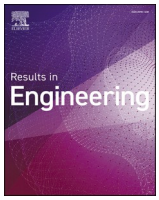
Results in Engineering 16 (2022) 100639

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering



Comparison study of hardware architectures performance between FPGA and DSP processors for implementing digital signal processing algorithms: Application of FIR digital filter

Omar Diouri [a,*], Ahmed Gaga [b], Hamid Ouanan [c], Saloua Senhaji [a], Sanaa Faquir [a], Mohammed Ouazzani Jamil [a]

[a] Laboratory of Intelligent Systems, Energy and Sustainable Development, Private University of Fez - UPF, Fez, Morocco
[b] Laboratory of Physics and Engineering Sciences, Research Team in Embedded Systems Engineering, Automation, Signal, Telecommunications and Smart Materials, Department of Physics, Polydisciplinary Faculty, Sultan Moulay Sliman University - USMS, Beni-Mellal, Morocco
[c] National School of Applied Sciences, Sultan Moulay Sliman University - USMS, Beni-Mellal, Morocco

ARTICLE INFO

Keywords:
Digital signal processors
FPGA
FIR filter
Digital signal processing
Hardware architecture

ABSTRACT

The objective of this project is to make a detailed study on the Hardware architectures of DSP and FPGA then to carry out a comparison between the two platforms in order to achieve a better implementation while keeping to increase the performances and obtain a better signal processing quality. For this raison, we wanted to make a digital FIR filter of 40 order bandpass and a symmetrical parallel architecture of this FIR filter. Matlab's FDA-TOOL is used to design this filter. This tool allows us to generate the VHDL code of the filter to implement it on Altera Cyclone III FPGA which is placed on the Texas Instruments TSW6011 board, this tool also allows us to generate all the coefficients of our filter in a header file C in order to use them in a program of the FIR filter written in C which will then be loaded on the DSP TMS320C6713. This achievement allows us to distinguish between different hardware architectures and make a comparison in terms of execution speed and power consumption.

1. Introduction

The real revolution came in the early 1990s with the emergence of multimedia applications, the Internet and mobile telephony. The complexity of these applications and the strong constraints in terms of performance related to these fields force designers of specialized circuits for digital signal processing to consider new hardware and software solutions. Thus we have seen the appearance in signal processing processors of modern architectural techniques such as VLIW (Very Long Instruction Word) instruction sets, superscalar and highly pipelined architectures, multiprocessor architectures, etc [1,2].

The constraints related to the embedded systems associated with the computing power required by the signal processing applications lead during the implementation [3], to the use of processors specialized in signal processing [4],. Their architecture is optimized to perform a large number of arithmetic operations on data tables and therefore all combinations of operations involved in signal processing applications [5,6]. The target applications in signal processing share a certain number of characteristics. The first of these is the real-time constraint: the system must process the data at the minimum speed required by the application, these speeds can be very high. The main mathematical processing to be performed are convolutions, matrix products or complex transformations (fast Fourier transform). These processing operations are particularly greedy in computing power, hence the need to have powerful operators, in order to be able to perform the processing operations in due time, and with the required precision. In addition, the systems must be able to manage very large data flows. Low power consumption and size are also important points, particularly for on-board applications. And of course. The low cost criterion remains a determining factor [7].

Regularly digital signal processing implementations can be used three of these components: one is using traditional processor which is a DSP chip; the second is using the Application Specific Integrated Circuits (ASIC); the third is using programmable logic devices (CPLD/FPGA). FPGA has a full parallel processing architecture, that is represents the advantages in real-time signal processing, portable code, etc [8]. The

* Corresponding author.
E-mail address: diouri@upf.ac.ma (O. Diouri).

https://doi.org/10.1016/j.rineng.2022.100639
Received 24 May 2022; Received in revised form 16 July 2022; Accepted 12 September 2022
Available online 16 September 2022

real-time implementation of digital signal processing requires the design of hardware capable of matching the sample rate of the application to the processing rate of the hardware. Sample rate information alone cannot be used to choose the architecture. The algorithms complexity is also an important consideration. The demands for high data throughput and increased algorithmic complexity in next-generation devices present a challenge in meeting power budgets [9–14].

Digital Signal Processors (DSP) are a specialized form of microprocessor with an architecture optimized for the fast operational needs of DSP applications. A DSP works well in signal processing applications because it is optimized to process signals efficiently, is relatively not expensive, and has a well-defined development path and fixed hardware configuration [15]. Field Programmable Gate Array (FPGA) is a very customizable chip used for logic functions. Hardware Description Language (HDL) is used to program the logical gates inside of the FPGA. This FPGA chip design is usually limited by the number of these gates on the chip [16]. As the number of gates available on the FPGA increases, complex designs are increasingly and can be placed on the chip.

Among the most complex algorithms, digital image processing plays a very important role in several fields such as medical, industry, security, etc [17–21]. For this reason, real-time image processing is required using robust systems based on FPGAs and DSP processors [19,22–26]. And among the algorithms that require a speed of calculation are the digital filters applied in the field of signal processing.

A filter is a frequency-selective system, which is used to modify an input signal and filter out unwanted signals to aid in processing. Basically there are two types of filters, analog and digital. Analog filters are themselves divided into several categories: passive filters which essentially use high quality inductors and capacitors. Until the 1970s, these were the only filters designed. They are currently used for high frequencies (use of quartz). Active filters consist of capacitors, resistors and active elements which are basically operational amplifiers (OA). They are less bulky, easy to design and less expensive than passive filters but remain limited in frequency and they consume more and require a power source. On the other hand, digital filters are widely used in different fields because these filters have the potential to achieve a better signal-to-noise ratio than analog filters [27–30]. The basic operation of the digital filter is to take a sequence of input numbers and calculate a different sequence of output numbers. There is a range of different digital filters. Finite Impulse Response filter (FIR) and Infinite Impulse Response filter (IIR) are the two common forms of filter. A disadvantage of IIR filters is that closed-form IIR designs are initially limited to low-pass, band-pass, and high-pass filters, etc. second, FIR filters can have an accurate linear phase [31]. Also, in the case of FIR filters, closed form design equations do not exist and the FIR filter design problem is much more mastered than the IIR design problem. A FIR is a filter structure that can be used to digitally implement almost any type of frequency response. It is usually implemented using a series of delays, multipliers and adders to create the filter output [32–34].

This paper concerns the field of signal processing. The first part defines the different targets with their architectures and their specific signal processing characteristics. In the second part, a digital filter, specialty FIR filter, will discussed in order to know the role, the objectif and the different features and parameters of this digital filter. Next part, the signal processing system which is simulated for the comparison must process the operation of a bandpass FIR filter in the signal processing domain. This filter is designed and simulated using specific platforms for FPGA and DSP. After that, FPGA development is done using MATLAB FDATOOL, with this tool, a complete description of the FIR filter is obtained where the values of the coefficients are converted into Very High Speed Integrated Circuit Hardware Description Language (VHDL) [35]. DSP processor development is accomplished by the Code Composer Studio which targets a TMS320C6713 from Texas Instruments. The last part details the different implementation solutions by specifying their performance in terms of speed and power consumption.

2. Description of the different hardware architectures of FPGAs and DSPs

2.1. FPGA

An FPGA is a reprogrammable silicon circuit. Using pre-built logic blocks and programmable routing resources, we can configure this circuit to implement custom hardware features [36–38]. In addition, FPGAs are completely reconfigurable and can instantly take on a new “personality”. However, the generalization of high-level design tools is changing the rules of FPGA programming, thanks to new technologies making it possible to convert graphical diagrams or even C code into digital hardware circuits [39,40].

The architecture of an FPGA is mainly described by the topology of routing resources and basic configurable logic elements. There are two classic architectures, the compute island architecture (originally used in Xilinx components) and the hierarchical architecture (originally used in Altera components). However, a trend appears with the latest generations of circuits, the architectures are mainly in the style of calculation islands. The following sections give some details about these architectures [41,42]. In the past, other architectures have been used. We can cite the logarithmic routing architecture used by Xilinx for its XC6000 circuit. Unfortunately, the routing placement tools were not adapted to this architecture to allow the designer to take full advantage of it.

2.1.1. Island-style FPGA routing architecture

The architecture most commonly used to make these circuits is of the island type. In this case the configurable resources are arranged in the form of a matrix, as can be seen in Fig. 1. Routing lines are arranged horizontally and vertically around the configurable resources. Connection blocks connect configurable resources to connection lines. Connection matrices connect the horizontal and vertical routing lines [43,44].

