

# Low Power FPGA Implementation of Digital FIR Filter Based on Low Power Multiplexer Base Shift/Add Multiplier

Bahram Rashidi, Farshad Mirzaei, Bahman Rashidi, and Majid Pourormazd

**Abstract**—In this paper, the authors present implementation of a low power and low area digital Finite Impulse Response (FIR) filter. The we method for reduce dynamic power consumption of a digital FIR filter is use of low power multiplexer based on shift/add multiplier without clock pulse and we applied it to fir filter until power consumption reduced thus reduce power consumption due to glitching is also reduced. The minimum power achieved is 56mw in fir filter based on shift/add multiplier in 100MHZ with 8bits inputs and 8bits coefficients. The proposed FIR filter was synthesized implemented using Xilinx ISE V7.1 and Virtex IV FPGA to target device xc4vlx200 also power is analized using Xilinx XPower analyzer.

**Index Terms**—Low Power, shift/add multiplier, barrel shifter.

## I. INTRODUCTION

Their impulse response, which can be either finite or infinite. The methods for designing and implementing these two filter classes differ considerably. Finite impulse response (FIR) filters are digital filters whose response to a unit impulse (unit sample function) is finite in duration. This is in contrast to infinite impulse response (IIR) filters whose response to a unit impulse (unit sample function) is infinite in duration. FIR and IIR filters each have advantages and disadvantages, and neither is best in all situations. FIR filters can be implemented using either recursive or nonrecursive techniques, but usually nonrecursive techniques are used. FIR filters are widely used in digital signal processing (DSP) systems that are characterized by the extensive sequence of multiplication operations. FIR filters are widely used in various DSP applications. In some applications, the FIR filter circuit must be able to operate at high sample rates, while in other applications, the FIR filter circuit must be a low-power circuit operating at moderate sample rates. The low-power or low-area techniques developed specifically for digital filters can be found in. Parallel (or block) processing can be applied to digital FIR filters to either increase the effective throughput or reduce the power consumption of the original filter. While sequential FIR filter implementation has been

given extensive consideration, very little work has been done that deals directly with reducing the hardware complexity or power consumption of parallel FIR filters [1]. Traditionally, the application of parallel processing to an FIR filter involves the replication of the hardware units that exist in the original filter. The topology of the multiplier circuit also affects the resultant power consumption. Choosing multipliers with more hardware breadth rather than depth would not only reduce the delay, but also the total power consumption [2]. A lot of design methods of low power digital FIR filter are proposed, for example, in [3] they present a method implementing fir filters using just registered adders and hardwired shifts. They extensively use a modified common sub expression elimination algorithm to reduce the number of adders. In [4] they have proposed a novel approach for a design method of a low power digital baseband processing. Their approach is to optimize the bitwidth of each filter coefficient. They define the problem to find optimized bitwidth of each filter coefficient. In [5] presents the method reduce dynamic switching power of a fir filter using data transition power diminution technique (DPDT). This technique is used on adders, booth multipliers. In [6] this research proposes a pipelined variable precision gating scheme to improve the power awareness of the system. This research illustrates this technique is to clock gating to registers in both data flow direction and vertical to data flow direction within the individual pipeline stage based on the input data precision. The rest of the paper is structured as follow. Section2 gives a summary of fir filter theory and in Section3 presents the architecture adopted in our implementation. Comparison of our implementation with those done is given at section4. Finally section5 provides the conclusion of this paper.

## II. FIR FILTER THEORY

A digital filter takes a digital input, gives a digital output, and consists of digital components. In a typical digital filtering application, software running on a DSP reads input samples from an A/D converter, performs the mathematical manipulations dictated by theory for the required filter type, and outputs the result via a D/A converter. An analog filter, by contrast, operates directly on the analog inputs and is built entirely with analog components, such as resistors, capacitors, and inductors. There are many filter types, but the most common are lowpass, highpass, bandpass, and bandstop. A lowpass filter allows only low frequency signals (below some specified cutoff) through to its output, so it can be used to eliminate high frequencies. A lowpass filter is handy, in that

