

## FINITE IMPULSE RESPONSE FILTER POWER REDUCTION THROUGH ARCHITECTURE OPTIMIZATION

C. M. ALBINĂ<sup>1</sup>

*În această lucrare a fost prezentată o metodă standard de implementare a filtrelor digitale cu răspuns finit la impuls (FIR). Avantajele și dezavantajele diferitelor arhitecturi și a diferitelor metode de modelare a circuitului au fost discutate utilizând estimarea numărului de tranziții în interiorul circuitului, simularea la nivel de poartă logică și analiza rezultatelor obținute în urma sintezei modulului descris în limbaj VHDL. Scopul principal a fost ilustrarea faptului că se poate obține o reducere de pînă la 60% a consumului de putere prin alegerea unei arhitecturii optime combinată cu o descriere VHDL corespunzătoare. Analiza a fost efectuată folosind bibliotecile standard pentru o tehnologie CMOS de 150 nm și  $V_{DD}$  1.5 volți.*

*In this paper is presented a digital finite impulse response (FIR) filter using the standard digital design workflow. The advantages and disadvantages of several architectures and of the circuit modeling were discussed using a standard toggle-based method for the circuit power estimation, gate-level simulations and synthesis. The main idea was to show that we can achieve up to 60% power reduction from the beginning by carefully selecting the right architecture and optimizing the VHDL code description of the module. The analysis was made based on the unity delay model and not on the physical extracted layout for a 150 nm CMOS technology and  $V_{DD}$  1.5 volts.*

**Keywords:** FIR filter, synthesys, RTL, architecture, power

### 1. Introduction

Over the years, the state-of-the-art technologies pushed the VLSI chips to higher clock speed and packing density. The trends for the coming years are defined already by the International Technology Roadmap for Semiconductors ITRS. Applications using Digital Signal Processing (DSP) continue to expand, driven by trends such as the increased use of video, audio and still images and the demand for increasingly reconfigurable systems such as Software Defined Radio (SDR). Many of these applications combine the need for significant DSP processing with cost sensitivity, creating demand for high-performance, low-cost DSP solutions.

<sup>1</sup> Dipl.-Ing., Gesellschaft für Mikroelektronik-Entwicklungen mbH, Unterhaching, Germany, IEEE Member, E-mail: cristian.albina@gmail.com

As indicated in Table 1, for most aggressively scaled DRAM, the integration scale will reach 720 Millions of transistors by the year 2012 [1].

Table 1

Selected data from the 2005 ITRS Roadmap

Year	1997	1999	2001	2003	2006	2009	2012
Technology (μm)	0.25	0.18	0.15	0.13	0.10	0.07	0.05
Nr. Of Transistors (Mil.)	11	21	40	76	200	520	720
On Chip CLK (MHz)	750	1200	1400	1600	2000	2500	2800
Wafer Area (mm <sup>2</sup> )	300	340	385	430	520	620	650
Wiring levels	6	6-7	7	7	7-8	8-9	9-10
V <sub>DD,logic</sub> [V]	2.5-1.8	1.8-1.5	1.5-1.2	1.5-1.2	1.2-0.9	0.9-0.6	0.6-0.5
T <sub>ox, equivalent</sub>	4-5nm	3-4nm	2-3nm	2-3nm	1.5-2nm	1.5nm	1nm

While a vast array of digital signal processing functions are implemented by designers, Finite Impulse Response (FIR) filters, Infinite Impulse Response (IIR) filters, Fast Fourier Transforms (FFTs) and mixers are common to many applications. Fast Fourier Transforms are used for a variety of applications, ranging from image compression to determining the spectral content of a data sample. Each of these functions requires a combination of multiply elements along with addition, subtraction and accumulation. Because the power consumption has become an important factor in the design process of a chip due to the limited lifetime of the battery fast and accurate power estimation tools are needed for each level of the circuit in order to ensure that the energy and the design constraints are met. There are already several power estimation tools [2], [3], [4] which can help the designer to this task but the more detailed the simulation is the longer the time and the bigger the amount of data to be simulated. In this paper I'll use the standard power estimation for the CMOS circuits using gate-level and RTL (Register Transfer Level) simulations combined with Synopsys [5] synthesis tool to show the effect of the RTL architecture and of the VHDL description on the final area of the module as well as on the total power, solution which offers the advantage of small designs with lowest power consumption and which are highly correlated to the final layout.

