FPGA Implementation of Daubeshies Polyphase-Decimator filter

Abdelhakim SAHOUR
Dept. sciences and technology,
University of Khenchela
Route de Constantine BP:1252, El Houria, 40004
Khenchela, Algeria

Mohamed Benouaret
Dept. of Electronic, University of Annaba
Université Badji Mokhtar -Annaba- B.P.12, Annaba,
23000 Algeria.

ABSTRACT
This paper presents a fast multi-rate structure of Daubechies polyphase decimator which is required in the development of telecommunications systems and real time processing. It is an optimized approach which offers an increased efficiency in both size and speed, aspects that are well suited to reconfigurable architecture task heretofore implementation in FPGA platform which offers the potential of designing high performance systems at low cost. Hence, in order to evaluate the features of this method and to check the proposed extension of the basic Daubechies wavelet with four coefficients, a computer simulation was performed. It always simple and quick testing of the algorithm behavior of the proposed method for a wide class of signal processing. The Matlab/Simulink package and Modelsim were chosen as the programming environments for computer simulations.

Keywords
Daubechies wavelet, Filter Decimator, Multirate system, Altera FPGA. Xilinx FPGA, Modelsim, Matlab/simulink.

1. INTRODUCTION
Digital signal processing algorithms are increasingly employed in modern wireless communications and multimedia consumer electronics, such as cellular telephone and digital cameras. Traditionally, such algorithms are implemented using programmable DSP chips for low-rate applications [1], or VLSI application specific integrated circuits (ASICs) for higher rates [2]. However, advancements in Filed Programmable Gate Arrays (FPGAs) provide a new vital option for the efficient implementation of DSP algorithms [3]. Thus, the need to grow fast filtering methods. Indeed, making it imperative that we often think to adjust the sampling rate according to the signal of interest. Systems with different sampling rates are referred to as multirate systems [4]. Polyphase is a methode of doing sampling rate conversion that leads to very efficient implementation. But more that, it leads to very general viewpoints that are useful in building decimator or interpolator filter. So, in this paper we focus our attention upon the polyphase structure of the Daubechies filter which has gained the reputation of being a very effective signal analysis tool for many practical applications. In fact, after A/D conversion, the signal data can be found in small frequency band (typically, lowpass or bandpass), then it is reasonable and logic to perform a filtering operation followed by an appropriate sample rate reduction. A narrow filter followed by a downsampler is frequently referred to as a decimator [4]. The filtering, downsampling, and the effect on the spectrum is depicted in Figure 1. We can reduce the sampling rate up to the limit called the "Nyquist rate" which says that the sampling rate must be higher than the bandwidth of the signal, in order to avoid aliasing. So, for a bandpass signal, the frequency band of interest must fall within an integer band. If fs is the sampling rate, and M is the desired downsampling factor, then the band of interest must fall between.

$$k\frac{f_s}{2M} < f < (k+1)\frac{f_s}{2M} \qquad k \in N$$

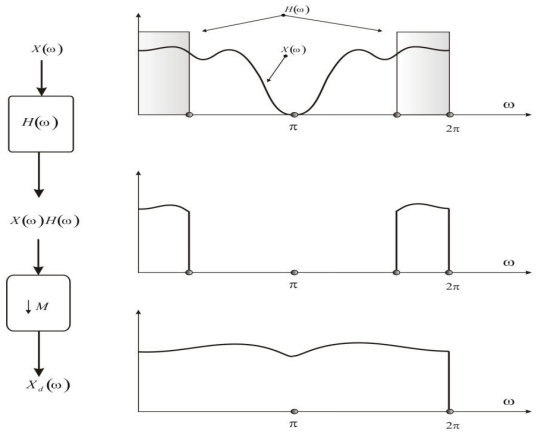

Figure 1. Decimation effect of the signal x(n) → X(ω)

2. NOBLE IDENDITY
When manipulating signal flow graphs of multirate systems it is sometimes very suited and useful to rearrange the filter and decimator/interpolator, as illustrated in figure 2. This is the so-called "Noble" relation. For the downsampler, it follows:

$$(\downarrow M)\,H(z) = H(z^M)(\downarrow M) \qquad (1)$$

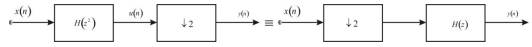

Figure 2. Equivalent multirate systems for M=2 (Noble relation)