Manuscript received September 14, 2012; revised November 29, 2012.  
Bahram Rashidi and Farshad Mirzaei are with the University of Tabriz, Iran (e-mail: bahram88@ms.tabrizu.ac.ir, Farshad.mirzaei@ymail.com).  
Bahman Rashidi is with the Iran University of Science and Technology, Tehran, Iran (e-mail: b\_rashidi@comp.iust.ac.ir).  
Majid Pourormazd is with the University of Shahid chamran kenman , Iran (e-mail: Puorormazd@gmail.com).

regard, for limiting the upper most range of frequencies in an audio signal; it's the type of filter that a phone line resembles. A highpass filter does just the opposite, by rejecting only frequency components below some threshold. An example highpass application is cutting out the audible 60Hz AC power "hum", which can be picked up as noise accompanying almost any signal in the U.S. The designer of a cell phone or any other sort of wireless transmitter would typically place an analog bandpass filter in its output RF stage, to ensure that only output signals within its narrow, government-authorized range of the frequency spectrum are transmitted. Engineers can use bandstop filters, which pass both low and high frequencies, to block a predefined range of frequencies in the middle. A. Frequency response Simple filters are usually defined by their responses to the individual frequency components that constitute the input signal. There are three different types of responses. A filter's response to different frequencies is characterized as passband, transition band, or stopband. The passband response is the filter's effect on frequency components that are passed through (mostly) unchanged. Frequencies within a filter's stopband are, by contrast, highly attenuated. The transition band represents frequencies in the middle, which may receive some attenuation but are not removed completely from the output signal. In Fig. 1, which shows the frequency response of a lowpass filter,  $\omega_p$  is the passband ending frequency,  $\omega_s$  is the stopband beginning frequency, and  $A_s$  is the amount of attenuation in the stopband. Frequencies between  $\omega_p$  and  $\omega_s$  fall within the transition band and are attenuated to some lesser degree.

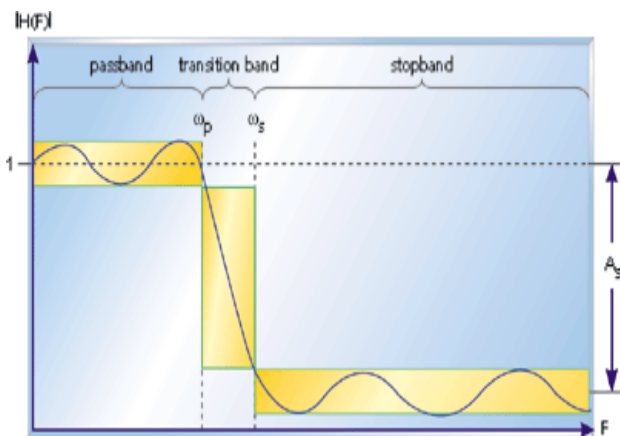


Fig. 1. The response of a lowpass filter to various input frequencies.

Given these individual filter parameters, one of numerous filter design software packages can generate the required signal processing equations and coefficients for implementation on a DSP. Before we can talk about specific implementations, however, some additional terms need to be introduced. Ripple is usually specified as a peak-to-peak level in decibels. It describes how little or how much the filter's amplitude varies within a band. Smaller amounts of ripple represent more consistent response and are generally preferable. Transition bandwidth describes how quickly a filter transitions from a passband to a stopband, or vice versa. The more rapid this transition, the higher the transition bandwidth; and the more difficult the filter is to achieve.

Though an almost instantaneous transition to full attenuation is typically desired, real-world filters don't often have such ideal frequency response curves. There is, however, a tradeoff between ripple and transition bandwidth, so that decreasing either will only serve to increase the other.[7]

Digital filters are typically used to modify or alter the attributes of a signal in the time or frequency domain. The most common digital filter is the linear time-invariant (LTI) filter. An LTI interacts with its input signal through a process called linear convolution, denoted by  $y = h \times x$  where  $h$  is the filter's impulse response,  $x$  is the input signal, and  $y$  is the convolved output. The linear convolution process is formally defined by:

$$y[n] = x[n]h[n] = \sum_{k=0}^{L-1} x[n]h[n-k] = \sum_{k=0}^{L-1} h[k]x[n-k]. \quad (1)$$

An FIR with constant coefficients is an LTI digital filter. The output of an FIR of order or length  $L$ , to an input time-series  $x[n]$ , is given by a finite version of the convolution sum given in (1), namely:

$$y[n] = x[n] \times h[n] = \sum_{k=0}^{L-1} h[k]x[n-k] \quad (2)$$

where  $h[0] \neq 0$  through  $h[L - 1] \neq 0$  are the filter's  $L$  coefficients.[8]

Fig. 2 shows the basic block diagram for an FIR filter of length  $N$ . The delays result in operating on prior input samples. The  $h_k$  values are the coefficients used for multiplication, so that the output at time  $n$  is the summation of all the delayed samples multiplied by the appropriate coefficients.

