SANDIA REPORT

SAND2003-0286
Unlimited Release
Printed January 2003

Implementation of a High Throughput Variable Decimation Pane Filter Using the Xilinx System Generator

Jerahmie W. Radder

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND2003-0286
Unlimited Release
Printed January 2003

Implementation of a High Throughput Variable Decimation Pane Filter Using the Xilinx System Generator

Jerahmie W. Radder*
Synthetic Aperture Radar II Department
Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185-0529

ABSTRACT

The Xilinx System Generator for Simulink provides DSP designers with a unique new approach to implementing DSP algorithms in FPGA hardware. Using the System Generator, a MAC-based variable decimation FIR filter, commonly known as a pane filter, has been designed for use in a digital radar system. Design and simulation of the pane filter was performed exclusively in the System Generator and MATLAB/Simulink environments. This pane filter architecture has been successfully compiled and implemented in FPGA (Xilinx Virtex II) hardware, and the output of the pane filter agrees with a conventional decimating FIR filter. The existence of hardware multipliers and block RAM in the Xilinx Virtex II FPGA makes for efficient mapping of the pane filter to the reconfigurable logic resources.

A method for implementing a parallel version of the variable-decimation pane filter architecture is also presented. This parallel pane filter increases filter throughput at the expense of hardware resources, but retains a constant FPGA clock frequency. This method addresses the need for high-throughput systems, which are limited by a fixed-frequency system clock rather than hardware resources.

---

* The author is a graduate student in electrical engineering at the University of Wisconsin. This work was performed under the Sandia National Laboratories student intern program.

ACKNOWLEDGEMENTS

This work was funded under the Sandia National Laboratories LDRD (Laboratory Directed Research and Development) program. The project is entitled “Low Cost Digital Radar for Fuzing, Tags, SAR Imaging, and Targeting” and is funded under the Emerging Threats SBU.

Additional funding was provided under the Sandia National Laboratories LDRD project entitled “Detection and Exploitation of Spread-Spectrum Waveforms” which is funded under the Nonproliferation and Materials Control SBU.

CONTENTS

This page left intentionally blank.

1 Introduction

In a Synthetic Aperture Radar (SAR) system, the purpose of the receiver is to process incoming radar signals in order to obtain target information and ultimately construct an image of the target area. Incoming raw signals are usually in the microwave frequency range and are typically processed with analog circuitry, requiring hardware designed specifically for the desired signal processing operations.

A more flexible approach is to process the signals in the digital domain. Recent advances in analog-to-digital converter (ADC) and Field Programmable Gate Array (FPGA) technology allow direct digital processing of wideband intermediate frequency (IF) signals. Modern ADCs can achieve sampling rates in excess of 1GS/s, and modern FPGAs can contain millions of logic gates operating at frequencies over 100 MHz. The combination of these technologies is necessary to implement a digital radar receiver capable of performing high speed, sophisticated and scalable DSP designs that are not possible with analog systems. Additionally, FPGA technology allows designs to be modified as the design parameters change without the need for redesigning circuit boards, potentially saving both time and money.

For typical radars receivers, there is a need for operation at multiple ranges, which requires filters with multiple decimation rates, i.e., multiple bandwidths. In previous radar receivers, variable decimation was implemented by switching between SAW filters to achieve an acceptable filter configuration. While this method works, it is rather "brute force" because it duplicates a large amount of hardware and requires a new filter to be added for each IF bandwidth. By implementing the filter digitally in FPGAs, a larger number of decimation values (and consequently a larger number of bandwidths) can be implemented with no need for extra components.

High performance, wide bandwidth radar systems also place high demands on the DSP throughput of a given digital receiver. In such applications, the maximum clock frequency of a given FPGA is not adequate to support the required data throughput. This problem can be overcome by employing a parallel implementation of the pane filter. The parallel pane filter uses a polyphase parallelization technique to achieve an aggregate data rate which is twice that of the FPGA clock frequency. This is achieved at the expense of roughly doubling the FPGA resource usage.

2 Hardware Design Process

A wide variety of tools exist for the development of digital systems. The most basic method is to use a hardware description language (HDL) such as VHDL or Verilog. While this method offers a great deal of control over a design, it is also the most complicated. A large VHDL design may require hundreds of pages of code, which can be a daunting task for even an experienced VHDL designer. Other options include higher-level design tools that allow designs to be created with functional blocks and schematics in a graphical environment.

Xilinx System Generator [2] for Simulink is a relatively new high-level design tool that allows digital system design and simulation to be performed in the MATLAB/Simulink [1] environment. Working in the MATLAB/Simulink environment offers several key advantages over other high-level design tools. System Generator allows hardware blocks to be placed in a Simulink model and directly simulated. Simulink provides a wide variety of tools that can be used to examine signals within a design, allowing the designer the ability to monitor any signal in the system. Additionally, simulation data can be exchanged between Simulink and MATLAB, allowing the designer to change parameters, plot data, or process data with customized MATLAB scripts.

Once a design has been created and simulated, it can be converted to a bitstream file to program a FPGA using the System Generator and Xilinx ISE [3] implementation tools. A demonstration board has been created to test designs using Xilinx Virtex-II FPGAs. A series of MATLAB functions have been created to program the FPGA and transfer data between a PC and the demonstration board RAM. These features allow a design to be tested directly in hardware and the results to be extracted from RAM and analyzed within the MATLAB/Simulink environment.

3 Pane Filter Theory

Decimation finite-impulse response (FIR) filtering is a fundamental digital signal processing (DSP) function. Most decimation filtering hardware solutions, however, are geared towards fixed bandwidth and hence fixed decimation factors. For pulsed radar and other systems requiring variable filter bandwidth, a true variable decimation FIR filter implementation is required. The pane filter satisfies this requirement.

Filtering with decimation can be represented in the time domain as a discrete convolution of signal and coefficient values followed by decimation [4]:

$$y(m) = \{x(m) * c(m)\}(\downarrow D) = \sum_{i=-\infty}^{+\infty} c[i] \cdot x[Dm - i] = \sum_{i=0}^{N-1} c[i] \cdot x[Dm - i].$$

Equation 1. FIR filter with decimation

Equation 1 can be implemented in a number of ways, the simplest being a conventional FIR filter followed by decimation. This method is wasteful of computational resources because only every Dth value is retained after computation. A more efficient approach is to compute only the desired outputs using an architecture such as the polyphase, systolic, or cascaded integrator comb (CIC) filter [4][5][6]. These methods efficiently compute the decimated FIR output, but are generally restricted to fixed decimation rates.

In order to achieve variable decimation rates, a pane filter architecture was adopted. A pane filter is a variable decimation multiplier-accumulator (MAC) based filter that uses multiple "panes" to directly perform a time-domain convolution in Equation 1 [7]. The structure of a MAC or "pane" is shown in Figure 1, and the general form of the pane filter is shown in Figure 2.

