Signal Processing Using Digital Technology

Proposal By:
Jeremy Barsten
Jeremy Stockwell

Advisors:
Dr. Vinod Prasad
Dr. Thomas Stewart

Senior Capstone Project

December 12, 2002

Abstract

This project deals with the design and implementation of a general-purpose signal processor using digital technology. Through various types of digital design, specifically VLSI and FPGA technology, a general digital signal processor will be designed and implemented on a Xilinx FPGA board. This processor will consist mainly of the following sub-system blocks: a multiplier and adder, a digital filter, and a type of “shift-and-rotate” data management. Once completed, this processor can be easily adapted for a variety of specific applications.

Table of Contents

Introduction

In the world today, digital technology is ever growing, and the development of digitally based products is rising. Various industries such as audio, video, and cellular industry rely heavily in this digital technology. A great part of this deals with digital signal processing. This aspect in engineering has gained increasing interest, especially with much of the world now turning to wireless technology and its applications to keep businesses and industries connected. The world of digital technology is certainly one that will be present for many years to come.

Certain areas of engineering and technology have their own application-specific digital signal processing that accompanies it. The purpose of this project is to mirror these processors by designing and implementing a general-purpose digital signal processor through the use of VHDL, field-programmable gate arrays (FPGA), and VLSI. The basic overall block diagram for this processor is shown in Figure 1. For the overall system, there is only one input signal, specific to the type of application, and one output signal, the digitally processed signal.

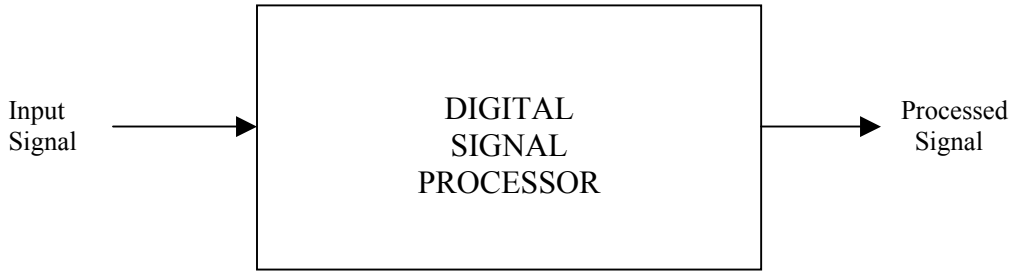

Figure 1: Overall block diagram of the digital signal processor.

This processor contains many different subsystems, each of which aid in processing the signal, depending on the specific application. These blocks, along with preliminary work done for them, will be discussed in the next sections.

They include the following: signal converters, a digital filter, a multiplier and adder, and data management subsystems.

Signal Converters

Many of the signals dealt with today are analog in nature. These signals cannot be sent directly to the processor without being converted first to a digital signal. Therefore, an analog-to-digital converter is required at the input stage of the processor. Also, if a signal is sent to the processor as an analog one, then the processed signal should produce an analog one as well. For that reason, a digital-to-analog converter is required at the output stage of the processor as well. This is shown in the block diagram in Figure 2 below.

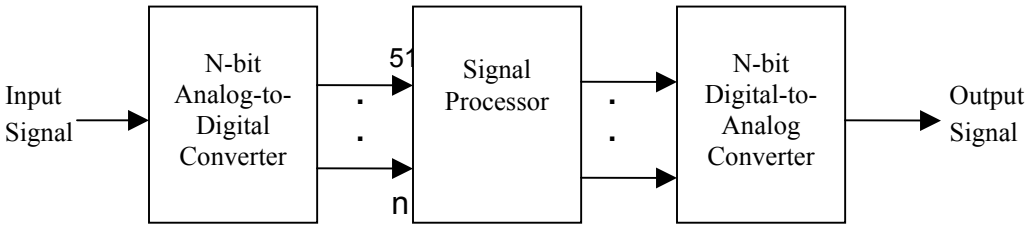

Figure 2: Block diagram showing signal converters' connection.

