Chapter 4

Alternative Computing Platforms to Implement the Seismic Migration

Sergio Abreo, Carlos Fajardo, William Salamanca and Ana Ramirez

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51234

1. Introduction

High performance computing (HPC) fails to satisfy the computational requirements of Seismic Migration (SM) algorithms, due to problems related to computing, the data transference and management ([4]). Therefore, the execution time of these algorithms in state-of-the-art clusters still remain in the order of months [1]. Since SM is one of the standard data processing techniques used in seismic exploration, because it generates an accurate image of the subsurface, oil companies are particularly interested in reducing execution times of SM algorithms.

Computational migration needed for large datasets acquired today is extremely demanding, and therefore the computation requires CPU clusters. The performance of CPU clusters had been duplicating each 24 months until 2000 (satisfying the SM demands), but since 2001 this technology stop accelerating due to three technological limitations known as Power Wall, Memory Wall and IPL Wall [10, 23]. This encouraged experts all over the world to find new computing alternatives.

The devices that have been highlighted as a base for the alternative computing platforms are FPGAs and GPGPUs. These technologies are subject of major research in HPC since they have a better perspective of computing [10, 14]. Different implementations of SM algorithms have been developed using those alternatives platforms [4, 12, 17, 21, 22, 29]. Results show a reduction in the running times of SM algorithms, leading to combine these alternative platforms with traditional CPU clusters in order to get a promising future in HPC for seismic exploration.

This chapter gives an overview of the HPC with an historical perspective, emphasizing in the technological aspects related to the SM. In the section two we will show the seismic migration evolution together with the technology, section three summarizes the most important aspects of the CPU operation in order to understand the three technological walls, section four presents the use of GPUs as a new HPC platform to implement the SM,

section five shows the use of FPGAs as another HPC platform, section six discusses the advantages and disadvantages of both technologies and, finally, the chapter is closed with the acknowledgments.

At the end of this chapter,it is expected that the reader have enough background to compare platform specifications and be able to choose the most suitable platform for the implementation of the SM.

2. The technology behind the Seismic Migration[1]

One of the first works related with the beginnings of the Seismic Migration (SM) is from 1914 and it was related with the construction of a sonic reflection equipment to locate icebergs by the Canadian inventor Reginald Fessenden. The importance of this work was that together with the patent named "Methods and Apparatus for Locating Ore Bodies" presented by himself three years later (1917), he gave us the first guidelines about the use of reflection and refraction methods to locate geological formations.

Later, in 1919, McCollum and J.G. Karchner made other important advance in this field because they *received a patent for determining the contour of subsurface strata that was inspired by their work on detection of the blast waves of artillery during World War I,*[6]. But it was only in 1924 when a group led by the German scientist Ludger Mintrop, could locate the first Orchard salt dome in Fort Bend County, Texas [15].

In the next year (1925), as a consequence of the Reginald's patent in 1917, Geophysical research corporation (GRC) created a department dedicated to the design and construction of new and improve seismographic instrumentation tools.

2.1. Acceptance period

In the next three years (1926 - 1928) GRC was testing and adjusting his new tools, but at that time there was an air of skepticism due to the low reliability of the instruments. It can be said that the method was tested and many experiments were performed, but it was only between 1929 and 1932 when the method was finally accepted by the oil companies.

Subsequently, the oil companies began to invest large sums of money to improve it and have a technological advantage over their competitors. *As the validity of reflection seismics became more and more acceptable and its usage increased, doubts seemed to reenter the picture. Even though this newfangled method appeared to work, in many places it produced extremely poor seismic records. These were the so-called no-record areas* [6].

Only until 1936 when Frank Reiber could recognize that the cause of this behavior was the steep folding, [38], faulting or synclinale and anticlinal responses and he built an analog device to model the waves of different geological strata. This discovery was really important for the method, because it could give it the solidity that the method was needing at that moment; but finally the bad news arrived with the beginning of the world war II, because it stopped all the inertia of this process.

[1] The main ideas of this section has been taken from the work of J. Bee Bednar in his paper A brief history of seismic migration

2.2. World War II (1939-1945)

With the World War II all efforts were focused on building war machines. During this period, important developments were achieved, which would be very useful in the future of the SM. Some of these developments were done by Herman Wold, Norbert Weiner, Claude Shannon and Norman Levinson from MIT, and established the fundamentals of the numerical analysis and finite differences, areas that would be very important in future of seismic imaging. For example, Shannon proposed a theorem to sample analog signals and convert them into discrete signals and then developed all the mathematical processing of discrete signals starting the digital revolution.

On the other hand, between 1940 and 1956, appeared the first generation of computers which used Vacuum Tubes (VT). The VT is an electronic device that can be used as an electrical switch to implement logical and arithmetic operations. As the VT were large and consumed too much energy, the first computers were huge and had high maintenance and operation costs. They used magnetic drums for memory, their language was machine language (the lowest level programming language understood by computers) and the information was introduced into them through punched cards[2] [36].

One of the first computers was developed in Pennsylvania University, in 1941 by John Mauchly and J. Prester and it was called **ENIAC** [36]. This computer had the capacity of performing 5000 additions and 300 multiplications per second (Nowadays, a PC like Intel core i7 980 XE can perform 109 billions of Floating Point Operations [27]). In the next years were developed the **EDVAC** (1949), the first commercial computer **UNIVAC** (1951) [36].

One final important event on this period, was the general recognition that the subsurface geological layers weren't completely flat (based on Rieber work previously mentioned), leading the oil companies to develop the necessary mathematical algorithms to locate correctly the position of the reflections and in this way strengthen the technique.

2.3. Second generation (1956-1963): Transistors

In this generation the VT were replaced by transistors (invented in 1947). The transistor also works as an electric switch but it's smaller and consumes less energy than the VT (see figure 1). This brought a great advantage for the computers because made them smaller, faster, cheaper and more efficient in energy consumption.

Additionally this generation of computers started to use a symbolic language called assembler (see figure 2) and it was developed the first version of high level languages like COBOL and FORTRAN. This generation also kept using the same input method (punched cards) but changed the storage technology from magnetic drum to magnetic core.

On the other hand, the SM in this period received a great contribution with the J.G. Hagedoorn work called "A process of seismic reflection interpretation" [18]. In this work Hagedoorn introduced a "ruler-and-compass method for finding reflections as an envelope of equal traveltime curves" based on Christiaan Huygens principle.

Other important aspect in this period was that the computational technology began to be used in seismic data processing, like the implementation made by Texas Instrument Company in

[2] A punched card, is an input method to introduce information, through the hole identification

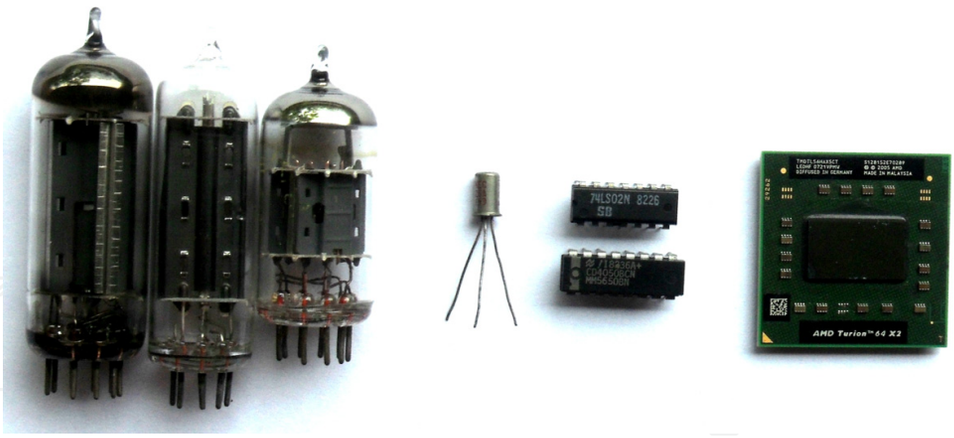

Figure 1. In order from left to right: three vacuum tubes, one transistor, two integrated circuits and one integrated microprocessor.