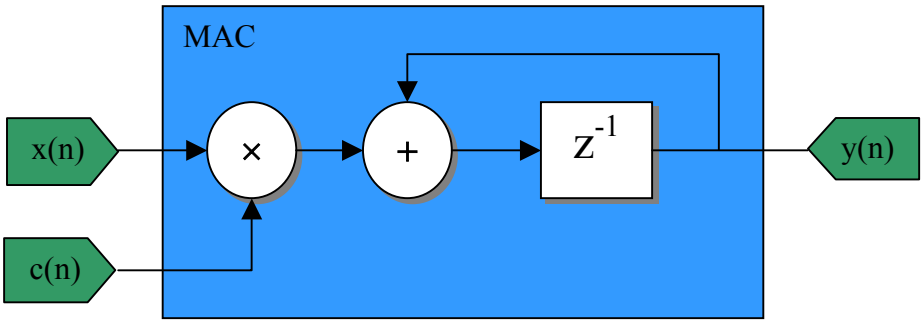

Figure 1. General MAC architecture

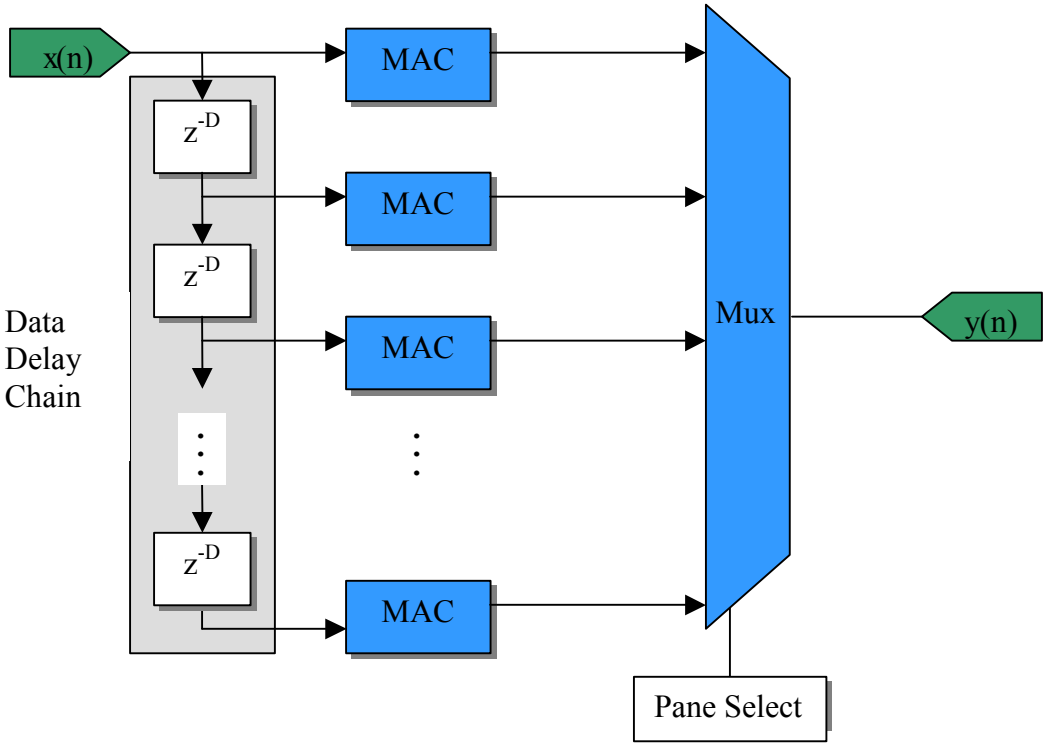

Figure 2. Pane filter architecture

4 Pane Filter Design

The pane filter architecture can be implemented in two possible ways. The first way is to keep the data fixed and staggering the filter coefficients between panes. The other method fixes the input filter coefficients, but delays the input data between panes. Both methods are equally valid because of the commutative property of convolution. The fixed filter coefficients and delayed data input method is presented here.

The components of the pane filter and the System Generator implementations are described below. A pane filter consists of essentially six functional blocks: a data delay chain, a MAC (pane), a barrel shifter, a multiplexer, FIR filter coefficients, and a data valid signal. A pane filter with six panes has been designed and simulated in the System Generator environment. Figure 3 displays the System Generator implementation of the pane filter.

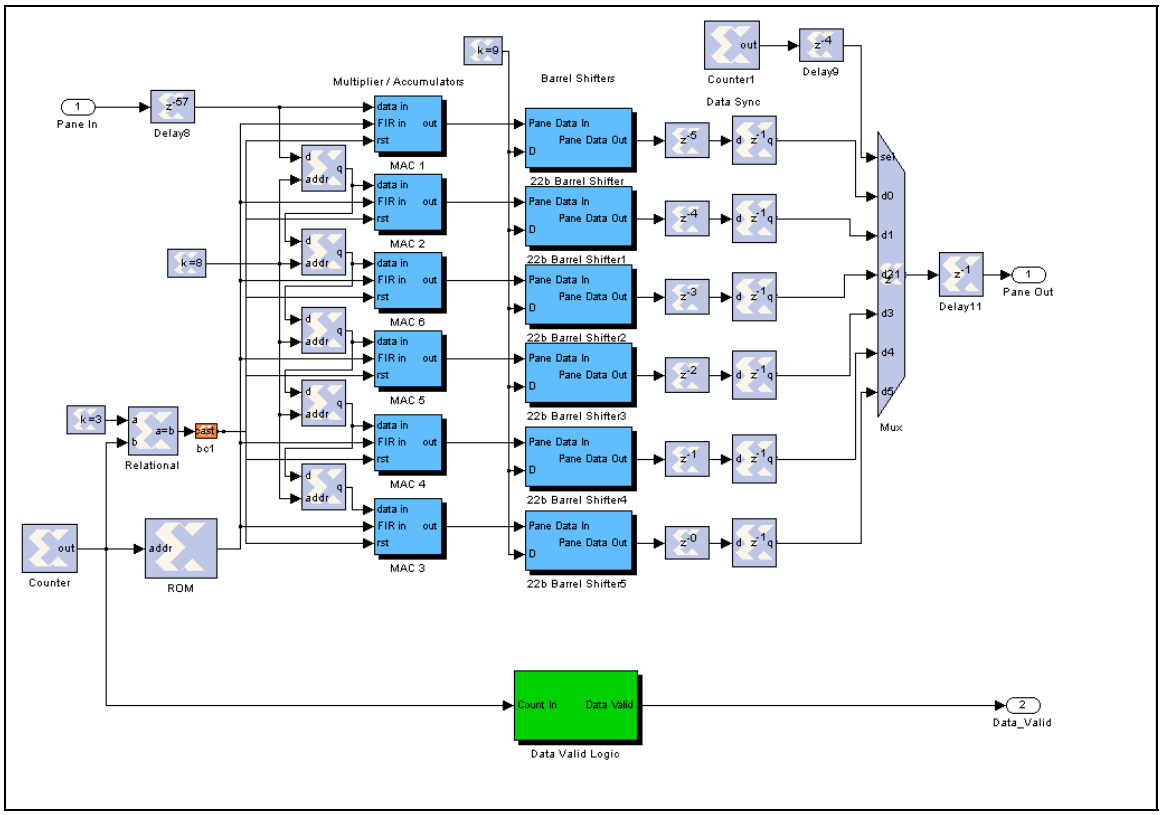

Figure 3. Pane filter implementation in System Generator for Simulink

4.1 Data Delay Chain

The data delay chain (Figure 4) simply delays the input signal by the decimation rate, *D*, between each pane of the filter. This delay provides the variable decimation property of the pane filter output. Each tap off the data delay chain is supplied to an individual pane in the pane filter. These parameterized delay blocks are implemented with Xilinx "Addressable Shift Register" blocks, were the address is D-1. These blocks do not allow an address of zero to be sent to the block, so D>= 2. The special case of D = 1 will bypass the pane filter entirely.

