Implementation of an Image Processing Algorithm on a Platform DSP Tms320c6416

Farah Dhib Tatar
Department of Electrical Engineering
National school of the studies of engineer of Tunis
Tunis, Tunisia

Mohsen Machhout
Faculty of Sciences of Monastir
Monastir, Tunisia

***Abstract*—In the context of emerging technologies, Cloud Computing (CC) was introduced as a new paradigm to host and deliver Information Technology Services. Cloud computing is a new model for delivering resources. However, there are many critical problems appeared with cloud computing, such as data privacy, security, and reliability, etc. But security is the most important between these problems. Biometric identification is a reliable and one of the easiest ways to recognize a person using extractable characteristics. In addition, a biometric application requires a fast and powerful processing systems, hence the increased use of embedded systems in biometric applications especially in image processing. Embedded systems have a wide variety and the choice of a well-designed processor is one of the most important factors that directly affect the overall performance of the system. This study highlights the performance of the Texas Instrument DSP for processing a biometric fingerprint recognition system.**

***Keywords*—Fingerprint; biometrics; images processing; embedded systems; DSP**

I. INTRODUCTION

Security is a natural human need that continues to increase as applications requiring access control are developed. In recent times, a large number of users have used Internet communication for managing different multimedia data such as health-care, e-business, social networking, intelligent transport systems. [1]

In the context of emerging technologies, Cloud Computing (CC) was introduced as a new paradigm to host and deliver Information Technology Services. In such an environment, privacy and security issues are critical areas that still require to be deeply explored. [2]

With the cloud computing technology, variety of devices are used such as PCs, laptops, smart phones, and PDAs to enable users access programs, storage, and application. Cloud computing can be viewed as a collection of services, which can be presented as a layered cloud computing architecture [3], [4].

The attack of unauthorized users to access the cloud services is the dangerous threat to authorized user as well as to the computing environment. To prevent such environment there are various techniques designed some of them are biometric and some are non-biometric [5].

The techniques of authentication can be divided into three categories: what we possess (ID card, smart card, magnetic badge), what we know (password, PIN code) [6], [7] and what we are (this is biometric identification).

Although a growing number of work is done in studying searchable encryption, most of them can only support exact keyword search. [8]

When customer issues a query on the protected multimedia files, it is quite likely for him to specify a keyword that is a synonym of the present keyword in the encryption phase, which may cause erroneous authentication [9].

To provide secured authentication biometric authentication technique is used by author. The fingerprint image is considered as a biometric input. The mobile phone camera is used as a biometric sensor to capture fingerprint image. If the captured image match with the image stored in the database then authentication is provided to user [5].

Biometrics is the mathematical analysis of the biological characteristics of a person in order to determine his identity irrefutably. Contrary to what we know or what we possess, biometrics is based on what we are and thus avoids duplication, theft, forgetting or loss.

The characteristics used must be universal (i.e. common to all individuals), unique (to be able to differentiate two individuals) and permanent (i.e. time-invariant for each individual).

The biometric identification techniques are various: we can analyze the shape of the hand [10], the design of the iris [11], the voice, the vascularization of the retina and the shape of the face [12]. Likewise, the dynamic recognition of the signature [13], analyzed in real time (speed, pressure on the pen, etc.), but the most used technique is the fingerprint. The use of the fingerprint accounts for more than one third of the market for biometric processes. It is clearly the preferred solution for companies working in this field. The strength of this process is that the use of the fingerprint is generally easier to accept by the community and is one of the most effective and least expensive techniques.

The use of embedded biometric systems is interesting in view of the multitude of fields of application of biometric recognition which require a mobile system or a system which does not take a large place or which will be used as a biometric recognition terminal hence the importance of the use of embedded systems for data processing [14].

Among these applications, we note:

- Control of access to companies and management of staff schedules. This application is very widespread and in great demand in the market.
- The biometric passport which contains an electronic chip embedded in the passport which contains an image of the signature of the owner of this passport [15].
- The BIOCart is a smart card that allows the storage of the 10 fingerprints of the wearer. The comparison time is less than 1 second [16].

Biometric embedded systems use several technologies to give acceptable reliability and execution times. The main technology used is the DSP (Digital Signal Processor). Indeed, DSPs are processors specific to signal processing in particular images. They are characterized by an important power of mathematical calculations.[1]*

In this article we study the performance of a DSP TMS320C6416 (Texas Instruments) by implementing a fingerprint recognition algorithm on a DSK TMS320C6416 hardware platform.

II. Related Works

Since security is a need that keeps increasing, several works have been developed; some of them present an integration between two areas in full evolution: biometrics and cloud Computing, we note for example the authors of [17] who presented the most important standards and recommendations for cloud biometrics and capitalize on the challenges the most important ones encountered during development work on biometric services. The authors [18] cited various security problems, different authentication algorithms and the most recognized security attacks in cloud computing. On the other hand in [19] authors presented and adoptable different cryptographic algorithms to improve the security of the cloud by noting the main problems encountered [3].

In [20], the authors proposed a new remote authentication protocol for network services based on the concept of sharing secrets. Three phases are illustrated in the proposed protocol, namely the initialization phase, the registration phase and the authentication phase. In [21], "a secure and blind biometric authentication protocol is proposed that addresses privacy concerns, model protection, and user trust issues". In [22], the authors proposed a work plan that eliminates concerns about data privacy using encryption algorithms to improve cloud security from the different perspectives of cloud clients [3]. On the other hand the authors of [23] have proposed a pattern of fingerprint identification based on the cloud and preserving privacy, based on a matrix operation. In their work, the user's fingerprint data is sent to the data server. Then, the database owner encrypts all fingerprint data and outsources the database to the cloud. In the identification phase, candidate fingerprint data is first sent to the database owner for encryption. Upon receipt of the encrypted fingerprint data, the cloud performs raster operations to determine the best match in the database. Matrix-based encryption is much faster than existing asymmetric biometric identification schemes based on encryption. The security property of this system, however, has been reviewed and found to be unsafe [24]. Therefore, the authors of [24] have proposed a security-based matrix-based biometric identification scheme called CloudBI [24]. First, they show that the database and identification requests are publicly disclosed because of the lack of randomness in encryption. Then they build a new construction for biometric identification preserving privacy. The attack model in their schema, however, does not cover opponents who are able to arbitrarily enter selected data into the database [25].

