FPGA and Field Programmable Devices architectures : A tutorial

Hachour Ouarda

Abstract— Configurable hardware is an approach for realizing optimal performance by tailoring its architecture to the characteristics of a given problem FPGAs Field Programmable Gate Arrays .The complexity of VLSI circuits is being more and more complexes. Recently, the development of new type of sophisticated Field Programmable Devices (FPDs) has dramatically changed the process of designing digital hardware. Unlike previous generations of hardware technology in which board level designs included large numbers of SSI (Small Scale Integration) chips containing basic gates, virtually every digital design produced today consists mostly of high-density devices. . This is true not only of custom devices such as processors and memory but also of logic circuits such as state machine controllers, counters, registers, and decoders. When such circuits are destined for high volume systems, designers integrate them into high-density gate arrays. AS an attractive solution for the computationally –intensive functions FPD, we present the FPGA Field Programmable Gate Arrays which are used to prototype ASIC designs and are considered as specific purposeThe complex issue of programming FPGA may be approached in a wide range of ways. One extreme is to consider that the designer holds only have to ketch his design in an abstract way, leaving to automatic tools as much of the implementation job as possible, with as little human intervention as possible. This hands-off approach reduces development time and costs, at their expense of the performance of the implementation. At the other extreme, when performance is critical, the designer has to intervene in the whole design process. This may include low-level implementation work and require important expert knowledge and much longer development time. FPGAs are often used to prototype ASIC designs or to provide a hardware platform on which to verify the physical implementation of new algorithms. However their low development cost and short time to market mean that they are increasing finding their way into final products. In this paper we present a review of FPD Field Programmable Devices; where we present the importance of using FPGA circuit**.**

Keywords— Field Programmable Devices (FPD), (FPGAs) Field Programmable Gate Arrays, ASICs (Application Specific Integrated Circuits), VLSI (Very Large Scale Integration), Field Programmable Logic Array (PLA).

I. Introduction

The FPD market has grown over the past decade to the point where there is now a wide assortment of devices to choose from. To choose a product, designers face the daunting task of researching the best uses of the various chips and learning the intricacies of vendor-specific software.

Recently, the development of new type of sophisticated Field Programmable Devices (FPDs) has dramatically changed the process of designing digital hardware.

Unlike previous generations of hardware technology in which board level designs included large numbers of SSI (Small Scale Integration) chips containing basic gates, virtually every digital design produced today consists mostly of high-density devices. This is true not only of custom devices such as processors and memory but also of logic circuits such as state machine controllers, counters, registers, and decoders. When such circuits are destined for high volume systems, designers integrate them into high-density gate arrays.

However, the high nonrecurring engineering costs and long manufacturing time of gate arrays make them unsuitable for prototyping or other low volume scenarios. Therefore, most prototypes and many production designs now use FPDs. The most compelling advantages of FPDs are now startup cost, low financial risk, and, because the end user programs the device, quick manufacturing turnaround and easy design changes.

The complex issue of programming Field Programmable Gate Arrays FPGA which is a best component of FPD may be approached in a wide range of ways. One extreme is to consider that the designer holds only have to ketch his design in an abstract way, leaving to automatic tools as much of the implementation job as possible, with as little human intervention as possible.

There are many different types of digital integrated circuits ICs, including "jelly-bean logic" (small components containing a few simple fixed logical functions), memory devices, and micro processors (Ups) of particular interest to use here, however , are programmable logic devices (PLDs) , Application Specific Integrated Circuits (ASICs) , Application Specific Standard Parts (ASSPs) and of course FPGA.

When they first arrived on the scene in the mid 1980s, FPGAs were largely used to implement glue logic (the term glue logic refers to the relatively small amounts of simple logic that are used to connect ("glue") and interface between larger logical blocks, functions or devices).

Decade after, the sophistication of FPGAs stated to increase their big market at that time were in the telecommunication and networking arenas, both of which involved processing large blocks of data and pushing that data around later are used in consumer, automotive, and industrial applications underwent a humongous growth spurt.

