Making Biometrics the Killer App of FPGA Dynamic Partial Reconfiguration

Run-time reconfigurable hardware technology brings key advantages in the design of automatic personal recognition systems.

by Francisco Fons
PhD Candidate
University Rovira i Virgili, Tarragona, Spain
francisco.fons@estudients.urv.cat

Mariano Fons
PhD Candidate
University Rovira i Virgili, Tarragona, Spain
mariano.fons@estudients.urv.cat
In the current era of communications and information technologies, automatic biometric personal recognition systems represent the state of the art in high-performance signal- and image-processing applications. In fact, it is not difficult to find in our daily lives systems requesting our personal authentication/identification before allowing us to use them; electronic tellers, computers, mobile phones and even cars require such authorization. Many end-user applications that demand better levels of security than PINs, passwords or ID cards use personal recognition algorithms based on biometric (physiological or behavioral) characteristics, usually delivering them as a kernel.

As a proof of concept, we developed an automatic fingerprint authentication system (AFAS) on the second smallest Xilinx® FPGA device in the Virtex®-4 LX family, making use of the Xilinx Early Access Partial Reconfiguration design flow and tools. The experimental results demonstrate it is possible to embed a full, highly demanding biometric recognition algorithm in such a small FPGA at an extremely low cost, processing it in real-time while preserving data accuracy and precision in its physical implementation by multiplexing functionality on the fly over a reduced set of resources placed in a partially reconfigurable region (PRR) of the device. These promising results, together with the proven maturity of the technology we used, encourage us to move this solution from research to industry, in an attempt to make partial reconfiguration (PR) available to the consumer world in the way of secure commercial products.

**Basics of Biometrics**

Computationally complex applications processed in real time, driven at low rates of power consumption and synthesized at low cost are unavoidable requirements today in the design and development of embedded systems, particularly when addressed to mass-production niches. In this context, dynamic partial self-reconfiguration of single-context FPGAs arises as a firm technological alternative, able to deliver a high functional density of resources to efficiently balance all those demands for time-, power- and cost-sensitive applications.

Software-defined radio, aerospace missions and cryptography are some of the known applications that exploit the benefits of dynamic partial reconfiguration of programmable logic devices today. In this context, our group is applying PR to an application space that hasn’t traditionally leveraged it: biometrics. As security has become a major issue in today’s digital information environment, especially for application fields like e-commerce, e-health, e-passports, e-banking or e-voting, among others, we believe the use of PR in biometrics holds great promise.

However, biometrics is complex. It requires stringent and computationally intensive image/signal processing in real time, along with a great deal of flexibility.

**Given that progress in biometrics technology is expected to continue in the future, biometric products already in the market will have to admit upgrades in the field just to avoid getting obsolete, and for this they require open system architectures.**

In addition, personal recognition algorithms are in continuous evolution. As the research community expends major effort in this field, error rates like false acceptance and false rejection are improving. As a consequence, consumers are growing more confident about biometric systems, and acceptance is increasing. Given that progress in biometrics technology is expected to continue in the future, biometric products already in the market will have to admit upgrades in the field just to avoid getting obsolete, and for this they require open system architectures. In this regard, the flexible hardware found in run-time reconfigurable FPGA devices enables the versatility and scalability needed.

Finally, cost-effectiveness is probably the most important reason for biometrics to make use of partial reconfigurability. In aggressive markets like consumer electronics or automotive, vendors must market their systems at a competitive cost. Customers demand products with the highest level of security at the lowest possible price point.

The way to improve security and reliability is by increasing the computational power of the biometric recognition algorithm. This increment of computation usually involves a like increment in execution time and also in cost (resources). However, the cost is hardly affected in those scenarios where the design is based on dynamic-partial-reconfiguration technology. Using PR, designers can partition that new computation and schedule it as

In the current sequential execution flow of the application. Thus, cost often can be held invariant to functional changes of the algorithm.

Designers can partition the biometric recognition algorithm into a series of mutually exclusive stages that are processed sequentially, where the outputs or results of one stage become the input data for the next. This sequential order means designers can multiplex hardware resources in time and customize them to execute a different task or role at each moment, increasing their functional density and thus keeping constant the total number of resources needed to process the entire algorithm. Moreover, the reconfiguration overhead is short enough
so as not to eclipse the benefits gained by hardware acceleration.

