DESIGNING DSP ALGORITHM WITH THE VIRTEX-4 XTREMEDSP SLICE

Anna S. Kuncheva, Dimitar Nikolov, Marin Hristov

Department of Microelectronics, ECAD Laboratory, FEET, Technical University of Sofia, 8,
Kliment Ohridski St, 1000 Sofia, Bulgaria, e-mail: anika@ecad.tu-sofia.bg;

Finite impulse response (FIR) filter is the key functional block in DSP (Digital Signal Processing) designs and nearly always form the starting point for analyzing an architecture.

This paper contains a new filter architecture, along with FPGA- Virtex-4 (Xilinx) and XtremeDSP (DSP48) slice. The XtremeDSP slice is a high performance multiplier and arithmetic unit with great flexibility that can form the building block of DSP algorithms implemented in FPGAs. Traditional adder-tree approach limited the performance and extensibility of given filter implementation. By using adder-chain style of implementation these limitations are lifted.

A novel approach in both design and implementation of digital FIR filter using the DSP system element of Virtex-4 (Xilinx) architecture – XtremeDSP (DSP48) slice is presented. Embedded nature of the XtremeDSP slice has a radical impact on reducing the power consumed by high-speed multiply and add function.

Keywords: DSP, FIR, XtemeDSP(DSP48), FPGA

1. INTRODUCTION

Digital Signal Processing (DSP) systems based on software are flexible, but due to the sequential nature of microprocessors, they often suffer from insufficient processing capability. Dedicated hardware on the other hand can provide the highest processing performance, but is less flexible for changes. Reconfigurable hardware devices offer both the flexibility of computer software, and the ability to construct custom high performance computing circuits. Thus, in many cases they make a good compromise between software and hardware solutions. The flexible nature of these devices open up for a new range of circuits that exploit their reconfigurable property. A large variety of real-world applications exist for such hardware [1]. Digital filters are a very important part of DSP, and there is a present need for their hardware implementation.

The dedicated DSP blocks in high-end FPGAs – such as the Xilinx® XtremeDSP™ slice (also referred to as DSP48) in Virtex™-4 devices – are playing a critical role in designing highperformance DSP systems.

This paper shows how a Xilinx Virtex XC4VFX12-10FF668 FPGA (a reconfigurable device) can be used to implement a low-pass FIR filter (Fig. 1). However, it must take this adder-chain configuration into account when designing functions that exploit the XtremeDSP slice. Herein lies the fundamental change in the approach to filter design. The simple, traditional adder-tree approach limited the performance and extensibility of a given filter implementation. By using adder-chain-style implementations, these limitations are lifted and the huge benefits Virtex-4 FPGAs offer are possible.

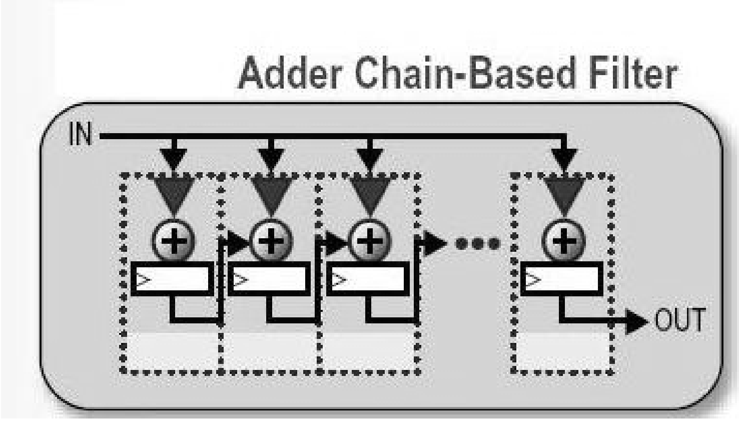

Fig. 1: Structure of FIR filter

2. DIGITAL FILTERS

Digital filters are the digital counterpart to analog filters. Digital filters operate on numbers opposed to analog filters which operates on voltages. The basic operation of a digital filter is to take a sequence of input numbers (e.g. samples), and compute a different sequence of output numbers. There exists a range of different digital filters, where the most common is the finite impulse response filter (FIR). Digital filters are used in a wide range of DSP-applications.

2.1 Filter Techniques

A FIR-filter (Finite input response filter) is a digital filter that is widely used in digital signal processing applications. The FIR-filter computes an output from a set of input-samples. The set of input samples is multiplied by a set of coefficients and then added together to produce the output. The filter behavior is determined by the filter-coefficients.

