AC 2011-754: DIGITAL DESIGN MEETS DSP

Christopher S Greene, University of Saint Thomas

Christopher Greene received his Ph.D. in Electrical Engineering from the Massachusetts Institute of Technology (MIT) and proceeded to a 25 year career in industry. At Honeywell, he did research on adaptive control and navigation systems before becoming Program Manager for several large aerospace programs. At Horton and Nexen, he was responsible for the development of industrial control products. In 2002, Dr. Greene joined the engineering department at the University of St. Thomas where he currently is the Program Director for Electrical Engineering and teaches classes in signals and systems, controls and digital design as well as the Senior Design class.

PAUL IAN NYOMBI, UNIVERSITY OF STTHOMAS

Paul Nyombi, is originally from Uganda, East Africa. He is currently senior at the University of St. Thomas persuing a double major in Electrical Engineering and Computer Science. Through his education at St. Thomas, he's been able to explore the various engineering fields through the different courses he has taken. He has a developed a high interest in high power systems (distribution and transmission) especially as regards the incorporation of renewable technology on the power grid. Over 10 months ago, he joined Xcel Energy (internship) from where he has enjoyed being challenged and stretched intellectually by experienced engineers. He intends to advance to the graduate level after his undergraduate course.

Digital Design meets DSP

One common class taken early in most undergraduate electrical engineering curricula is Digital Design. This course typically teaches students to design relatively simple digital circuits using the concepts of combinatorial and sequential design including the use of registers. However, more complicated issues such as the use of SPI and I^2C interfaces are typically beyond students experience until later in the curricula.

Background

At the University of St. Thomas (Minnesota), Signals and Systems is typically taken after Digital Design but frequently before or concurrently with a course teaching more complicated interfaces such as SPI and I^2C. Thus, students in Signals and Systems frequently have had Digital Design but do not yet fully understand the more complicated (and hardware efficient) interfaces to modern A/D and D/A chips. This leaves the instructor who wants to provide hands-on experience with using filtering in Signals and Systems with a dilemma. They can either use 'simple' hardware connections and provide 'canned' software to drive the interface or they can attempt to teach the complexities of SPI or I^2C interfaces. Our experience has been that these interfaces require significant debugging time which, while valuable in the long run, diverts attention from the Signal and Systems concepts being taught.

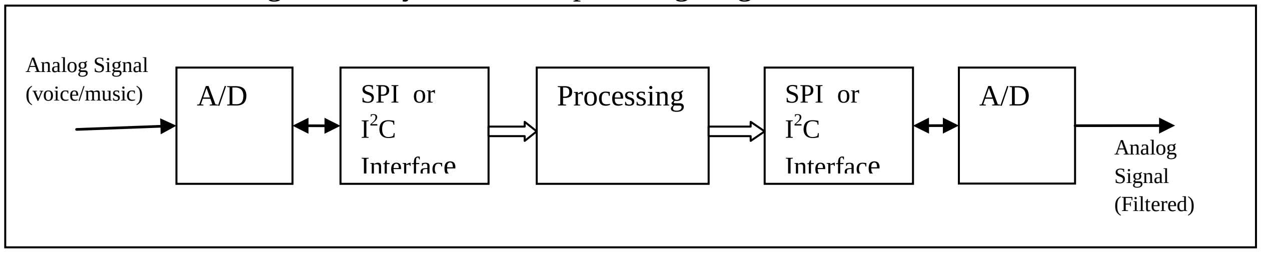

Figure 1 - "Preferred" configuration

In this paper we discuss a third alternative that has been developed, namely, to use the far simpler (in a logic sense), but older, 8-bit parallel interface A/D and D/A chips. In this way, we can decouple learning Signals and Systems from learning chip level interfacing such as SPI and I^2C. Most students who have completed Digital Design understand registers and the relatively simple sequential logic needed to trigger conversion in the A/D chip. And the D/A chip can be directly driven from a simple register. The resulting configuration is shown in Figure 2 below.