FPGAs are often used to prototype ASIC designs or to provide a hardware plat form on which to verify the physical implementation of new algorithms. However their low development cost and short time to market mean that they are increasingly finding their way into final products

Now, high performances FPGAs are containing millions of gates had become available. In this present work we present a tutorial of different field programmable devices. AS an attractive solution for the computationally –intensive functions FPD, we present the FPGA Field Programmable Gate Arrays which are used to prototype ASIC designs and are considered as specific purpose. In comparison to an FPGA, THE FPD devices contain a relatively limited number of logic gates, and the functions they can be used to implement are much smaller and simpler. At the other end of the spectrum are ASICs and ASSPs, which can contain hundreds of millions of logic gates and can be used to create incredibly large and complex function but are frozen to Silicon area.

FPGA offers more attractive features such as: high number of logic gates; short time to market, independency, an easiness of work, low cost, possibility of prototyping a lot of functions and feasibility. FPGA implementing design changes is much easier than an ASIC chip [7,8,9,10,11]. Without returning to the manufacturer, FPGA facilitates a lot of operations without penalty of time cost and the dependence of the vendor. The dream of manufacturing a hardware VLSI specific purpose chip is realized with FPGA circuit. FPGA based test plat form without having to incur the enormous nonrecurring engineering (NRE) costs or purchase the expensive toolsets associated with ASIC Design. This is clarified clearly with this tutorial.

II. Evolution of FPDs

The first user-programmable chip that could implement logic circuits was the Programmable Read Only Memory (PROM), in which address lines serve as logic circuit inputs and data lines as outputs. Logic functions, however rarely require more than a few product terms, and a PROM contains a full decoder for its address inputs. PROMs are thus inefficient for realizing logic circuits, so designers rarely use them for that purpose.

The first device developed specifically for implementing logic circuits was the field Programmable Logic Array PLA. This device consists of two levels of logic gates: a programmable, wired –AND plane followed by a programmable, wired OR plane.

A PLA’s structure allows any of its inputs (or their complements) to be ANDed together in the AND plane; each AND plane output can thus correspond to any product term of the inputs, see the figure 1. Here, the user can configure each OR plane output to produce the logical sum of any AND plane output. With this structure, PLAs are well-suited for implementing logic functions in sum of products form.
They are also quite versatile, since both the AND and OR terms can have many inputs (product literature often calls this feature “wide AND and OR gates”).

With the progress of PLA devices, we group all small FPD including PALs (Programmable Array Logic), PLAs and PAL like device into the single category of simple programmable-logic devices (SPLDs), whose most important characteristics are low cost and very high-pin-to-pin speed performance

It has been shown that the only feasible way to provide large-capacity devices based on SPLD architectures is to programmable interconnecting multiple SPLDs on a single chip. Now, many FPD products on the market today have this basic feature and are called as Complex Programmable Logic Devices CPLDs. CPLDs provide logic capacity up to the equivalent of about 50 typical SPLD devices, but extending these architectures to higher densities is difficult. Building FPDs with very high logic capacity requires a different appr

