# Low Cost System Identification Tool

Low Cost System Identification Tool Team
Jim Susong, Jose Benavides, Frank Cantua, Pourya Shahnaz

Advisor: Professor A. A. Rodriguez

December 1, 2005

Table of Contents


1. Project Definition
   - 1.1. Introduction and Scope ..... 5
   - 1.2. Requirements ..... 5
   - 1.3. Desired Attributes ..... 5
2. Technical Work Completed
   - 2.1. Research ..... 6
   - 2.2. Design Activities
     - 2.2.1. Finalized Hardware Configuration ..... 7
     - 2.2.2. VHDL Interfaces ..... 7
     - 2.2.3. Circuit PSPICE Analysis ..... 8
     - 2.2.4. PC104 ..... 8
     - 2.2.5. Wave Form Generator (WFG) ..... 9
     - 2.2.6. Printed Circuit Board (PCB) Design. ..... 9
     - 2.2.7. Package Design ..... 10
   - 2.3. Prototyping ..... 10
3. Evaluation
   - 3.1. Goals Reached ..... 10
   - 3.2. Teamwork Management ..... 10
   - 3.3. Goals yet to be met ..... 10
4. Final Budget ..... 11
5. Final Schedule ..... 12
6. A EC2000 prospective ..... 13
7. References ..... 14
8. Appendix A: Platform Comparison ..... 15
9. Appendix B: Testing Results ..... 16
10. Appendix C: Research ..... 19
11. Appendix D: Spice Plots ..... 22
12. Appendix E: Enclosure ..... 24
13. Appendix F: Schematic and BOM ..... 26
14. Appendix G: VHDL Code ..... 27
15. Appendix H: C Code ..... 34

LIST OF FIGURES


| *Number* | | *Page* |
|---|---|---|
| 1. | Overall Block Diagram | 7 |
| 2. | PC104 Simulink Model | 8 |
| 3. | PCAD Layout of the Interconnect Board | 9 |
| 4. | Fall Semester Schedule | 12 |
| 5. | Spring Semester Schedule | 12 |
| 6. | Output Denominator Coefficient as simulated and measured | 16 |
| 7. | The Output Nominator Coefficient as simulated and measured | 16 |
| 8. | FPGA Simulink Synthesizable System I.D. Model | 17 |
| 9. | Step input to system and Output form the system simulated | 18 |
| 10. | System Id Numerator Result | 18 |
| 11. | System Id Denominator Result | 18 |
| 12. | Pictorial representation of FPGA starter board | 19 |
| 13. | MATCAD page showing a least square method | 20 |
| 14. | The Wave Form Generator producing a wave on the oscilloscope | 21 |
| 15. | The outside of the WFG | 21 |
| 16. | *The inside of the WFG* | *21* |
| 17. | Pspice Plot 1 | 22 |
| 18. | Pspice Plot 2 | 22 |
| 19. | Pspice Plot 3 | 23 |
| 20. | Pspice Plot 4 | 23 |
| 21. | Schematic | 26 |

**ARIZONA STATE UNIVERSITY**

## ABSTRACT

SYSTEM IDENTIFICATION TOOL

This report details the progress made by the low cost system identification tool design team for the period starting 10/13/2005 to 12/05/2005. This report also serves as the final report in compliance with the senior design course at Arizona State University. More specifically, the technical work completed, budget, schedule and applicable EC2000 criterion will be discussed.

The design project was to build a low cost tool used for system identification. The tool would be targeted for use in development laboratories and educational facilities where rapid high-level determination of a system transfer function is useful. This project serves a second purpose as a study in the operation and implementation of a system identification application using FPGA technology. Limited success was achieved in constructing a working prototype.

# 1 Project Definition

## 1.1 Introduction and Project Scope

Our team's goal is to design and build a low cost system identification tool. System identification is made possible by analyzing the input and output signals of the system under test. This signal information is then processed to provide a transfer function that describes the system. The tool would be targeted for use in development laboratories and educational facilities where rapid high-level determination of a system transfer function is useful. This project serves a second purpose as a study in the operation and implementation of a system identification application using FPGA technology. Future students will be able to study this application and learn more about system identification.

The efforts of the design team were combined with work of the faculty advisor's graduate students in the field of system identification, to develop the system identification tool. Therefore, the scope of this project will focus more on the implementation aspect of this application including the input/output hardware interfacing, not necessarily the development of system identification algorithms.

The technology we will use to implement this tool will be based on the Xilinx Spartan III FPGA starter development board plus our custom designed interface circuitry. Reasons for choosing the Spartan III board include: cost, availability, capability (200K gates, 16MB SRAM), and its scalability.

## 1.2 Requirements

The minimum design requirements for the system identification tool are listed below. It was the determination of this team that these are the minimum requirements met by this tool if it is to be successful in meeting our goals.

- The time spent processing the input/output data must be less than 30 seconds. This is a requirement based on practicality. If this tool is to be useful, it can not spend an exorbitant amount of time running algorithm iterations.
- Data output will be streamed serially via the RS-232 communications protocol to a workstation. This is so that the calculation results may be quickly verified and any system under test may be easily analyzed using the tools available on the PC workstation.

## 1.3 Desired Attributes

Listed below are the desired attributes of the system identification tool.

Note: These attributes will heavily influence the hardware implementation.

- The total unit cost shall be less than $200. This tool should not be prohibitively expensive.
- The input/output signal magnitude shall be within +/- 2.5 VDC and its frequency range shall be between 1Hz and 1000Hz. The input magnitude and frequency range are based upon implementation cost and knowledge of the intended use of the tool.
- The data can be displayed as text on both the PC workstation's terminal and on an LCD screen. Not only can data be streamed through the serial port to a PC, but it can also be displayed in real time on an LCD screen located on the front of the tool. The tool should be easy to use from the user's perspective.
- The system identification algorithm shall be capable of calculating first order systems. This requirement is based both on the tool's processing capabilities and input signal quality.

  Note: Because this tool is intended to be cost effective, its processing capabilities will most likely limit the complexity of the system identification algorithms.

## 2 Technical Work Completed

The team has been very busy researching, designing circuitry, drawing schematics, creating a printed circuit board design, and writing VHDL/C code for the System ID. Tool here forward referred to as simply the “tool”.

### 2.1 Research

Our two main areas of research are in the areas of system identification (a subcategory of Controls engineering) and hardware research for finding the best implementation solution.

In the area of system identification, we have learned a great deal about how system identification algorithms work. Our advisor for the project, Dr. Rodriguez, is very knowledgeable in the area of controls and has written a book called “Analysis and Design of Feedback Control Systems”. In his book, he explains a few system id algorithms and concepts. We have had many meetings with Dr. Rodriguez and have learned a lot about System Identification. A preliminary simulation of the least square method was done using MATHCAD. (appendix)

Our hardware research led us to a comparative study on various hardware technology solutions for our tool. Hardware candidates include the Spartan III FPGA starter kit from Xilinx Inc., A PC-104, the dsPIC from Microchip Technology, and a PSoC development board from Cypress. Their capabilities have been studied and compared in reference to our requirements. This is further shown in table 2 in appendix A. The Spartan III FPGA platform from Xilinx Inc. was determined to be the best solution for our application.

## 2.2 Design Activities

The Design Activities are described below and are separated into the Finalized Hardware Configuration, VHDL Interfaces, Wave Form Generator (WFG), PCB Design, and Package Design.

### 2.2.1 Finalized Hardware Configuration

Figure 1 shown below describes the final configuration of the tool. Starting on the left, the input analog signals are conditioned using filters and then converted into a digital signal. At its heart is the FPGA which stands for Field Programmable Gate Array. VHDL, which is a hardware descriptive language, is used to program the FPGA to implement custom logic. There are three primary VHDL modules implemented in the FPGA including the serial A/D interface, the PSoC interface, and the system identification algorithm. The PSoC, which stands for Programmable System on a Chip, is used to produce the stimulant wave form and communicate the data results to both the LCD screen and the PC via the RS-232 interface.

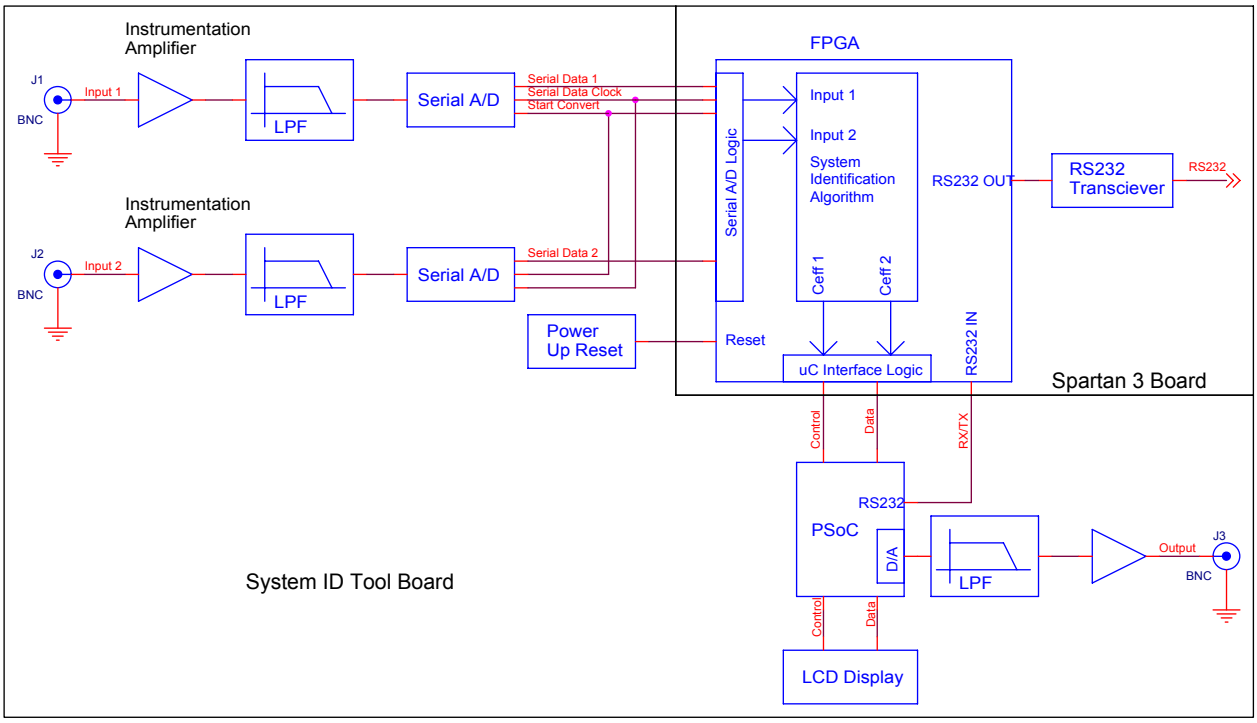



*Figure 1: Overall Block Diagram*

### 2.2.2 VHDL Interfaces

Almost all of the VHDL coding which includes the serial A/D interface, PSOC (Programmable System on a Chip) interface, and RS232 interface was finished in October. The A/D interface is designed to convert the digital signal coming from the A/D into a series of samples that the system id algorithm could understand. The PSoC interface facilitates the communication with the PSoC which controls the starting and stopping of each interaction and the presentation of the results. The PSoC communicates to the PC using the RS-232 level translation hardware on the Spartan-III FPGA development board by simply routing it's two signal lines through the proper pins on the FPGA.