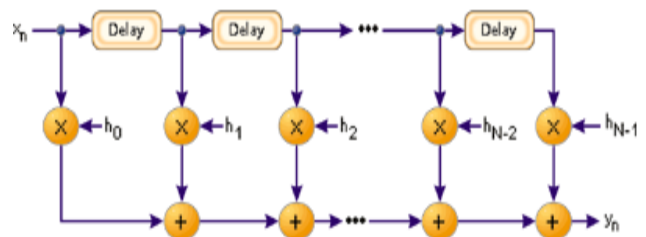


Fig. 2. The logical structure of an FIR filter.

The process of selecting the filter's length and coefficients is called filter design. The goal is to set those parameters such that certain desired stop band and pass band parameters will result from running the filter. Most engineers utilize a program such as MATLAB to do their filter design. But whatever tool is used, the results of the design effort should be the same:

- 1) A frequency response plot, like the one shown in Fig. 1, which verifies that the filter meets the desired specifications, including ripple and transition bandwidth.
- 2) The filter's length and coefficients.

The longer the filter (more taps), the more finely the response can be tuned. With the length,  $N$ , and coefficients, float  $h [N]=\{...\}$ , decided upon, the implementation of the

FIR filter is fairly straightforward. Listing 1 shows how it could be done in C. Running this code on a processor with a multiply-and-accumulate instruction (and a compiler that knows how to use it) is essential to achieving a large number of taps.

```

*Sample the input signal (perhaps via A/D).
sample = input();
*Insert the newest sample into an N-sample circular buffer.
*The oldest sample in the circular buffer is overwritten.
x[oldest] = sample;
*Multiply the last N inputs by the appropriate coefficients.
*Their sum is the current output.
y = 0;
for (k = 0; k < N; k++)
{
    y += h[k] × x[(oldest + k) % N];
}
oldest = (oldest + 1) % N;
*Output the result.
Output (y);
    
```

Listing 1. Implementation of an N-tap FIR filter in C As can see, an FIR filter simply produces a weighted average of its N most recent input samples. All of the magic is in the coefficients, which dictate the actual output for a given pattern of input samples.[7]

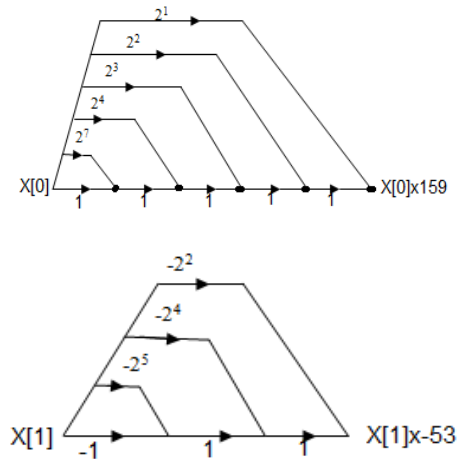
### III. IMPLEMENTATION FIR FILTER

In this paper, MAC architecture is considered. There is design and implementation low power multiplexer based shift/add multiplier for reduce the power consumption. Fir filter is implemented using the shift and add method. We perform all our optimization in the multiplier block. This multiplier is constitution of one shifter block (barrel shifter) one adder block and also control unit. Where in shifter block is based on multiplexer only. And proportional to select signal(shift[2..0]) input number left shift to magnitude 0 to 7 magnitude of shift specification by control unit i.e. based on weight of bits input signal to control unit. If slightly bit is zero we have non-shift and slightly bit is one to magnitude weight of bit shift is apply. That in continue show VHDL code of this block. And will has been explanation in related with shifter block. Adder block add together shifted number, final product favorable output. By implementation of shifter block based on multiplexer hardware and power consumption of FIR filter is very reduced. in Fig. 3 shown block diagram of low power multiplexer based shift/add multiplier. The constant multiplications are decomposed in to additions and shifts and the multiplication complexity is reduced. Its possible to implement the design in the two form described below:

1) The coefficients are changed to integer getting multiplied to a multiple power of 10, then we arrange these coefficients the positive power of 2. This procedure is shown below in graph form. Graph branches stand for left shift and nothes stand for sum.