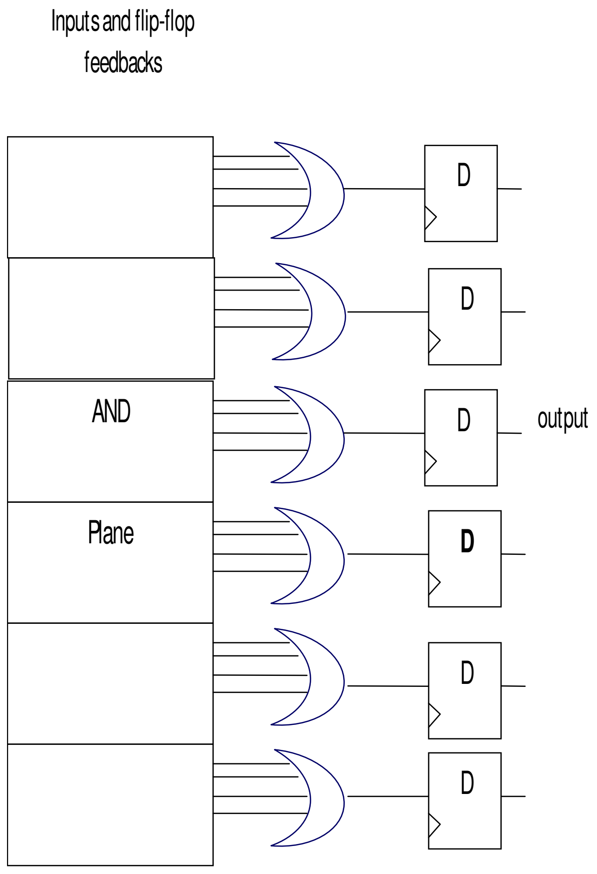

Fig. 1 PAL structure

We don't forget that the highest capacity general purpose logic ships available today are the traditional gate arrays sometimes referred to as Mask-Programmable Gate Arrays. An MPGA consists of an array of prefabricated transistors customized for the user's logic circuit by means of wide connections. Because the silicon foundry performs customized during chip fabrication; the manufacturing time is long, and the user's setup cost is high.

Although MPGAs are clearly not FPDs, but they are clearly not FPDs, they motivated the design of the field programmable equivalent, FPGAs .Like MPGAs, an FPGA consists of an array of uncommitted circuit elements (logic blocks) and interconnect resources, but the end user configures the FPGA though programming.

III. User programmable switch technologies

The key to user customization of FPDS is the user – programmable switches. The "fuse" used in PLAs are the first user-programmable switch developed. But with the newer technology and for higher density devices CMOS dominates the IC" Integrated Circuit" industry and different approaches to implementing programmable switches are necessary. The main switch technologies used in CPLD commercial products are floating gate transistors like those used in EPROM (Erasable Programmable Read Only Memory) and EEPROM (Electrically Erasable PROM) .

Just the manufacturer places the transistors between two wires to facilitate implantation of wired –AND functions, this done in order to use an EPROM or EEPROM transistors as a programmable switch. The figure 2 shows EPROM transistors connected in a CPLDs AND plane.

An input to the AND plane can drive a product wire to logic level 0 trough an EPROM transistor , if that input is part of the corresponding product term. For inputs not involved in a product term, the appropriate EPROM transistors are programmed as permanently turned off.

For FPGAs , they are (SRAM) Static Random Access Memory and anti fuse. To understand better the main use of switch type, the table 1 illustrates FPD programming technology.

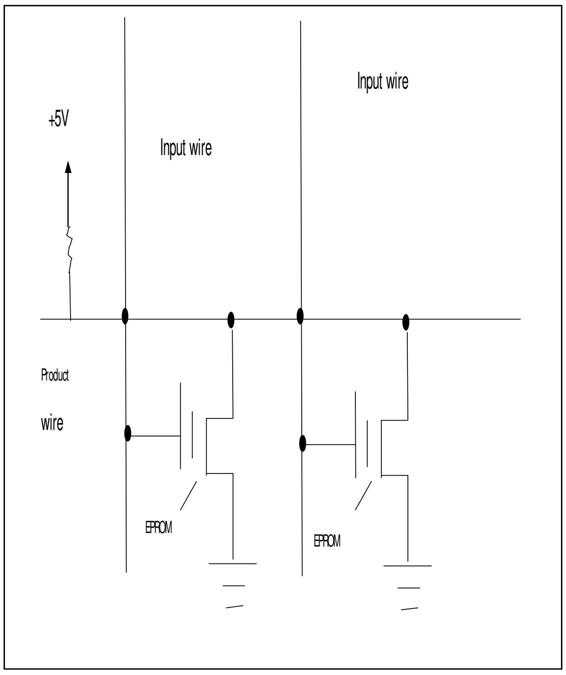

Fig. 2 EPROM programmable switches

Switch type	Reprogrammable?	Volatile?	Technology
Fuse	No	No	Bipolar
EPROM	Yes(out of circuit)	No	UVCMOS
EEPROM	Yes (circuit)	No	EECMOS
SRAM	Yes (in circuit)	Yes	CMOS
ANTIFUSE	No	No	CMOS+

Table 1: summary of FPD programming technology

No technical reason prevents application of EPROM or EEPROM to FPGAs; current commercial FPGA products use either SRAM or anti-fuse technologies. Whether an FPGA uses pass transistors, multiplexers or both depends on the particular product. Antifuses are originally open circuits that take on low resistance only when programmed. Antifuses are manufactured using modified CMOS technology.