Other work has been focused on the biometric domain regardless of the type of application it will be used, and since the fingerprint is the most widely used recognition tool in biometrics, there are some examples of work representing different types of fingerprint recognition algorithms.

The authors of [26] proposed a privacy-preserving fingerprint authentication scheme based on asymmetric homomorphic encryption. In the identification phase, their schema produces all the corresponding results for a specific threshold instead of the best match between the candidate fingerprint data and that of the database. However, due to the adoption of asymmetric encryption, the identification time is considerably longer than that of the biometric identification scheme based on the matrix operation [23]. To overcome the weakness of previous privacy-preserving biometric identification schemes, the authors of [27] proposed a more efficient protocol that exploited an enhanced Euclidean distance process to increase the speed of the identification phase. Nevertheless, the method has shown a serious flaw in that users take the lead of the identification system, resulting in huge computing and communication costs on the users' side [25].

In the course of this present work we will use an algorithm that we have elaborated and validated previously since we are concentrating this time on the implementation step: a demonstration of the properties of the biometric recognition with respect to the processor and the platforms hardware which is a necessary and important passage to be able to validate practically its algorithm.

III. Biometric Fingerprint Recognition

In order to be able to compare two fingerprints we must extract the useful information in each image: this is what is called ***minutiae.***

For this purpose, a well-dedicated image processing must be carried out starting from a gray-level image until obtaining the signature vector containing the necessary and sufficient information for the authentication of the two different fingerprints.

There are several fingerprint recognition algorithms based on various image processing techniques, some algorithms such as HMFA (Histogram-Partitioning, Median-Filtering Fingerprint Recognition Algorithm) are based on Gaussian filters to minimize noise existing on the image to be processed [28]; other studies have focused on improving the comparison phase to ensure rapid authentication [29].

[1] http://www.futura-sciences.com/tech/definitions/technologie-dsp-1839/

The fingerprint recognition algorithm that is used to validate this study is based on the improvement of the general aspect of the captured image by performing a well-defined pre-treatment consisting of five phases as shown in Fig. 1.

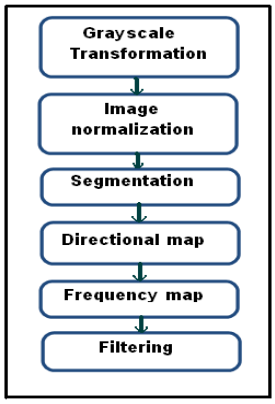

Fig. 1. Pre-processing steps.

The input of the algorithm is an RGB (Color Image) image, after the gray transformation the image becomes smaller because each pixel will be represented on 8 bits (from 0 to 255 gray levels) instead of 24 bits for the color image (RGB).

Normalization is used to standardize intensity values in an image by adjusting the range of gray levels. The structure of the image does not change and the variation of gray levels is standardized. Then the segmentation eliminates the edges of the image as well as areas too noisy.

The directional map makes it possible to specify the local direction of the constituent elements for each pixel (or pixel block) while the frequency map consists of estimating the local frequency of the elements of each pixel. Finally, we use a Gabor filter which consists on modifying the value of the pixels of an image, generally to improve its appearance.

After the preprocessing steps we can then proceed to the minutiae extraction procedure and that is, in turn, composed of five steps (Fig. 2), a minutiae extraction procedure making it possible to minimize the useful information for the comparative phase while keeping all system performances (Fig. 2).

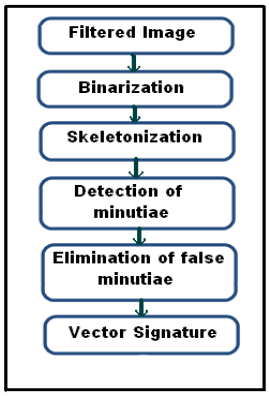

Fig. 2. Extracting signature steps.

The filtered image (which is in 256 levels of gray) is transformed into a binary image where the black pixels correspond to the streaks and the white pixels to the valleys.

Skeletonization involves reducing the thickness of the streaks to a point equal to one pixel while maintaining the connectivity of the streaks. The minutiae are extracted by examining the local neighborhood of each pixel in the image of the fingerprint using connectivity of 8 neighbors. Then a procedure for extracting false minutiae will make it possible to have at the end the true minutiae which will serve to define the vector characteristic of the fingerprint. The signature vector is a file containing information useful for the comparison of the two signatures.

IV. Hardware Platform DSK TMS320C6416

DSK TMS320C6416 is a development platform for low cost applications and designed to quickly develop high performance applications. It is based on a DSP (Digital Signal Processor) of the Texas Instruments TMSC64x family [30].

The C6416 DSK allows as to download and step through code quickly and uses Real Time Data Exchange (RTDX™) for improved Host and Target communications. The DSK utilities include Flash burn to program flash, Update Advisor to download tools, utilities and software and a power on self-test and diagnostic utility to ensure the DSK is operating correctly [31].

