A Design Methodology for Implementing DSP with Xilinx® System Generator for Matlab®

Matthew Ownby
Department of Electrical and Computer Engineering
Tennessee Technological University
Box. 5004
Cookeville, TN 38505 USA

Dr. Wagdy H. Mahmoud
Department of Electrical and Computer Engineering
Tennessee Technological University
Box. 5004
Cookeville, TN 38505 USA

Key Words: Simulink, DSP, Matlab, Xilinx

Abstract- **This paper presents a methodology for implementing real-time DSP applications on a reconfigurable logic platform using Xilinx® System Generator for Matlab®. The methodology aims at improving the efficiency of learning the use of such complex design system. The methodology steps will be demonstrated using DSP design examples built with Xilinx® System Generator for Matlab®. Assessment of the methodology is also discussed.**

I. INTRODUCTION

One of the most challenging processes in system design is identifying a starting point. In order to guarantee the development of such systems, we need to follow a disciplined methodology for the design process. Methodologies help us handle complex design efficiently, minimize design time, eliminate many sources of errors, reduce the need for a large number of experienced designers, minimize the manpower needed to complete the design, and generally produce optimal solution designs. The benefits of following such a methodology absolutely outweigh its development costs. Developing a methodology for the hardware implementation of complex, real-time DSP applications on a reconfigurable logic platform using Xilinx® System Generator for Matlab® is the main goal of this paper. The developed methodology provides a step-by-step template that eases the learning curve of the complex design tools.

Advances in VLSI process technology have been applied to the manufacturing of reconfigurable logic including field programmable gate arrays (FPGA) chips and helped their rapid growth in logic capacity, performance and popularity. The extreme flexibility of FPGA, the development of elaborate simulation and synthesis tools, and the widespread acceptance of hardware description languages such as VHDL and Verilog have made FPGA-based systems the medium of choice for the hardware development and implementation of high-performance, compute-intensive applications such as digital signal processing (DSP) and image processing. Field reconfigurabilty of FPGA systems allows programming their hardware for a specific instrument and/or computing system. The use of FPGA systems reduces the overall system development time. The physical and electrical testing of the hardware can be done concurrently with the detailed design and programming of the system. Design errors can be caught and easily rectified at any stage of the development process. This minimizes the effect of errors on the project schedule.

In recent years, the complexities of hardware designs have increased significantly. However, due to the ever-decreasing design cycle, and the scarcity of experienced designers, the on-time completion of such design projects has become a challenging task. Therefore, designers have been searching for more cost-effective and automated design tools. Since System Generator is presently considered cutting edge technology, a design methodology for its use to implement DSP applications into FPGA platform would be beneficial. The developed methodology addresses the following issues:

1.) Determining the design specifications
2.) Designing a system in Simulink® utilizing System Generator® design blocks
3.) Simulating the design in Simulink®
4.) System Generator® Software to produce VHDL code and test bench
5.) Simulate/ Implement VHDL code utilizing Modelsim®/ Xilinx® software

II. DETERMINING THE DESIGN SPECIFICATIONS

The DSP application chosen for the demonstration of the design methodology is a function that measures the time delay between two sine waves. This function has applications in a radar system. The goal of this function is to be able to measure the time delay between two signals ranging from 10 microseconds to 50 microseconds. Also, the design must be equipped to disregard any noise distortion. In order to detect a delay of 10 microseconds, a sine wave was set at 10 KHZ. Simulink® states that the time delay setting has to be greater then signal input