The last VHDL module completed interfaces between the system identification algorithm and the other VHDL modules. This proved to be extremely challenging and was met with a steep learning curve. After some trial and error the team was able to get the code to compile successfully. VHDL code created for this project can be found in the appendix of this document.

### 2.2.3 Circuit PSPICE Analyses

Preliminary circuit PSPICE analyses were performed on both the instrumentation amplifier circuit and the anti-aliasing filter circuit. The plots for these analyses can be found in the appendix. The instrumentation amplifier circuit adds the necessary gain and offset needed to maximize the useful resolution of the input analog signal. The anti-aliasing circuit is a $5^{th}$ order Bessel low pass filter that attenuates any high frequency noise that can distort the signal when digitized.

### 2.2.4 PC104

One device that was investigated for possible use as the hardware platform for the system ID tool was the PC-104. Although it was not chosen as the preferred solution, it did hold promise for other applications. The development team made the decision to use the Advantech PCM 3350 PC-104 processor along with a Diamond MM input/output board. The processor card is based on a 300Mhz Pentium chip while the input/output allows for 2 analog outputs and 16 digital I/O lines (8 in, 8 out) at 12-bit resolution. The model for the transfer function was created in Simulink and was loaded into the processor using the Xpc toolbox feature of Simulink. The developed model is contained in the figure below. The PC104 is now used as the system under test or the target system we are trying to characterize. This gives us a reconfigurable test setup for our tool.

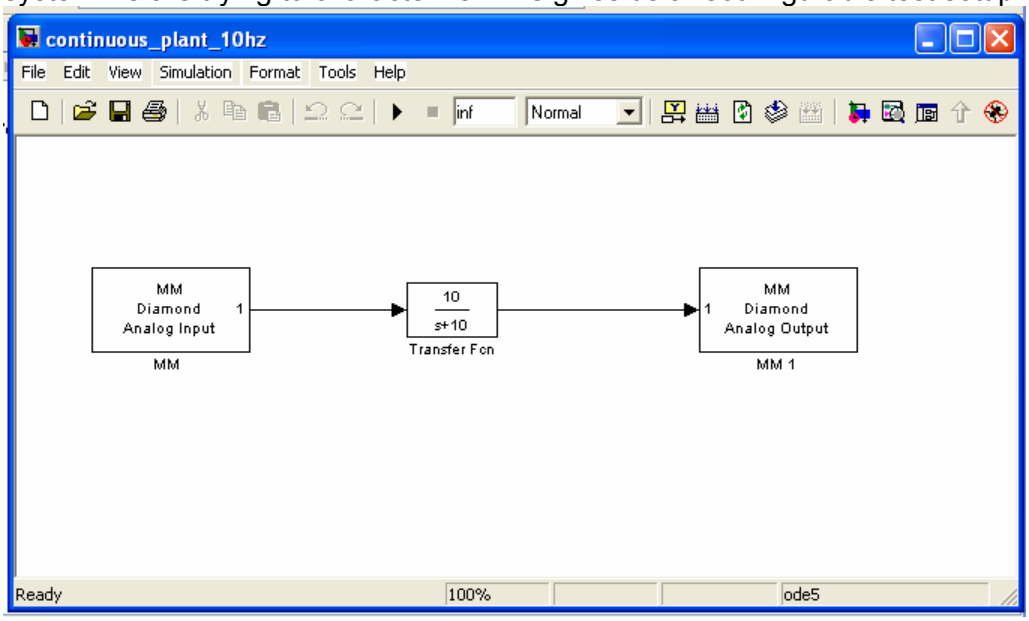



*Figure 2: PC104 Simulink Model*

### 2.2.5 Wave Form Generator (WFG)

A standalone waveform generator was built for the purpose of generating a stimulus to the system under test during the prototyping phase of the design. The WFG was made standalone for testing and verification purposes. A perforated board was used to construct the WFG circuitry and is shown in the appendix. The frequency range of the WFG is from 100 Hz to 100 KHz and is externally adjustable. The amplitude is +/- 3.3 volts and is adjustable from inside the unit. Symmetry and harmonic distortion are also adjustable from inside the unit. More figures showing the WFG can be found in the appendix.

2.2.6 Printed Circuit Board (PCB) Design

Once most of the design had been finished, we set out to build a custom Printed Circuit Board (PCB) that would bring together every aspect of our project. Two PCB's were made. The first one interconnects the Spartan-3 FPGA board to the Analog Interface (AI) board. The advantage of this interconnect board was that it provided a third connector for a second Spartan-3 FPGA board allowing us to expand the available re-configurable logic. This was done to add future scalability allowing for more complex system identification algorithms. The second PCB, the AI board, includes everything needed including the PSoC, analog input/output channels, and the LCD connector. This board also had a connector which allowed it to connect directly to the FPGA board without use of the interconnect board. Indeed, this is how the prototype is usually demonstrated. The schematic and bill of materials for both PCB boards are included in the Appendix.

Parts procurement began shortly after the PCB was sent out for fabrication. Almost all our components were found and ordered from Digikey.com. Small annoyances like part shortages were an issue but our team was able to overcome these by selecting suitable alternate parts. Learning to use PCAD was also a challenge. PCAD is the CAD software used to layout and design the two PCBs. Each board includes two signal layers where the traces are routed and a silkscreen layer where each part and section of the board is labeled in white. A picture of a finished and unpopulated AI board is pictured in the Appendix. Pictured below in figure 3, is he cad layout of the interconnect board.

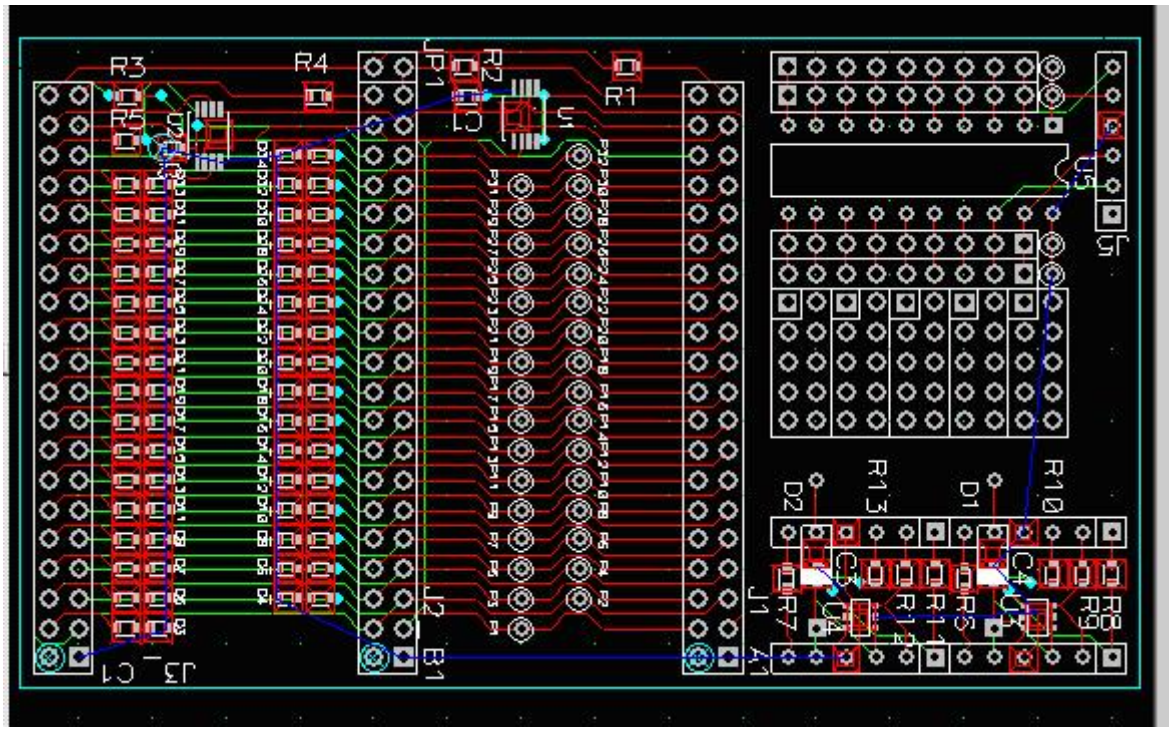



*Figure 3: PCAD Layout of the Interconnect Board*

Finally, after more than a week, the bare board and parts arrived and the board was assembled. The assembled board was then integrated into the system and tested. Troubleshooting began with testing the RS232 interface to verify data could be sent and received via a laptop computer. Next, the LCD interface was verified by displaying some test text information. Then the serial A/D channels were tested. Injecting a known input and adjusting the built-in potentiometers the desired signal scaling was achieved. Next, the step output signal was tested. This output also was scaled using an additional built-in potentiometer. With these circuits operational, testing of the system identification algorithm becomes possible.

2.2.7 Package Design

Initial package/enclosure designs are detailed in the Appendix. The final demo did not include an enclosure for the tool due to timing constraints.

2.3 Prototyping

Our intent is to simulate all of our system identification algorithms for validation and comparison purposes. Our primary tool for measuring each algorithm's performance is MATLAB. One of our initial system identification algorithm simulations using the least square method was done using MATHCAD. It is shown in Fig. 6 in the Appendix

A PC104 board was used to implement a continuous time model of a first order system and was stimulated with a step function. The output from the system under test was then fed into the system I.D. tool. On occasion the system I.D. tool produced meaningful results. For example, in the case where the system under test was $H(s) = \frac{10}{s+10}$ the discrete time model becomes $H(z) = \frac{0.09516}{z-0.9048}$ (sample time = 10ms) and due to the system's inherent quantization error the transfer function becomes $H(z) = \frac{0.1016}{z-0.8906}$. See the Xilinx Simulink model and test results in the appendix. For this test condition the tool would occasionally produce the correct $H(z) = \frac{0.1}{z-0.9}$. The problem to be solved is to understand why this result cannot be obtained consistently. More generally, It appears to be a convergence issue with our tool and unfortunately our team has run out of time to determine the cause.

## 3 Evaluation

3.1 Goals Reached

The team was successful with designing, building and testing of a printed circuit board of our own design. It fulfills all of our original requirements. With respect to these activities we were able to deliver on time. It is unfortunate, due to time constraints, that we were unable to better tune and enhance our tool to produce more consistent and reliable results.

3.2 Teamwork Management

As a team we learned some very valuable skills. Some of us became very adept at being able to capture a schematic, design and layout a printed circuit board, and procure the necessary components in time. Some of us also learned a great deal about filter design, application design, VHDL coding, C coding, and the use of System Generator in Matlab for FPGA algorithm development. Some unexpected things were learned as well. For example, finding a solution to a problem when no apparent solution exists was a lesson in endurance. As a team we were faced with many challenges with less than obvious solutions. We would meet to discuss the issues and then break the problem into smaller more manageable parts. Then draw upon each member's experience and determine a course of action.