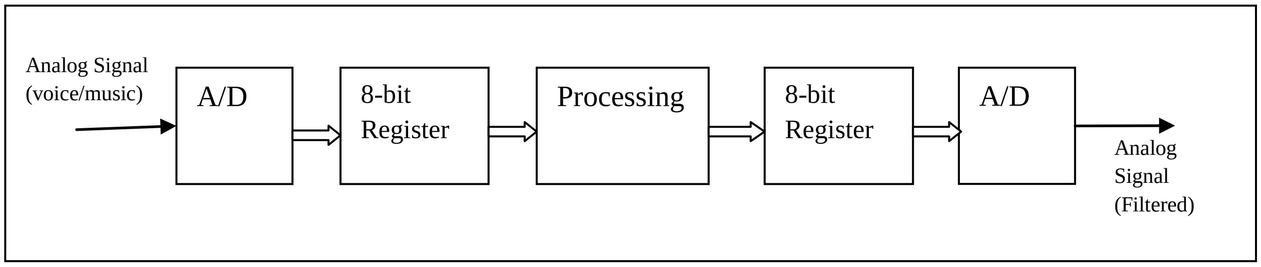

Figure 2 - Simpler configuration

This design allows any student who has completed Digital Design to fully implement complex DSP algorithms and directly see the effects of finite bit sizes and other implementation details. This is especially true when this hardware is teamed up with Xilinx's System Generator and Matlab's Simulink. The sample demonstrated in this paper utilizes the Xilinx FPGA and Mathwork's Matlab with Simulink and the Xilinx System Generator to demonstrate Model Based Design of a DSP filtering algorithm. This allows students to progress from floating point analysis through fixed point simulation and finally programming the Xilinx FPGA to allow the student to perform hardware testing and demonstration of their algorithm in a fixed point, FPGA environment. In addition, this experiment allows students to see how the Model Based Design philosophy permits a design to progress from theoretical analysis through to hardware implementation.

Hardware

The hardware of the FPGA digital filter system used is built around a Digilent Spartan 3 XC3S200 board with an expansion bread board connected for easier I/O and analog circuit construction. The system functions as follows:

- An analog input signal is fed into an analog-to-digital converter IC, converting the input to an 8-bit parallel signal.
- The digitized input signal is fed (in parallel) into eight of the Spartan 3's I/O ports.
- The signal is processed by the digital filter that has been programmed on the FPGA, and output through eight additional I/O ports forming an 8-bit register.
- The filtered digital output is then passed to the digital-to-analog converter IC.

The versatility of this system is its greatest strength – any digital filter that can be built and simulated using Xilinx's Simulink blockset can be implemented on this system. Any analog input can be used, with adequate output resolution for frequencies up to about 100 kHz.

The major hardware components used for this project included the Maxim ADC0820 analog-to-digital converter IC with 0 to +5V reference and 8-bit output and a Maxim Integrated Products MX7224 digital-to-analog converter IC. These chips have several operating modes. We selected the simplest modes in which the inputs and outputs look most like simple registers.

Maxim IC products were used primarily for two reasons; they were inexpensive and simple to interface.

Another important hardware component used was the 50MHz Digilent Spartan 3 board[1] with an 80-pin Breadboard 1 expansion board connected as shown below. This board uses the Xilinx XC3S200 and provides several useful IO facilities that are often used in introductory Digital Design courses. Other boards that allow access to the digital inputs and outputs can of course be used also.