This means that the decimator is done first, we can reduce the filter length H(zM) by factor of M.

u(n) can be expressed as: $u(n) = H(z^2)X(z)$ and the downsampling expression of the output signal can be written as :

$$y(z)=\frac{1}{2}\left[U\left(z^{1/2}\right)+U\left(-z^{1/2}\right)\right]$$

which involves that the obtained signal can be downsampled first:

$$y(z)=\frac{1}{2}H(z)\left[X\left(z^{1/2}\right)+X\left(-z^{1/2}\right)\right] \quad (2)$$

It follows that the polyphase form of the filter representation using first Nobel identity looks as depicted in Figure 3.

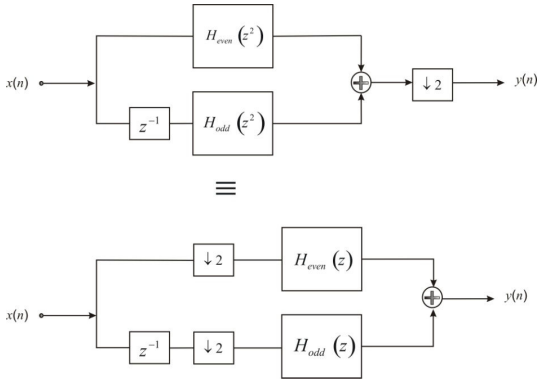

Figure 3: The first Noble identity representation of the polyphase structure with M=2.

3. POPLYPHASE DECONPOSITION

Polyphase decomposition is very useful when we need to implement architecture of decimator and or interpolator filter design. To illustrate this, consider the polyphase decomposition of an FIR decimation filter. If we add downsampling by a factor of M to the filter structure, we find that we only need to compute the output sequence y(n) at time instances

$$y(nM)=y(\downarrow M)$$

It follows that we do not need to compute the ordinary product

$$y(n)=\sum_{i=0}^{N-1}h(i)x(n-i)$$

convolution (3)

But it is therefore reasonable to split the input signal first into M separate sequences according to:

$$x_i(n)=\{x(i),x(M+i),x(2M+i),\ \ldots\} \qquad i=[0\ldots M-1] \quad (4)$$

And also to split the filter h(n) into M sequences :

$$h_i(n)=\{h(i),h(M+i),h(2M+i),\ \ldots\} \qquad i=[0\ldots M-1] \quad (5)$$

It follows from these that the filtered and decimate sequence was denoted by:

$$y(\downarrow M)=x_0(n)*h_0(n)+\sum_{i=1}^{M-1}x_i(n-1)*h_{M-i}(n) \quad (6)$$

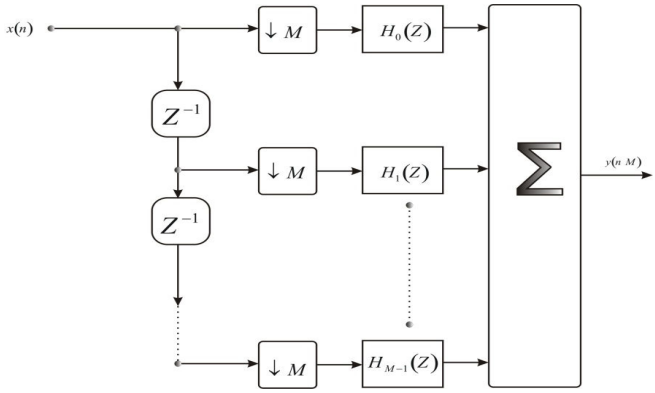

Figure 4: Polyphase realization of a decimator filter

Figure 4 shows a decimator filter topology using polyphase decomposition. Such a decimator is M times faster than the usual FIR filter followed by a downsampler. The filters hi(n) are called polyphase filters, because they all have the same magnitude transfer function, but they are separated by a sample delay, which introduces a phase offset.