3.3 Goals yet to be met

Even with the project complete, there are some goals we have yet to accomplish. First of which is to determine the root cause of the convergence issue. We would like to improve the consistency and reliability of our tools results. We are also more interested now more than ever before in the operation and theory behind system identification. We'd like to do more research into the theory behind the algorithms used for system identification.

## 4 Final Budget

For the duration of the spring 2005 semester, our design team did not have any expenses towards the system identification tool. This duration of time was mainly spent researching system id algorithms, defining hardware specification, and identifying development tools. The team had accumulated a total of two-hundred dollars not including the one-hundred dollars given to us by the EE department, which went towards the development phase of the project. By the end of summer 2005, hardware specification was well defined and chosen. The Xilinx's FPGA was chosen which financially, was beneficial in that the hardware and development tools were free and available through our faculty advisor Dr. Rodriguez. In addition, the PSoC development kit and display were given to the team as free samples. The teams expenditures were only the interconnections, discrete components (A/D, D/A etc..), the presentation poster, and the PCB. Thus, the team only spent $240 of the accumulated three-hundred dollars. Table 1 shows an itemized list of each hardware component and its cost for the fall 2005 semester.

| Item | Fall 2005 Expenditure | Description |
|---|---|---|
| Xilinx Development Software | FREE | Software used to develop VHDL code |
| PSoC Development Kit | FREE | Hardware and Software used for PSoC |
| Spartan 3 Board | ASU Provided | FPGA board used to run system id algorithm |
| Interconnections | $10 | Wires, cables, etc… |
| Discrete Components | $100 | A/D, Op-amps, resisters, sockets, etc… |
| Display | FREE | LCD screen |
| PCB | $90 | Printed Circuit Board used for application circuit |
| Poster | $40 | Poster used for senior design presentation |
| Total | $240 | |
| Ending Balance | $300 - $240 = | $60 |

*TABLE 4: Budget*

Each team member dedicated a man hour of 4-8 hours per week throughout the spring 2005, summer 2005, and fall 2005. For the spring and fall a total of 52-104 hours were invested. For the summer a total of 60-120 hours were invested. The average hours invested for the year is roughly 300 hours invested.

# 5 Final Schedule

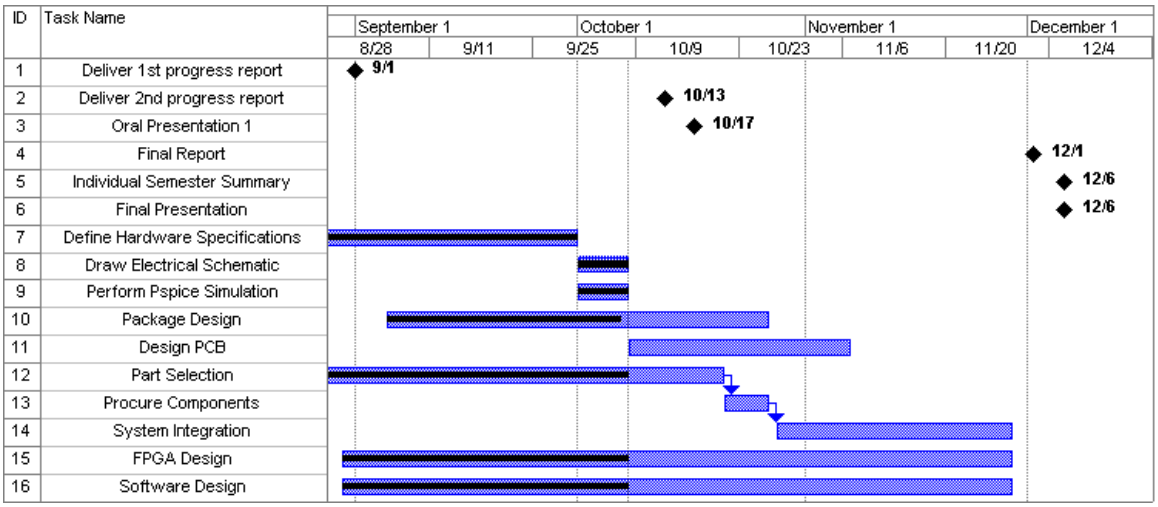



*Figure 4: Fall Semester Schedule*

The over-all semester final schedule is as shown above in figure 4. The schedule as of the end of last semester is shown below in figure 5. We did a good job adhering to the schedule as there were no major deviations.

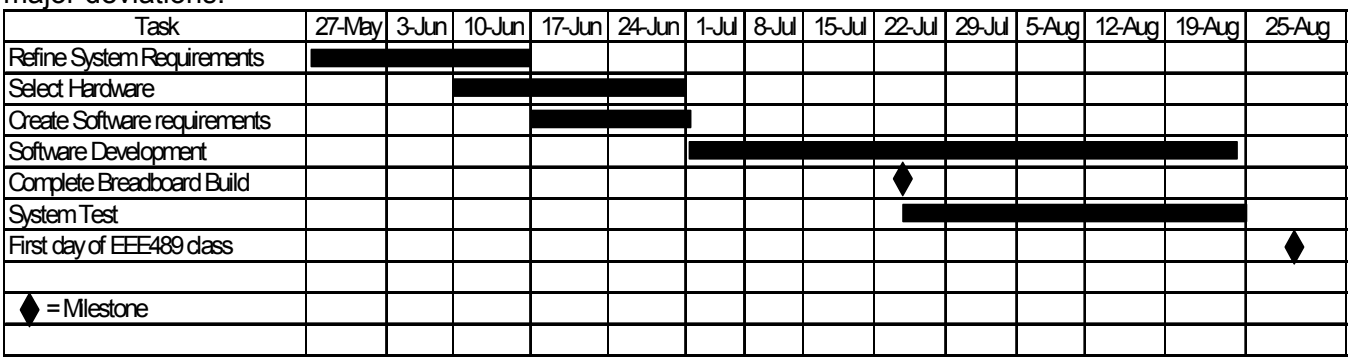



*Figure 5: Spring Semester Schedule*

## 6 An EC2000 Perspective

Our team has identified economic, ethical, manufacturability, environmental, and sustainability as the EC2000 Criterion 4 factors. We have considered economic to be the most important criterion in that we ultimately would like to provide a low cost portable system identification tool to the public. The system identification tool will help users better understand the design of their system, which will allow them to improve their design if needed. Thus, the tool will help users (engineers, technicians, etc..) improve technology by characterizing their system.

The second criterion considered is ethics. Our team has maintained a high ethical standard by respecting the intellectual property developed by others using do diligence in completing an extensive prior art search. At present time, POPLAB and Frequency Domain System Identification Toolbox are the only two system identification programs. These systems, however, are computer based programs. Our tool introduces the same identification, but with the convenience of portability at a lower cost.

For ease of mass production, the manufacturability of the system id tool is based on a modular design by designating the device into components (PCB, display, casing, etc…). The modular design will also allow an easily configurable solution to fit costumer needs. In addition, the design will benefit future developers who choose to improve the system id tool.

The team would like to account for a durable device to ensure consumers the sustainability of the tool. The enclosure of the tool is based on an ergonomic design made from durable plastic. The type of plastic chosen withstands the relative degree of mechanical strength such as tensile strength, flexural strength, and impact strength needed for our tools expected environment. Furthermore, other factors such as heat deflection, the relative coefficient of thermal expansion, and volume resistivity were considered in the type of plastic. To maintain sustainable standards, the tool will also be based on open and freely available resources and scalable and open ended architecture.

The last criterion considered is the environmental criterion. Our electronic parts is RoHS (Restriction of Hazardous Substances) compliant, which avoids materials such as Cadmium (Cd), mercury (Hg), hexavalent chromium (Cr (VI)), polybrominated biphenyls (PBBs), polybrominated diphenyl ethers (PBDEs), and lead (Pb). Furthermore, the tool is also designed by which rechargeable batteries can be incorporated in future development. However, currently our tool is based on low power usage, which is environmentally safe.

# 7 References


[1] Spartan-3 Starter Kit Board User Guide
UG130 (v1.0.3) Xilinx October 15, 2004

[2] Final Data Sheet CY8C29466, CY8C29566, CY8C29666, and CY8C29866
© Cypress MicroSystems, Inc. 2003-2004
Document No. 38-12013 Rev. *G November 12, 2004

[3] AD7303 Data Sheet
© Analog Devices, Inc., 1997

[4] AD7823 Data Sheet
© Analog Devices, Inc., 1997

# 8 Appendix A: Platform Comparison

| Option | Pro's | Con's |
|---|---|---|
| Microchip uC | A/D is built-in | Development software is expensive >$700 |
| | Many software examples are available | Does not utilize Dr. AAR's grad student's help |
| | 1 Development Unit is available | |
| | 144KB Flash, 8KB RAM, 4KB EEPROM, 1cycle MAC (30MIPS), 10bit 100ksps A/D, Matrix transpose in 232cycles in 7.2us | |
| Cypress PSoC | Development software is free | Does not utilize Dr. AAR's grad student's help |
| | Hardware is inexpensive and available | Analysis may be slow |
| | DFT algorithm is complete | Limited 2 KB RAM 32KB ROM |
| | Analog/Digital Blocks are configurable | |
| | Little learning curve (Jim's expertise) | |
| Xilinx's FPGA | Hardware is available at ASU | Steep learning curve |
| | Some system ID work already done by Siva (Dr. AAR's grad student) | Requires designing AAF and A/D interface |
| Xport GBA FPGA | Uses Xilinx's FPGA (utilize Siva's help) | Requires designing AAF and A/D interface |
| | Wow factor, the graphical display is really cool | Moderate Learning Curve |
| | 1 Development Unit is available | 50K gates |
| | 4MB Flash, 16MB SDRAM, external ARM uC | |

*Table 2: The pro's and Con's of each development system initially considered*

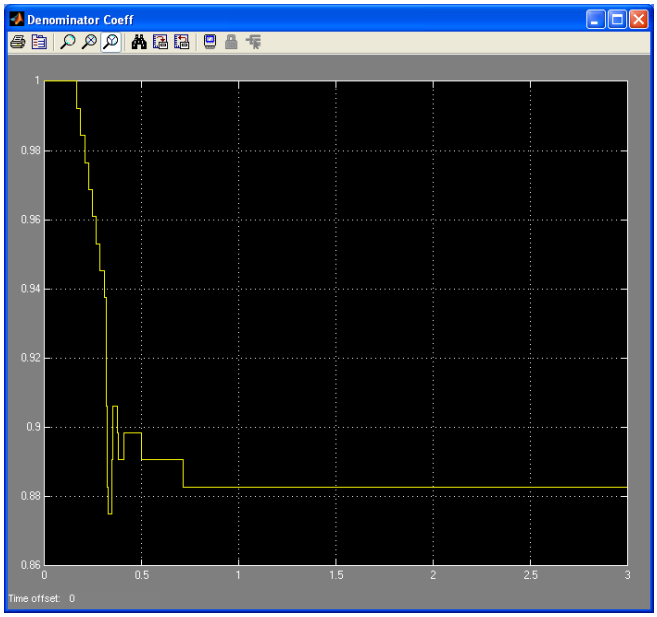
## Appendix B: Testing Results

General Equation: $$H(z) = \frac{a}{z - b}$$

a = Numerator Coefficient

b = Denominator Coefficient

Simulink vs. Measured (Denominator Coefficient)