## 2. Digital finite impulse response filter concept

The finite impulse response filter stores a series of n data elements, each delayed by an additional cycle. These data elements are commonly referred to as taps. Each tap is multiplied by a coefficient and the results summed to produce the output. Some implementations perform all the multiplications in parallel. More generally, the implementation is broken down into N stages, with an accumulator passing the partial result from one stage to the next. This implementation trades

speed for functional resources, taking  $N$  computation stages and requiring  $n/N$  multipliers. Depending upon whether the coefficients are static or dynamic and the design of the coefficient values, there are a number of other design optimizations commonly used that are beyond the scope of this article.

A finite impulse response (FIR) digital filter is called 'finite' because its response to an impulse ultimately settles to zero. This is in contrast to infinite impulse response filters which have internal feedback and may continue to respond indefinitely. The FIR transfer function contains  $M$  poles for  $z = 0$ . Since all poles are at the origin, all poles are located within the unit circle of the  $z$ -plane; therefore all FIR filters are stable. This useful property together to the fact that they don't require feedback which can cause rounding errors in the summed iterations and because they're having a linear and minimum phase make sometimes this type of filter preferable to an IIR filter. The direct-form (Fig. 1) and transpose-form (Fig. 2) structures are most commonly used to implement FIR filters.

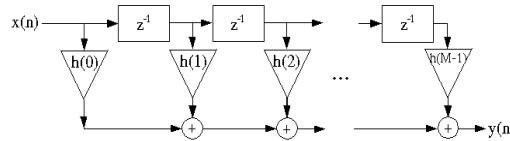


Fig. 1. Direct-form FIR functional diagram

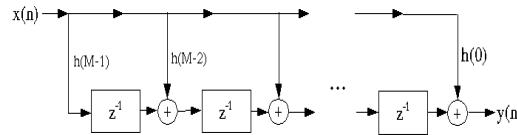


Fig. 2. Transpose-form FIR functional diagram

### 3. Digital FIR implementations using different architectures

To design a filter means to select the coefficients such that the system has specific characteristics. The required characteristics are stated in filter specifications. Most of the time filter specifications refer to the frequency response of the filter. I decided to used in this paper a 29<sup>th</sup> order single channel FIR (30 coefficients) with a system clock frequency of 100 MHz and for the evaluation I used the input and output sample rate equal to 100MHz/32 instead of 30 due to the easy implementation with 5 bits. The filter pass-band is 1 MHz and the stop-band is 1.2 MHz at which the filter attenuation is -60 dB. The whole module was implemented using a 150nm CMOS technology models and  $V_{DD}$  1.5

volts. In each case the synthesis parameters where: clock period 10ns, critical path and area analysis, use enclosed wire load model, and I used clock gating option for the synthesis.

### 3.1 Direct implementation based on the C++ description

In most cases the digital designer starts his implementation based on the direct behavioral description delivered by the system concept engineer. This description is usually written in C++ language and has no hardware related physical parameters included. The filter architecture is illustrated in Fig. 3 and contains a shift register, a coefficient ROM and a multiply and accumulate unit (MAU).

