Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 53(67), Fascicola 2, 2008

FPGA-based Discrete Wavelet Using Distributed Arithmetic

Gavrincea Ciprian George [1], Tisan Alin[2]

Abstract – This paper presents theoretical and practical aspects in conjunction with hardware implementation of wavelet transform using distributed arithmetic. Implementing the multiplier unit using distributed logic, the designer can save hardware resource. The solution presented in this paper is using MatLab-Simulink environment to implement the algorithm in FPGAs.
Keywords: FPGA, wavelet, distributed arithmetic

I. INTRODUCTION

Wavelet analysis algorithms demands a lot of inner products computation. The key element for inner product computation is the multiplier unit. A method of hardware implementation of multiplier unit using distributed arithmetic is presented in this paper.
Nowadays, companies are putting a big value on time to market interval. To help researchers and engineers to achieve better time to market results, Xilinx had developed Simulink Matlab toolbox. With the help of this toolbox, any engineer can develop VHDL (VHSIC hardware description language) code for FPGAs (Field Programmable Gate Array) using Matlab knowledge instead of VHDL design knowledge. The hardware solution presented in this paper take advantage of the Simulink toolbox features.
During the last several years the wavelet transform (WT) has emerged as an important signal processing research tool. Wavelet transform theory as it relates to filter banks was refined by several researchers in the late 1980s and early 1990s, including Mallat (1989), Daubeehies (1992), Vetterli and Herley (1992), and Vaidyanathan (1993). Though work on wavelet theory has been extensive, research into wavelet transform applications is still in its infancy. Of particular interest is the use of the wavelet transform for the analysis of transient signals, since it is seen as an improvement over the traditionally used short-time Fourier transform (STFT). This is based on the fact that by using different scales of the analyzing wavelet, the wavelet transform is able to locate in.

II. WAVELET TRANSFORM

The basic idea underlying wavelet analysis consists of expressing a signal as a linear combination of a particular set of functions obtained by shifting and dilating one single function called a mother wavelet. Several different mother wavelets have been studied in Daubechies (1988) and Meyer (1989). The decomposition of the signal into the basis of wavelet functions implies the computation of the inner products between the signal and the basis functions, leading to a set of coefficients called wavelet coefficients. The signal can consequently be reconstructed as a linear combination of the basis functions weighted by the wavelet coefficients. In order to obtain an accurate reconstruction of the signal, a sufficient number of coefficients have to be computed.
A fundamental property of the wavelet transform (WT) is that the time resolution and frequency resolution vary in the time- frequency plane. The continuous wavelet transform allows a variable coverage of the time-frequency plane. The transform is defined as:

$$CWT_x(\tau,a)=\frac{1}{\sqrt{a}}\int x(t)\Psi^*\left(\frac{t-\tau}{a}\right)dt \qquad (1)$$

where a is called the scaling factor, τ is the translation parameter, and Ψ is the window function or wavelet.[2]

A. *Discrete wavelet transform*

Though the CWT is useful for the mathematical derivation of wavelet transform theorems and properties, in computational applications (where signals and filters are discrete), it is the discrete wavelet transform (DTWT) that is used.
Discrete wavelet transform can be implemented as a set of filter banks comprising a high-pass and a low-pass filters, each followed by down sampling by two. The low-pass filtered and decimated output is

[1] North University of Baia Mare, Electronic and Computer Engineering Department, V. Babes 62A, Maramures, e-mail gcg@ubm.ro
[2] North University of Baia Mare, Electronic and Computer Engineering Department, V. Babes 62A, Maramures, e-mail atisan@ubm.ro

recursively passed through similar filter banks to add the dimension of varying resolution at every stage. This is mathematically expressed:

$$DWDT_{x(n)} = \begin{cases} d_{j,k} = \sum x(n) h_j^*\left(n - 2^j k\right) \\ a_{j,k} = \sum x(n) g_j^*\left(n - 2^j k\right) \end{cases} \quad (2)$$