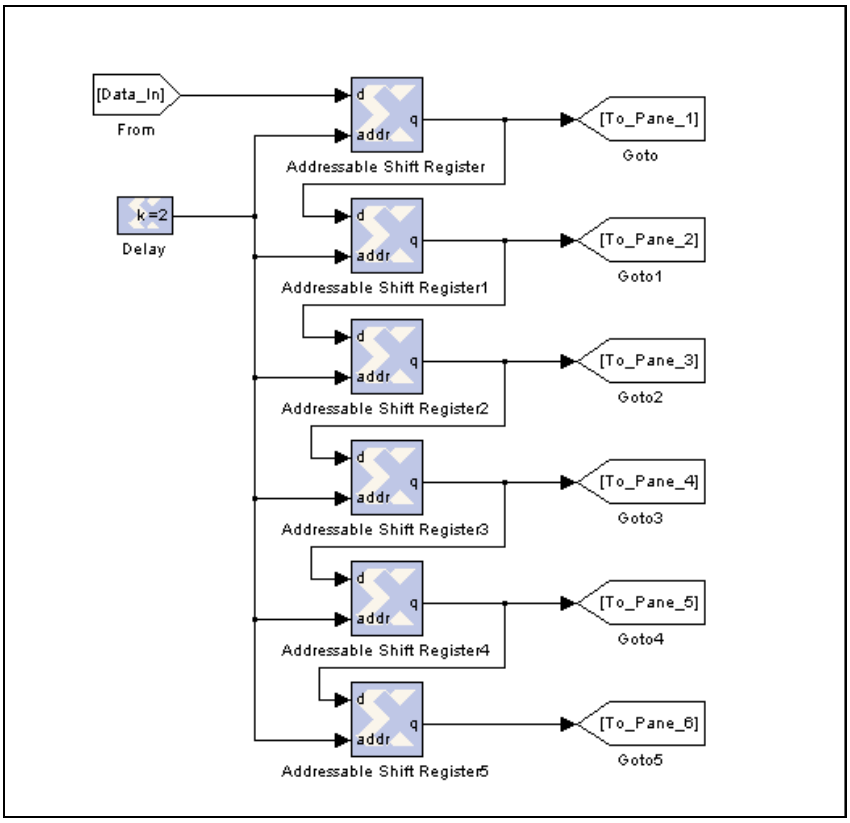

Figure 4. Data delay chain in System Generator

4.2 FIR Filter Coefficients

The FIR filter coefficients are calculated in the same fashion as a conventional FIR filter. Unlike polyphase filters and some other decimating FIR filter architectures, which must reorganize filter coefficients to calculate the decimated filter output, the pane filter accepts FIR coefficients sequentially and does not rely on any symmetry of the coefficients. This produces a very general decimating FIR architecture. However, if reduction of system resources is critical, some loss of generality can be made to take advantage of filter symmetries and anti-symmetries, which can potentially reduce the amount of memory necessary to store the filter coefficients.

4.3 MAC

The pane filter architecture contains one multiplier/accumulator (MAC) for each pane. The MAC blocks perform the time-domain convolution operations, where each MAC utilizes a full-width hardware multiplier followed by a 44-bit accumulator. This portion of the filter is implemented using the System Generator "multiplier" and "accumulator"

blocks (Figure 5). A conservative (on the high-side) estimate for the size of the accumulator which minimizes the truncation error for the MAC operation is given by:

$$acc\_width \approx data\_width + FIR\_coeff\_width + \log_2((\#\ panes)\times(max\_decimation)).$$

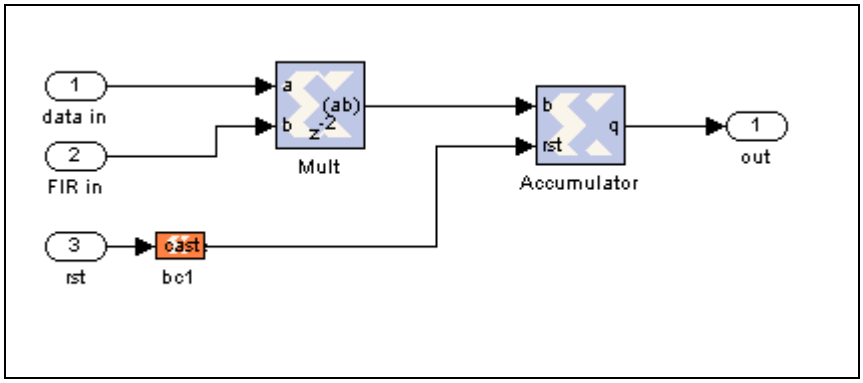

Figure 5. Multiplier-Accumulator implementation in System Generator

4.4 Barrel Shifter And Output

A barrel shifter converts the 44-bit output of each MAC to 22-bits such that the dynamic range of the MAC output is maximized. The outputs of each pane are then sent through a multiplexer, which switches through the individual panes to select the proper output. For a given coefficient set, a one-to-one mapping from the decimation factor D to the barrel shifter MSB offset can be readily obtained.

4.5 Data Valid Signal

The output is valid only for *Npn* cycles out of every *T* cycles. Since *Npn* is typically much smaller than T, the output is said to be "bursty". At any of the cycles where the output is not valid, a partial calculation of the decimated FIR output is present at the pane filter output. A state machine is responsible for driving a data valid signal, which determines whether or not the output is indeed valid. This data valid signal is required by the next digital block in the radar system to determine if the data from the pane filter is indeed valid.

5 Pane Filter Simulation Results

In order to test the pane filter model in Sytem Generator, a linear chirp signal is presented to the input of the pane filter and a conventional FIR filter with decimation. Simulink generates the input signal and the output is exported to MATLAB for plotting. Both time and frequency domain results of the simulation are shown in Figure 6 and Figure 7. Note that the pane filter output is identical to the FIR output in both the frequency and time domains.

