WP116 (v1.0) April 5, 2000

Xilinx Spartan-II FIR Filter Solution

Author: Antolin Agatep

Introduction

Traditionally, digital signal processing (DSP) algorithms are implemented using general-purpose programmable DSP chips for low-rate applications. Alternatively, special-purpose, fixed function DSP chipsets and application-specific integrated circuits (ASICs) are used for high-performance applications. Technological advancements by Xilinx in Field Programmable Gate Arrays (FPGAs) in the past 15 years have opened new paths for DSP design engineers. The FPGA maintains the high specificity of the ASIC while avoiding its high development cost and its inability to accommodate design modifications after production. Highly adaptable and design-flexible, FPGAs provide optimal device utilization through conservation of board space and system power—important advantages not available with many stand-alone DSP chips.

The fast pace of twenty-first century computing demands solutions that target aggressive time-to-market windows of opportunities. Spartan™-II is effectively taking over many traditional ASSP markets. Unlike typical ASSP solutions, Xilinx Spartan-II allows designers to respond quickly and economically to the emergence of new, more efficient IP core algorithms.

Xilinx Spartan-II FPGAs provide digital designers with a potentially unlimited array of highly reconfigurable solutions. When the design demands more than 100 MIPS, when time-to-market is critical, or when design adaptability is crucial, Spartan-II is the best solution. The most common digital building blocks, like serial peripherals, DMA controllers, PCI controllers, and synthesizable processors, are all readily realizable using a Spartan-II device. In fact, all the most basic operations performed by analog or digital electronic devices—filtering, amplification, modulation, storage, and computation—can be implemented with Spartan-II. Figure 1 shows how Spartan-II can do all these things as a digital signal processor.

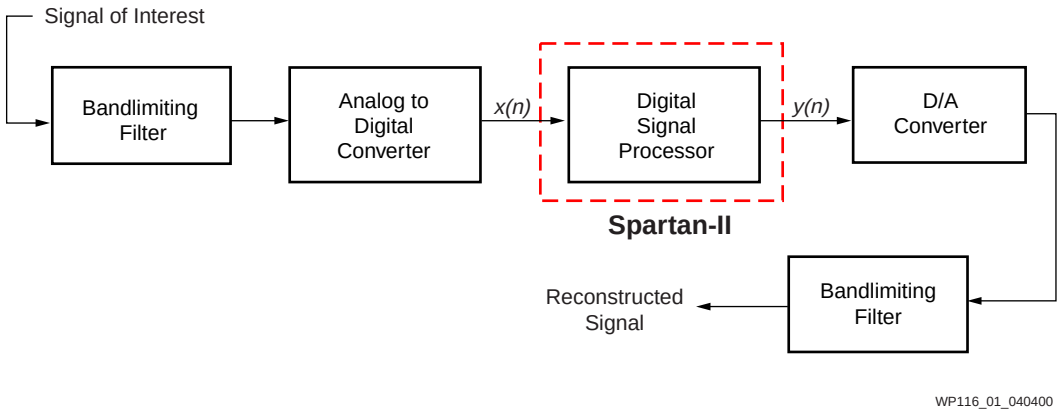


Figure 1: **Digital Signal Processing Blocks**

Advantages of the Xilinx FIR Filter Solution

FIR filterx are used in every aspect of present-day technology because filtering is one of the basic tools of information acquisition and manipulation. Xilinx FIR filter solutions have many advantages over traditional DSP chips:

- High-performance finite impulse response (FIR), half-band, Hilbert transform and interpolated filters
- Highly parameterizable

- 2-to-256 tap symmetrical impulse response
- 2-to-128 tap non-symmetrical impulse response
- 1-to-32-bit input data precision
- signed or unsigned input data
- 1-to-32-bit coefficient precision
- 1-to-8 channels
- Coefficient symmetry exploited (symmetric/negative-symmetric) to produce compact implementations
- Data-flow style core interface and control
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement technology
- Incorporates Xilinx Smart-IP technology for maximum performance
- Use version 2.1i or later of the Xilinx CORE Generator System

This paper focuses on one of the various signal-processing tasks at which Spartan-II excels, FIR filtering. Finite impulse response (FIR) filters have two very attractive and useful characteristics, phase linearity and stability. Compared to FIR filters, infinite impulse response (IIR) filters are not linear at the fringes, and are not always stable.

FIR filters can be expressed in both equation and graphical format. Figure 2 shows two equations. The top equation (a) is expressed as a difference equation, while the bottom equation (b) is expressed as a Z-transform equation. The FIR filter direct form (transversal) structure is shown in Figure 3.

$$\textbf{(a)} \quad y(n) = \sum_{k=0}^{N-1} h(k)\, x\,(n-k)$$

$$\textbf{(b)} \quad H(z) = \sum_{k=0}^{N-1} h(k)\, z^{-k}$$

Figure 2: **Filter Equations: (a) Convolution Sum Difference Equation (b) Z-Transform**

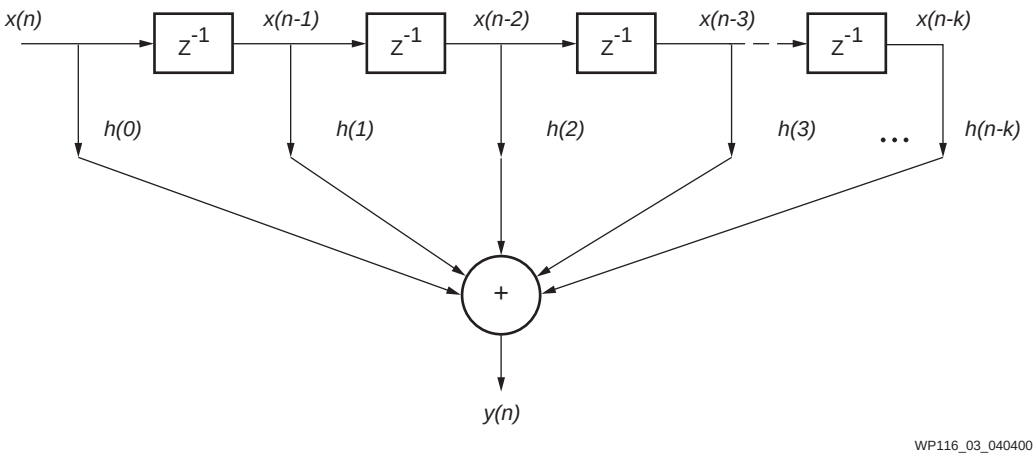


Figure 3: **FIR Direct Form or Transversal Form**

There are generally five steps in the design of a digital filter:

1. Specify the filter requirements.
2. Calculate the filter coefficients.
3. Use a suitable structure to represent the filter.
4. Analyze the effects of finite word length on the filter's performance.
5. Implement the filter in software and/or hardware.

Steps 2, 3, and 4 are usually grouped together when using automated tools.