4. DAUBECHIES DECIMATOR FILTER

Consider a Daubechies length-4 filter as shown in figure 3 with H(z) and M=2

$$H(Z)=\left[\left(1+\sqrt{3}\right)+\left(3+\sqrt{3}\right)Z^{-1}+\left(3-\sqrt{3}\right)Z^{-2}+\left(1-\sqrt{3}\right)Z^{-3}\right]\frac{1}{4\sqrt{2}} \quad (7)$$

$$H(Z)=0.48301+0.8365Z^{-1}+0.2241Z^{-2}-0.1294Z^{-3}$$

So, the mathematical model describing the Daubechies decimator filter can be represented by the following evolution matrix under (assuming of M = 2).

$$y(\downarrow 2)=\begin{bmatrix}x(0) & 0\\ x(2) & x(0)\\ x(4) & x(2)\\ x(6) & x(4)\\ x(8) & x(6)\\ \vdots & \vdots\end{bmatrix}\begin{bmatrix}0.48301\\ \\ \\ \\ 0.2241\end{bmatrix}+\begin{bmatrix}0 & 0\\ x(1) & 0\\ x(3) & x(1)\\ x(7) & x(3)\\ x(9) & x(7)\\ \vdots & \vdots\end{bmatrix}\begin{bmatrix}0.8365\\ \\ \\ \\ -0.1294\end{bmatrix} \quad (8)$$

($x_0(n)$ over the first column, $x_1(n-1)$ over the second column, Heven over the first coefficient vector, Hodd over the second coefficient vector)

Quantizing the filter at a precision of 8 bits results in the following model:

$$H(Z) = \left(124 + 214Z^{-1} + 57Z^{-2} - 33Z^{-3}\right) \Big/ 256$$

$$H(Z) = H_0\left(Z^2\right) + H_1\left(Z^2\right) \quad (9)$$

$$= \underbrace{\left(\frac{124}{256} + \frac{57}{256}Z^{-2}\right)}_{H_0\left(Z^2\right)} + Z^{-1}\underbrace{\left(\frac{214}{256} - \frac{33}{256}Z^{-2}\right)}_{H_1\left(Z^2\right)}$$

and it follows that

$$H_{even}(Z) = \left(\frac{124}{256} + \frac{57}{256}Z^{-1}\right) \qquad H_{odd}(Z) = Z^{-1}\left(\frac{214}{256} - \frac{33}{256}Z^{-1}\right) \quad (10)$$

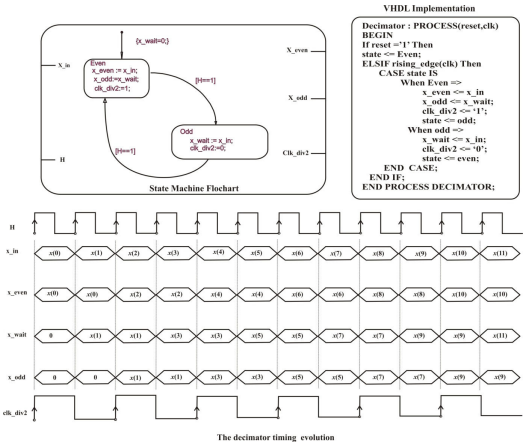

Figure 5: Decimator timing diagram.

Our goal is to be more pronounced in all stages of the implementation task of such filter decimator, so we strive properly to dissect each part of this design. By carefully designing the coefficients, the structure allows to obtain a very high performance and relatively easily implementable of the filter decimator. Obviously the first process is modeled by the FSM chart, which includes the control flow and the splitting of the input stream at the sampling rate into even and odd samples. The second task includes the Reduced Adder Graph (RAG) multiplier and hosts the two filters in a transposed structure.

The multiplication operation by 124 gain required effectively means data shifting with five bits towards the Most Significant Bit (MSB) followed by subtracting operation between the obtained data and the input stream (x_even) as in Figure 6. In order to take care of the negative numbers in two's complement arithmetic, the Least Significant Bit (LSB) has been propagated to the next five bits. At last the output data has been shifted by four bits. The output is given in 17 bits without any loss of precision. Actually 17 bits is enough to provide the full accuracy. However, 17 bits sizing have been chosen for the consistency with the other multiplier by the factor of 0 … 255.