TMS320C6416 platform uses a USB connection which will later be transformed into a JTAG interface. The DSK features the C6416 DSP, a 600-MHz device delivering up to 4800 MIPS and designed to meet the needs of high-performing, memory-intensive applications, such as networking, video, imaging and most multi-channel systems. Other hardware features of the C6416 DSK board (Fig. 3) include:

- Embedded JTAG support via USB
- A high-quality 24-bit stereo codec
- Four 3.5-mm audio jacks for microphone, line in, speaker and line out
- 512 Kwords of Flash and 16 MB of SDRAM
- An expansion port connector for plug-in modules
- An on-board standard IEEE JTAG interface
- A +5-V universal power supply

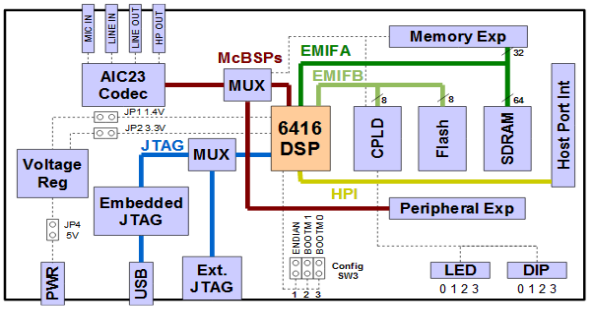

Fig. 3. Block diagram TMS320C6416 DSK.

A. Platform Architecture:

The main component, the DSP, is linked to the various peripherals surrounding it by different types of interfaces. The memory circuits are connected through an EMIF (External Memory Interface) interface. The device expansion port and audio codec are linked via a Multichannel Buffered Serial Port (MCBSP) interface. There is also an expansion port to establish a Host Port Interface (HPI).

The various interfaces are internal peripherals to the DSP. There are also 8 configuration switches to select the operating mode of the platform.

B. Platform Components

The DSP TMS320C6416 represents the core of the system. It is a member of the Texas Instruments (TI) fixed-point DSP family of CSPs that perform well. Indeed, the C6416 is designed according to the architecture VelociTI.2 ™, invented by TI, which is based on the architecture VLIW (Very Long Instruction Word).

With clock rates at 500 and 600 MHz, the C6412 DSP enables as to increase system performance while reducing overall system cost through its combination of low-price point, high-frequency performance, integrated peripherals, large on-chip memory and more channels per processor. The new C6416 DSK (part number TMDSDSK6416) provides an easy-to-use, cost-effective development tool, allowing designers to evaluate the C6412 DSP in their high-performance design [32]. Fig. 4 shows a synoptic diagram of the internal architecture of the DSP chip.

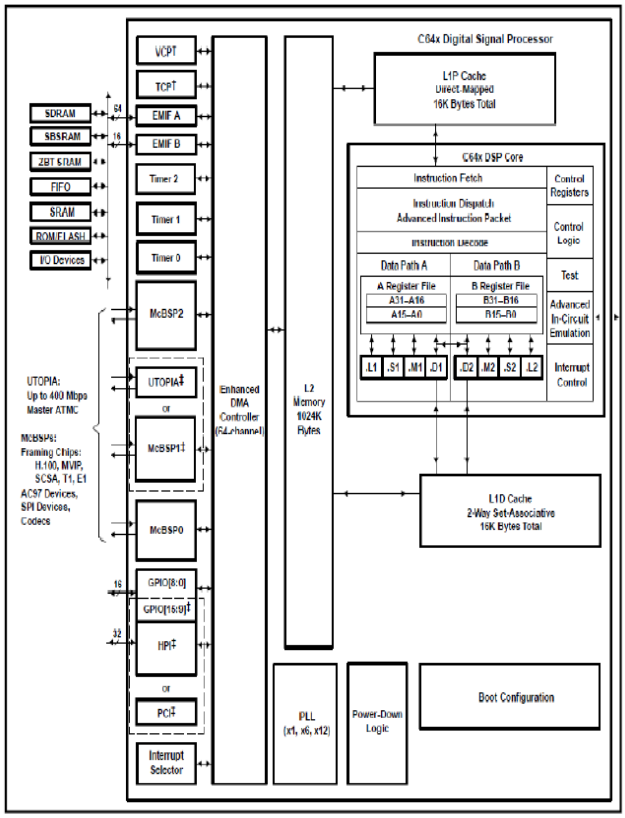

Fig. 4. Internal architecture of the DSP.

1) DSP Core:

The core of the DSP is based on the Very Large Instruction Word (VLIW) architecture, which provides eight 32-bit instructions for every clock stroke, and with a maximum frequency of 1 GHz (generated by an internal PLL). Can therefore reach 8000 MIPS (Million of Instructions Per Second).

VLIW offers the advantages of superscalar implementations without the overhead expense of instruction scheduling hardware since all instructions are scheduled at compile time.

The instructions are grouped into 256-bit packets. There are two types of packages: “fetch” packages and “execute” packages. The first type has a fixed size of 256 bits, the packets are searched from the program memory, while the second type varies in size. It contains the instructions that will be executed in parallel, linked together by a bit to 1 in the LSB (Low Significant Bit) of each instruction, the instructions are said to be chained. The presence of a 0 in the LSB of an instruction will cut the string and place the following instructions efficiently in another packet, one deduces that within an execute package the instructions are linked by 1 at the level Of the LSB.

A “fetch” packet can contain one or more “execute” packets (Fig. 5):

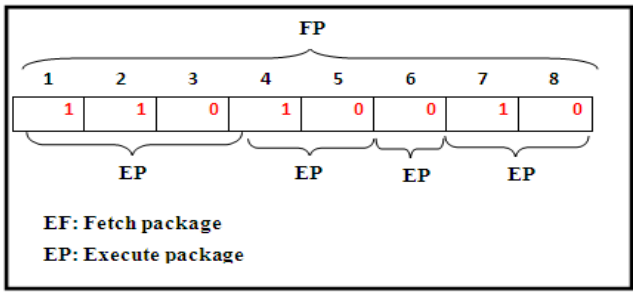

Fig. 5. Fetch package and execute package.