Figure 4 shows that the filter coefficients basically correspond to the impulse response of the filter being characterized. Theoretically, the impulse response of a filter is obtained by watching the output after applying an impulse at the filter's input. Figure 4 also shows how the FIR filter fits into the scheme of digital signal processing. There can be several FIR filters used in one application.

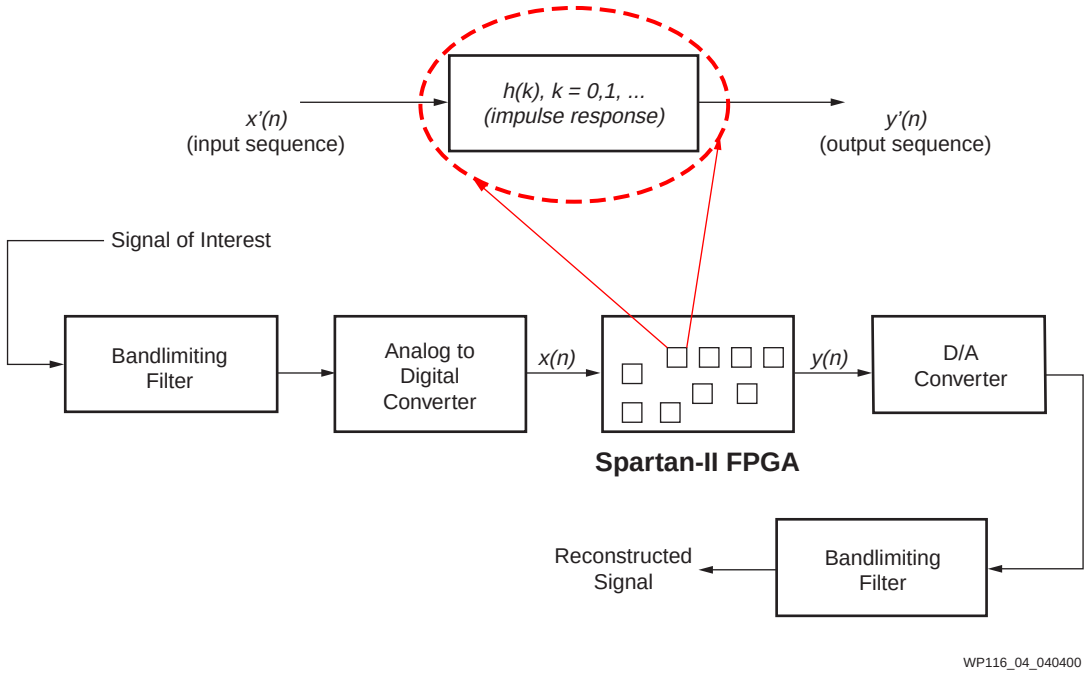


Figure 4: **FIR Filter(s) in an Application**

Figure 5 shows the end-user-supplied specifications for an example filter, a low-pass filter in this case. The specifications for band-pass and high-pass filters are very similar. Basically, one specifies the passband frequency (**fp**), transition width **(fs** – **fp**), stopband attenuation (**ds**), and the sampling frequency. The |**H(f)**| axis represents the magnitude of the signal being filtered and the **f** axis is the frequency.

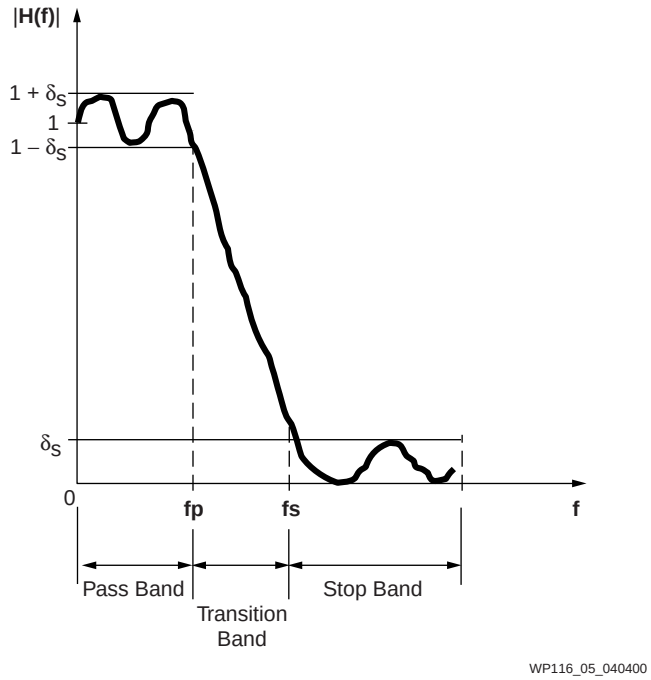


Figure 5: **Example Showing Filter Requirements for a Low Pass Filter**

As an exercise in FIR filtering, the window method will be used. Although it is not as efficient as the optimal method, which generally gives s smaller number of coefficients than other methods, the window method will suffice.

Example of FIR Filter Coefficient Calculation

Calculate the coefficients of a low pass FIR filter given the following specifications and using the window method: [3]

Passband edge frequency	2 kHz
Transition width	0.5 kHz
Stopband attenuation	> 50 dB
Sampling frequency	5 kHz

The impulse response of an ideal low-pass filter is given by:

$$h_D(n) = 2f_c[\sin(n\omega c)/ n\omega c] \quad \text{for } n \neq 0,$$

$$h_D(n) = 2f_c \quad \text{for } n = 0.$$

Note that $\mathbf{f_c}$ is the cutoff frequency and $\mathbf{h_D(n)}$ is the impulse response of an ideal low-pass filter. Also, the ideal low-pass filter, as well as band-pass and high-pass filters, needs to be multiplied by a window function which serves to bridge the gap in the realization of the filter. The ideal low-pass filter by itself is not a very good filter because it exhibits the Gibbs Phenomenon, which appears when the Fourier approximation of the filter is truncated. Of course, this phenomenon will go away when one uses an infinite number of waveforms or when one truncates to a large enough number of waveforms.

Therefore, $\Delta f = 0.5/5 = 0.1$, and $N = 3.3/\Delta f = 33$(number of FIR coefficients).

From the specifications, the Hamming window is chosen for simplicity and Table 1 is a summary of its characteristics[3].

Table 1: **Hamming Characteristics Summary**

Transition Width	Passband Ripple (dB)	Main Lobe Relative to Side Lobe (dB)	Stopband Attenuation (dB) (maximum)	Window Function w(n), w(n) [(N-1)/2]
3.3/N	0.0194	41	53	0.54 + 0.46 cos(2πn/N)

The FIR filter coefficients, **h(n)**, are given by:

$$h_D(n)w(n) \qquad -16 \le n \le 16,$$

where:

$$h_D(n) = 2f_c[\sin(n\omega c)/ n\omega c] \qquad \text{for } n \neq 0,$$

$$h_D(n) = 2f_c \qquad \text{for } n = 0.$$

$$w(n) = 0.54 + 0.46 \cos(2\pi n/33) \qquad -16 \le n \le 16$$

Note that $\mathbf{h_D(n)}$ is the ideal impulse response of the low-pass filter.