The multiplication by the 57 factor requires a subtraction operation between the eight shifted version of the input and the original version of it. The second step is the addition of the data obtained previously and that of the original one as depicted in Figure 7. For such a case 17-bits are required to provide the output at the full accuracy.

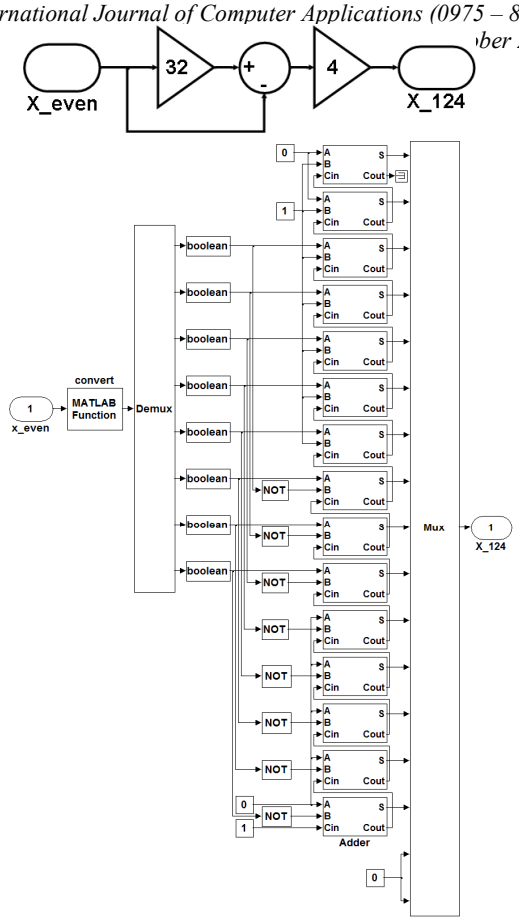

Figure 6: The fixed-point 17 bits 124 gain structure

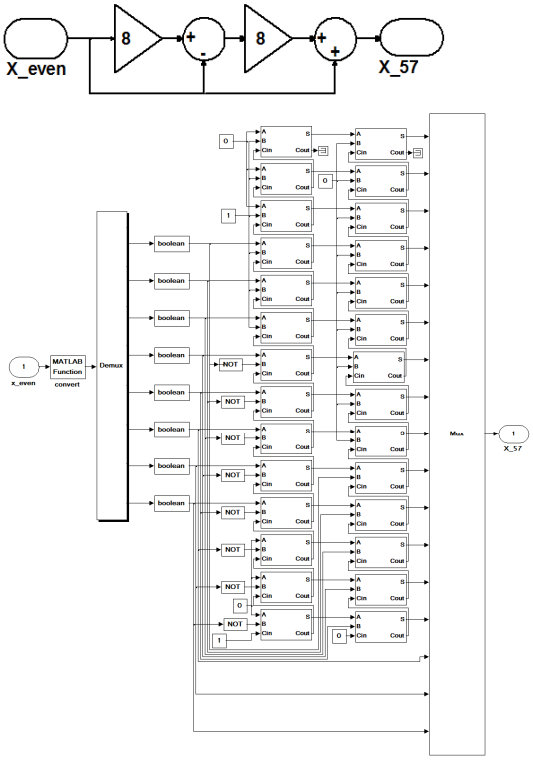

Figure 7: the fixed-point 17 bits 57 gain structure

The 17-bit implementation of the 33 gain structure has been illustrated in figure 8. So no loss precision happens here as the format of the data is such that the possibility of a carry bit set does not influence the performance structure.

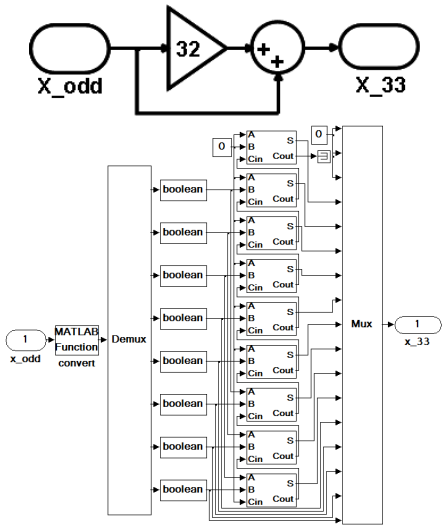

Figure 8: The fixed-point 17 bits 33 gain structure

In the same manner the implementation of 214 gain shown in figure 9, can be achieved by adding the 2 shifted version of the previous structure of 33 gain to the original one, and the obtained result is added to the 8 shifted x_odd sample before it undergo a two left shift operation to form finally the desired gain