The use of configurable connection matrices is essential to ensure module connectivity, but configurable connection matrices degrade signal characteristics, reduce performance (operating frequency and power consumption) and require efficient placement-routing tools. In Fig. 1, we can see point-to-point links between two configurable elements in bold. These links use: the input/output ports of the configurable elements, the configurable connections which allow the connection of the configurable elements to the routing network, the

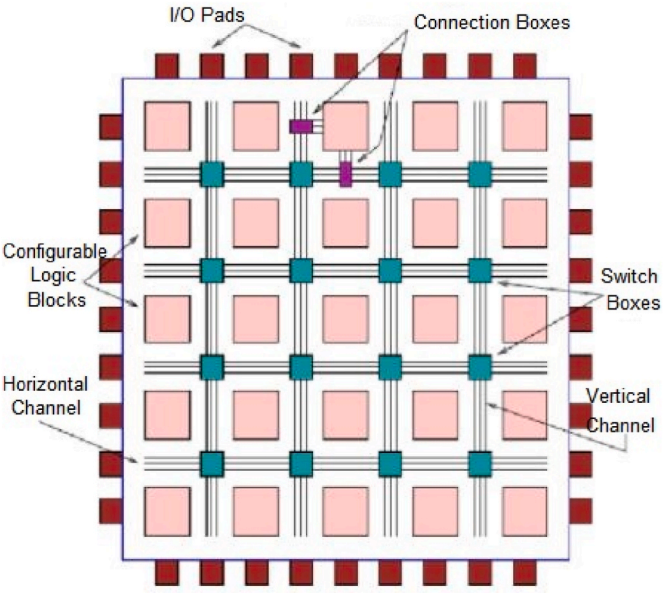


Fig. 1. Routing architecture of island-style of FPGA.

routing lines and the matrices of configurable connections. So many elements covered which degrade the performance of the circuit but which allow significant flexibility [45,46].

2.1.2. Hierarchical FPGA architecture

The hierarchical architecture commonly used for FPGA circuits consists of four or three levels. At each of these levels, routing resources are available to communicate between the circuit's own elements. Fig. 2 schematizes a four-level hierarchical architecture using an example of an architecture commonly encountered in ACTEL FPGAs. At the highest level of the hierarchy, the circuit is made up of matrix arranged tiles. Tiles are made up of logical clusters and memory banks. Finally, the logical clusters group together the configurable logical (and/or arithmetic) elements [47,48]. This style of architecture can be very energy efficient because it allows localization of intense communications and limits the use of long routing lines. However, it is necessary for placement and routing algorithms to take into account the specific characteristics of these architectures [49].

2.2. DSP

A DSP is a processor whose architecture is optimized to perform complex calculations in one clock stroke using the VLIW (Very Long Instruction Word) method, but also to easily access a large number of inputs-outputs (digital or analog). The main function used in the DSP is the MAC (Multiply and Accumulate) function, i.e. multiplication followed by addition and storage of the result (function widely used in servo and filtering) [50–52].

So as their name suggests, these circuits are primarily intended for the most efficient possible execution of signal processing algorithms. They differ in this sense from the microprocessors whose use is much more general due to the very diversity of applications. Their architecture is designed in such a way as to be able to efficiently manage intensive calculation loops and to ensure good numerical precision for the data. For this, they have the following characteristics:

- Specialized arithmetic operators for DSP calculations.
- high bandwidth for data linked to the use of independent buses for data and instructions (Harvard Architecture).
- address calculation units allowing the parallelization of calculations and access to data, and offering specific addressing modes for signal processing.
- Embedded hardware allowing rapid execution of software loops.

The DSP of the TMS320 family encompasses 16–32 bit, fixed or floating point processors. Their development began in 1982 with the TMS32010, a fixed-point DSP.

In order to fully understand how DSPs work, you have to see this processor as a set of interconnected blocks. Fig. 3 shows the detail of the internal architecture of the DSP based on the C6000 core (from which the C67 is derived). The CPU core has:

A program loading unit (program fetch), an instruction dispatch unit

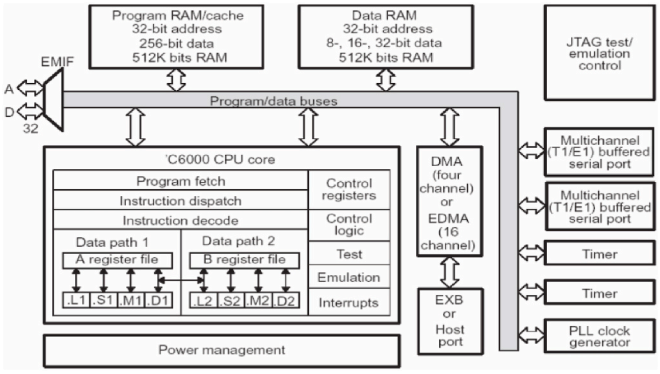


Fig. 3. Internal architecture of C6000 core.

(dispatch instruction), an instruction decoding unit, 32 32-bit registers separated into two times 16 registers (A and B), two channels for data (data path) each containing four processing units, two control registers and a control logic.

Units perform Logical, Shifting, Hardware Multiplication, and Data address operations. All data transfers between the register queue and the memory are done exclusively by the "Data-addressing" units [53].

3. FIR digital filter

In electronics, a digital filter is an element that performs filtering using a succession of mathematical operations on a discrete signal [54]. That is to say, it modifies the spectral content of the input signal by attenuating or eliminating certain unwanted spectral components. Unlike analog filters, which are made using an arrangement of physical components (resistor, capacitor, inductor, transistor, etc.), digital filters being generally made by processors, they are defined using Programming languages. Digital filters can, in theory, achieve all of the filtering effects that can be defined by mathematical functions or algorithms [55]. The two main limitations of digital filters are speed and cost. The filter speed is limited by the processor clock speed. As for the cost, it depends on the type of processor used. On the other hand, the price of integrated circuits continues to decrease, and digital filters are found everywhere in our environment, radio, cell phone, television, MP3 players, etc. A digital filter can be defined by a difference equation, i.e. the mathematical operation of the filter in the (discrete) time domain.

The general form of the filter of order M is the following:

$$y[n] = \sum_{k=0}^{N} b_k * x[n-k] - \sum_{k=1}^{M} a_k * y[n-k]$$

$$y[n] = b_0 x[n] + b_1 x[n-1] + \ldots + b_N x[n-N] - a_1 y[n-1] - a_2 y[n-2] - \ldots - a_M y[n-M]$$

In digital signal processing, the finite impulse response filter or FIR filter (Finite Impulse Response filter or FIR filter) is a digital filter which

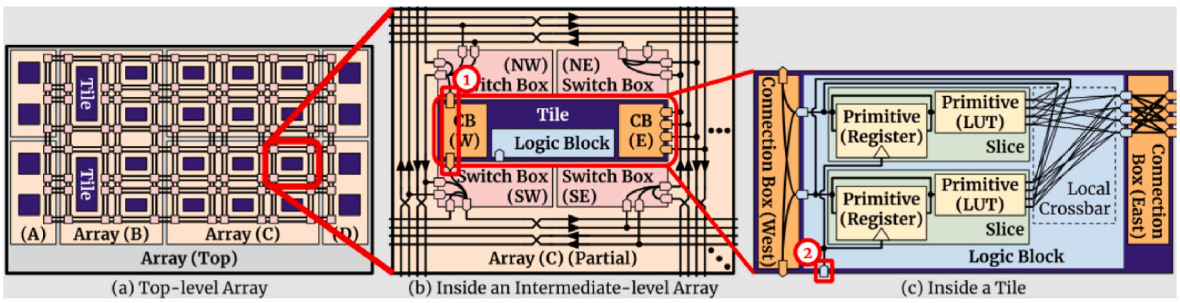


Fig. 2. Hierarchical architecture of FPGA.

is characterized by a response based solely on a finite number of values of the input signal. Therefore, whatever the filter, its impulse response will be stable and of finite duration depending on the number of filter coefficients.

Among linear filters, finite impulse response filters are opposed to infinite impulse response filters (RII filters) which can only be achieved with recursive implementations which replace a convolution over an infinite range, by a finite number d internal states that depend on the past history of the filter.

In general, the finite impulse response filter is described by the following linear combination:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \ldots + b_Nx[n-N]$$

where x[i] $1 \leq i \leq n$ represent the input signal values and y[i] $1 \leq i \leq n$ the output signal values.

Since the answer is a sum of a finite number of values, the FIR filter is naturally stable according to the Bounded In/Bounded Out criterion. The following general remarks can be made on FIR filters:

- FIR filters are necessarily stable, regardless of the coefficients used
- The complexity of an IIR filter (Infinite Impulse Response filter) is less than that of an FIR filter of the same order. This property can be useful on platforms with limited computing power
- Generally, FIR filters are less sensitive to quantization errors than IIR filters. The lack of recursion prevents cumulative errors.
- An FIR filter is less selective than an IIR filter of the same order. That is to say that the transition between the passband and the rejected band is slower than in the case of the IIR filter.
- Unlike an IIR, an FIR filter can have a symmetrical impulse response and introduce a signal delay but no phase shift.

Digital filters can be made using three basic elements or operations. Consider the gain element, the summation element and the unit delay. These elements are sufficient to realize all the possible linear digital filters. The realization shown in the following Fig. 4 is a direct type 1 realization of the FIR filter.

FIR filters never have jitter problems, because the output is just a finite sum of the input samples. Another advantage of FIRs is the constant group delay, which allows for minimal phase distortion on the processed signal. The impulse response in the RIF case is perfectly controllable.

4. Design and simulation of FIR filter on FPGA and DSP

To use FPGA in a digital filtering application, it must be programmed in VHDL or VeryLog, which implies a very complicated and difficult programming, as well as the realization period is very slow.

