ACCELERATION OF COMPLEX ALGORITHMS ON A FAST RECONFIGURABLE EMBEDDED SYSTEM ON SPARTAN-3

Enrique Cantó, Mariano Fons

Department. DEEEA
University Rovira i Virgili
Tarragona (Spain)
email: ecanto@etse.urv.es

Mariano López, Rafael Ramos

Dep. Electronics Engineering
Technical University of Catalonia
Vilanova i la Geltrú (Barcelona)
email: lopezg@eel.upc.edu

ABSTRACT

Complex algorithms usually require several computation stages. Many embedded microprocessors have not enough computational performance to resolve these algorithms in a reasonable time, so dedicated coprocessors accelerate them although the main drawback is the area devoted to them. A reconfigurable coprocessor can drastically reduce the area, since it accommodates a set of coprocessors whose execution is multiplexed on time, although the reconfiguration speed reduces the overall system performance. Although self-reconfigurable systems are possible on Spartan-3 FPGAs, it requires a hard design task due to the lack of software and hardware support available on higher-cost families. This paper describes the architecture of a fast self-reconfigurable embedded system mapped on Spartan-3, used as computation platform to solve a complex algorithm, such as the image-processing carried out in a fingerprint biometric algorithm. In order to reduce the reconfiguration time, the system uses our custom-made memory and reconfiguration controllers. Moreover, the dynamic coprocessor can access directly to external memory through our memory controller to improve processing time.

1. INTRODUCTION

Complex algorithms, such as image processing used for fingerprint features extraction, require several computation stages. Generally, they are implemented as software executed on computers built with high-performance microprocessors. Many embedded systems have not enough computational performance to resolve these algorithms in a reasonable time, so they are usually modified to reduce the computational requirements, although the biometric FAR/FRR (False Acceptation Ratio/False Rejection Ratio) performance can be decreased. A better approach is to accelerate the most time demmanding computations using dedicated coprocessors, improving the response time. The main drawback is the area devoted to dedicated coprocessors, which is greatly increased with the number of coprocessors due to the computation stages of an algorithm, although most of the time only one of the coprocessors is active while the rest ones are inactive. A reconfigurable coprocessor reduces the area, since it can accommodate a set of coprocessors whose execution is multiplexed on time, depending on which moment their utilisation is required. The number of coprocessors that can be mapped in a reconfigurable coprocessor is limited by the capacity of the memory that stores the set of bit-streams. A key parameter of reconfiguration systems is the reconfiguration time, since it must be fast enough to not degrade significantly the overall computational performance.

Traditionally, self-reconfigurable systems are mapped on reconfigurable FPGAs, such as Virtex-2/4/5 FPGAs due to the design software support, such as the PlanAhead with Partial Reconfiguration [1], availability of the internal reconfiguration controller (ICAP) and glitchless reconfiguration. The main drawback of these Xilinx devices is the high cost when they are compared with the low-cost families, such as the Spartan-3.

This paper describes the architecture of a self-reconfigurable embedded system, mapped on Spartan-3 FPGA, designed to accelerate complex algorithms such as a fingerprint biometrics. The architecture is divided in two sections, a static section that maps an embedded soft-core processor (Microblaze) and its peripherals, while the dynamic section maps a reconfigurable coprocessor. The system architecture focuses to improve reconfiguration time, using our custom-made memory and reconfiguration controllers that can directly retrieve bit-streams from external SRAM/FLASH, so the reconfiguration rate can be greatly increased and compete with other reconfigurable systems mapped on higher cost devices (Virtex-2/4). The memory controller also permits direct access to external memory from coprocessors in order to improve processing time of algorithms.

Section 2 resumes the previous works about self-reconfigurable embedded systems on Spartan-3 FPGAs, focussing on their main drawbacks. Next section describes the system architecture of the system, including the

Authors would like to thank financial support to Ministerio de Educación y Ciencia of Spain, under grant TEC2006-12365

custom-made memory and reconfiguration controllers. Section 4 overviews the biometrics algorithm accelerated with the self-reconfigurable embedded system. The experimental results are presented in Section 5, while the conclusions are presented in the last section.

