TEACHING AND RESEARCH IN FPGA BASED DIGITAL SIGNAL PROCESSING USING XILINX SYSTEM GENERATOR

Shahrukh Athar Member IEEE*, Muhammad Ali Siddiqi* and *Shahid Masud* Senior Member IEEE

Department of Electrical Engineering,
Lahore University of Management Sciences,
Sector-U, D.H.A., Lahore 54792, Pakistan
E-mail: {shahrukh.athar, muhammad.siddiqi, smasud} @lums.edu.pk

ABSTRACT

This paper presents an efficient approach for the implementation of typical DSP structures studied in class or conceived during research. This scheme is beneficial where the objective is to implement the physical working of complex DSP structures or algorithms without requiring detailed knowledge of hardware design and hardware description languages. The approach is based on the Xilinx System Generator for DSP tool, which integrates itself with the MATLAB based Simulink Graphics environment and relieves the user of the textual HDL programming. In addition to introducing this scheme for teaching of DSP, some useful examples based on Delta Sigma Modulators are also presented. These modulators are selected because they are an integral part of modern Analog to Digital Converters and encompass many important Signal Processing concepts. The advantage of this approach in DSP research in terms of reducing concept-to-Silicon design time and effort is also highlighted.

Index Terms— System Generator for DSP, FPGA, Delta Sigma Modulators

1. INTRODUCTION

The fastest and efficient way to implement a Digital Signal Processing (DSP) algorithm is by designing an Application Specific Integrated Circuit (ASIC). However, ASICs are not reconfigurable and are cost effective only if developed in large enough numbers. Programmable chips offer an immediate benefit in that they are reusable and thus save Silicon and associated ASIC design time. DSP microprocessors form one such class of reusable chips. They have a fixed chip level architecture and DSP algorithms are implemented on them by writing code in the appropriate high level or assembly language. DSP applications are highly parallel in nature and this poses performance constraints on DSP microprocessors. Although DSP microprocessors incorporate parallel hardware and pipelining, they still have unique architectures which cater for various DSP algorithms requiring different levels of parallelism. Field Programmable Gate Array (FPGA) technology allows for reconfiguring at the hardware level and thus they do not encounter the constraints suffered by DSP microprocessors. Using FPGAs, customized hardware can be built for various DSP algorithms. This results in area efficient and low power DSP solutions [1].

DSP courses, taught at many universities, have an effective demonstration and laboratory component. Some universities even have a separate course on DSP laboratory. Mostly DSP microprocessor based boards are used in these demonstrations and lab exercises such as [2, 3]. Working with FPGA boards at a lower level of abstraction requires pre-requisite knowledge of Digital System Design and Hardware Description Languages (HDLs) such as Verilog or VHDL. In the diverse Electrical Engineering curriculum as prevalent these days, students taking a DSP course may not have this HDL knowledge. This has hindered the use of FPGA boards as development platforms in DSP courses [4]. Similarly, DSP researchers who lack the requisite knowledge about HDLs seldom use FPGAs as platforms for conducting research.

The availability of FPGA design tools that work at a higher level of abstraction allow users to work with FPGAs without detailed knowledge of HDLs. One such tool is the LabVIEW FPGA. The use of this tool in DSP teaching is discussed in [4]. Another tool that also works at a higher level of abstraction and is conducive for those having prior knowledge of MATLAB and Simulink is the Xilinx System Generator for DSP. A basic DSP design methodology using this tool has been presented in [5]. In this paper we propose the use of the System Generator tool for FPGA implementation in DSP teaching and research by using Delta Sigma ($\Delta\Sigma$) Modulators as examples.

The rest of the paper is organized in the following manner. Section-2 gives a brief introduction to $\Delta\Sigma$ Modulators. Section-3 describes the Xilinx System Generator for DSP in some detail. Section-4 elaborates the use of System Generator in DSP Teaching while Section-5

describes the use of this tool in conducting DSP research. Section-6 concludes this discussion.