To use the DSP processor in this kind of digital filtering application, the same as FPGA, it must be programmed in C language, which also requires a long time to design this filtering system as well as a complexity in terms of the programming logic.

The Filter Designer Application (FDATOOL) on Matlab allows us to design and analyze digital filters. This technique is faster, more efficient and very easy to use while our goal is to test the same digital filter on two different platforms.

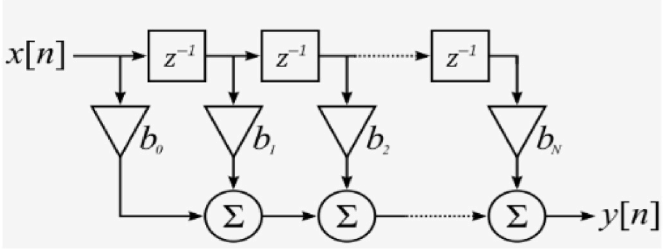


Fig. 4. Structure of FIR filter.

The objective of this application is to generate the VHDL file to be implemented on the FPGA target and to generate the C Header file with variables that contain filter parameter data to be also implemented on DSP target.

4.1. Design and simulation of FIR filter on FPGA

In order to test the good performance of our FIR filter, we did not use the ADC of the electronic card, on the other hand, we proposed as a solution the creation of digital data inside the FPGA. These data were taken from the test bench generated by Matlab's FDATOOL and were stored in the FPGA in order to use them as inputs for the filter, and to achieve this we added a VHDL memory on the FPGA to store these data and then read from memory, so the output of the memory is the input of the filter [28].

The first step is the realization of the FIR digital filter of band pass type using the FDATOOL tool of Matlab, then we chose the characteristics of the filter according to our needs.

In our case, we have chosen a bandpass FIR filter of 40 order with the first cutoff frequency which is equal to 7.6 KHz and the second is 14 KHz, as shown in the following Fig. 5:

After defining the filter and entering all the characteristics, clicking on "Set quantization parameters", allows us to configure the number of I/O bits and the type of implementation between Fixed-point or Floating-point and other parameters as shown in the following Fig. 6:

When all the parameters are well determined, we have generated the VHDL code specific to our filter which has already been configured. To do this, in the menu, select the 'Target' button then 'Generate HDL' to configure all the parameters associated with the filter concerning the architecture and the TestBench. After the generation of the VHDL code and the TestBench, in order to verify the filter code in VHDL and its operation, the following results were obtained on the ModelSim software as shown in Fig. 7.

We note that the input signal generated by the TestBench is a sinusoidal signal whose variable frequency as shown in the previous Fig. 7, and the output is a filtered signal which just passes the frequencies included in this band. In this simulation the first cutoff frequency is 8 KHz and the second cutoff frequency is 13 KHz.

4.2. Design and simulation of FIR filter on DSP

The filter coefficients are responsible for changing the type of the filter. From these coefficients we can configure the nature of the filter and the cutoff frequencies. Always with the tool FDATOOL, we can generate these coefficients after having configured all the parameters corresponding to our objective.

On the menu we click on the 'Target' button then 'generate the C Haeder', this file contains all the coefficients of our filter that we can use in the C program on the Code Composer Studio software. After writing the program in C on Code Composer Studio, we entered all the necessary libraries as shown in the following Fig. 8:

We simulated the program on the software and using a graph display tool on CSS we found the following results:

- In the time domain, we note that it represents the operation of a band pass filter as shown in the following Fig. 9.
- In the frequency domain (see Fig. 10), we obtained the same graph that we have already mentioned in the FDATOOL, and we notice that the first cutoff frequency is 10 KHz and the second is 12 KHz, we notice that the bandwidth is a slightly reduced compared to that which has already been configured in the FDATOOL Matlab, this is due to the output value of the filter recorded in a register, so there is a loss of information when storing this value in a number well determined bits.

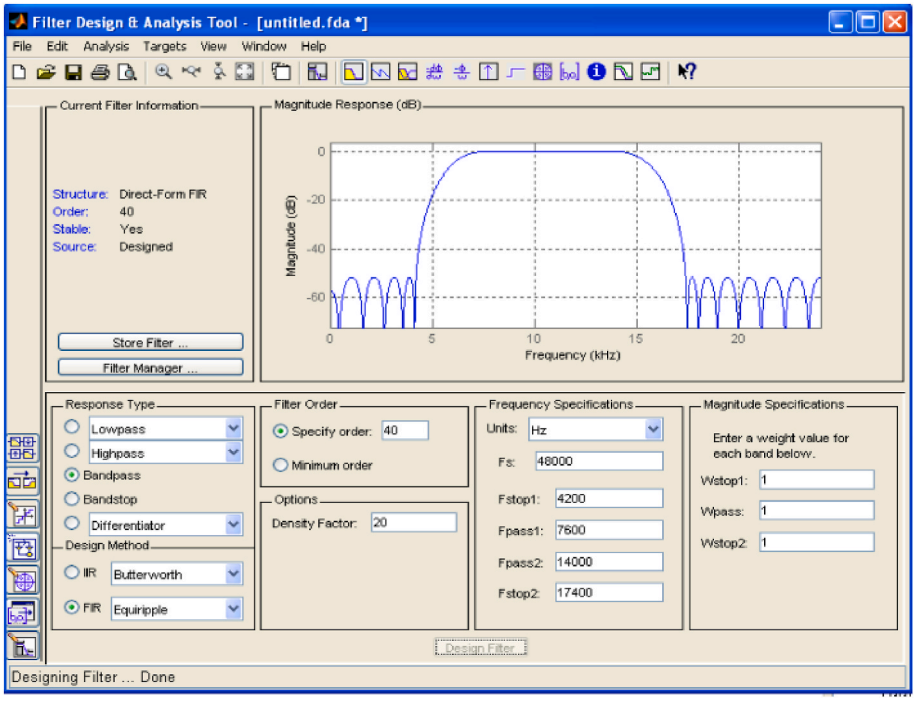


Fig. 5. Filter design on FDATOOL.

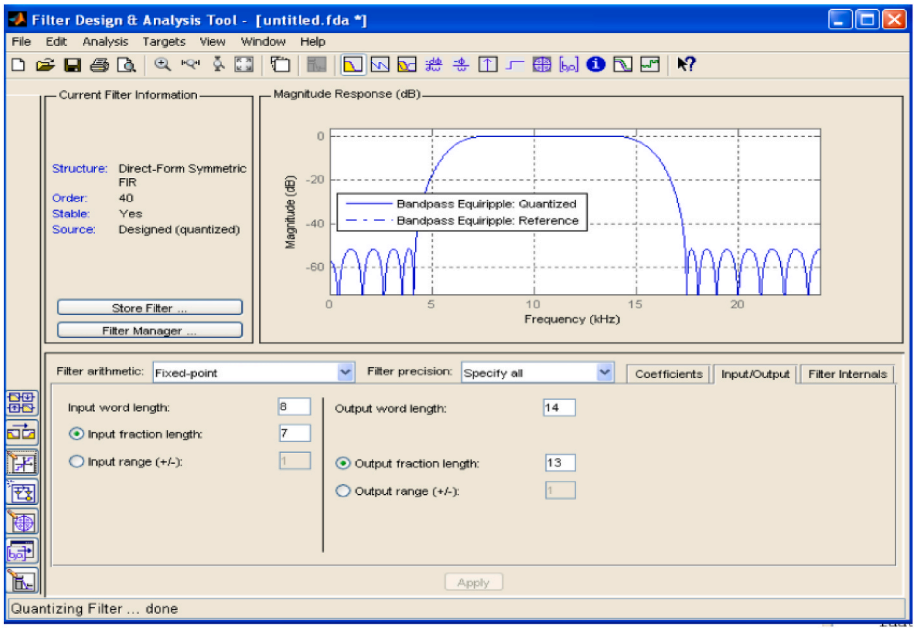


Fig. 6. Set quantization filter parameters.

5. Experimental results

The experimental phase is essential for the verification and validation of the previous part which was the simulation. This experimental part requires implementing the coefficients of our band-pass FIR digital filter, generated from MATLAB on the two hardware architectures FPGA and DSP using a frequency-variable input signal to have a signal at the output which will be filtered, in order to compare the performance of the two architectures in terms of speed, complexity of interior architecture and power consumption.

Two well-known companies, Xilinx and Altera, are currently vying for the top spot in the silicon technology market. These are the best FPGA manufacturers, not only for universities, but also for large companies developing digital technologies. Xilinx is basically the inventor of the FPGA, and is currently the biggest name in the FPGA world, while Altera is only second best. Altera tools have a more intuitive feel to the graphical or graphical user interface. With Xilinx, you don't need to worry about intermediate generated files. Altera can open a vital path in a chip viewer, and paths with all logical functions will be highlighted so we can even browse through all individual routing delays on segments.