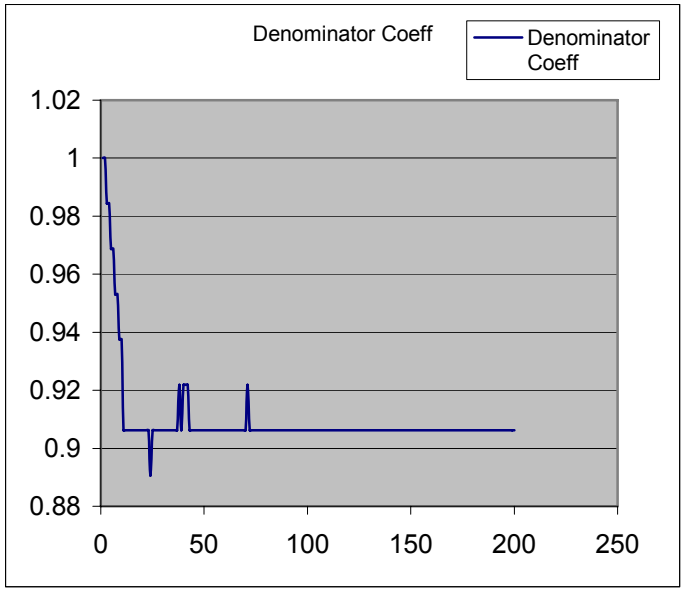
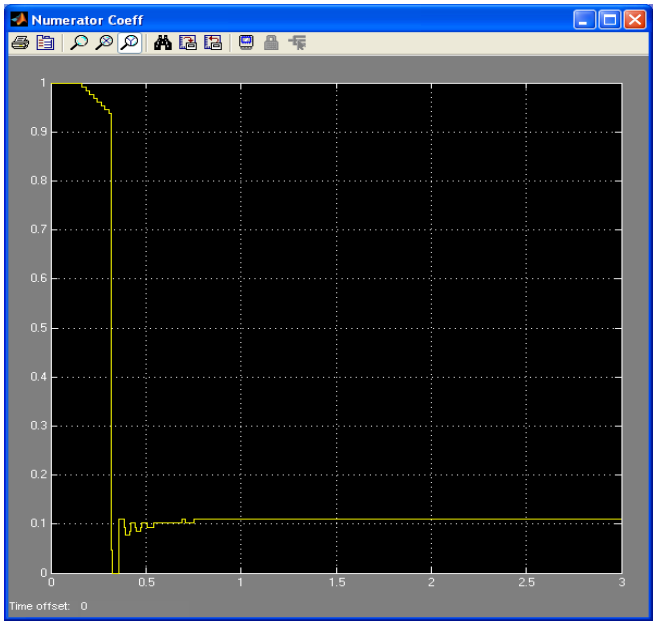




*Figure 6: Output Denominator Coefficient as simulated (Left) and Measured with the tool (right)*

Simulink vs. Measured (Numerator Coefficient)



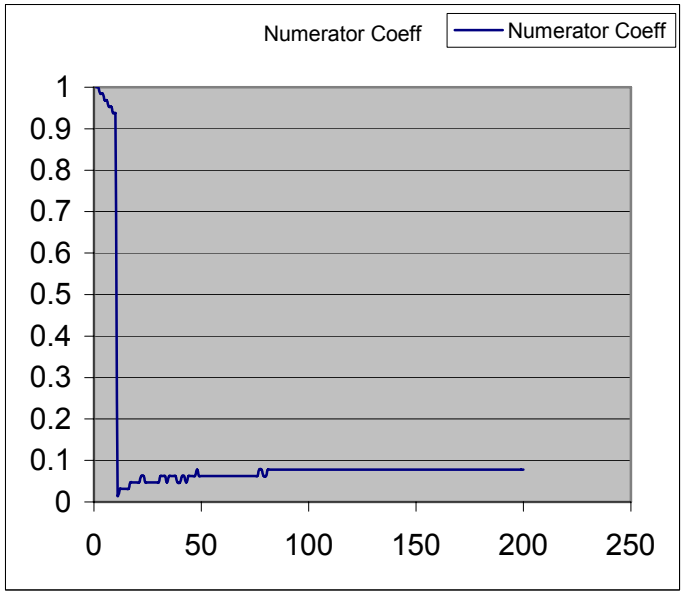




*Figure 7: The Output Nominator Coefficient as simulated (left) and measured with the tool (right)*

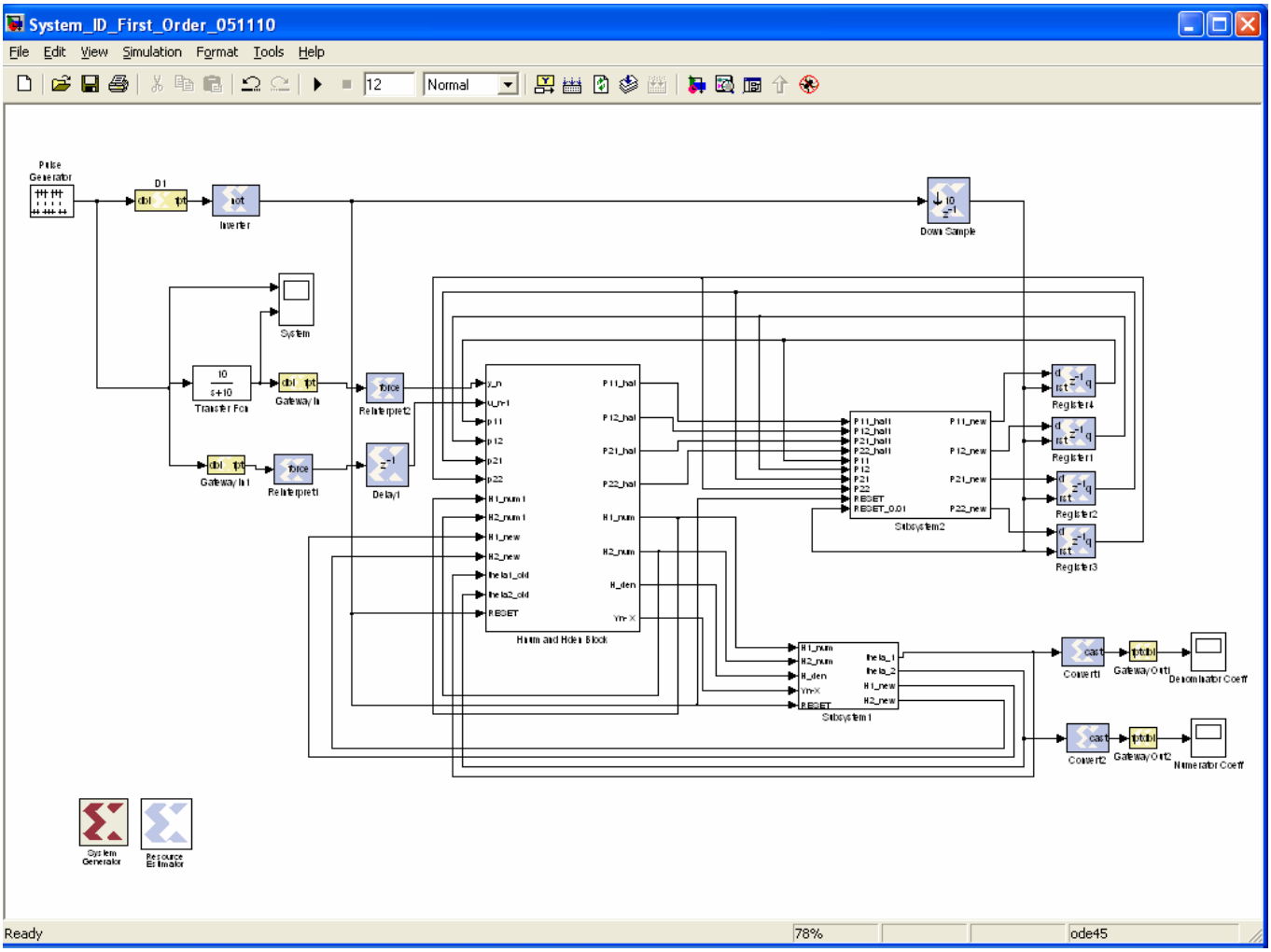



*Figure 8: FPGA Simulink Synthesizable System I.D. Model*

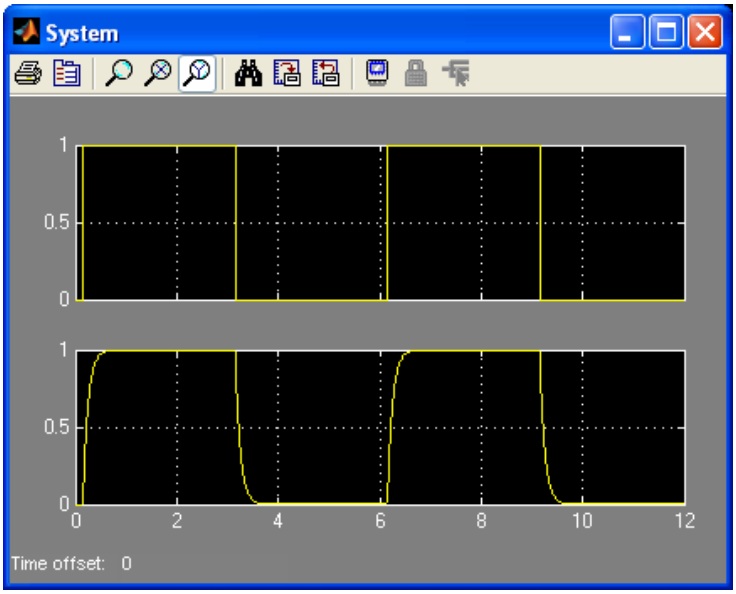



*Figure 9: Upper trace = Step input to system, Lower trace = Output from the system*

