Implementation of Dynamic Compensation for Analog-to-Digital Converters

By
Norman G. Dutil

B.S. University of Maine, 1994

A THESIS
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science
(in Electrical Engineering)

The Graduate School
University of Maine
August 1997

Advisory Committee:

Donald M. Hummels, Associate Professor of Electrical and Computer Engineering, Advisor

Fred H. Irons, Castle Professor of Electrical and Computer Engineering

Eric P. Beenfeldt, Lecturer in Electrical and Computer Engineering

Bruce E. Segee, Assistant Professor of Electrical and Computer Engineering

LIBRARY RIGHTS STATEMENT

Implementation of Dynamic Compensation for Analog-to-Digital Converters

By Norman G. Dutil

Thesis Advisor: Dr. Donald M. Hummels

An Abstract of the Thesis Presented

in Partial Fulfillment of the Requirements for the

Degree of Master of Science

(in Electrical Engineering)

August 1997

Signal distortion is introduced during the analog-to-digital conversion process. This distortion results in reduced ADC dynamic range. Analog-to-digital converter compensation has been shown to improve dynamic range by as much as 20 dB using software methods.

This thesis shows that ADC compensation results demonstrated using software techniques can be duplicated in electronic hardware. It accomplishes this by implementing the Phase-Plane Compensation(PPC) method using an off-the-shelf ADC. This method corrects the ADC samples based on the ADC state and estimate of the input signal slope.

To implement PPC, circuits were designed to perform the functions implemented in the software solution. Circuits include the ADC Compensator, Difference Generator or Slope Estimator, and various interfaces to the University of Maine ADC Testbed. This thesis provides the circuit design details.

Application of these circuits to an Analog Devices AD9027 ADC show that compensation in hardware is as effective as software methods. A dynamic-range

before-and-after comparison shows the dynamic-range improvements in hardware compensation to be equivalent to software results.

ACKNOWLEDGMENTS

This thesis could not have been written without the constant support and encouragement from Professors Don Hummels and Fred Irons. They have been patient, understanding and generous with their time and resources. I would also like to thank Ernie Hollis at Sanders for providing an opportunity to grow professionally and investigate state-of-the-art ADCs. I am also indebted to the staff at National Semiconductor for their patience, constant encouragement, and motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Demand for high-frequency, wide dynamic-range analog-to-digital converters(ADC) by industry and consumer electronics continues to grow. The fields of digital communications and radar receiver design continue to push the digital domain closer to the antenna. Unfortunately, distortion sources that limit dynamic range are inherent in the ADC design and manufacturing. Designers continue to develop new and improved architectures which limit distortion to meet new requirements. Additionally, manufacturing processes will continue to improve as studies of the data identify and remove sources of distortion. These improvements take time and require significant corporate investment in research, development, and manufacturing. While improvements progress, a less-costly method of improving existing off-the-shelf ADCs is necessary.

1.1 Background

Early compensation techniques attempted to remove the distortion from the ADC output samples by subtracting an error term from the ADC output based exclusively on the binary output value. ADC error was considered highly dependent upon the amplitude of the input signal. While this method compensated signals whose frequencies fell within a narrow bandwidth of the calibration frequency of the ADC it failed to work well over the full ADC Nyquist bandwidth.

Irons & Rebold showed that ADC non-linearities are strongly related to input signal amplitude and first derivative [6]. Phase Plane Compensation(PPC) of ADCs presented by Irons and Rebold [6] reduced harmonic distortion by calibrating the ADC over a full Nyquist frequency band. This method utilizes calibration procedures that compute error terms over a range of input signal amplitudes and frequencies.

Kennedy [7] presented an iterative method of ADC calibration and optimized input signal slope estimation. A non-iterative ADC calibration procedure based on measured harmonic distortion was developed later [3].

Several efforts have investigated distortion sources relating to ADC architecture. Selection of ADC error basis functions to match the ADC architecture [5] has been used to examine distortion caused by comparator hysteresis, signal dependent sample-time errors, and reference voltage errors for flash and folding converters. Compensation for intermodulation distortion within time-interleaved converters using modified compensation techniques was also considered [4].

A more accurate method of measuring the ADC error characteristic uses sinewave histograms [8]. For simulated ADCs, this method removes all errors based on the output code and input signal slope.