2. BRIEF INTRODUCTION TO DELTA SIGMA MODULATION

ΔΣ Converters encompass many crucial signal processing concepts such as sampling and quantization, feedback loops, multiple order structures, effects of oversampling, noise shaping, word length growth, sum accumulation, etc. They have been presented as effective DSP teaching tools in [6]. We will use the example of FPGA implementation of ΔΣ Modulators in DSP teaching and research.

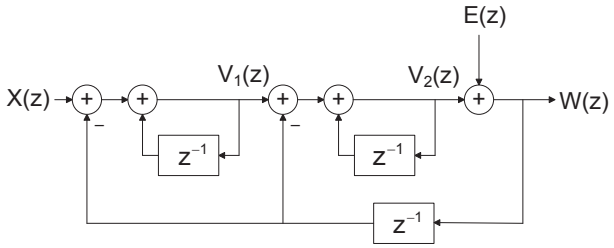

Figure 1: A 2nd Order ΔΣ Modulator

ΔΣ Modulators oversample an input signal thereby reducing the overall noise floor and increasing the distance between the spectral images of the signal. They use noise shaping to push the quantization noise away from the band of interest from where it can be removed by the use of an appropriate filter. ΔΣ Modulators find widespread use in Analog to Digital Converters, Digital to Analog Converters, Frequency Synthesizers, etc. Detailed information on ΔΣ Modulators can be found in [7, 8]. Figure 1 shows a 2nd order ΔΣ Modulator (DSM2) which uses a 2nd order accumulator by cascading two 1st order accumulators. The order of the accumulator defines the order of the ΔΣ Modulator. It uses a single bit quantizer which is represented by a noise source E(z) in Figure 1. The implementation of recursive DSP structures such as Figure 1 is considered complex for hardware implementation. This paper presents a quick top-down methodology that can capture such designs with accuracy using the System Generator.

3. OVERVIEW OF THE XILINX SYSTEM GENERATOR

The Xilinx System Generator for DSP is a system level modeling and design tool that facilitates FPGA design and has the ability to work at a higher level of abstraction [9]. It enables the use of the MathWorks graphical model based Simulink design environment for FPGA design. The System Generator integrates itself with Simulink and FPGA designs are captured by using the Xilinx specific blocksets. Thus, designing a hardware model in Simulink is as simple as designing any other Simulink model with the only difference being the use of Xilinx blocksets instead of those found in Simulink. The System Generator provides many DSP building blocks in the form the Xilinx DSP blockset for the Simulink environment. The variety in this blockset ranges from common DSP blocks such as adders, multipliers, registers etc to more complex blocks such as FFTs, filters, memories, forward error correction etc [9]. Thus, previous experience with low level system design and HDLs is not required when using this tool. The System Generator uses the Xilinx ISE software and IP core generators to convert a designed model into the equivalent HDL code. The remaining FPGA implementation steps including synthesis, place and route, etc. are automatically performed to generate a bit file that is downloaded on to the FPGA.

The System Generator can be used in different modes, two of which are: (i) *Hardware Emulation* and (ii) *Hardware Co-Simulation*. In Hardware Emulation mode, the hardware model can be run on the computer as if it was running on an FPGA and this can be regarded as hardware simulation. Simulation done using Simulink differs from actual FPGA hardware implementation results since Simulink works with floating point numbers whereas FPGAs use fixed point format. However, there is no discrepancy between Hardware Emulation and hardware implementation results since System Generator uses the fixed point format. In the Hardware Co-Simulation mode, the System Generator generates the HDL library block of the hardware model. A new Simulink model is then made containing this library block along with sources and sinks of Simulink. After the generated bit file is downloaded on to the FPGA, the input to the device can be given from Simulink and the device output can be received back in Simulink. This enables extensive testing as the data from the FPGA can be directly exported to the MATLAB environment where spectral analysis, Signal to Quantization Noise Ratio (SQNR) estimation, etc. can be performed.