The window will tend to smear the filter response. To account for this, the cutoff frequency, $\mathbf{2f_c}$, will be adjusted to the center of the transition band. Now, the new cutoff frequency, $\mathbf{f_c'}$, is given by:

$$f_c' = f_c + \Delta f/2 = (2.0 + 0.25)\ \text{kHz} = 2.25\ \text{kHz} = 2.25/5 = 0.45$$

Noticing that **h(n)** is symmetric, the values for **h(0)** through **h(16)** need only be calculated.

Table 2: **FIR Coefficients Arranged in a Table Format**

Coefficient Values		
h(0)	0.9	h(0)
h(-1)	0.09755	h(1)
h(-2)	-0.09047	h(2)
h(-3)	0.07957	h(3)
h(-4)	-0.06606	h(4)
h(-5)	0.05136	h(5)
h(-6)	0.03689	h(6)
h(-7)	0.02399	h(7)
h(-8)	-0.01314	h(8)
h(-9)	0.00519	h(9)
h(-10)	0	h(10)
h(-11)	0.00277	h(11)
h(-12)	0.00372	h(12)
h(-13)	-0.00353	h(13)
h(-14)	0.00284	h(14)
h(-15)	-0.00209	h(15)
h(-16)	0.00155	h(16)

Figure 7 shows the impulse response plot of the desired low-pass filter. Once the impulse response of a filter is known, *everything* about that filter's characteristics is known. Just use the difference equation of Figure 2(a) to calculate the resulting output sequence from an input sequence. The window method is straightforward to implement, but the optimal method can provide a smaller number of filter coefficients by which to characterize the filter of interest.

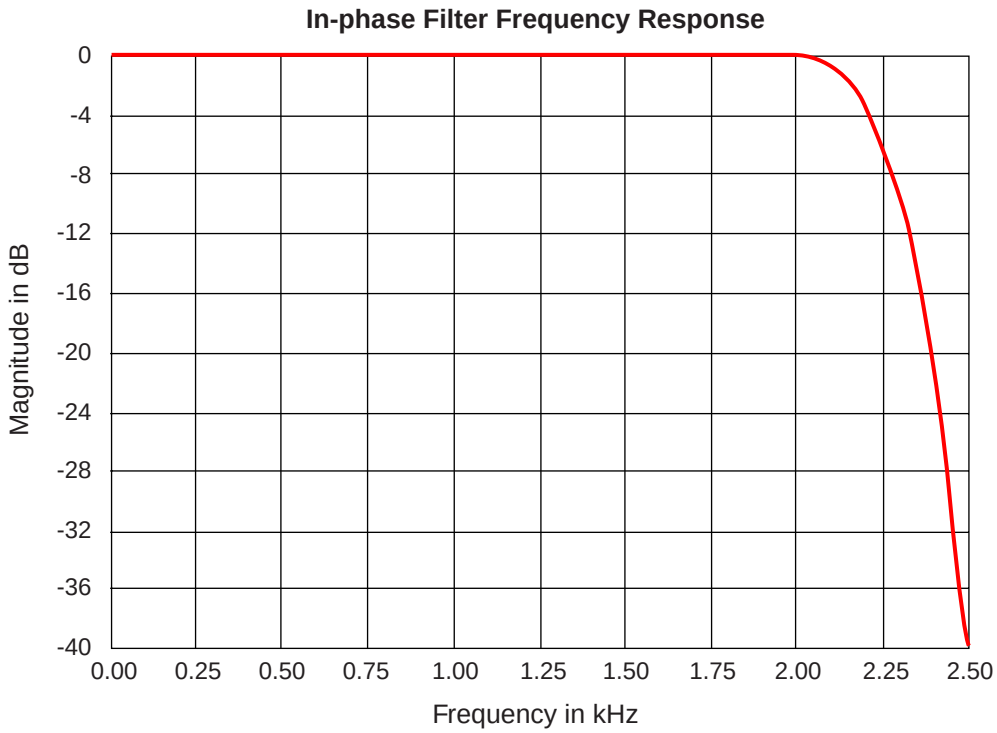


Figure 6: **Magnitude Plot of the Calculated Low-Pass Filter Using Windowing**

[

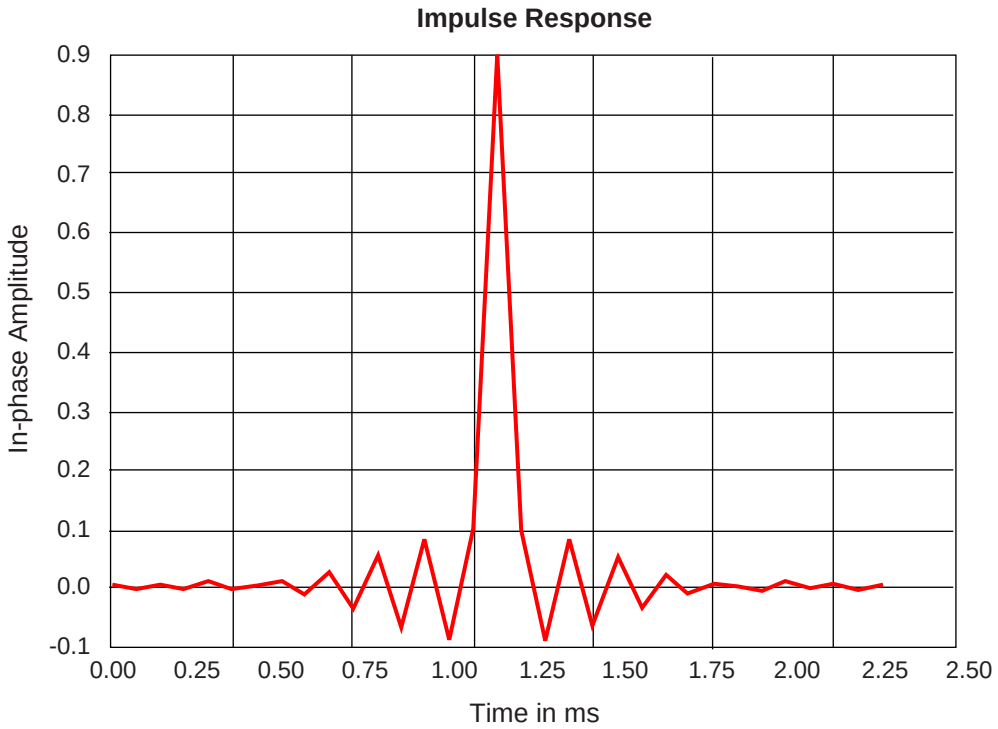


Figure 7: **Impulse Response Plot of the Calculated Low-Pass Filter Using Windowing**

Real-World Applications of FIR Filters and Adaptive FIR filters

A few popular applications for FIR filters are listed below:

- Echo cancellation
  - Telecommunications
  - Data communications
  - Wireless communications
    - Multi-path delay compensation
- Ghosting cancellation
  - HDTV
  - DTV
  - Video processing
- Speech synthesis
- Waveform synthesis
- Filtering
  - High-speed modems
    - ADSL
    - SDSL
    - IDSL
    - VDSL
    - ISDN
- Image enhancement
  - HDTV
  - DTV
  - Video Processing

- Digital Cameras
- Digital Video Camcorders

- Special effects
  - Reverberation/echo effect
  - Video, audio
- Wireless/satellite communications security
  - Spread-spectrum jamming compensation
- Biomedical signal processing
  - Compensation of EOG contamination of EEG reading
  - De-emphasis of maternal ECG to easily observe fetal ECG

Adaptive Digital Filters

There are instances when the condition that a filter monitors changes. It is advantageous for the filter to adapt to the changes in order to extract the desired signal. This brings about the term “adaptive” digital filter. In the area of adaptive digital filtering, most of the filters are realized using FIR filters because of their stability and simplicity.

The adaptive noise cancellation circuit shown in Figure 8 can be used in areas like bio-medical signal processing. A case in point is the merging of the heartbeat of the pregnant mother with that of the fainter heartbeat of the fetus. The adaptive noise cancellation technique can be used to filter out the mother’s stronger heartbeat to make the fetal heartbeat more readily observable.

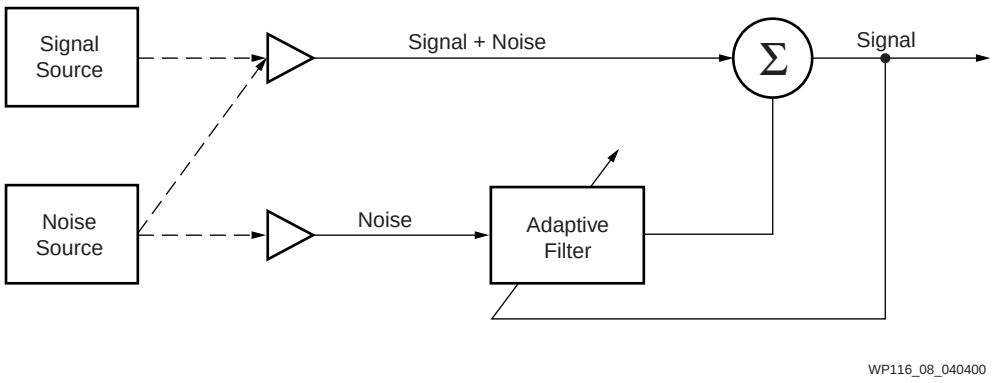


Figure 8: **Adaptive Noise Canceling Filter**

As Figure 9 clearly shows, the collected signals from the chest and abdominal electrodes of the mother are adaptively filtered so that the mother's ECG signature is made less pronounced and the fetus' ECG signature is easier to recognize and analyze.

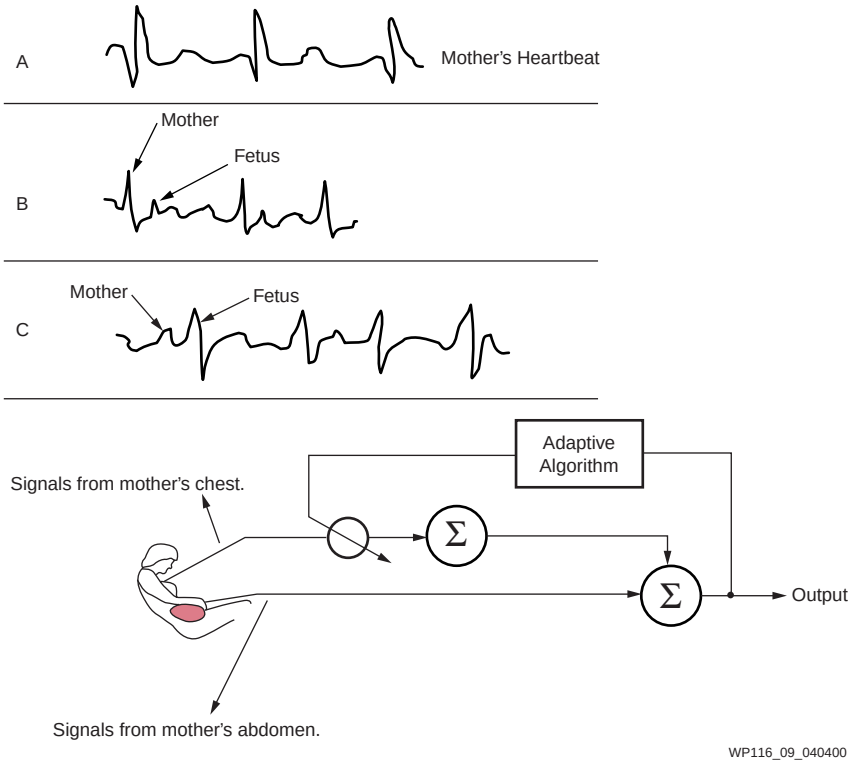


Figure 9: **Adaptive Cancellation of Maternal ECG from Fetal ECG[3]**

Adaptive filtering is also found in long distance telephony (Figure 10), in spread spectrum communications (wireless, satellite, etc), and in radar signal processing to remove signal clutter and noise components.