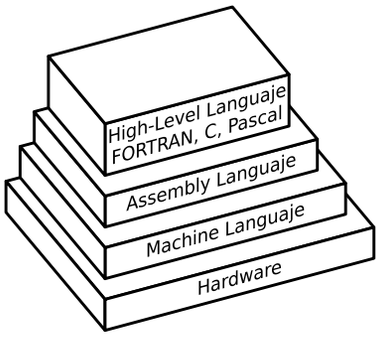

Figure 2. Assembly language.

1959 on the computer TIAC. In the next year (1960) Alan Trorey from Chevron Company developed one of the first computerized methods based on Kirchhoff and in 1962 Harry Mayne obtained a patent on the CMP stack [30]; this two major contributions would facilitate later the full computational implementation of the SM.

2.4. Third generation (1964-1971): Integrated circuits

In this generation the transistors were miniaturized and packaged in integrated circuits (IC) (see figure 1). Initially the IC could contain less than 100 transistors but nowadays they can contain billions of transistors. With this change of technology, the new computers could be faster, smaller, cheaper and more efficient in the consumption of energy than the second generation [36].

Additionally, the way in which the user could introduce the information to the computers also changed, because the punched cars were replaced by the monitor, keyboard and interfaces based on operating systems (OS). The OS concept appeared for first time, allowing to this new generation of computers execute more than one task at the same time.

On the other hand, the SM also had a significant progress in this period. *In 1967, Sherwood completed the development of Continuous Automatic Migration (CAM) on an IBM accounting machine in San Francisco. The digital age may have been in its infancy, but there was no question that it was now running full blast,* [6] and, in 1970 and 1971, Jon Claerbout published two papers focused on the use of second order, hyperbolic, partial-differential equations to perform the

imaging. Largely, the Claerbout work was centered in the use of finite differences taking advantage of the numeric analysis created during the World War II. The differences finite work, allowed to apply all these developments over the computers of that time.

2.5. Fourth generation (1971- Present): Microprocessors

The microprocessor is an IC that works as a data processing unit, providing the control of the calculations. It could be seen as the computer brain, [36].

This generation was highlighted because each 24 months the number of transistors inside an IC was doubled according with the Moore Law, who predicted this growing rate in 1975, [32]. This allowed that the development of computers with high processing capacity were growing very fast (see table 1). In 1981 IBM produced the first computer for home, in 1984 Apple introduced the Macintosh and in 1985 was created the first domain on Internet (symbolics.com). During the next years the computers were reducing their size and cost; and taking advantage of Internet, they raided in many areas.

Microprocessors	Year.	Number of transistors	Frequency	bits
Intel 4004	1971	2,300	700 KHz	4
Intel 8008	1972	3,300	800 KHz	8
Intel 8080	1974	4,500	2MHz	8
Intel 8086	1978	29,000	4MHz	16
Intel 80386	1980	275,000	40MHz	32
Intel Pentium 4	2000	42,000,000	3.8GHz	32
Intel Core i7	2008	731,000,000	3GHz	64

Table 1. Summary of the microprocessors evolution.

Moore law was in force approximately until 2000, because of power dissipation problems produced by the amount of transistors inside a single chip, this effect is known as the Power Wall (PW) and remains stagnating the processing capacity.

From that moment began to surface new ideas to avoid this problem. Ideas like fabricate processors with more than one processing data unit (see figure 3); these units are known as cores, seemed to be a solution but new problems arose again. Problems like the increase of cost, power consumption and design complexity of the new processors, didn't reveal to be a good solution. This is the reason that even today we can see that the processors fabricated with this idea only have got 2, 4, 6, 8, or in the best way 16 cores, [5].

In figure 4 we can see the Moore Law tendency until 2000, and after this date we can see the emergence of multi-core processors.

2.6. HPC's birth

A new challenge to computing researchers was the implementation of computationally expensive algorithms. For this purpose was necessary to design new strategies to do an optimal implementation over the more powerful computers. That group of implementation strategies over supercomputers has been known as High Performance Computing (HPC).

The HPC was born in Control Data Corporation (CDC) in 1960 with Seymour Cray, who launched in 1964 the first supercomputer called CDC 6600 [24]. Later in 1972 Cray left CDC

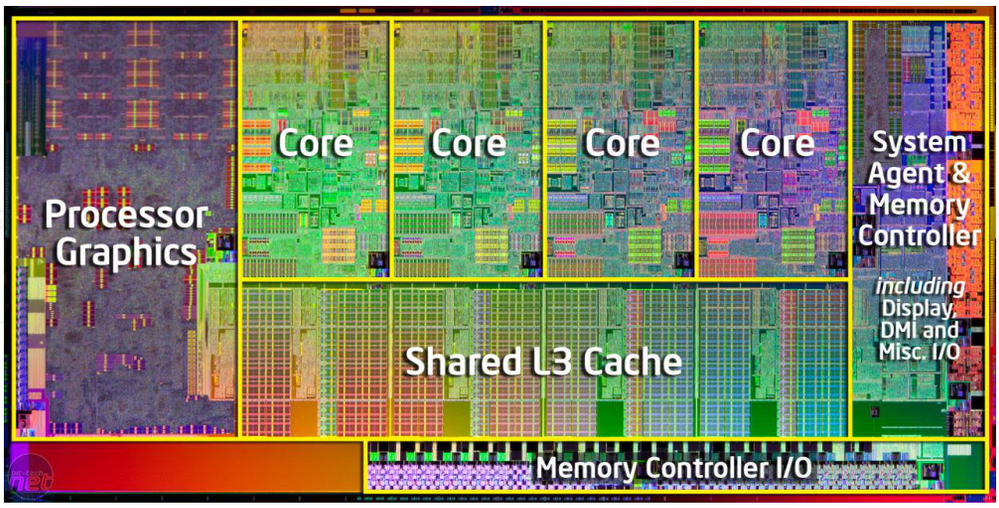

Figure 3. Intel multi-core processor die. Taken from [8]