For our study, we chose the reference Cyclone III for FPGA, because the board we have in our laboratory is Texas Instruments TSW6011EVM which has Cyclone III. This board primarily targets customers who need a high-performance FPGA with fast clock speeds and many I/O pins. We

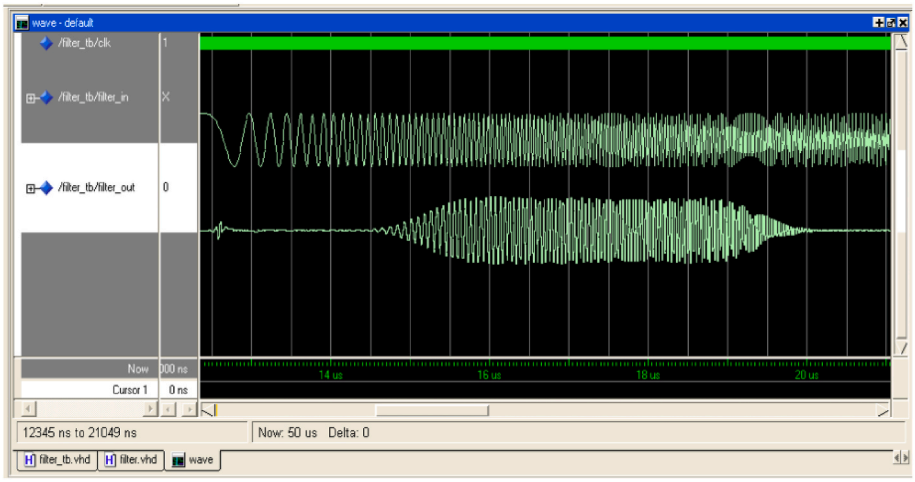


Fig. 7. Filter simulation on ModelSim.

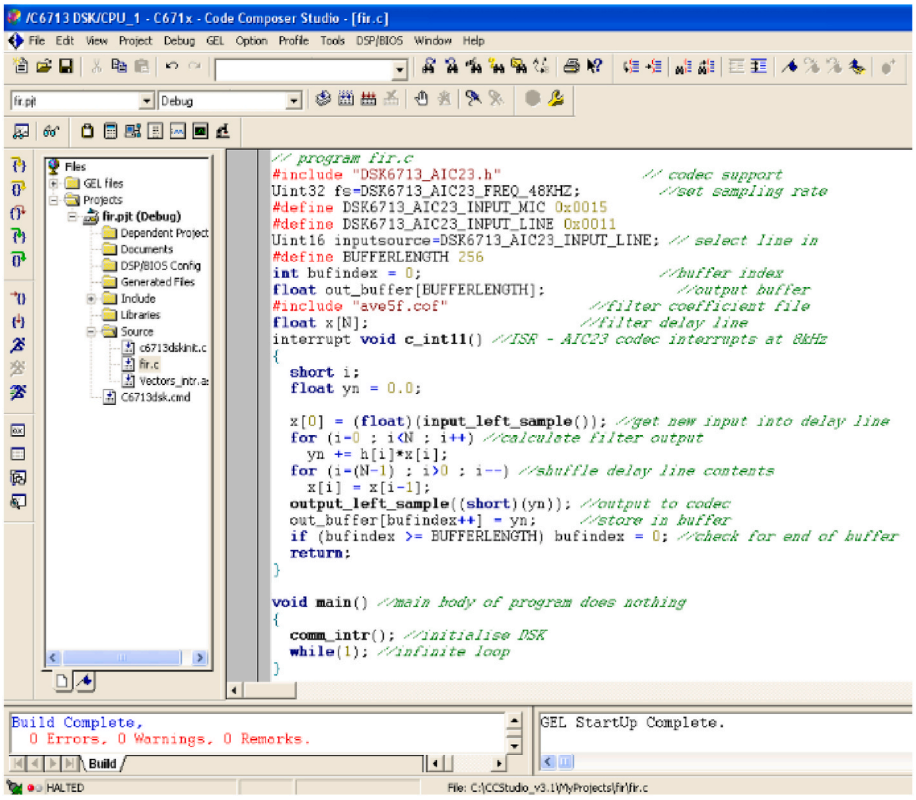


Fig. 8. Programming FIR filter on Code Composer Studio.

can use the board in both commercial and industrial design projects. The Cyclone III series design will help to meet the needs of experienced users.

There are two classifications of DSPs which are general DSP and special DSP. The general is optimized for applications like digital filtering, convolution and correlation. For the special DSP has features that are optimized for only and unique applications such as audio processing. The general DSP is focused in this study.

The major companies of the DSP industry are Texas Instruments (TI), Analog Devices, and Motorola. TI and Analog Devices offer fixed-point and floating-point DSP families, while Motorola offers only fixed-point DSP families. We are concentrated on Texas Instruments families, regarding to their hardware architectures performance and study real time implementation.

5.1. Implementation on FPGA

5.1.1. Introducing the Texas Instruments TSW6011EVM

Texas Instruments' TSW6011EVM evaluation board (Fig. 11) is the only RX channel (receive channel) board that can be used to demonstrate both the performance of a TRF3711 device and the IQ (I: In phase, Q: Phase quadrature) which is integrated in the FPGA. The TSW6011EVM board contains the following basic circuits: A TRF371125 RF signal receiver which also contains a low pass filter; an ADS5282 Analog-to-Digital data converter; an Altera Cyclone III FPGA for adaptive digital signal processing for IQ correction; a CDCE62005 clock

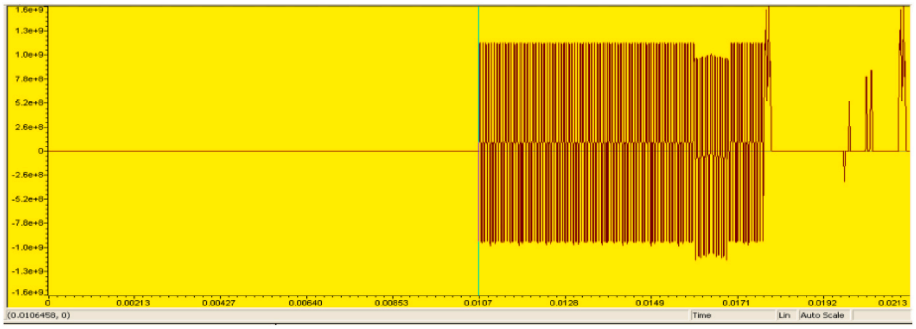


Fig. 9. Simulation in the time domain.

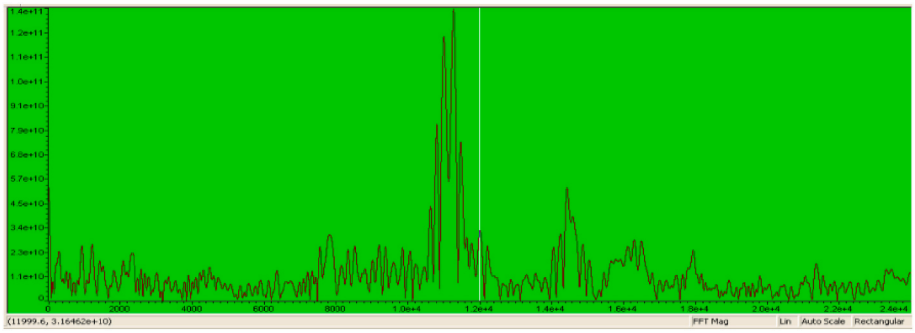


Fig. 10. Simulation in the frequency domain.

Fig. 11. TSW6011 board.

generator to generate system clocks and a DAC5672 Digital-Analog converter to facilitate performance measurement through a spectrum analyzer. The TSW6011EVM board provides options to send an input RF signal directly to the input of the TRF371125 or through one of the two LNA low noise amplifiers by moving two resistors. Additionally, there is an option to activate two ADCs (Analog to Digital Conversion) that one needs from an external source. This source can be single-ended or differential signal.

The reference of FPGA used in this board, which is the important element, is EP3C25 and its features are represented in Table 1.

The block diagram of the TSW6011 board is shown in Fig. 12. Output data can also be analyzed using two other outputs. An output from a CMOS connector that provides digital data, this interface has an RC network on all data and clock signals to allow the user to plug a logic analyzer CMOS connector directly into the connector.

The main components on TSW6011 are:

- TRF3711: directly integrated the quadrature demodulator.
- ADS5282: 12-bit with 8 ADC channels (up to 65 MSPS (Million Symbol Per Second)).
- CDCE62005: 5/10 output clock generator.
- DAC5672: 14 bit.

5.1.2. *Results of implementation of FIR filter on FPGA*

We used Quartus II software from Altera, which represents an analysis and synthesis software to create the. SOF file allowing the transfer of the hard to the FPGA circuit. So as we have already mentioned in the proposed solutions, we need memory to store the digital data and to achieve this we used a very powerful tool integrated into Quartus II, it is the use of the MegaCore tool which allows to generate the code in VHDL or in Verilog of the different blocks such as: RAM memory, image and video processing filters, block of the Fourier transform and other blocks and functions.

4000 digital values of the input signal that was generated by the TestBench, so we used a RAM memory of 4096 8-bit words, that is to say a 12-bit address bus ($2^{12} = 4096$). The input of the filter represents the data stored on the 8-bit RAM memory (the MegaCore tool of Quartus II only allows to generate 8-bit memories) for this we have chosen an 8-bit vector for the input. Although, the filtered output is coded on 14 bits because the input of the digital/analog converter DAC is coded on the

Table 1
FPGA features.

