Bitstreams Repository Hierarchy for FPGA Partially Reconfigurable Systems

Pierre Bomel, Jean-Philippe Diguet, Guy Gogniat, Jeremie Crenne
LAB-STICC, Université Européenne de Bretagne, CNRS UMR 3192, Lorient, France
{pierre.bomel, jean-philippe.diguet, guy.gogniat, jeremie.crenne}@univ-ubs.fr

Abstract

In this paper we present a hierarchy of bitstreams repositories for FPGA-based networked and partially reconfigurable systems. These systems target embedded systems with very scarce hardware resources taking advantage of dynamic, specific and optimized architectures. Based on FPGA integrated circuits, they require a single FPGA with a network controller and less external memories to store reconfiguration software, bitstreams and buffer pools used by today's standard communication protocols. Our measures, based on a real implementation, show that our repository hierarchy is functional and can download bitstreams with a reconfiguration speed ten times faster than known solutions.

1. Introduction

FPGAs provide reconfigurable SoCs 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 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 and high speed communication interfaces bring extreme flexibility at hardware and software levels, as well as at fine and coarse grains. Xilinx's Virtex FPGA reconfiguration can be exploited in different ways, partially or globally, externally (exo-reconfiguration) or internally (endo-reconfiguration). In this context Virtex's dynamic and partial reconfiguration (DPR) requires additional resources to store the numerous partial configurations bitstreams. Today, researchers exploit local FLASH memories as bitstreams repositories and remote file servers accessed through standard protocols like FTP or NFS. Because memory is a scare resource in low-cost, high-volume, embedded systems, Huebner et al. reduce up to 50% of the bitstream memory footprint with the help of a small hardware decompressor [1]. Then, we face the migration of silicon square mm from FPGAs to memories. Although memories's low cost is 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 area, power consumption and MTBF, is negative.
- Third, for a single function to implement, the space of possible bitstreams can be extremely large and become bigger than the local memories. There are three combinatorial explosion factors indeed:
  - the FPGAs families with their numerous devices, sizes, packages and speed grades variations,
  - the number of possible configurations, unfortunately depending on spatial features like shape and location of IP area on the 2D grid,
  - and the natural commercial life of the IPs producing regularly new versions and updates.

Regarding this, we are convinced a bitstreams repository hierarchy becomes necessary and must communicate through adapted physical channels and network protocols with the partially reconfigurable FPGAs. These repositories will deliver all versions of a single IP to all the portfolio of targeted FPGAs. The necessary hierarchy is a three levels one (Fig. 1):

- L1 a board local bitstreams cache in memory,
- L2 an extremely rapid bitstreams server located in a dedicated LAN using a data link level protocol,
- L3 a standard global "slower" bitstreams server located anywhere else and accessed via TCP or UDP based protocols.

While levels L1 and L3 have already been covered by research works and standard software and protocols, level L2 is actually totally "unexplored" for partial reconfiguration of FPGAs.

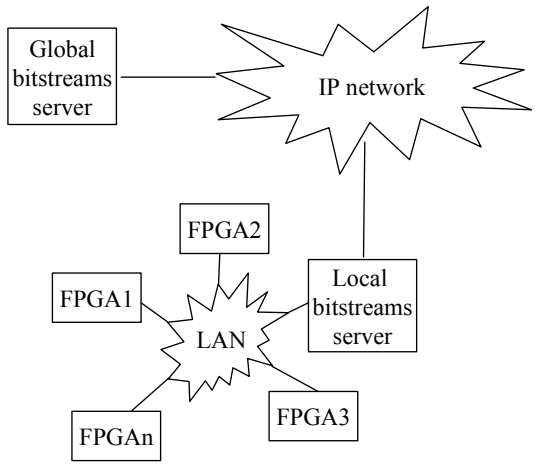

Figure 1 – LAN/WAN Networking architecture

In this paper we propose a bitstreams repository hierarchy and focus on the current trend of embedded systems to be naturally connected to standard networks. We present under which conditions and with which performances this repository hierarchy allows us to build a DPR lightweight system. The experimental results obtained at L2 level prove that these systems can reach reconfiguration speeds ten times faster than today's ones at L3 level. Moreover the DPR software memory footprint is small enough to be stored in FPGA internal memory hard blocks. When the reconfiguration delay is acceptable for a given application there is even no more need for local RAM or FLASH memories (L1 cache level) to cache partial bitstreams. Targeted applications range from adaptive embedded systems like automotive embedded electronics or mobile robotics to dynamic protocols downloading for multi-standards software defined radio. We will conclude that, with a good enough LAN, memory caches for bitstreams can be reduced and systems can be simplified.