At the current time, the specific converters for this application have not been chosen. This is due to the fact that the exact number of bits used in the processor still has not been determined. However, once this is decided upon, the correct analog-to-digital and digital-to-analog converters will be selected and implemented in the design.

Digital Filter

As the digital signal enters the processor from the analog-to-digital converter, it will pass though a digital filter. This processor is designed so that it can be used in many different applications. The digital filter is what makes this

possible. For this processor, a second order Direct Form II realization of an IIR filter will be used. This design can be seen in Figure 3. By utilizing this format,

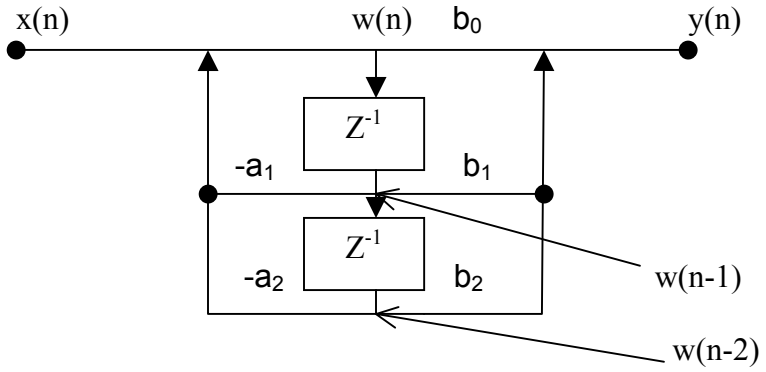

Figure 2: Direct Form II realization of IIR Filter

the following equations can be derived:

$$y(n)=b_0*w(n) + b_1*w(n-1) + b_2*w(n-2), \quad (1)$$

where $w(n)=x(n) – a_1*w(n-1) – a_2*w(n-2)$. This then simplifies down to

$$Y(n)=b_0*x(n) + b_1*x(n-1) + b_2*x(n-2) – a_1*y(n-1) – a_2*y(n-2). \quad (2)$$

The coefficients b_0, b_1, and b_2 are what allow the processor to adapt to many different applications. By changing these coefficients, the frequency range of the filter will change, and, thus, the specific application of this processor will also change. (Different applications, cellular phones, audio, video, etc. . ., operate at distinct frequencies.) The coefficient inputs, the signal input, and the filtered output are shown in the filter block diagram in Figure 3.

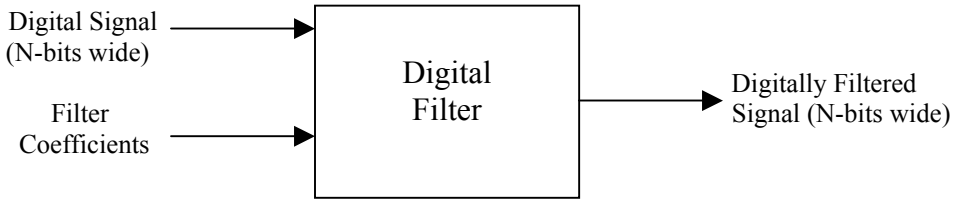

Figure 3: Block diagram of the general digital filter.

Adder and Multiplier Stage

Addition and multiplication are the means by which a signal is processed. This stage is the most important in the development of the processor. It will consist of a n-bit by n-bit multiplier, which will result in a 2∗n-bit number. This number will then be delivered to a 2∗n-bit adder where it will be summed with previous values that have been stored in memory. (Again, the actual number of bits that will be used is still not determined.) As shown in Figure 4, the filtered signal and previous values will be fed into the input stage of this multiplier and adder, and the digitally processed signal is the output of this block.