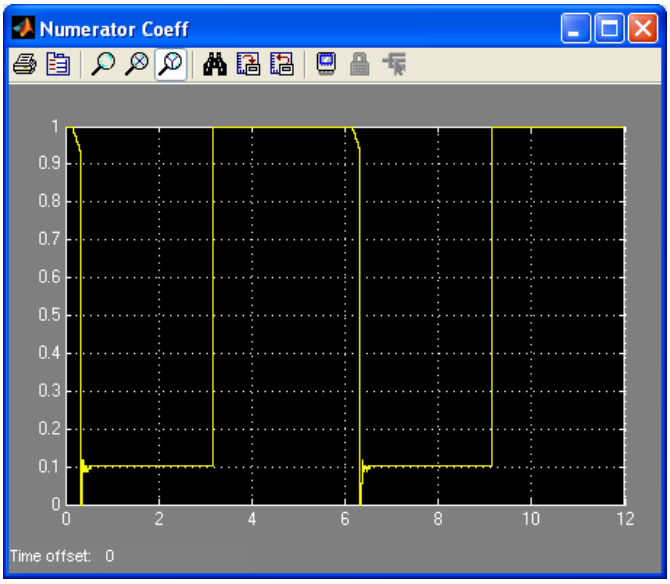



*Figure 10: System Id Numerator Result*

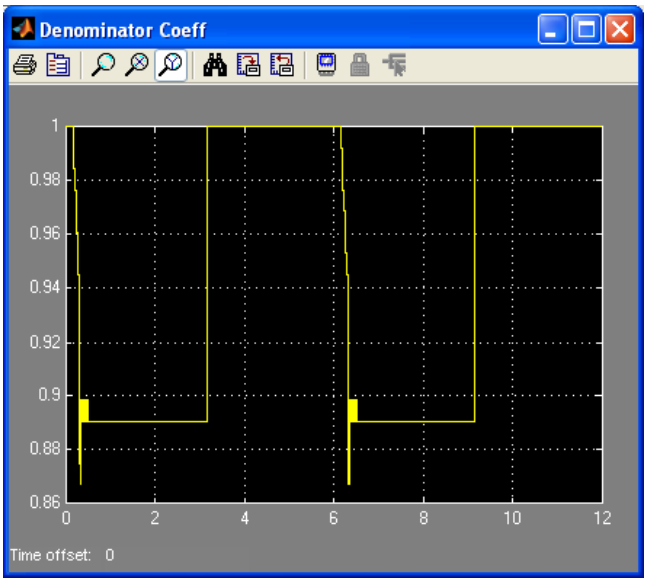



*Figure 11: System Id Denominator Result*

Appendix C: Research

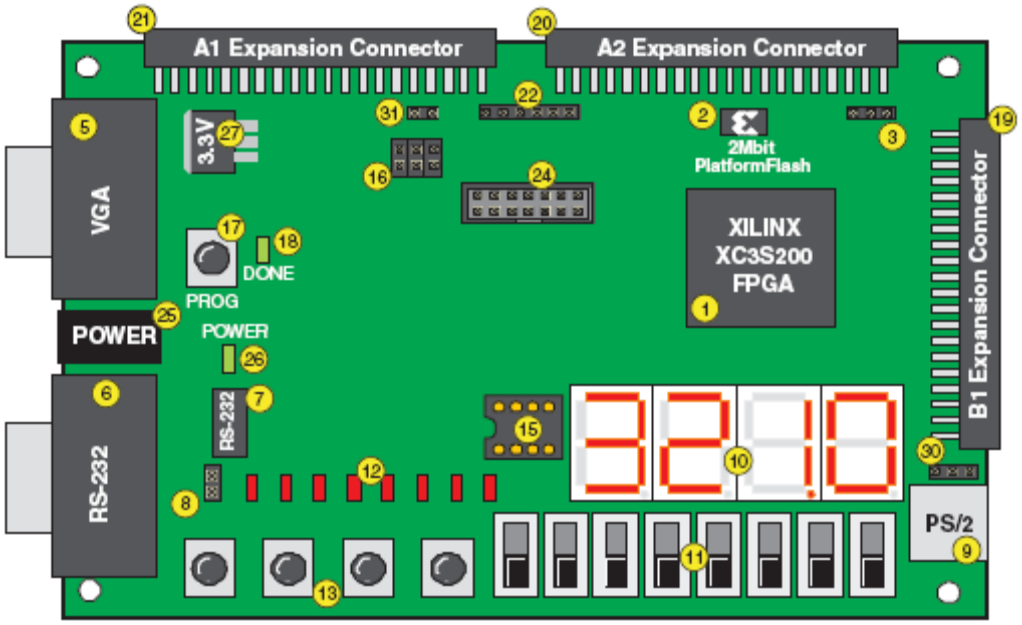



*Figure 12: This is a pictorial representation of what is located on the FPGA starter board*

$f_s := 200 \cdot 10^3$ $T := \frac{1}{f_s}$ $T = 5 \times 10^{-6}$ Sampling Rate

$N_{max} := 2^{16}$ $N_{max} = 65536$ $N := 0 .. N_{max}$ $k := 0 .. N_{max} - 1$ $N_{max} \cdot T = 0.328$ Sampling Window

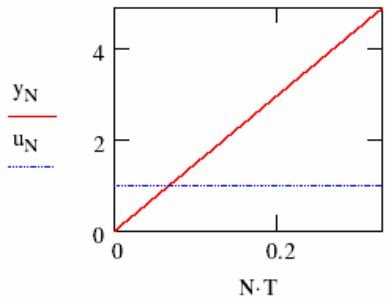
First Case:

$\tau := 1 \cdot 10^2$ $c_1 := \frac{1}{\tau}$ $c_1 = 0.01$ $c_2 := 15$ T.F. Coefficients.

$u_N := \Phi(N \cdot T)$

$y_N := \frac{c_2}{c_1} \cdot \left(1 - e^{-c_1 \cdot N \cdot T}\right)$ $A := \text{augment}(-y, u)$

$b_{k+1} := \frac{y_{k+1} - y_k}{T}$ $b_0 := 0$ $x := \left(A^T \cdot A\right)^{-1} \cdot A^T \cdot b$ $x_0 = 9.72 \times 10^{-3}$ $x_1 = 14.999$ Calculated T.F. Coeff.




Graph of input and output functions.

Second Case:

$\tau := 1 \cdot 10^{-2}$ $c_1 := \frac{1}{\tau}$ $c_1 = 100$ $c_2 := 0.01$ T.F. Coefficients.

$u_N := \Phi(N \cdot T)$

$y_N := \frac{c_2}{c_1} \cdot \left(1 - e^{-c_1 \cdot N \cdot T}\right)$ $A := \text{augment}(-y, u)$

$b_{k+1} := \frac{y_{k+1} - y_k}{T}$ $b_0 := 0$ $x := \left(A^T \cdot A\right)^{-1} \cdot A^T \cdot b$ $x_0 = 99.922$ $x_1 = 9.992 \times 10^{-3}$ Calculated T.F. Coeff.

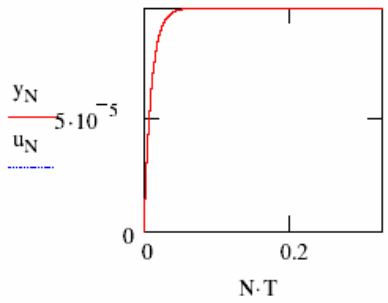



*Figure 13: This MATCAD page shows the validation of the least square method*

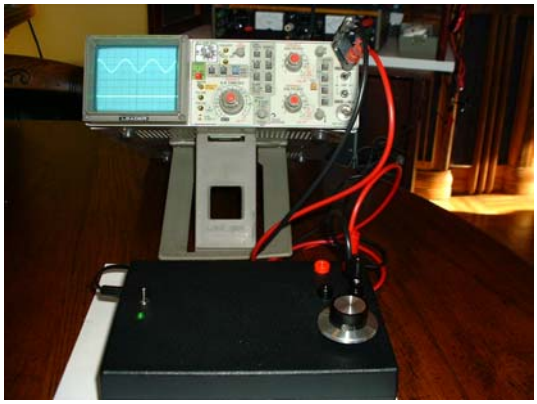



*Figure 14. (left) The Wave Form Generator producing a wave on the oscilloscope.*
*Figure 15. (right) The outside of the WFG*

Figure 16. The inside of the WFG

Appendix D: Spice Plots

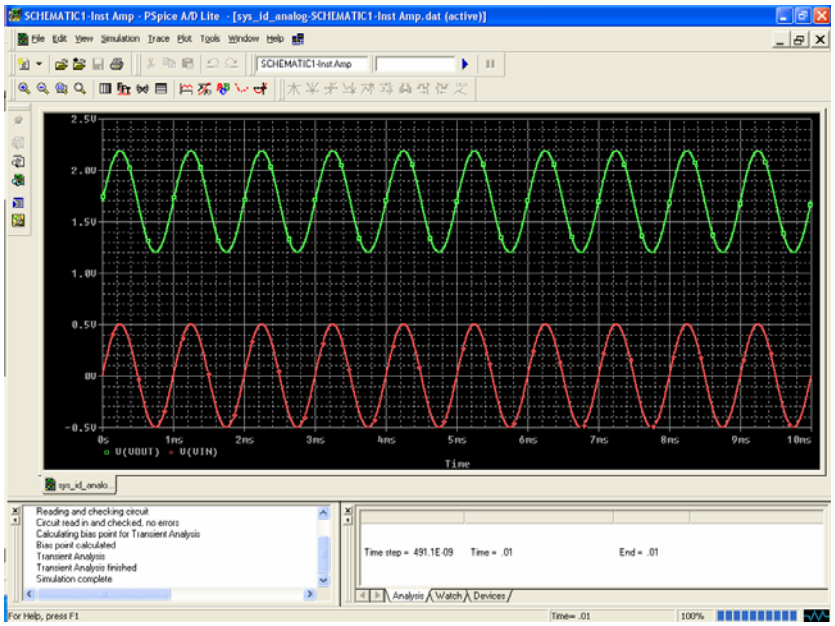



*Figure 17: Pspice Plot 1*

This shows the output of the amplifier stage to a sinusoidal input. Notice the gain is 1 and the output is offset by +1.65VDC. The DC offset is required to keep the input to the A/D converter biased at half of the 3.3VDC supply voltage.