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 repository hierarchy. In Sect. 4 we provide 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 focus on.

2. Related works

To our best knowledge up to now, no other LAN than standard Ethernet has been used for DPR. However, Wifi for nomadic computing and communicating equipments as well as CAN for automotive electronic systems seem pertinent candidates to us.

The DPR community agrees on the fact that, in applicative domains with strong real-time constraints, DPR latency is one of the most critical aspects in its implementation. If not brief enough, the DPR interest to build efficient systems can be jeopardized and the research field be interesting in theory but dead in practice. Compton and Hauck [2] give an interesting survey of the whole problematic. More recently, Walder and Platzner [3] confirm this fact in the field of the "wearable-computing". They conclude that the DPR latency can be neglected in their model of on-line scheduling if its effective latency is negligible when compared to the applications computing time. To conclude, extreme rapidity of DPR is a strong assumption in application domains where FPGAs are usually required for performances constraints. Researchers investigate two strategies to reduce the DPR latency. These are the systematic reduction of the bitstreams size and the speedup of their downloading. The first strategy relies on off-line tools and methodologies for FPGA design: "Module Based" and "Difference Based" [4] are two design flows from Xilinx. The second strategy is an on-line approach based on RTOS-like 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 [5] (Internal Configuration Access Port). Virtex2 PRO, Virtex4 VFX and Virtex5 FXT all contain this port and a set of one or two PPC405 hard core processors. The ICAP port can be interfaced with any pure hardware IP as well as the synthesizable soft core processor Microblaze. In the PPC405 case, the ICAP has been wrapped into the HWICAP component which implements around it a standard OPB interface for a cost of 150 slices and a single BRAM (Fig. 2). Thus, it can be connected to the OPB bus and accessed by the PPC405. The reconfiguration peak rate announced by Xilinx is exactly of one byte per clock cycle, it means 100 MB/s (100 MegaBytes) for systems running at 100 MHz. Because systems work at different frequencies, we'll express all measures in number of bits transmitted per seconds and per MHz. The

reference ICAP bandwidth of 100 MB/s becomes a performance of 8 Mb/s.MHz (8 Megabits).

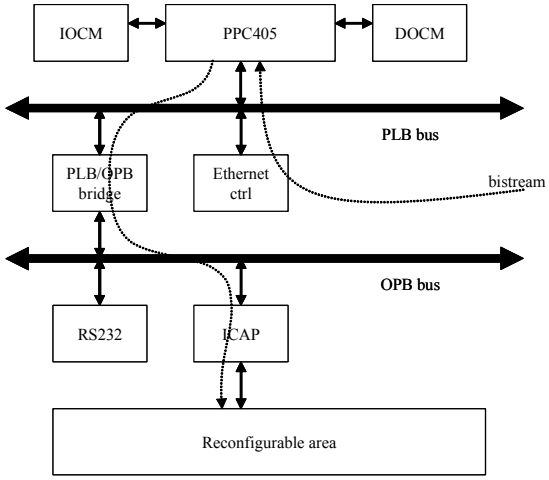

Figure 2 – A PPC405 based architecture to download bitstreams though Ethernet and the ICAP

Claus et al. [6] 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. The experimental platform is a Virtex2 inside which a PPC405 executes the software (no RTOS specified) managing the DPR. Claus's paper lets us think that no functional system was ready at publication time because the ICAP management is presented as being part of a future work. Nevertheless it illustrates the pertinence of DPR for real-time systems.

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 files via a 100 Mb/s Ethernet LAN. The protocol stack is Dunkel's lwIP [8] 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 performances. This rate is 200 times lesser 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 build 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/Ethernet 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 RDP 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/UDP/IP/Ethernet the bitstreams located there can be naturally downloaded into the ICAP. The system is built with a Virtex2 and the processor executing µClinux 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 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. Such a gap in speed and such a memory footprint for a DPR 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 less reliable data links) of the protocols, is such that only hundredths of KB/s can be achieved on traditional LANs. Excessive memory foot-print and maladjusted protocols are the bottlenecks we intended to reduce with a “networked, lightweight and partially reconfigurable platform” [11]. This L1/L2 levels platform is able to download bitstreams at an average sustained performance of 400 Kb/s.MHz, which is at least one order of magnitude faster than all previous

works. Moreover, the executive software and protocol buffers need a memory footprint under 100 Kbytes to provide such a service. This system was build with a Virtex2 VP30 and used one of the two PPC405 hard cores clocked at 100 MHz. Average speed reached over Ethernet was 40 Mb/s and allowed to download partial bitstreams (50-100 KB) in about 10 ms. These 10 ms have to be compared with Lagger's latencies of about 1200 to 2380 ms.