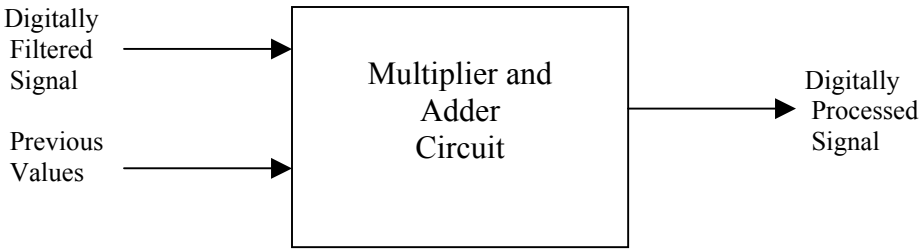

Figure 4: Block diagram of the multiplier and adder circuit.

There are many different ways in which this block of the project can be implemented. It can be constructed as a serial multiplier and adder, a parallel one, or a combination of the two. The full adder and multiplier will be implemented on a Xilinx FPGA board. However, there is a tradeoff here between the area of the board consumed by the adder and multiplier and the speed of the system. Also, an adder and multiplier will be designed and implemented using VLSI. Again, however, the full implementation of these on a fabricated chip would be too large for this project, so a smaller adder and multiplier will be designed instead. (This will be discussed later in the preliminary work section.)

Data Management

The final element in this signal-processing project is a data management device, shown in Figure 5. Here, previously values are truncated, stored, and then fed back to the adder and multiplier block. Also, the coefficients of the system are also stored here. Both Figure 2 and Equation 2 involve values that

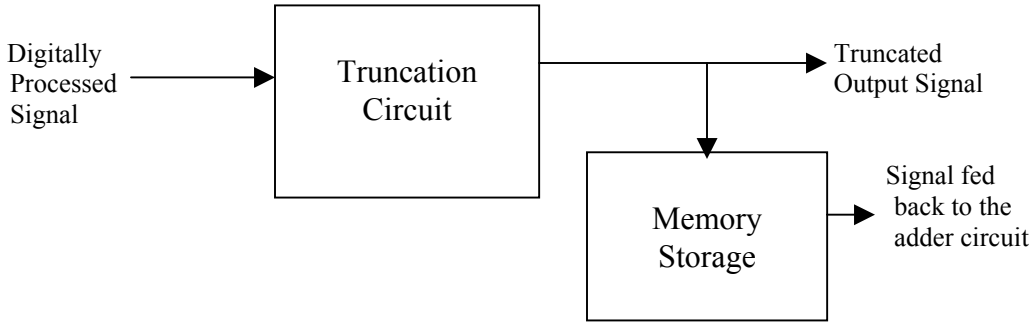

Figure 5: Block diagram of the truncation circuit and memory storage circuitry.

Are delayed as they are sent through the system. These delayed values will be stored in this memory and delivered at the appropriate time to the adder and multiplier subsystem. Therefore, part of the memory will be set up in a “shift-and-rotate” fashion where previous values will be discarded once they have been used. A diagram showing the consecutive steps of the memory is show in Figure 6.

This data management also plays a big part in the reliability of the processor. Depending on the inputs to the adder and multiplier, a situation may arise where the resultant value may be evaluated incorrectly. If there is a carry out of the most significant bit from the adder, the truncation block will discard that value and return the maximum possible value allowed. Also, the carry out could also be read as a sign bit, and, thus, the value, which should be a large positive