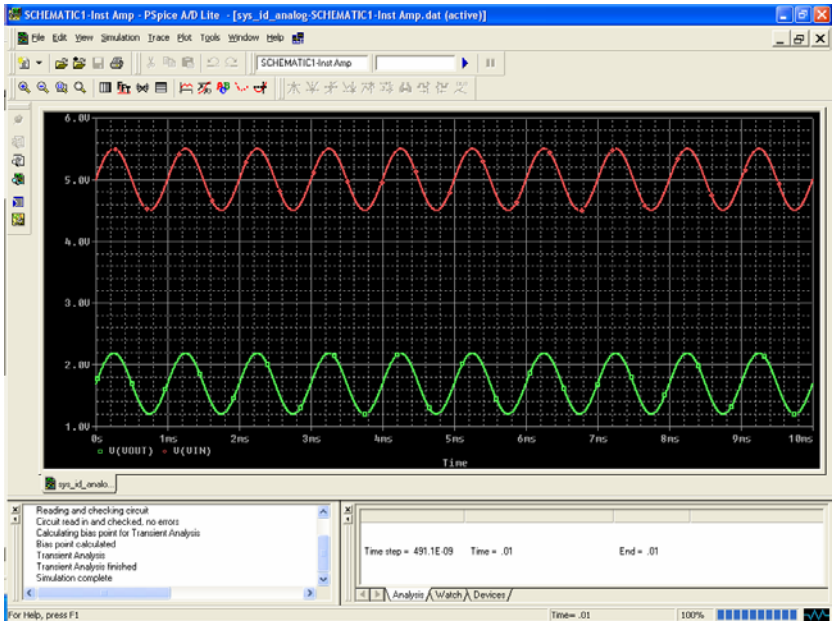



*Figure 18: Pspice Plot 2*

*This shows how the amplifier will perform when an input voltage with DC bias is applied. Notice the output (in green) is not affected by the 5VDC offset applied to the input.*

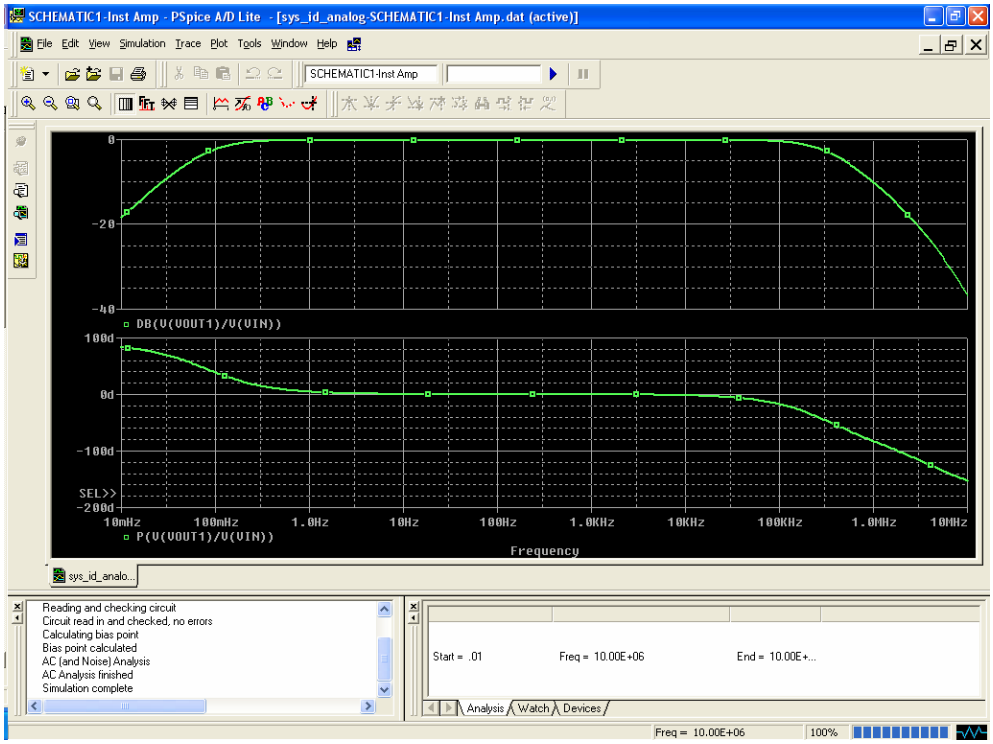



*Figure 19: Pspice Plot 3*

This shows the magnitude and phase response of the instrumentation amplifier stage. Notice the amplifier possesses unity gain and nearly zero phase shift from 1Hz to 10kHz.

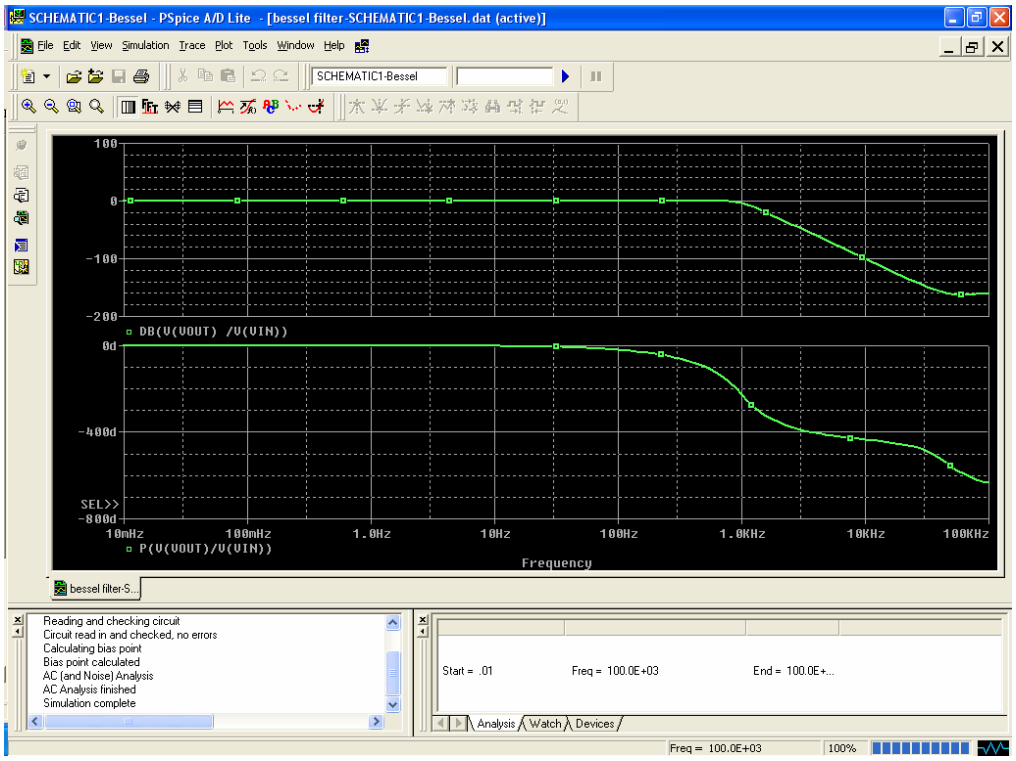
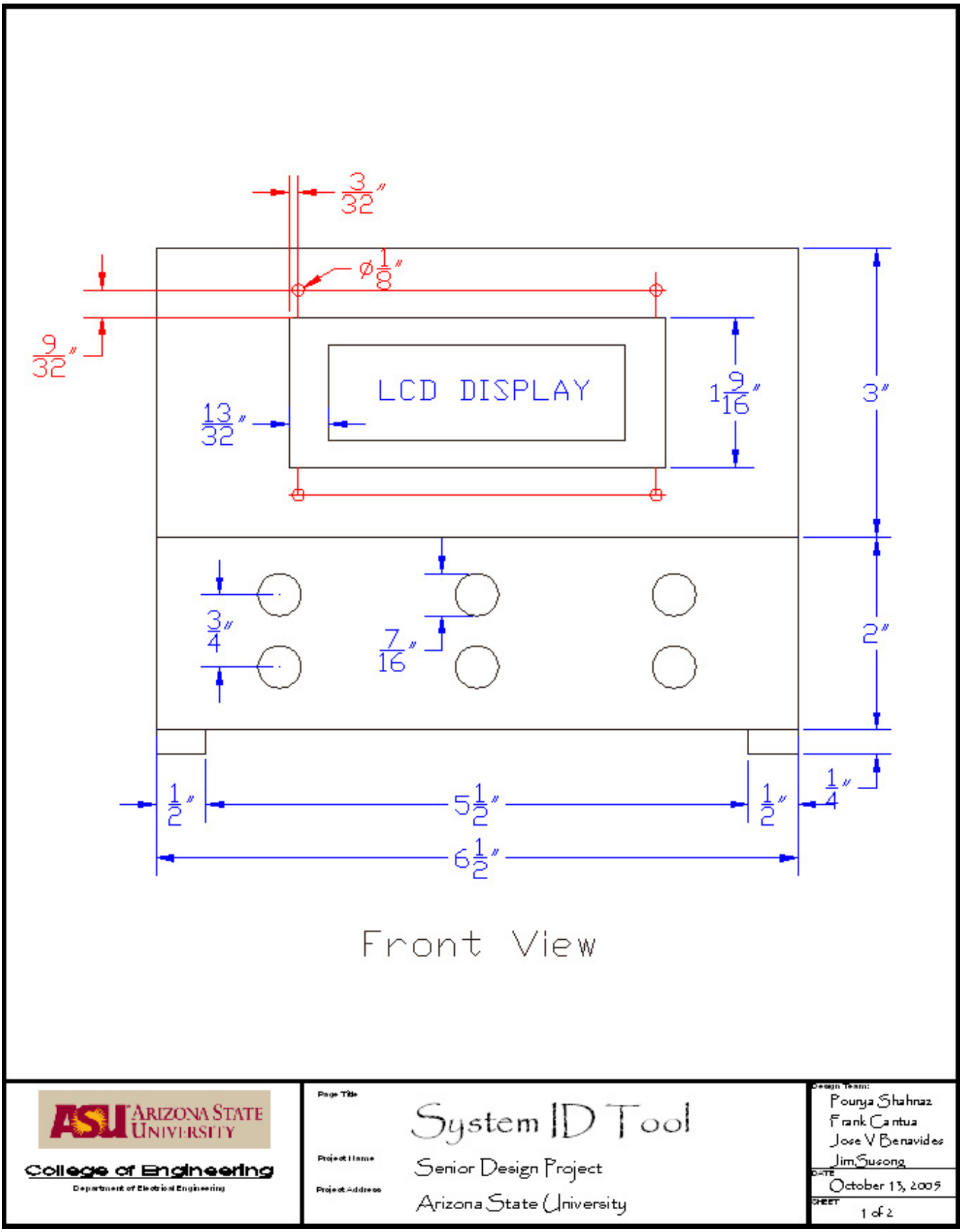
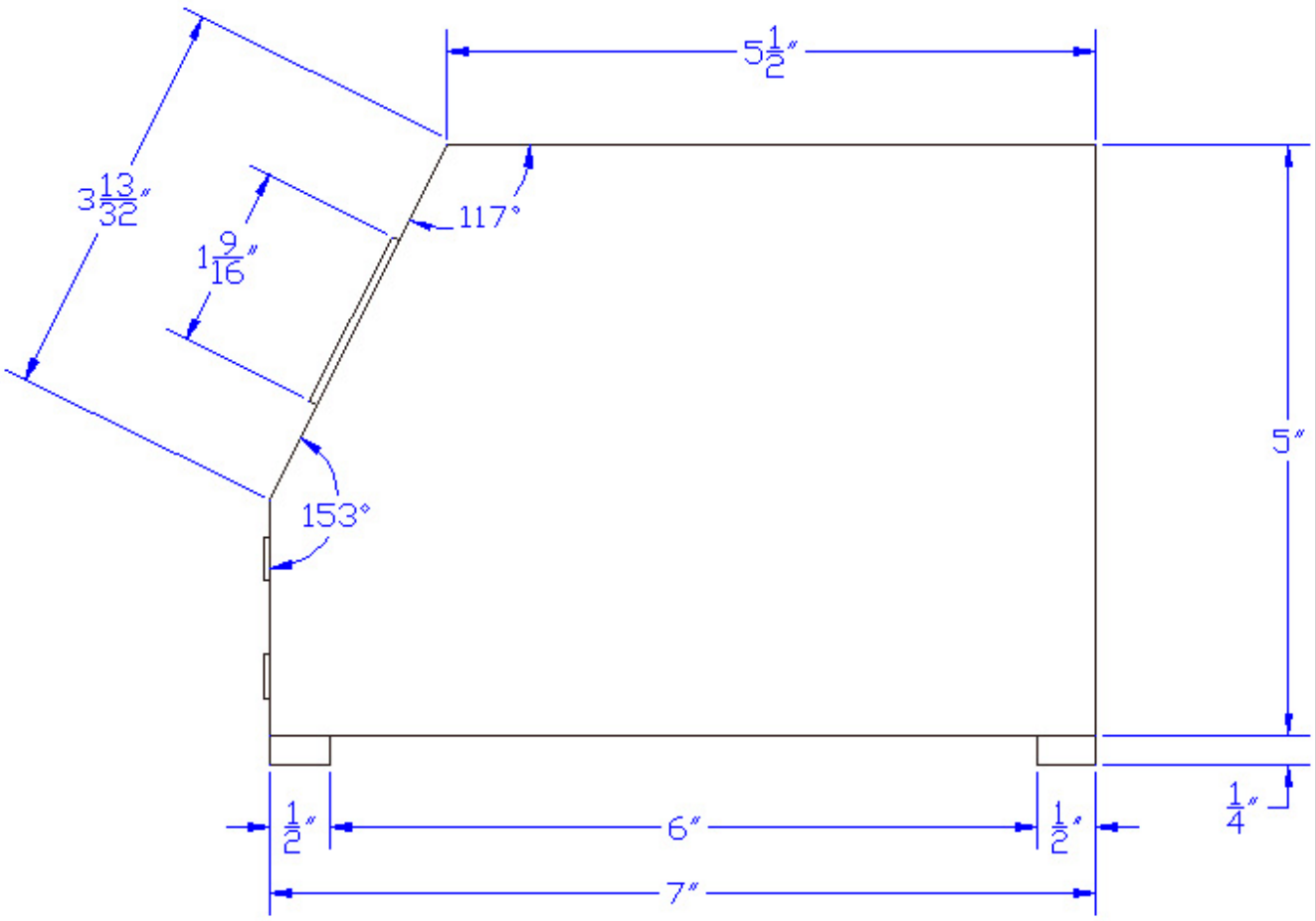