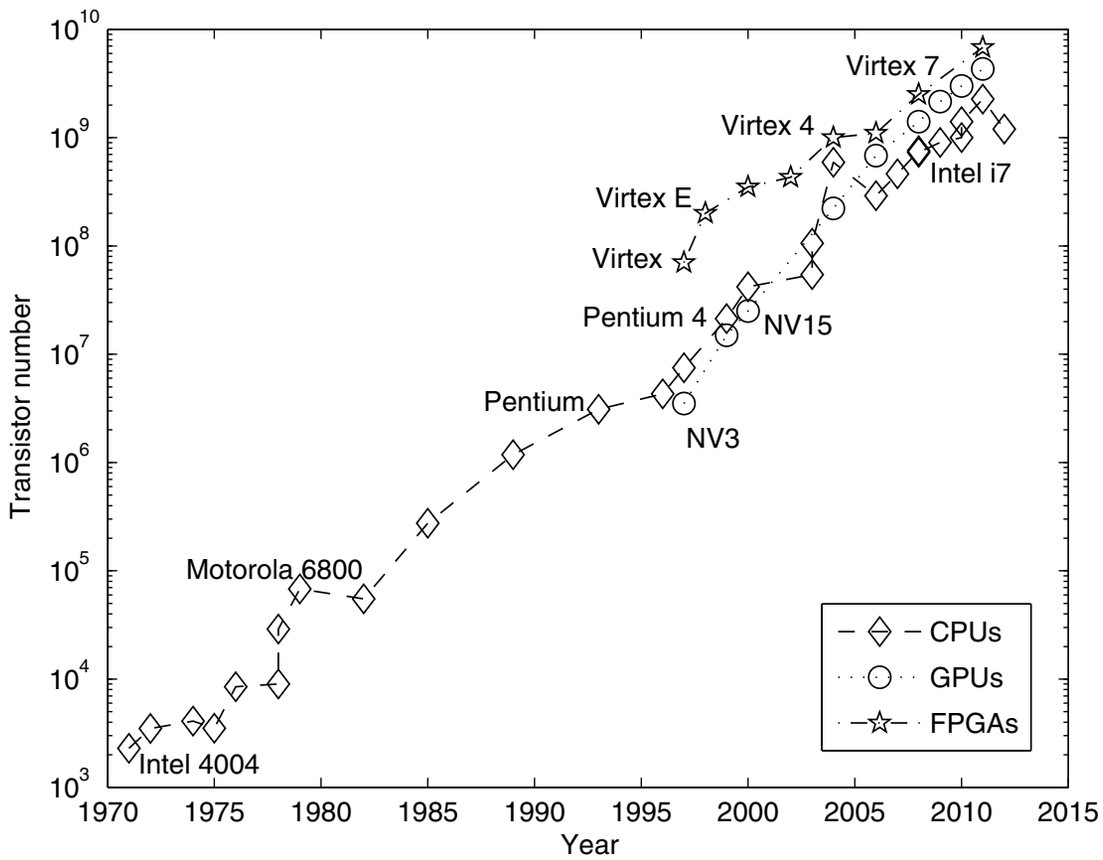

Figure 4. CPU, GPU and FPGA Transistor Counts

to create his own company and four years later in 1976, Cray developed Cray1 which worked at 80MHz and it became in the most famous supercomputer of the history.

After that, the HPC kept working with clusters of computers using the best cores (processors) of each year. Each cluster was composed by a front-end station and nodes interconnected through the Ethernet port (see figure 5). So, in that way the HPC evolution was focused on increase the quantity of nodes per cluster, improve the interconnection network and the development of new software tools that allow to take advantage of the cluster. This strategy began to be used by other companies in the world like Fujitsu's Numerical Wind Tunnel, Hitachi SR2201, Intel Paragon among others, giving very good results.

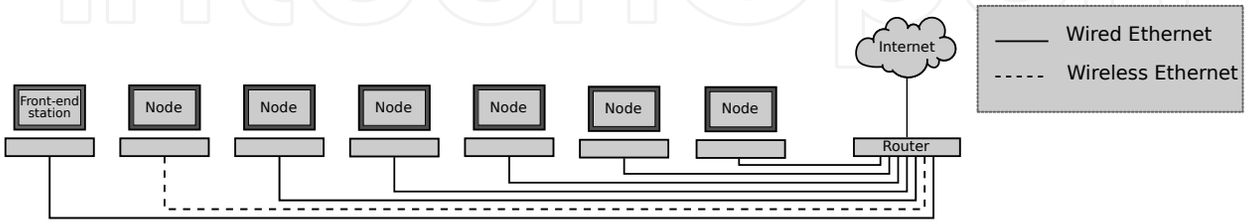

Figure 5. General diagram of a cluster

2.7. SM consolidation

On the other hand, the SM also received a great contribution by Jon Claerbout, because in 1973 he formed The Stanford Exploration Project (SEP). The SEP together with the group GAC from MIT built the bases for many consortia that would be formed in the future like Center For Wave Phenomenon at Colorado School of Mines. In 1978 R. H. Stolt presented a different approach to do the SM, which was called "Migration by Fourier Transform", [41]. Compared with the approach of Claerbout, the Stolt migration is faster, can handle up to 90 degrees formation dips but it's limited to constant velocities; while as Claerbout migration is insensitive to velocity variations and can handle formation dips up to 15 degrees.

In 1983 three papers were published at the same time about a new two-way solution to migration; these works were done by Whitmore, McMechan, and Baysal, Sherwood and Kosloff. Additionally, in 1987 Bleistein published a great paper about Kirchhoff migration, using a vastly improved approach to seismic amplitudes and phases. From that moment, the SM could be done in space time (x,t), frequency space (f,x), wavenumber time (k,t), or almost any combination of these domains.

Once the mathematical principles of the SM were developed, the following efforts focused on propose efficient algorithms. The table 2 summarizes some of these implementations, [6].

But perhaps one of the most important software developments between 1989 and 2004 for seismic data processing was performed by Jack Cohen and John Stockwell at the Center for Wave Phenomena (CWP) at the Colorado School of Mines. This tool was called Seismic Unix (SU) and has been one of the most used and downloaded worldwide.

Moreover, the combination of cluster of computers with efficient algorithms began to bear fruit quickly, allowing pre-stack migrations in depth at reasonable times. An important aspect to note is that almost all algorithms that we use today were developed during that time and have been implemented as the technology has been allowing it.

Year	Author	Contribution
1971	W. A. Schneider's	tied diffraction stacking and Kirchhoff migration together.
1974 and 1975	Gardner et al	Clarified this even more.
1978	Schneider	Integral formulation of migration.
1978	Gazdag	Phase shift method.
1984	Gazdag and Sguazzero	Phase shift plus interpolation.
1988	Forel and Gardner	A completely velocity-independent migration technique.
1990	Stoffa	The split step method.

Table 2. Development of efficient algorithms.

Over the years the seismic imaging began to be more demanding with the technology, because every time it would need a more accurate subsurface image. This led to the use of more complex mathematical attributes. Additionally it became necessary to process a large volume of data resulting from the 3D and 4D seismic and the use of multicomponent geophones. All this combined with the technological limitations of computers since 2000, made HPC developers began to seek new technological alternatives.

These alternatives should provide a solution to the three barriers currently faced by computers (Power wall, Memory wall and IPL wall; these three barriers will be explained in detail below). For that reason the attention was focused on two emerging technologies that promised to address the three barriers in a better way than computers, [14]. These technologies were GPUs and FPGAs.

In the following sections it's explained how these two technologies have been facing the three barriers, in order to map out the future of the HPC in the area of seismic data processing.

3. Understanding the CPUs

In this section initially we will present to the reader an idea about the internal operation of a general purpose processing unit (CPU), assuming that the reader doesn't have any previous knowledge. Subsequently, we will analyze the contribution made by the alternative computing platforms in the high performance computing.

Let's imagine an automobile assembly line completely automated like the shown in the figure 6, which produce type A automobiles. The parts of the cars can be found in the warehouse and are taken into the line, passing through the following stages: engine assembly, bodywork assembly, motor drop, painting and quality tests. Finally we get the vehicle and it is carried to the parking zone. In this way, in the seismic data processing, a computing system can access the data stored in memory, adapt it, calculate the velocity model, and finally migrate them to produce a subsurface image that is stored again in memory.