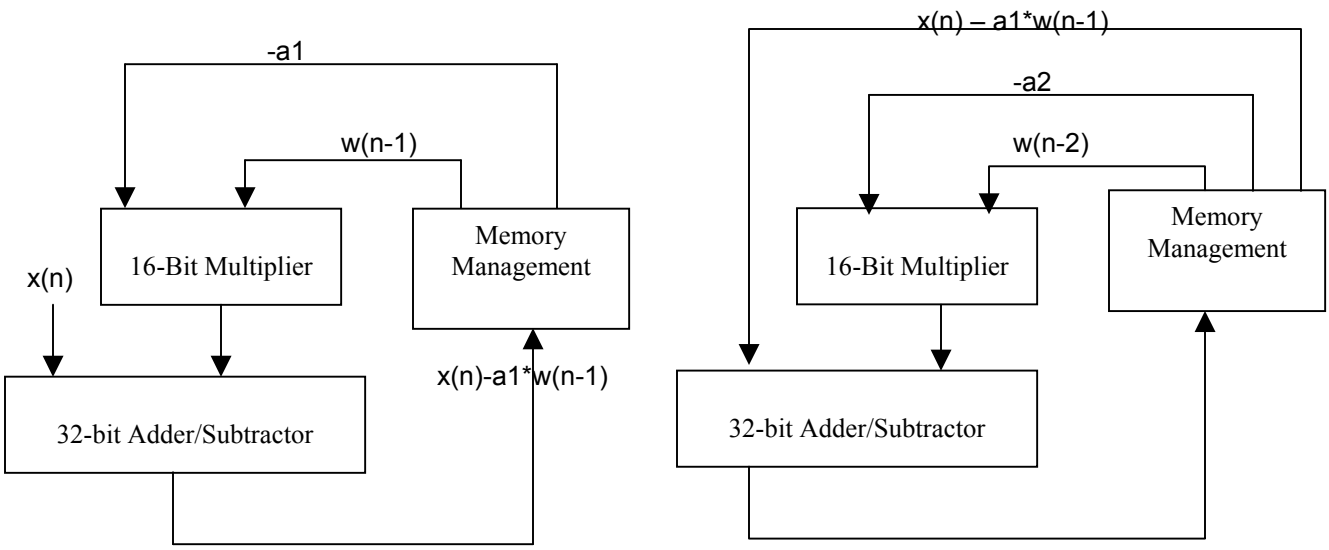

Figure 4: Block diagram showing consecutive steps in the data management.

Number, is seen as a large negative number. Here, the processor will recognize this, disregard the sign bit, and return the value to the correct large positive number.

Preliminary Work

Some preliminary work has been done to implement a full adder using VLSI. Using L-Edit Pro v.8.2, a CMOS circuit was designed to implement the basic one-bit adder circuit shown in Figure 5[1]. Once this simulates correctly, four of them will be cascaded to form a four-bit full adder. However, at this time, the L-Edit design is still incomplete. Once completed, the circuit will be cascaded as stated before, and work on the multiplier can then begin.

The multiplier will be implemented in VLSI by utilizing cellular array multiplication. Here, the two binary values are sent into the multiplier as shown in Figure 6. They are multiplied and summed according to their significance.

[1] This circuit was taken from Physical Design of CMOS Integrated Circuits Using L-EDIT by John P. Uyemura.