3. Contribution

In this section we present our proposal of bitstreams repository hierarchy and describe each level in terms of hardware and software architectures and communications protocols. We intend also to provide a specification and an optimized implementation of a minimal software layer abstracting the access to the involved hardware resources and the use of a specific data-link level protocol. This work has been patented (FR 08 50641) and we can provide a practical proof of our concept/proposal.

In comparison with the latest layer models of software execution environments for reconfigurable FPGAs proposed by Kettelhoit and Porrmann [12] and Dittmann and Franck [13], our layer is respectively located at the "Configuration layer" and "Execution layer". It has a small size, is written in C language and VHDL and is compiled/synthesized exclusively with standard tools (GCC, ISE).

3.1. Bitstreams repository hierarchy

All FPGA systems 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 memories (L1, latency 1 ms) to the ones of a remote file server (L3, latency 1s). We have also explained that there is a "middle position": a LAN server using a data link level protocol (L2, latency 10 ms) [11]. Our proposal of bitstreams repository hierarchy is based on the correct use of all these levels together to organize a memory caching hierarchy with an LRU updating strategy at FPGA/L1 level (Fig. 3).

Level L1 is the "board level" where designers "glue" together FPGAs and FLASH or RAM memories. Bitstreams can be stored in huge 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 remember, the less memories there are, the cheaper the system is to produce in high volume. L1 is 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. So, instead of being the only bitstreams repository we propose it to become a bitstreams cache for the LAN/L2 and WAN/L3 repositories.

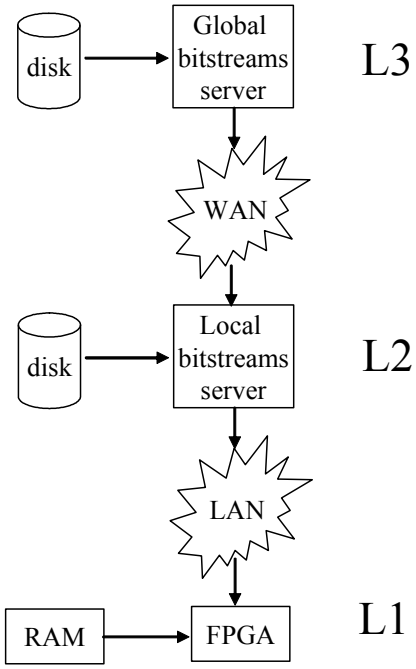

Figure 3 – Bitstreams repositories hierarchy levels L1, L2 and L3

Level L3 is the WAN level with today protocols like FTP. Latencies can degrade quickly and links can break. For real-time embedded systems, it seems unrealistic to rely on such an infrastructure to provide a vital service. Remember downloading of bitstreams is analogous to software scheduling into a RTOS. This is a critical service impacting the whole performances of the system. L3 is geographically the most far away repository from the FPGA with the biggest latency. Our proposal is to consider the global bitstreams server has an equipment dedicated only to refreshing the L2 bitstreams server during night or periodically with tools like "cron".

Level 2 is the LAN level with a specific data link level protocol. It can provide a reconfiguration service with an average latency of 10 ms. Because the LAN is geographically smaller and is possessed by a single institution it can be devoted to a single task. Ethernet,

in its simplest usage, is a medium sharing mechanism on top of which many protocols have been added. But it can also be "seen" as an excellent serial line. In term of buying cost and ease of deployment it is an excellent candidate to transfer bitstreams between close equipments like our FPGAs and the LAN bitstreams server. Now we justify why we consider Ethernet as a good serial line.

3.2. Data link over Ethernet 100 Mb/s

Ethernet, 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 together computers. 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 the broadcast packets). To characterize in speed this LAN topology, we have tested at what maximum speed Ethernet packets could be sent from our portable PC to the XUP board. We developed an application which sends packets as fast as possible, with no specific protocol, no flow control and no error detection. We reused the IEEE 802.3 standard packets, with the EtherType field holding the data payload size in bytes instead of a protocol ID for an upper layer. Direct access to the Ethernet controller MAC level can be done thanks to the Linux "raw" sockets. This test demonstrates that a speed limit of 60 Mb/s is reachable. It depends only on our PC and switch performances. The absence of transmission 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 TCP-like and complex error detection and restart strategy. In line with Xilinx's strategy to reduce bitstreams, we can also estimate the error rates for these small files.