Now let's suppose that we want to modify the assembly line in order to make it produce type B automobiles. Therefore, it's necessary either to add some stages to the assembly line or to modify the existing modules configuration. In like manner operates a CPU, where it can be executed different kind of algorithms as well as the assembly line can now produce different

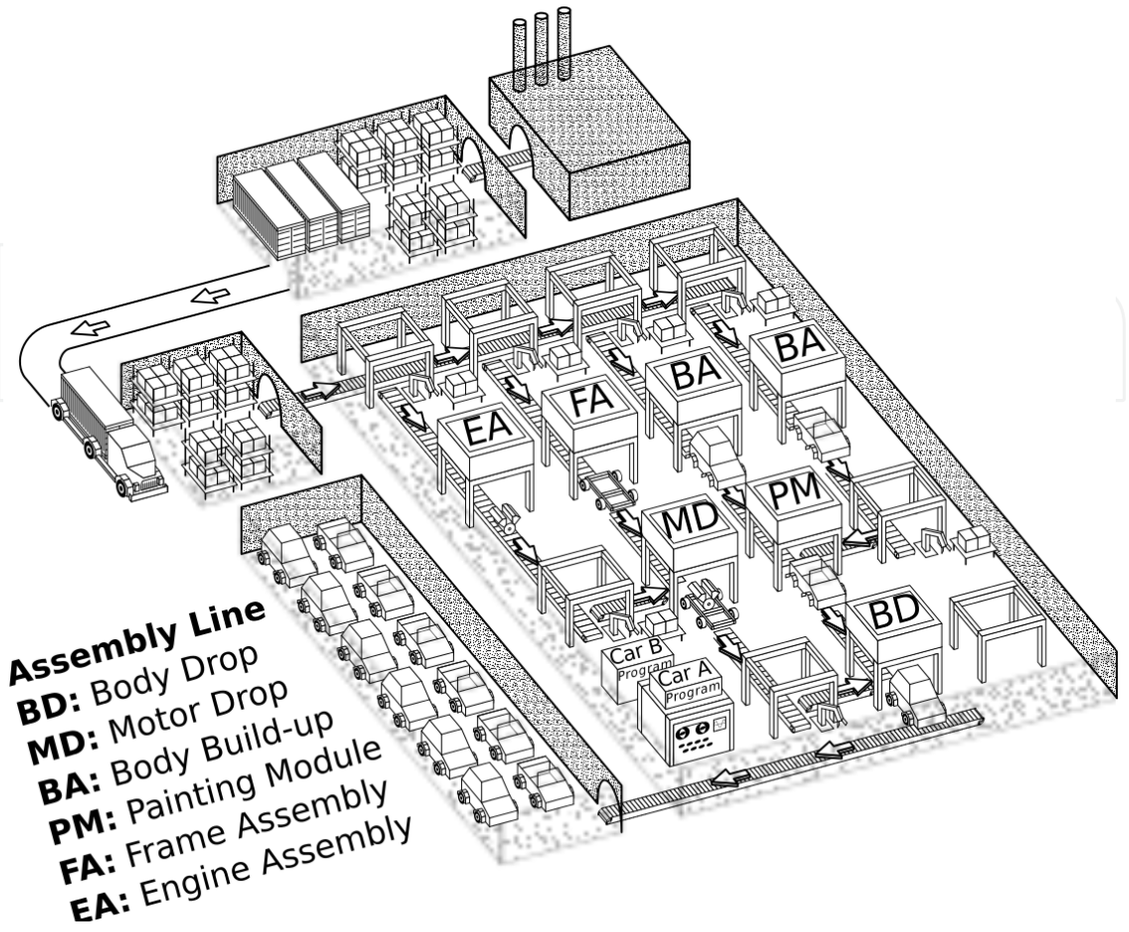

Figure 6. CPU analogy

kinds of automobiles. A CPU includes different kind of modules that allow it execute several instructions which can execute the desired algorithm.

As well as the CPU receive and deliver data, the assembly line receive parts and deliver vehicles. But to assemble a particular type of car, the line have to get prepared. In this way, the line can now select the right parts from the warehouse, transport it and handle it suitably. In a CPU, the program is the responsible to indicate how to get each data and how to manipulate it to obtain the final result.

A program is composed by elementary process called instructions. In the assembly line, an instruction matched simple actions like bring a part from the warehouse, adjust a screw, activate the painting machine, etc. In a CPU, an instruction executes a elementary calculation over the data such as fetch data from memory, carry out a sum, multiplication, etc.

3.1. CPU operation

In a CPU, the instructions are stored in the memory and are executed sequentially one by one as it's shown in the figure 6. To execute one instruction, it must fetch it from memory, decode (interpret) it, read the data required by the instruction, manipulate the data and finally return the results to the memory. This architecture is known as Von Neumann. Moderns CPUs have evolved from their original concept, but its operating principle is still the same.

Nowadays almost any electronic component that we use is based on CPUs like our PC, cell phones, videogames, electrical appliance, automobiles, digital clocks, etc.

As was described, the CPU is in charge to process the data stored in memory as the program indicated. In our assembly line we can identify several blocks that handle, transport and fit the vehicle parts. In a CPU, this set of parts is known as the datapath. The main functionality of the datapath is to temporally store, transform and route the data in a track from the entrance to the exit.

In the same way, in the assembly line we can find the main controller which is in charge to monitor all the process and to activate harmonically the units on each stage to execute the assembly process. All this, taking into account the requested assembly process. In a CPU we can find the control unit, in charge to address orderly the data through the functional units to execute each one of the instructions that the program indicates and thus perform the algorithm.

3.2. CPU performance

The performance of an assembly line, could be measured as the time required to produce certain amount of vehicles. In the same way, the CPU performance is measured as how long it takes to process some data. In first place, the performance of the assembly line could be limited by the speed of its machines as well as in the CPU the integrated circuit speed is proportional to its performance. The second aspect that affects the performance is how fast can be put the parts at the beginning of the assembly line. In the same way, the data transference rate between CPU and memory could limit its performance. The CPU performance is related with the execution time of an algorithm, for that reason we are going to analyze some aspects that have slowed down the CPU performance.

We will analyze the assembly line operation to make the best use of its machines, and increase its performance. We can observe that the units on each stage could simultaneously work over different vehicles, and it is not necessary to finish one vehicle to start the next one (see figure 6). In the same way, the CPU can process several data at the same time provided that it has been designed using the technique called pipeline, [19]. This technique segments the execution process and allows that the CPU handles several data in parallel. This is one of the digital techniques developed to improve the CPU speed although nowadays developments on this area are stuck. This phenomenon is one of the greatest performance improvement constraint and it's called the Instruction Level Parallelism (ILP) wall [23].

The ILP can be associated with a technique that tries to reorganize the assembly line machines, in order to improve the performance. Additionally, there have been other different ways to improve the performance, one of them is using new machines more efficient that can assemble the parts faster. Also these new machines could be smaller in size, occupying less space, allowing, therefore more machines in the same area, increasing the productivity. In the CPU context, it has been the improvement of the integrated circuit manufacturing which has allowed the deployment of smaller transistors. This supports the implementation of more dense systems (i.e. the greatest number of transistors per chip) and faster (i.e. that can perform more instructions per unit time).

The growth rate of the amount of transistors in integrated circuits has faithfully obeyed Moore's Law, it allows to have smaller transistors on a chip. As the transistors were becoming

smaller, their capacitances were reducing allowing shorter charges and discharges times. All this allow higher working frequencies and there is a direct relationship between the working frequency of an integrated circuit and power dissipation (given by equation 1 taken from [9]).

$$P = C\rho\, f V_{dd}^2 \tag{1}$$

where ρ represents the transistor density, f is the working frequency of the chip, V_{dd} is the power supply voltage and C is the total capacitance. Nowadays the power dissipation has grown a lot and has become an unmanageable problem with the conventional cooling techniques. This limitation on the growth of the processors is known as the power wall and has removed one of the major growth forces of the CPUs.