4. USING SYSTEM GENERATOR FOR DSP TEACHING

The 2nd order ΔΣ Modulator (DSM2) of Figure 1 was implemented on a Digilent Spartan-3E Starter Kit that has the Xilinx Spartan-3E XC3S500E FPGA device with an on-board clock of 50 MHz [10]. This board is supported by the Xilinx System Generator for DSP. The System Generator model of DSM2 can be seen in Figure 2. The *Gateway-In* and *Gateway-Out* ports interface the Simulink double data type and the FPGA fixed point environments. Only Xilinx blockset blocks are used between the *Gateway-In* and *Gateway-Out* ports. The accumulators were implemented by using a delay element and an AddSub block in the adder mode. The difference signals were generated by using the AddSub block in the subtraction mode. The one bit quantizer was implemented by using a relational operator with a multiplexer. The System Generator allows for setting the latency of various blocks and hence delays can be

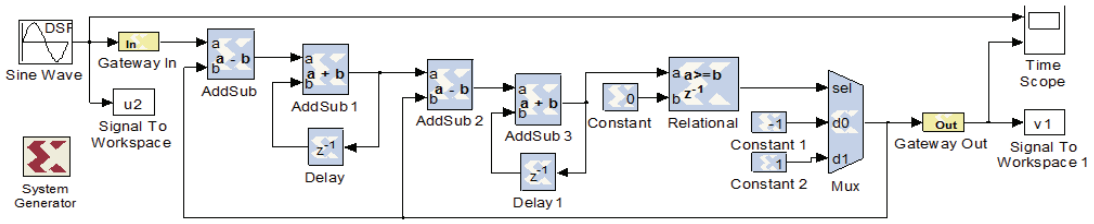

Figure 2: Xilinx System Generator hardware implementation of the 2nd order ΔΣ modulator (DSM2)

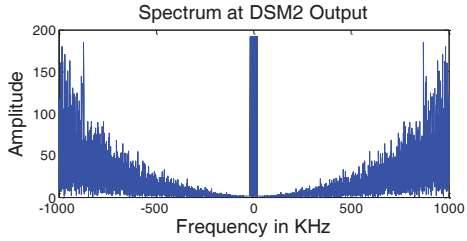

Figure 3: DSM2 output signal spectrum at OSR 256

incorporated in feedback loops. It can be seen from Figure 2 that the input signal was given to DSM2 from Simulink. In this particular case a signal of bandwidth 20 KHz was used as input and it was composed of five different sinusoidal components. The output signal of DSM2 was received back in Simulink. This was made possible due to the Hardware Co-Simulation mode of operation.

Both the input and output signals were exported to MATLAB. The spectral plot of the DSM2 output from the FPGA was taken in MATLAB and this can be seen in Figure 3. It is evident that students can operate a DSP design on FPGA devices and at the same time they can perform analysis of data received from the FPGA device in MATLAB, without going to a lower level of abstraction.

Various experiments can be designed by changing parameters of the ΔΣ Modulator such as order of the accumulator, resolution of the quantizer and Oversampling Ratio (OSR) of the modulator. Students can compare the performance of various designs by estimating the SQNR, the dynamic range and by observing the resulting spectral plots in MATLAB, specifically looking for spectral purity in the band of interest and noise shaping elsewhere.

The Xilinx System Generator has a short learning curve and those already familiar with the Simulink environment can start designing using System Generator with ease. Laboratory experiments similar to the implementation of DSM2 can be developed for DSP courses. Adding these labs to the MATLAB based labs of these courses will have two main benefits, (i) students will experience working with real time DSP with hardware in the loop and (ii) students can be introduced to the FPGA platform which may be new to them. This may also enable students to find out if they are interested in the field of hardware design and whether they should take advanced courses in digital system design.

5. USING SYSTEM GENERATOR FOR DSP RESEARCH

To demonstrate the validity of this approach in DSP research, we have designed and implemented a 2nd order Adaptive ΔΣ Modulator (ADSM2) on a FPGA device in an earlier work [11]. The implementation was done using the Xilinx System Generator. This tool reduced our design time and enabled us to implement different types of ΔΣ Modulators for comparison with the adaptive version ADSM2. The System Generator allowed us to work with different FPGA platforms and eventually we decided in favor of using the Digilent Spartan-3E Starter Kit. Figure 4 shows the System Generator model of ADSM2 which consists of a *Modulator Stage* and an *Adaptation Stage*.