Furthermore, reconfiguring one set of resources on the fly will not interrupt the rest of the resources available in the FPGA. In this way, the resources that are not reconfigured continue to operate and guarantee the link with the exterior world for the entire life cycle of the application.

Our challenge in this work consisted in demonstrating that PR fits well in the development of complex personal recognition algorithms based on biometric characteristics, making use of a two-dimensional design abstraction level through which the functionality is managed not only in space but also in time. We describe this target step by step in the next sections.

**Automatic Fingerprint Authentication System**

Fingerprint verification is one of the most popular and reliable biometric techniques used in automatic personal recognition. Essentially, the technique splits the AFAS application into two processes or stages carried out at different times and in different conditions: enrollment and recognition.

Enrollment is the system configuration process through which the user gets registered. Generally, the user exposes his or her fingerprint to the system, which submits it to a set of computationally intensive image-processing phases aimed at extracting all relevant, permanent and distinctive information that will permit the system to unequivocally recognize the fingerprint’s genuine owner. This set of characteristics becomes the user ID, which the system stores in its database. This process is normally conducted off-line, in a secure environment and under the guidance of expert staff.

Once the user is registered, the next time his or her fingerprint is exposed to the system in the recognition stage, the system will check to see if it corresponds with any authorized member within the database. All the processing tasks performed in the enrollment are repeated now to again extract those distinctive characteristics from the live fingerprint sample. The system then compares these characteristics with the information stored as user templates in the database to conclude whether the live scan matches any of the registered templates. Recognition comes in two modalities depending on the size of the database: authentication, when a one-to-one (or one-to-few) matching is processed; and identification, when the matching is one-to-many due to the fact that many users are registered in the system. Recognition is normally performed online in a less-secure environment and under real-time constraints.

Each of these stages is, in its turn, partitioned into a series of mutually exclusive tasks designed to extract from the fingerprint image such information as will distinguish one user from the others. With that object in view, the system carries out specific computations, such as image processing (2D convolution, morphologic operations), trigonometrics (sin, cos, atan, sqrt) [1] or statistics (average value, variance).

Thus, the biometric application is organized in a set of tasks that are processed following a sequential flow. A task cannot start unless the previous task has finished, since the output data of a given task is the input data for the next one in the chain. Moreover, most of these tasks are repeated in both enrollment and recognition stages.

Figure 1 enumerates the tasks that take place in the presented algorithm. The first task is the image acquisition. Depending on the size of the sensor, a system may acquire the whole image at one touch (complete image sensor) or in slices (sweeping sensor). In the second scenario—which was the case we used—an additional image reconstruction phase is necessary. The full fingerprint image gets composed by the set of consecutive and partially overlapped slices acquired [2].

Once we have the whole reconstructed image, the next task consists of segmenting it in the foreground (that is, the region of interest, based on the ridges and valleys of the fingertip skin) from the background. We perform this process by convolving the image, pixel by pixel, with directional filters made up of Sobel masks of kernel 5x5. Afterwards, we normalize the image at a specific mean and variance.

Next, we enhance this normalized image through an isotropic filtering, which retrieves relevant image information from some potential regions of the captured
image initially lost or disturbed by the noise in the acquisition phase, making use of a kernel 13x13 [3]. Once this step has improved the quality of the image, the next task is to compute the field orientation map, which determines the dominant direction of ridges and valleys in each local region of the image foreground. The resultant field orientation is then submitted to a new filtering stage (kernel 5x5) to obtain a refined field orientation map.

Until this point, the image has been worked at 8-bit gray scale. Now, in the binarization process, Gabor directional filters of kernel 7x7 convolve the gray-scale image to improve the definition of the ridges and valleys and convert each of the gray-scale pixels to a 1-bit binary (black or white) dot. The image is then submitted to a new loop to smooth and redraw the shapes of the resultant ridges and valleys. Later, the thinning or skeletonization task converts the black-and-white image to one with black ridges one pixel wide. From that image it is not difficult to extract the fingerprint characteristic points or minutiae, that is, the ridge endings and bifurcations.