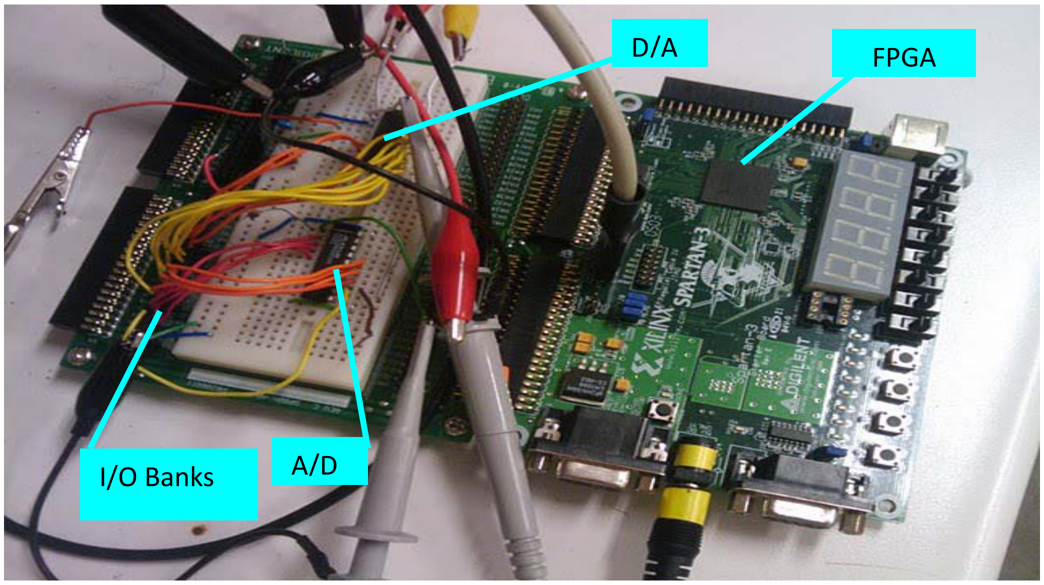

Figure 3 - Spartan 3 system

IC Control VHDL:

The ADC and DAC chips require digital control lines to enable their conversion processes. The ADC chip has four data control ports and makes use of tri-state gates for its conversion process. By setting these data lines either high or low, the chips control the entire process of data acquisition. One of the control data lines is the status output data line. It issues "WAIT" states to both the ADC chip and the reading in of data by the Xilinx FPGA to which it's connected. On the rising edge of the status data line signal, a read instruction is issued to the 8 data ports of the chip by the Xilinx processor. This next stage of data conversion is uninterruptable. The ADC can also be used to measure high speed signals without the need for an external sample and hold mechanism. The typical time of this chip for reading the signal and making it available to the 8 bit ports is about 100ns.

Like the ADC, the DAC chip has an 8 bit resolution. Data conversion is also controlled using four data control signals, that essentially control the two main registers (Input and DAC registers) associated with this chip. The active low LDAC and WR signals control the DAC register from which data is output. The WR and CS signals control the data transfer between the input register and DAC register. The RESET signal is used for clearing both registers.

The ADC and DAC chips are interfaced with the Xilinx FPGA in parallel. A program was generated using a stand-alone VHDL program (available through http://courseweb.stthomas.edu/csgreene/Teaching_Material.htm) that ran in the Simulink simulation inside a Xilinx Black Box. After implementation, the VHDL runs on the FPGA and creates two identical control lines for the ADC's read (RD) and chip select (CS) inputs, which determine the effective sampling rate of the analog input shown in Figure 4.

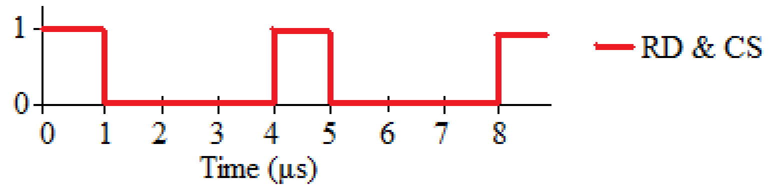

Figure 4 - Waveform of ADC control signals

The 4µs, or 250 kHz, resultant sample rate is within 1µs of the fastest achievable rate and is generated by dividing down the Spartan 3's 50MHz system clock to a 1MHz (1µs) clock, and then drawing a counter off of that. Twenty five percent (25%) duty cycle ensures that conversion of the input signal is complete before reading the digitized values to the Xilinx FPGA. It would be possible to increase sampling speed significantly by testing for conversion completion but this added complexity was not consistent with maximum simplicity.