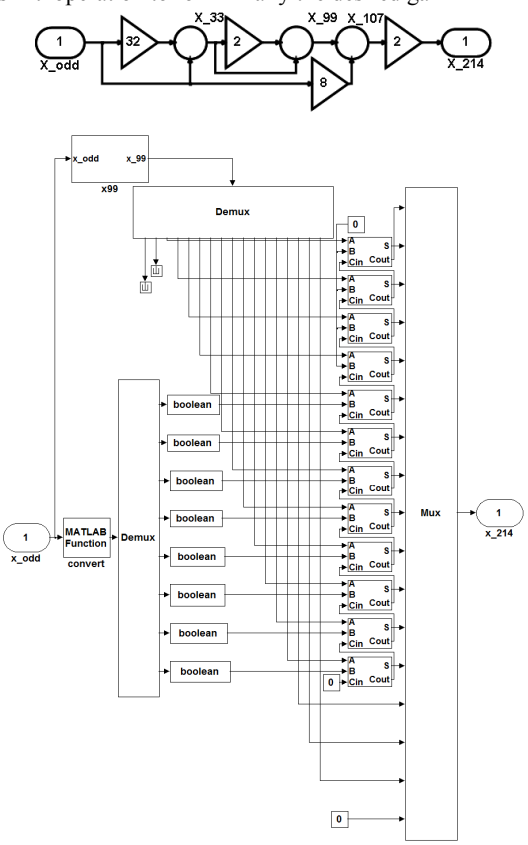

Figure 9: The fixed-point 17 bits 214 gain structure

The result of multiplication by 124 coefficient is being added to the delayed output sample of the 57 gain. Thus, the second way constituting in the subtraction between the sample having undergone a multiplication by 214 coefficient and the weighted sample using 33 coefficient after it has been delayed.

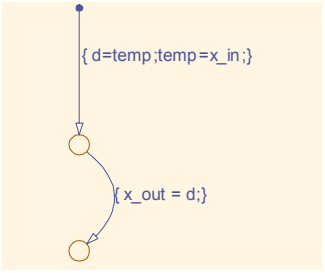

Figure 10: The fixed-point 17-bits delayer block Simulink implementation

The decimator filter will accept 8-bits inputs and will output also an 8-bits samples. The arrangement for this decimator is shown in figure 11, and here, the internal word length increases by the arithmetic operations, finally the length of decimator output will be limited to eight bits. The multiplication in this example is undertaken using a digital multiplier. So, the design here is purely combinational logic which includes the divider with 256 to perform the output decimator.

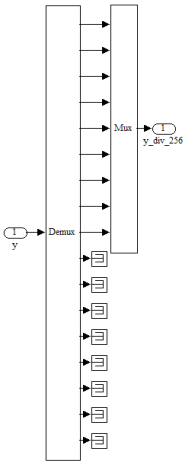

Figure 11: The fixed point 8-bits right shifter structure Simulink implementation

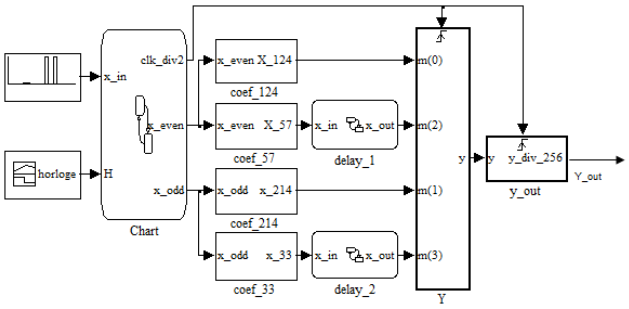

Figure 12 : The fixed point Daubechies decimator filter Simulink implementation.

5. IMPLEMENTATION

The various blocks used in this application are materialized by using the language of hardware description VHDL. Making it possible to implementing it on a reconfigurable platform (Altera or Xilinx type).