1.2 Purpose of the Research

The purpose of this thesis is to develop and demonstrate an ADC compensation solution in electronic hardware. The phase-plane compensator(PPC) presented here is one such circuit. It is designed for universal interfacing to various ADCs, slope estimating hardware, and host computers.

1.3 Thesis Organization

This thesis is divided into 5 chapters. Chapter 2 introduces possible phase-plane compensator architectures and input slope estimation schemes. Chapter 2 also gives the design requirements for building a generic ADC compensator. Chapter 3 illustrates the compensator circuits and interfaces. To make the compensator useful, additional circuitry must be added to support interfacing an off-the-shelf

ADC. Chapter 4 gives the design and implementation of additional circuitry required to support an application. It also presents the compensator operating procedure. Chapter 5 gives application results. Finally, conclusions and suggestions for future work are given.

CHAPTER 2

ADC Compensation Methods

This chapter introduces the concept of ADC compensation. An introduction to static compensation illustrates the fundamental concept followed by a presentation of the PPC architecture. Derivative estimation methods are discussed next. They give the reader four methods of implementing an integral component to the PPC, the input slope estimator. The reader is next introduced to the concept of the phase-plane and error curves. Next, a discussion of the current compensation methods is given. The design and hardware requirements for implementing a real-time hardware compensator complete the chapter.

2.1 Introduction

The purpose of compensation is to remove harmonic distortion introduced into the sampled signal by the ADC. A non-linear sampling process adds undesirable distortion to a sampled signal. This distortion reduces the dynamic range by elevating harmonic power levels. Some sources of distortion include amplifier compression, timing jitter, track-and-hold limitations, and hysteresis.

The process of quantization also produces elevated harmonic levels [6]. However, quantization harmonics are not the result of ADC imperfections and must be removed in order to observe true distortion. Dithering the ADC input signal has the effect of distributing quantization harmonic power over the entire sampling bandwidth. It is accomplished by adding filtered white noise to the ADC input signal with equivalent power to that of the quantization power.

Once quantization harmonics are reduced, measurement of ADC error may be accomplished using an off-line calibration process. Calibration involves the estimation of the average error for each ADC output. Error terms or compensation values represent the difference between an ideal uniform staircase ADC characteristic and values obtained through measurement, typically within a few least significant bits(LSBs) of the idealized converter. Error terms are subtracted from the ADC outputs to obtain compensated samples.

2.2 Static Compensation

Static compensation illustrates the concept of error removal from an ADC output stream. The principle involves removing errors from sample values or states. In this instance, the errors generated by an off-line calibration process makes up a one-dimensional vector. Each output value has one error associated with it. Equation 2.1 shows the compensated samples z_i, where x_i is the ADCs i-th output sample and $e(x_i)$ is the error at the i-th state.

$$z_i = x_i - e(x_i). \tag{2.1}$$

While static compensation effectively removes distortion caused by some sources like amplifier compression, it fails to take into effect the first-order effects of input signal slope on error. Also, this method has limited effectiveness at reducing distortion over bandwidths greater than 10 % of the sampling rate [6]. A more effective compensation method that utilizes information about the dynamic nature of the input signal is required.

2.3 Phase-Plane Compensation

Phase plane compensation, proposed by Irons and Rebold [6, 9] , removes converter distortion based on the output value and an estimate of the corresponding input signal slope. Slope estimation implies a relationship to frequency. This relationship determines the distortion component due to ADC dynamic error sources. Equation 2.2 shows that the error term of (2.1) now depends on the input slope estimate, y_i, as well as state, x_i.

$$z_i = x_i - e(x_i, y_i). \tag{2.2}$$

2.3.1 Architecture

PPC architecture is illustrated in Figure 2.1. An analog input is applied to an ADC and a derivative estimator. Sample and slope estimate values are synchronously applied to the compensator hardware. Within the hardware, these two values index a two-dimensional error table created when the converter is calibrated. Compensator hardware typically subtracts the error from the ADC output or state yielding a compensated sample. Alternatively, the complete corrected sample could be indexed and output directly. This architecture lends itself to straight-forward digital design. However, implementation of the input signal slope estimator requires consideration.

