© 2014 IJIRT | Volume 1 Issue 6 | ISSN : 2349-6002

Digital Signal Processing: A Hardware-Based Approach

NAVEEN KUMAR,PRAVESH SHARMA,RAHUL SHARMA

(DRONACHARYA COLLEGE OF ENGINEERING)

I. INTRODUCTION

Teaching Digital Signal Processing (DSP) has included the utilization of a simulation tool (ST) for student projects and homework. The leading ST in academia is MATLAB by MathWorks. MATLAB is a vector based environment that is conducive to DSP simulation. Specifically, filter design is simulated utilizing a C-like code. Students are able to enter a filter design as a discrete time sequence or a discrete transfer function. MATLAB has build in functions that generate deterministic and non-deterministic signals which can then be inputted to the designed filter. The output of the filter can then be analyzed in the time-domain or frequency-domain utilizing other MATLAB functions. MATLAB, however, is not conducive to teaching the structural aspects of filter design.

Simulink is a block based design system that provides a graphical environment and a customizable set of block libraries that allows the user to simulate and test a variety of systems such as digital filters[1]. Simulink has an extensive DSP library that contains blocks for implementing everything from signal generation to adaptive filtering. Even though Simulink is a more realistic implementation environment than MATLAB, it is still purely simulation. Realizing the need for users to be able to perform real hardware implementations, MathWorks collaborated with Xilinx to produce System Generator (SG). SG is a group of extensive libraries that are included in Simulink. The SG libraries include hardware based blocks that interact with traditional Simulink blocks. Therefore, hardware designs can be synthesized, downloaded to a Xilinx Field Programmable Gate Array (FPGA) and then compared in real time to their Simulink simulation counterparts.

Simulink IIR Filter Design

IIR filter design is a standard topic in any DSP course. Specifically, second order lowpass, highpass, bandpass and bandstop filters can be implemented and analyzed. Shown in equation 1, is a well known transfer function of a second order notch IIR filter[2].

$$H(z) = 1 - K\beta(1(1 - + 2\alpha\beta)zz - - 11 + + \alpha z - 2z) - 2) \quad (1)$$

In the transfer function of equation 1, αdetermines the 3-dB bandwidth, βdetermines the center frequency and *K* determines the maximum magnitude value. This filter can be implemented in Simulink in various ways. One implementation method is the utilization of a built in IIR filter block where the user can specify the numerator and denominator coefficients of the transfer function. Shown in figure 1, however, is a discrete time equation implementation of the notch IIR filter in equation 1 withα=.1,β=.2, *K* =.55. In this design, a white noise sequence is sampled at 10 000 samples per second and inputted to the notch filter. Its output spectrum is then analyzed using a fast Fourier transform (FFT) block.

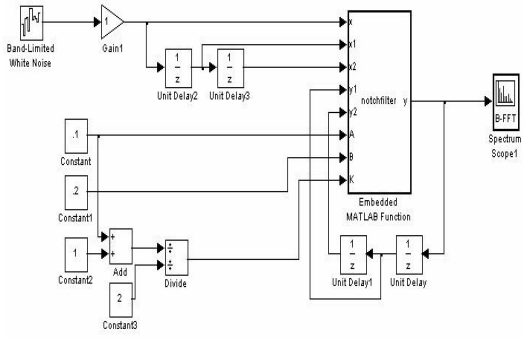

Figure 1: Simulink Notch Filter

The notch filter of equation 1 is implemented using an embedded MATLAB function and delay blocks as shown in figure 1. This implementation is a direct implementation of the discrete time equation that represents the transfer function as shown in equation 2. $y[n] = \beta(1+\alpha)y[n-1] - \alpha y[n-2] + Kx[n] - 2K\beta x[n-1] + Kx[n-2]$ (2)

System Generator IIR Filter Design

The Simulink implementation of the IIR filter as shown in figure 1, does not take into account structure.

Shown in figure 2, is a well known structure for IIR filters is Direct Form II[1]. Direct Form II is a canonic IIR filter structure wherein the number of delays is equal to the order of the filter. Canonic structures are important to minimize hardware components.

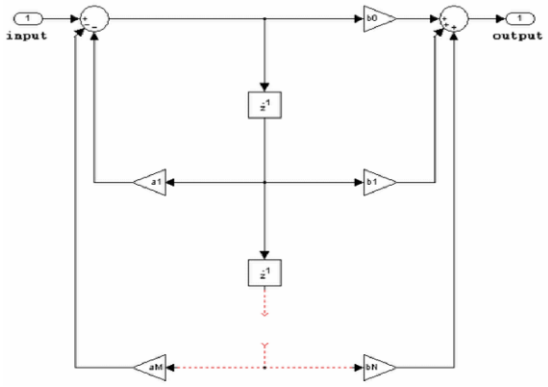

Figure 2: Generalized Direct Form II IIR Structure

A Direct Form II structure that represents the notch filter in equation 1 can be implemented utilizing hardware blocks in SG[3]. Specifically, the only SG hardware blocks that will be needed are delays, adders and multipliers. Shown in figure 3, is the SG implementation of the Direct Form II structure.