As an Example, figure 3 depicts Actel's PLICE (programmable logic interconnect circuit element). As we can see, the antifuse is positioned between two interconnect wires, consists of three sandwiched layers: conductors at top and bottom and an insulator in the middle.

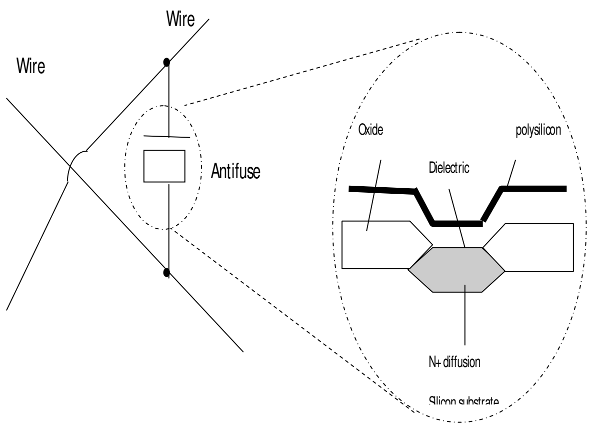

Fig. 3 Actel's PLICE Antifuse structure

IV. FPGA AND FPD

The complex issue of programming Field Programmable Gate Arrays FPGA may be approached in a wide range of ways. One extreme is to consider that the designer holds only have to ketch his design in an abstract way, leaving to automatic tools as much of the implementation job as possible, with as little human intervention as possible.

This hands-off approach reduces development time and costs, at their expense of the performance of the implementation. At the other extreme, when performance is critical, the designer has to intervene in the whole design process. This may include low-level implementation work and require important expert knowledge and much longer development time.

Usually the implementation of a design on FPGA fall somewhere in the middle of these two extremes. The tools, while increasingly useful, still require a lot of technology-dependent knowledge from the designer. The balance between automation and manual intervention ha to be considered in the three steps of a typical implementation flow for an FPGA: technology-mapping, placement and routing. One of the factors to consider in each of these steps, for example, is whether to keep the design hierarchy or flatten it to perform a global analysis. Technology mapping is obviously technology dependent, and currently well handle by automatic tools.

The complexity of VLSI (Very Large Scale Integration) circuits is being more and more complexes. Nowadays, the key of the art design is focused around high level synthesis which is a top down design methodology, that transform an abstract level such as the VHDL language (acronym for Very High Speed Integrated Circuits Hardware Description Language) into a physical implementation level

In addition, the synthesis tools allow designers to realize the mainly reasons: the need to get a correctly working systems at the first time, technology independent design, design reusability, the ability to experiment with several alternatives of the design, and economic factors such as time to market. The result is a netlist ready for place and root using some tools. The intended objective is to, realize architecture that taxes into account the parallelism, performance, flexibility and their relationship to silicon area.

Field Programmable Gate Arrays FPGA are digital Integrated Circuits (ICs) that contain configurable programmable blocks logic CLB along with configurable interconnects between these blocks as shown in the figure 4. As it is shown, the CLB block connected into the interconnection are the main body of this component.