2. PREVIOUS WORKS

The software and hardware support for Virtex-2/4/5 families promoted a large number of research works regarding to self-reconfiguration, but very limited for the low-cost Spartan-3 family. System reconfigurability on Spartan-3 FPGAs requires a hard design task due to the lack of software and hardware support available for higher-cost families, but this work demonstrates the viability of mapping efficient self-reconfigurable embedded systems on these devices. At the design flow level, Xilinx tools, such as the PlanAhead with Partial Reconfiguration, do not support Spartan-3. At the hardware level [2], Spartan-3 lacks of the internal reconfiguration controller ICAP, configuration granularity is a column, and reconfiguration is not glitchless.

Previous works, dealing with self-reconfigurable embedded systems on Spartan-3, have some limitations. In [3] authors focussed on the system and the design flow, where the microprocessor executes the reconfiguration process driving a GPIO (General Purpose Input Output) peripheral externally connected to the SelectMap port, to solve the ICAP lack. The microprocessor retrieves bit-streams from an external host during the reconfiguration process, and the I/O pins connected to external host and memory must be allocated in the static section due to the not glitchless reconfiguration. Moreover, the dynamic section must occupy less than the 50% of the FPGA area, in order to place the central column into the static section to avoid the clock disconnection to the system during the reconfiguration. The work presented in [4] presents an ICAP version to be externally connected to the JTAG serial port, increasing notably the reconfiguration time. Moreover, it presents the same constraints about the area of the dynamic section, and the I/O placement presented in [3]. The work [5] focussed on the system design and design flow to map a self-reconfigurable embedded system, where bit-streams are retrieved from external memory without the constraints of the placement of I/O pins, and where the dynamic section occupies more than the 50% of the FPGA area. To solve these problems the clock was distributed into different routing for static and dynamic sections, and the microprocessor executes a reconfiguration routine. The routine analyses bit-streams from external SRAM and separates them into individual column frames that are temporally stored into BRAM, to reconfigure column by column the large dynamic section. To improve the poor performance of the GPIO as a reconfiguration controller, it developed a more efficient controller that drives the 8-bit wide SelectMap port from a 32-bit word. Although the reconfiguration time is acceptable for many applications, it is desirable to speed-up the reconfiguration time to improve the overall performance of complex algorithms on the self-reconfigurable system.

Neither of coprocessors of the previously commented works had the ability to access directly to external memory. The coprocessors presented in [3][4] are linked to Microblaze FSLs (Fast Simplex Link), meaning that the microprocessor has to read/write data from/to memory and exchange it with the coprocessor through FSL links. The coprocessors presented in [5] are attached to the microprocessor OPB (On-chip Peripheral Bus), but they are not able to process data directly from external memory.

3. SYSTEM ARCHITECTURE

The system architecture, depicted in Fig. 1(a), is divided into the static section, which maps the microprocessor and peripherals, while the dynamic section maps the reconfigurable coprocessor, linked using a set of bus-macros. The static section occupies the 34% of the