The coefficients $a_{j,k}$ and $d_{j,k}$ refer to the approximation and detail components in the signal, respectively. The functions h(n) and g(n) in this equation represent the coefficients of the high-pass and the low-pass filters.[2]

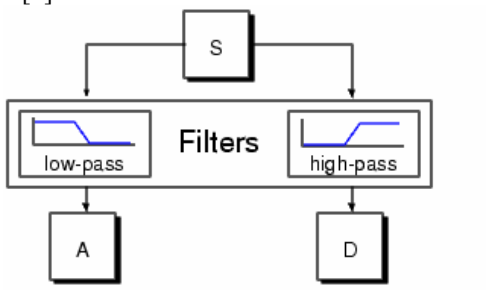

Fig. 1. Wavelet analysis concept.

III. DISTRIBUTED ARITHMETIC

Distributed arithmetic is a bit level rearrangement of a multiply accumulate to hide the multiplications. It is a powerful technique for reducing the size of a parallel hardware multiply-accumulate that is well suited to FPGA designs. The distributed arithmetic approach is an inherently serial process, but the computation of each bit can be performed in parallel to obtain more performance if necessary [6].

Using the scaling accumulator multiplier, we can construct a multiple product term function in a relatively small space if we are willing to accept a serial input. In this case, we feed four parallel scaling accumulators with unique serialized data. Each multiplies that data by a possibly unique constant, and the resulting products are summed in an adder tree as shown in figure 2.

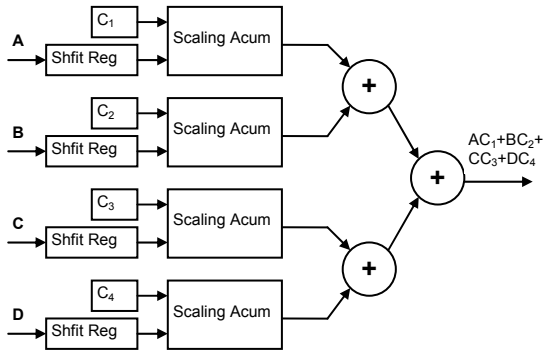

Fig. 2

If we consider that the scaling accumulator multiplier is really just a sum of vectors, then we can rearrange the circuit. The adder tree combines the 1 bit partial products before they are accumulated by the scaling accumulator. All we have done is rearranged the order in which the 1xN partial products are summed. Now instead of individually accumulating each partial product and then summing the results, we postpone the accumulate function until after we've summed all the 1xN partials at a particular bit time. This simple rearrangement of the order of the adds has effectively replaced N multiplies followed by an N input add with a series of N input adds followed by a multiply. This arithmetic manipulation directly eliminates N-1 Adders in an N product term multiply-accumulate function. For larger numbers of product terms, the savings becomes significant.

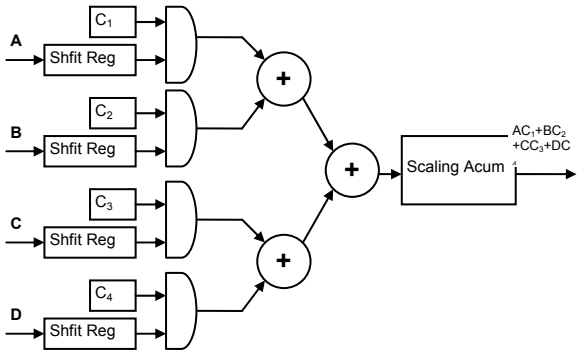

Fig. 3

Further hardware savings are available when the coefficients C_n are constants. If that is true, then the adder tree shown above becomes a boolean logic function of the 4 serial inputs. The combined 1xN products and adder tree is reduced to a four input look up table. The sixteen entries in the table are sums of the constant coefficients for all the possible serial input combinations.