2.3.1.1 Derivative Estimation

Four methods of estimating the input signal slope are presented with emphasis on the implementation feasibility. Limiting factors for any technique include the hardware and bandwidth requirements of the application. Typically, bandwidth for a derivative estimator would be on the order of the ADC input bandwidth. Hardware requirements for methods illustrated are summarized in

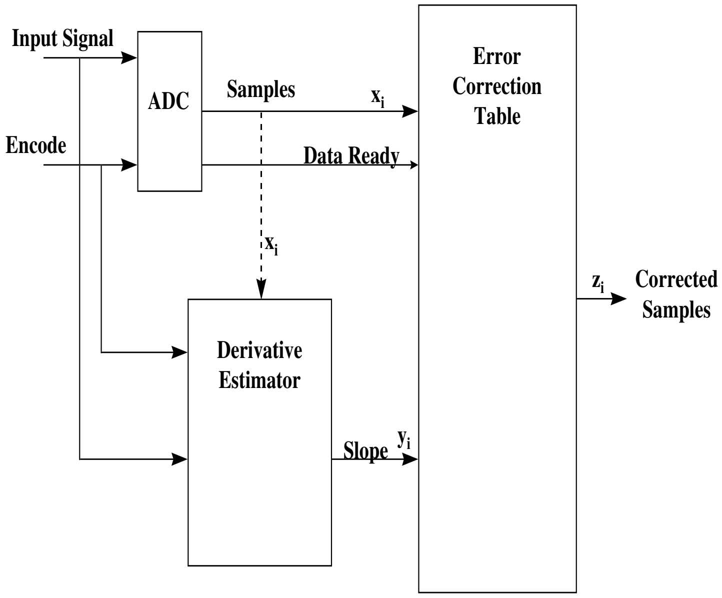


Figure 2.1: PPC Architecture

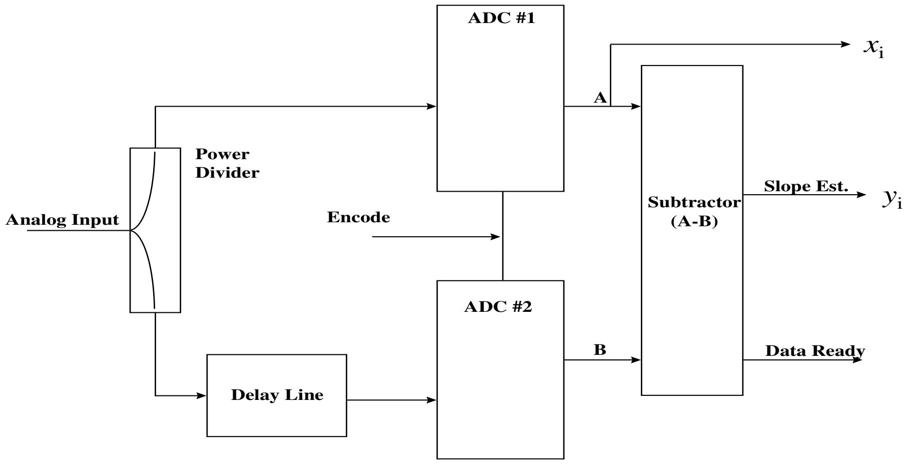


Figure 2.2: Dual-Sample Difference Generation

Table 2.1 following the discussion.

The dual-sample method explored by Kennedy [7] is illustrated in Figure 2.2. A divided input signal is applied to two synchronous ADCs. ADC #1 is the converter to be compensated and serves a dual purpose. First, it provides samples of the input signal to the subtractor circuit. Secondly, Figure 2.1 shows samples applied to the compensator hardware for indexing an error table. ADC#2 is necessary for sampling a delayed version of the input signal. The difference between the normal sample and the delayed sample is determined by the subtractor. This difference in bits provides the required slope estimate. Kennedy demonstrated one method for optimizing the time-delay based on the number of converter bits and sample period [7].