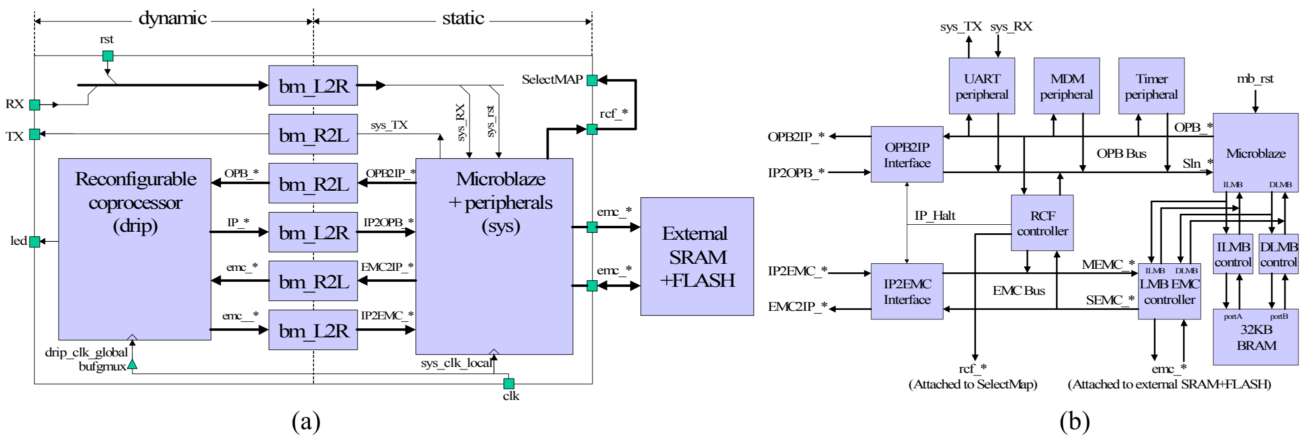


Fig. 1. .(a) The overall system architecture, and (b) the static section architecture

XC3S1500 FPGA area, while the dynamic section is the 66%, meaning that the clock column belongs to the dynamic section. Due to this fact and the not glitchless reconfiguration, the input pin of the clock (40 MHz) is placed into the static section, and distributed in a local low-skew routing for the static section, and in a global routing to the dynamic section [5]. The FPGA is linked to the SelectMAP interface to permit self-reconfiguration, in addition to external SRAM and FLASH that store bit-streams and programs. In the presented system, the I/O pins linked to SRAM/FLASH are placed into the static section to avoid the execution of the reconfiguration routine [5] by the microprocessor, which increases the reconfiguration time.

The static section, depicted in Fig. 1(b), maps the Microblaze with FPU (Floating-Point Unit) and the peripherals UART, MDM and timer, for debugging and testing purposes. The Instruction (ILMB) and Data (DLMB) LMBs (Local Machine Bus) connect the microprocessor with the internal BRAM. Instead of using the OPB-EMC (External Memory Controller) from Xilinx, we developed an LMB-EMC controller and a dedicated bus EMC-Bus to improve access time to external memory and to speed-up reconfiguration time. The reconfiguration controller (RCF) is connected as a master of the EMC-Bus, to permit direct access to external memory, while it is linked as an slave of the OPB, in order to monitor/control it from the microprocessor. The dynamic processor can also access directly to external memory through our LMB-EMC controller, in the same way as the RCF controller, improving processing time. Due to the not glitchless reconfiguration of Spartan-3, the static section is isolated from the dynamic section using isolation interfaces, to avoid unexpected arriving glitches in the busses that would hang up the system when it is reconfigured. The RCF asserts an IP_Halt signal during reconfiguration that drives the isolation interfaces.

The Xilinx PlanAhead with Partial Reconfiguration tool facilitates the design of reconfigurable systems on Virtex-2/4/5 devices, but it does not support the low-cost Spartan-3 family. The design flow is quite tricky, and executes the 'Modular Design' ISE tools and a custom made tool to merge partial layouts [5], to complete the design.

3.1. LMB External Memory Controller (LMB-EMC)

The LMB busses were designed to link Microblaze with internal dual-port synchronous BRAM, and permits to read/write in 1 T_{CLK} and simultaneous access to data and instruction memory. The Xilinx OPB-EMC facilitates the design of embedded systems attached to external memory, due to the very limited capacity of the internal BRAM. However, programs executed on Microblaze from external SRAM suffer a great speed decrease, due to the poor performance of the OPB bus and the OPB-EMC. Figure 2(a) depicts a read cycle, a write cycle and another read cycle, reading two instructions and writing a data from/to a 12 ns asynchronous SRAM, using the OPB-EMC in an embedded system running at 40 MHz. It is divided in four groups showing the main: I/O connections to the SRAM, ILMB, DLMB, and OPB signals, where the opb_xferack signal indicates the end of an OPB bus cycle. The write cycle takes 8 T_{CLK} (clock cycles), where 3 T_{CLK} are devoted to write the word to SRAM, and the rest 5 T_{CLK} are devoted to register data between LMB to OPB and OPB to EMC. A read cycle takes 12 T_{CLK} (or 10 T_{CLK} if a previous write cycle), where 4 T_{CLK} are devoted to read the word from SRAM, and the rest are devoted to register data between LMB, OPB, and EMC.