The DSP core contains a three-stage pipeline: “fetch”, “decode” and “execute” (Fig. 5). An execution cycle begins with the “fetch” or the search of the instruction packet from the program memory. Then, in the “decode” stage, the dispatch (DP) or the “fetch” packet is divided into “execute” packets, and for each instruction the appropriate functional unit is assigned. Each instruction is then decoded (DC). The following “fetch” packet is searched only when the “execute” packets are sent to the decoder. The implementation phase is divided into five phases. The first phase is used by any type of instruction for reading the operands (E1). The second phase is used to send addresses and data (E2). Subsequently, the data memory (E3) is accessed to be sent to the CPU (E5). Finally, the results are written in the registers (E5). These registers are 32 bits in number and 64 are divided into two parts to have two data paths A and B.

2) The Internal Memory:

The internal memory, of size 1Mo, is an important advantage of this DSP. It is a fast on-chip memory, performance approaching the cache memories.

3) Cache Memories:

The C64x uses a two-level cache-based architecture and has a powerful and diverse set of peripherals. The Level 1 program cache (L1P) is a 128-Kbit direct mapped cache and the Level 1 data cache (L1D) is a 128-Kbit 2-way set-associative cache. The Level 2 memory/cache (L2) consists of an 8-Mbit memory space that is shared between program and data space. L2 memory can be configured as mapped memory or combinations of cache (up to 256K bytes) and mapped memory.

4) External memories:

The 6416DSK platform has two types of external memory:

- Synchronous DRAM: The DSK uses a pair of industry standard 64 megabit SDRAMs in CE0 of EMIFA. The two devices are used in parallel to create a 64-bit wide interface. Total available memory is 16 megabytes. The DSK uses an EMIFA clock of 100MHz. The integrated SDRAM controller is started by configuring the EMIF in software. When using the SDRAM, we note that one row of the memory array must be refreshed at least every 15.6 microseconds to maintain the integrity of its contents.

- The DSK uses a 512Kbyte external Flash as a boot option. It is connected to CE1 of EMIFB with an 8-bit interface. Flash is a type of memory which does not lose its contents when the power is turned off. When read it looks like a simple asynchronous read-only memory (ROM). Flash can be erased in large blocks commonly referred to as sectors or pages. Once a block has been erased each word can be programmed once through a special command sequence. After than the entire block must be erased again to change the contents. The Flash requires 70ns for both reads and writes. The general settings used with the DSK use 8 cycles for both read and write strobes (80ns) to leave a little extra margin.

5) Improved DMA (EDMA):

A DMA (Direct Memory Access) is a component that allows the transfer of data blocks without the intervention of the processor; it only gives the transfer order and the addresses of the source and destination.

The EDMA (Enhanced DMA), is a peripheral that can be set up to copy data from one place to another without the CPU's intervention. The EDMA can be setup to copy data or program from a source (external/internal memory, or a serial port) to a destination (e.g. internal memory). After this transfer completes, the EDMA can "autoinitialize" itself and perform the same transfer again. It can also be reprogrammed.

6) Other components:

We can also note the following components:

- *Interface with EMIF external memories*: this is an interface between the DSP and the external memories. It is programmable and can be initialized, modified configuration, etc.

- *The Timers:* The DSP has 3 general-purpose 32-bit timers used to count events, generate pulses and interrupts, and send timing events to EDMA

- *The ports McBSP, HPI and PCI:* The Multichannel Buffered Serial Port (MCBSP) serial port operates in full-duplex mode, meaning that data can be sent and received at the same time. The number of transmission channels can be up to 128 channels. The Host Port Interface (HPI) port is a parallel port that allows a host processor (such as a PC) to directly access the memory space of the DSP CPU. The PCI (Peripheral Component Interconnect) port allows connection of the DSP with a PCI host via a "PCI master / slave bus" integrated interface.

- *Embedded coprocessors VCP (Viterbi Decoder Coprocessor) and TCP (Turbo Decoder Coprocessor):* they are designed for channel encoding / decoding. They are used to decode the AMR (Adaptive Multi Rate) channels with different rhythms and constraints. The communication between these coprocessors and the CPU is done through the EDMA.

C. Software Libraries

These libraries feature a highlight of the C6416DSK platform. They are made up of APIs (Application Programming Interface) or pre-written functions in C language (sometimes in assembler). These make it easier to handle most of the components that make up the platform.

We notice:

- The CSL library (a set of functions and macros in C language)

- The BSL library (Board Support Libraries) is intended for handling a few card devices such as flash memory, LEDs, SWITCH and audio codec.

V. Implementation of the Algorithm

The properties of the development platform TMS320C6416 DSK will be exploited to implement the algorithm.

A. Development Environment: Code Composer Studio

Code Composer Studio is an integrated development environment (IDE) that supports TI's Microcontroller and Embedded Processors portfolio. Code Composer Studio comprises a suite of tools used to develop and debug embedded applications (Fig. 6). It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and many other features. The intuitive IDE provides a single user interface taking you through each step of the application development flow. Familiar tools and interfaces allow users to get started faster than ever before. Code Composer Studio combines the advantages of the Eclipse software framework with advanced embedded debug capabilities from TI resulting in a compelling feature-rich development environment for embedded developers [33].

Code Composer Studio includes the following components:

- TMS320C6000 code generation tools
- Code Composer Studio IDE (Integrated Development Environment)
- DSP / BIOS
- RTDX and Hardware Emulation

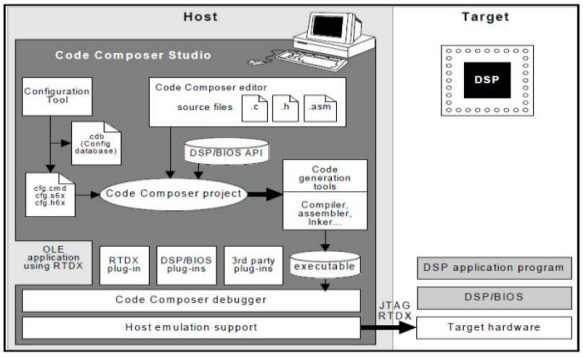

Fig. 6. Code composer studio components.