Familly	Device	Logic elements	Number of M9K Blocks	Total RAM Bits	18x18 Multipliers	PLLs	Global Clock Networks	Maximum User I/Os
Cyclone III	EP3C25	24624	66	608256	66	4	20	215

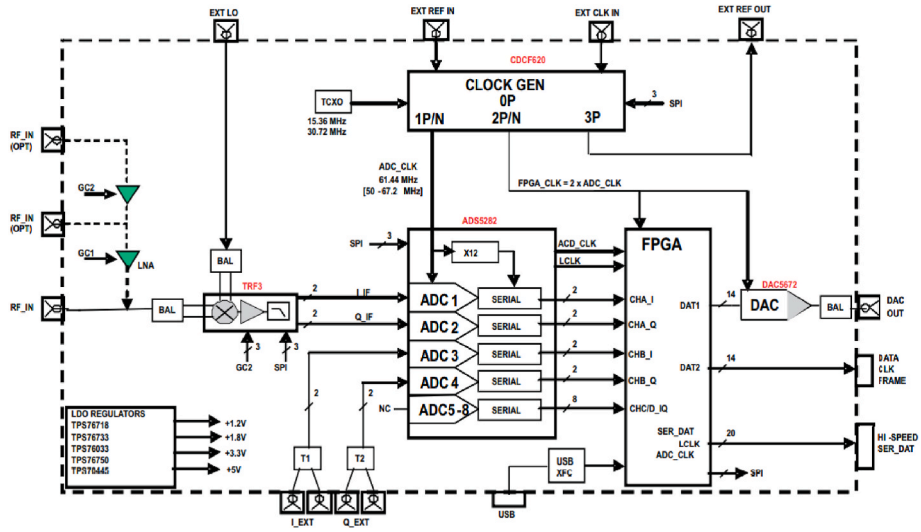


Fig. 12. Bloc diagrame of TSW6011 board.

same number of bits as shown in Fig. 13.

To test the proper functioning of the filter, we used the 'Signal Tap Logic Analyzer' tool of Quartus II (Fig. 14), which allows to visualize the states of the signals at the Pins of the FPGA in real time.

5.2. Implementation on DSP

5.2.1. Introducing the C6713DSK card

To carry out the project and design a program, the following tools are needed:

The DSK package (DSP Starter Kit) includes:

- Code Composer Studio (CCS), which provides the necessary supporting tools. CCS provides an integrated development environment (IDE), bringing together the C compiler, assembler, linker, debugger.
- The board shown in Fig. 15 contains the (C6713) TMS320C6713 floating-point digital signal processor, along with a 32-bit stereo codec at the input and output. Table 2 shows some features of this reference of DSP processor.
- A universal synchronous bus cable (USB) which connects the DSK board to a PC.
- A 5 V power supply for the DSK card.

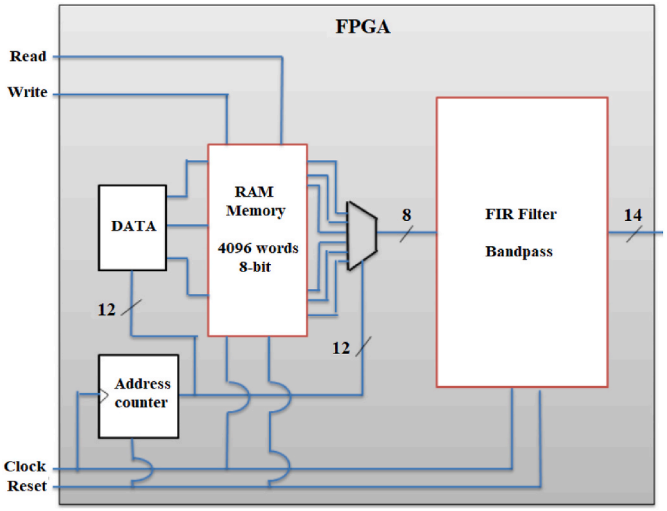


Fig. 13. Bloc diagram of FIR filter implementation on FPGA.

The DSK package is still relatively inexpensive and powerful with the hardware and software support tools needed for real-time signal processing. It is a complete DSP system. The DSK board is approximately 5*8 inches in size, and includes a C6713 floating point digital signal processor and a TLV320AIC23 (AIC23) 32-bit stereo codec.

The bloc diagram of C6713DSK is showed in Fig. 16, which the AIC23 codec uses sigma-delta technology which offers an ADC and DAC. It connects to a system whose sampling frequency is 12 MHz, and which can be easily adjusted and can vary from 8 to 96 kHz. The DSK card includes 16 MB of synchronous dynamic random access memory (DRAM) and 256 kB of flash memory. Four connectors on the card provide inputs and outputs: MIC IN, line in, line out and a headphone output (multiplexed with the line out). The status of the four user DIP switches on the DSK board can be read from a program and provides the user with a feedback control interface. This DSK card operates at 225 MHz and incorporates voltage regulators providing a voltage of 1.26 V for the C6713 core and 3.3 V for memory and peripherals.

5.2.2. Results of implementation of FIR filter on DSP

For the implementation on the C6713DSK card, the program was loaded on the DSP TMS320C6713 then the 'Run' operation was carried out to execute the program in real time. So in the end we obtained a band pass filter with a band varying from 8 KHz to 14 KHz, and we notice that these are the same frequencies that we have already mentioned on the FDATOOL. The oscilloscope gives the following result as shown in Fig. 17:

6. Comparison of the results

The performance required by an application is directly a function of the real-time constraint and the complexity of the algorithms implemented. The stronger real-time constraint, the less time the system has to execute the required algorithms. If the algorithms themselves are complex, i.e. the number of operations required to perform them is very high, then the computational load (the number of operations to be performed per unit time) required by the application is important and the system to achieve it must be powerful. As a general rule, the real-time constraint of a system is strongly linked to the data sampling frequency.

Power consumption has become the most important criterion for the embedded field. Autonomy has become a major selling point and manufacturers seek to minimize the consumption of systems in their products. As the prospects for increasing the capacity of batteries are limited, efforts are turning towards the design of circuits providing high

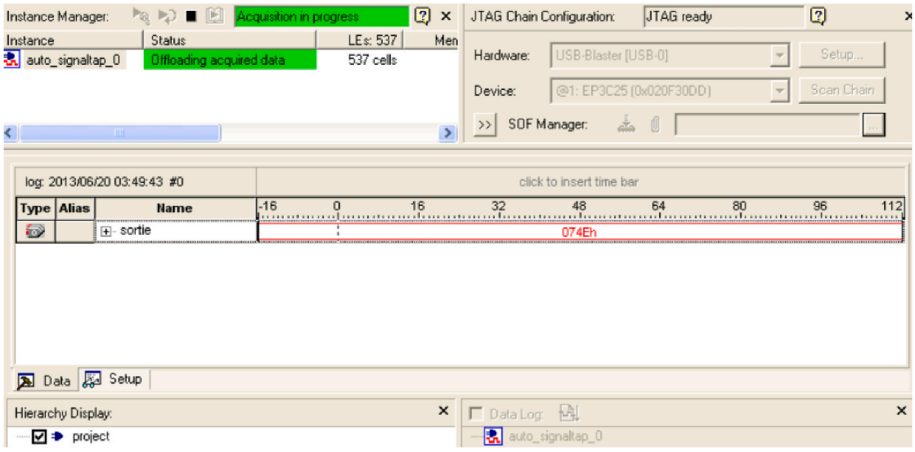


Fig. 14. Real-time visualization of the output signal on Signal Tap.

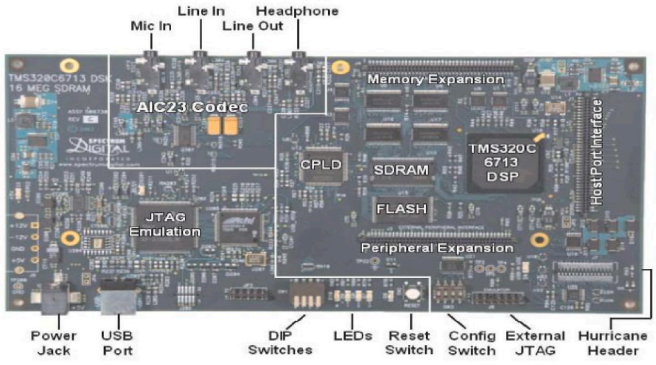


Fig. 15. C6713DSK board.

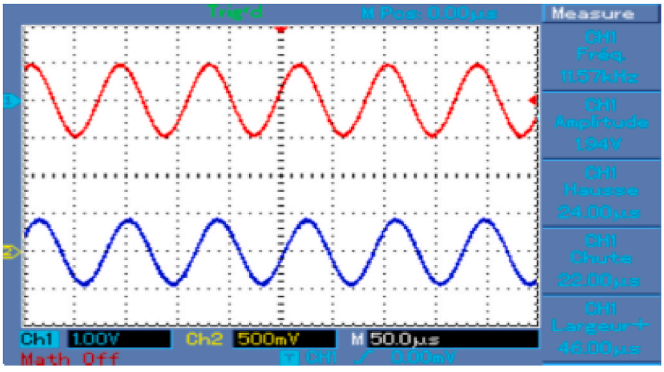


Fig. 17. Visualization of filter output on oscilloscope.