Finally, with the minutiae and the field orientation data already obtained, the fingerprint template and sample can be aligned. The first way of accomplishing this is through a brute-force algorithm that moves one image over the other—taking into consideration both translation and rotation movements as well as some admissible tolerances due to the image distortion coming from the skin elasticity in the acquisition phase—to find the best alignment between them [4]. The next step is to match the sample and template to obtain a level of similarity between them, which the automatic system will use to decide if both images correspond to the same person [5].

All this processing, illustrated in Figure 4, is performed on fingerprint images of 500-dpi resolution, 8-bit gray scale and up to 280 x 512 pixels, acquired through sweeping technology via the thermal fingerprint sensor FingerChip from Atmel Corp. and computed in the Xilinx Virtex-4 XC4VLX25 FPGA device.

System Architecture
The Virtex-4 FPGA device becomes the computational unit of the AFAS platform. Flash memory plays the role of system database, storing nonvolatile information like bitstreams as well as specific application data such as user fingerprint templates or configuration settings of the biometric algorithm. The system also uses DDR-SDRAM memory to temporarily store intermediate data or images obtained in each processing stage.

We implemented a serial communication link, in our case an RS-232 transceiver connected to a UART controller—the latter synthesized in the resources of the FPGA—to use for debugging purposes, just to transfer the resulting image of each stage to a PC in order to visualize the fingerprint images or results of each step. Finally, a sweeping fingerprint sensor, addressed to capture the biometric characteristic of the user, acts as input of the recognition algorithm, as depicted in Figure 2.

Regarding the computation unit, the FPGA is detached in two regions, as shown in Figure 3: a static region occupied by a

![Figure 2 – System architecture and functional components breakdown of the suggested AFAS.](image-url)
whole multiprocessor CoreConnect bus system; and a partially reconfigurable region that is used to place—on demand and multiplexed in time as long as the processing advances—the custom biometric coprocessors or IP responsible for the different sequential tasks of the recognition algorithm. The multiprocessor CoreConnect bus system mainly comprises a MicroBlaze™ processor and other standard peripherals along with a custom reconfiguration controller, this one linked to the ICAP port.

All the processing tasks are enumerated from 0 (static) to B in Figure 1, according to sequential execution order. Custom hardware coprocessors implement all the tasks in the PRR, with the exception of the fingerprint acquisition process, which the MicroBlaze performs in software.

The reason behind this specific hardware/software partitioning is that the sweeping sensor needs an integration time of 5 milliseconds to acquire consecutive slices. That’s enough time for it to perform the image reconstruction on the fly directly in software under MicroBlaze control. Therefore, it is not necessary to implement this image reconstruction with a custom hardware coprocessor.

The image acquisition consists of capturing 100 slices at a rate of 5 ms per slice, with each slice consisting of 280 x 8 pixels. Software handles the reconstruction in real time by detecting the overlapping of rows of pixels between each two consecutive image slices.

We implemented the rest of the tasks, however, as custom hardware coprocessors in the PRR of the FPGA simply because of real-time constraints. Once the processing of each particular task is finished, the reconfiguration controller, located on the static region of the device and instructed by the MicroBlaze processor, replaces the coprocessor currently instantiated in the PRR by the one corresponding to the next stage of the biometric algorithm. The reconfiguration controller does this job by simply downloading the new partial bitstream into the PRR and transferring this data directly from DDR-SDRAM to the internal FPGA configuration memory via the ICAP interface.

It is important to note that we used a standard interface based on FIFO memories and flip-flop registers between the static and the reconfigurable regions. This allows us to develop standard biometric coprocessors or IP placed in the PRR that are totally independent of the multiprocessor bus the system uses, be it AMBA®, CoreConnect, Wishbone or some other, as depicted in Figure 2. This point is fundamental in order to guarantee standardization and portability of the biometric algorithm to different platforms.
Reconfiguration Controller

The design of an efficient reconfiguration controller is key to success in the deployment of PR systems oriented to single-context FPGAs. Although the nonreconfigured area of the FPGA remains in operation while the PRR is reconfigured, the PRR resources are not operative at that time, so it is desirable to speed up the reconfiguration process as much as possible so as to minimize this overhead. The reconfiguration time depends on three factors: data bus width, reconfiguration frequency and bitstream size—the first two relate to interface aspects, while the last is closely tied to the PRR size and the design complexity of the partially reconfigurable module (PRM) located there.

