 Hardware Architecture

The hardware architecture is completely similar to the one in section 3. However the dynamic partial reconfiguration protocol as to be detailed in depth. A packet reception is divided into four steps (Figure 12). It goes through internal Ethernet FIFO to the reconfigurable area:

1. First, an Rx interruption occurs. Received packet is stored into an internal Ethernet Controller FIFO.
2. Second, FIFO buffer is copied to RAM where memory pool (circular buffer) allocated by lwIP is available.
3. Third, Memory pool content is transferred to ICAP BRAM.
4. Finally, ICAP BRAM is written to the reconfigurable area.

To ensure that our software/hardware partitioning is the best, some evaluations are done, based on four basic architectural options:

1. With processor data cache disable.
2. With processor data cache enable.
3. With processor data cache disable and DMA used to transfer memory pool content to ICAP BRAM.
4. With processor data cache enable and DMA for transfering.

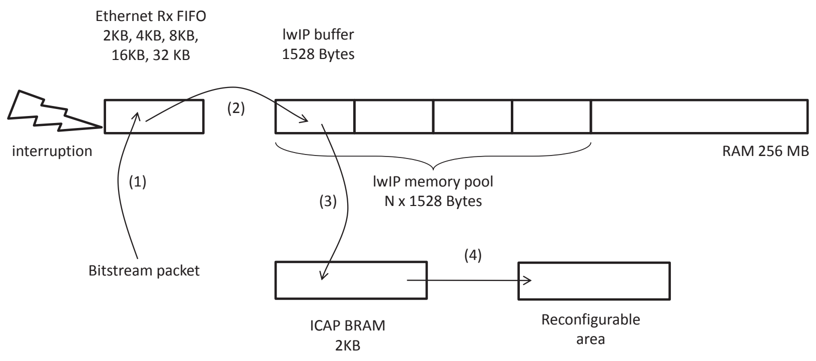

Fig. 12 Bitstream path from Ethernet to ICAP.

In all these cases, the PPC405 instruction cache is activated and set to 16 KB large (8 BRAMs). When enabled the data cache is also 16 KB large. Table 4 demonstrates that software data copy with data cache enable is the best setup. This can be explained because EDK's DMA engine has no internal buffering, and doesn't do burst transfers. For processor without instruction cache, it might make sense to add a DMA, otherwise the inner loop of the optimized memory copy would be in cache and be executed at 2 cycles per instruction. The limiting factor will become the OPB latency (reading/writing from/to the OPB RAM). Indeed, when a data cache is enabled and as the processor exhibits cache coherency anomalies, it has to remain clean. It is the responsibility of the developer to ensure that any buffers in cache which are passed to the DMA are flushed from it. In addition the cache has to be invalidated prior to using any buffers resulting of a DMA operation. That is why we decided to rely on pure software concerning all data transfers and mainly between lwIP memory pool buffers and ICAP's BRAM. Moreover, this saves some hardware resources and decreases significant overhead due to the OPB to PLB bridge as well as avoiding additional managements by the PPC.

	Cache Enable	Cache Disable	Cache Disable + ICAP DMA	Cache Enable + Both DMAs
throughput	-	+	-	+
hardware memory footprint	+	+	-	-
software memory footprint	+	+	-	-
overall	++	+++	-	+

Table 4 Hardware/software partitioning options results.

4.5 Results

One can establish that the optimum packet size given is always close to the Maximum Ethernet Transmission Unit (MTU), 1500 Bytes. Burst size is set to the maxi-