Table 1

Address	**Data**
0000	**0**
0001	**C_1**
0010	**C_2**
…	
1110	**$C_4+C_3+C_2$**
1111	**$C_4+C_3+C_2+C_1$**

Most often, we have more than 4 product terms to accumulate. Increasing the size of the LUT might look attractive until you consider that the LUT size grows exponentially. Considering the construction of the logic we stuffed into the LUT, it becomes obvious

that we can combine the results from the LUTs in an adder tree. The area of the circuit grows by roughly 2n-1 using adder trees to expand it rather than the 2n growth experienced by increasing LUT size. For FPGAs, the most efficient use of the logic occurs when we use the natural LUT size (usually a 4-LUT, although and 8-LUT would make sense if we were using an 8 input block RAM) for the LUTs and then add the outputs of the LUTs together in an adder tree.

IV. HARDWARE IMPLEMENTATION

The wavelet transform processes a signal by decomposing it into successive approximation and detail signals. The approximation signal is re-sampled at each stage, and the detailed coefficients are kept. For decomposition into J scales, the transform coefficients consist of J scales of detailed coefficients and the Jth scale approximation coefficient. The process of signal decomposition using wavelet transform is called wavelet analysis.

Figure 4 illustrates a 4 scale wavelet analysis circuit. On each stage, input data is passed through a low pass filter and a high pass filter. The output of the low pass filter provides approximation signal, while the output of the high pass filter provides detailed signal. Because output of each filter has the same number of sample as the original signal, detailed and approximation signals are decimated by 2.

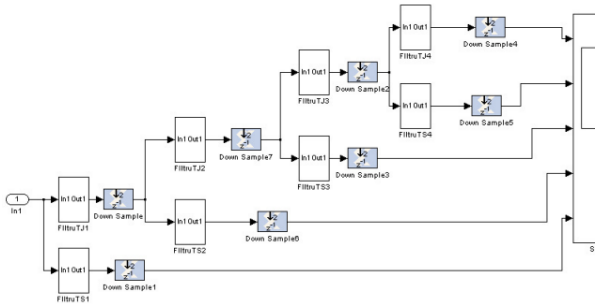

Fig. 4. 4 scale wavelet analysis.

Perfect reconstruction filters are used to perform analysis and synthesis of the signal. Filter coefficients can be obtained from Matlab environment. For this study we use a four level decomposition using Daubechies 4 wavelet. FIR filters are implemented using a dual ram memory for storing filter coefficients and data. The solution for the hardware implementation of the FIR filter, using distributed arithmetic is illustrated in figure 5.

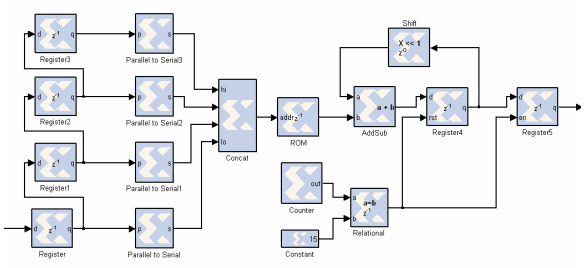

Fig. 5. FIR using distributed arithmetic

The main blocks for implementation of FIR filter consist in a ROM memory for storing filter coefficients, a scaling multiplier, and 4 parallel to serial registers. The parallel to serial register are replacing the shifting registers presented in the previous section. The adder tree is replaced by a ROM memory. The scaling accumulator is implemented using an adder, a register and a shifting register.

Table 2 presents estimated resources needed to implement a 4 scale wavelet filter, on XILINX Spartan3E XC3S500E, using hardware solution presented in [7]. In order to save hardware resources is best to use hardware multiplier. Because FPGA chips have a small number of hardware multipliers, a careful management of this type of resources is required.