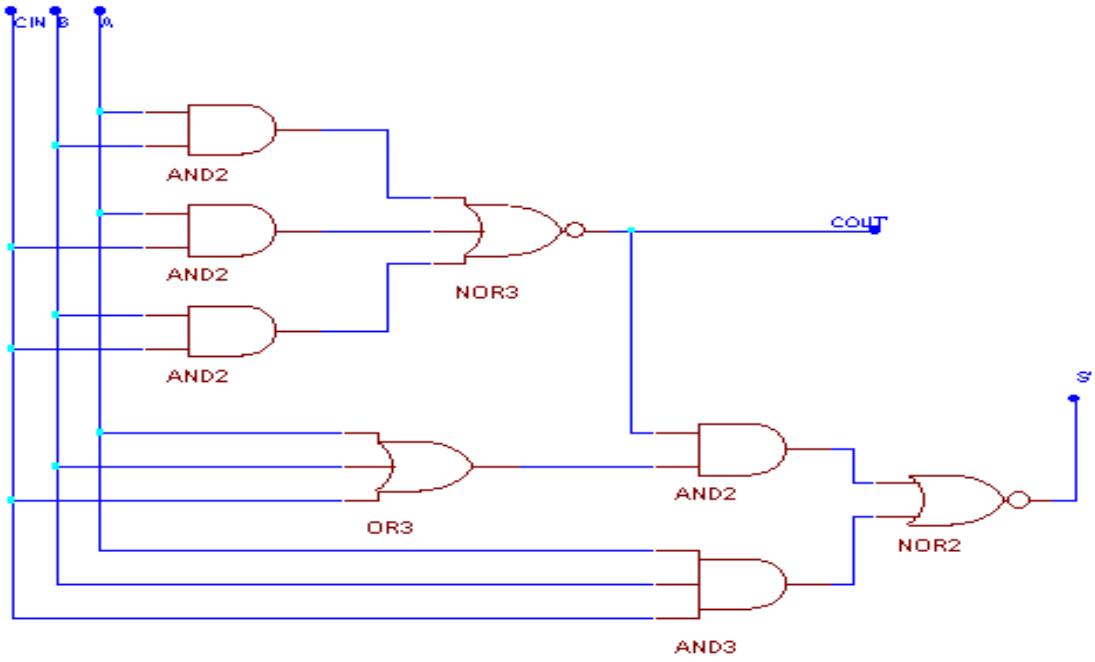

Figure 5: Logical diagram for a one bit full-adder circuit.

The more significant bits are shifted so that when added, they are in the correct location. The carry bits are then sent though the entire adder (which are represented by the diagonal lines in the figure).

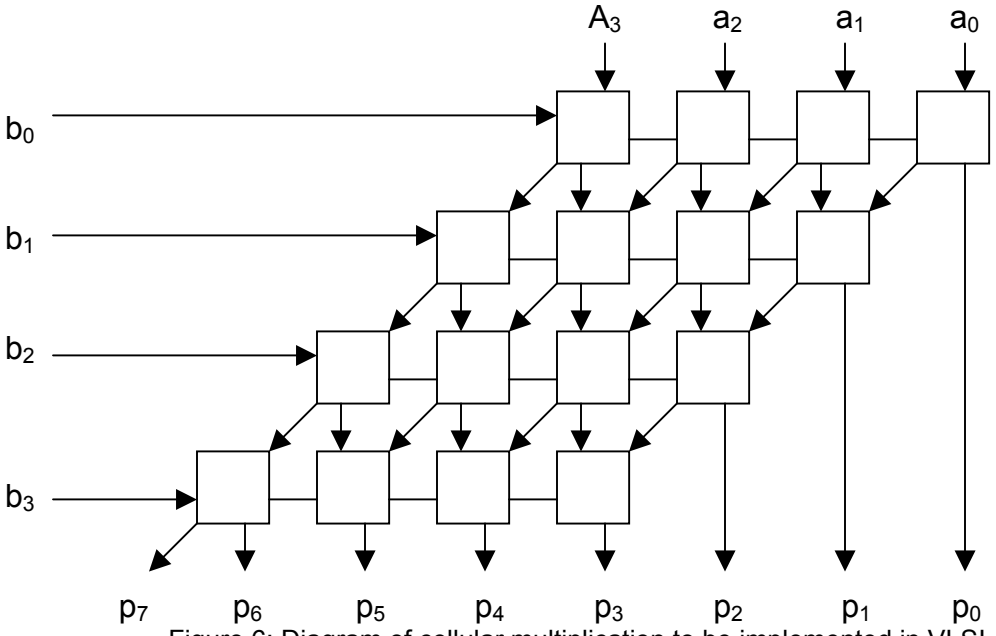

Figure 6: Diagram of cellular multiplication to be implemented in VLSI.

Work has also been done on the adder and multiplier with FPGA's and VHDL. The first task was to determine the capabilities of the Xilinx compiler. To do this a ripple carry adder and a carry lookahead adder were created with VHDL.