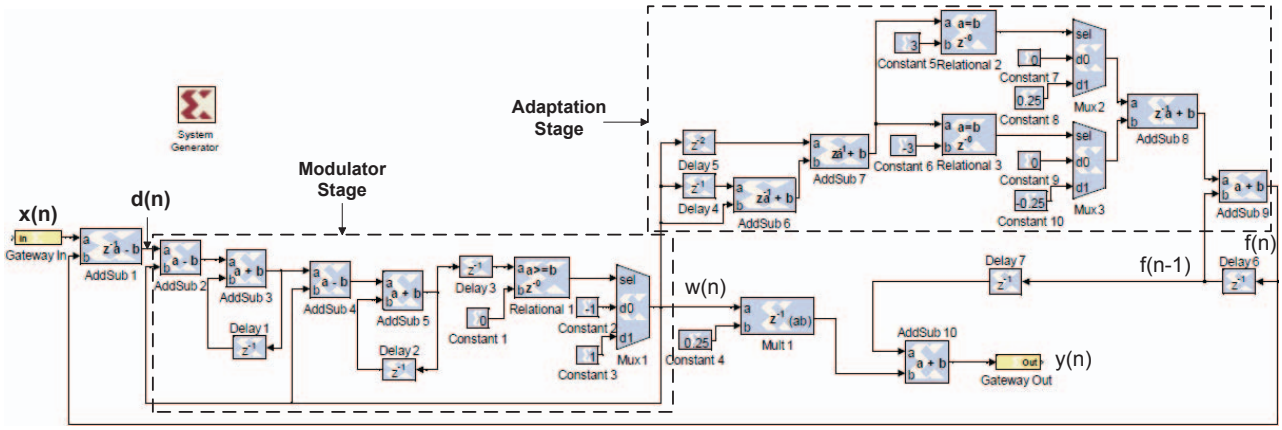

Figure 4: Xilinx System Generator hardware implementation of the 2nd order Adaptive ΔΣ modulator with one bit quantization (ADSM2)

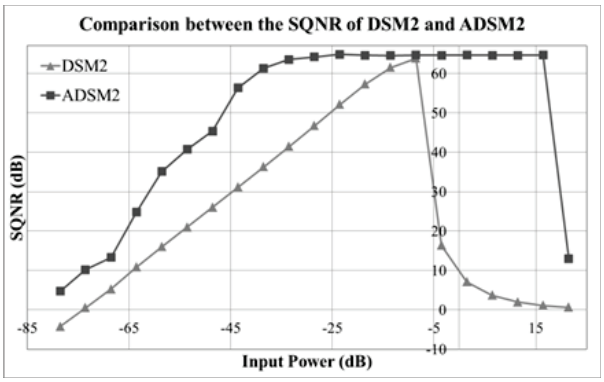

Figure 5: SQNR analysis of DSM2 and ADSM2 at OSR 512

The Modulator Stage consists of DSM2, already presented in Section-4 while the Adaptation Stage generates an adaptive feedback signal that follows the input signal. The difference of the input signal and the adaptive feedback signal becomes the input to the modulator stage. Thus, the modulator stage (DSM2) operates on a difference signal that is in a reduced range leading to the generation of less quantization noise in ADSM2 as compared to a standalone DSM2. This leads to a superior SQNR performance by ADSM2. Figure 5 compares the SQNR performance of ADSM2 and DSM2 at an OSR of 512. It clearly demonstrates that ADSM2 is the better of the two modulators not only in terms of SQNR performance but also in terms of dynamic range. The adaptive feedback signal was generated using the adaptation algorithm presented in [12]. It can be seen in Figure 4 that the ADSM2 model was developed entirely of blocks from the Xilinx blockset of the System Generator tool. The use of Hardware Co-Simulation mode enabled the input signal to be given from Simulink and the output of ADSM2 was received back in Simulink. The SQNR analysis that resulted in Figure 5 was done by exporting the results to MATLAB and performing computations there.