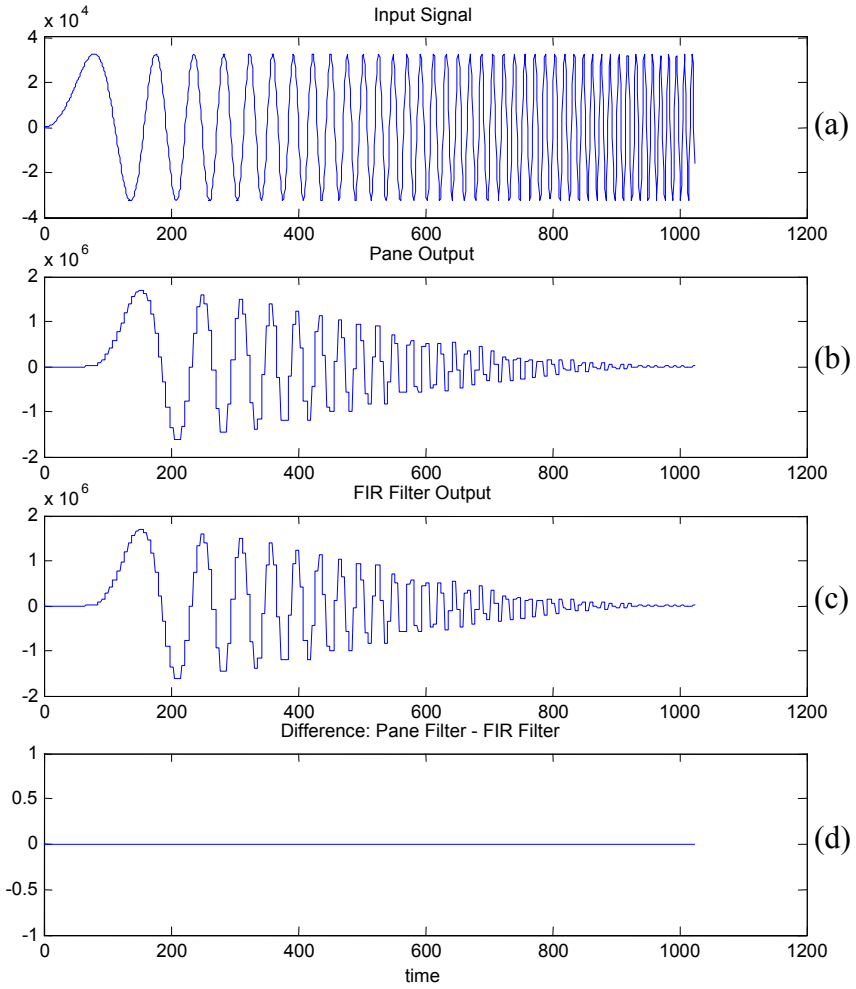

Figure 6. Time domain simulation: a) Input chirp signal, b) Pane filter response, c) FIR filter response, d) Difference between pane and FIR response, (D = 6)

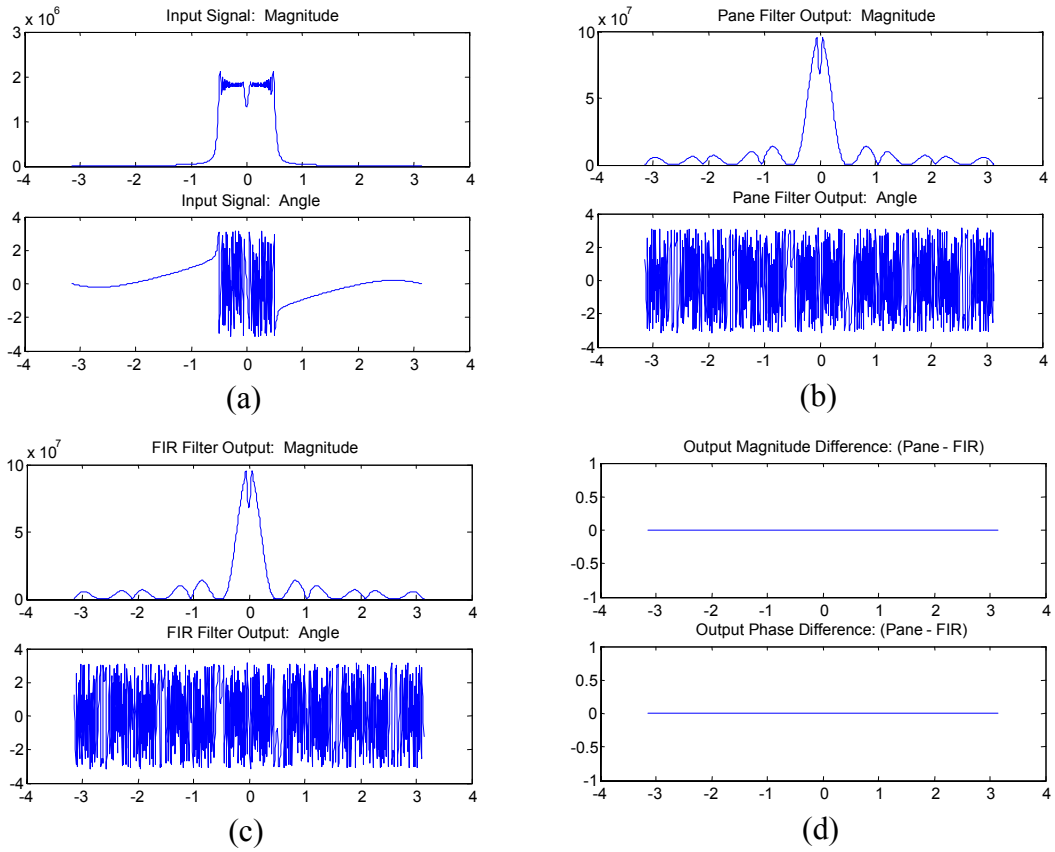

Figure 7. Frequency domain representation: a) Input signal, b) Pane filter, c) FIR filter, d) Difference between FIR filter and pane filter, (D = 6)

6 Pane Filter Hardware Results

The System Generator was used in conjunction with Xilinx ISE development software to create and compile the pane filter design for hardware implementation. The pane filter was then loaded into Xilinx Virtex II FPGA hardware. Using MATLAB test scripts, a chirp waveform test vector was loaded into the FPGA RAM via a VMEbus interface, processed using the pane filter, and the filtered output was read back out of RAM. The results for a D = 8 case are shown in Figure 8 along with a conventional FIR filter with decimation, as well as the difference between the pane and FIR implementations. As can be seen, the pane and FIR filter implementations are in agreement, validating the pane filter architecture in FPGA hardware.

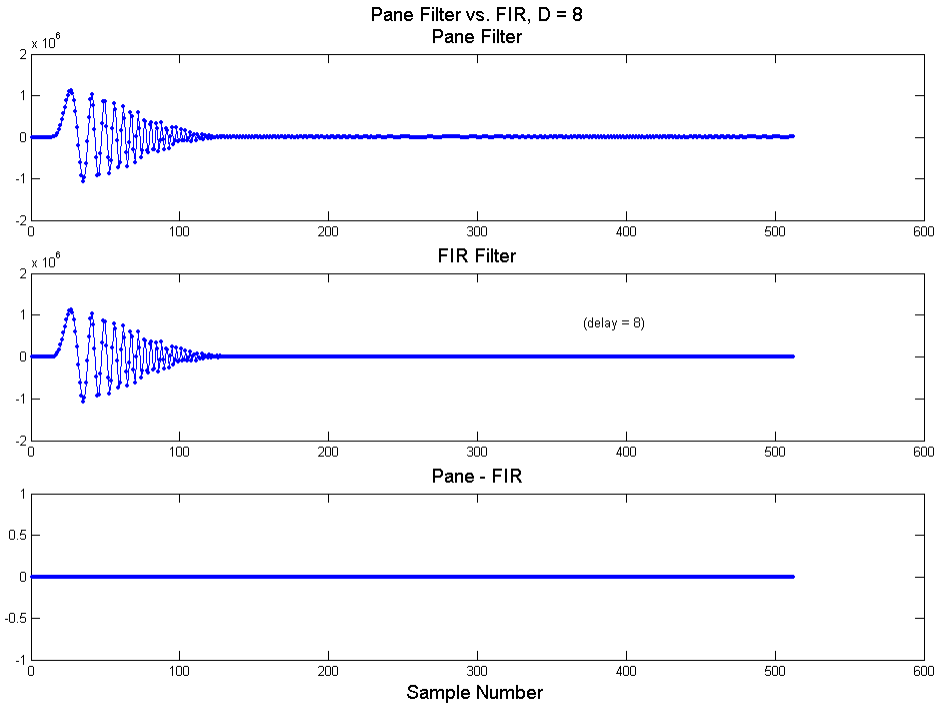

Figure 8. Hardware results of pane filter, (D = 8)