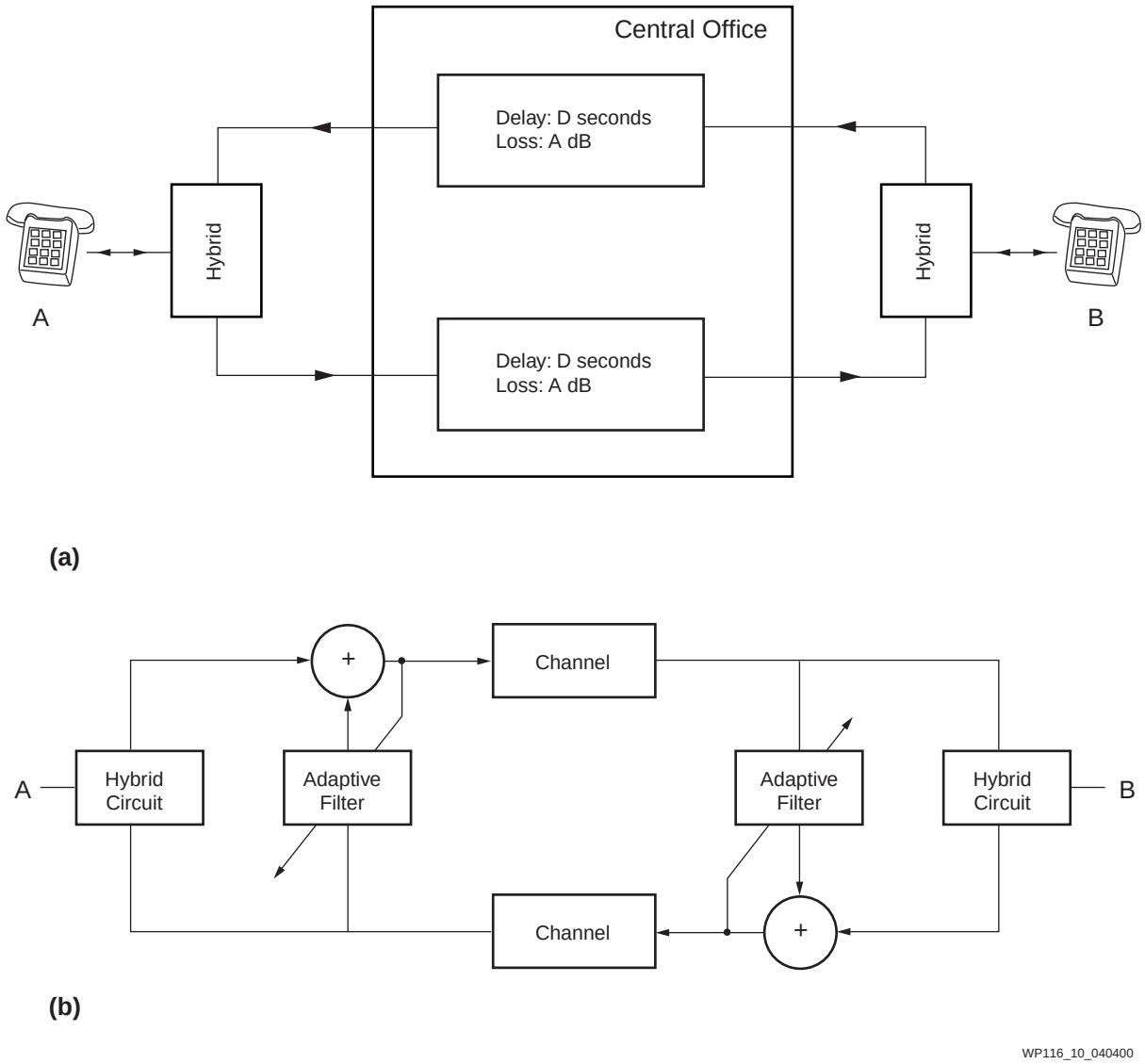


Figure 10: **Echo Cancellation in Telephony – (a) Telephone Network (b) Adaptive Circuitry**

Also, as Figure 11 shows, using Spartan-II enables one to quickly create products in areas where standards are still evolving, such as in the areas of video and image processing.

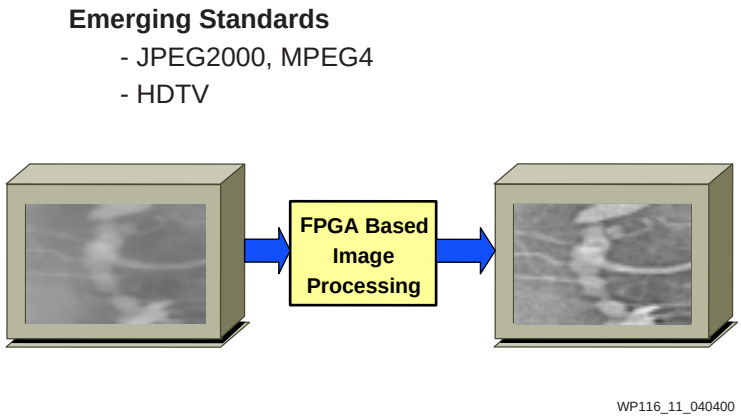


Figure 11: **Spartan-II in Products with Evolving Standards**

The real strength of the Spartan-II family of FPGAs is that FIR filtering, as well as digital signal processing tasks in general, can be implemented in a variety of ways without being hindered by DSP CPU architecture (Figure 12). This means that the designer can take full advantage of the FPGAs programmable fabric to fit the requirements of any application.

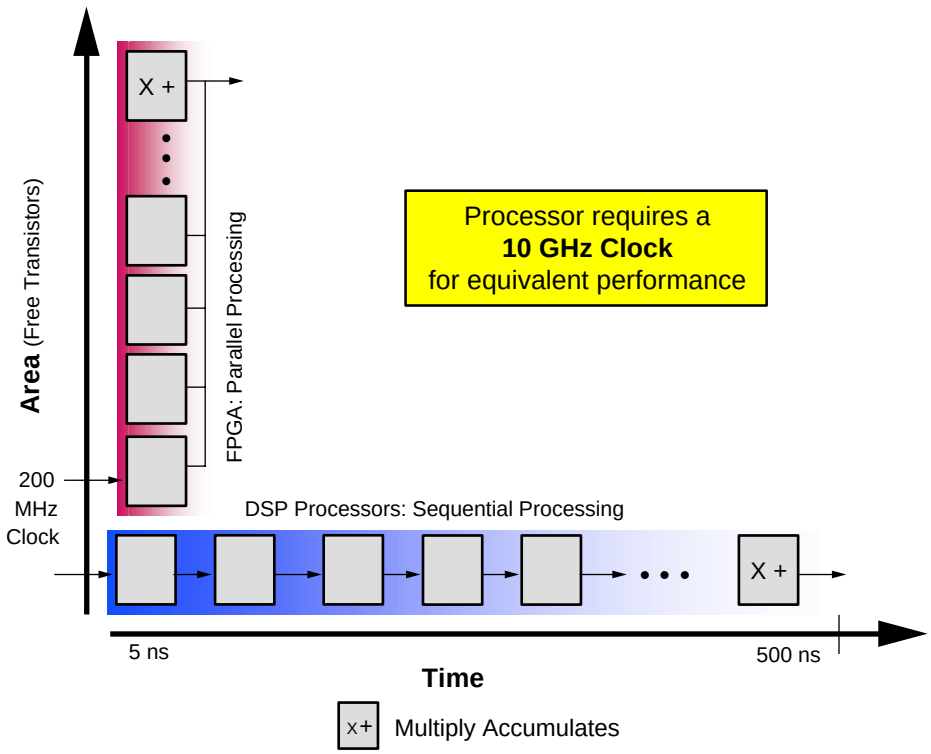


Figure 12: **FPGA vs. DSP Architecture – Solution Implementation**

The SRAM-based architecture of the Xilinx FPGA is well suited for Multiply and Accumulate (MAC)-intensive DSP functions, as Figure 12 and Figure 13 show.