An analog differentiator method is shown in Figure 2.3. First the analog input is divided and applied to the ADC to be compensated and an analog differentiator. The second ADC samples a scaled version to the analog differentiator output yielding the slope estimate directly. Note that ADC #2 does not have to

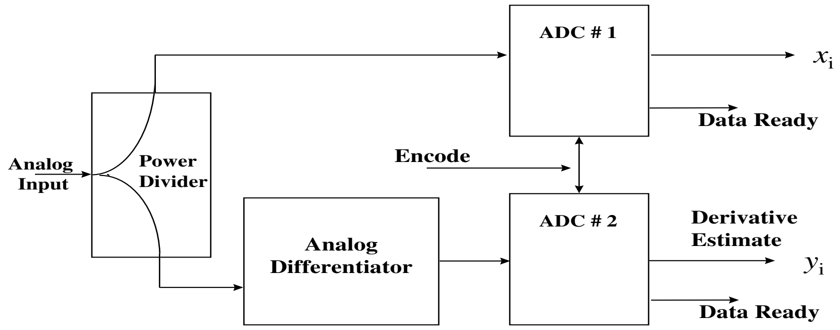


Figure 2.3: Analog Differentiator Slope Estimation

have the same resolution as ADC #1. However, it does have to operate at the same sample rate and latency.

The delayed-sample method, illustrated by Figure 2.4, requires the least amount hardware but is bandwidth limited. The ADC output captured by the register is delayed by a full sampling period. The previous sample is subtracted from the current ADC sample yielding a backward difference. The limitation of this method lies in the sample rate. If the ADC is sampling at twice the input signal frequency, then the difference could range from 0 to full-scale.

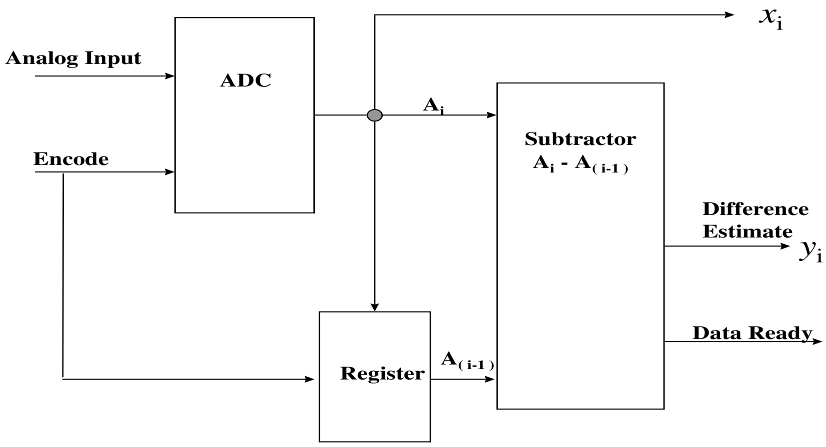


Figure 2.4: Delayed-Sample Slope Estimation

A fourth method using a discrete time filter, illustrated in Figure 2.5, is a flexible method utilizing DSP algorithms to implement the slope estimator. The benefit of this method lies in the ability of DSP devices to calculate a slope estimate using multiple samples.

Table 2.1 summarizes the hardware requirements for the four methods of difference generation.

2.3.1.2 Compensator

The final block of the PPC architecture is the compensator. The purpose of the compensator is to replace distorted ADC samples with compensated samples based on the state and slope estimate of the signal. Functionally, this can be accomplished in many ways. Essentially, an error table must be stored in memory and a means of indexing that table is required. Error terms or completely corrected samples could be stored given sufficient storage space.