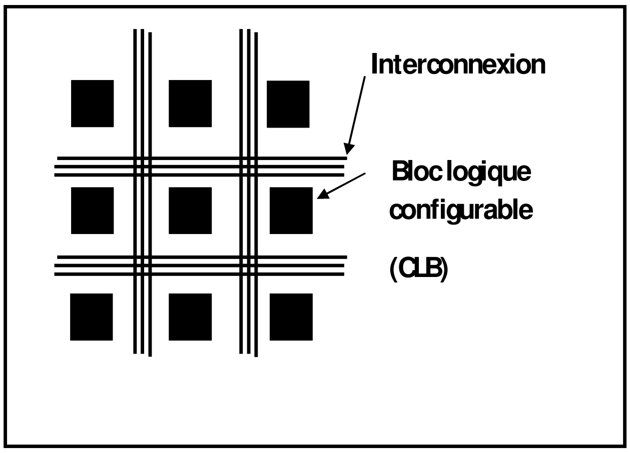

Fig. 4 FPGA Architecture

Depending on the way in which they are implemented , some FPGAs may only be programmed a single time, while the others may be reprogrammed over and over again. We note that , a device that can be programmed only one time is referred to as "***one time programmable OTP***" . The ***field programmable*** portion of the FPGA's name refers to the fact that is its programming takes place in the field , as opposed to devices whose internal functionality is hard-wired by the manufacturers and cannot be changed by user demand. If a device is capable of being programmed while remaining resident in a higher –level system, it is referred to as being In-System Programmable (SP).

Field Programmable Gate Array (FPGAs) are usually programmed using languages and methods inherited from the domain of VLSI (Very Large Scale Integration) synthesis. These methods, however, have not always been adapted to the new possibilities opened by FPGA, nor to the new constraints do they impose on a design.

As an example, CLB architecture (top down of CLB) of some products of FPGA Xilinx is presented in the figure 5. This architecture Illustrates how it is the main body of some Xilinx FPGA architecture is done.

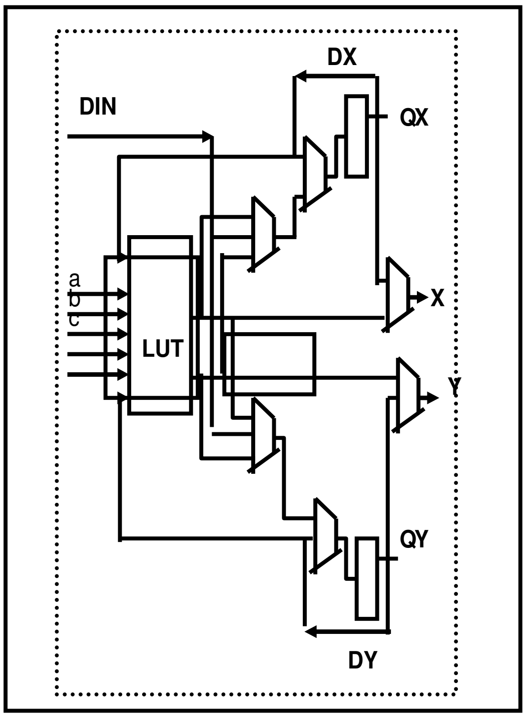

Fig. 5 Configurable Block logic CLB Xilinx – family XC3000 Architecture

V. Commercially available FPDs

In this section, we present a summary of commercial FPD products and their applications.

A. SPLDs

As a staple of digital hardware designers for the past two decades, SPLDs are very important devices. They have the highest speed performance of all FPDs and are expensive. Two of the most popular SPLDS are AMD (Advanced Micro Devices) 16R8 and 22V10PALs. Both of these devices are industry standards, widely second-sourced by other companies the designation 16R8 means that the PAL has a maximum of 16 inputs (eight dedicated inputs and eight inputs/outputs) and a maximum of eight outputs, and that each output is registered by a D Flip-Flop.

Many other SPLD products are available from a wide array of companies. All share common characteristics such as logic planes (AND, OR, NOR, or NAND), but each offers unique features suitable for particular applications.

Another widely used and second sourced SPLD is the Altera classic EP610. This device is similar in complexity to PALs, but offers more flexibility in the production of outputs and has larger AND and OR planes. The EP610's outputs can be registered, and the flip-flops are configurable as D, T, JK and SR or can be transparent [4,5,6].

B. CPLDs

This category consists of multiple SPLD-like blocs on a single chip. However, CPLD products are much more sophisticated than SPLD, even at the level of their basic SPLD-like blocks. Between the most widely used devices, we have:

Altera Max

Altera has developed three families of CPLD chips: MAX5000, MAX7000, and 9000. For the wide use of the 7000 because of its large use and state -of the-art logic capacity and speed performance. MAX5000 represents an older technology that is characterized by the cost effective solution. MAX9000 is similar to MAX7000 but offers higher logic capacity (the industry's highest for CPLDs).