Table 2
DSP features.

Reference	Classification	On-Chip Memory	Frequency	Multipliers	User I/Os
C6713B	FLOATING-POINT	264 K	225 MHz	25	124

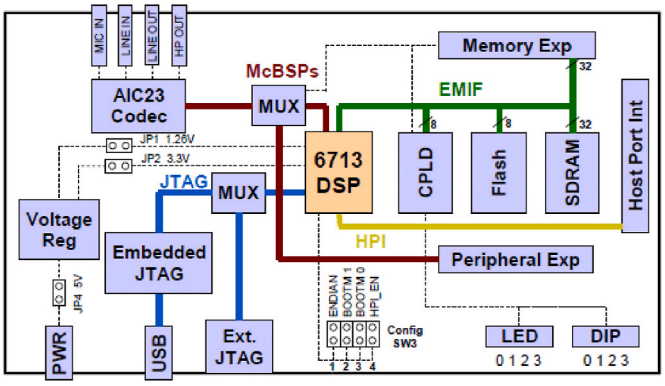


Fig. 16. Bloc diagram of C6713DSK board.

computing power for the lowest possible consumption.

6.1. *Interpretation of implementation results*

6.1.1. *On FPGA*

After having written the VHDL program of our filter on the Quartus II software, it allowed us to analyze and synthesize the filter by several parameters that come into play. The synthesis of the program gives us the following results (Fig. 18):

We note that we have consumed up to 15488 out of 24624 logic elements, therefore an occupation percentage of 63% in the Altera Cyclone III FPGA. For the registers, we have a consumption of 689 out of 24624, therefore a percentage of 3%, concerning the internal memory of the M9K RAM type of the FPGA, we have consumed 32768 bits out of

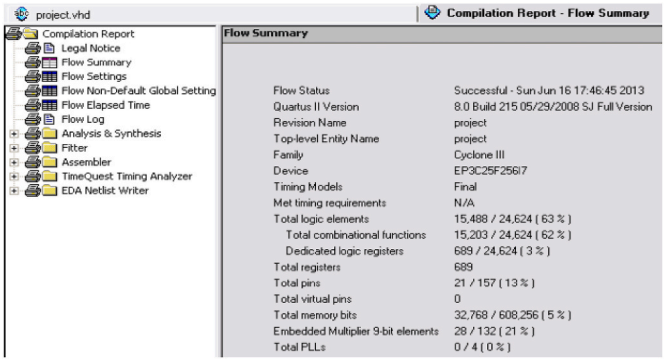


Fig. 18. Compilation report on Quartus II.

608256, i.e. the FPGA contains 66 RAM of type M9K therefore corresponds to 594 kbits = 74.25 kB. And for the 9-bit embedded multipliers, the filter consumes 28 multipliers among the 132 existing on the FPGA.

The operating frequency of the FPGA is 122 MHz so as an estimate of the speed we get:

- FPGA at 122 MHz with 28 multipliers:

122,000,000 * 28 = 3,416,000,000 operations per second.
And a maximum estimate of the execution speed is:

- FPGA at 122 MHz with 132 multipliers:

122,000,000 * 132 = 16,104,000,000 operations per second.

Quartus II also allowed us to estimate the power consumed by the FPGA using a compilation tool called 'PowerPlay', it is a powerful tool for analyzing the system and making a power estimate by giving the result with the Watt and also it allows to develop many electrical optimizations which are transparent for the designer, but offer an optimal use of the FPGA architecture to reduce the consumed power. In our case, the estimated power is 95.05 mW as shown in the following Fig. 19:

6.1.2. On DSP

With the Code Composer Studio software, profiling can be done, ie knowing the number of execution cycles of a function or a loop. To get there to know the cycle number of the 'for' loop which multiplies the data entering the DSP with the filter coefficients, we did the following two tests on the Code Composer Studio software:

To profile the filter code with optimization it is necessary to check that the compiler uses (–g –o3) and for the linker (-c –o) as shown in Fig. 20:

And after performing the profiling with optimization we get the following result as shown in Fig. 21. For a single access of the loop, a count of 63 clock cycles, i.e. this 'for' loop requires 63 clock cycles to execute inside the DSP.

The operating frequency of the DSP is 225 MHz, a maximum estimate of the execution speed is:

- DSP at 225 MHz with 8 units:

225,000,000 * 8 = 1,800,000,000 operations per second.

And according to the Datasheet of the DSP TMSC320C6713 we find that if we have 1800 MIPS (1800 Million Instructions Per Second) this result corresponds to 1350 MFLOPS (1350 Million Floating Point Operations Per Second).

From the Power Consumption Summary of the C6713 DSPs [56], the estimated power consumption is 200 mW.

6.2. Discussion

Table 3 presents a qualitative comparison of possible hardware solutions for the implementation of a digital signal processing system (FIR filter). Each type of solution offers advantages and disadvantages.

From the point of view of raw computing performance, the advantage goes of course to parallel hardware solutions of the FPGA type. We see that FPGA solutions are both the fastest but also those with the lowest surface * time cost, due to the excellent computational density of these solutions. By calculation density, we mean the number of achievable operations reduced per unit area. From this point of view, the weakness of programmable processors is due to the fact that they use large capacity memories to memorize the software code. These

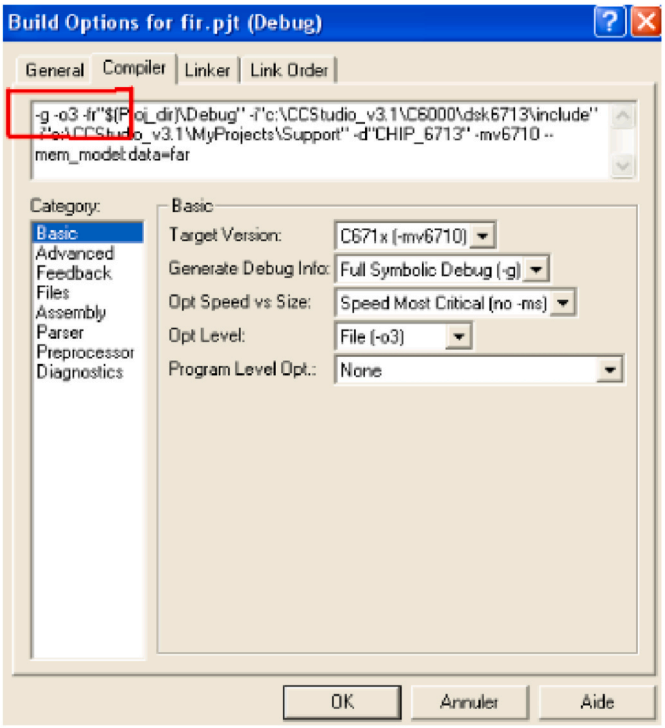


Fig. 20. Compiler configuration with optimization.

memories are expensive in terms of silicon area, often more expensive than the processor itself. Conversely, for FPGAs, reconfigurability is ensured by the configuration bits of the CLB (Configurable Logic Block) and interconnections, which form a single, very large instruction for the entire circuit. Using a single instruction to memorize hardware behavior makes FPGAs much more densely represent the state and description of a given computation.

The other advantage of these solutions is that they operate at bit granularity, unlike processors which use words (16, 32, 64 bits). A processor having to process information of 1 bit width will use word-width operators, reducing the efficiency in terms of calculation density. The better use of hardware in wired solutions is also reflected in power consumption. This parameter, which is difficult to estimate for a complete circuit, is however very strongly linked to the silicon surface.

Bearing that in mind, VLSI (Very Large Scale Integration) technology has focused on increasing the number of transistors as well as clock frequency, all at the cost of increased power consumption. The comparison in terms of power consumption is focused on static type which caused by the current due to the value of the input voltage [57]. In our case using FPGA board (Altera Cyclone III), the estimated power is 95.05 mW. For DSPs processors, it is very difficult to estimate the power consumption of this DSP, for this reason and from TMS320C6713B power consumption summary [56], the total power at 200 Mhz is 900 mW, assuming 60% CPU utilization.

FPGAs that are denser in terms of area and whose resource utilization rate is much higher than for processors, therefore have much better performance/consumption yields. The use of specialized programmable processors instead of purely wired solutions is however more and more frequent, particularly in the field of circuits dedicated to DSP applications. The growing complexity of applications is a first explanation.

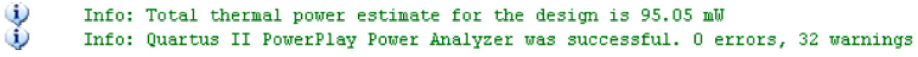


Fig. 19. Power consumption estimation on FPGA.