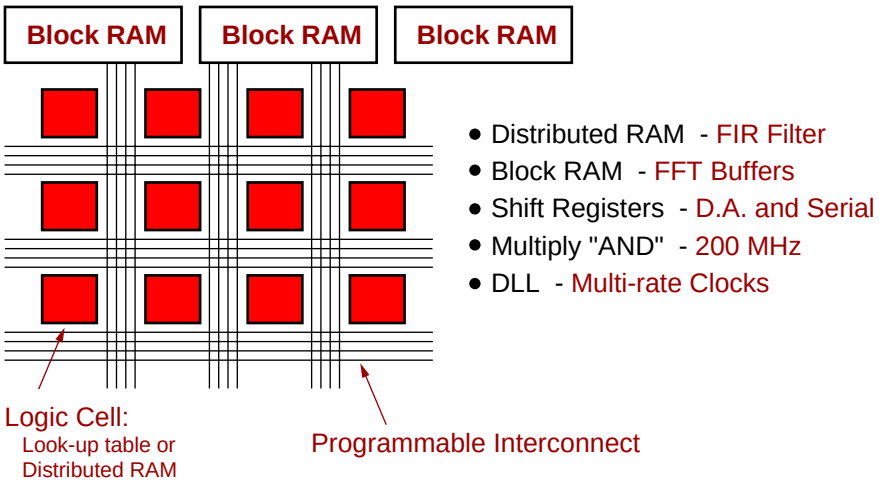


Figure 13: **Spartan-II Architecture**

A wide range of arithmetic functions for fast Fourier transforms (FFTs), convolution and filtering algorithms, as well as the surrounding peripheral circuitry can be implemented into the Xilinx FPGA and reconfigured on-the-fly. When designing a DSP system in an FPGA, the data can be processed taking advantage of single-chip, paralleled structures (Figure 12 and Figure 13) and arithmetic algorithms to exceed the performance of a single general-purpose DSP device. Distributed Arithmetic[2] used for array multiplication is just one of the approaches used to increase data bandwidth and throughput by as much as several order of magnitudes in FPGAs.

One example is a 16-tap, 8-bit FIR filter. It can support more than 780 MIPS at 5 million samples per second while occupying less than 1050 gates, or 70% of the 50 CLBs in the smallest member of the Spartan-II family, the XC2S15. This device has a total of 96 CLBs and runs at 200 MHz with a 2.5V supply. Table 3 below shows all the Spartan-II family members.

Table 3: **Spartan-II Family – 200 MHz at 2.5V Operation**

Device	Logic Cells	Total Gates	Total CLBs	BlockRAM Bits	Max I/O
XC2S15	432	6,000 - 15,000	96	16,384	86
XC2S30	972	13,000 - 30,000	216	24,576	132
XC2S50	1,728	23,000 - 50,000	384	32,768	176
XC2S100	2,700	37,000 - 100,000	600	40,960	196
XC2S150	3,888	52,000 - 150,000	864	49,152	260
XC2S200	5,292	71,000 - 200,000	1,176	57,344	284

Differing from conventional data processing, DSP operations lend themselves to highly pipelined parallel execution, and the Spartan-II architecture, with its thousands of look-up tables and flip-flops, is the ideal platform for this. Compared to general-purpose DSP processors, Xilinx FPGAs achieve identical results much more quickly, and at a lower cost. Compared to dedicated ASIC solutions, Xilinx FPGAs offer faster time-to-market and less risk.

And Xilinx has software tools, as Figure 14 shows, to help automate this method of implementing familiar DSP algorithms.

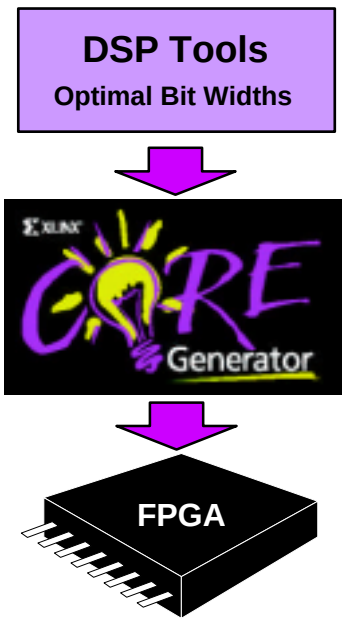


Figure 14: **Xilinx Software Tool: CORE Generator**

Xilinx' DSP LogiCORE products are complex, parameterized DSP building blocks that deliver performance and density equal to or better then a hand-tuned implementation. LogiCORE DSP modules can be used with VHDL, Verilog, or schematic-capture design methodologies. Higher level DSP cores are available from Xilinx AllianceCORE partners.

Figure 15 and Figure 16 show the Xilinx CORE Generator GUI.