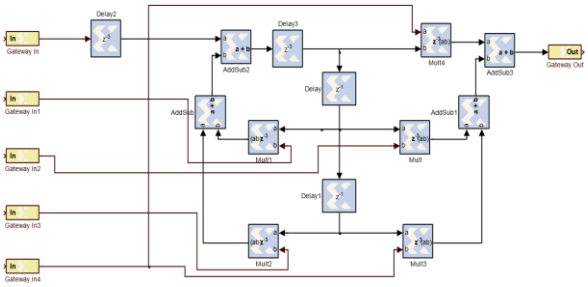

Figure 3: SG Implementation of Direct form IIR Filter

The Gateway-In blocks are essentially analog to digital converters (ADC). The Gateway-In blocks sample and quantize signals from the Simulink environment so they can be processed by digital SG hardware blocks. The Gateway-Out block is the opposite of the Gateway-In block. The Gateway-Out acts as a digital to analog converter (DAC) for outputting signals to be analyzed back to the Simulink environment.

One distinct difference between the Simulink structure in figure 2 and the SG structure in figure 3 is the addition of two extra delays (delay2 and delay3). The structure in figure 2 assumes that multipliers a1aM and b1-bN have a latency of zero. In hardware design, however, blocks sometimes have latency which can throw off the synchronization of the data pipeline. Since SG multipliers Mult1, Mult2, Mult3, Mult4 and Mult5 each have a default latency of 3 samples, it is necessary to add delay2 and delay3 which have a latency of 3 in order to maintain the integrity of equation 2. Furthermore, unlike the structure in figure 2, quantization error becomes an important issue. Specifically, each SG hardware block of figure 3 has a finite wordlength. Before setting the fixed number of bits for each block, the user must have knowledge of the range of data points being used in the design. Knowledge of the data range allows for optimal setting of integer and fractional bits.

ML402 Board

After the initial design, the next logical step is to implement and verify the functionality of the SG hardware design on an FPGA. The ML402 is a development platform used for hardware verification

that includes a Xilinx Virtex 4 FPGA, push buttons, slide switches, LEDs and an LCD. Shown in figure 4, is the ML402 board[4].

Figure 4: ML402 Board

An important feature of the ML402 is the Xilinx Virtex 4 FPGA[5]. The Virtex 4 can come in a variety of device packages. Specifically, one device package is the Virtex 4 XC4VSX35. The XC4VSX35 is a DSP focused device that comprises 192 embedded multipliers, 192 18KB of block RAM and 34,560 logic cells. This is the device that was targeted in this publication.

Hardware Analysis and Verification

Hardware synthesis is the process of translating and mapping a hardware design into a targeted architecture. The SG implementation of an IIR filter as shown in figure 3 must be synthesized to the XC4VSX35 FPGA of the ML402 board. Shown in figure 5, is a screen shot of the SG window wherein synthesis options are selected.

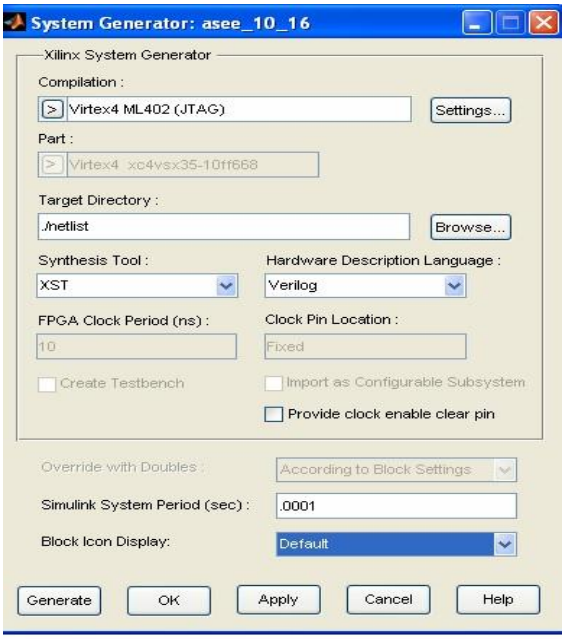

Figure 5: SG Synthesis Options

In figure 5, it is shown that the compilation device is selected as the Virtex 4 ML402 board. This selection will allow Simulink to pass signals and retrieve signals from the ML402 board during a process called Hardware Co-Simulation. Hardware Co-Simulation is a process wherein the synthesized design is running on the hardware board while the Simulink simulation model is running on the PC. This allows the output of the hardware implementation to be simultaneously compared with the output of the simulation for verification purposes. When synthesis is complete, SG provides a report which details the amount and type of FPGA resources that were needed to implement the design. The IIR filter design of figure 3 utilized 20 embedded multipliers (10% of available multipliers) and 1100 hardware slices (7% of available slices). The design utilized 20 embedded multipliers instead of the five shown in the design because of the data wordlength. For this particular synthesis, the data wordlength was set at 32 bits for each block. On the Virtex 4, however, each embedded multiplier can only perform an 18-bit by 18-bit multiplication. Therefore, in order to produce a 32-bit by 32-bit multiplication, four 18-bit multipliers were utilized for each fully pipelined multiplication. This multiplier pipeline

also increased the minimum latency of each multiplication to 5 clock cycles. Shown in figure 6, is the hardware co-simulation setup after a successful synthesis. The synthesized block is shown on the bottom wherein it is connected to the Simulink inputs and outputs in the same manner as the other models.