Although Xilinx does not recommend connecting other peripherals to LMBs, we designed an LMB-EMC controller to avoid the poor OPB-EMC performance. The new LMB-EMC minimises the registers between the LMBs and the EMC, and optimises read and write cycles. Figure 2(b) depicts the LMB-EMC case, where the last group of signal shows the EMC signals instead of the OPB.

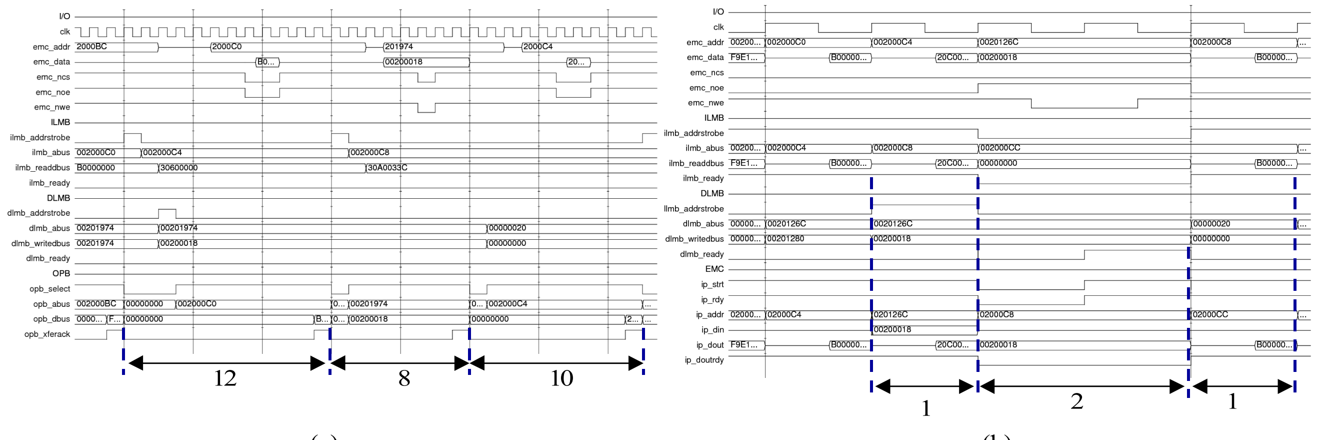


Fig. 2. (a) SRAM access using the Xilinx OPB-EMC controller (b) using the LMB-EMC controller

The ilmb_ready/dlmb_ready signals indicate the final of an access to instruction/data memory. It shows that a write access requires 2 T_{CLK}, while a read access requires just 1 T_{CLK}, for the same SRAM.

Xilinx provides the MCH (MultiChannel) OPB-EMC to accelerate the execution of programs from external SRAM when Microblaze incorporates cache memory, but our LMB-EMC is still faster, it occupies less area, and the Microblaze does not need to devote logic and memory to the cache. Moreover, designing the cache logic and memory in a coprocessor would increase its area and complexity, thus we choose our LMB-EMC to reduce area, design effort and processing time.

In the other hand, the LMB-EMC permits fast page read from FLASH memory, not available in the (MCH) OPB-EMC, to improve the reconfiguration time when retrieving bit-streams from FLASH. In our case, from a Intel StrataFlash J3, which features 120 ns to a new page read and 25 ns to next reads in page, the initial page read devotes 5 T_{CLK}, while each of the next 3 reads devote 2 T_{CLK}. Thus the average read takes 2'75 T_{CLK} when data is sequentially retrieved from FLASH.

The LMB-EMC includes an arbiter to permit the time-sharing of the memory accesses between several masters, such as the microblaze LMBs, the reconfiguration controller (RCF) and the dynamic coprocessor.

3.2. Reconfiguration Controller (RCF)

The RCF controller is connected as a master of the EMC bus. This way, the reconfiguration task executed on the microprocessor is reduced to write the control registers of the RCF to configure the start address and size of a bit-stream. Then, the RCF retrieves a bit-stream form SRAM/FLASH and it drives the SelectMAP interface, without the intervention of the microprocessor to speed-up the reconfiguration time. The RCF devotes a small input buffer of two 32-bit words, permitting to drive the SelectMAP according to a stored word while it reads a new word from external memory.