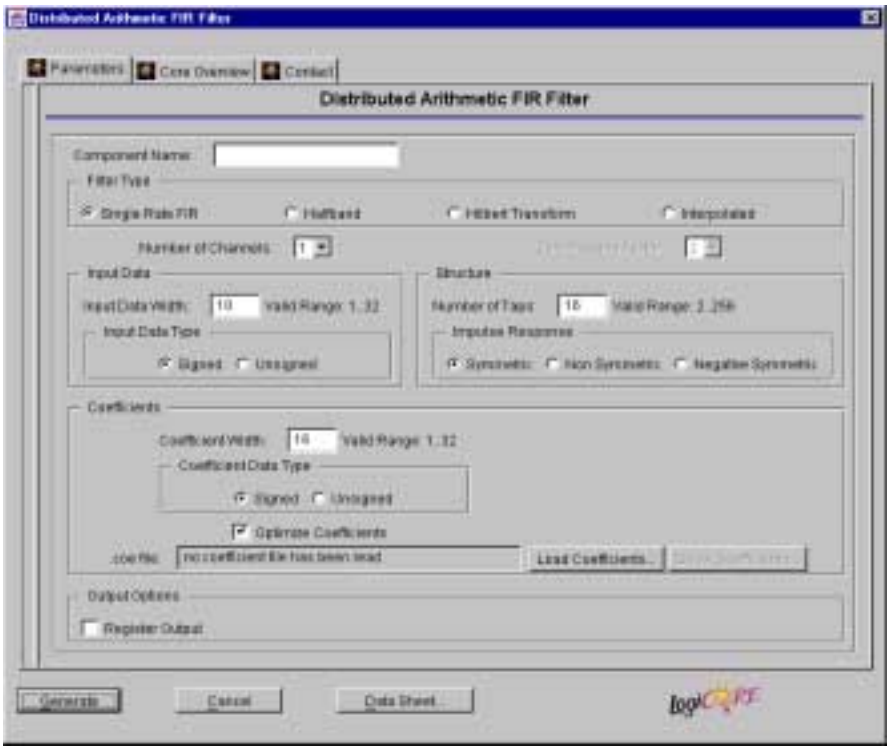


Figure 15: **Xilinx CORE Generator – General Menu**

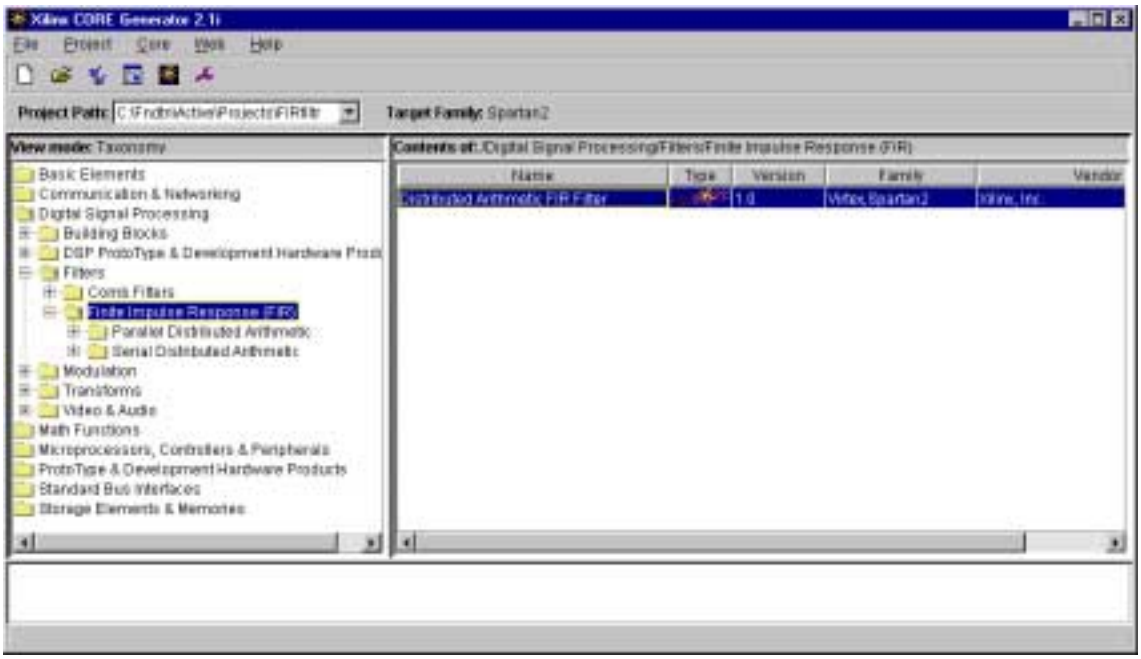


Figure 16: **Xilinx CORE Generator – FIR Parameters**

Spartan-II Summary

Flexibility of DSP Processors

- Off-the-shelf devices
- Faster time-to-market
- Rapid adoption of new standards
- Real-time prototyping

Performance of Custom ICs

- Parallel processing
- Support high data rates
- Optimal bit widths
- No real-time software coding

Advantages of the Xilinx FIR Filter Solution

- High-performance finite impulse response (FIR), half-band, Hilbert transform and interpolated filters
- Highly parameterizable
- 2-to-256 tap symmetrical impulse response
- 2-to-128 tap non-symmetrical impulse response
- 1-to-32-bit input data precision
- signed or unsigned input data
- 1-to-32-bit coefficient precision
- 1-to-8 channels
- Coefficient symmetry exploited (symmetric/negative-symmetric) to produce compact implementations
- Data-flow style core interface and control
- High performance and density guaranteed through Relational Placed Macro (RPM) mapping and placement and Smart-IP technology for max performance

As Figure 17 shows, Xilinx' solution, takes advantage of the entire spectrum of FIR filter implementations which give cost-effective and high-performance solutions both for FIR filtering and general purpose digital signal processing.

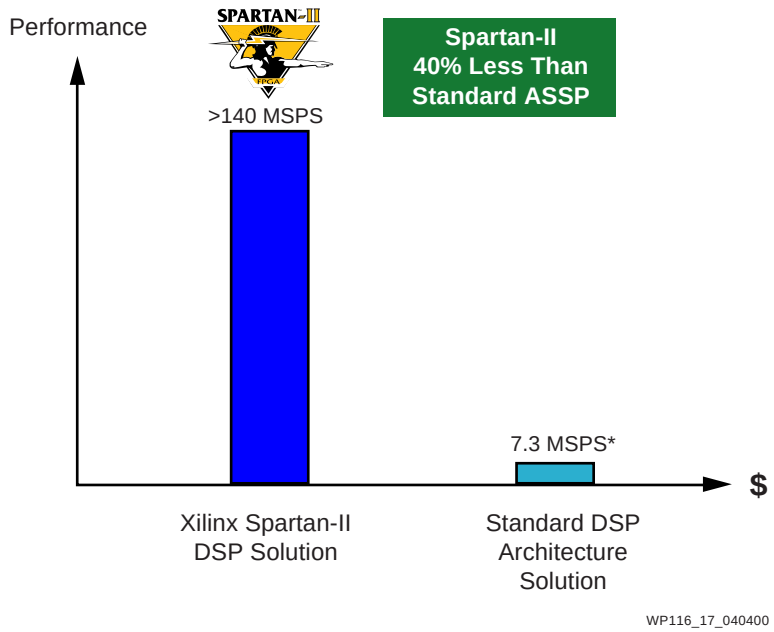


Figure 17: **Spartan-II Advantage over DSP Implementation of FIR Filter**

Notes:
1. 604 cycles for optimized 16-tap FIR in a 1.1 GHz DSP.
2. FIR filter realized in FPGA using fully Parallel Distributed Arithmetic.

References

1. *Implementing Area-Optimized Narrow-Band FIR Filters Using Xilinx FPGAs by Chris Dick*
2. *Using Xilinx FPGAs to Design Custom Digital Signal Processing Devices*
3. *Digital Signal Processing A Practical Approach by Ifeachor and Jervis*
4. *Discrete Time Signal Processing by Oppenheim and Schafer*

Conclusion

The Xilinx FIR filter Core produces a highly parameterizable, area-efficient, high-performance implementation. Highly optimized single-rate, half-band, Hilbert transform, and interpolated FIR filters can be fully realized with the filter compiler.

Appendix A – FIR Filter Coefficient (Impulse Response) Calculations

n=0: hD(n) = 2 fc' = 2 3 0.45 = 0.90
w(0) = 0.54 + 0.46 cos(0) = 1
h(0) = hD(0)w(0) = 0.90

n=1: hD(1) = [(230.45)/(2p30.45)] sin(2p30.45)
= [(230.45)/(2p30.45)] sin(36030.45) = 9.83631e-2
w(1) = 0.54 + 0.46 cos(2p/33) = 0.54 + 0.46 cos(360/33) = 0.99169
h(1) = h(-1) = hD(1)w(1) = 0.09755

n=2 : hD(2) = [(230.45)/(232p30.45)] sin(232p30.45)
= [(230.45)/(232p30.45)] sin(2336030.45) = -9.35489e-2
w(2) = 0.54 + 0.46 cos(232p/33) = 0.54 + 0.46 cos(23360/33) = 0.96705
h(2) = h(-2) = hD(2)w(2) = -0.09047