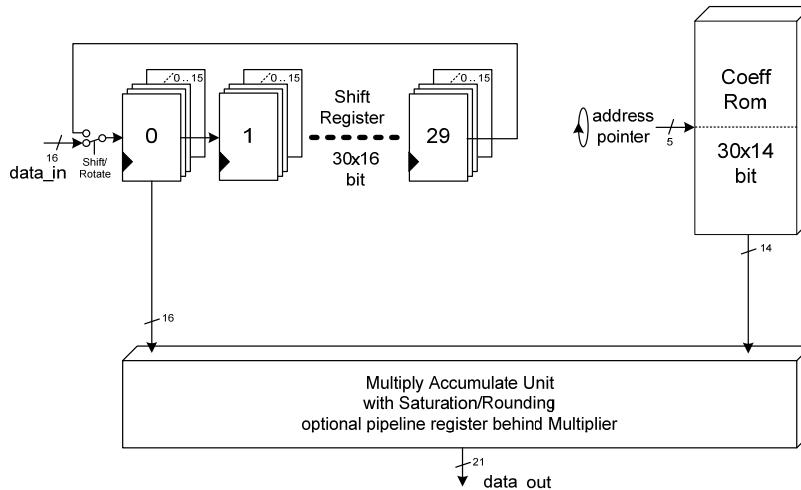


Fig. 3. Digital FIR architecture based on direct C++ code implementation

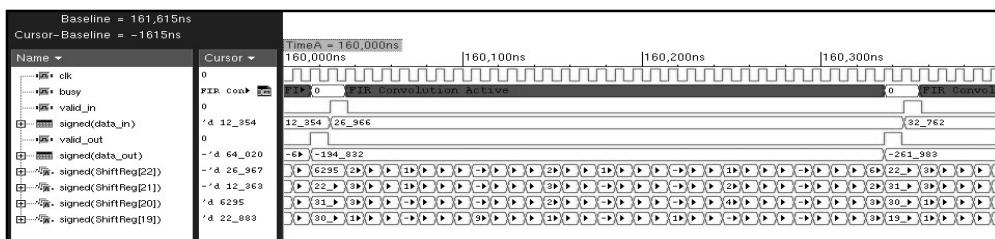


Fig. 4. Digital FIR architecture simulation results

After running the gate-level and the RTL simulations like in Fig. 4 we obtain the following power consumption summary for this straight forward implementation:

Table 2

Standard FIR Architecture Power Consumption

Hierarchy Level	Switch Power (mW)	Internal Power (mW)	Leak Power ( $\mu$ W)	Total Power (mW)	Power %
FIR	6.649	10.61	15.34	17.279	100
Top level FIR logic and ROM	2.810	4.273	2.869	7.594	42.4
Adder inside MAU	0.641	0.309	0.147	1.239	12.5
Multiplier inside MAU	3.186	4.596	3.452	7.785	45.1

### 3.2 Better FIR implementation using a Read Decoder

In order to reduce the module power consumption we'll have to try to minimize the number of cycles necessary to read the filter coefficients out of the ROM memory. One way to achieve this is by carefully choosing the coefficients and their position inside the memory in such a way that the whole process takes only half of the time like in the architecture illustrated in Fig. 5 and by using a read decoder register. The decoder uses the fact that the filter is symmetric, and that two samples can be added before multiplication with the same coefficient. Reducing the number of multiplication by 2 we can reduce the dynamic power consumption significantly.

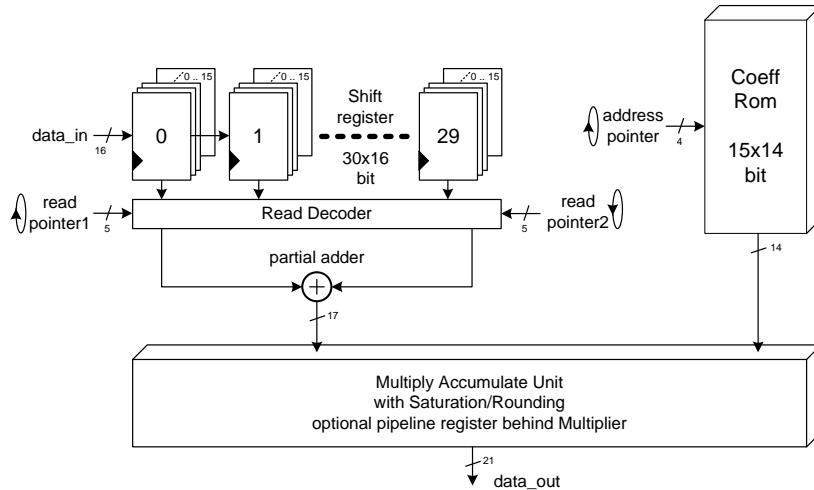


Fig. 5. Improved Digital FIR architecture using a Read Decoder

Redoing the RTL simulations, synthesis and gate-level simulations like in Fig. 6 we obtain the following results for the module power consumption:

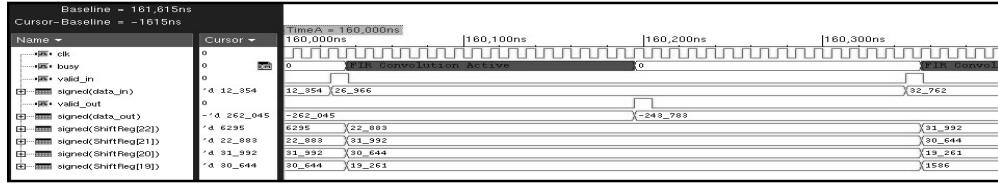


Fig. 6. Improved Digital FIR architecture simulation results

We can notice the reduction of the filter convolution activity on the data bus, second signal in the diagram underneath the clock signal. Furthermore there is an additional switching and internal power reduction inside the MAU (multiplier and adder) and on the top level due to the fact that the implementation needs less digital logic on the top level, most of the operations being already done by the read-decoder.