7 Parallel Pane Filter Design

One disadvantage to using FPGAs for FIR filter implementation is that data throughput is limited by the clock rate of the FPGA. One method of increasing data throughput is to create a parallel implementation of the pane filter. The goal of this parallel architecture is to keep the system clock rate unchanged, while increasing the data-processing rate at the expense of hardware resources. The inputs and outputs of the parallel pane filter architecture are illustrated in Figure 9. In this representation, a pane filter with a single data input and output is replaced by a parallel pane architecture with two inputs and two outputs operating at half the original data rate.

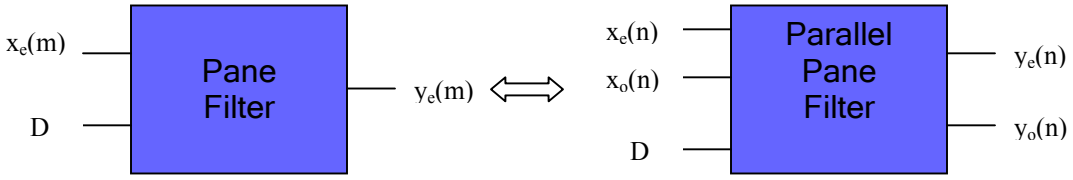

Figure 9. Parallel pane filter replaces a single-input, single-output system with a two-input, two-output system.

For this analysis, we will assume a discrete input sequence, *x(m)*, of length *N*. This input sequence can be split into two inputs, $x_{even}(n)$ and $x_{odd}(n)$, which are each approximately half the length of the original sequence. In this form, x_{even} and x_{odd} represents the even numbered indices and the odd numbered indices of the original signal, respectively. In equation form, this is expressed as:

$$\begin{aligned} x(m) &= x_0, x_1, x_2, x_3, \ldots x_{N-1}, \\ x_{even}(n) &= x(2m) = x_0, x_2, x_4, x_6, \cdots, x_{\left\lfloor \frac{N-1}{2} \right\rfloor}, \\ x_{odd}(n) &= x(2m+1) = x_1, x_3, x_5, \cdots, x_{\left\lfloor \frac{N-2}{2} \right\rfloor}. \end{aligned}$$

Equation 2. Decimated even and odd input from original input signal

In the System Generator, the even and odd outputs can be easily implemented with two down sample blocks and a single delay block (Figure 10). The down sample operation is chosen to return the first element in each frame, rather than the last sample. For this implementation, x_{even} is the most delayed input signal.

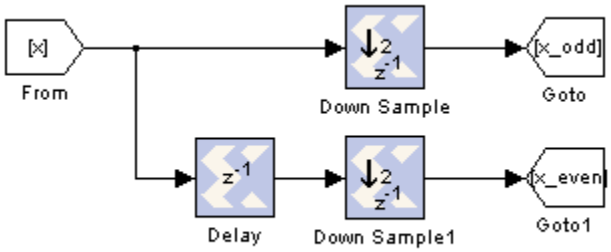

Figure 10. System Generator model for generating x_{even} and x_{odd} from input sequence, x

7.1 FIR Filtering with Decimation by D

Filtering with decimation has been expressed as the discrete convolution of data, *x(m)*, and FIR filter coefficients, *c(m)*, followed by decimation, where *D* is the integer decimation factor. This filtering and decimation function is given in Equation 1. By replacing the index *m* with *2n* and *2n+1* for the even and odd output indices, respectively, Equation 1 can be split to yield the following output expressions:

$$y_{even}(n) = y(2n) = \sum_{i=0}^{N-1} c[i] \cdot x[2nD - i],$$

Equation 3. Even-indexed filtered output

$$y_{odd}(n) = y(2n+1) = \sum_{i=0}^{N-1} c[i] \cdot x[(2n+1)D - i],$$

Equation 4. Odd-indexed filtered output

where, $n = \left\lfloor \frac{m}{2} \right\rfloor$, $m = 0,1,2,3,4,...$, where $\lfloor \cdot \rfloor$ is the floor function.

The general approach to create a parallel implementation is to divide the coefficients and data into even and odd components, c_{even}, c_{odd}, x_{even}, and x_{odd}, and substitute into Equation 3 and Equation 4 to create a new expression for y_{even} and y_{odd}. The output expressions are different for the even and odd decimation cases, and will be treated separately.

7.1.1 Even Decimation, D = 2,4,6,8…

The parallel pane filter outputs are examined first for the case of even decimation (D = 2,4,6,8…). An expression for the even output, $y_{even}(n)$, is obtained by replacing the summation index, i, with a new summation variable, k, to allow y_{even} to be expressed as a sum of even and odd-indexed terms:

$$\begin{aligned}
y_{even}(n) = y(2n) &= \sum_{i=0}^{N-1} c[i] \cdot x[2Dn - i], \\
&= \sum_{k=0}^{\frac{N}{2}-1} c[2k] \cdot x[2Dn - 2k] + c[2k-1] \cdot x[2Dn - 2k - 1], \\
&= \sum_{k=0}^{\frac{N}{2}-1} c[2k] \cdot x[2(Dn - k)] + c[2k-1] \cdot x[2(Dn - k) - 1], \\
&= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k] \cdot x_{even}[Dn - k] + c_{odd}[k] \cdot x_{odd}[Dn - k - 1].
\end{aligned}$$

Equation 5. Parallel even-indexed output with even decimation

Here, the odd coefficients are indexed by *2k-1* instead of *2k+1* in order to decimate on the first value of the output frame. This ensures the first output value is *y(0) = c(0)x(0)*, and therefore consistent with the output of decimating pane filter or equivalent polyphase FIR filter.

The derivation of y_{odd} for even decimation follows a similar approach to the y_{even} derivation. In Equation 4, the coefficients and data are split into even and odd components, producing the parallel expression for y_{odd}:

$$\begin{aligned} y_{odd}(n) &= y(2n+1) = \sum_{i=0}^{N-1} c[i] \cdot x[D(2n+1) - i], \\ &= \sum_{k=0}^{\frac{N}{2}-1} c[2k] \cdot x[D(2n+1) - 2k] + c[2k-1] \cdot x[D(2n+1) - 2k - 1], \\ &= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k] \cdot x_{even}\left[\frac{D}{2}(2n+1) - k\right] + c_{odd}[k] \cdot x_{odd}\left[\frac{D}{2}(2n+1) - 1 - k\right]. \end{aligned}$$