3.3. Bitstreams error rates

Partial bitstreams sizes are in the range of tenth of Kbytes, say a maximum of 100 KB, be 800 Kbits. We know that each transmitted bit has a probability p of being erroneous. With today hubs these error rates are very small, at least in the magnitude 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 1.

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 1 – Estimated error rates for bitstreams downloading through network

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 level protocol described in [11] is a good candidate for such data transmissions. Moreover, when external perturbations occur, they will be concentrated on several adjacent packets. Their erroneous bits will have a higher correlation. Because we are not in the field of RF transmissions, we do not need to decorrelate error bits with Reed-Solomon interleavers. This 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 bistream transmission is only required when it is not present in the local memories and the bitstreams traffic is reduced.

3.4. Is IP necessary ?

The fact that two PCs alone on a network with unloaded Pentiums running Linux or Windows at 2 GHz be only able to transmit a few hundreds of KB per second through a private 100 Mb/s LAN illustrates that today IP protocols implementation is not adapted to low latency and high bandwidth data transfers. But, to optimize an operating system and its protocols stack C source codes is a too long and complex task to just validate that a link-layer protocol is enough. So, we have decided to develop a specific and very simple protocol without interfering with kernel code. With more engineering work, we think a full UDP/IP optimized version would offer the same service but at a much lower speed because of the IP layer processing. This will be part of a future work with UDP and will be compared with this data link level work when functional.

Coming back to Ethernet, its evolution is such that tenths, and even hundredths, of Mb/s are now available at very low costs and with quasi-null error rates. With

our repository hierarchy the bitstream server is connected to the same LAN than our lightweight 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 downloading of a bitstream from any machine over Internet. But, we consider such a complexity is not necessary in LAN-connected systems where downloading speed is critical.

3.5. Software

The software in charge of dynamic reconfiguration is located in the FPGA and executed by the PPC405. It is adapted to both modes of the DPR protocol : master mode and slave mode.

In master mode it checks if a given bitstream is present in local memory before requiring the LAN server to send the bitstream. The politics about memory usage is LRU based, which means the less used bitstreams will be replaced by the most used ones is 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. With Virtex4 and Virtex5, a real 2D partial reconfiguration can be applied and the "column constraint" of Virtex2s is no more a bottleneck. Hence the average size of partial bitstreams is smaller. Based on previous works, we planned a "cache line length" of about 10 Kbytes. This is a parameter that can be tuned. The smaller it is, the less padding space will be lost when loading bitstreams which sizes are not pure multiple of 10 Kbytes. Defragmentation of free space in L1 can be done "off line" while there is no bitstream transfer ongoing. Defragmentation has not been implemented yet, because it has no impact on the downloading latencies. It is part of future work.

In slave mode it reacts to the LAN server requests to force and update of the local bitstreams cache if the bitstream is present in the cache. This possibility is necessary to maintain coherency between local bitstreams and the latest versions available on the servers.

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 on the producer-consumer packets buffer size (*2P+1*) allocated to the DPR protocol and on the cache size and line size. In order to have a worst case measurement we have voluntarily tested with no cache (or cache "off") to provoke systematic downloading of bitstreams. The two curves at the top (plain and dashed curves) represent respectively measured speeds for 60 KB and 200 KB bitstreams.

One can establish that, in both cases, when the packets burst has a size greater or equal to three packets (P = 3), a maximum speed ranging from 375 to 400 Kb/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 * 1.5 KB) 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. Dotted curves at the bottom represent the average speeds reached by Xilinx, Lagger and probably Williams. Our DPR protocol exhibits a reconfiguration speed around 40 Mb/s closer to our local 60 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 DPR software fits into 32 KB of data memory and 40 KB of executable code memory.

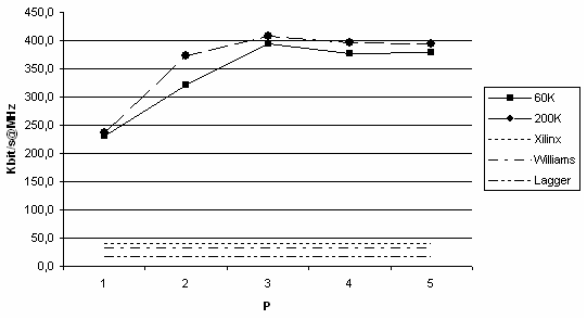

Figure 4 – Cache "off", worst case performances

When cache is on and big enough to store the requestes bitstreams for a given period of time, the downloading latencies are in the range of the ms and bitstreams updates can occur at the speed of Fig. 4.