*Figure 20: Pspice Plot 4*

This shows the magnitude and phase response of the anti-aliasing filter stage. Notice the amplifier possesses unity gain from DC to ≈1kHz. This is desirable because the sampling frequency of the tool is 10KHz.

Appendix E: Enclosure






Side View

ASU Arizona State University
College of Engineering
Department of Electrical Engineering

Page Title: System ID Tool
Project Name: Senior Design Project
Project Address: Arizona State University

Design Team:
Pourya Shahnaz
Frank Cantua
Jose V Benavides
Jim Susong

DATE: October 13, 2009
SHEET: 2 of 2

## Appendix F: Schematic


*Figure 21: Schematic*

## Appendix G: VHDL Code

```
-- ********************************************************************************
--
-- Owner:   System I.D. Team
-- Date:        9/8/05
-- File:    RS232.vhd
--
-- Purpose: Interface PSoC to RS232 driver on Spartan 3 Board
--
-- ********************************************************************************

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
USE ieee.STD_LOGIC_unsigned.all;

entity RS232_Driver is

    port(
          TX_in  : in STD_LOGIC;  -- RS232 Transmit in
          RX_in  : in STD_LOGIC;  -- RS232 Receive in
          TX_out : out STD_LOGIC; -- RS232 Transmit out
          RX_out : out STD_LOGIC  -- RS232 Receive out
        );

end RS232_Driver;

architecture RTL of RS232_Driver is
begin
RS232Proc:process(TX_in, RX_in) begin
   TX_out <= TX_in;
   RX_out <= RX_in;
end process RS232Proc;
end RTL;
```

```
-- ************************************************************************
--
-- Owner: System I.D. Team
-- Date:  11/4/05
-- File:  SerialA2D.vhd
--
-- Purpose: Interface to Analog Devices AD7823 8 bit Serial A/D to Xilinx FPGA
--
-- ************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
USE ieee.STD_LOGIC_unsigned.all;

entity Serial_A2D is

    port(
          rst_n      : in STD_LOGIC;  -- Reset active low
          data_in1   : in STD_LOGIC;  -- Serial data from A/D
          data_in2   : in STD_LOGIC;  -- Serial data from A/D
          src_clk_1  : in STD_LOGIC;  -- System 50MHz Clock
          src_ce_1   : in STD_LOGIC;  -- Clock Enable
          a2d_clk    : out STD_LOGIC; -- Serial A/D data clock
          convst_n   : out STD_LOGIC; -- Serial A/D start conversion
          data_out1  : out STD_LOGIC_VECTOR (7 downto 0);  -- A/D Data
          data_out2  : out STD_LOGIC_VECTOR (7 downto 0)   -- A/D Data
        );

end Serial_A2D;

architecture Definition of Serial_A2D is

constant RESET_ACTIVE : std_logic := '0';

type StateType is (idle,toggle_convst,wait_conv,read_data);

signal a2d_clk_sig : STD_LOGIC;
signal bit_cnt_sig : STD_LOGIC_VECTOR (3 downto 0);
signal count_sig : STD_LOGIC_VECTOR (7 downto 0);
signal data_int_sig1 : STD_LOGIC_VECTOR (7 downto 0);
signal data_int_sig2 : STD_LOGIC_VECTOR (7 downto 0);
signal present_state_sig : StateType;

begin
datareg:process(rst_n,src_clk_1) begin
  if (rst_n = RESET_ACTIVE) then
     data_out1 <= x"00";
     data_out2 <= x"00";
  elsif (src_clk_1'event and src_clk_1 = '0') then
     if(src_ce_1 = '1') then
        data_out1 <= data_int_sig1;
        data_out2 <= data_int_sig2;
     end if;
  end if;
end process datareg;

    --Main state machine
    state_comb:process(present_state_sig,rst_n,src_clk_1) begin
    if (rst_n = RESET_ACTIVE) then
        a2d_clk_sig <= '0';
        bit_cnt_sig <= (others => '0');
        convst_n <= '0';
        count_sig <= (others => '0');
        data_int_sig1 <= (others => '0');
        data_int_sig2 <= (others => '0');
        present_state_sig <= idle;
    elsif (src_clk_1'event and src_clk_1 = '0') then
        case present_state_sig is
            when idle =>
                convst_n <= '1';        -- Set convst_n bit high
                if (src_ce_1 = '1') then
                    convst_n <= '0';
```

```
                present_state_sig <= toggle_convst;
            end if;
        when toggle_convst =>
            count_sig <= count_sig + 1;
            if (count_sig = 50) then
                count_sig <= (others => '0');
                convst_n <= '1';
                present_state_sig <= wait_conv;
            end if;
        when wait_conv =>
            count_sig <= count_sig + 1;
        if (count_sig = 249) then
            count_sig <= (others => '0');
                present_state_sig <= read_data;
            end if;
        when read_data =>
            count_sig <= count_sig + 1;
            if (count_sig = 50) then
            count_sig <= (others => '0');
                a2d_clk_sig <= not(a2d_clk_sig);
                if(a2d_clk_sig = '1') then
                    data_int_sig1 <= data_int_sig1(6 downto 0) & data_in1; -- concatenation
                    data_int_sig2 <= data_int_sig2(6 downto 0) & data_in2;
                    bit_cnt_sig <= bit_cnt_sig + 1;
                end if;
                if (bit_cnt_sig = 8) then
                  bit_cnt_sig <= (others => '0');
                  present_state_sig <= idle;
                                                        -- Invert MSB to make it 2's complement.
--                  data_out1 <= data_int_sig1 xor x"80";
--                  data_out2 <= data_int_sig2 xor x"80";
                    a2d_clk_sig <= '0';
            end if;
            end if;
        end case;
    end if;
end process state_comb;

a2d_clock:process(rst_n,src_clk_1) begin
   if (rst_n = RESET_ACTIVE) then
      a2d_clk <= '0';
--  elsif (src_clk_1'event and src_clk_1 = '0') then
   else
      a2d_clk <= a2d_clk_sig;
   end if;
end process a2d_clock;

end Definition;
```

```
-- ************************************************************************
--
-- Owner: System I.D. Team
-- Date:  11/05/05
-- File:  SystemID.vhd
--
-- Purpose: System ID Tool Main Code
--
-- ************************************************************************
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
USE ieee.STD_LOGIC_unsigned.all;

entity SystemID is

   port(
         rst_n     : in STD_LOGIC;  -- Reset active low
         data_in1  : in STD_LOGIC;  -- Serial data from A/D
         data_in2  : in STD_LOGIC;  -- Serial data from A/D
         src_clk_1 : in STD_LOGIC;  -- System 50MHz Clock
--         src_ce_1  : in STD_LOGIC;  -- Clock Enable (Not Used)
         EnableIn  : in STD_LOGIC;  -- Chip EnableIn
         ReadIn    : in STD_LOGIC;  -- Read Signal
         TX_in  : in STD_LOGIC;  -- RS232 Transmit in
         RX_in  : in STD_LOGIC;  -- RS232 Receive in
         a2d_clk   : out STD_LOGIC; -- Serial A/D data clock
         convst_n  : out STD_LOGIC; -- Serial A/D start conversion
         DataBus   : inout STD_LOGIC_VECTOR (7 downto 0); -- uC Data bus
         TX_out : out STD_LOGIC; -- RS232 Transmit out
         RX_out : out STD_LOGIC  -- RS232 Receive out
       );

end SystemID;

architecture RTL of SystemID is

constant RESET_ACTIVE : std_logic := '0';

signal data_out_sig1       : STD_LOGIC_VECTOR (7 downto 0);
signal data_out_sig2       : STD_LOGIC_VECTOR (7 downto 0);
signal data_out_sig3       : STD_LOGIC_VECTOR (7 downto 0);
signal data_out_sig4       : STD_LOGIC_VECTOR (7 downto 0);

--signal DataReqIn_sig : STD_LOGIC;
--signal SubSysWriteOut_sig : STD_LOGIC;
--signal start_conv_sig : STD_LOGIC;
--signal conv_comp_sig : STD_LOGIC;

component uC_Interface
  port(
        Coeff1In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 1 register
        Coeff2In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 2 register
        Coeff3In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 3 register
        Coeff4In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 4 register
        EnableIn     : in STD_LOGIC;  -- Chip EnableIn
        rst_n        : in STD_LOGIC;  -- Reset active low
        ReadIn       : in STD_LOGIC;  -- Read Signal
        src_clk_1    : in STD_LOGIC;  -- System 50MHz Clock
--        src_ce_1     : in STD_LOGIC;  -- Clock EnableIn (Not Used)
--        SubSysWriteIn : in STD_LOGIC;  -- Subsystem write
--        StatusIn     : in STD_LOGIC_VECTOR (7 downto 0);   -- Subsyst. status reg.
--        DataReqOut   : out STD_LOGIC;                      -- Data Request
        DataBus      : inout STD_LOGIC_VECTOR (7 downto 0) -- uC Data bus
      );
end component;

component system_id_first_order_051105_nosim_clk_wrapper
  port (
       ce: in std_logic := '1';
       clk: in std_logic;
       data_in1: in std_logic;
```

```
        data_in2: in std_logic;
        reset_1: in std_logic;
        reset_2: in std_logic;
        rx_in: in std_logic;
        tx_in: in std_logic;
        a2d_clk: out std_logic;
        analog1: out std_logic_vector(7 downto 0);
        analog2: out std_logic_vector(7 downto 0);
        convst_n: out std_logic;
        den_coeff: out std_logic_vector(7 downto 0);
        num_coeff: out std_logic_vector(7 downto 0);
        rx_out: out std_logic;
        tx_out: out std_logic
  );
end component;


begin

uC_Interface_1: uC_Interface
   port map
   ( Coeff1In => data_out_sig1,
     Coeff2In => data_out_sig2,
     Coeff3In => data_out_sig3,
     Coeff4In => data_out_sig4,
     EnableIn => EnableIn,
     rst_n => rst_n,
     ReadIn => ReadIn,
     src_clk_1 => src_clk_1,
--     src_ce_1 => src_ce_1,
--     SubSysWriteIn => SubSysWriteOut_sig,
--     StatusIn => data_out_sig1,
--     DataReqOut => DataReqIn_sig,
     DataBus => DataBus
   );

wrapper_1: system_id_first_order_051105_nosim_clk_wrapper
   port map
   (
        clk => src_clk_1,
        data_in1 => data_in1,
        data_in2 => data_in2,
        reset_1 => rst_n,
        reset_2 => rst_n,
        rx_in => RX_in,
        tx_in => TX_in,
        a2d_clk => a2d_clk,
        analog1 => data_out_sig3,
        analog2 => data_out_sig4,
        convst_n => convst_n,
        den_coeff => data_out_sig1,
        num_coeff => data_out_sig2,
        rx_out => RX_out,
        tx_out => TX_out
   );

end RTL;
```

```
-- ********************************************************************************
--
-- Owner: System I.D. Team
-- Date:  11/05/05
-- File:  uC_Interface.vhd
--
-- Purpose: Interface uC to FPGA
--
--
-- ********************************************************************************

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
USE ieee.STD_LOGIC_unsigned.all;

entity uC_Interface is
  port(
    Coeff1In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 1 register
    Coeff2In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 2 register
    Coeff3In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 3 register
    Coeff4In     : in STD_LOGIC_VECTOR (7 downto 0);  -- Coeff. 4 register
    EnableIn     : in STD_LOGIC;  -- Chip EnableIn
    rst_n        : in STD_LOGIC;  -- Reset active low
    ReadIn       : in STD_LOGIC;  -- Read Signal
    src_clk_1    : in STD_LOGIC;  -- System 50MHz Clock
--    src_ce_1     : in STD_LOGIC;  -- Clock EnableIn (Not Used)
--    SubSysWriteIn : in STD_LOGIC;  -- Subsystem write
--    StatusIn     : in STD_LOGIC_VECTOR (7 downto 0);   -- Subsyst. status reg.
--    DataReqOut   : out STD_LOGIC;                      -- Data Request
    DataBus      : inout STD_LOGIC_VECTOR (7 downto 0) -- uC Data bus
     );

end uC_Interface;

architecture rtl of uC_Interface is

constant RESET_ACTIVE : std_logic := '0';

--signal StatusRegSig  : STD_LOGIC_VECTOR (7 downto 0);
signal Coeff1RegSig  : STD_LOGIC_VECTOR (7 downto 0);
signal Coeff2RegSig  : STD_LOGIC_VECTOR (7 downto 0);
signal Coeff3RegSig  : STD_LOGIC_VECTOR (7 downto 0);
signal Coeff4RegSig  : STD_LOGIC_VECTOR (7 downto 0);
signal Mux1OutSig    : STD_LOGIC_VECTOR (7 downto 0);
signal Mux1SelRegSig : STD_LOGIC_VECTOR (6 downto 0);

begin
  ClockProc1:process(rst_n,src_clk_1) begin
  if (rst_n = RESET_ACTIVE) then
--    StatusRegSig <= (others => '0');
    Coeff1RegSig <= (others => '0');
    Coeff2RegSig <= (others => '0');
    Coeff3RegSig <= (others => '0');
    Coeff4RegSig <= (others => '0');
  elsif (src_clk_1'event and src_clk_1 = '1') then
--    StatusRegSig <= StatusIn;
    Coeff1RegSig <= Coeff1In;
    Coeff2RegSig <= Coeff2In;
    Coeff3RegSig <= Coeff3In;
    Coeff4RegSig <= Coeff4In;
  end if;
  end process ClockProc1;

  ClockProc2:process(rst_n,src_clk_1) begin
  --if (rst_n = RESET_ACTIVE) then
    -- Mux1SelRegSig <= (others => '0');
  if (src_clk_1'event and src_clk_1 = '1'
                      and ReadIn = '0' and EnableIn = '1') then
    Mux1SelRegSig <= DataBus(6 downto 0);
  end if;
  end process ClockProc2;
```

```
--   ClockProc3:process(rst_n,src_clk_1) begin
--   if (rst_n = RESET_ACTIVE) then
--      DataReqOut <= '0';
--   elsif (src_clk_1'event and src_clk_1 = '0'
--                              and ReadIn = '0' and EnableIn = '1') then
--      DataReqOut <= DataBus(7);
--   end if;
--   end process ClockProc3;


   CombProc:process(Mux1SelRegSig,Coeff1In,Coeff2In,
                            Coeff3In,Coeff4In,EnableIn,ReadIn,Mux1OutSig,rst_n)
               begin
   case Mux1SelRegSig is
      when "0000000" =>
         Mux1OutSig <= Coeff1In;
      when "0000001" =>
         Mux1OutSig <= Coeff2In;
      when "0000010" =>
         Mux1OutSig <= Coeff3In;
      when "0000011" =>
         Mux1OutSig <= Coeff4In;
      when others =>
         Mux1OutSig <= (others => '0');
      end case;
   if(EnableIn = '1' and ReadIn = '1') then
      DataBus <= Mux1OutSig;
   else
      DataBus <= (others => 'Z');
   end if;

   end process CombProc;
end rtl;
```