Table 3  
Improved FIR Architecture Power Consumption

Hierarchy Level	Switch Power (mW)	Internal Power (mW)	Leak Power ( $\mu$ W)	Total Power (mW)	Power %
FIR	3.495	4.543	15.72	8.054	100
Top level FIR logic and ROM	0.63	0.753	0.695	1.368	17.4
Adder inside MAU	$6.47 \times 10^{-2}$	0.242	0.338	0.307	3.8
Partial adder	$6.01 \times 10^{-2}$	0.176	0.414	0.264	2.9
Multiplier inside MAU	2.683	3.365	3.277	6.051	75.1
Read decoder	$5.71 \times 10^{-2}$	$6.48 \times 10^{-3}$	$1.88 \times 10^{-2}$	$6.36 \times 10^{-2}$	0.8

### 3.3 Optimal FIR implementation using a Read/Write decoder

A lot of parallel architectures and additional algorithm improvements were discussed in the past years in the [7]-[10]. For the further power reduction we can use an additional write decoder combined with a special read/write pointer (Circular Buffer) which reduces the toggle activity. Such architecture is proposed in the Fig. 7. In contrast to the previous implementations, the additional Write Decoder selects only one address location of the Circular Buffer and writes the new sample into it. This mechanism avoids the complete rotation of the registers like it happens in the previous shift register implementations and therefore the amount of toggles gets reduced. Once the new incoming data is stored, the write

pointer is incremented and points now to the next address location of the Circular Buffer.

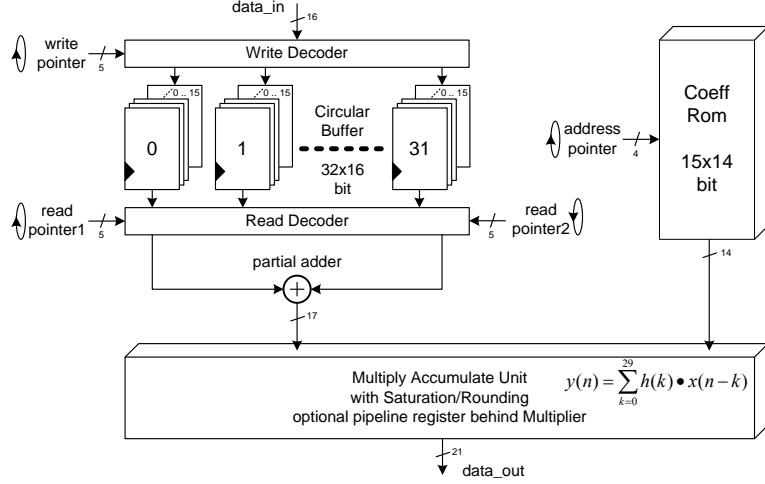


Fig. 7. Optimal Digital FIR architecture with additional Write Decoder

The VHDL code was implemented using the diagram from Fig. 8.