2) First we arrange decimal coefficients according to negative and positive power of 2 (no need to them into integer). So the filter hardware and power consumption will reduce.

Design 1:  
 $c[0] = 0.159, c[1] = -0.53 \xrightarrow{\times 1000} c[0] = 159, c[1] = -53$   
 $c[0] = 128+16+8+2+1 = 2^7+2^4+2^3+2^2+2^1+2^0$   
 $c[1] = -32-16-4-1 = -2^5-2^4-2^2-2^0$



Design 2:  $c[0] = 3.75$   
 $c[0] = 2^1+2^{-1}+2^{-2}+2^0$

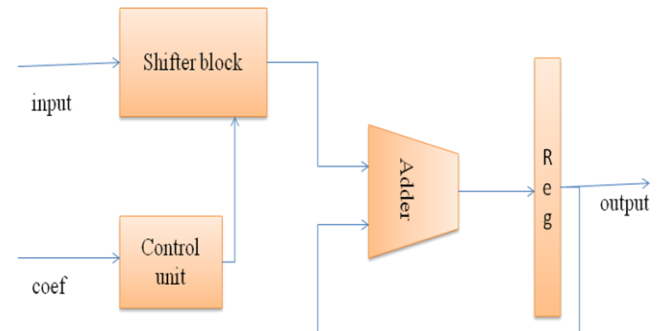
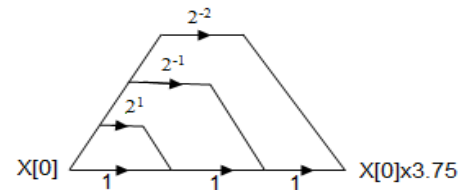


Fig. 3. Block diagram of low power multiplexer based shift/add multiplier.

The diagram of a shifter block (barrel shifter) is shown in Fig. 4 the input is an 8-bit vector. The output is a shifted version of the input, with the amount of shift defined by the “shift” input (from 0 to 7). The circuit consists of three individual barrel shifters. Notice that the first barrel has only one ‘0’ connected to one of the multiplexers (bottom left corner), while the second has two, and the third has four. For larger vectors, we would just keep doubling the number of ‘0’ inputs. If shift “001”, for example, then only the first barrel should cause a shift; on the other hand, if shift “111”, then all barrels should cause a shift.[9]

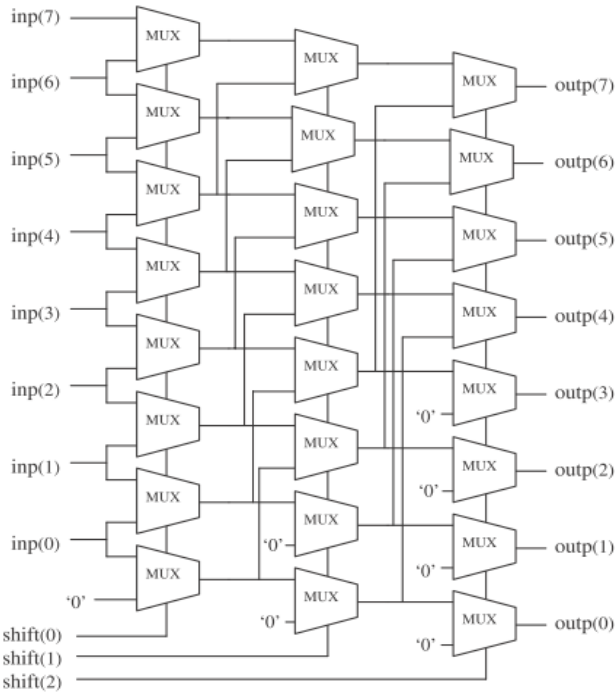


Fig. 4. Figure of multiplexer based on shifter block.