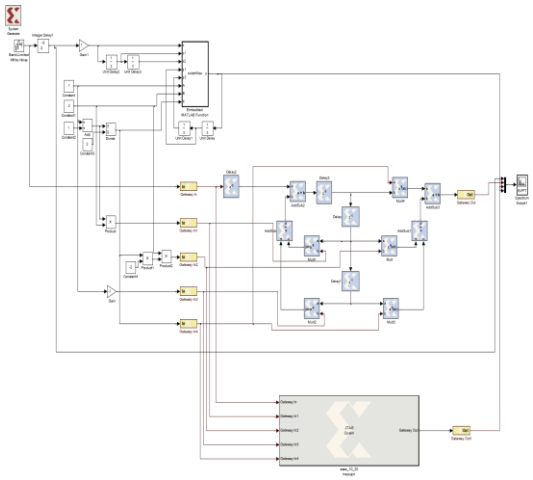

Figure 6: Simulink Model (Top), Un-Synthesized Hardware (Middle), Synthesized Hardware (Bottom)

After the synthesized block has been connected in the model, hardware co-simulation can be executed. During co-simulation, the synthesized design is downloaded to the target device (ML402 Virtex 4). The design then runs on the FPGA and communicates with the host PC via a JTAG cable. JTAG cable is a standard communication protocol that is widely used in Xilinx development boards. JTAG allows Simulink to pass the inputs through the Gateway-In to the Virtex 4. The Virtex 4 then performs the notch filter as designed and passes the output back to Simulink through the Gateway-Out. This provides a means for comparing the simulation output with the actual hardware output output. Shown in figure 7 and 8, are comparisons of the spectrums of the Simulink white noise input, Simulink notch filter output and hardware notch filter output. Figures 7 and 8 have data wordlengths of 8 bits and 12 bits respectively. The spectrum output of the 8-bit hardware implemented notch filter in figure 7 does not perform as well as the Matlab simulation. The spectrum output of the 12-bit hardware implementation notch filter in figure 8, however, performs similar to the Matlab Simulation.

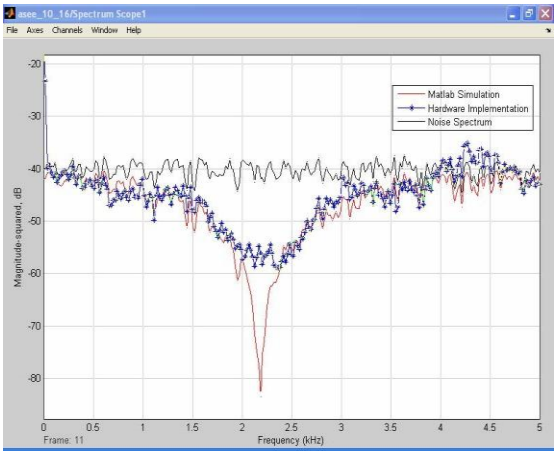

Figure 7: Filter Output Spectrum (8-bit word)

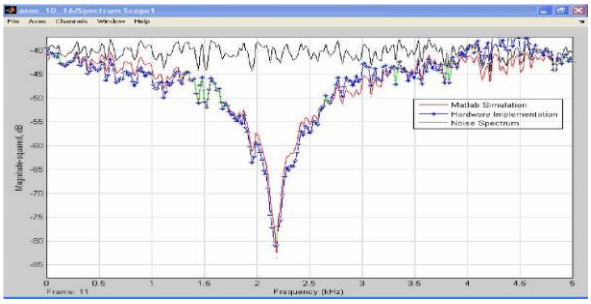

Figure 8: Filter Output Spectrum (12-bit word)

Conclusion

Teaching and understanding pertinent topics of Digital Signal Processing can no longer be limited to computer based simulation. Particularly, analysis of multirate filtering and filter structures can benefit from a hardware based approach. Hardware implementation of fundamental DSP topics introduces the student to effects that are oblivious to computer simulation. Specifically, finite wordlength effects, floating point vs. fixed point number representation and pipeline synchronization. Xilinx System Generator provides an easy to use software/hardware hybrid platform for basic to advanced hardware designs. Furthermore, students are not required to study hardware description

language, which translates to a fluent introduction of System Generator into the course.

REFERENCES

[1] Sanjit K. Mitra, "Digital Signal Processing: A Computer Based Approach", McGraw Hill, 2006.

[2] MathWorks, Simulink, http://www.mathworks.com/products/simulink/.

[3] Xilinx, System Generator, http://www.xilinx.com/ise/optional_prod/system_generator.htm.

[4] Xilinx, Virtex 4, http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/index.htm.

[5] Xilinx, ML402, http://www.xilinx.com/products/boards/ml402/reference_designs.htm.