0-7803-7339-1/02/$17.00 ©2002 IEEE

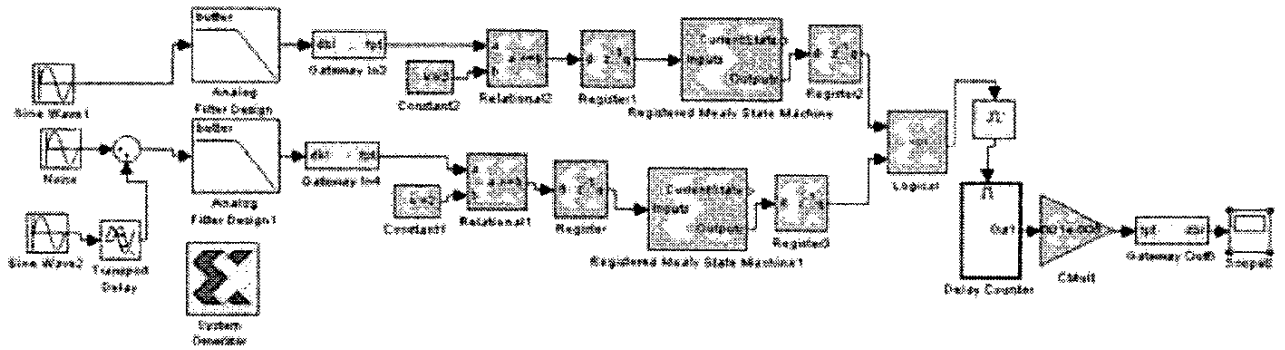

Figure 1: System Design

Based on the assumption that the signal needs to be ten times greater then the transport delay, a frequency of 10 KHZ was used for the project (10KHZ also works for the 50 microsecond case.). This first step of determining design specifications is a standard step for any design methodology.

III. DESIGN A SYSTEM IN SIMLINK® UTILIZING SYSTEM GENERATOR® DESIGN BLOCKS

Familiarity to Simulink® is helpful at this point. Simulink® is a graphical interface that permits a designer/engineer to develop a graphical model of a system using Matlab® functions. If you are familiar with Simulink® to design DSP applications, this step is a matter of using the Xilinx® design blocks instead of standard Simulink® design blocks. If Simulink® is a new experience, the Simulink® User's Guide [1] and Simulink® demos are sources of introduction for Simulink®. Figure 1 is the design of the time delay detect system in Simulink®.

As shown in Figure 1, the system consists of five major sections consisting of:

1.) Analog Input
2.) A/D converter and Comparing
3.) Set Flag
4.) Counter
5.) Output

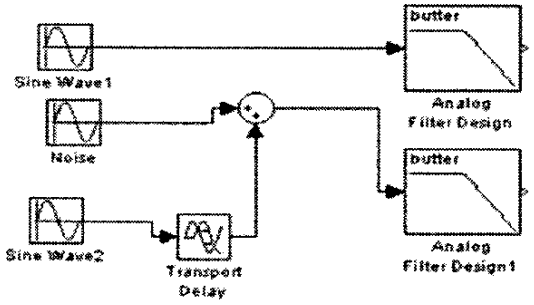

Figure 2: Analog Input Stage

1.) Analog Input

The first stage of the design is the simulated analog input stage as shown in Figure 2. The two sine wave generators are set to 10 KHZ in order to properly recognize a 10-microsecond delay. The signal generator in Simulink uses radians per second instead of hertz. Since the frequency must be approximately 62,831 radians per second (~ 10 KHZ), any setting at or above 62,831 radians per a second should be adequate for simulation. For the simulation, 62,831 radians/second was initially tested. With digital logic ranging from 3.3 to 5 volts, the simulation input signal was set to an amplitude of 5 volts. For the simulation, the amplitude of the noise was set to 0.2 volts with a frequency of 125,663 radians/second (~20 KHZ). The final stage of the analog system is the low pass filters. The purpose of a low pass filter is often to eliminate noise such as white noise, transmission line noise, etc., from a signal. For the purpose of this simulation, an eighth order Butterworth low pass filter was selected [2]. Since the input frequency is set at 62,831 radians/second, the cut off frequency needs to be greater then 62,831 and less then 125,663 radians/second (for the simulation, the cutoff frequency was set to 75,000 radians per second.)

2.) A/D Converter and Comparison Stage

The second stage of the design is the conversion of the analog (or continuous time) signal into a digital (or discrete time) signal and the comparison of the two input waves (Figure 3).