The general FIR filter equation is a summation of products (also known as an inner product) defined in the equation:

$$y_n = \sum_{k=0}^{N-1} x_{n-k} * h_k$$

In this equation, a set of N coefficients is multiplied by N respective data samples, and the results are summed to form an individual result. The values of the coefficients determine the characteristics of the filter: low-pass, band-pass, or high-pass.

With symmetrical coefficient, and assuming an even number of coefficients, we have:

$$\begin{aligned} &h_0 = h_{N-1} \\ &h_1 = h_{N-2} \\ &\ldots \\ &h_{N/2-1} = h_{N/2} \end{aligned}$$

Therefore, the output is now:

$$y_n = \sum_{k=0}^{\frac{N}{2}-1} (x_{n-k} + x_{n-N+h+1}) \times h_k$$

This is will half the number of multiplications required.

The frequency band attenuation for each stage can be determined using the following formula:

$$\textit{Passband: } 0 \le f \le F_p$$
$$\textit{Stopband: } F_i - F_s/2M < f < F_{i-1}/2$$
$$Ripple_{passband} : \delta_p/I$$
$$Ripple_{Stopband} : \delta_s$$

where:

Fs: Sampling frequency of filter
I: Total number of stage
Fi= F_{i-1}/Mi;   i is number of stage, (i=1, 2, ...I);

The requirements for such a filter are highlighted in Table1:

Filter parameter	***FIR filter***
Sampling frequency:	Fs = 75.24 MHz
Passband frequency:	Fpass = 100 kHz
Stopband frequency:	Fstop = 375 kHz
Passband Max ripple:	Apass = 0.00025 dB
Stopband attenuation:	Astop = 90 dB

Table1: Requirements for two stages of FIR filtering

2.2 Quantization of coefficients:

Next step in this design is to select the number of bits used to represent each coefficient. It is obvious that quantization of the filter coefficient reduces the filter performance. Once again, Matlab tool is used to evaluate the effect that of quantization of the coefficient has on the response of the FIR filter [3]. When using 20 bits for quantization, the response of the quantized filter matches closely the reference filter, both in the passband and stopband. It is shown in figure 2. Therefore, 20 bits for the coefficients, the FIR filter have been selected.

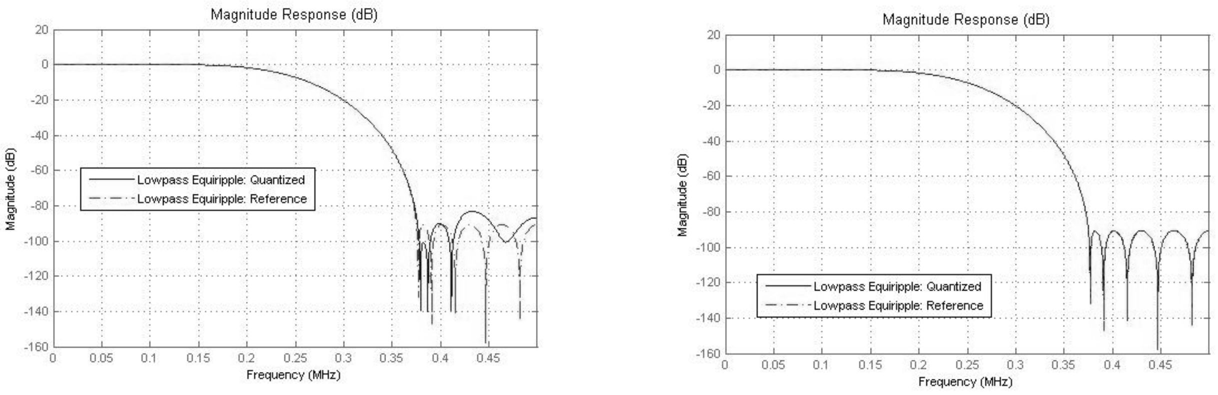

Fig. 2: Frequency response of the first FIR filter with and without coefficients quantization
a) using 16-bit quantization, b) using 20-bit quantization

3. HARDWARE IMPLEMENTATION

3.1. XtremeDSP Slice

The XtremeDSP slice (DSP48) is a high-performance multiplier and arithmetic unit with great flexibility that can form the building block of many DSP algorithms implemented in FPGA. A detailed diagram of the DSP48 structure is shown in Fig. 3.

In the Virtex-4 architecture, XtremeDSP slices are arranged in columns [2]. The most important aspect about the column is cascade logic and routing between each block, which exists both the input and output stages of each slice. This dedicated routing enables a number of filters to be built entirely within the XtremeDSP slice, thus removing the need for signals to be routed trough the FPGA interconnect.