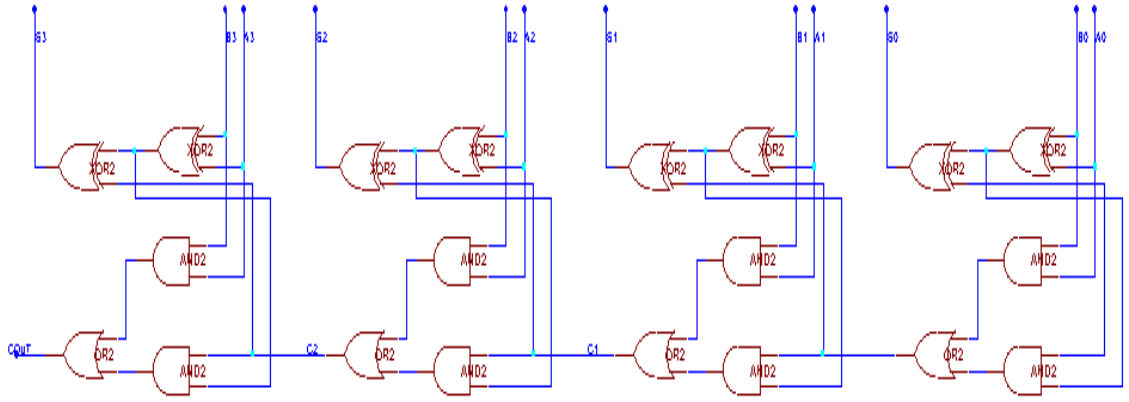

Figure 7: Logical diagram for a 4-bit ripple carry adder.

A carry ripple adder has a delay of 2n+2 gate delays. To decrease this delay a carry lookahead adder can be created. This type of adder requires more logic, but the speed advantage is significant. For example, a 16-bit carry ripple adder results in 34 gate delays, while a carry lookahead adder only has 10 gate delays.

To test the Xilinx compiler a simple a+b=c adder was created in VHDL. Then all three adders were simulated and the delays were compared to determine what type of adder the compiler created. The results showed that the complier created a carry lookahead adder or an equivalent one.