Equation 6. Parallel odd-indexed output with even decimation

7.1.2 Odd Decimation, D = 1,3,5,7…

For odd decimation, the expression for the even output is identical to the expression for even decimation case. This can be attributed to the fact that product *2nD* is always an even number, regardless of the value of D. This does not change the indexing of x_{even} and x_{odd} and leaves the final output unchanged from the even decimation case. This is not the case for the odd output for odd decimation. In this case, the index *(2n+1)D* is always odd, leading to a different expression for y_{odd}:

$$\begin{aligned} y_{even}(n) &= y(2n) = \sum_{i=0}^{N-1} c[i] \cdot x[2Dn - i], \\ &= \sum_{k=0}^{\frac{N}{2}-1} c[2k] \cdot x[2Dn - 2k] + c[2k-1] \cdot x[2Dn - 2k - 1], \\ &= \sum_{k=0}^{\frac{N}{2}-1} c[2k] \cdot x[2(Dn - k)] + c[2k-1] \cdot x[2(Dn - k) - 1], \\ &= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k] \cdot x_{even}[Dn - k] + c_{odd}[k] \cdot x_{odd}[Dn - k - 1], \end{aligned}$$

Equation 7. Parallel even-indexed output with odd decimation

$$
\begin{aligned}
y_{odd}(n) = y(2n+1) &= \sum_{i=0}^{N-1} c[i]\cdot x[D(2n+1)-i], \\
&= \sum_{k=0}^{\frac{N}{2}-1} c[2k]\cdot x[D(2n+1)-2k] + c[2k-1]\cdot x[D(2n+1)-2k-1], \\
&= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k]\cdot x_{odd}\left[\left\lfloor \frac{D}{2} \right\rfloor (2n+1)-k\right] + c_{odd}[k]\cdot x_{even}\left[\left\lfloor \frac{D}{2} \right\rfloor (2n+1)-1-k\right], \\
&= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k]\cdot x_{odd}\left[Dn+\left\lfloor \frac{D}{2} \right\rfloor -k\right] + c_{odd}[k]\cdot x_{even}\left[2n+\left\lfloor \frac{D}{2} \right\rfloor -1-k\right], \\
&= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k]\cdot x_{odd}\left[Dn+\frac{D-1}{2}-k\right] + c_{odd}[k]\cdot x_{even}\left[Dn+\frac{D-1}{2}-1-k\right],
\end{aligned}
$$

Equation 8. Parallel odd-indexed output with odd decimation

where we use $\left\lfloor \frac{D}{2} \right\rfloor = \frac{D-1}{2}$.

The four cases are summarized for the even and odd decimation cases:

<u>D = even:</u>

$$
\begin{aligned}
y_{even}(n) &= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k]\cdot x_{even}[Dn-k] + c_{odd}[k]\cdot x_{odd}[Dn-k-1] \\
y_{odd}(n) &= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k]\cdot x_{even}\left[\frac{D}{2}(2n+1)-k\right] + c_{odd}[k]\cdot x_{odd}\left[\frac{D}{2}(2n+1)-1-k\right]
\end{aligned}
$$

<u>D = odd:</u>

$$
\begin{aligned}
y_{even}(n) &= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k]\cdot x_{even}[Dn-k] + c_{odd}[k]\cdot x_{odd}[Dn-k-1] \\
y_{odd}(n) &= \sum_{k=0}^{\frac{N}{2}-1} c_{even}[k]\cdot x_{odd}\left[Dn+\frac{D-1}{2}-k\right] + c_{odd}[k]\cdot x_{even}\left[Dn+\frac{D-1}{2}-k\right]
\end{aligned}
$$

7.2 System Generator Implementation

A block diagram of the even and odd decimation parallel pane filters are shown in Figure 11 and Figure 12. The delays represented in the block diagram for the data inputs are determined by setting $n=k=0$ for the indices of x_{even} and x_{odd} of the even and odd output expressions. Note that coefficients are swapped for the odd outputs between the even and odd decimation cases.

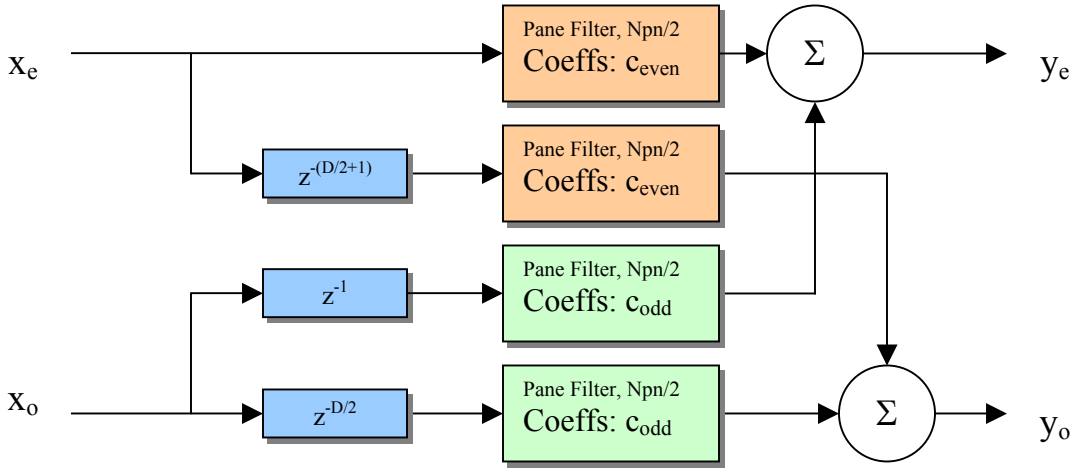

Figure 11. Block diagram of parallel pane filter for even decimation

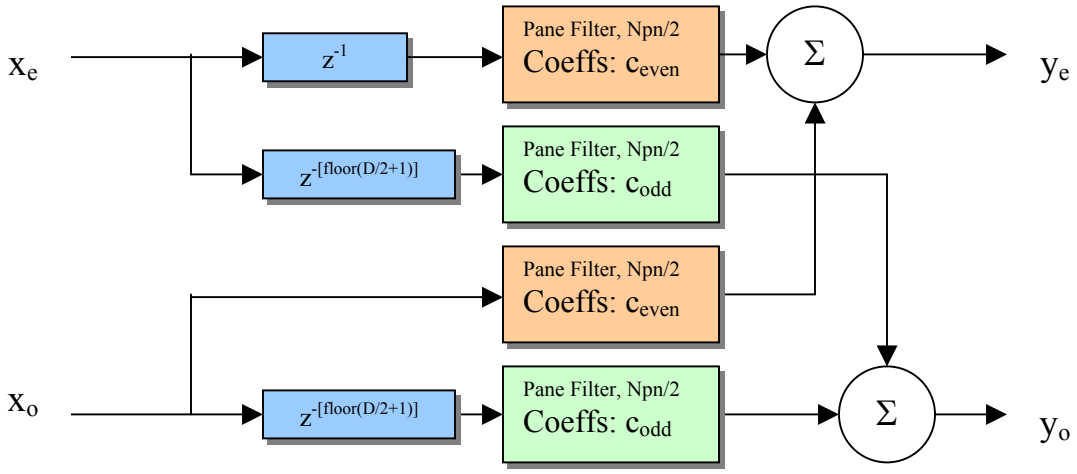

Figure 12. block diagram of parallel pane filter for odd decimation.