VHDL code for implementation of conversion any number based on power of 2 or control unit:

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;
entity power_2_control_unit is
port(
    b:in std_logic_vector(7 downto 0);
    out_33:out std_logic_vector(2 downto 0));
end power_2_control_unit;

architecture arch_ power_2_control_unit of
power_2_control_unit is
begin
process(b)
begin
if(b(0)='1')then
    out_33<="000";
elsif(b(1)='1')then
    out_33<="001";
elsif(b(2)='1')then
    out_33<="010";
elsif(b(3)='1')then
    out_33<="011";
elsif(b(4)='1')then
    out_33<="100";
elsif(b(5)='1')then
    out_33<="101";
elsif(b(6)='1')then
    out_33<="110";
elsif(b(7)='1')then
    out_33<="111";
end if;
end process;
end arch_ power_2_control_unit;
    
```

IV. COMPARISON

Design equipped to 8bit and 16bits adders, 8bit multiplier and is accomplished to structural description via VHDL hardware description language by using Xilinx ISE V.71 software synthesized and implemented on FPGA in Virtex IV family. Also power is analyzed using Xilinx XPower analyzer. Tables [II, III, IV, V] shows the comparison between power consumption , numbers of LUTs, numbers of Slices and FFs and the type of device that has been used in different articles and the characteristic of our filter designed has been shown in tables VI,VII.

TABLE II: SLICES, LUTS AND FFS COMPARISONS [3].

| Filter(#taps)<br>Virtex II [3] | Slices | LUTs | FFs |
|--------------------------------|--------|------|-----|
| 6                              | 264    | 213  | 509 |
| 10                             | 474    | 406  | 916 |
| 13                             | 386    | 334  | 749 |

TABLE III: SLICES COMPARISONS [3]

| Filter(#taps)<br>Virtex IV [3] | Slices(add method) | shift | Slices(MAC) |
|--------------------------------|--------------------|-------|-------------|
| 6                              | 264                |       | 219         |
| 10                             | 475                |       | 418         |
| 13                             | 387                |       | 462         |

TABLE IV: POWER CONSUMPTION [4]

| Digital filter[4]   | Power consumption(mw) |
|---------------------|-----------------------|
| 16-bits coefficient | 1248                  |
| 8-bits coefficient  | 502                   |
| Optimized bandwidth | 450                   |

TABLE V: POWER CONSUMPTIONS 8 TAPS, 16BITS COEFFICION, 8BITS INPUT [5]

| Freq | Signed array mult (mw) | Booth without DPDT (mw) | Booth with DRD | Booth with DPDT using REG (mw) | Booth with DPDT using and gate (mw) |
|------|------------------------|-------------------------|----------------|--------------------------------|-------------------------------------|
| 25   | 612                    | 469                     | 423            | 326                            | 868                                 |
| 50   | 1170                   | 879                     | 851            | 596                            | 669                                 |
| 75   | 1773                   | 1297                    | 1256           | 881                            | 1018                                |
| 100  | 2293                   | 1703                    | 1658           | 1137                           | 1283                                |

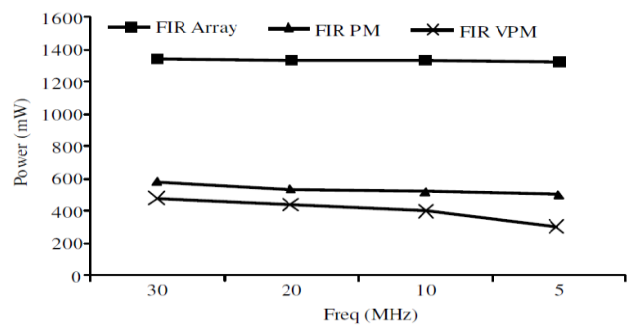


Fig. 7. Power consumption of 8bits and 8 tap FIR filter [6].

TABLE VI: POWER CONSUMPTION PROPOSED FILTER.

|                                |           |           |           |            |
|--------------------------------|-----------|-----------|-----------|------------|
| Proposed filter                | 25<br>MHZ | 50<br>MHZ | 75<br>MHZ | 100M<br>HZ |
| Proposed method (mw)           | 38        | 44        | 50        | 56         |
| Fir_base (8bits,8taps)<br>(mw) | 130       | 190       | 250       | 300        |

TABLE VII: CHARACTERISTIC OF OUR FILTERS DESIGNED.