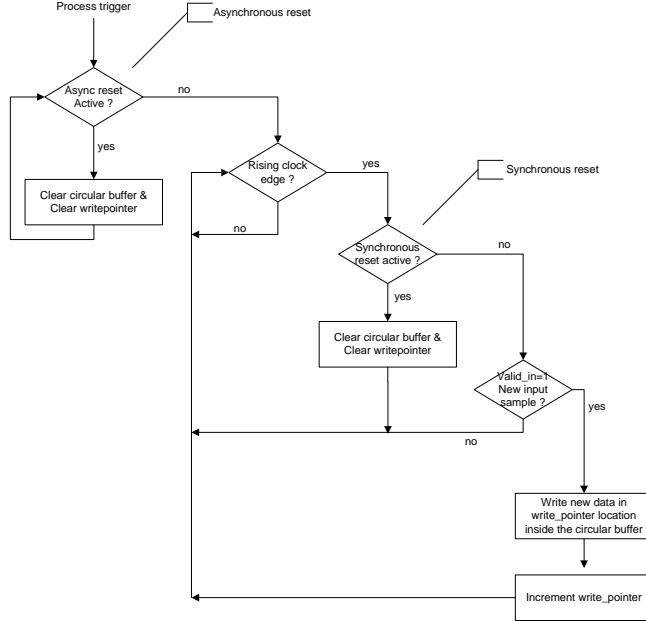


Fig. 8. Optimal Digital FIR implementation flowchart

To implement the shift register and the circular buffer with read/write pointers I used the following VHDL source code:

```
*****
***          FIR Shift Register improved      ***
*****
process (clk, res)
begin
  if (res = '0') then -- asynchronous reset
    ShiftReg <= (others => (others => '0'));
  elsif
    (clk'event and clk = '1') then
      if (syncres = '1') then -- synchronous reset
        ShiftReg <= (others => (others => '0'));
      else
        if (valid_in='1') then -- shift in new data, clock gating enable
          for i in 29 downto 1 loop
            ShiftReg(i) <= ShiftReg(i-1);
          end loop;
          ShiftReg(0) <= data_in;
        end if;
      end if;
    end if;
  end process;

*****
***          FIR write pointer for circular buffer      ***
*****
process (clk, res)
begin
  if (res = '0') then -- asynchronous reset
    CircBuffer <= (others => (others => '0'));
    writepointer <= (others => '0');
  elsif
    (clk'event and clk = '1') then
      if (syncres = '1') then -- synchronous reset
        CircBuffer <= (others => (others => '0'));
        writepointer <= (others => '0');
      else
        if (valid_in='1') then -- write data, clock gating enable
          CircBuffer(to_integer(writepointer)) <= data_in;
          writepointer <= writepointer + 1;
        end if;
      end if;
    end if;
  end process;
```

Based on this flowchart I decided to use the following architecture presented in Fig. 9 for the final implementation of a dual channel digital FIR (I and Q) to insure that I can achieve the minimum power consumption required for the implementation of the module inside a mobile device and also the optimal area inside the chip.