Equation 5, Equation 6, Equation 7, Equation 8, and the models represented in Figure 11 and Figure 12 were verified using the Xilinx System Generator. Figure 13 and Figure 15 display the System Generator models for even and odd decimation, respectively. A conventional pane filter implementation contains N_{pn} panes, where $N_{pn} = T/D$ (T is the

number of taps in the FIR filter). The corresponding parallel pane filter contains four pane filters, each containing $N_{pn}/2$ panes. The parallel pane filter implementation uses twice as many panes as the original pane filter, but also has twice the throughput. The simulation results of the parallel pane filter are in agreement with a simulation of a standard FIR filter followed by decimation, as shown in Figure 14 and Figure 16.

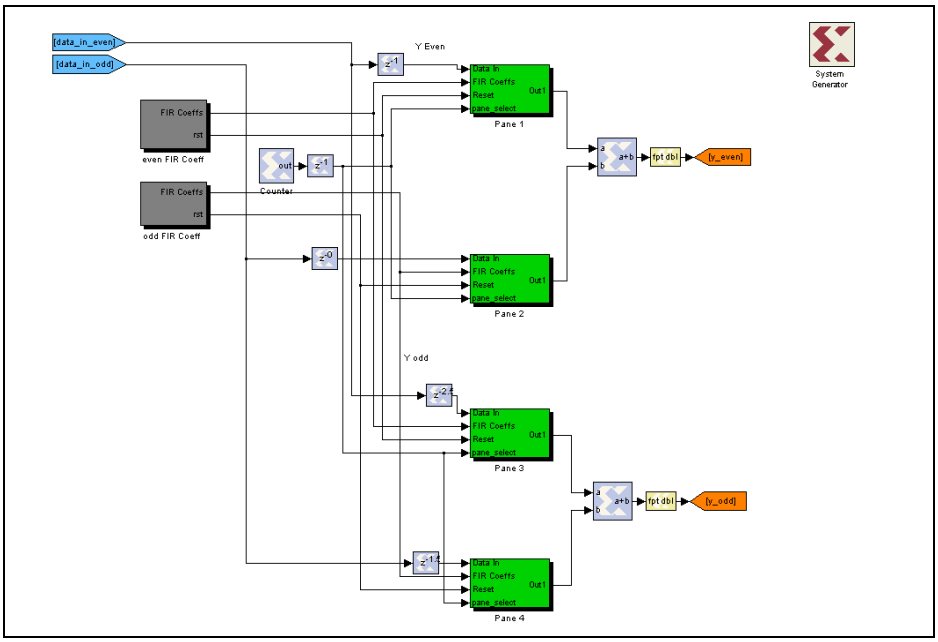

Figure 13. System Generator model of parallel pane filter for even decimation

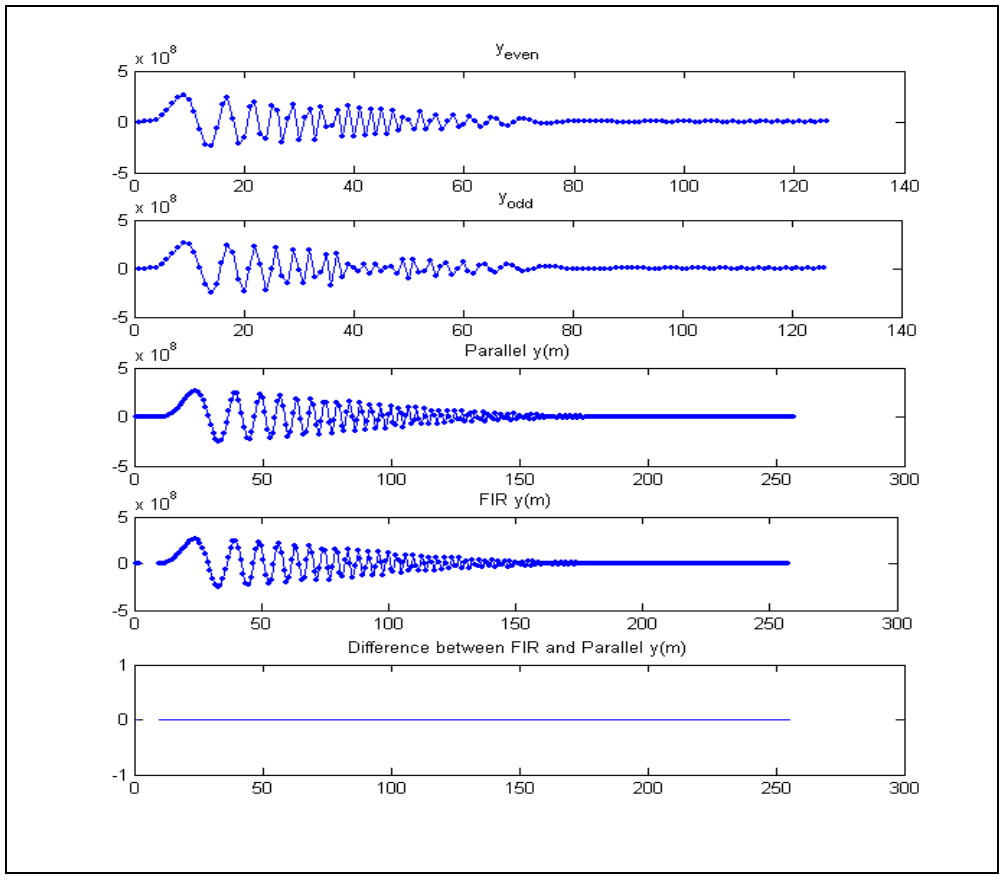

Figure 14. System generator simulation of parallel pane filter for even decimation