Our work implements a reconfiguration controller that is able to transfer partial bitstreams from external memory to the FPGA’s on-chip configuration memory at run-time with a high bandwidth. It is possible to reach the maximum reconfiguration bandwidth of Virtex-4 technology with no constraints in the partial bitstream size and with the external memory as a shared resource that different processors can access concurrently from the system buses.

In the system initialization, the partial bitstreams to be downloaded at run-time into the FPGA configuration memory move from the external nonvolatile memory (flash) to the external DDR-SDRAM. This memory is connected to a multiport memory controller (MPMC), so it becomes a shared resource accessible by any master or slave processor in the system. Different buses can be connected to the MPMC, for instance the CoreConnect PLBv46 bus, used as general-purpose system bus, or even the Xilinx CacheLink (XCL) bus, oriented to fast instruction and data caches of the CPU. The system CPU (MicroBlaze) is, in fact, connected to these two buses.

Our reconfiguration solution, however, is based on a new bus, the Native Port Interface (NPI), which is specifically designed to establish a fast link between the external DDR-SDRAM repository and the ICAP primitive. As part of our reconfiguration controller, we have designed a master memory-management unit (MMU) that handles the NPI protocol. The link between external DDR-SDRAM (partial bitstreams) and the ICAP primitive goes through an internal FIFO memory. In this way, we can implement two different made-to-measure interfaces, with independent data bus size and speed—one coupled to the NPI protocol and the other to the ICAP protocol.

The write port of the FIFO is connected to the NPI and uses a 64-bit data bus. The read port of the FIFO, joined to the ICAP, uses a data width of 32 bits—the maximum data width of ICAP in Virtex-4 devices. Regarding frequency, both read and write ports of the FIFO (on the NPI and ICAP sides) run at 100 MHz, although the NPI side could work at a higher rate if necessary. To keep the transfer latency to a minimum, the master MMU performs the bitstream reconfiguration in 64-word (32-bit) burst transfers to the internal FIFO. This is the maximum length of burst accepted, so all the partial bitstream transactions are done at the lowest burst latency. On the other side, the reconfiguration controller reads the stored FIFO data and transfers it in 32-bit format to the ICAP primitive, as long as the FIFO is not empty. The reconfiguration controller (just the master MMU) is handling the direct memory access (DMA) to huge DDR-SDRAM memory. We set up this part with several configuration registers implemented in another custom slave MMU controller connected to the PLBv46 bus and directly managed by the CPU.

In this way, the CPU only needs to do two things: configure the initial address and size of the partial bitstream to be downloaded in the PRR, and then give the go-ahead command to the MMU master to start the reconfiguration process. At that point, the MMU master starts the bitstream DMA transfer to the internal FIFO and from this to the ICAP primitive. Once the transfer is finished, the reconfiguration controller notifies the CPU.

<table>
<thead>
<tr>
<th>AUTOMATIC FINGERPRINT AUTHENTICATION SYSTEM</th>
<th>PROCESSING TIME (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC PLATFORM</td>
<td>EMBEDDED SYSTEM</td>
</tr>
<tr>
<td>SW Core2Duo 1.83 GHz</td>
<td>SW MicroBlaze Virtex-4 100 MHz</td>
</tr>
<tr>
<td>Acquisition</td>
<td>500.000</td>
</tr>
<tr>
<td>Segmentation</td>
<td>2.810</td>
</tr>
<tr>
<td>Normalization</td>
<td>0.470</td>
</tr>
<tr>
<td>Enhancement</td>
<td>7.030</td>
</tr>
<tr>
<td>Field Orientation</td>
<td>2.500</td>
</tr>
<tr>
<td>Filtered Orientation</td>
<td>0.620</td>
</tr>
<tr>
<td>Binarization</td>
<td>15.940</td>
</tr>
<tr>
<td>Smoothing</td>
<td>14.220</td>
</tr>
<tr>
<td>Thinning</td>
<td>1.410</td>
</tr>
<tr>
<td>Features Extraction</td>
<td>0.630</td>
</tr>
<tr>
<td>Alignment</td>
<td>3224.530</td>
</tr>
<tr>
<td>Matching</td>
<td>4.220</td>
</tr>
<tr>
<td>TOTAL</td>
<td>3774.380</td>
</tr>
</tbody>
</table>