The VHDL created also generates four control lines for the DAC shown in Figure 5. These signals are simply held high or low as briefly explained earlier, to run the chip in its transparent conversion mode:

Signal Name	Value
CS	0
WR	0
LDAC	0
RESET	1

Figure 5 - Values of DAC control signals for transparent mode

Also specified in the VHDL is the Simulink clock signal. The associated clock definitions had to adhere to the following Xilinx generator requirements and restrictions:

- Clock and clock enable ports in black box HDL should be expressed as follows: Clock and clock enables must appear as pairs (i.e., for every clock, there is a corresponding clock enable, and vice-versa). Although a black box may have more than one clock port, a single clock source is used to drive all the clock ports. Only the clock enable rates differ.
- Each clock name (respectively, clock enable name) must contain the substring clk, for example my_clk_1 and my_ce_1. This is done under the entity section of the VHDL.
- The name of a clock enable must be the same as that for the corresponding clock, but with ce substituted for clk. For example, if the clock is named src_clk_1, then the clock enable must be named src_ce_1.
- Falling-edge triggered output data cannot be used[2].

Importing VHDL to Simulink

After testing and ensuring that the VHDL worked properly, it was imported to Simulink using the Xilinx block set. In the process of VHDL importation, an M-file, defining the timing parameters of the black box (that contains the VHDL) was created. The M-file has a "config" substring such as "TestSystem_Gen_config.m". By default the sample rates of the black box's output ports match the black box's input sample rate of 50 MHz (which is specified by the Xilinx processor defined in the process). Additionally, the block's port data types are set to "unfixed" with no fractional bits. The created M-file (Matlab code) provided the platform from where all the above could be varied. That is the black box's port sample frequencies, data type and the fractional bit sizes could all be changed as found necessary.

Port's sample frequencies, data type and port fractional bit sizes were adjusted to match this project's requirements:

- The "DATA_OUT" port data type was changed from the default "UFix 8_0" to "Fix 8_7" - fixed-point data type with seven fractional bits
- As mentioned earlier, the ADC and DAC were tuned to run at 250 kHz. This was attained through the hand coded imported VHDL for the Black Box element. To ensure synchronism in data transfer, "DATA_OUT" port's sample rate was scaled down from the default 50 MHz to 250 kHz.

These changes were made in the M-code generated in order for the Simulink system to operate. The black box port data types can also be easily changed. To alter the frequencies, the "setRate" Matlab command was used. It doesn't define the rate at which the desired port should run but instead it gives the positive integral value that is the ratio of the desired input sample period to that of the simulink system clock period (20ns for the 50 MHz) defined by the System Generator dialog box. Alternatively, it can be defined as the ratio of the simulink system clock frequency to the desired output frequency. For illustration, the following alteration was made to change the "DATA_OUT" port frequency: The Matlab code section "uniqueInputRates = unique(this_block.getInputRates);" was commented out and replaced with "DATA_OUT_port.setRate(200);" statement where 200 is the positive ratio.

System Generator dialogue box

A System Generator block was required in any System Generator project. The System Generator dialog box has two timing parameters that were of concern to us. First is the "FPGA clock period" which is always given in nanoseconds. It provides the global period constraint used for the Xilinx implementation tools. And it's a non integral value though all the other multicycle paths have to be confined to its integer multiples. For this project, an FPGA clock frequency of 50 MHz (period of 20 ns) was selected, which was the clock frequency for the board used.

Secondly, the system generator dialog box has the "Simulink system period" defined in seconds. It is the greatest common divisor of all the sample periods that are used in the model, which are set in the block diagrams used (such as the Gateway-out and in). Thus, the Gateway-in sample period cannot be less than the "Simulink system period". The System Generator block can also be used to display the desired type of information on the block icons which by default is set to "Default". From the pull down menu were Normalized Sample Periods, Pipeline stages, HDL port names, Input data types, Output data types, and Sample frequencies in MHz. The "sample frequency" option was selected to display the input and output block sample periods when the model was run.