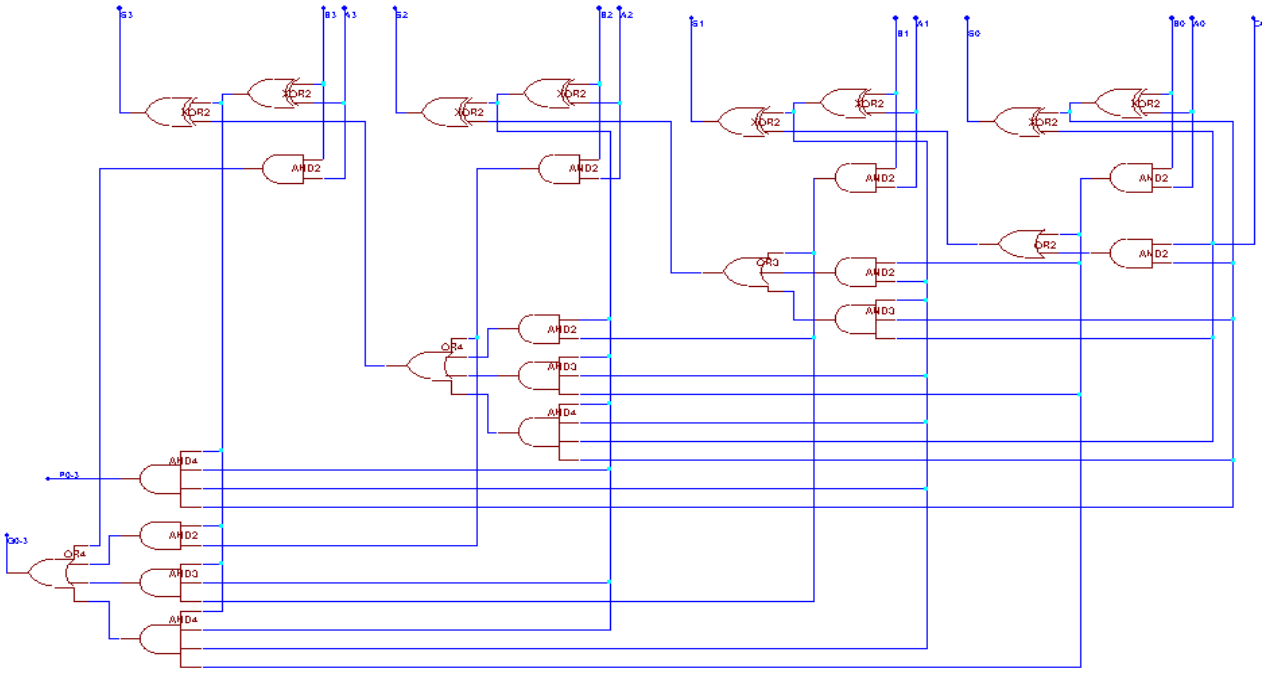

Figure 8: Logical diagram for a 4-bit carry look ahead adder

	Carry Lookahead Adder	Ripple Adder	Simple Adder
Delay Times	11.05ns	22.43ns	11.05ns

Table 9: Table of delay times from CLA, carry ripple, and simple adders

The next task was to determine the best trade-off between size and speed for the multiplier circuit. Two types of multipliers, serial and parallel, were investigated. A serial multiplier is easier to create and takes less area on the FPGA, but the speed is decreased nearly n times for a n-bit multiplier. A parallel multiplier has the advantage of speed, and would be utilized if area was of no concern.

To conclude what course to take, first, a serial multiplier was created. Digital signal processors utilize signed numbers, so a signed multiplier needed to be designed. A parallel multiplier was created using the Xilinx compiler. 4-bit, 8-bit, and 16-bit multipliers were created for both.