AMD MAC

AMD offers a CPLD family comprising five subfamilies called MACH. Each MACH device consists of multiple PAL-like blocks (or optimized PALs). MACH 1 and 2 consist of optimized 22V16 PALS, MACH 3 and 4 consist of several optimized 34V16 PALs, and MACH 5 is similar to MACH 3 and 4 but offers enhanced speed performance. All MACH chips use EEPROM technology ,and together the five subfamilies provide a wide range of selection, from small, inexpensive chips to larger, state-of-the-are ones.

Lattice pLSI and ispLSI

Lattice offers a complete range of CPLDS, with two main products lines: the pLSI and the ispLSI. Each consists of three families of EEPROM CPLDs with different logic capacities and speed performance. The ispLSI devices are in-system programmable

Cypress flash370.

Cypress has recently developed CPLD products similar to the AMD and Lattice devices in several ways Cypress flash370 CPLDs use flash EEPROM technology and offer speed performance of 8.5 to 15 ns pin-to-pin delays The Flash370 are not in system-programmable

Xilinx XC7000

Although primarily a manufacturer of FPGAs, Xilinx also offers the XC7000 series of CPLDs. The two main XC7000 families are the 7200 series (originally marketed by plus logic Hiper EPLDs) and the 7300series developed by Xilinx. The 7200 offers speed performance of about 25-ns pin-to-pin delays.

VI. FPGA AND ASIC

One of the first things usually notice about printed circuit board design is the use of the "big chips". It is not hard to find the microcontroller, the memories, the FPGAs and the ASICs (Application Specific Integrated Circuits). And, sometimes the unused capacity in one FPGA is enough to replace several

ASICs. Note that if we want to reduce the overall development/manufacturing/test costs it often makes sense to incorporate ASIC functions into our FPGAs [1,2,3].

By designing ASIC functions into an FPGA, we usually save money, power, and board space. Once designed the function becomes part of our company's intellectual property, and be re-used. For example, if an application needs to change from one state to another state , often only the FPGA program needs to change, and the board remains the same. In fact some designs initialize depending on configurations stored in memory, once the application sis selected.

Another good example of the "no more ASICs" design philosophy is in forward error correcting. Most communication channels have errors that occur (fibber, radio, magnetic, or metallic based channels) and need some amount of error correction to make the channel useful. Rather than buying ASICs to do the job, again it makes sense to perform the functions in FPGA.

Some forward error correction algorithms require a large memory block, but using an external RAM device is still less expensive than the ASIC alternative

Some error correcting schemes are fairly easy to implement, and require only feedback shift register structures, such as Reed-Solomon codes.

The simple schemes have a high overhead; they do not correct many errors per block or byte in relation to the extra bits required.

Frequency synthesis is another area where the often expensive and single –sourced parts may be easily replaced by an FPGA. Fractional synthesisers, pulse swallowers, direct digital frequency synthesizers, as well as the phase detectors for phase locked loops, are all easily implemented in FPGAs.

By comparison, the cost of creating an FPGA design is much lower than that for an ASIC or (ASSPs) "Application Specific Standard Parts".

However, designing and building ASICs and ASSPs is an extremely time-consuming and expensive hobby, with the added disadvantage that the final design is "frozen in Silicon" and cannot be easily modified.

FPGAs were largely used to implement glue logic, medium complexity state machines, and relatively limited data processing tasks.

FPGAs are often used to prototype ASIC designs or to provide a hardware platform on which to verify the physical implementation of new algorithms. However their low development cost and short time to market mean that they are increasing finding their way into final products. FPGA vendors actually have devices that they specifically market as competing directly against ASICs. FPGAs are containing millions of gates had become available. The main thing focused about FPGAs is that "***programmable***" , hence, it is the main key of distinguishing an FPGA from another ASIC.

VII. TECHNOLOGY MAPPING

Technology mapping is obviously technology dependent, and currently well handle by automatic tools. The tools, while increasing useful still require a lot of technology dependent knowledge from the designer. The placement is a difficult hard optimization problem. for this point, most designs involve regular of similar components, and there is often a straight forward "ideal" placement, ideal in the sense that it places connected components close to another.