A Code Composer Studio project (Fig. 7) brings together the source files (C or assembly language), configuration files, and APIs (Application Programming Interface) from DSP / BIOS to switch to code generation tools and the compiler. These provide an executable file to load it into the target platform. Through the debugger and the RTDX tool, we can check the code line by line and view the results on the host PC without affecting the operation of the application.

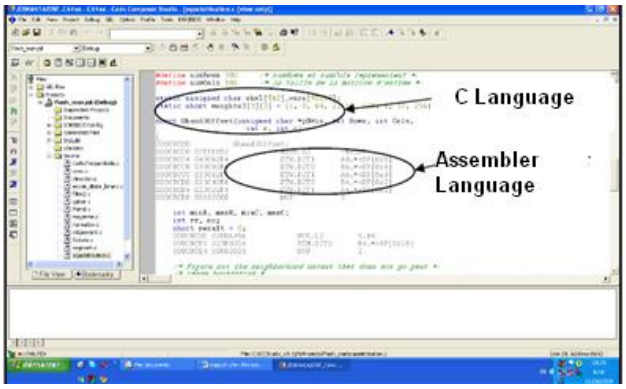

Fig. 7. Code Composer Studio project.

B. Writing to FLASH Memory

To test the functions implemented on the platform, we need test images.

Since we do not have a sensor connected to the evaluation board, we had the idea to save some images in Flash memory. For this, we took advantage of the Flash Burn tool. This allows you to specify the location in the Flash memory where you want to write the data. However, this tool only writes specific file formats (.hex, .dat, or .txt). The type of .dat files has been chosen (Fig. 8), which contains the values of the pixels of the images already transformed into gray levels in order to gain memory size.

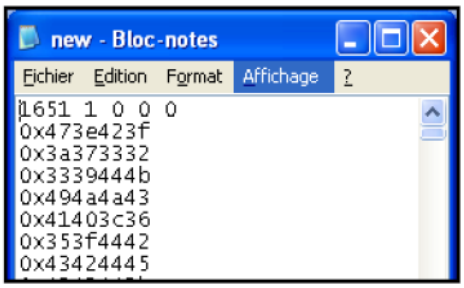

Fig. 8. Format of DAT files.

DAT files, in general, do not have a specific shape. They are raw, header-free data files. However, for this case, this type of file must have a well-defined header so that it can be read by the Flash Burn tool (Fig. 5). This header must begin with a magic number “1651”, then the format of the data (1 for hexadecimals, 2 for integers and 3 for real). Finally, the useful part can be written according to the format used. In this project, the hexadecimal format was used. During reading, the tool used reads in 4 bytes, that is to say 4 pixels at a time by storing them as they are in the memory.

Let’s take the example of Fig. 5. The number 0x473e423f contains the value of 4 pixels in hexadecimal of respective values 0x47, 0x3E, 0x42 and 0x3F.

C. Memory Allocation

Unlike the PC that has only one RAM, this platform contains two types of memory (internal and external RAM) with separate physical address ranges. On a PC, the memory management is done by the operating system, programmer C simply typing the commands of static allocation of variables (type variable_name or type table [] for the allocation of a vector) or dynamic (Malloc, calloc, realloc ...) and the operating system takes the responsibility of allocating memory areas for each variable.

For this development platform, you can use these memory allocation instructions, but you must add a command file (.cmd) and include it in the project. This file must fix the different memory sections. This operation is complex, and if you want to modify the project, you must also modify the partitioning memory.

A very practical solution is to use the Texas Instruments DSP / BIOS real-time kernel that is well suited to the architecture of the DSP used.

DSP/BIOS™ kernel is a scalable real-time multi-tasking kernel, designed specifically for the TMS320C6000™, TMS320C55x™, and TMS320C28x™ DSP platforms. Together with its associated networking, microprocessor-DSP communications, and driver modules, DSP/BIOS kernel provides a solid foundation for even the most sophisticated DSP applications. DSP/BIOS kernel is also one of the world’s mostly widely used real-time operating systems.

The advantage of this kernel is that it provides standardized APIs across C6000™, C55x™ and C28x™ DSP platforms to support rapid application migration and is also optimized to run on the DSP cores (Fig. 9).

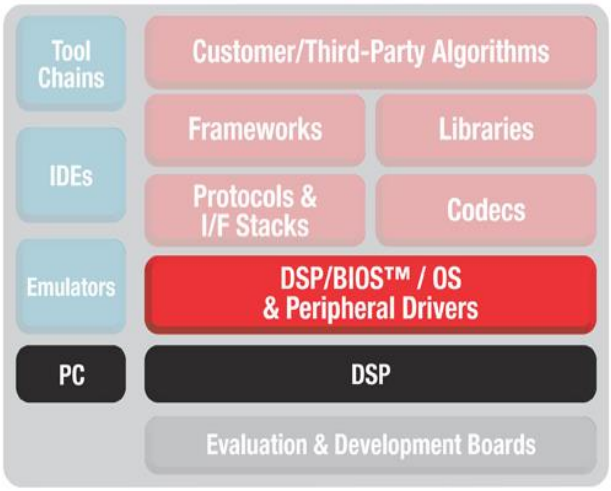

Fig. 9. DSP BIOS kernel.

DSP/BIOS kernel is available both standalone and as an integral part of the Code Composer Studio™ Interactive Development Environment (IDE) and includes graphical Kernel Object Viewer and Real-Time Analysis tools specifically focused on debugging and tuning multitasking applications.

D. Memory Access Problem

An image processing application requires a large data size. Now the fast internal memory cannot contain all the data, so one must then use the external memory which is of sufficient size. Similarly, for this type of applications, there are also a large number of accesses to the external memory. However, with a CPU that can operate up to a frequency of 1 GHz and an external memory of 125 MHz, the large difference is noticeable, in this case the CPU remains in seven waiting cycles at each memory access.

Therefore, to process on a 300 × 300 or 90 000 pixel image, a number of 7 × 90 000 = 630 000 lost clock cycles is lost, which gives 630 µsec that represents a great loss and degrades performance considerably.