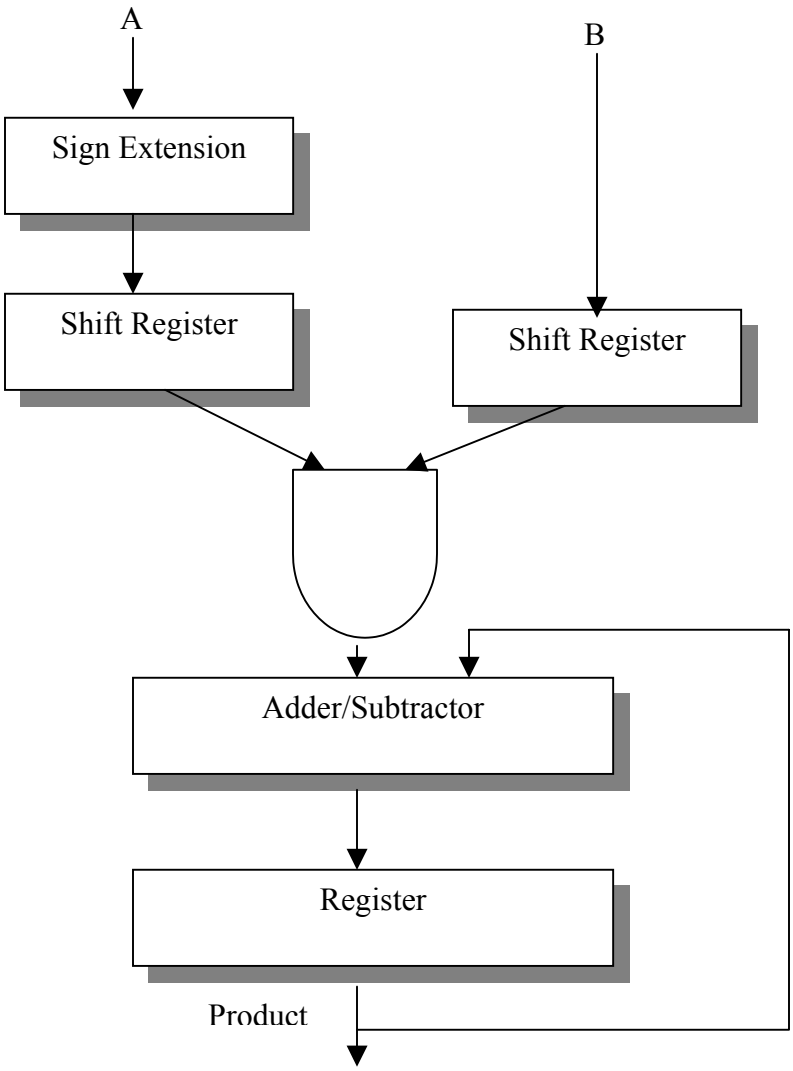

Figure 10: Data path for signed 2's complement serial multiplier

	4-bit		8-bit		16-bit	
	Area	Delay	Area	Delay	Area	Delay
Serial Multiplier	7.14%	96.88ns	13.52%	210.8ns	24.49%	503.68ns
Parallel Multiplier	10.71%	27.41ns	2781%	38.76ns	107.40%	51.06ns

Table11: Table of area used and delay from serial and parallel multipliers.

Examining these results shows that as the number of bits increases the delay of the serial multiplier increases in a linear fashion, and the area used by the parallel multiplier increases exponentially. To combat the area problems that

arise with the parallel multiplier, a combination of parallel and serial multipliers will be used. The exact arrangement will be determined in future lab work.

Standards and Patents

After searching the Internet, specific patents and standards dealing with general digital signal processors were not readily available. However, there were many different ones that dealt with application specific signal processors. Since the application of the processor has yet to be determined, the specific standards and patents relating will be research more thoroughly.

Schedule

There are still many things that need to be accomplished throughout next semester. The first task, and probably the most important one, is the adder and multiplier blocks. Throughout next semester during January and February, this area will be researched further, designed, implemented, and tested on a FPGA board. Also, a smaller adder and multiplier designed using VLSI will be implemented and simulated to test it. By March, the specific application for this processor will be determined. Through use of MATLAB, the exact filter coefficients will be obtained, and the filter will then be designed accordingly. In April, the entire processor design will be implemented, and any major troubleshooting will be done during this time. Therefore, by May, the entire project will be completed, and preparation for the final presentation will begin.

Parts List

The following is a list of hardware and software programs that will be used in this project:

- Xilinx Foundation Software
- Leonardo Spectrum
- L-Edit Pro v8.2
- PSpice Circuit Simulator
- MATLAB
- Xilinx XC4005ePC84 FPGA Board
- A/D and D/A converters

The only parts in this list that may need to be ordered are the converters. However, as stated before, the actual models have not been decided upon. Once determined, a request to order specific models will be made if the chips are not already readily available.

Bibliography

Barsten, Jeremy, and Jeremy Stockwell. "Signal Processing Using Digital Technology: Functional Description." Bradley University, 2002.

Barsten, Jeremy, and Jeremy Stockwell. "Signal Processing Using Digital Technology: System Block Diagram." Bradley University, 2002.

Goslin, Gregory Ray. A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application-Specific Digital Signal Processing Performance. Xilinx, Inc. 1995. San Jose, CA: 1995.

Kime, Charles R., and M. Morris Mano. Logic and Computer Design Fundamentals: Second Edition. Upper Saddle River, NJ: Prentice Hall, 2000.

Uyemura, John P. Physical Design of CMOS Integrated Circuits Using L-EDIT. Boston, MS: PWS Publishing Company, 1995.