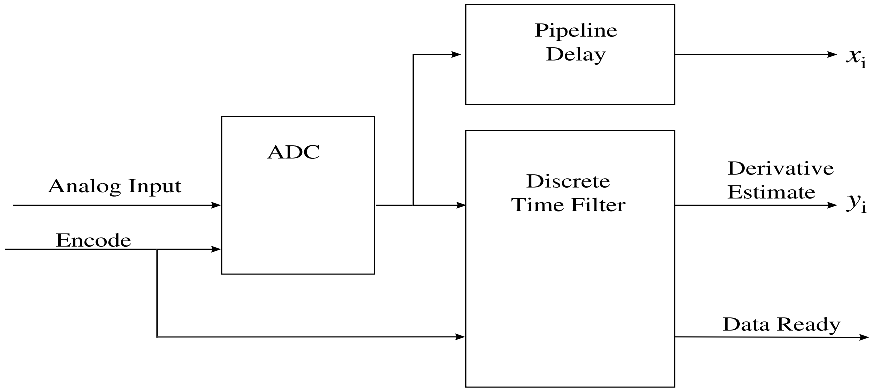


Figure 2.5: Discrete Time Filter Slope Estimation

Estimation Method	Hardware Requirements	Operating Bandwidth
Dual-Sample	2 identical ADCs 1 power splitter	$\frac{1}{2}$ Sampling Frequency
Analog Differentiator	2 non-identical ADCs 1 power splitter	$\frac{1}{2}$ Sampling Frequency
Delayed Sample	1 Power Splitter digital hardware 1 ADC	$\leq \frac{1}{4}$ Sampling Frequency
Discrete Time Filter	Digital Signal Processing Device	Algorithm Dependent
	1 ADC	

Table 2.1: Difference Generator Summary

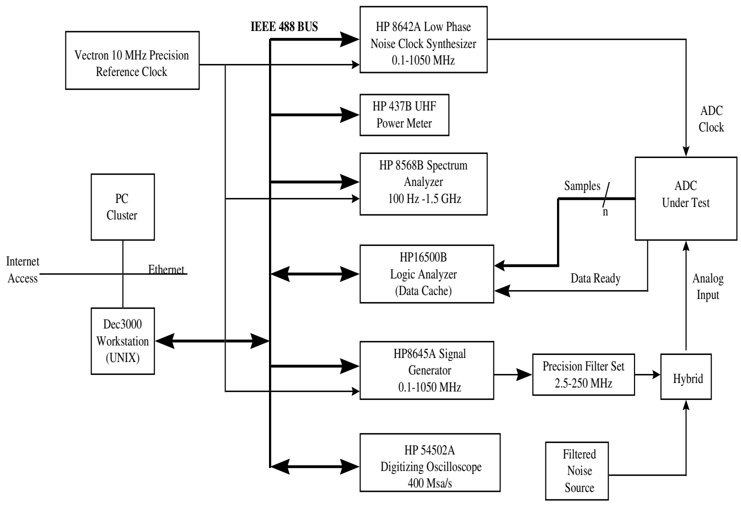


Figure 2.6: University of Maine ADC Testbed

2.3.2 Calibration

Initial calibration of an ADC on a dedicated ADC testbed is required to obtain an error table. Calibration refers to the process of measureing the ADC error characteristic over a specified operating bandwidth and input range. Calibration bandwidth is limited to converter bandwidth and amplitude is limited by the operating range. Figure 2.6 shows the hardware used by the University of Maine to calibrate ADCs. Command of the IEEE-488 networked instrumentation is provided by a DEC3000 workstation. Instrument synchronization is maintained by an oven-controlled Vectron 10-MHz precision reference clock. A sinusoidal input signal, used to calibrate the ADC, is provided by a HP8645A signal generator fed through a precision filter set maintaining signal purity to -95 dB. A noise source provides dither for spreading quantization harmonic power over the entire converter bandwidth. A HP8642A low phase noise clock synthesizer maintains consistent converter clocking. ADC output sample collection is performed by the HP16500B Logic Analyzer acting as the sample cache. Post processing of the samples is accomplished by transfering the cache collected samples over the IEEE-488 bus to a DEC3000 workstation. DSP routines convert the calibration data into a useful error table related to input signal state and slope. Execution of error table building routines is typically reserved for large workstations with large amounts of RAM, in order to manipulate the large matrices.

2.3.3 Compensation Formulation

Possible mathematical formulations for building an error table are given in [3, 8]. The principle behind collecting state and slope data for PPC calibration requires selecting an optimum number of calibration signals to populate the phase plane. Figure 2.7 illustrates a phase plane where normalized input signal slope is plotted against ADC state for a sinusoidal test signal. The result is an ellipse covering a range of states and slopes, also known as a trajectory. This figure shows a single trajectory for a 12-bit ADC. Selecting several amplitudes and several frequencies will tend to populate the phase plane with trajectories that intersect each other and create a well behaved solution. Signals must be chosen so that input signal harmonics do not alias to the same frequency bin. If harmonics are allowed to combine then the error terms may not accurately reflect the true error of the converter.