Some other design examples about the use of the Xilinx System Generator tool in DSP research can be found in [13] and [14]. The former uses this tool in the FPGA implementation of an unmanned vehicle control system whereas the later uses it in the FPGA implementation of parallel 2-D MRI image filtering.

6. CONCLUSION

The use of the Xilinx System Generator tool for DSP education and research is presented. It is shown that this tool is ideal for developing FPGA based hardware without the requirement of learning HDLs and Hardware Design. This claim was corroborated with the help of some ΔΣ Modulator based implementations. A simple 2nd order ΔΣ Modulator was implemented to show how the System Generator can be used in Signal Processing education while an advanced 2nd order Adaptive ΔΣ Modulator was developed to demonstrate its importance in research. The implementation of various designs was carried out on a Xilinx Spartan-3E FPGA. The System Generator facilitates extensive testing to take place due to the Hardware Co-Simulation mode of operation which allows for input signals to be given from Simulink and receiving output signals back in Simulink. The use of the System Generator reduces design time and allows working with different FPGA platforms. It also enables the design and quick implementation of various designs that helps in making useful comparisons.

7. REFERENCES

[1] R. Woods, J. McAllister, G. Lightbody and Y. Li, "FPGA based Implementation of Signal Processing Systems," A John Wiley & Sons Inc, publication, 2008.

[2] Cameron H. G. Wright, Thad B. Welch, Delores M. Etter and Michael G. Morrow, "A systematic model for teaching DSP," *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol.4, no., pp.IV-4140-IV-4143, 13-17 May 2002.

[3] Andres Kwasinski, "In-class demonstrations with a portable laboratory for teaching DSP to Computer Engineering majors," *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, vol., no., pp.2896-2899, 22-27 May 2011.

[4] N. Kehtarnavaz and S. Mahotra, "FPGA implementation made easy for applied digital signal processing courses," *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, vol., no., pp.2892-2895, 22-27 May 2011.

[5] M. Ownby and W.H. Mahmoud, "A design methodology for implementing DSP with Xilinx® System Generator for Matlab®," *System Theory, 2003. Proceedings of the 35th Southeastern Symposium on*, vol., no., pp. 404- 408, 16-18 March 2003.

[6] R. Saint-Nom and D. Jacoby, "Sigma-Delta Converters as a SP Teaching Tool," *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol.2, no., pp.II, 14-19 May 2006.

[7] G. I. Bourdopoulus, A. Pnevmatikakis, V. Anastassopoulos and T. L. Deliyannis, "Delta-Sigma Modulators: Modeling, Design and Applications," Imperial College Press, 2003.

[8] R. Schreier & G. Temes, "Understanding Delta-Sigma Data Converters," A John Wiley & Sons Inc., publication, 2005.

[9] Xilinx System Generator for DSP User Guide, r10.1.1, April 2008.

[10] Digilent Spartan-3E Starter Board with Xilinx XC3S500E FPGA (http://digilentinc.com/Products/Detail.cfm?NavPath=2,400,792&Prod=S3EBOARD)

[11] S. Athar, M.A. Siddiqi and S. Masud, "Design and FPGA Implementation of a 2nd Order Adaptive Delta Sigma Modulator with One Bit Quantization," *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, vol., no., pp.388-393, Aug. 31 2010-Sept. 2 2010

[12] C. M. Zierhofer, "Adaptive sigma-delta modulation with one-bit quantization," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol.47, no.5, pp.408-415, 2000.

[13] S.N. Murthy, W. Alvis, R. Shirodkar, K. Valavanis and W. Moreno, "Methodology for implementation of unmanned vehicle control on FPGA using system generator," *Devices, Circuits and Systems, 2008. ICCDCS 2008. 7th International Caribbean Conference on*, vol., no., pp.1-6, 28-30 April 2008

[14] S. Hasan, A. Yakovlev and S. Boussakta, "Performance efficient FPGA implementation of parallel 2-D MRI image filtering algorithms using Xilinx system generator," *Communication Systems Networks and Digital Signal Processing (CSNDSP), 2010 7th International Symposium on*, vol., no., pp.765-769, 21-23 July 2010