1) Use of the EDMA

A first idea is the use of EDMA (Enhanced Direct Memory Access) since it does not require the intervention of the CPU only to launch the transfer that takes place in the background.

We know that the internal memory is a fast memory. It was therefore thought to use the EDMA to bring the data to be used from the external memory to the internal memory, to process them and then to return those to their original place while using fixed size memory areas as a kind of buffer.

Example of using the EDMA: How do we setup the six EDMA parameters registers to transfer 4 byte-wide elements from loc_8 to myDest? (Fig. 10)

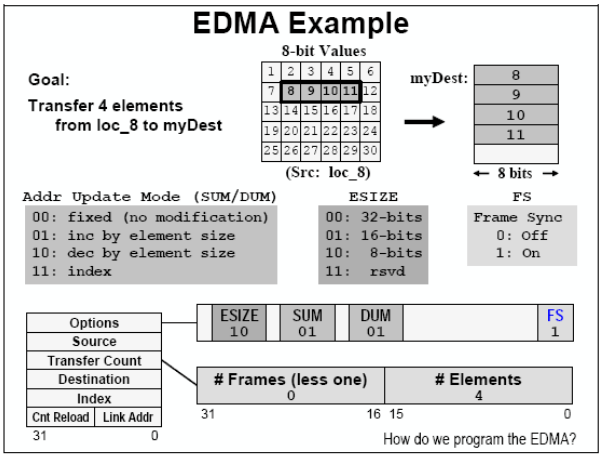

Fig. 10. EDMA example.

- Looking at the EDMA parameters we register at a time
- ESIZE should be self-explanatory based on our previous definitions.
- SUM and DUM fields indicate how the source and destination addresses are to be modified between elements and writes.
- Frame Sync (FS) indicates how much data should be moved whenever the EDMA is triggered to run.
- Source should have the source address of loc_8.
- Transfer counter will have the value 4. Actually it is 0x00000004.

It has been found that this solution is presents some problems such as the management of the order of transfer of the vectors (it is only possible to make one transfer at a time) and if there is a processing which requires a size of Which is greater than the size of the internal memory, then it is necessary to use the SDRAM memory. Moreover, on the performance side, we did not notice a big change concerning the execution time.

We then thought about the activation of the second cache level.

2) Using the cache memory

Why using the cache?

In order to understand why the C6000 family of DSP uses cache, let's consider a common problem. Take, for example, event where a lot of people want to get to one place at the same time (Sporting event, symphony...etc.) so how we can handle parking? We can only have so many parking spots close to the event (Fig. 11). Since there are only so many of them, they demand a high price. They offer close, fast access to the event, but they are expensive and limited. (This is the case of the fast internal memory of the DSP). The other option is the parking garage. It has plenty of spaces and it's not very expensive, but it is a ten minute walk and so you be late (this is the case of the external memory of the DSP).

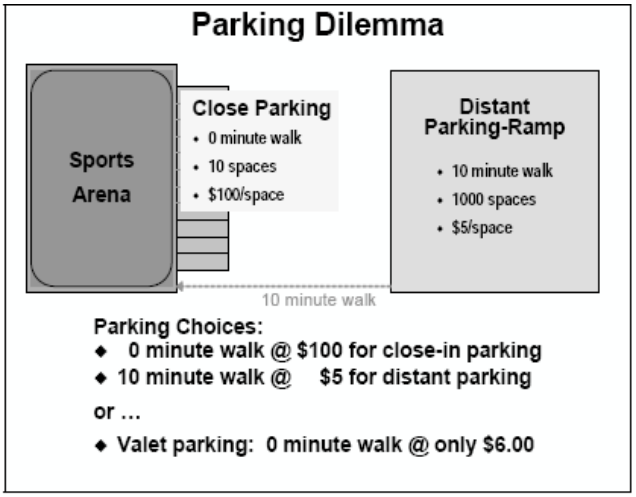

Fig. 11. Example without using cache.

So what is the solution? We must have a valet service which gives the same access as the close parking for just a little more cost than the parking garage. So we can be on time without spending a lot of money: this is the case of cache memory in the DSP.

Cache is the valet service of DSPs. Memory that is close to the processor and fast can only be so big. We can attach plenty of external memory but it is slower. Cache helps solve this problem by keeping what we need close to the processor (Fig. 12). It makes the close parking spaces look like the big parking garage around the corner.

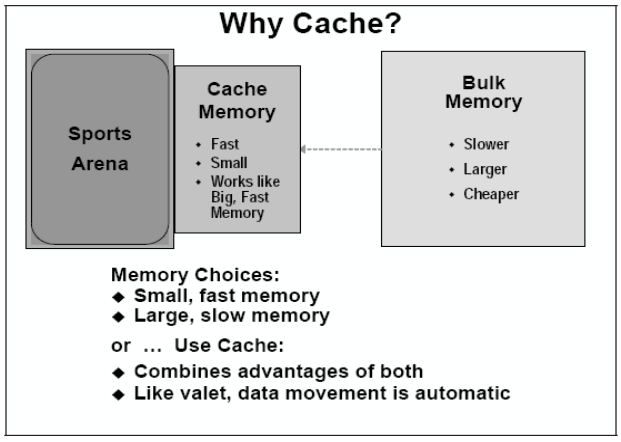

Fig. 12. Example with cache.

One of the often overlooked advantages of cache is that it is automatic. Data that is represented by the CPU is moved automatically from slower memories to faster memories where it can be accessed quickly.

The cache feature of the C6000 allows us to store code automatically in large off-chip memories, while executing code loops from fast on-chip memory (Fig. 13). That is, the cache moves burden of memory management from the designer to the cache controller which is built into the device.