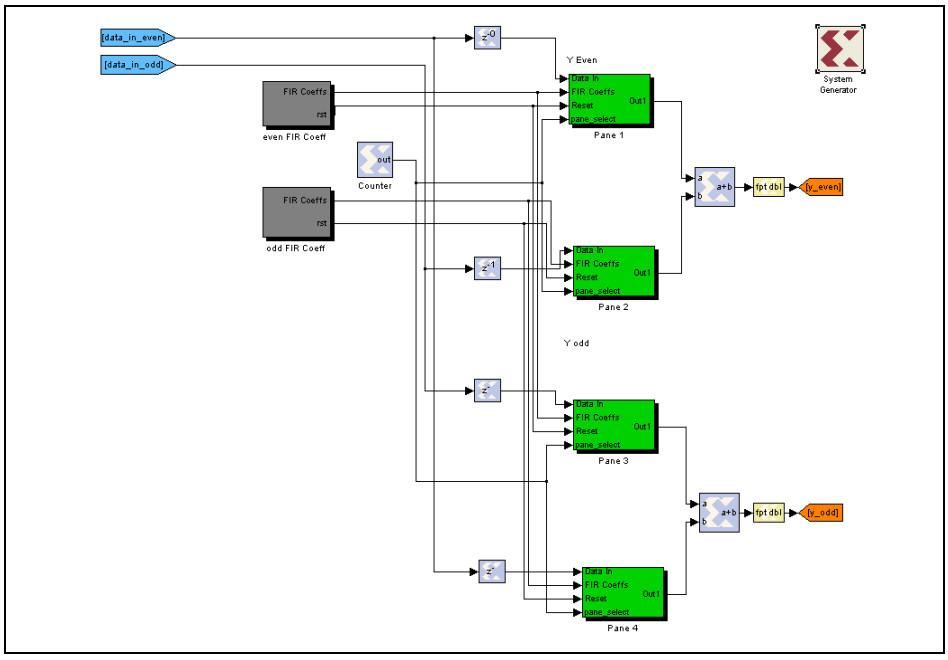

Figure 15. System Generator model of parallel pane filter for odd decimation

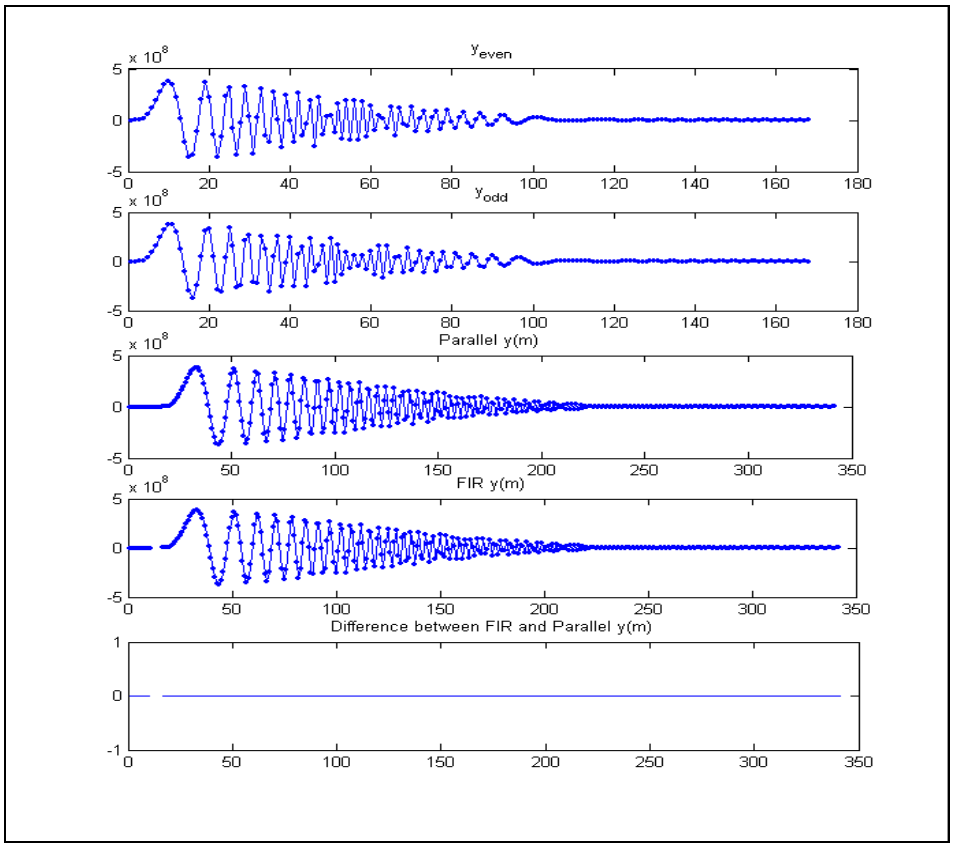

Figure 16. System generator simulation results of parallel pane filter for odd decimation

We note that the even and odd decimation cases are treated separately. Both cases can, however, be combined into one model. This can be accomplished by using programmable delay blocks on each of the four parallel pane filter branches and employing logic to properly commutate the pane coefficients between the even and odd decimation cases.

Conclusions

The Xilinx System Generator has been used to design a variable-decimation pane filter for use in a digital radar receiver. The pane filter is a multiplier-accumulator (MAC) based FIR filter, which efficiently calculates a decimated FIR filter output using a set of "panes." Each "pane" directly computes output values, provided that $T = Npn*D$, where T is the number of FIR filter coefficients, Npn is the number of panes and D is the maximum integer decimation rate of the system. The System Generator was chosen over other design and simulation tools for its intuitive Simulink interface and its ability to directly access the MATLAB software environment. The ability to create and evaluate a DSP model in the MATLAB/Simulink environment allows a DSP model to be created and evaluated quickly with little logic design skills. Using the System Generator, the pane filter with six panes has been show to agree with a FIR filter followed by decimation for both Simulink simulations and hardware implementations.

In order to increase the data throughput of a variable decimation pane filter, a parallel pane filter has been developed. The output expression of a time-domain FIR filter can be divided into even and odd indexed components and then further divided into an expression of even and odd indexed data and filter coefficients. The parallel pane filter expressions have been modeled successfully using the System Generator for Simulink and the results agree with a FIR filter followed by decimation.

REFERENCES

[1] Matlab and Simulink Technical Computing Environment, Release 12.1, The Mathworks, Inc., May, 2001.

[2] The System Generator, Release 2.2, Xilinx, Inc., May, 2002.

[3] Foundation Integrated Software Environment (ISE), Release 5.1i, Xilinx, Inc., 2002.

[4] Crochiere, R. E., Rabiner, L., R., *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[5] Abdel-Raheem, E., El-Guibaly, F., Antoniou, A., "Systolic Implementation of FIR Decimators and Interpolators," IEE Proc.-Circuits Devices Syst., Dec. 1994, vol. 141, no. 6, p. 489-492.

[6] Wasserman, L., Wilson, A. N., "A Variable-Rate Filtering System for Digital Communications," Int. Conf. Acoustics, Speech, Signal Processing, March 15, 1999, p. 1497-1500.

[7] Lambert, K. H., Longley, L. A., "Improved Decimating FIR Filter," Motorola Technical Developments, Oct. 5, 1985, vol. 5, p. 39-40.

DISTRIBUTION

Unlimited Release

1	MS 0537	R. E. Heintzleman	2333
1	MS 0537	L. M. Feldner	2333
1	MS 0529	B. L. Remund	2340
1	MS 0529	B. L. Burns	2340
1	MS 0505	S. M. Becker	2341
1	MS 0505	P. A. Dudley	2341
1	MS 0505	P. Ortiz	2341
1	MS 0505	D. C. Sprauer	2341
1	MS 0505	D. K. Werling	2341
1	MS 0505	C. Wojahn	2341
1	MS 0519	L. M. Wells	2344
1	MS 0519	J. T. Cordaro	2344
1	MS 0519	A. W. Doerry	2344
1	MS 0519	J. J. Mason	2344
1	MS 0519	M. S. Murray	2344
1	MS 0519	R. C. Ormesher	2344
1	MS 0529	K. W. Sorensen	2345
1	MS 0529	J. G. Baldwin	2345
1	MS 0529	B. C. Brock	2345
1	MS 0529	F. E. Heard	2345
5	MS 0529	D. F. Dubbert	2345
1	MS 0529	G. R. Sloan	2345
1	MS 0529	A. Martinez	2346
1	MS 0529	C. W. Ottesen	2346
1	MS 0519	W. H. Hensley	2348
1	MS 0519	T. P. Bielek	2348
1	MS 0519	S. M. Devonshire	2348
1	MS 0519	J. A. Hollowell	2348
1	MS 0519	K. R. White	2348
1	MS 0965	R. M. Axline	5711
1	MS 1207	N. E. Doren	5912
5		Jerahmie Radder	University of Wisconsin
1	MS 9018	Central Technical Files	8945-1
2	MS 0899	Technical Library	9616
1	MS 0612	Review & Approval Desk (DOE/OSTI)	9612