3.3. Memory performance

Returning to our assembly line, we saw it with a lot of high speed machines, but now its performance could be limited by the second constraint of the system performance. The speed of the incoming parts to the assembly line could not be enough to keep all these machines busy. If we don't have all parts ready, we will be forced to stop until they become available again. In the same way, faster CPUs require more data ready to be processed. The limitation presented nowadays in the communication channels to supply the demand of data processing units is called memory wall.

This wall was initially treated by improving the access paths between the warehouse and the assembly line, which in a CPU, means improving the transmission lines between memory and CPU on the printed circuit board.

The second form, the memory wall was faced, is more complex, but takes into account all the process since the manufacturer deliver. Analyzing the features of the warehouse parts, we can find different kinds. The part manufacturers have large deposits with a lot of parts ready to be used, but use a part from these deposits is expensive in time. On the other hand, the warehouse next to the assembly line have stored a smaller number of pieces of different kinds, because we must remember that the assembly line can produce different types of cars. The advantage of this warehouse is that the parts reach faster the points of the production line where they are needed. Some deposits are larger but their transportation time to the line is slower, besides, the nearest deposit has a lower storage capacity. The line could improve its performance taking advantage of this deposit features.

Similarly occurs with the computing system memory. High capacity memory such as hard disks, are read by blocks, and access one single data requires to transport a large volume of information that is locally stored in RAM, which represents a nearest deposit. Then the single data can be read and taken to the CPU. The memory organization in a modern computing system is arranged hierarchically as the figure 6 shows. Even in this organization, the assembly line has provided small storage spaces next to the machines that require specific parts. In a CPU this is called CPU cache.

The memory hierarchy creates local copies of the data that will be handled in certain algorithm on fast memories to speed up its access. The challenge of such systems is always have available the required data in the fast memory, because otherwise, the CPU must stop while the data is accessed in the main memory.

Currently, the three computing barriers are present and they are the main cause of stagnation in which CPU based technology has fallen. Some solutions have been proposed based on alternative technologies, such as the graphics processing units (GPU) and Field Programmable Gate Array (FPGA), and have intended to mitigate these phenomena, [4, 22]. They seem to be a short and mid-term solutions respectively.

4. GPUs: A computing short term alternative

GPUs are the product of the gaming industry development that was born in the 50' and started a continuous growing market on 80'. Video-games require to execute intensive computing algorithms to display the images, but unlike other applications such as seismic migration, the interaction with the user requires the execution in a very short time. Therefore, since the 90' have been developed specialized processors for this purpose. These processors have been called GPU, and they have been widely commercialized in video-game consoles and PC video acceleration cards, [37].

GPUs are specific application processors that reach high performance on the task that they were designed for, in the same way that the assembly line would outperform itself if it were dedicated to assembly a little range of vehicle types. This is achieved because the unnecessary machines can be eliminated and the free area is optimized. Likewise improve the availability, storage and handling of the parts(See figure 7).

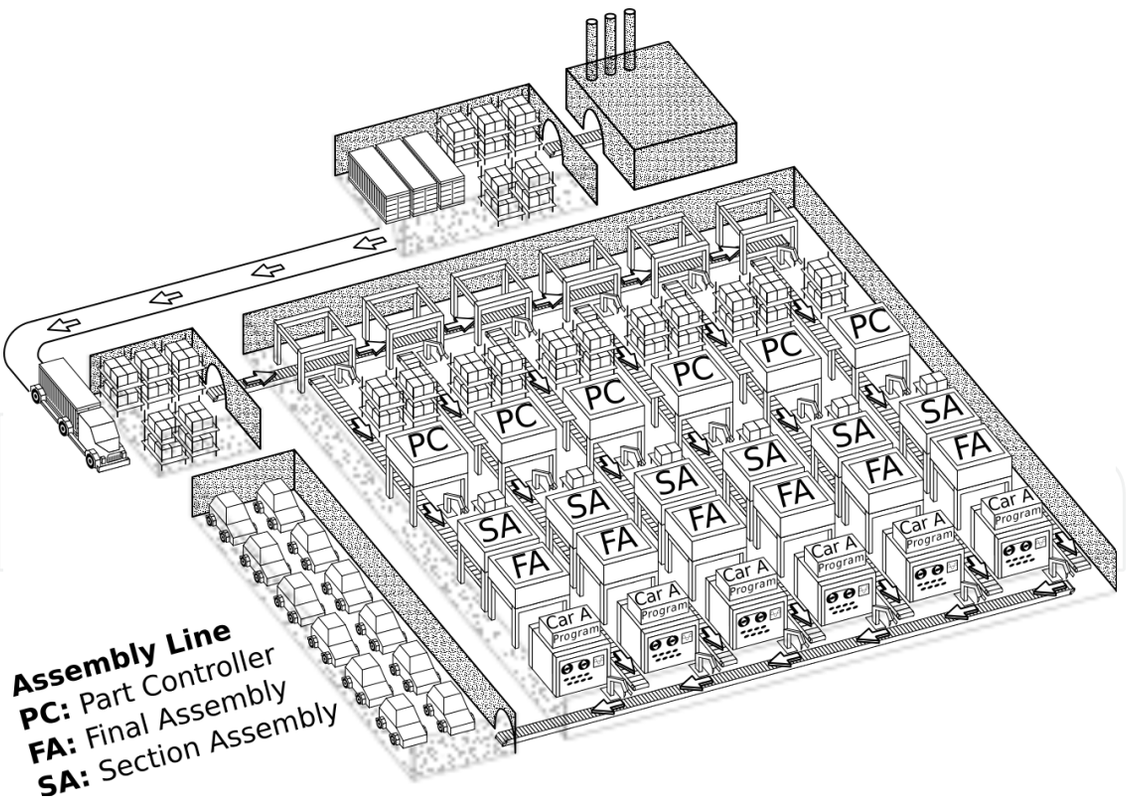

Figure 7. GPU analogy

The task of a GPU is an iterative process that generates an image pixel by pixel (point by point) from some data and input parameters. This allows to process in parallel each output

pixel and therefore GPUs architecture is based on a set of processors that perform the same algorithm for different output pixels. It is very similar to the sum of the contributions made in Kirchhoff migration, so in that way this architecture is a good option to try to speed up a migration process.

The memory organization is another relevant feature of the GPU architecture. This allows all the processors to access common information to all pixels that are being calculated and in the same way each processor can access particular pixel information.

Like our assembly line, the GPU task can be segmented in several stages, elementary operations or specific graphics processing instructions. Initially GPUs could only be used in graphical applications and although their performance in these tasks was far superior than a CPU, its computing power could only be used to carry out this task. For this reason the GPU manufacturers made a first contribution in order to make them capable to execute any algorithm; they make their devices more flexible and have become General Purpose GPU (GPGPU).

This make possible to exploit the computing power of these devices in any algorithm, but it was not an easy task for programmers. Making an application required a deep understanding of the GPGPUs instructions and architecture, so programmers were not very captivated. Therefore, the second contribution that definitely facilitated the use of these devices was the development in 2006 of programming models that does not require advanced knowledge on GPUs such as Compute Unified Device Architecture (CUDA).

GPGPUs are definitely an evolved form of a CPU. Their memory management is highly efficient, which has reduced the memory wall; and because of the number of processing cores on a single chip, they have been forced to reduce the working speed and to operate at the limit of the power wall. Its growth rate seems to continue stable and it promises to be in a mid-term with the technology that drives the high performance computing. But these three barriers are still present and this technology soon will be faced them.