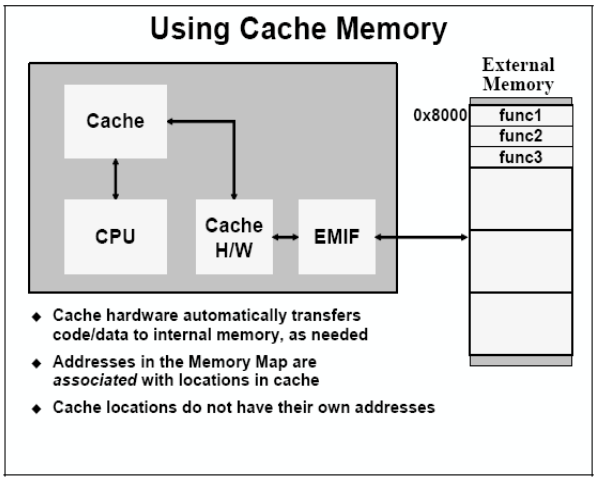

Fig. 13. Using cache memory for C6000 DSP.

Notice that cache, unlike the normal memory, does not have an address. The instructions that are stored in cache are associated with addresses in the memory map.[2]

Application to the Hardware Platform

The hardware platform used makes good use of this concept. It has two caches level L1: one data (L1D) and one program (L1P). A portion of the internal RAM can be configured as an L2 cache memory (Fig. 14).

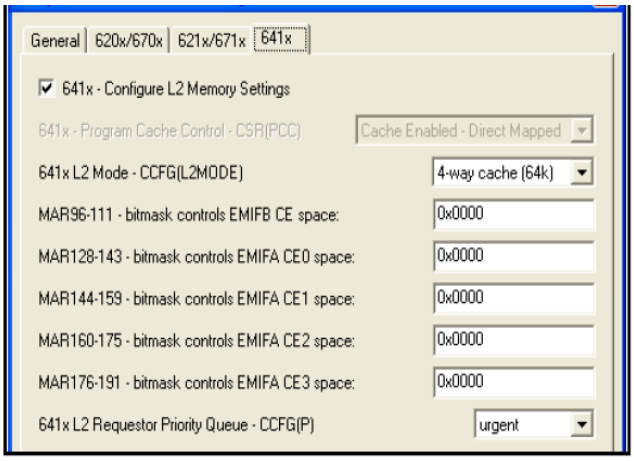

Fig. 14. Configuring the L2 level of cache.

Activation of the L2 cache level can be done through the DSP / BIOS configuration interface. This architecture will be very interesting in this project, because the image processing requires a lot of loops and access to the external memory, and especially for the convolution operation.

VI. Result and Discussion

In order to evaluate performance, we note the execution times of the different pre-processing (Table 1) and minutiae extraction (Table 2) functions.

[2] https://e2e.ti.com/cfs-file/__key/communityserver-discussions-components-files/115/iw6000_5F00_workshop.pdf

TABLE I. PRE-PROCESSING RUN TIMES

Function	Estimated execution time
Standardization & Segmentation	54.12 ms
Directional Map	314.22 ms
Frequency map	89.71 ms
Gabor Filtering	1215.8 ms

TABLE II. EXECUTION TIMES FOR BIOMETRIC DATA EXTRACTION FUNCTIONS

Function	Estimated execution time
Binarization	0.47 ms
Skeletonization	62.7 ms
Detection of Minutiae	0.98 ms
Elimination of false minutiae	0.72 ms
Comparing Two Signatures	804.06 ms

The total execution time will be the sum of the different times obtained. The total time is equal to **2.54278 s.**

Interpretation:

We note that the function that took most of the execution time is the Gabor filtering. Indeed, the operation of this filter rests on a large number of convolutions of blocks of larger or smaller sizes. This explains the value obtained. We can also, to better highlight the results obtained, make a comparison with other processor, note for example the work [35] where the authors have carried out an implementation on an FPGA of the two interesting steps of the fingerprint recognition algorithm which are the binarization and the skeletonization (thinning), the following table represents a comparison of the results of implementation (Table 3):

TABLE III. COMPARAISON BETWEEN FPGA AND DSP

Platform	Binarization	Skeletonization
Matlab	14.762 ms	80.40 ms
FPGA	1.489 ms	7.673 ms
DSP	0.47 ms	62.7 ms

We note that the processor used have good performance compared to the FPGA since it is a dedicated processor for signal processing so it is the most suitable for this type of treatment.

VII. CONCLUSION

The use of fingerprinting is nowadays one of the most reliable technologies on the market to authenticate an individual. This technology is simple to use and quick to implement.

On the other hand, the increased evolution of the embedded systems allows the improvement of the execution time and the reliability of the system.

The image processing, since it presents a significant number of morphological operations and mathematical iterations, requires a dedicated processor.

The platform used, despite having limited memory resources, remains adequate for this type of processing and gives good performance compared to an ordinary computer.

To be able to solve the problem of memory access and at the same time improve DSP processing advantages, our future work will be focused on the study of mixed platforms integrating several types of processors, we can so implement the parts of the algorithms differently (one part on FPGA and another part on DSP for example).

REFERENCES

[1] Anil K. Jain, Karthik Nandakumar, and Abhishek Nagar, Review Article: Biometric Template Security, Journal on Advances in Signal Processing Volume 2008, Article ID 579416.

[2] Muhammad Yaasir Khodabacchus, Krishnaraj Madhavjee Sunjiv Soyjaudah and Gianeswar Ramsawok "Fingerprint code authentication protocol on cloud" in IEEE *Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies* (EmergiTech), 2016

[3] Heba M. Sabri, Kareem Kamal A.Ghany, Hesham A. Hefny andNashaat Elkhameesy "Biometrics Template Security on Cloud Computing" in IEEE International Conference on Advances in Computing, Communications and Informatics, 978-1-4799-3080-7,pp 672-676, ICACCI 2014.

[4] B. Furht and A. Escalante, "Handbook of Cloud Computing", Springer, 2010.