Figure 3(a) and (b) depicts an initial bit-stream retrieving from FLASH or SRAM, and the SelectMAP ports used during the reconfiguration. The RCF retrieves 32-bit words from SRAM/FLASH using the LMB-EMC, and drives the 8-bit SelectMap interface accordingly. The SelectMAP takes 4 T_{CCLK} to drive in a single 32-bit word, where CCLK is the configuration clock. The theoretical reconfiguration rate RR(Mb/s) can be expressed as:

$$RR(Mb/s) = 32 \cdot min\left\{ \frac{f_{CCLK}}{4}, \frac{f_{CLK}}{N_{32b-READ}} \right\} \cdot 10^{-6} \quad (1)$$

where f_{CCLK} and f_{CLK} are the CCLK and system clock frequencies, and $N_{32b\text{-}READ}$ is the average number of T_{CLK} cycles taken to read a 32-bit word from memory. In our RCF design, a DDR (Double Data Rate) register, which clock input is attached to the 40 MHz system clock, drives the CCLK. The LMB-EMC takes profit of the fast page read from FLASH, as depicted in Fig. 3(a), to improve reconfiguration time when retrieving bit-streams from FLASH. Taking into account than the average read time from SRAM or FLASH is smaller than 4 T_{CCLK}, the reconfiguration rate is limited by the SelectMap interface instead of retrieving a bit-stream from external memory. Thus, the RR achieves 320 Mb/s when retrieving bit-streams from SRAM or FLASH, once the input buffer is filled up. This way, the system avoids copying a bit-stream from FLASH to SRAM before performing a reconfiguration, increasing the overall performance.

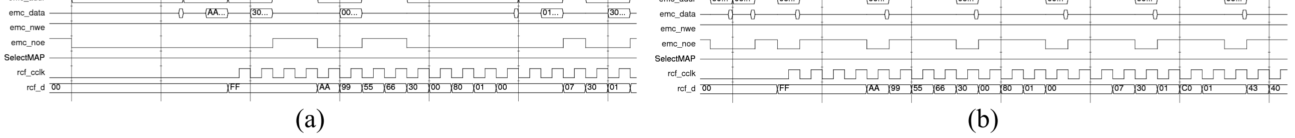
4. IMAGE PROCESSING ALGORITHM

Fingerprint is a widely employed technique in automatic verification biometrics due to its advantages, such as the invariability and distinctiveness features, the availability of low-cost sensors, and the ease-of-use. The fingerprint features generally used are the minutia, the set of points where the fingerprint ridges are bifurcated or ended. Fingerprint verification includes two separated stages: feature extraction and matching. Feature extraction is basically an image-processing algorithm, to enhance the fingerprint image taken from a sensor before extracting distinctive features, while the matching stage scores the extracted minutia when compared with the user template. The feature extraction is the most time demanding stage, since it requires several image-processing steps such as the segmentation, normalization, filtering, and the orientation-field computation, developed in [6].

Segmentation splits the acquired image in a foreground region and a region of interest, where fingerprint image has good contrast. The image is divided in blocks (8*8 pixels per block) to compute the gradient of the block according to:

$$\underset{8x8}{G}(y,x) = \sum_{j}^{8}\sum_{i}^{8}\left|g_y(y+j, x+i)\right| + \sum_{j}^{8}\sum_{i}^{8}\left|g_x(y+j, x+i)\right| \quad (2)$$

where g_x, g_y are the directional gradients at each pixel



(a) (b)

Fig.. 3 (a) Reconfiguration retrieving a bit-stream from FLASH, (b) and from SRAM which are calculated using a Sobel convolution mask. Finally, the block gradients are compared with a threshold value to segment the image.

After segmentation, the image is normalized pixel by pixel according to a desired mean M_0 and variance V_0, to facilitate next processing steps, according to:

$$p_0(y,x) = \begin{cases} M_0 + \sqrt{\dfrac{V_0\left(p(y,x) - M^2\right)}{V}}, if\, p(y,x) > M \\ M_0 - \sqrt{\dfrac{V_0\left(p(y,x) - M^2\right)}{V}}, \ otherwise \end{cases} \quad (3)$$

where M and V are the measured mean and variance from the image. These M,V values are calculated during the previous image-processing step executed on the segmentation coprocessor.