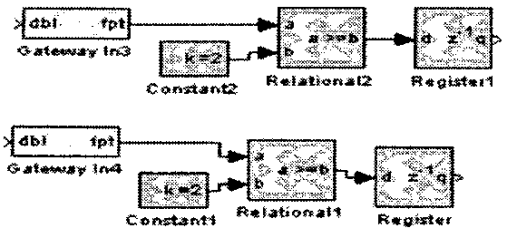

Figure 3: A/D Converter and Comparison Stage

Stages 2 through 5 all use the Xilinx® design blocks so that System Generator® can convert the design from Simulink® to VHDL. To allow an analog input for the FPGA chip, an A/D converter is needed. The A/D converter was set to signed, 16 and 8 binary point bits to allow for the analog signal to swing from +5 volts to – 5 volts. A sample period of 1e-5 was set in order to see a delay as little as 10 microseconds. The comparison blocks are used as a trigger. When the input signals are greater then 2 volts, a signal is sent from the comparer stage to the next stage to set a flag for the counter.

3.) Set Flag Stage

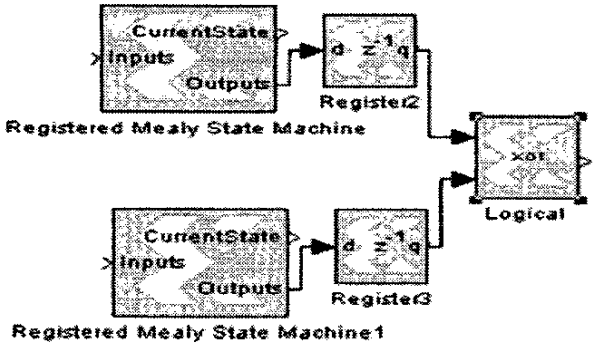

Figure 4: Set Flag Stage

In order to make the counter work properly, a flag needs to be set and held until the simulation ended or the system was reset. With any digital system, the best way to hold a value at a certain time would be to use a state machine. With the System Generator® state machine block, the state is controlled by a single bit. The best way to set and hold a flag was to set the output to logical '1' when the state machine received a signal from the comparer in stage 2. This is why two state machines are used (See Figure 4.) When the first state machine receives a signal from the comparer, a signal is sent to the next stage to start the counting. When the second comparer sends the signal, the XOR logic stops the signal from going to the counter.

4.) Counter Stage

Figure 5a shows the System view of the counter stage. Figure 5b shows the design of the delay counter. The counter stage is the heart of the design. The counter starts counting when the first signal reaches 2 volts and stops counting when the second signal reached 2 volts. Since the counter is set to count every 1e-5 samples (the same as the sample rate of the A/D converter), the counter is counting the delay between the two signals.

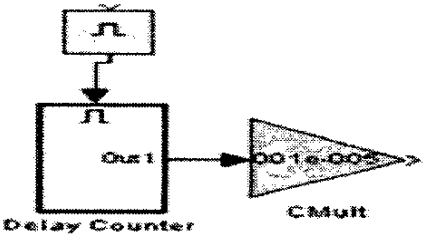

Figure 5a: Counter Stage (System View)

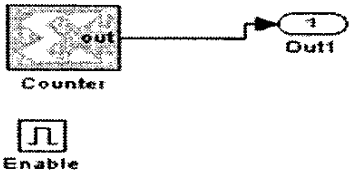

Figure 5b: Delay Counter

After the counter finishes counting, the output data has to be normalized. For example, a 50-microsecond delay will have a result from the counter of 50. To get the data to the appropriate output, the output is multiplied by a constant value of 1e-5. At this point, the data is sent to the output stage.

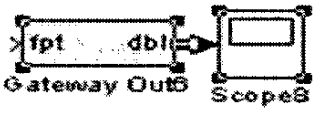

Figure 6: Output Stage

5.) Output Stage

The output stage (Figure 6) consists of a D/A converter and an output scope. This stage is not needed for the design to work. For simulation purpose in Simulink®, the D/A is needed in order to display the data to the scope.

IV. SIMULATION OF THE DESIGN IN SIMULINK®

The input signal frequency, noise frequency, and cutoff frequency for the low pass filter are set to the data selected above (62,831 radians/second, 125,663 radians/second, and 75,000 radians/second, respectively). The next step is to vary the time delay from 10 microseconds to 50 microseconds. Since the design can run continuously, the simulation run will be restricted from 0 to 0.002 seconds. Figure 7 shows the output at 10 microseconds. Figure 8 shows the output with the delay set to 50 microseconds.