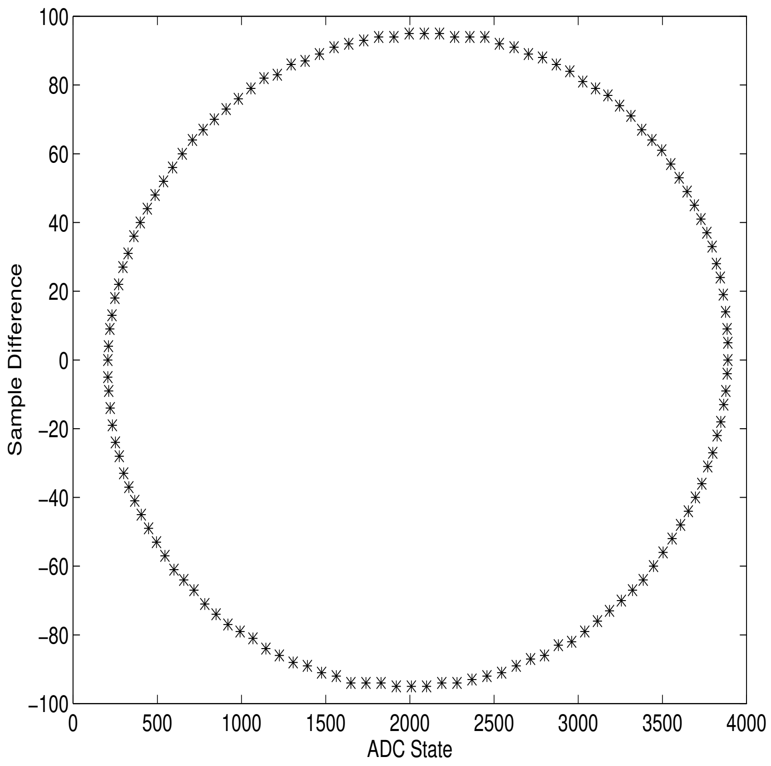


Figure 2.7: Calibration Trajectory in Phase Plane

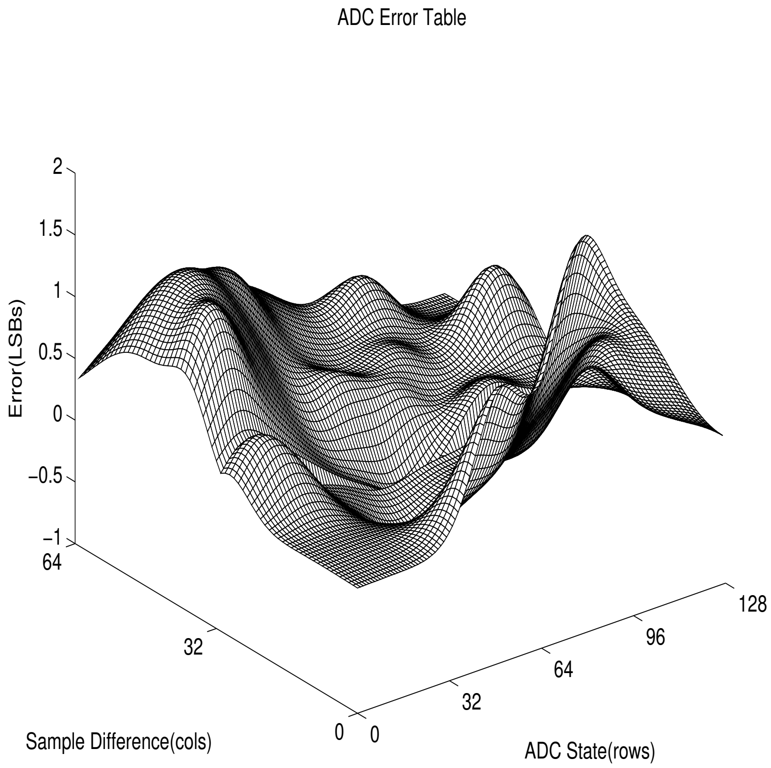


Figure 2.8: Error Table

In practice, ten frequencies and five amplitudes are sufficient to cover a majority of state and slope combinations. Post sampling DSP is performed to obtain the slope values from the sample arrays. Error table behavior or smoothness is highly dependent on ADC architecture. An example error table output from the above procedure might appear as shown in Figure 2.8. This table shows a plot of ADC error over 128 rows and 64 columns. Putting error tables into hardware for ADCs to be compensated is the focus of this work. The following section illustrates the design requirements set forth by the author and thesis advisor.