Table 1 – Processing time breakdown (in milliseconds) of the different tasks executed in different AFAS platforms: a software-only approach on a personal computer platform, embedded-software approach on an Xilinx Virtex-4 XC4VLX25 FPGA and HW/SW co-design based on partial reconfiguration.
The automatic fingerprint authentication system described here is the embedded automatic fingerprint recognition module, which runs the program flow in internal BRAM cache, freeing the access to the external DDR-SDRAM to the reconfiguration controller. It is important to note here that this DDR-SDRAM memory where both partial bitstreams and software application are allocated is not a dedicated resource but a shared resource. Even so, this scheme significantly improves upon other existing reconfiguration controller approaches, since it reaches the maximum reconfiguration throughput of Virtex-4 technology (transfer of the partial bitstream at maximum throughput for instance, not to exceed 2 or 3 seconds in an ergonomic standpoint, that could mean, for example, to confirm that we would obtain the same results, and then on an embedded microprocessor like the MicroBlaze synthesized in our FPGA device.

In this approach, the Virtex-4 device implements a software-only solution based on MicroBlaze, without any custom hardware coprocessor in use and without reaching real-time performance. To improve the time, and based on the resultant tasks profiling we obtained, our next step consisted of switching to a HW/SW co-design solution by introducing the PRR, where we located the different custom biometric coprocessors. At this point, we have fully developed the system in both the C programming language and VHDL hardware description language.

We have conducted some recognition tests with 8-bit gray-scale fingerprint images of 268 x 460 pixels. We deployed the same tests in two platforms: in our PR system based on Virtex-4 and also in a personal computer based on an Intel Core 2 Duo T5600 processor running at 1.83 GHz. We then ran the same algorithm, either implemented purely in software or by combining software with flexible hardware, just to compare the performance in both enrollment and recognition stages.

We obtained identical recognition results in both platforms, as expected. However, the processing time spent in each case differed dramatically. Table 1 shows the time needed when the algorithm is deployed on different platforms and architectures: a software approach on the Intel Core 2 Duo PC platform; embedded-software approach on an ML401 platform powered by a Virtex-4 XC4VLX25 FPGA based on a MicroBlaze processor at 100 MHz; and HW/SW co-design approach on an identical ML401 platform equipped with dedicated biometric coprocessors running at either 50 or 100 MHz, instantiated in the PRR and reconfigured on demand.

Table 2 – Time and resources breakdown of the different tasks executed by the AFAS driven by partial reconfiguration technology on a Virtex-4 XC4VLX25 FPGA composed of 21,504 flip-flops, 21,504 four-input LUTs, 72 RAMB16 blocks and 48 DSP48 blocks.

As a result, we achieve the transfer of the partial bitstream at maximum throughput even if the DDR-SDRAM is accessed by the CPU via XCL or PLBv46 buses at the same time. That’s because, in the end, the CPU runs the program flow in internal BRAM cache, freeing the access to the external DDR-SDRAM to the reconfiguration controller. It is important to note here that this DDR-SDRAM memory where both partial bitstreams and software application are allocated is not a dedicated resource but a shared resource. Even so, this scheme significantly improves upon other existing reconfiguration controller approaches, since it reaches the maximum reconfiguration throughput of Virtex-4 technology (transfer of the partial bitstream to the ICAP through a 32-bit data bus at a rate of 100 MHz, or 3.2 Gbps).

Experimental Results
The embedded automatic fingerprint authentication system described here is essentially a high-performance image-processing application, since it exhibits a great deal of parallelism and demands a real-time authentication response. From an ergonomic standpoint, that could mean, for instance, not to exceed 2 or 3 seconds in the authentication process of any user.

The design flow entails several development loops. Initially, we fully developed the algorithm in software in MATLAB® on a PC platform. Afterward, we ported this software code to embedded software in the C programming language and executed it first in the same PC, just to confirm that we would obtain the same results, and then on an embedded microprocessor like the MicroBlaze synthesized in our FPGA device.

In this approach, the Virtex-4 device implements a software-only solution based on MicroBlaze, without any custom hardware coprocessor in use and without reaching real-time performance. To improve the time, and based on the resultant tasks profiling we obtained, our next step consisted of switching to a HW/SW co-design solution by introducing the PRR, where we located the different custom biometric coprocessors. At this point, we have fully developed the system in both the C programming language and VHDL hardware description language.