Table 2

Resources	Spartan3 HW multipliers	%	Spartan3	%
Slice	2372	51	4340	93,2
FFs	3616	38,8	7008	75,2
Block RAM	16		16	
LUT	2104	22,6	6640	71,3
Multiplier	16/20	80	0	0

We can observe that by using built-in hardware multiplier in our implementation we can save around 40% of hardware resources. But the number of built-in hardware multiplier is small and only some FPGA circuit has them. Another way of reducing the number of hardware resources is by using distributed arithmetic.

Table 3 presents estimated resources needed to implement a 4 scale wavelet filter, using the distributed arithmetic method presented above, on XILINX Spartan3E XC3S500E. The amount of hardware resources used for this implementation is with 25% less compare with the previous case, and without using built in multipliers. If we compare with the built in hardware multiplier implementation we can notice that the hardware multiplier have better results. If we are using an FPGA version that doesn't have built in multipliers or if we want to use the hardware multiplier for another application than using distributed arithmetic it is a good solution for saving hardware resources.

Table 3

Resources	Spartan3 HW multipliers	%
Slice	3168	68,03
FFs	5520	59,1
Block RAM	-	-
LUT	2688	28,8
Multiplier	0/20	-

V. CONCLUSION

This paper presents a hardware solution for implementing wavelet analysis algorithm using a reduce number of hardware resources. The reduction on the amount of hardware resources is obtain by using distributed arithmetic for implementation of FIR filters used in each level of wavelet decomposition. By using this approach we save hardware resources but the cost is timing performance. This is because distributed arithmetic solution is bit serial in nature. A bit serial structure processes the data one bit at a time. The advantage is that all of the bits pass through the same logic, resulting in a huge reduction in the required hardware. Typically, a bit serial design requires only about $1/n^{th}$ of the hardware needed for the equivalent n-bit parallel design. The price of this logic reduction is that the serial hardware takes n clock cycles to execute. The method presented in this paper could be a solution for application where time budget it is relaxed and which require a better hardware economy.

Design of current DSP applications using state-of-the art multi-million gates devices requires a broad foundation of the engineering skills ranging from knowledge of hardware-efficient DSP algorithms to CAD design tools. The requirement of short time-to-market, however, requires replacing the traditional HDL based designs by a MatLab/Simulink based design flow. This not only allows MatLab users to design FPGAs but also to by-pass the hardware design engineer leading to a significant reduction in development time.

Simulink design flow for FPGAs is an interesting alternative both for a University lab as well as for the professional developers in industry. The design flow allows a software developer to quickly explore FPGA design options in terms of size and speed and to check if the resulting design fulfills the design constrains.

REFERENCES

[1] D. Moshou, I Hostens, G Papaioannou, H Ramon, “Wavelets and self-organising maps in electromyogram analysis”, ESIT 2000, 14-15 September 2000, Aachen, Germany

[2] C. S. Burrus, R. A. Gopinath, Introduction to Wavelets and Wavelet Transforms, Prentice Hall Inc.,Upper Saddle River, New Jersey, 1998.

[3] Yousef M. Hawwar_, Ali M. Reza_, Robert D. Turney, „Filtering (denoising) in the wavlet transform domain“,

[4] S.G. Mallat, “A theory for multiresolution signal decomposition: The wavelet representation”, IEEE Trans.Pattern Anal. Machine Intell., vol. 2, pp. 674-693, 1989.

[5] S. Masud and J.V. McCanny, “Rapid design of biorthogonal wavelet transforms”, IEEE Proceedings of Circuits, Devices and Systems, Volume: 147 Issue: 5, 2000, pp: 293 -296

[6] R. Andraka and A. Berkun, “FPGAs make a radar signal processor on a chip a reality”, IEEE Proceedings of Asilomar Conference on Signals,Systems and Computers, Monterey , 1999

[7] C. Gavrincea, A. Tisan, A. Buchman, St. Oniga, Survey of wavelet based denoising filter design, Proceedings of the 30th International Spring Seminar on Electronics Technology, Cluj Napoca, Romania, May 9-13, 2007, ISBN 978-973-713-174-4, pag 72-74.