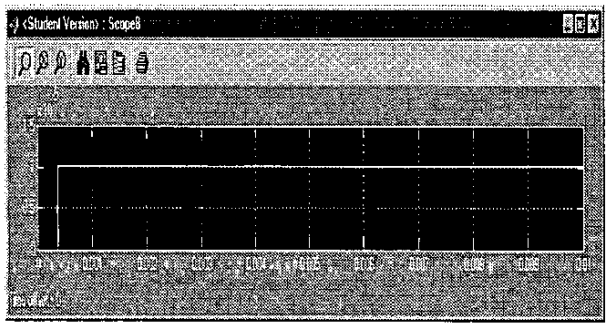

Figure 7: Output with a delay set to 10 microseconds
(Note: Y-axis shows the calculated time delay)

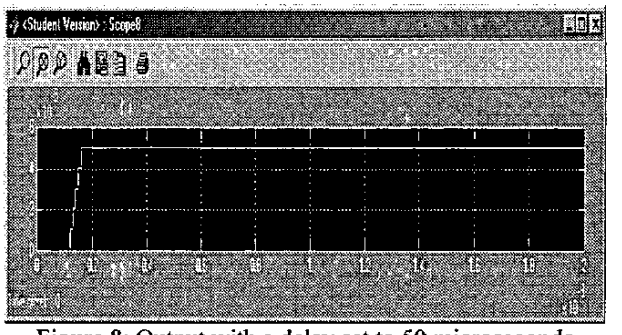

Figure 8: Output with a delay set to 50 microseconds
(Note: Y-axis shows the calculated time delay)

V. THE SYSTEM GENERATOR® SOFTWARE TO PRODUCE VHDL CODE AND TEST BENCH.

In Figure 1, a Xilinx® block label System Generator® is part of the design (Figure 9.) The System Generator® block is the program that generates VHDL code for the Simulink® model that has been created using the Xilinx® blocks.

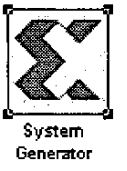

Figure 9: System Generator Xilinx Block

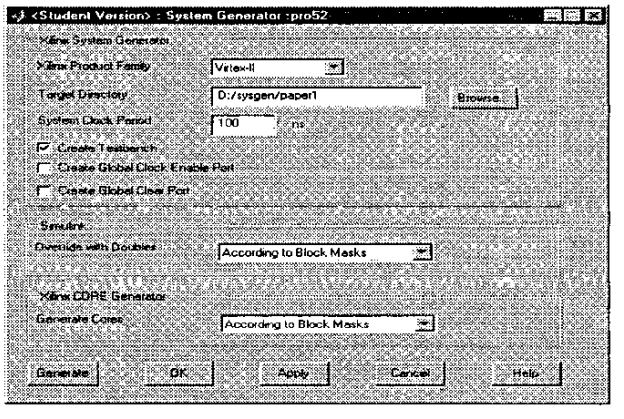

Figure 10: System Generator® Window

Figure 10 shows the System Generator® window.This window tells the Xilinx® ISE 4.2 the chip to be used and the clock speed of the design to be set. For simulation purpose, the major switch is the "Create Test bench". This switch when selected will automatically create a test bench for the system in VHDL along with creating the VHDL code of the system.

VI. SIMULATE/ IMPLEMENT VHDL CODE UTILIZING MODELSIM®/ XILINX® SOFTWARE

Along with creating the VHDL code, the System Generator® program also creates a project file for the Xilinx ISE 4.2 software. The project created allows the designer to simulate the code in Modelsim® as well as program the code into FPGA chip using the Xilinx® ISE 4.2.

VII. CONCLUSION/FUTURE WORK

Using System Generator®, digital designers can implement DSP applications into digital designs faster and easier using Simulink® and a simple design methodology.

At the time of this paper release, Xilinx® has developed a newer version of System Generator®. With a newer version of the software, the design block for System Generator® should improve and become easier to use. With this development, the design step in this methodology may need to be updated to reflect new changes in the software.

IX. REFERENCES

[1] MathWorks Inc., "Matlab® Student Version: Learning Simulink® 4, " The MathWorks Inc., 2001.

[2] C. Williams, "Designing Digital Filters," Prentice-Hall Inc., 1986.

[3] Xilinx Corp, "Xilinx System Generator for Simulink®", Version 2.1, April 2001.