[5] Nileshree R. Darve and Deepti P. Theng "Comparison of Biometric and Non-Biometric Security Techniques in Mobile Cloud Computing" in IEEE *International conference on electronics and communication system*, 978-1-4788-7225-8, pp 213-216, ICECS 2015

[6] Debnath Bhattacharyya, Rahul Ranjan1, Farkhod Alisherov A., and Minkyu Choi, Biometric Authentication: A Review, International Journal of u- and e- Service, Science and Technology, Vol. 2, No. 3, September, 2009

[7] A.Ross, K. Nandakumar, and A. K. Jain.Handbook of Multibiometrics (International Series on Biometrics).Springer-Verlag New York, Inc.,Secaucus, NJ, USA, 2006

[8] Yang Yang, Xianghan Zheng, Victor Chang and Chunming Tang, "Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage", Wiley, May 2017

[9] Yang Yang, Xianghan Zheng, Victor Chang, Shaozhen Ye and Chunming Tang," Lattice assumption based fuzzy information retrieval scheme support multi-user for secure multimedia cloud", Springer, March 2017.

[10] Aythami Morales, Miguel A. Ferrer, Carlos M. Travieso and Jesus B. Alonso "Multisampling approach applied to contactless hand biometrics" International Carnahan Conference on Security Technology (ICCST) , IEEE 2012

[11] Aly I. Desoky, Hesham A. Ali and Nahla B. Abdel-Hamid "Enhancing iris recognition system performance" The International Conference on Computer Engineering & Systems. 2010

[12] Navaneeth Bodla, Jingxiao Zheng, Hongyu Xu, Jun-Cheng Chen, Carlos Castillo and Rama Chellappa "Deep Heterogeneous Feature Fusion for Template-Based Face Recognition" IEEE Winter Conference on Applications of Computer Vision (WACV). 2017

[13] Gang Li, Rui Zhang, Matthew Ritchie and Hugh Griffiths "Sparsity-based dynamic hand gesture recognition using micro-Doppler signatures" IEEE Radar Conference (RadarConf).2017

[14] E. Vijay Sekar, J. Anuradha, Anshita Arya, Balamurugan Balusamy and Victor Chang," A framework for smart traffic management using hybrid clustering techniques", springer, May 2017

[15] http://www.referencement-internet-web.com/15777-Passeport-biometrique-empreintesdigitales- numerisees.html

[16] http://www.orcanthus.com/main/product/product.php?idpr=4

[17] P.Peer and J. Bule , "Building Cloud-based Biometric Services", Informatica , vol. 37 ,pp. 115-122 , 2013.

[18] K.S. Suresh and K.V. Prasad , "Security Issues and Security Algorithms in Cloud Computing ", International Journal of Advanced Research in Computer Science and Software Engineering , vol. 2(10), pp. 110-114, 2010.

[19] R. Nigoti, M. Jhuria, and S. Singh , "A Survey of Cryptographic Algorithms for Cloud Computing " , International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS) , vol. 4(2) , pp. 141-146, 2013.

[20] P. Wang, C.C. Ku and T.C. Wang, "A New Fingerprint Authentication Scheme Based on Secret-Splitting for Enhanced Cloud Security" ,intech , pp. 183-196 , 2011.

[21] K. Hemanth, S. Asadi, D. Murali, N. Karimulla and M. Aswin , "High Secure Crypto Biometric Authentication Protocol", International Journal of Computer Science and Information Technologies, vol. 2(6),pp. 2496-2502, 2011.

[22] M. Kaur and M. Mahajan , "Using encryption Algorithms to enhance the Data Security in Cloud computing", International Journal of Communication and Computer Technologies , vol. 1(3), pp. 56-60 , 2013.

[23] J. Yuan and S. Yu, "Efficient Privacy-Preserving Biometric Identification in Cloud Computing" Proc. of IEEE INFOCOM, 2013.

[24] Q. Wang, S. Hu, K. Ren, M. He, M. Du, and Z. Wang, "CloudBI: Practical Privacy-Preserving Outsourcing of Biometric Identification in The Cloud," Computer Security- ESORICS, 2015.

[25] Changhee Hahn, Hyungjune Shin, and Junbeom Hur "Cloud-based Biometrics Processing for Privacy-Preserving Identification" in IEEE International Conference on Ubiquitous and Future Networks, 978-1-5090-4749-9, pp 595-600, ICUFN 2017

[26] M. Barni, T. Bianchi, D. Catalano, M. D. Raimondo, R.D. Labati, and P. Faillia, "Privacy-preserving FingercodeAuthentication.," MM&Sec', 2010.600

[27] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient Privacypreserving Biometric Identification," NDSS, 2011.

[28] Ayyüce M. Kizrak , Figen Özen "A new median filter based fingerprint recognition algorithm", Haliç University, Electronics and Communications Engineering Department, Suracevizler St. No.29, Bomonti, sisli, Istanbul 34363, Turkey, Elsevier 2011.

[29] Christel-Loïc TISSE, Lionel MARTIN, Lionel TORRES et Michel ROBERT, « Système automatique de reconnaissance d'empreintes digitales. Sécurisation de l'authentification sur carte à puce »,Advanced System Technology Laboratory STMicroelectronics – ZI Rousset – 13106 Rousset, France, Université de Montpellier, UMR 5506, L.I.R.M.M.161, rue Ada -34392 Montpellier, France.

[30] R. Stefanelli, A. Rosenfeld, Some parallel thinning algorithms for digital pictures, Journal of the ACM, Vol.18, No2, pp.255-264, April 1971.

[31] http://www.ti.com/tool/tmdsdsk6416

[32] https://www.ti.com/seclit/wp/spry051/spry051.pdf

[33] http://www.ti.com/tool/ccstudio]

[34] Rahul Kr Das, Abhishek De, Chandrajit Pal and amlan Chakrabarti "DSP Hardware Design for Fingerprint Binarization and Thinningon FPGA" in IEEE *conference on control, Instrumentation, Energie & Communication*, 978-1-4799-2044-0, pp 544-549,CIEC 2014