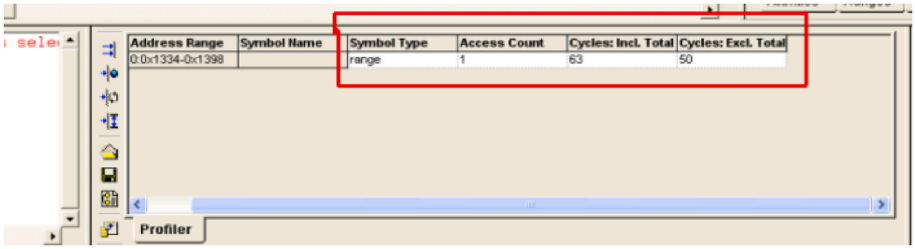


Fig. 21. Number of cycles execution of the 'for' loop with optimization.

Table 3
Comparison between FPGA and DSP.

Technology	Performance	Size	Consumption	Flexibility
FPGA	Very high	Big	Medium	Medium
DSP	high	medium	High	Very high

7. Conclusion

This study allowed us to implement in two differents development KITs from Texas Instruments, the TSW6011 EVM which is based on an Altera FPGA, and the C6713 DSK which integrates a digital signal processing processor. The use of MATLAB tools helped us to facilitate the programming of the filters by the generation of code in VHDL and also the coefficients in C. After the implementation of the program on the FPGA and the DSP, we made a comparative study between them in terms of complexity, execution speed and energy consumption in order to choose the platform to be used for data processing applications. Currently, the primary reason most engineers choose to use an FPGA over one of the digital signal processors is driven by application MIPS requirements and power consumption. Thus, the comparison between the digital signal processor and the FPGA emphasizes the MIPS comparison, and the consumption which are not the only advantages of an FPGA. This one is well suited for digital signal processing. In this analysis, this has been shown for the special case of filters. The devices have enough power to allow complex processing in parallel. Development tools make development time reasonable and simulation allows for good sizing and performance analysis. The different synthesis tools do not all have the same performance, because they do not perform the same type of optimizations. An FPGA combines the best of both hardware and software. Reconfigurable hardware like FPGAs offers high and good performance also can consequently be significantly faster than the general purpose processors with less power consumption.

Author contributions

Omar Diouri: Conceptualization, Methodology, Software, Validation, Writing – original draft; Ahmed Gaga: Methodology, Validation, Resources; Hamid Ouanan: Visualization, Resources; Saloua Senhaji: Formal analysis, Resources; Sanae Faquir: Visualization, Writing – review & editing; Mohammed Ouazzani Jamil: Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M. Saecker, V. Markl, Big data analytics on modern hardware architectures: a technology survey, in: M.-A. Aufaure, E. Zimányi (Eds.), Business Intelligence, vol. 138, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 125–149, https://doi.org/10.1007/978-3-642-36318-4_6.

[2] M. Wang, Y. Wang, D. Liu, Z. Qin, Z. Shao, Compiler-assisted leakage-aware loop scheduling for embedded VLIW DSP processors, J. Syst. Software 83 (5) (May 2010) 772–785, https://doi.org/10.1016/j.jss.2009.11.727.

[3] C.-H. Chang, A.S. Molahosseini, A.A.E. Zarandi, T.F. Tay, Residue number systems: a new paradigm to datapath optimization for low-power and high-performance digital signal processing applications, IEEE Circ. Syst. Mag. 15 (4) (2015) 26–44, https://doi.org/10.1109/MCAS.2015.2484118.

[4] Y. Jia, P. Xu, Noise cancellation in vibration signals using an oversampling and two-stage autocorrelation model, Results Eng 6 (Jun. 2020) 100136, https://doi.org/10.1016/j.rineng.2020.100136.

[5] V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, Low-power digital signal processing using approximate adders, IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 32 (1) (Jan. 2013) 124–137, https://doi.org/10.1109/TCAD.2012.2217962.

[6] A. Gaga, O. Diouri, M.O. Jamil, Design and realization of nano satellite cube for high precision atmosphere measurement, Results Eng (Apr. 2022) 100406, https://doi.org/10.1016/j.rineng.2022.100406.

[7] Y. Bajot, 'Panorama des processeurs de traitement du signal', LIP6, Research Report lip6.1998.028, Jun. 1998. May 19, 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02547769.

[8] W. Xin, H. Zhi-Qiang, J. Da-Qing, S. Zhi-Xiao, Z. Yu, L. Yong, Several implementation methods of signal processing algorithm based on FPGA, IOP Conf. Ser. Mater. Sci. Eng. 565 (1) (Jun. 2019), 012010, https://doi.org/10.1088/1757-899X/565/1/012010.

[9] E.M.E. Khattabi, O. Diouri, M. Mharzi, M.O. Jamil, Enhancing the energy performance of passive building through the Internet of things, in: Artificial Intelligence and Industrial Applications, Cham, 2021, pp. 279–286, https://doi.org/10.1007/978-3-030-53970-2_26.

[10] A. Gaga, O. Diouri, N. Es-sbai, F. Errahimi, Design and realization of an autonomous solar system, in: IOP Conf. Ser. Mater. Sci. Eng, vol. 186, Mar. 2017, p. 12031, https://doi.org/10.1088/1757-899X/186/1/012031.

[11] O. Diouri, N. Es-Sbai, F. Errahimi, A. Gaga, C. Alaoui, Control of single phase inverter using back-stepping in stand-alone mode, in: 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems, WITS), Apr. 2019, pp. 1–6, https://doi.org/10.1109/WITS.2019.8723761.

[12] O. Diouri, A. Gaga, N. Es-Sbai, F. Errahimi, Design and simulation of a novel cascaded transformer multilevel inverter topology for photovoltaic system, in: 2015 3rd International Renewable and Sustainable Energy Conference (IRSEC), Marrakech, Morocco, Dec. 2015, pp. 1–5, https://doi.org/10.1109/IRSEC.2015.7454957.

[13] B. Sirmacek, R. Vinuesa, Remote sensing and AI for building climate adaptation applications, Results Eng 15 (Sep. 2022) 100524, https://doi.org/10.1016/j.rineng.2022.100524.

[14] Y. Cheddadi, O. Diouri, A. Gaga, F. Errahimi, N. Es-Sbai, Design and simulation of an accurate neural network state-of-charge estimator for lithium ion battery pack, Int. Rev. Autom. Control IREACO 10 (2) (Mar. 2017) 186, https://doi.org/10.15866/ireaco.v10i2.11957.

[15] Q. Zhang, Q. Xie, K. Duan, B. Liang, M. Wang, G. Wang, A digital signal processor (DSP)-based system for embedded continuous-time cuffless blood pressure monitoring using single-channel PPG signal, Sci. China Inf. Sci. 63 (4) (Apr. 2020) 149402, https://doi.org/10.1007/s11432-018-9719-9.

[16] R. Raja Sudharsan, J. Deny, Field programmable gate Array (FPGA)-Based fast and low-pass finite impulse response (FIR) filter, in: Intelligent Computing and Innovation on Data Science, 2020, pp. 199–206, https://doi.org/10.1007/978-981-15-3284-9_21. Singapore.

[17] S.L. Kumar, H.B. Aravind, N. Hossiney, Digital image correlation (DIC) for measuring strain in brick masonry specimen using Ncorr open source 2D MATLAB program, Results Eng 4 (Dec. 2019) 100061, https://doi.org/10.1016/j.rineng.2019.100061.

[18] A. Ennouni, M.A. Sabri, S. Senhaji, A. Aarab, Robust approach for textured image clustering, in: 2016 4th IEEE International Colloquium on Information Science and Technology (CiSt), Tangier, Morocco, Oct. 2016, pp. 465–470, https://doi.org/10.1109/CIST.2016.7805093.

[19] S. Senhaji, A. Aarab, A new and robust image watermarking technique using contourlet-DCT domain and decomposition model, in: Int. Rev. Comput. Softw, vol. 8, IRECOS, Mar. 2013, https://doi.org/10.15866/irecos.v8i3.3164, 3, Art. no. 3.

[20] M. Momeny, A.M. Latif, M. Agha Sarram, R. Sheikhpour, Y.D. Zhang, A noise robust convolutional neural network for image classification, Results Eng 10 (Jun. 2021) 100225, https://doi.org/10.1016/j.rineng.2021.100225.

[21] A. Abdulqader, D.C. Rizos, Advantages of using digital image correlation techniques in uniaxial compression tests, Results Eng 6 (Jun. 2020) 100109, https://doi.org/10.1016/j.rineng.2020.100109.

[22] S. Senhaji, S. Abdelouahed, A. Aarab, A new and robust image watermarking technique based on the partial differential equations, Int. J. Commun. Antenna Propag. 1 (Aug. 2011) 330–334.

[23] M.V.G. Rao, P.R. Kumar, A.M. Prasad, Implementation of real time image processing system with FPGA and DSP, in: 2016 International Conference on Microelectronics, Computing and Communications (MicroCom), Durgapur, India, Jan, 2016, pp. 1–4, https://doi.org/10.1109/MicroCom.2016.7522496.

[24] A.H. Fredj, M.B. Abdallah, J. Malek, A.T. Azar, Fundus image denoising using FPGA hardware architecture, Int. J. Comput. Appl. Technol. 54 (1) (2016) 1, https://doi.org/10.1504/IJCAT.2016.077791.