Indeed, before implementing our application we used ModelSim to write and execute test-benches to validate the correct operation of each component separately, and consequently to make the analysis and comparison with the results obtained in the Matlab/Simulink environment. ¶This stage consists in generating signals with the Matlab language which will be used thereafter in Modelsim.

The vhdl code designed for the example Daubechies decimator filter has been subject to synthesis in code through logic synthesizer, included in the Foundation Software. in the appropriates devices of FPGA.

6. RESULTS AND DISCUSSIONS

The implementation was elaborate following has subjective study.One was interested in the number of logic elements necessary for the implementation of the filter. It happened to be 190 of 10570 which represents 2% for a device Altera Stratix EP1S10B672C6 [5] and 112 of 2352 4% for a device Xilinx Spartan II [6]

To implement our approach we recommended to generate a signal characterized by a harmonic component of three separate sequences that each one having its own frequency shown in figure 13.

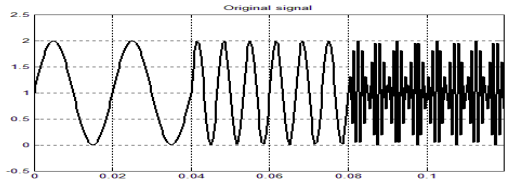

Figure 13. The original signal using Matlab/Similink

Figure 14 shows the result of analyzing and synthesizing with MATLAB/SIMULINK of the original signal using filter decimator previously described. Figure 15 illustrates the result of doing the same filtering operation using the Modelsim environment. The results of simulation show a best conformity between Matlab/simulink and Modelsim environment.

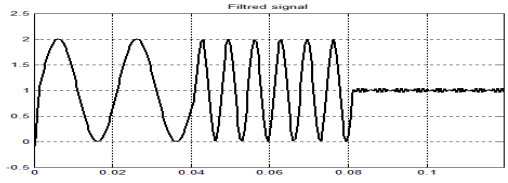

Figure 14. The result of simulation with Matlab/Similink

We easily observe a significant attenuation concerning the high frequency component due to the operation of the filtering imposed by the structure of this decimator filter

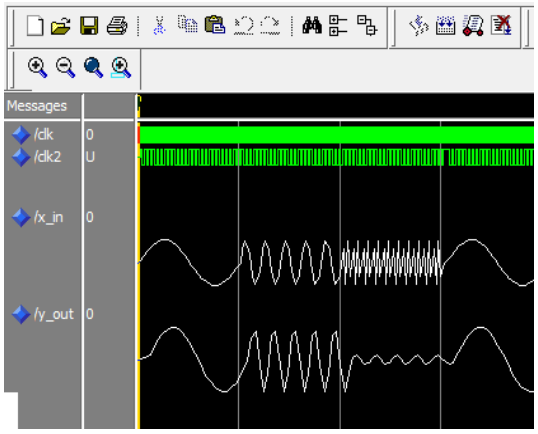

Figure 15. The result of simulation with ModelSim of Daubechies decimator VHDL filter implementation

7. CONCLUSION

In this paper, a design approach was implemented to realize a low power Daubechies decimator filter. The structure of this decimator has advantages in high speed operations such as digital RF/RI signal processing and the performance in term of power consumption and gate count were also compared. So, in order to achieve low power consumption, the operating clock frequency and hardware reduction concept were implemented. The important applications discussed here include the digital sequence decimation of eight-bit unsigned component of three frequencies. The fixed point Matlab/Simulink is used to perform a confrontation between simulation and hardware implementation task.

8. REFERENCES

[1] Texas Corporation, www.ti.com

[2] M. Smith, 1997 Application-specific integrated circuits. USA: Addison Wesley Longman,.

[3] R. Seals and G. Whapshott, 1997 Programmable Logic: PLDs and FPGAs. UK: Macmillan,

[4] Uwe Meyer-Baese., Digital Signal Processing with Field Programmable Gate Arrays, Third Edition, ISBN 978-3-540-72612-8 Springer Berlin Heidelberg NewYork.

[5] http://www.altera.com/products/devices/stratix-fpgas/stratix/stratix/stx-index.jsp

[6] http://www.xilinx.com/support/documentation/spartan-ii.htm