Experiment -- Low Pass Filter

The purpose of the hardware set up described above is to permit students in Signal and Systems to perform filter design and experimentation without sophisticated hardware interfaces. In order to demonstrate this, a low pass filter in designed. To meet the hardware requirements of the FPGA on to which the filter was to be implemented, a sampling frequency of 250 kHz was used. In order to minimize manual recopying of filter parameters, a fifth order filter was designed. The cut off frequency was set at 0.01MHz as shown beside in Figure 6.

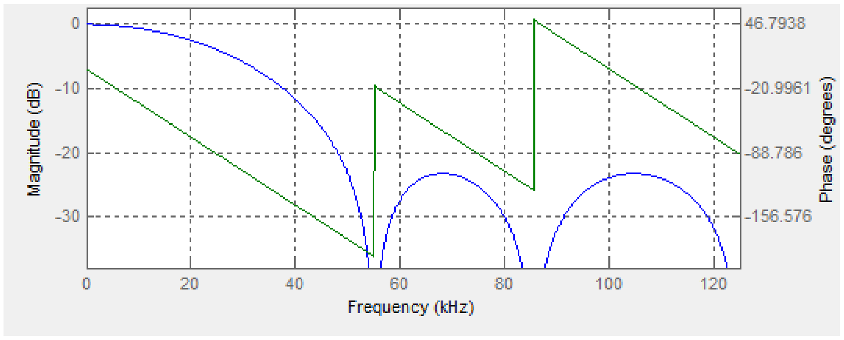

Figure 6 – Bode Plot of a 5th order low pass filter with cut-off of 25 kHz

The filter coefficients were obtained from the Matlab Filter Design Toolkit and using the Xilinx block set, a series of add/subtract blocks were combined to reproduce the created filter in the form of blocks that were later combined into a subsystem. See Figure 7.

The filter was first tested separately by running a sine wave through the generated filter subsystem block and displaying it on the Simulink scope. Gateway-in block and gateway-out blocks were used to facilitate the data communication between regular Simulink and Xilinx blocks.

Finally all the different design parts were put together, simulations were run and the whole Simulink block system, shown in Figure 7, was tested.

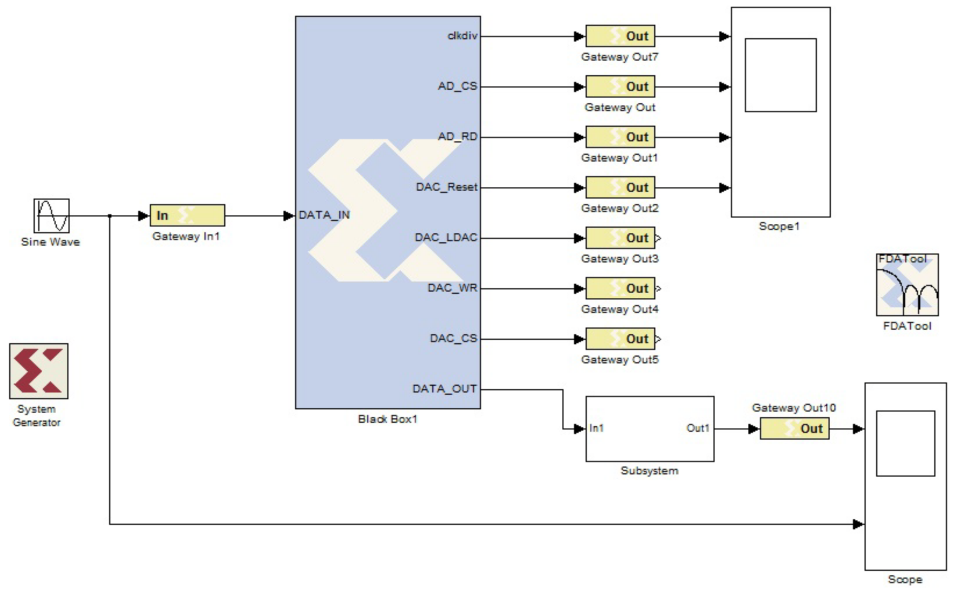

Figure 7 - Block diagram digital filter system

Xilinx ISE Project

Post System Generator:

After the System Generator compiled the Simulink model into a Xilinx ISE Project, the system was ready to be implemented on the Spartan 3 FPGA. Several steps had to be taken in order for the programming to be successful.