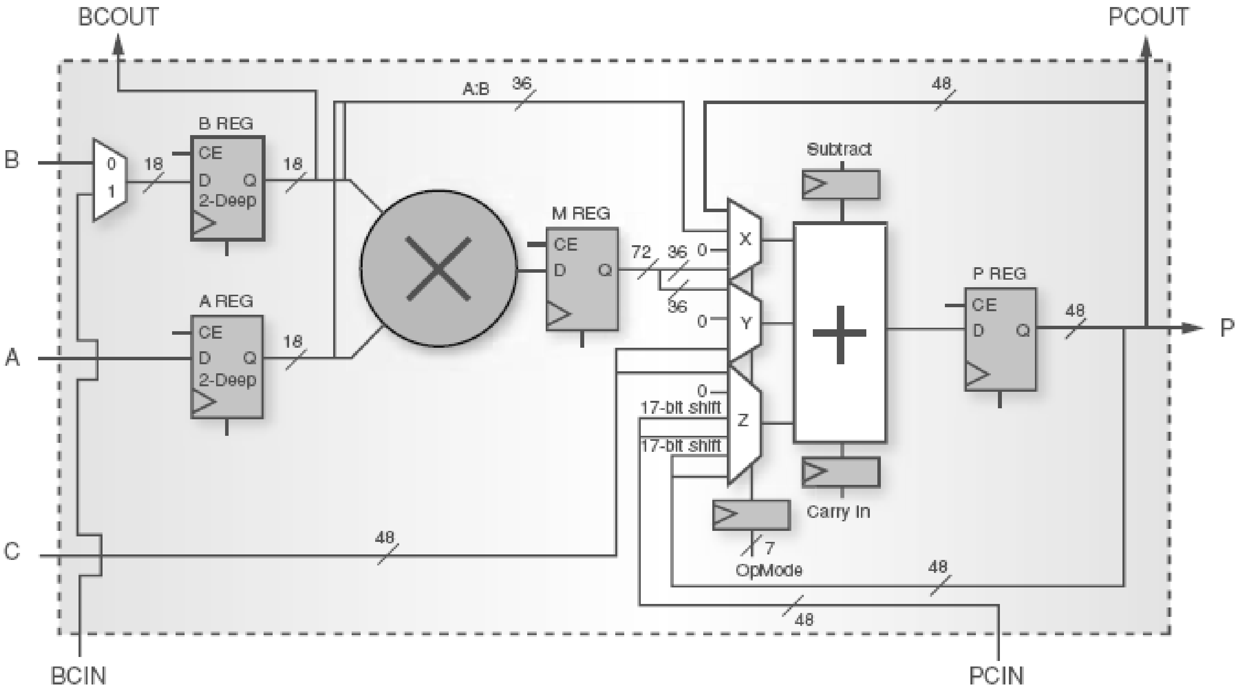

Fig. 3 XtremeDSP

The XtremeDSP slice comprises four main sections:

- I/O registers
- 18 x 18 signed multiplier
- Three-input adder/subtractor
- Op-mode multiplexers

The I/O registers ensure a maximum clock performance of 500 MHz in the fastest speed grade device (400 MHz in the slowest speed grade), also ensuring support for higher sample rates. The dynamic op-mode multiplexers are key to the functionality of the structure; they are responsible for the DSP48's great flexibility.

3.2. FPGA Implementation

Hardware implementation of FIR-filters allows the filterfunctions to be executed in a parallel manner which makes improved filter processing speed possible.

The proposed FIR filter has 20 coefficients and a sample rate of 75.24 MHz. As noted earlier, the maximum capable clock speed of the XtremeDSP slice is 400 MHz in the slowest speed grade (-10). Therefore, we have a total of five clock cycles to perform the required 20 multiply and adds to form the result. This equation determines how many multipliers to use for a particular semiparallel architecture:

Number of Multipliers = (Maximum Input Sample Rate x Number of Coefficients) / Clock Speed

For this design, the required number of multipliers will be four. Once we have determined the required number of multipliers, there is an extendable architecture using the XtremeDSP slices that can serve as the basis for the filter.