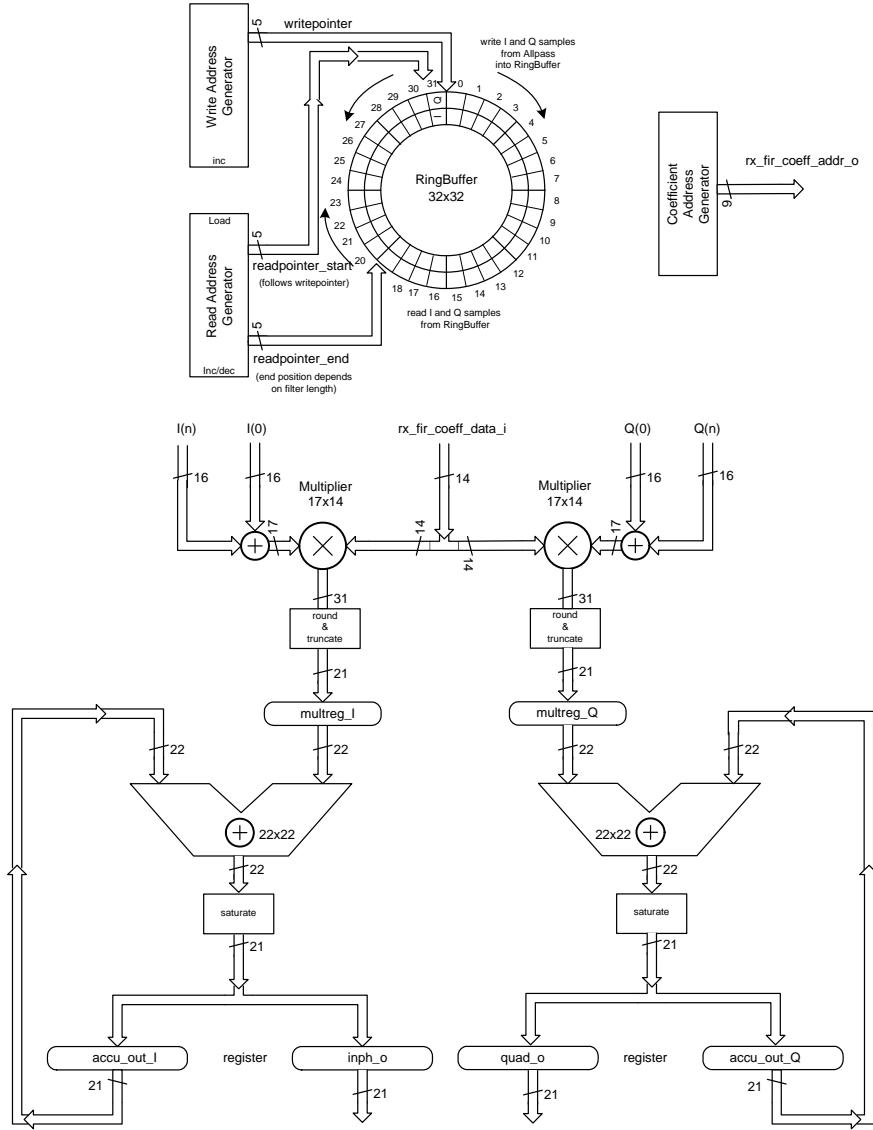


Fig. 9. Digital dual channel FIR final architecture with circular buffer and parallel pipe-lines

Based on the simulation results from the Fig. 10 we've got the following results for the module power consumption:

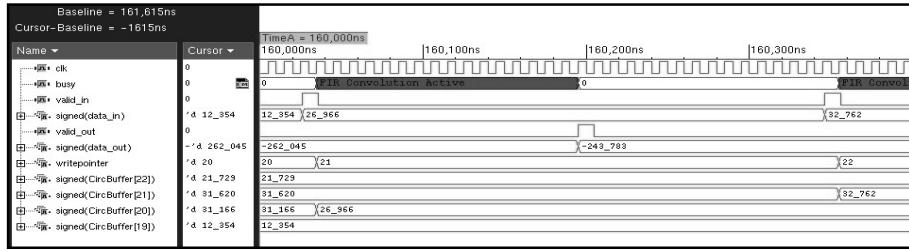


Fig. 10. Optimal Digital FIR simulation results

Table 4

## Optimal FIR Architecture Power Consumption

Hierarchy Level	Switch Power (mW)	Internal Power (mW)	Leak Power (µW)	Total Power (mW)	Power %
FIR	3.007	4.189	17.69	7.214	100
Top level FIR logic and ROM	0.538	0.673	1.358	1.225	17.9
Adder inside MAU	$4.17 \times 10^{-2}$	0.153	0.253	0.195	2.7
Multiplier inside MAU	2.293	3.143	3.277	5.44	75.4
Circular Buffer read decoder	$2.03 \times 10^{-2}$	$4.06 \times 10^{-3}$	$9.78 \times 10^{-3}$	$2.43 \times 10^{-2}$	0.3
Circular Buffer write decoder	$1.49 \times 10^{-3}$	$2.13 \times 10^{-3}$	$9.78 \times 10^{-3}$	$3.63 \times 10^{-3}$	0.1
Circular Buffer 5 bit inc/dec	$6.83 \times 10^{-3}$	$2.59 \times 10^{-2}$	$5.25 \times 10^{-2}$	$3.28 \times 10^{-2}$	0.5
Read decoder	$4.64 \times 10^{-2}$	$5.48 \times 10^{-3}$	$1.41 \times 10^{-2}$	$5.19 \times 10^{-2}$	0.7
Partial adder	$4.87 \times 10^{-2}$	0.126	0.194	0.175	2.4

## 4. Conclusions

Summarizing the three implementation methods we can see that by careful architecture and code implementation we can significantly reduce the total module power consumption up to 60% without a big increase of the circuit area, just by reducing the amount of toggles. For the presented architectures I used SR (shift register) fully (32bit) or half (16 bit) addressed, CG (clock gating), CB (circular buffer) and parallel processing of the coefficients using pipe-line structures. This filter was already implemented using a 150 nm CMOS technology and the measurements done on the final silicon confirmed the simulation results for the main filter parameters.