n=3 : hD(3) = [(230.45)/(332p30.45)] sin(332p30.45)
= [(230.45)/(332p30.45)] sin(3336030.45) = 8.58394e-2
w(3) = 0.54 + 0.46 cos(332p/33) = 0.54 + 0.46 cos(33360/33) =0.92698
h(3) = h(-3) = hD(3)w(3) = 0.07957

n=4: hD(4) = [(230.45)/(432p30.45)] sin(432p30.45)
= [(230.45)/(432p30.45)] sin(4336030.45) = -7.56827e-2
w(4) = 0.54 + 0.46 cos(432p/33) = 0.54 + 0.46 cos(43360/33) = 0.87292
h(4) = h(-4) = hD(4)w(4) = -0.06606

n=5: hD(5) = [(230.45)/(532p30.45)] sin(532p30.45)
= [(230.45)/(532p30.45)] sin(5336030.45) = 6.36620e-2
w(5) = 0.54 + 0.46 cos(532p/33) = 0.54 + 0.46 cos(53360/33) = 0.80683
h(5) = h(-5) = hD(5)w(5) = 0.05136

n=6: hD(6) = [(230.45)/(632p30.45)] sin(632p30.45)
= [(230.45)/(632p30.45)] sin(6336030.45) = -5.04551e-2
w(6) = 0.54 + 0.46 cos(632p/33) = 0.54 + 0.46 cos(63360/33) = 0.73109
h(6) = h(-6) = hD(6)w(6) = 0.03689

n=7: hD(7) = [(230.45)/(732p30.45)] sin(732p30.45)
= [(230.45)/(732p30.45)] sin(7336030.45) = 3.69971e-2
w(7) = 0.54 + 0.46 cos(732p/33) = 0.54 + 0.46 cos(73360/33) = 0.64845
h(7) = h(-7) = hD(7)w(7) = 0.02399

n=8: hD(8) = [(230.45)/(832p30.45)] sin(832p30.45)
= [(230.45)/(832p30.45)] sin(8336030.45) = -2.33872e-2
w(8) = 0.54 + 0.46 cos(832p/33) = 0.54 + 0.46 cos(83360/33) = 0.56189
h(8) = h(-8) = hD(8)w(8) = -0.01314
n=9: hD(9) = [(230.45)/(932p30.45)] sin(932p30.45)
= [(230.45)/(932p30.45)] sin(9336030.45) = 1.09292e-2

w(9) = 0.54 + 0.46 cos(932p/33) = 0.54 + 0.46 cos(93360/33) = 0.47454
h(9) = h(-9) = hD(9)w(9) = 0.00519

n=10: hD(10) = [(230.45)/(1032p30.45)] sin(1032p30.45)
= [(230.45)/(1032p30.45)] sin(10336030.45) = 0
w(10) = 0.54 + 0.46 cos(1032p/33) = 0.54 + 0.46 cos(103360/33) = 0.00518
h(10) = h(-10) = hD(10)w(10) = 0

n=11: hD(11) = [(230.45)/(1132p30.45)] sin(1132p30.45)
= [(230.45)/(1132p30.45)] sin(11336030.45) = -8.94211e-3
w(11) = 0.54 + 0.46 cos(1132p/33) = 0.54 + 0.46 cos(113360/33) = 0.31
h(11) = h(-11) = hD(11)w(11) = 0.00277

n=12: hD(12) = [(230.45)/(1232p30.45)] sin(1232p30.45)
= [(230.45)/(1232p30.45)] sin(12336030.45) = 1.55915e-2
w(12) = 0.54 + 0.46 cos(1232p/33) = 0.54 + 0.46 cos(123360/33) = 0.238764
h(12) = h(-12) = hD(12)w(12) = 0.00372

n=13: hD(13) = [(230.45)/(1332p30.45)] sin(1332p30.45)
= [(230.45)/(1332p30.45)] sin(13336030.45) = -1.98091e-2
w(13) = 0.54 + 0.46 cos(1332p/33) = 0.54 + 0.46 cos(133360/33) = 0.17842
h(13) = h(-13) = hD(13)w(13) = -0.00353

n=14: hD(14) = [(230.45)/(1432p30.45)] sin(1432p30.45)
= [(230.45)/(1432p30.45)] sin(14336030.45) = 2.16236e-2
w(14) = 0.54 + 0.46 cos(1432p/33) = 0.54 + 0.46 cos(143360/33) = 0.13114
h(14) = h(-14) = hD(14)w(14) = 0.00284

n=15: hD(15) = [(230.45)/(1532p30.45)] sin(1532p30.45)
= [(230.45)/(1532p30.45)] sin(15336030.45) = -2.12207e-2
w(15) = 0.54 + 0.46 cos(1532p/33) = 0.54 + 0.46 cos(153360/33) = 9.86332e-2
h(15) = h(-15) = hD(15)w(15) = -0.00209

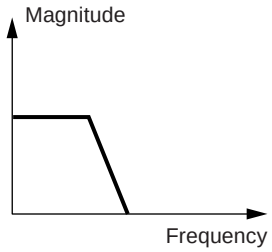
n=16: hD(16) = [(230.45)/(1632p30.45)] sin(1632p30.45)
= [(230.45)/(1632p30.45)] sin(16336030.45) = 1.89207e-2
w(16) = 0.54 + 0.46 cos(1632p/33) = 0.54 + 0.46 cos(163360/33) = 8.20829e-2
h(16) = h(-16) = hD(16)w(16) = 0.00155

Appendix B – Terms and Definitions

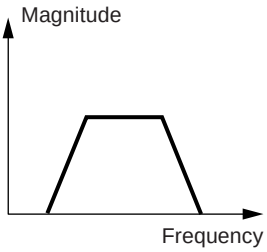
Phase linearity – means that the delay in the output sequence in relation to the input sequence is characterized by a linear relationship.

Stability – determines whether a system exhibits oscillatory behavior.

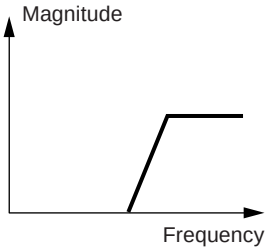
Low-pass filter –



Band-pass filter –



High-pass filter –



Direct Form (Transversal) – filter implementation that only has delays, summations, and multiplication, but no feedback.

Convolution Sum – the equation $y(n) = \sum h(k)\, x(n-k)$, which is the summation of the products of the impulse response and the input sequence.

Z-Transform – an important transform used in the analysis of digital filter response and stability.

Gibbs Phenomenon – a phenomenon observed when a rectangular pulse is approximated by too small a number of waveforms, resulting in a smudged, non-rectangular version of the pulse.

Distributed Arithmetic – a method which breaks down a DSP algorithm to its basic components of ANDs and ORs, allowing serial to fully parallel realizations in the Xilinx FPGA architecture.

EEG – Electroencephalography (brainwave measurement)

EOG – Electro-oculography (measurement of potentials defined by eye area)

ECG (or EKG) – Electrocardiography (heartbeat measurement)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/03/00	1.0	Initial Xilinx release.