[25] H. Ouanan, O. Diouri, A. Gaga, M. Ouanan, B. Aksasse, A novel face recognition system based on gabor and zernike features, in: Advanced Intelligent Systems for Sustainable Development, AI2SD'2019, Cham, 2020, pp. 9–15, https://doi.org/10.1007/978-3-030-36677-3_2.

[26] H. Ouanan, A. Gaga, O. Diouri, M. Ouanan, B. Aksasse, Development of Deep Learning-Based Facial Recognition System, 2020, pp. 45–52, https://doi.org/10.1007/978-3-030-36677-3_6.

[27] A.A. PhD, Digital Filters: Analysis, Design, and Signal Processing Applications, McGraw-Hill Education, 2018. May 19, 2022. [Online]. Available: https://www.accessengineeringlibrary.com/content/book/9780071846035.

[28] L.B. Jackson, Digital Filters and Signal Processing: with MATLAB® Exercises, Springer Science & Business Media, 2013.

[29] D. Datta, H.S. Dutta, Efficient FPGA implementation of FIR filter using distributed arithmetic, in: Energy Systems, Drives and Automations, Singapore, 2020, pp. 151–160, https://doi.org/10.1007/978-981-15-5089-8_14.

[30] C. Diaz, G. Sanchez, J.-G. Avalos, G. Sanchez, J.-C. Sanchez, H. Perez, Spike-based compact digital neuromorphic architecture for efficient implementation of high order FIR filters, Neurocomputing 251 (Aug. 2017) 90–98, https://doi.org/10.1016/j.neucom.2017.04.012.

[31] C.-L. Hu, Design and verification of FIR filter based on Matlab and DSP, in: 2012 International Conference on Image Analysis and Signal Processing, Huangzhou, Zhejiang, China, Nov, 2012, pp. 1–4, https://doi.org/10.1109/IASP.2012.6425042.

[32] R. Thakur, K. Khare, High speed FPGA implementation of FIR filter for DSP applications, Int. J. Model. Optim. (Jan. 2013) 92–94, https://doi.org/10.7763/IJMO.2013.V3.242.

[33] E. Al-Rawachy, R.P. Giddings, J. Tang, Experimental demonstration of a real-time digital filter multiple access PON with low complexity DSP-based interference cancellation, J. Lightwave Technol. 37 (17) (Sep. 2019) 4315–4329, https://doi.org/10.1109/JLT.2019.2923546.

[34] O. Diouri, F. Errahimi, N. Es-Sbai, Regulation of the output voltage of an inverter in case of load variation, in: IOP Conf. Ser. Mater. Sci. Eng, vol. 353, May 2018, 012021, https://doi.org/10.1088/1757-899X/353/1/012021.

[35] P. Marwedel, G. Goossens, Code Generation for Embedded Processors, Springer Science & Business Media, 2013.

[36] M. Bakiri, C. Guyeux, J.-F. Couchot, A.K. Oudjida, Survey on hardware implementation of random number generators on FPGA: theory and experimental analyses, Comput. Sci. Rev. 27 (Feb. 2018) 135–153, https://doi.org/10.1016/j.cosrev.2018.01.002.

[37] A. Pamuk, An FPGA implementation architecture for decoding of polar codes, in: 2011 8th International Symposium on Wireless Communication Systems, Aachen, Germany, Nov, 2011, pp. 437–441, https://doi.org/10.1109/ISWCS.2011.6125398.

[38] A.X.M. Chang, E. Culurciello, Hardware accelerators for recurrent neural networks on FPGA, in: 2017 IEEE International Symposium on Circuits and Systems, ISCAS), Baltimore, MD, USA, May 2017, pp. 1–4, https://doi.org/10.1109/ISCAS.2017.8050816.

[39] A. Podobas, S. Matsuoka, Hardware implementation of POSITs and their application in FPGAs, in: 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, May 2018, pp. 138–145, https://doi.org/10.1109/IPDPSW.2018.00029.

[40] M. Dagbagi, L. Idkhajine, E. Monmasson, I. Slama-Belkhodja, FPGA implementation of Power Electronic Converter real-time model, in: International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion, Sorrento, Italy, Jun. 2012, pp. 658–663, https://doi.org/10.1109/SPEEDAM.2012.6264543.

[41] M. Alçın, İ. Pehlivan, İ. Koyuncu, Hardware design and implementation of a novel ANN-based chaotic generator in FPGA, Optik 127 (13) (Jul. 2016) 5500–5505, https://doi.org/10.1016/j.ijleo.2016.03.042.

[42] M.I. AlAli, K.M. Mhaidat, I.A. Aljarrah, Implementing image processing algorithms in FPGA hardware, in: 2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Amman, Jordan, Dec. 2013, pp. 1–5, https://doi.org/10.1109/AEECT.2013.6716446.

[43] M. Qasaimeh, A. Sagahyroon, T. Shanableh, FPGA-based parallel hardware architecture for real-time image classification, IEEE Trans. Comput. Imaging 1 (1) (Mar. 2015) 56–70, https://doi.org/10.1109/TCI.2015.2424077.

[44] B. Koziel, R. Azarderakhsh, M. Mozaffari-Kermani, Fast hardware architectures for supersingular isogeny diffie-hellman key exchange on FPGA, in: O. Dunkelman, S. K. Sanadhya (Eds.), Progress in Cryptology – INDOCRYPT 2016, vol. 10095, Springer International Publishing, Cham, 2016, pp. 191–206, https://doi.org/10.1007/978-3-319-49890-4_11.

[45] C. Wang, X. Li, Y. Chen, Y. Zhang, O. Diessel, X. Zhou, Service-oriented architecture on FPGA-based MPSoC, IEEE Trans. Parallel Distr. Syst. 28 (10) (Oct. 2017) 2993–3006, https://doi.org/10.1109/TPDS.2017.2701828.

[46] S. Mittal, A survey of FPGA-based accelerators for convolutional neural networks, Neural Comput. Appl. 32 (4) (Feb. 2020) 1109–1139, https://doi.org/10.1007/s00521-018-3761-1.

[47] U. Farooq, Z. Marrakchi, H. Mehrez, FPGA architectures: an overview, in: U. Farooq, Z. Marrakchi, H. Mehrez (Eds.), Tree-based Heterogeneous FPGA Architectures: Application Specific Exploration and Optimization, Springer, New York, NY, 2012, pp. 7–48, https://doi.org/10.1007/978-1-4614-3594-5_2.

[48] E. Aridhi, D. Popescu, A. Mami, FPGA based co-design of a speed fuzzy logic controller applied to an autonomous car, Int. J. Reconfigurable Embed. Syst. IJRES 10 (3) (2021), https://doi.org/10.11591/ijres.v10.i3.pp195-211. Art. no. 3, Nov.

[49] N. Iyer, P.V. Anandmohan, D.V. Poornaiah, V.D. Kulkarni, Efficient Hardware Architectures for AES on FPGA, Computational Intelligence and Information Technology, Berlin, Heidelberg, 2011, pp. 249–257, https://doi.org/10.1007/978-3-642-25734-6_37.

[50] A. Sen, P. Mitra, D. Datta, Low Power MAC Unit for DSP Processor, vol. 1, 2013, p. 3, 6.

[51] L.-H. Khoi-Nguyen, D.-D. Anh-Vu, V. Thanh, D.-D. Quoc-Minh, L. Vy, B. Trong-Tu, A proposed RISC instruction set architecture for the MAC unit of 32-bit VLIW DSP processor core, in: 2014 International Conference on Computing, Management and Telecommunications (ComManTel), Da Nang, Vietnam, Apr. 2014, pp. 170–175, https://doi.org/10.1109/ComManTel.2014.6825599.

[52] H.Y. Cheah, S.A. Fahmy, D.L. Maskell, iDEA: a DSP block based FPGA soft processor, in: 2012 International Conference on Field-Programmable Technology, Seoul, Korea (South), Dec. 2012, pp. 151–158, https://doi.org/10.1109/FPT.2012.6412128.

[53] B. Harish, M.S.S. Rukmini, K. Sivani, Design of MAC unit for digital filters in signal processing and communication, Int. J. Speech Technol. (Mar. 2021), https://doi.org/10.1007/s10772-021-09824-0.

[54] S. Bhattacharjee, S. Sil, A. Chakrabarti, Evaluation of power efficient FIR filter for FPGA based DSP applications, Procedia Technol 10 (2013) 856–865, https://doi.org/10.1016/j.protcy.2013.12.431.

[55] J. Zhao, Modeling and Simulation of Digital Filter, Dec. 2015, pp. 1333–1338, https://doi.org/10.2991/nceece-15.2016.235.

[56] G. Ivan, 'TMS320C6711D, C6712D, C6713B Power Consumption Summary', Texas Instruments, Application Report [Online], Aug. 2005. Available: https://www.ti.com/lit/an/spra889a/spra889a.pdf?ts=1651686506016.

[57] A.D. Garcia Garcia, L.F. Gonzalez Perez, R.F. Acuna, Power consumption management on FPGAs, in: 15th International Conference on Electronics, Communications and Computers (CONIELECOMP'05), Puebla, Mexico, 2005, pp. 240–245, https://doi.org/10.1109/CONIEL.2005.60.