5. FPGAs: Designing custom processors

Another alternative to accelerate the SM process are the FPGAs (Field Programmable Gate Array), these devices are widely used in many problems of complex algorithms acceleration, for this reason, since a couple of years, some traditional manufacturers of high performance computing equipment began to include in their brochure, computer systems that include FPGAs.

To get an initial idea of this technology, let us imagine that the car assembly is going to be amended as follows:

- The assembly plant will produce only one type of vehicle at a time, the purpose is to redesign the entire assembly plant in order to concentrate efforts and increase production.
- After selecting the vehicle that will be produced, the functional units required for assembly the vehicle are designed.
- After designing all the functional units, the assembly line is designed. Both (the assembly line and the control unit) will be a little bit simpler, because now they have less functional

units (remember that now only have the functional units required to assembly only a vehicle type).

- In order to increase the production will be replicated the assembly line as many times as possible and the number of replicates will be determined taking into account two aspects: in the first place, the speed with which the inputs can be placed at the beginning of the assembly line and secondly by the space available within the company.

5.1. FPGA architecture

An FPGA consists mainly of three types of components (see Figure 8): Configurable Logic Blocks (or Logic Array Block depending on vendor), reconfigurable interconnects and I/O blocks.

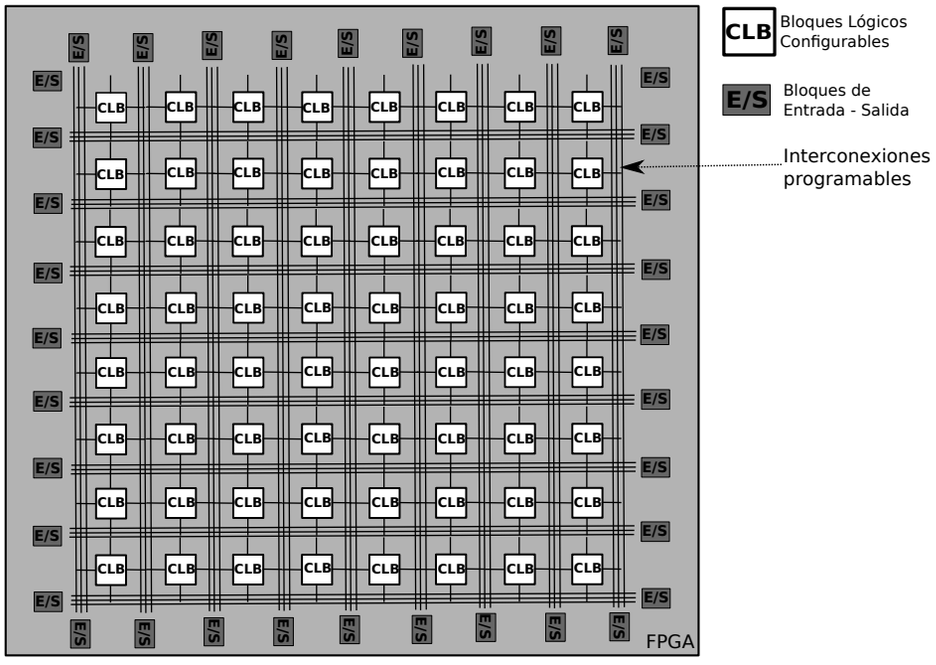

Figure 8. FPGA components.

In the analogy of the car assembly line, the Configurable Logic Blocks (CLBs) come to be the raw material for the construction of each one of the functional units required to assemble the car. In an FPGA the CLBs are used to build all the functional units required in an specific algorithm, these units can go from simple logical functions (and, or, not) to mathematical complex operations (algorithms, trigonometric functions, etc.). Therefore, a first step in the implementation of an algorithm over an FPGA is the construction of each one of the functional units required by the algorithm to be implemented (see figure 9). The design of these functional units is carried out by means of Hardware Description Languages (HDLs) like VHDL or Verilog.

The design of a functional unit using HDLs is not simply to write a code free of syntax errors, the process corresponds more to the design of a digital electronic circuit [13] which requires a good grounding about digital hardware. It must be remembered that this process consumes more time than an implementation on a CPU or a GPU.

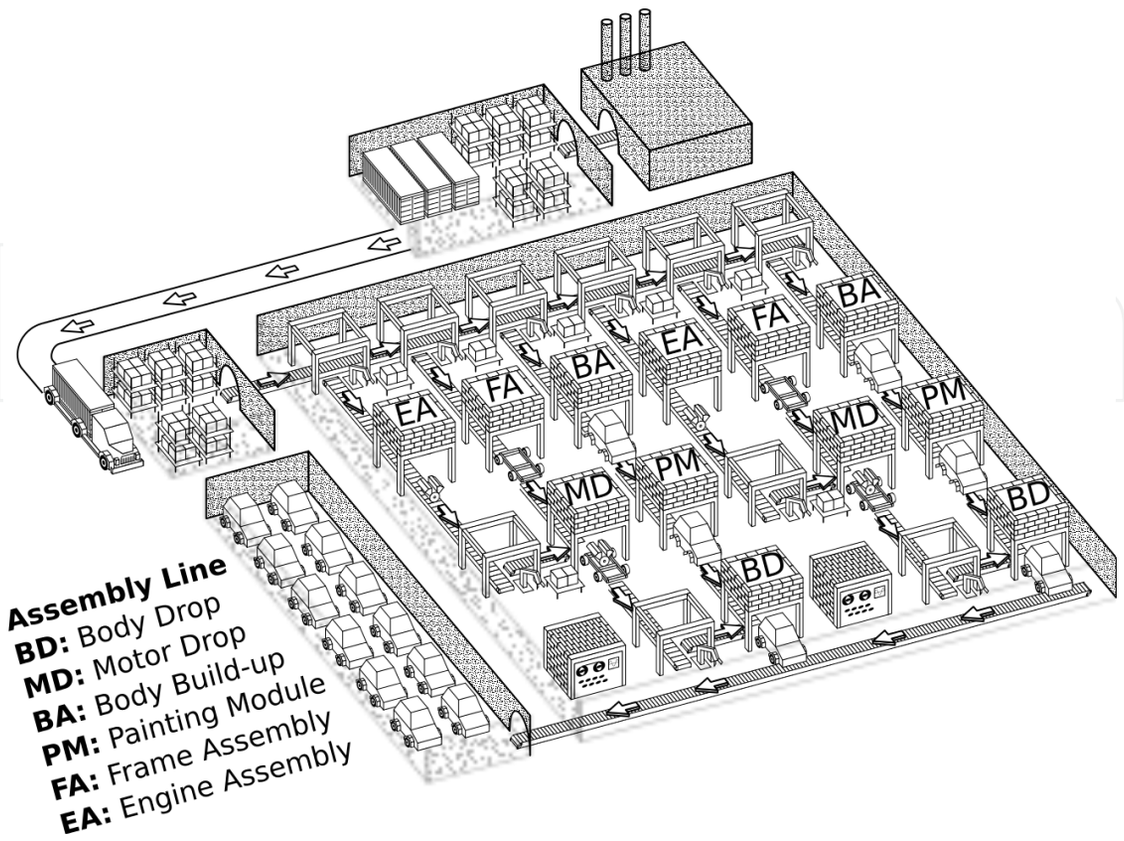

Figure 9. FPGA analogy.

Currently, providers of FPGAs (within which highlights Xilinx [42] and Altera [2]) offer predesigned modules that facilitate the design, in this way, commonly used blocks (as floating point units) should not be designed from scratch for each new application; these modules are offered as an HDL description (softcore modules) or physically implemented inside the FPGA (modules hardcore).