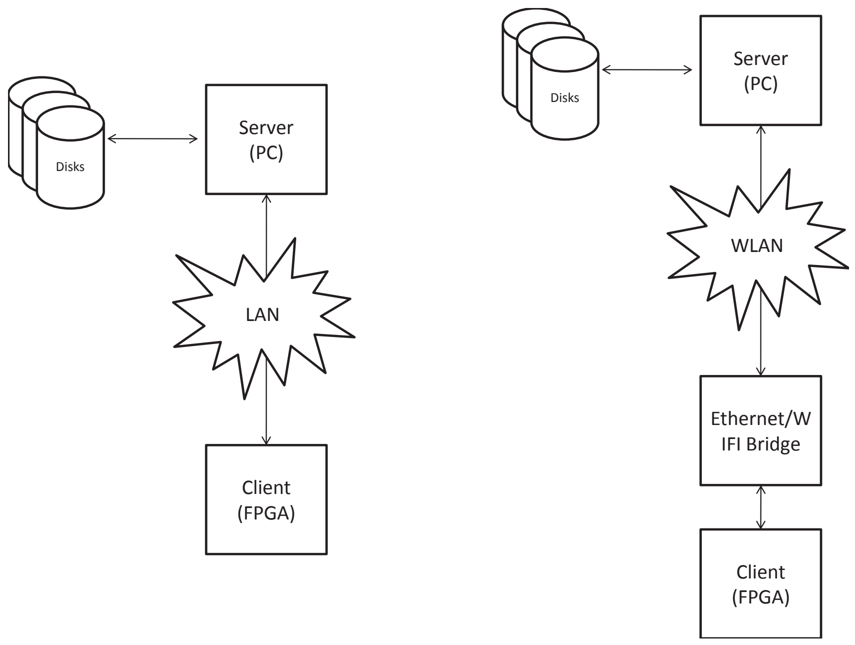

Fig. 13 LAN (on the left) and WLAN (on the right) protocol performance rating configurations.

mum, be the total number of frames needed to send one bitstream depending on it's size. The Ethernet controller FIFO is fixed to 8 KB. The server is running the application protocol in slave mode so it is initiating the transfer to the FPGA. Figure 14 sums up throughputs results as a function of the packets size and according to the network configurations given in Figure 13.

The LAN configuration must be first envisaged to ensure the global effectiveness of the protocol. It linked directly server and client machine via a crossover cable. Results obtained depend as we could expect, on transmitted packets size. We obtain a sustainable 60 Mb/s throughput with an average packet size of 1472 bytes. This high transfer throughput matches with WIFI WLAN rate where 30 Mb/s is reachable.

The WLAN configuration describes the server which is connected to the client with a wireless link. FPGA is connected to an Ethernet/WIFI bridge removing the need of specific drivers. WIFI network type is set to ad-hoc allowing two or more wireless clients to be connected each other without any central controller. In this context, a constant 30 Mb/s throughput is reachable with the same LAN scenario software memory usage. This is the maximum physically achievable as the 54Mb/s possible by the 802.11g standard is pure theory. To our best knowledge, no other works have proposed such a deal with dynamic and partial reconfiguration.

In terms of software, 100 KB are dedicated to executable code memory and 100 KB are allocated for data memory. This is 4 times less software memory compared to known previous works. In terms of hardware, all software instruction and data codes fits into BRAMs and then doesn't need any additionnal DDR RAM controller. Thus only 8% of the V2p chip, 2524 over 32383 slices, is occupied. In a Virtex V2p a slice consists of two 4-input look-up tables and two flip-flops. This size is sufficient to implement user cicuits with the protocol on a one single FPGA. This brings our entire design to a lightweight implementation.

5 Conclusion and Perspectives

The proposed bitstreams repository hierarchy shows there is still opportunities to improve cache-level, LAN level and IP-level, caching strategies and protocols in order to provide an efficient and remote reconfiguration service over a standard network. Never fully taken into consideration in previous works, our PR platform takes advantage of the three specific levels L1, L2 and L3. For L1, where the reconfiguration throughput is 200 Mb/s at 100 MHz and is already in a very good shape for futures enhancements. For L2, the reconfiguration throughput is 80 Mb/s be a multiplication factor of 20 compared to other works. For L3, the wireless technology is used with its high 30 Mb/s throughput which is the physical maximum achievable. Morevover, when used in a LAN topology, the implementation exhibits an order of

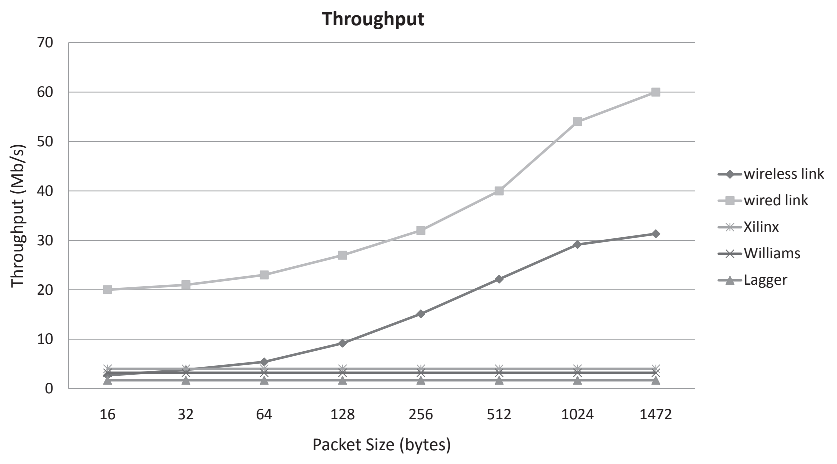

Fig. 14 Throughput curves for LAN and WLAN.

magnitude gain (x15) compared to the best previous work, be 60 Mb/s. The underlying protocol is also simple and could be reused ”as is” with a low software memory cost : 200 KB for data and instructions, and only 8% of the V2p chip is needed for the biggest design. Table 5, sums up the respective speeds and memory footprints for every level architectures presented in this chapter.