5. Conclusion and future extensions

Our bitstreams repository hierarchy shows there is still opportunities to improve cache-level, LAN-level, and probably IP-level, caching strategies and protocols in order to provide an efficient and remote reconfiguration service over a standard network. Our

implementation exhibits an order of magnitude gain 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". The use of a DMA to transfer data between the buffer and the ICAP port in order to get closer to the 100 Mb/s Ethernet LAN is a rather natural optimization. Integration of the DPR into the lwIP stack will be a way to promote its usage as well as the customization of UDP/IP stack inside embedded versions of Linux. With Virtex5 FPGAs, Microblaze software versions and full hardware implementation of the DPR protocol might be welcome when targeting systems without PPC405 hard cores. At last, DPR derivatives will be necessary when lightweight systems will be connected to other LANs like Wifi or CAN.

6. References

[1] M. Hubner, M. Ullmann, F. Weissel, J. Becker, "Real-time Configuration Code Decompression for Dynamic FPGA Self-Reconfiguration", Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico, USA, April 2004.

[2] K. Compton, S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software",
ACM Computing Surveys, Vol. 34, NO. 2, June 2002, pp. 171-210.

[3] Herbert Walder, Marco Platzner, "Online Scheduling for Block-partitioned Reconfigurable Devices", Proceedings of the Design And Test in Europe International Conference (DATE'03), Munich, Germany, March 2003.

[4] Xilinx XAPP290, "Two Flows for Partial Reconfiguration: Module Based or Difference Based", September 2004.

[5] B. Blodget, S. McMillan, P. Lysaght, "A lightweight approach for embedded reconfiguration of fpgas", Proceedings of the Design And Test in Europe International Conference (DATE'03), Munich, Germany, March 2003.

[6] Christopher Claus, Johannes Zeppenfeld, Florian Muller, Walter Stechele, "Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System", Proceedings of the Design And Test in Europe International Conference (DATE 2007), Nice, France, April 2007.

[7] Xilinx, XAPP433, "Web Server design using MicroBlaze Soft Processor", October 2006.

[8] Adam Dunkels, "lwIP", Computer and Networks Architectures (CNA), Swedish Institute of Computer Science, http://www.sics.se/~adam/lwip/.

[9] A. Lagger, A. Upegui, E. Sanchez, "Self-Reconfigurable Pervasive Platform For Cryptographic Application", Proceedings of International Conference on Field Programmable Logic and Application (FPL'06), Madrid, Spain, August 2006.

[10] J. Williams, N. Bergmann, "Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip", Proceedings of the 2004 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'04), Las Vegas, Nevada, USA, June 2004, ISBN 1-932415-42-4.

[11] P. Bomel, G. Gogniat, J.-P. Diguet, "A Networked, Lightweight and Partially Reconfigurable Platform", in Proceeding of the International Workshop on Applied reconfigurable Computing (ARC'08), London, United Kinddom, March 2008. Patent FR 08 50641 - N/R BFF 08P0055.

[12] B. Kettelhoit, M. Porrmann, "A Layer Model for Systematically Designing Dynamically Reconfigurable Systems", Proceedings of International Conference on Field Programmable Logic and Application (FPL'06), Madrid, Spain, August 2006.

[13] F. Dittmann, S. Frank, "Hard Real-Time Reconfiguration Port Scheduling", Proceedings of the Design And Test in Europe International Conference (DATE 2007), Nice, France, April 2007.

[14] M. Huebner, T. Becker, J. Becker, "Real-Time LUT-based Network Topologies for Dynamic and Partial FPGA Self-Reconfiguration", in 17th Symposium on Integrated Circuits and Systems Design (SBCCI'04), Pernambuco, Brazil, September 2004.

[15] C. Bobda, M. Majer, A. Ahmadinia, T. Haller, A. Linarth, J. Teich, "The Erlangen Slot Machine: Increasing Flexibility in FPGA-Based Reconfigurable Platforms", Journal of VLSI Signal Processing Systems, Volume 47, Issue 1 (April 2007), Pages: 15-31, ISSN:0922-5773.