The next processing step is the image filtering to remove noise acquired from the sensor. The algorithm uses the DSS (Dyadic Scale Space) theory, where the image is convolved with a set of Gaussian filters used to remove noise at different scales. The partial results are then composed to obtain an enhanced image. This process is implemented by means of the convolution of the normalized image with an isotropic filter H of size 13x13, according to:

$$q(y,x) = \sum_{j}^{13} \sum_{i}^{13} p_0(y+j, x+i) \cdot h(j,i) \quad (4)$$

where $p_0(y,x)$ represents the pixels of the normalized image, and h(j,i) the elements of the filtering kernel.

Both previous processing steps are implemented in the same coprocessor to improve performance, paralleling the filtering of a pixel with the normalization of the next one.

The next processing stage consists in the computation of the dominant orientation of the fingerprint ridges, to improve the definition of ridges of the fingerprint image in by convolving it with directional filters. The filtered image is divided in blocks of 8*8 pixels, and the local orientation of each block is computed according to:

$$\underset{8x8}{\theta}(y,x) = \frac{1}{2} tan^{-1} \left(\frac{\sum_{j}^{8} \sum_{i}^{8} 2 g_x(y+j,x+i) \cdot g_y(y+j,x+i)}{\sum_{j}^{8} \sum_{i}^{8} \left(g_x^2(y+j,x+i) - g_y^2(y+j,x+i) \right)} \right) \quad (5)$$

where g_y, g_x are the directional gradients at pixel level, computed as in the segmentation step.

The described steps take part of the computing in the fingerprint feature extraction process where coprocessors compute several image-processing steps, accessing directly to external memory, which stores images, to improve response time.

5. EXPERIMENTAL RESULTS

The presented image-processing algorithms were tested using 512*280 pixels (8-bit grey levels) fingerprint images. The embedded system is mapped on a AVNET SP3DK board [7], allocating a XC3S1500 FPGA plus SRAM and FLASH memory.

The software implementation cannot be allocated into the internal BRAM, due to its limited storage capacity, thus they were executed on Microblaze from external SRAM. Table 1 shows the execution time using the Xilinx (MCH) OPB-EMC versus our LMB-EMC, for the image-processing steps and an initial image arrange, and the synthesis results. The LMB-EMC achieves a great speed improvement over the OPB-EMC, about 9 times faster. When Microblaze incorporates cache (8KB+8KB) and the MCH OPB-EMC, the LMB-EMC is about 11% faster and requires less area. Moreover, Microblaze with cache occupies about 10% more CLBs and 10 additional BRAMs. However the main advantage of the LMB-EMC is that coprocessors do not need to incorporate cache logic and memory to speed-up processing time, simplifying their design and devoted area.

Table 1. Execution on microblaze

	OPB-EMC	MCH OPB-EMC	LMB-EMC
Arrange image	34ms	3.8ms	3.5ms
Segmentation	6.10s	0.65s	0.63s
Norm+Filter	79.3s	9.37s	8.65s
Orientation	13.1s	1.86s	1.37s
Total time	98.6s	11.9s	10.7s
CLB slices	196	616	185
Flip-flops	323	730	171
LUTs	135	705	347
Frequ. (MHz)	136	71	125

Table 2 compares the reconfiguration rate for the other reconfigurable embedded systems [3-5] reported in the previous works section. It also shows the measured reconfiguration time in our case, where the bit-stream size of a coprocessor is 409 KB (66% area of the XC3S1500). In the presented paper, the measured reconfiguration rate is quite close to the theoretical 320 Mb/s and demonstrates the high efficiency of the LMB-EMC and RCF controllers. The achieved reconfiguration rate beat other self-reconfigurable embedded systems mapped on higher cost devices, such as the 41 Mb/s [8] in a Virtex-2, or the 13'6 Mb/s [9] on a Virtex-4.