In such case, expressing this placement should greatly improve both implementation time and quality of the result: a general rule of place and route tools (both in VLSI and FPGA) is that if the placement is good (i.e. if it minimizes the distance between connected logic blocks) then the routing will be good as well and, perhaps more important, obtained quickly.

Another strong motivation of expressing placement is run-time partial reconfiguration, as allowed by some recent FPGA. The idea is that one can change a small part of a design run-time, for example to change the value of hard-wired constant without the time penalty of reconfiguring the whole design.

VIII. VHDL CIRCUITS HARDWARE DESCRIPTION LANGUAGE

Field Programmable Gate Array (FPGAs) are usually programmed using languages and methods inherited from the domain of VLSI (Very Large Scale Integration) synthesis. These methods, however, have not always been adapted to the new possibilities opened by FPGA, nor to the new constraints do they impose on a design. For FPGA circuit we can use the VHDL language as hardware description (acronym for Very High Speed Integrated Circuits Hardware Description Language).

The key of the art design is focused around high level synthesis which is a top down design methodology that transforms an abstract level using VHDL description. the synthesis tools allow designers to realize the mainly reasons: the need to get a correctly working systems at the first time, technology independent design, design reusability, the ability to experiment with several alternatives of the design, and economic factors such as time to market.

VHDL allows for the description of hardware behavior from system to gate levels. The system level focuses on the description of the functionalities of the system (what is does) and tries to avoid it implementation description (how it is constituted). The notion of time is essentially a notion of causality: one action implies another. A constant is to forge useless details, which would imply architectural choices too early in the design methodology. Too detailed a system description is a drawback for it restricts further architectural choices or implies a given technology. Therefore, hiding the information structure is desirable and the notion of concurrency may not be necessary at this phase.

To fit this level of description, the language has to offer lager degrees of abstraction, powerful algorithmic, wide capabilities for merging different description levels, and an easiness expression of causality, and also the possibility of introducing non determinism, which may be an interesting feature. To date, this level of description has not been synthesizable: no explicit architecture is described and no tool on the market offers a real and an efficient architectural synthesis (except for some specific target architecture).

The language must allow a description of the model at this level with a sufficient level of abstraction toward the physical level. Clock, sequentiality, dataflows, and combinational art have to be easily expressed. A large degree of parameterization is also required. The figure 6 shows the different possible description levels