## Appendix H: C Code

```
//-------------------------------------------------------------------------
//  Project Name: main.c
//  Author:  J. Susong
//  Date: 10/04/2005
//  Description:  This code reads data from system I.D. FPGA
//
//-------------------------------------------------------------------------

#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#include "math.h"
#include "stdlib.h"
#include "string.h"
#include <m8c.h>        // part specific constants and macros

#pragma interrupt_handler Timer16_1_ISR
// #pragma interrupt_handler GPIO_ISR

//Function Prototypes.
void clear_display(void);
void GPIO_ISR(void);
void lcdpr_cs(char a, char b,const char *c);


// Global variables.
char timer1_flag;
char gpio_flag;

BYTE rxBuf[20];
BYTE txCBuf[20];

void main()
{
  char k,l,m,p,string[6];
  float n;
  int *status,j;
  char *strPtr,*strPtr2,*ptr1;

  timer1_flag = 0;

  UART_1_CmdReset();                  // Initialize receiver/cmd buffer
  UART_1_IntCntl(UART_1_ENABLE_RX_INT); // Enable RX interrupts
  UART_1_Start(UART_PARITY_NONE);     // Enable UART

  DAC8_1_Start(DAC8_1_FULLPOWER);

  LCD_1_Start();                              // Initialize LCD

  M8C_DisableIntMask(INT_MSK0,INT_MSK0_GPIO);         // Disable GPIO interrupts.

  Timer16_1_EnableInt();

  M8C_EnableGInt;

  Timer16_1_Start();

  Timer16_1_WritePeriod(320);

  lcdpr_cs(0,1,"System I.D. Tool");
  lcdpr_cs(6,4,"Version 1.0");

  while(TRUE)
  {
    DAC8_1_WriteStall(0);

    PRT0DR &= ~0x01; // Set RESET low
    PRT0DR &= ~0x04; // Step Out low

    UART_1_CPutString("\r\nSystem I.D. Tool Version 1.0\r\n");
    UART_1_CPutString("\r\nPress Enter To Start the Test");
```

```
while( UART_1_bCmdCheck()==0 );
strPtr2 = UART_1_szGetParam();

for(j=0;j<500;j++)
{
   while (timer1_flag == 0);
   timer1_flag = 0;

   if(j==150)
   {
      PRT0DR |= 0x05; // Set Step high and Reset high
      //UART_1_CPutString("\rStep occurs\r\r");
   }

   clear_display();
   lcdpr_cs(0,0,"Den Coeff.");
   lcdpr_cs(1,0,"Num Coeff.");
   lcdpr_cs(2,0,"System O/P");
   lcdpr_cs(3,0,"System I/P");


   for(l=0; l < 4; l++)
            {
           //Write Sequence
    PRT0DR &= ~0x40; // Set RD-WR low

    PRT2DR = 0x80+l; //Select Mux Channel l
    PRT0DR |= 0x80; // Set ENABLE high

    PRT0DR &= ~0x80; // Set ENABLE low
    PRT2DR = 0xFF;  // Allow the data bus to be driven

    //Read Sequence

    PRT0DR |= 0x40; // Set RD-WR high
    PRT0DR |= 0x80; // Set ENABLE high

    k = PRT2DR;

          n = (float)k * 0.0078125;

    LCD_1_Position(l,11);
    LCD_1_PrString(ftoa(n,status));

    UART_1_PutString(ftoa(n,status));
              UART_1_CPutString("\t");

    PRT0DR &= ~0x80; // Set ENABLE low

   }// end of for

  UART_1_CPutString("\r");

 }// end of for

  UART_1_CmdReset();

  for(j=0;j<500;j++)
{
   PRT0DR &= ~0x04; // Step Out low
   DAC8_1_WriteStall(0);
   while (timer1_flag == 0);
   timer1_flag = 0;
          }

  for(j=0;j<500;j++)
{
   PRT0DR &= ~0x01; // Set RESET low
   DAC8_1_WriteStall(0);
   while (timer1_flag == 0);
   timer1_flag = 0;
          }
```

```
   }// End of While loop
} // End of main


void GPIO_ISR(void)              // GPIO ISR
{
  lcdpr_cs(1,0,"isr1 ");
//  LCD_1_PrHexByte(gpio_flag);
  gpio_flag++;
}
void clear_display(void)
{
              char i;
              for(i=0; i < 4; i++)
              {
                LCD_1_Position(i,0);
                LCD_1_PrCString("                  ");
              }
}
void lcdpr_cs(char a, char b,const char *c)
{
  LCD_1_Position(a,b);
  LCD_1_PrCString(c);
}
void Timer16_1_ISR(void)
{
  timer1_flag = 1;
  PRT0DR^=0x02;      //equivalent to instruction "xor reg[PRT0DR],0x02"
}
```