Table 5

## Synthesis Results Comparison

Architecture	Module Area (mm <sup>2</sup> )	Toggle Count (Mils Tc)	Used Registers	Clock gate efficiency	Power (mW)	Details
FIR1	0.050	64.2	568	0.0%	17.279	SR, full
FIR2	0.054	33.7	589	90.3%	8.054	SR, half, pipe, CG
FIR3	0.057	30.2	604	99.5%	7.214	CB, half, pipe, CG

If we introduced the simulated values of the VLSI schematic of the implemented filters obtained during synthesis and gate-level simulations in the Matlab waveform viewer we can visualize the filter frequency response to an in-band signal as well as the rejection of an off-band signal as presented in Fig. 11.

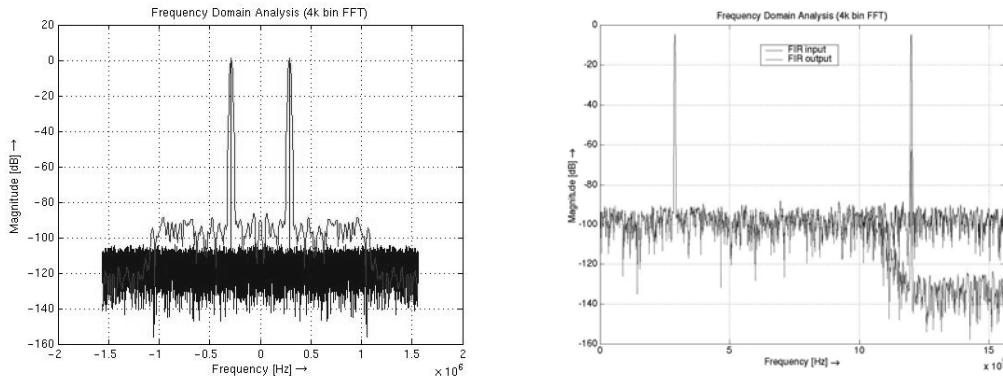


Fig. 11. Digital FIR frequency domain analysis of an in-band and off-band signal

There is a lot of work going around trying to find even better architectures and better coefficient quantization methods to improve furthermore the overall power consumption combined in parallel with the usage of modern submicron technologies. The general tendency is towards low power modules combined with intelligent algorithms for easy VLSI implementation.

## R E F E R E N C E S

- [1] International Technology Roadmap for Semiconductors, 2005 Edition, <http://www.itrs.net/Links/2005ITRS/Home2005.html>.
- [2] Design Power Tools, <http://www.synopsys.com/products/solutions/galaxy/power/power.html>.
- [3] *M. Karunaratne, C. Ranasinghe, A. Sagahyroon*, “A dynamic switching activity generation technique for power analysis of electronic circuits,” Circuit and Systems, 2005, 48<sup>th</sup> Midwest Symposium on, Page(s) 1884-1887 **Vol.2**, 7-10Aug. 2005.
- [4] *A. Raghunathan, S. Dey, N. K. Jha*, “High-level macro-modeling and estimation techniques for switching activity and power consumption”, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, **Vol. 11**, Issue 4, Aug. 2003 Page(s):538 – 557.
- [5] Synopsys, Power Compiler Reference Manual, <http://www.synopsys.com/products/power/index.aspx/>.
- [6] Cadence, NC-VHDL Simulator, [http://www.cadence.com/products/functional\\_ver/nc-vhdl/index.aspx/](http://www.cadence.com/products/functional_ver/nc-vhdl/index.aspx/).
- [7] *P. K. Merakos, K. Masselos, K. Stouraitis*, “Optimization techniques for reducing global bus switching activity in realizations of sum-of-products computations in DSP systems,” Circuits, Devices and Systems, IEEE Proceedings-**Vol. 150**, Issue 1, Feb. 2003 Page(s): 16-25.
- [8] *Jen-Ming Wu; Yang-Chun Fan*, “Coefficient Ordering Based Pipelined FFT/IFFT with Minimum Switching Activity for Low Power WiMAX Communication System” Consumer Electronics, IEEE Tenth International Symposium on, ISCE ‘06, 2006.
- [9] *Jongsun Parl et al.*, “High performance and low power FIR filter design based on sharing multiplication”, Low Power Electronics and Design, Proceedings of the 2002 International Symposium on, ISLPED '02, 2002 Page(s):295 – 300.
- [10] *C. Cheng, K.K. Parhi*, “ Low- Cost Parallel FIR Filter Structures With 2-Stage Parallelism”, Circuits and Systems I: Regular Papers, IEEE Transactions on [Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on], **Vol. 54**, Issue 2, Feb. 2007, Page(s): 280-290.