Table 2. Reconfiguration rate and time (409 KB)

	[3]	[4]	[5]	Presented
Rate (Mb/s)	16.1	2	36.3	319.8
Time (ms)	203	1636	90	10.23

Table 3 shows the processing time of the image-processing algorithms when they are executed on the coprocessors. It also reports the number of occupied CLB Slices, BRAM and multiplier blocks. As it can be deducted, the execution time is greatly improved when it is compared with the Microblaze execution time. The coprocessors access directly to SRAM, and execute several computations in parallel to improve their speed performance. Moreover, they make computations using fixed-point arithmetic instead of floating-point to improve processing time and reduce area. The last two rows show the execution time and area of the three coprocessors when they are statically or dynamically implemented. The static implementation requires attaching the three coprocessors to the embedded system, thus the devoted area is the addition of the coprocessor areas. The processing time is also the addition of the processing time of the three coprocessors, because a computation stage starts when the previous stage has been completed, and the image arrange. In the dynamic implementation, the area devoted to the reconfigurable coprocessor can be constrained to the maximum of the areas for each coprocessor, while the processing time is the devoted by the static one adding the reconfiguration time for three reconfigurations. Processing time is quite competitive when compared with the execution on a high-performance PC (Intel Core Duo 2GHz) which takes 16.7 ms.

Table 3. Execution time and area of coprocessors

	Time (ms)	CLB Slices	Block RAM	MUL 18x18
Segmentation	11.7	2489	2	4
Norm+Filter	23.6	6519	4	4
Orientation	12.0	2468	2	8
STATIC	50.8	11476	8	16
DYNAMIC	81.4	6519	4	8

6. CONCLUSIONS

This works demonstrates that the low-cost family Spartan-3 FPGAs can map a fast self-reconfigurable embedded system, to accelerate complex algorithms such as fingerprint biometrics. The dynamic coprocessor maps a set of coprocessors multiplexed on time, to accelerate the computational stages of the algorithm. The dynamic coprocessor can access memory directly through an efficient LMB-EMC controller, in order to improve computational performance. Moreover, the RCF controller accesses to memory through the LMB-EMC, which take profit of the fast-access page of a FLASH memory, and beat other reconfigurable systems mapped on Virtex-2/4 devices, but at lower cost. The reconfiguration rate is not limited due to the retrieving of bit-streams from external memory, but in the SelectMap interface. The self-reconfigurable embedded system accelerates about 145 the processing time of a biometrics algorithm, when compared with the software execution by an embedded system built with the Microblaze with cache.

7. REFERENCES

[1] N. Dorairaj, E. Shiflet, M.Gossman, "PlanAhead Software as a Platform for Partial Reconfiguration", XCELL Journal, 4th Quarter 2005

[2] "Spartan-3 Advanced Configuration", Xilinx XAPP452, June 2008

[3] I. Gonzalez, E. Aguayo, S. López Ubedo, "Self-Reconfigurable Embedded Systems on Low-Cost FPGAs", IEEE Micro, Vol.27, Issue 4, pp.49-57, July-Aug. 2007

[4] K. Paulsson, M. Hübner, G. Auer, M. Dreschamann, L. Chen , J. Becker, "Implementation of a Virtual Configuration Access Port (JCAP) for Enabling Partial Self-Reconfiguration on Xilinx Spartan III FPGAs", Field Programmable Logic and Applications (FPL'07), Aug. 2007

[5] E. Cantó, M.López, F. Fons, "Self-Reconfiguration of Embedded Systems Mapped on Spartan-3", 4th International Workshop on Reconfigurable Communication Centric SoCs(ReCoSoC'2008), pp. 117-123, July 2008

[6] F. Fons, M.Fons, E. Cantó, M. López, "Flexible Hardware for Fingerprint Image Processing", Conf. on PhD Research in Microelectronics and Electrics, Vol 10.3, Sept. 2007

[7] "Xilinx Spartan-3 Development Kit. User Guide", AVNET, 2003

[8] L. Braun, K. Paulsson, H. Kröner, M. Hübner, J. Becker, "Data Path Driven Waveform-Like Reconfiguration", Field Programmable Logic and Applications (FPL'2008), pp. 607-610, Aug. 2008

[9] I. Zaidi, A. Nabina, C. Canagarajh, J. Nunez-Yanez, "Evaluationg Dynamic Partial Reconfiguration in the Integer Pipeline of a FPGA-Based OpenSource Processor", Field Programmable Logic and Applications (FPL'2008), pp. 547-550, Aug. 2008