We have conducted some recognition tests with 8-bit gray-scale fingerprint images of 268 x 460 pixels. We deployed the same tests in two platforms: in our PR system based on Virtex-4 and also in a personal computer based on an Intel Core 2 Duo T5600 processor running at 1.83 GHz. We then ran the same algorithm, either implemented purely in software or by combining software with flexible hardware, just to compare the performance in both enrollment and recognition stages.

We obtained identical recognition results in both platforms, as expected. However, the processing time spent in each case differed dramatically. Table 1 shows the time needed when the algorithm is deployed on different platforms and architectures: a software approach on the Intel Core 2 Duo PC platform; embedded-software approach on an ML401 platform powered by a Virtex-4 XC4VLX25 FPGA based on a MicroBlaze processor at 100 MHz; and HW/SW co-design approach on an identical ML401 platform equipped with dedicated biometric coprocessors running at either 50 or 100 MHz, instantiated in the PRR and reconfigured on demand.

Without considering the acquisition task, which is fixed at 500 ms due to the sweeping-sensor restrictions (100 slices captured with an integration time of 5 ms and image reconstructed from them on the fly), the PR approach reduces latency due to the rest of the processing tasks to 205 ms. That compares with latency of 3,274 ms in the pure-software approach on the PC, which means a speedup of 16x in favor of the PR solution.

Thus, Table 1 makes it evident that real-time authentication is feasible with HW/SW co-design that exploits parallelism and pipeline techniques, along with PR technology, thanks to its low reconfiguration latency. Furthermore, in the PR approach, each
task can run at a specific frequency; this frequency is established each time we reconfigure the PRR to download a new module with new specific characteristics. Our approach ran all the tasks performed in hardware at either 50 or 100 MHz.

Furthermore, the reconfiguration process was always performed at 100 MHz, transferring 32-bit words per clock, a fact that guarantees the lowest reconfiguration latency on the Virtex-4. Each reconfiguration process took between 0.8 ms (for example, normalization) and 1.1 ms (e.g., binarization), depending on the bitstream complexity of each PRR hardware context. This reconfiguration time is negligible in comparison to the total processing time of the biometric recognition application, as depicted in Table 2.

But we not only addressed time in this PR design. We also carefully considered cost-effectiveness by means of the time-sharing of the resources involved. The XC4VLX25 FPGA device contains 21,504 slice flip-flops, 21,504 four-input LUTs, 72 18-kbit RAMB16 blocks and 48 DSP48 blocks. Regarding the partitioning of resources in both static and reconfigurable regions, the reconfigurable region takes 11,264 slice flip-flops, 11,264 four-input LUTs, 22 18-kbit RAMB16 blocks and 44 DSP48 blocks, while the rest of the resources of the device keep static for the entire life cycle of the application.

The PRR is in charge of the execution of up to 11 different sequential tasks of the recognition algorithm. As shown in Table 2, the same application synthesized on a fully static design would not fit fully on the XC4VLX25 FPGA; therefore, that would typically force designers to choose a bigger and more expensive device with the proper amount of resources. However, using PR eliminates this issue. Table 2 definitely demonstrates that automatic personal authentication can be performed at extremely low cost today with the reuse of logic resources thanks to PR technology.

The set of tools we used, available in the Xilinx Early Access Partial Reconfiguration Tools Lounge, are ISE® 9.02.04i together with the PR_12 patch, EDK 9.02.02i, and PlanAhead™ 9.2.7. Finally, we validated the system on real fingerprint images acquired by the system as well as other fingerprint images that exist in public databases based on the same fingerprint sweeping sensor (Fingerprint Verification Competition databases).

Now that we have successfully completed the proof of concept, we plan to port this prototype to the coming next generation of low-end Xilinx 28-nanometer FPGA devices provided with PR capability in the Artix™-7 family, and the new PR design flow based on partitions that Xilinx recently released. Our goal is to design a system able to embed high-performance and real biometric security in any consumer electronics product at the lowest possible cost.

The time for run-time reconfigurable computing in biometric applications is definitely now. For further information about this project, you can contact the authors at {francisco.fons, mariano.fons}@estudants.urv.cat.

References