2.4 Compensator Hardware and Design Requirements

Factors influencing compensator design include ADC resolution, bandwidth, resolution of the error value, and hardware availability. The ADC resolution gives the minimum number of memory cells required to represent all states. For example, a 12-bit ADC requires 4096 cells for each state represented in the error table. Converter bandwidth determines the range of possible signal slopes, and dictates the clock frequencies for the ADC and compensator hardware. The error value resolution refers to the width of the memory cell. For compensators implemented on a workstation, typically double-precision 64-bit resolution. For hardware implementations, a more realistic approach is required. Realistic requirements for output sample resolution should dictate compensator memory requirements. Chapter 3 describes one possible compensator implementation based on the following requirements:

- The compensator should be configurable for an arbitrary error table, subject to memory size limitations. An error table size is application dependent with higher resolution tables requiring more memory than low resolution applications. Higher resolution error tables would be required for ADCs that require a unique error term for each ADC output and slope combination. Alternatively, a lower resolution application could apply an error term to a range of ADC outputs and slopes.

- A method for collecting uncompensated ADC samples must be provided. This will be used to collect baseline samples for later comparison with compensated samples.

- ADC state and slope inputs should be configurable from four to sixteen bits.

- Operator must be able to vary the number of state and slope bits for indexing the error table. It is desirable to retain the ability to reconfigure the error table address lines based on the error table characteristics. For example, it may be necessary to apply greater indexing resolution to the error table along a state or slope axis.

- Compensation must be real-time and operate in the 20 MHz region. This requirement was selected based on the need to minimize design complexity. A 20-MHz TTL solution simplifies the design compared with a higher speed logic family like emitter-coupled-logic(ECL).

- Compensator output resolution must exceed the ADC resolution. Greater resolution is required to represent the corrected samples. It is the price to pay for improving the ADC linearity.

- Must have a flexible bus interface for computer control of error table downloading and memory monitoring. This requirement permits alternative interfacing.

- The output must be designed for connectivity to the HP16500B Logic Analyzer. Each output must have an adjacent ground return for logic analyzer probes.

CHAPTER 3

Hardware Implementation

This chapter illustrates compensator circuitry and modes of operation. A functional block diagram is used to give the reader a big picture of the circuitry, prior to presenting detailed circuitry descriptions.

3.1 Overview

Figure 3.1 gives the compensator block diagram. The system is based on a 64k x 16-bit 9 ns synchronous static RAM memory. The 64k memory limit implies only 16 address lines can be used to look-up ADC error values. To do this, address multiplexing and user configurable selection switches are provided. These switches determine which ADC state or difference bits map to the RAM address inputs.

The hardware has two modes of operation, program and compensate. In program mode the RAM cells are stuffed with error table values, while compensate mode places the RAM in a read mode. Mode selection is accomplished by the PROG control line from the external interface. A PROG-high signal places the RAM in write mode, receiving address and data from the external interface. Alternatively, a PROG-low signal places the RAM in read mode allowing ADC state and difference inputs to address the RAM. Error tables are programmed by downloading preformatted error data as a serial stream of 16-bit words. These words are applied through the external interface error transceivers to the RAM data inputs over the error bus. Program addresses associated with the error data propagate to the RAM address inputs through the external interface buffer and address MUX(AMUX). PCLK is selected for RAM timing by PROG-high. The rising edge of PCLK strobes error data into memory or clocks addressed memory contents onto the error bus. Once error table loading is completed, control signals are configured for compensate mode.

Compensate mode is established at power-up or following error table loading by placing PROG in logic low state. RAM address lines are allocated to state or difference inputs selected by the configuration switches. State and difference inputs are applied through separate interfaces to state latch #1 and diff latch to the address MUX(AMUX) block. Addresses propagate through the AMUX to the RAM address inputs. RAM timing is selected by the PROGN-high signal at the clock select block. MCLK rising edge strobes the addressed memory cell contents onto the error bus. In parallel, the ADC state signal is pipeline delayed through state latch #2. State and error values are latched simultaneously at the state latch #3 and the error latch, respectively. The latch outputs propagate through a 16-bit arithmetic logic unit(ALU) which subtracts the error value from the state. The resultant 16-bit compensated state is applied to the output latch made available through the output interface.