| Proposed filter           | Slices | FF<br>s | LUT<br>s | Dsp | Lat<br>ch | Frequency<br>(MHZ) |
|---------------------------|--------|---------|----------|-----|-----------|--------------------|
| Proposed method           | Reg 19 | 16      | 51       | 0   | 3         | 386.250            |
| Fir_base<br>(8bits,8taps) | 92     | 64      | 112      | 8   | 0         | 1040.583           |

### V. CONCLUSION

In This paper we presented a low power and low area FIR filter. For reduce power consumption and area we using of shift/add multiplexer based multiplier . This filter was compared for area and power with other common implementations and it demonstrated that our approach is most effective for implementations with the constraints of low cost and low power. The proposed FIR filters have been synthesized and implemented using Xilinx ISE V7.1 and Virtex IV FPGA to target device xc4vlx200 also power is analized using Xilinx XPower analyzer.

### REFERENCES

[1] H. J. G. Chung and K. K. Parhi, "Frequency spectrum based low-area low-power parallel fir filter design," *EURASIP Journal on Applied Signal Processing* 2002, vol. 31, pp. 944-953.

[2] A. F. Shalash and K. K. Parhi, "Power efficient folding of pipelined LMS adaptive filters with applications," *Journal of VLSI Signal Processing*, pp. 199-213, 2000.

[3] S. Mirzaei, A. Hosangadi, and R. Kastner, "FPGA implementation of high speed fir filters using add and shift method," *IEEE*, 2006.

[4] K. Tarumi, A. Hyodo, M. Muroyama, and H. Yasuura, "A design method for a low power digital FIR\_lter indigital wireless communication systems," 2004.

[5] A. Senthilkumar, A. M. Natarajan, and S. Subha, "Design and implementation of low power digital FIR filters relying on data

transition power diminution technique," *DSP Journal*, vol. 8, pp. 21-29, 2008.

[6] A. Senthilkumar and A. M. Natarajan, "FPGA implementation of power aware FIR filter using reduced transition pipelined variable precision gating," *Journal of Computer Science*, pp. 87-94, 2008.

[7] W. Brian and M. Barr, "Introduction to digital filters," *Embedded Systems Programming*, December 2002, pp. 47-48.

[8] M.-B. Uwe, "Digital signal with field programmable gate arrays," *Springer-Verlag Berlin Heidelberg* 2007.

[9] V. A. Pedroni, "Circuit design with VHDL," *MIT Press Cambridge, Massachusetts London, England*, 2004.



**Bahram Rashidi** was born in 1986 in Boroujerd-Lorestan, Iran. He received his B.SC. degree in Electrical Engineering from the Lorestan university, Iran, in 2009 and he received his M.SC. in the Tabriz university, Iran also he is now Ph.D. student in Isfahan University of technology, respectively. His researcterests include digital signal processing, DSP processors, computer vision, modeling with hardware description languages VHDL and VERILOG, He now continues on his interest in digital circuits with research in embedded microprocessor systems and VLSI digital chip design.



**Bahram Rashidi Farshad Mirzaei** received his B.SC. degree in computer engineering from the Science & Technology Sepahan Isfahan University, Iran, in 2009. His research interests include inofmation security, computer network and computer architecture.



**Bahman Rashidi** received his B.SC. degree in Computer Engineering from the Science & Technology Sepahan Isfahan University, Iran, in 2009 and he is now M.SC. in the Iran University of Science and Technology, Tehran, IRAN, respectively. He has accepted and published 2 refereed conference papers. His research interests include Computer Architecture, Computer vision, Distributed System, Cloud Computing.



**Majid Pourormazd** received his B.SC. degree in electronic engineering from the shahid chamran kerman university, Iran, in 2010. He has accepted and published 2 refereed conference papers and also send 1 papers to different journals. His research interests include FPGA, design electronic systems and VLSI design. He is also interested in the field of Digital programming include VHDL and VERILOG.