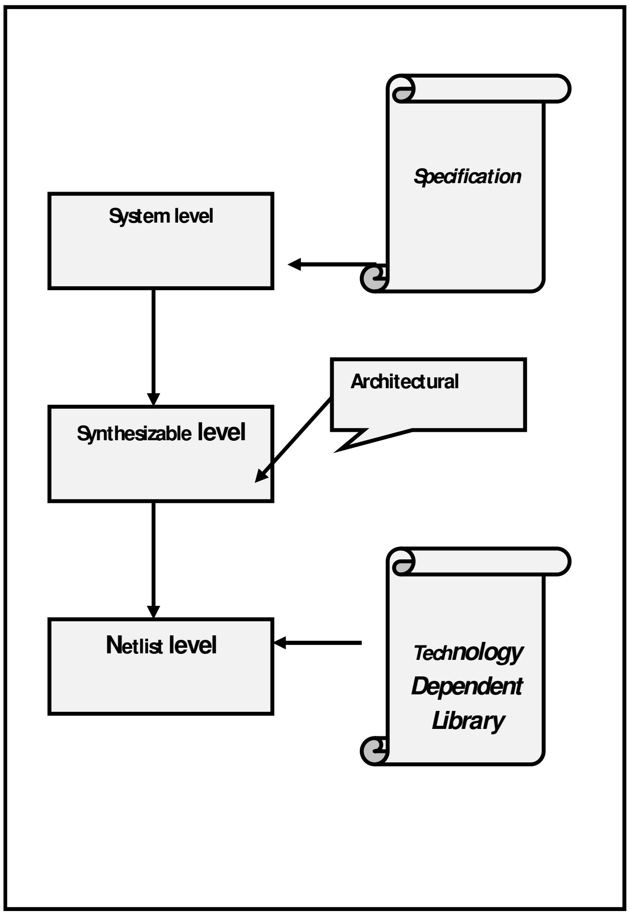

Fig. 6 Different level of Description

IX. Conclusion

In this present work we have presented the main concept of Field Programmable Devices (FPD). From these devices, we have presented the best filed programmable which is: (FPGAs) Field Programmable Gate Arrays. FPGAs were largely used to implement glue logic (the term glue logic refers to the relatively small amounts of simple logic that are used to connect ("glue") and interface between larger logical blocks, functions or devices).

FPGAs are often used to prototype ASIC designs or to provide a hardware plat form on which to verify the physical implementation of new algorithms. However their low development cost and short time to market mean that they are increasingly finding their way into final products. Now, high performances FPGAs are containing millions of gates had become available.

The cost of an FPGA design is much lower than that of an ASIC. At the same time, implementing design changes is much easier in FPGASs, and the time to market for such design is much faster. FPGAs facilitate some operations, this means they allow individual engineers or small groups of engineers to realize their hardware and software concepts on an FPGA based test plat form without having to incur the enormous nonrecurring engineering (NRE) costs or purchase the expensive toolsets associated with ASIC Design.

Although ASICs offer the ultimate in size (number of transistors) complexity, and performance, designing and building one is an extremely time-consuming and expensive process with the added disadvantage that the final design is "frozen in silicon " and cannot be modified without creating a new version of the device. FPGA do the inverse, a million of logic gates can be used to implement extremely large and complex functions and can be changed without any problem .

FPGA offers more attractive features such as: high number of logic gates; short time to market, independency, an easiness of work, low cost, possibility of prototyping a lot of functions and feasibility. FPGA implementing design changes is much easier than an ASIC chip. Without returning to the manufacturer, FPGA facilitates a lot of operations without penalty of time cost and the dependence of the vendor. The dream of manufacturing a hardware VLSI specific purpose chip is realized with FPGA circuit. So "time-money-design" are now really with the dream of feasibility hardware implementation so that more ideas can be prototyped without limits.

References

[1] O.Hachour, "The Proposed Genetic FPGA Implementation For Path Planning of Autonomous Mobile Robot", *International Journal of Circuits , Systems and Signal Processing,* Issue 2, vol2 ,2008,pp151-167.

[2] O. Hachour AND N. Mastorakis, Avoiding obstacles using FPGA –a new solution and application ,*5th WSEAS international conference on automation & information (ICAI 2004) , WSEAS transaction on systems* , issue9 ,vol 3 , Venice , Italy , , ISSN 1109-2777, November 2004, pp2827-2834

[3] G.R.Goslin, "using Xilinx field programmable Gate Array's (FPGA's) for Application –Specific Digital Signal Processing Performance", Xilinx Coprporate Application Group,pp150-153

[4] S.K.Knapp, INC & the ASM group, "using Programmable Logic to accelerate DSP functions" ,Xilinx, 1995,pp1-8

[5] XAPP 057 july7, 1996(version1.0)

[6] R. Airiau, J.M Berger, V. Olive, *Circuit synthesis with VHD,* Kluwer Academic Publishers, 1994.

[7] S.D.Brown, R.J., J.Francis Rose, and Z.G.Vranesic : *Field-Programmable Gate Array*, *Kluwer Academic Publishers*, 1997.

[8] M.Cummings and S.Haruyama, FPGA in the soft radio, *IEEE Communication Magazine*, 0163-1999, pp.108-112.

[9] *GALILEO HDL Synthesis Manual,* Exemplar Logi*c,*1995.

[10] A.Gonzalez and R.Perez. : SLAVE : A genetic learning System Based on an Iterative Approach, *IEEE, Transaction on Fuzzy systems*, Vol 7, N.2, April 1999, pp.176-191.

[11] J.Legenhausen, R.Wade, C.Wilner, and B. Wilson,: *VHDL for programmable logic*, Addison- Wesley, 1996