The reconfigurable interconnects, allows to interconnect the CLBs and the functional units each other, in the analogy of the car assembly line, the reconfigurable interconnects can be viewed as the conveyor belt where circulating the parts necessary for the assembly of the cars.

Finally, the I/O blocks allow to communicate the pins of the device and the internal logic of the FPGA, in the analogy of the car assembly line, the I/O blocks are the devices that place the parts at the beginning of the conveyor belt and also the devices that allow to bring out the cars of the factory. The blocks of input-output are responsible for controlling the flow of data between physical devices (extern) and the functional units in the interior of the FPGA. The different companies design these blocks in order to support different digital communication standards.

5.2. Algorithms that can be accelerated in an FPGA

Not all the applications can be benefited in the same way in an FPGA and due to the difficulty of implementing an application in these devices, it is advisable to do some previous estimates in order to determine the possibilities to speed up an specific algorithm.

Applications that require processing large amounts of data with little or no data dependency, [3] are ideal for implementing in the FPGAs, in addition it is required that these applications are limited by computing and not by data transfer, that is to say, the number of mathematical operations is greater than the number of read and write operations, [20]. In this regard, the SM have in favor that requires processing large amounts of data (in order of tens of Terabytes) and is required to perform billions of mathematical operations. However, migration also have a large number of read and write instructions which cause significant delays in the computational process.

Furthermore, the accuracy and the data format are other influential aspects on the performance of the applications over the FPGAs, with a lower data accuracy (least amount of bits to represent data), the performance of the application inside the FPGA increase; regarding the data format, the FPGAs get better performances with fixed point numbers than floating point numbers. The SM has been traditionally implemented using floating point numbers with single precision, [1, 21]. [14, 16] show that it is possible to implement some parts of the SM using fixed point instead of floating point and produce images with very similar patterns (visually identical), reducing computation times.

The FPGAs have a disadvantage in terms of the operation frequency, if they are compared with a CPU (10 times lower) or a GPGPU (5 times), for this reason, in order to accelerate an application inside an FPGA is required to perform at least 10 times more computational work per clock cycle than the performed in a CPU, [20]. In this regard, it is important that the algorithm has a great potential of parallelization. So in this aspect, it is helpful for the SM that the traces can be processed independently, due that the shots are made in different times, [4], which facilitates the parallelization of the process.

5.3. Implementation of the SM on FPGAs

Since 2004, it began to be implemented on the FPGAs, processes related with the construction of subsurface imaging. These implementations have made great contributions to the problems of processing speed and memory.

5.3.1. Breaking the power wall

As mentioned above, the operating frequency of traditional computing devices has been stalled for several years, due to problems related to the power dissipation. On the other hand, to mitigate the problems associated with the processing speed inside the FPGAs, it has been implemented modules that optimize the development of expensive mathematical operations[4], for example in [21] functional units were designed using the Cordic method, [3] to perform the square root, in [17] are used Lee approximations, [28] to perform trigonometric functions. Other research has addressed this problem from the perspective of the representation of numbers [17], the purpose is to change the single-precision (32-bit) floating-point format to fixed-point format (this is not possible, either for CPUs or GPUs) or a floating point format of less than 32 bits, in order to that the mathematical operations can be carried out in less time.

[3] The data dependency is one situation in which the instructions of a program require results of previous ones that have not yet been completed.

[4] The expensive mathematical operations are those that take more clock cycles to complete the process and consume more hardware resources; the expensive operations that are used in seismic processing are the square roots, logarithms and trigonometric functions, among others.

All these investigations have reported significant reductions in processing times of the SM sections when were compared with state of the art CPUs.

5.3.2. Breaking the memory wall

Computing systems based on FPGAs have both on-board memory[5] as on-chip memory [6], these two types of memory are faster than the external memories. Some research has made implementations in order to reduce the delays caused in the reading and writing operations [22], in their researches, have been designing special memory architectures that allow to optimize the use of different types of memory with FPGAs. The intent is that the majority of read and write operations must be performed at the speed of the on chip memory because this is the fastest, however, the challenge is to put all the data required in each instruction in on-chip memory because this is the one of the smaller capacity.

6. Wishlist

In spite of the great possibilities that have both FPGAs and GPGPUs to reduce the computation times of the seismic processes, their performance at this time is braking, because the rate at which seismic data are transmitted from the main memory to the computing device (FPGA or GPU) is not enough to maintain its computing potential 100% busy. The communication between the FPGAs or GPUs with the exterior can be carried out using large amount of output interfaces, currently one of the most used is Peripheral Component Interconnect Express (PCIe) port that transfer data between the computing device (FPGA or GPU) and a CPU at speeds of 2GB/sec, [14] and this transfer rate can not provide all the necessary data to keep the computing device busy.

Currently, at the Universidad Industrial de Santander (Colombia), a PhD project is being carried out that seeks to increase the transfer rate of seismic data between the main memory and the FPGA using data compression.

In addition, there are currently two approaches to try to reduce the design time on FPGAs (the main problem of this technology): The first strategy is to use compilers CtoHDL [11, 25, 35, 39, 43], these from a C code generates a description in a HDL, without having to manually perform the design; the second strategy is to use high level languages, these languages are the called Like-C as Mitrion-C [31] or SystemC [26]; despite their need, these languages have not yet achieved wide acceptance in the market, because its use still compromise the efficiency of the designs. The research regarding the compilation CtoHDL and high-level languages are active [29, 33, 34, 40, 44] and some results have been positive [7], but the possibility of having access to all the potential of the FPGAs from a high-level language is still in development.

On the other hand, it can be seen that the technological evolution of the GPUs is similar to the beginnings of the PCs evolution, where their progress was subject to Moore's law. It is therefore expected that in coming years new GPGPUs families continue to appear, increasingly so much that are going to raid in many areas of the HPC, but definitively it will come the time when this technology will stagnate for the same three barriers that stopped the

[5] This is the available memory on the board which contains the FPGA
[6] These are blocks of memory inside the FPGA

computers. When that time comes it will be expected that the FPGAs will be technological mature and can take the HPC baton during the next years.

At the end, we believe that the HPC future is going to be built of heterogeneous cluster, composed by these three technologies (CPUs, GPUs and FPGAs). These clusters will have an special operating system (O.S.) that will take advantage of each of these technologies, reducing the three barriers effect and getting the best performance in each application.

Acknowledgments

The authors wish to thank the Universidad Industrial de Santander (UIS), in particular the research group in Connectivity and Processing of Signals (CPS), head by Professor Jorge H. Ramon S. who has unconditionally been supporting our research work. Additionally we thank also to the Instituto Colombiano del Petroleo (ICP) for their constant support in recent years, especially thanks to Messrs William Mauricio Agudelo Zambrano and Andres Eduardo Calle Ochoa.

Author details

Sergio Abreo, Carlos Fajardo, William Salamanca and Ana Ramirez
Universidad Industrial de Santander, Colombia

7. References

[1] Abreo, S. & Ramirez, A. [2010]. Viabilidad de acelerar la migración sísmica 2D usando un procesador específico implementado sobre un FPGA The feasibility of speeding up 2D seismic migration using a specific processor on an FPGA, *Ingeniería e investigación* 30(1): 64–70.

[2] *Altera* [n.d.]. `http://www.altera.com/`. Reviewed: April 2012.