	[9]	[10]	[7]	L1 Arch1	L2 Arch1	L2 Arch2	L3 LAN	L3 WLAN
Throughput (Mb/s)	1.7	3.2	4	200	40	80	60	30
Memory (bytes)	$> 1M$	$> 1M$	$< 100K$	$< 100K$	$< 100K$	$< 100K$	$> 200K$	$> 200K$

Table 5 Comparative architecture throughputs and software memory footprints (PPC at 100 MHz).

From here we could target other implementations and optimizations for reconfigurable embedded systems. First, IP addresses are assigned statically. One could plan to do it more automatically and dynamically by implementing a DHCP (Dynamic Host Configuration Protocol) on the server depending of the context. This can allow them to be added to a network without manual intervention. Next, when targeting systems without PPC405 hard cores, it might be welcome to port the application to a synthesizable Microblaze soft core at a cost of significant slices loss unfortunately. Moreover, it is worth noting that there is no security guarantee when exchanging data between the server and the client. Confidentiality, data integrity and authenticity (secure transaction in a word) are not addressed here, but this is anyway a good path to follow. In addition, even if not a standard, RUDP (Reliable UDP) should be investigated for being a protocol vastly employed for real time performances. As Virtex V2p is now considered as an efficient but sometimes deprecated part due to tools incompatibility, migrating to a Virtex V5 or V6 is becoming a must be. These should deliver smaller partial bitstreams, thus smaller reconfiguration times, as well as a reduction of FPGA slices consumption. Last and not least when talking about network, Quality of Service (QoS) is essential for both LAN and WLAN. Number of works [21], [22] include an additional software or hardware reliability layer over UDP to ensure correct data by implementing some simple algorithms. It could be a great idea to focus onto such methods where hostile environments could lead to packet looses.

References

1. Becker, M., Hubner, M., Ullmann, M. : Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations (2003)
2. Delahaye, JPh, Gogniat, G., Roland, C., Bomel, P. : Software radio and dynamic reconfiguration on a DSP/FPGA platform (2004)
3. Xilinx ICAP http://forums.xilinx.com/xlnx/attachments/xlnx/elinux/494/1/opb_hwicap.pdf
4. Hubner, M., Ullmann, M., Weissel, F., Becker J. : Real-time Configuration Code Decompression for Dynamic FPGA Self-Reconfiguration (2004)
5. Hubner, M., Becker, T., Becker, J. : Real Time LUT-based Network Topologies for Dynamic and Partial FPGA Self-Reconfiguration (2004)
6. Bodba, C., Majer, M., Ahmadinia, A., Haller, T., Linarth A., Teich, J. : the Erlangen Slot Machine: Increasing Flexibility in FPGA-Based Reconfigurable Platforms (2007)
7. Xilinx. Xapp433. Web server design using microblaze soft processor (2006)
8. Dunkels, A. : lwIP http://www.sics.se/ãdam/lwip/
9. Lagger A., Upegui, E., Sanchez, E. : Self Reconfigurable Pervasive Platform For Cryptographic Application (2006)
10. Williams, J., Bergmann, N. : Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip (2004)
11. Ethernet. Carrier sense multiple access with collision detection (csma/cd) access method and physical layer. IEEE Standard 802.3
12. Bomel, P., Gogniat, G., Diguet, JPh : A Networked Lightweight and Partially Reconfigurable Platform (2008). Patent FR 08 50641 - N/R BFF 08P0055
13. Rind, A.R., Shahzad, K., Qadir, M.A. : Evaluation and comparison of tcp and udp over wired-cum-wireless lan (2006)
14. Uchida, T. : Hardware-based tcp processor for gigabit ethernet (2007)
15. WIFI. Wireless lan medium access control (mac) and physical layer (phy) specifications
16. Ploplys, N.J., Alleyne, A.G. : Udp network communication for distributed wireless control (2003)
17. RFC1122. Requirements for internet hosts - communication layers (1989)
18. RFC793. Transmission control protocol (1981)
19. RFC768. User datagram protocol (1980)
20. National Instruments : Building networked applications with the lawindows /cvi udp suppoort library (2009)
21. Grewal, J., DeDourek, J.M. : Provision of QoS in wireless networks (2004)
22. Raknet www.jenkinssoftware.com