The user constraints file (UCF), which contains the pin locations for all generated I/O signals, must be consistent with the selected FPGA. This was originally accomplished by way of assigning each pin’s individual location through Xilinx’s PlanAhead Post-Synthesis I/O Pin Planning application. As long as the hardware stays the same, though, the contents of this .ucf file may be used in each new ISE project.

The Synthesize, Implement Design, and Generate Programming File processes were now run. With the Spartan 3 board connected to the computer via the standard 6-pin to serial connector cable, Configure Target Device was run to start the FPGA programming process.
A special option must be selected (Edit –> Preferences –> iMPACT –> Configuration Preferences –> Use HIGHZ instead of BYPASS) to program this specific board.
Within a Boundary Scan, Initialize Chain was selected and the XC3S200 core assigned the project’s .BIT file. The FPGA was now programmed and the digital filter tested using a signal generator and oscilloscope. The output signal was noisy and this was largely due to the noise resulting from the A/D and D/A chips and the electronic devices and connecting wires used.

Assessment

Assessment to date has been limited. Students helped develop this experiment and discussions with the students indicate the validity of the approach. To quote one of the students involved in this project, "Implementation of this filter system was not particularly difficult though it required a good level of understanding of both Xilinx and Matlab software. The whole system is largely made up of block diagrams which most students find easier to understand. It's only in a few places that hand-coded program code was used – VHDL for driving the ADC chip. And once one gets a good handle on how the system design was to flow, it becomes much simpler to build the different system parts, debug them, and then combine them to form the bigger filter system. Because of time constraints, we never got a chance to try implementing a low order BPF and notch filter, both of which weren't difficult. The only place in the whole software design that required changing was the filter portion of it." However, to date, no systematic assessment has been done. This needs to be the next step in this development. Since the Signals and Systems course does not have an associated lab, this experiment will be presented as an extended homework problem. At the University of St. Thomas School of Engineering, our students regularly have access to all electronics lab facilities and thus this is not expected to be a major obstacle.

Conclusion

We have described a development system that would allow a student who had successfully completed an introductory digital design course to design, develop and perform hardware demonstrations of filtering algorithms common in an introductory Signals and Systems course without becoming proficient in common serial interfaces such as I2C or SPI. Since the Xilinx Spartan FPGA and an introduction to VHDL are taught in the preceding digital design course, there is little additional material that needs to be taught. Also, as part of Signals and Systems, we already cover the use of Matlab and Sinulink and thus this lab should be a fairly straightforward extension of the use of Matlab's Simulink and Xilinx's System Generator technology to help students better understand filter design and DSP's. The assessment will of course include verification of these.

To test the hardware, a fifth order low pass filter, with a sampling frequency of 250kHz was built using the Matlab Filter Design Tool Kit. Its coefficients were used to implement the filter using addition and multiplication blocks from the Xilinx block set – for easier interface and data communication with the black box.

Lastly, after testing the whole Simulink filter system, using the Xilinx System Generator block, VHDL for the whole system was generated. It was successfully compiled and then implemented on the Xilinx processor. The hardware was powered accordingly and then the different input signals were successfully run on the system.

Bibliographic Information

1 Spartan-3 Starter Kit Board User Guide, Digilent Inc, May 13, 2005, www.digilentinc.com

2 Xilinx Inc, System Generator for DSP v12.3, www.xilinx.com/support/documentation