[3] Andraka, R. [1998]. A survey of cordic algorithms for fpga based computers, *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, FPGA '98, ACM, New York, NY, USA, pp. 191–200.
URL: *http://doi.acm.org/10.1145/275107.275139*

[4] Araya-polo, M., Cabezas, J., Hanzich, M., Pericas, M., Gelado, I., Shafiq, M., Morancho, E., Navarro, N., Valero, M. & Ayguade, E. [2011]. Assessing Accelerator-Based HPC Reverse Time Migration, *Electronic Design* 22(1): 147–162.

[5] Asanovic, K., Bodik, R., Catanzaro, B. C., Gebis, J. J., Husbands, P., Keutzer, K., Patterson, D. A., Plishker, W. L., Shalf, J., Williams, S. W. & Yelick, K. A. [2006]. The landscape of parallel computing research: A view from berkeley, *Technical Report UCB/EECS-2006-183*, EECS Department, University of California, Berkeley.
URL: *http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html*

[6] Bednar, J. B. [2005]. A brief history of seismic migration, *Geophysics* 70(3): 3MJ–20MJ.

[7] Bier, J. & Eyre, J. [Second Quarter 2010]. BDTI Study Certifies High-Level Synthesis Flows for DSP-Centric FPGA Designs, *Xcell Journal Second* pp. 12–17.

[8] Bit-tech staff. [n.d.]. Intel sandy bridge review., `http://www.bit-tech.net/`. Reviewed: April 2012.

[9] Brodtkorb, A. R., Dyken, C., Hagen, T. R. & Hjelmervik, J. M. [2010]. State-of-the-art in heterogeneous computing, *Scientific Programming* 18: 1–33.

[10] Brodtkrorb, A. R. [2010]. *Scientific Computing on Heterogeneous Architectures*, Phd thesis, University of Oslo.
[11] C to Verilog: automating circuit design [n.d.]. `http://www.c-to-verilog.com/`. Revisado: Junio de 2011.
[12] Cabezas, J., Araya-Polo, M., Gelado, I., Navarro, N., Morancho, E. & Cela, J. M. [2009]. High-Performance Reverse Time Migration on GPU, *2009 International Conference of the Chilean Computer Science Society* pp. 77–86.
[13] Chu, P. [2006]. *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability*, Wiley-IEEE Press.
[14] Clapp, R. G., Fu, H. & Lindtjorn, O. [2010]. Selecting the right hardware for reverse time migration, *The Leading Edge* 29(1): 48.
URL: *http://link.aip.org/link/LEEDFF/v29/i1/p48/s1&Agg=doi*
[15] Cleveland, C. [2006]. Mintrop, Ludger, *Encyclopedia of Earth* .
[16] Fu, H., Osborne, W., Clapp, R. G., Mencer, O. & Luk, W. [2009]. Accelerating seismic computations using customized number representations on fpgas, *EURASIP J. Embedded Syst.* 2009: 3:1–3:13.
URL: *http://dx.doi.org/10.1155/2009/382983*
[17] Fu, H., Osborne, W., Clapp, R. G. & Pell, O. [2008]. Accelerating Seismic Computations on FPGAs From the Perspective of Number Representations, *70th EAGE Conference & Exhibition* (June 2008): 9 – 12.
[18] Hagedoorn, J. [1954]. *A process of seismic reflection interpretation*, E.J. Brill.
URL: *http://books.google.es/books?id=U6FWAAAAMAAJ*
[19] Harris, D. M. & Harris, S. L. [2009]. *Digital Desing and Computer Architecture*, MK.
[20] Hauck Scott, D. A. [2008]. *Reconfigurable computing. The theory and practice of FPGA-BASED computing*, ELSEVIER - Morgan Kaufmann.
[21] He, C., Lu, M. & Sun, C. [2004]. Accelerating Seismic Migration Using FPGA-Based Coprocessor Platform, *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines* pp. 207–216.
[22] He, C., Zhao, W. & Lu, M. [2005]. Time domain numerical simulation for transient waves on reconfigurable coprocessor platform, *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, IEEE Computer Society, pp. 127–136.
[23] Hennessy, J. L. & Patterson, D. A. [2006]. *Computer Architecture A Quantitative Approach*, 4th edn, Morgan Kaufmann.
[24] IEEE Computer Society [n.d.]. Tribute to Seymour Cray, `http://www.computer.org/portal/web/awards/seymourbio`. Reviewed: April 2012.
[25] Impulse Accelerate Technologies [n.d.]. Impulse codeveloper c-to-fpga tools, `http://www.jacquardcomputing.com/roccc/`. Revisado: Agosto de 2011.
[26] Initiative, O. S. [n.d.]. Languaje SystemC, `http://www.systemc.org/home/`. Reviewed: April 2012.
[27] Intel [2011]. Intel core i7-3960x processor extreme edition, `http://en.wikipedia.org/wiki/FLOPS`. Reviewed: April 2012.
[28] Lee, D.-U., Abdul Gaffar, A., Mencer, O. & Luk, W. [2005]. Optimizing hardware function evaluation, *IEEE Trans. Comput.* 54: 1520–1531.
URL: *http://dl.acm.org/citation.cfm?id=1098521.1098595*
[29] Lindtjorn, O., Clapp, R. G., Pell, O. & Flynn, M. J. [2011]. Beyond traditional microprocessors for Geoscience High-Performance computing a pplications and Geoscience, *Ieee Micro* pp. 41–49.

[30] Mayne, W. H. [1962]. Common reflection point horizontal data stacking techniques, *Geophysics* 27(06): 927–938.
[31] Mitrionics [n.d.]. Languaje Mitrion-C, `http://www.mitrionics.com/`. Reviewed: April 2012.
[32] Moore, G. E. [1975]. Progress in digital integrated electronics, *Electron Devices Meeting, 1975 International*, Vol. 21, pp. 11–13.
[33] Necsulescu, P. I. [2011]. *Automatic Generation of Hardware for Custom Instructions*, PhD thesis, Ottawa, Canada.
[34] Necsulescu, P. I. & Groza, V. [2011]. Automatic Generation of VHDL Hardware Code from Data Flow Graphs, *6th IEEE International Symposium on Applied Computational Intelligence and Informatics* pp. 523–528.
[35] Nios II C-to-Hardware Acceleration Compilern [n.d.]. `http://www.altera.com/`. Revisado: Junio de 2011.
[36] Onifade, A. [2004]. History of the computer, *Conference of History of Electronics*.
[37] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E. & Phillips, J. C. [2008]. Gpu computing, *Proceedings of the IEEE* 96(5): 879–899.
[38] Rieber, F. [1936]. Visual presentation of elastic wave patterns under various structural conditions, *Geophysics* 01(02): 196–218.
[39] Riverside Optimizing Compiler for Configurable Computing: Roccc 2.0 [n.d.]. `http://www.jacquardcomputing.com/roccc/`. Revisado: Junio de 2011.
[40] Sánchez Fernández, R. [2010]. *Compilación C a VHDL de códigos de bucles con reuso de datos*, Tesis, Universidad Politécnica de Cataluña.
[41] Stolt, R. H. [1978]. Migration by fourier transform, *Geophysics* (43): 23–48.
[42] *Xilinx* [n.d.a]. `http://www.xilinx.com/`. Reviewed: April 2012.
[43] Xilinx [n.d.b]. High-Level Synthesis: AutoESL, `http://www.xilinx.com/university/tools/autoesl/`. Reviewed: April 2012.
[44] Yankova, Y., Bertels, K., Vassiliadis, S., Meeuws, R. & Virginia, A. [2007]. Automated HDL Generation: Comparative Evaluation, *2007 IEEE International Symposium on Circuits and Systems* pp. 2750–2753.