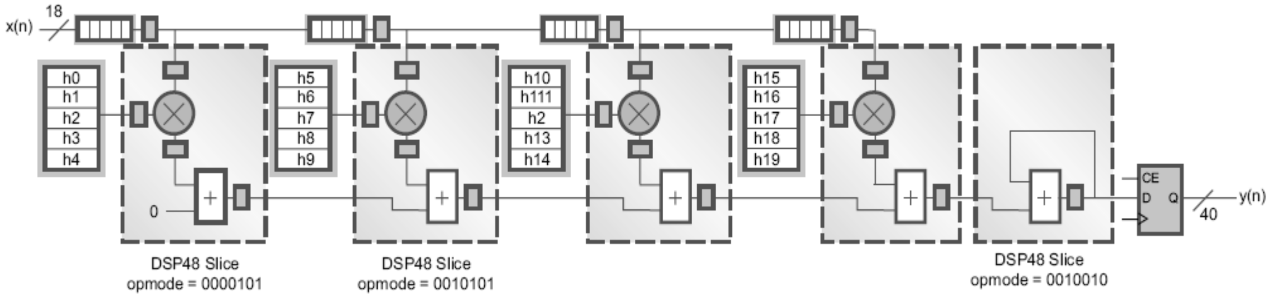

Fig. 4 The four multiplier FIR filter

XtremeDSP arithmetic units are designed to be chained together easily and efficiently thanks to dedicated routing between slices. Figure 4 illustrates how the four XtremeDSP multiply and add elements are cascaded together to form the main part of the filter. It is critical to highlight the usage of the adder chain here rather than the more traditional adder tree. The adder chain has a profound impact on the control logic required for the filter, as well as its efficiency, because of the mapping to the XtremeDSP slice.

Continuing to analyze the filter structure, an extra XtremeDSP slice is required to perform the accumulation of the partial results, thus creating the final result. A new result is created every five clock cycles. This means that for every five cycles the accumulation must be reset to the first inner product of the next result. This reset (or load) is achieved by changing the op-mode value of the XtremeDSP slice for a single cycle, from 0010010 to 0010000 (this is just a single bit change). At the same time, the capture register is enabled and the final result stored on the output.

The FIR filter was implemented in a Xilinx XC4VFX12-10FF668 FPGA. The structure of FIR filter from Figure 4 is implemented in an FPGA using VHDL. This new filter architecture, along with Virtex-4 devices and the XtremeDSP slice, addresses the demanding needs of current and future DSP designs. However, it is only one filter in an extremely large array of possible implementations, not to mention other DSP functions such as IIRs, FFTs, and DCTs.

The faster a particular function operates, the smaller it becomes. The semiparallel FIR filter used five XtremeDSP slices running at 375 MHz instead of 20 XtremeDSP slices running at 74.25 MHz.

Logic Slices	108
XtremeDSP Slice	5
Performance (Sample Rate)	75.24 MHz
Performance (Clock Frequency)	400 MHz

Table 2 – Resource utilization and performance of four-multiplier 20-tap FIR filter

3.3 Storage of filter-coefficients:

To be able to change the filter-coefficients in one clockcycle, Virtex LUTs set to operate in shift-register mode are between filter-coefficients takes less time than loading new samples. This enables computation of a fitness-value for each set of filter-coefficients every time a sample is loaded. The fitness value for a set of filter-coefficients is simply the difference between the output of the FIR-filter and a reference signal. When this difference is small, the FIR filter has efficiently adapted to remove the difference between the input-signal to the FIR-filter and the reference signal. To compensate for the phase shift of the filter, the reference signal is delayed by the same amount of clockcycles as the length of the FIR-filter. It is expected that the evolved filters will have a linear phase-shift.

4. CONCLUSION

The simple, traditional adder-tree approach limited the performance and extensibility of a given filter implementation. By using adder-chain-style implementations, these limitations are lifted and the huge benefits Virtex-4 FPGAs offer are possible.

Using tool such as Matlab or other, it is possible to quickly find the filer order, the required quantization level for the coefficients and their values. Finally, by analyzing the design, is finds an efficient way to implement the filter in hardware. The use of top-down design methodology decreased the total design time. The high level hardware description language VHDL fully supports arithmetic and binary manipulations that are specific for this design.

5. REFERENCES

[1] Vinger A. K, Jim Toresen, *Implementing Evolution of FIR-Filters Efficiently in FPGA.*

[2] XtremeDSP for Virtex-4 FPGAs User Guide

[3] Kuncheva A.S., T. Mougel, L. Fujcik, B. Donchev, M.H. Hristov, Design of decimation filter for novel sigma-delta modulator, The 14th International Scientific and Applied Science Conference - Electronics 2005, Sozopol, Bulgaria, September 2005, Book 5, pp. 56-61, ISBN 954 438 521 5