This chapter is divided into two groups of circuit specifications, input/output (I/O) interface and functional circuits. Within the I/O group, the ADC, difference generator, external control, and output interfaces are presented. Functional circuits include the memory address MUX, clock selection logic, RAM, and ALU. All devices used in this project come from the FAST TTL chip family manufactured by National Semiconductor, with the exception of the RAMs which are CMOS/TTL compatible devices from IC Works. Detailed schematics and timing are illustrated in Appendix A.

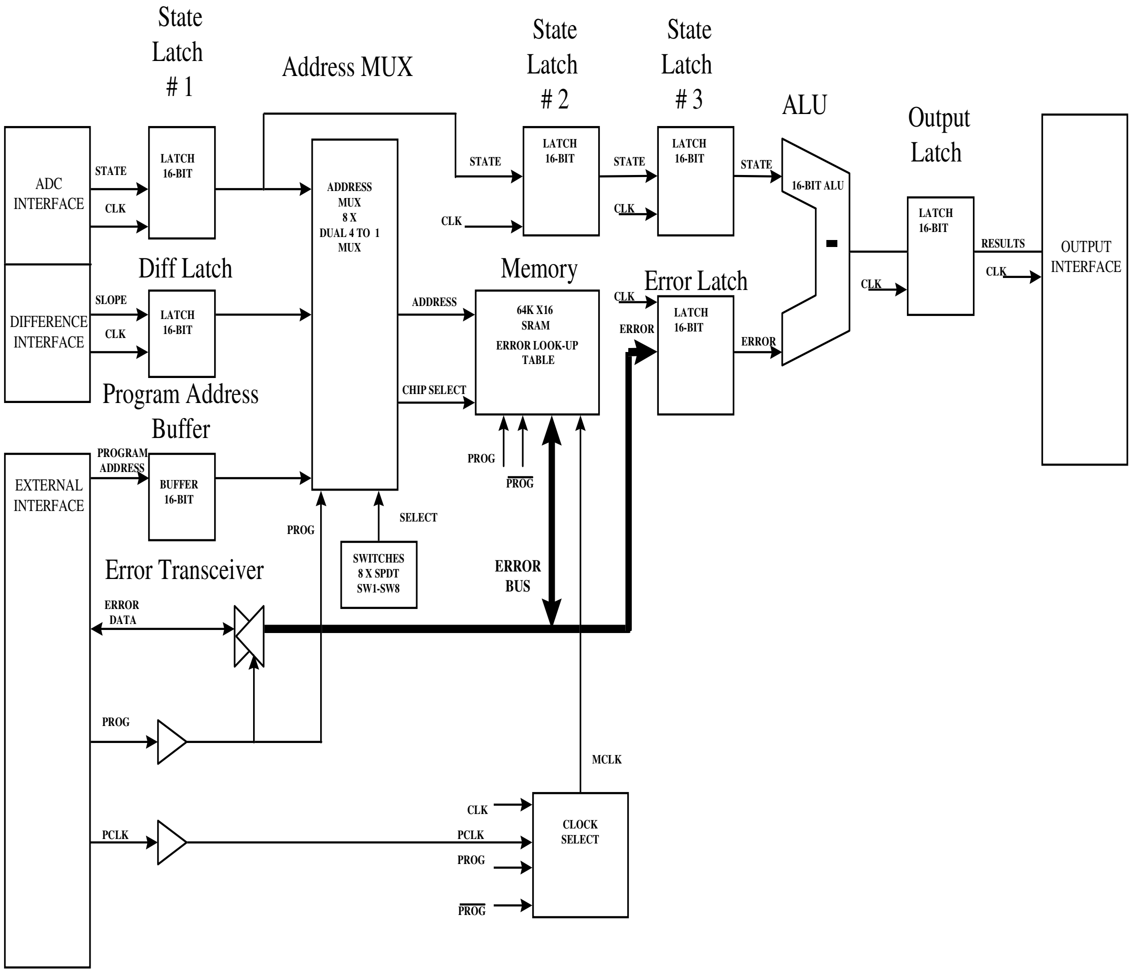


Figure 3.1: Compensator Block Diagram

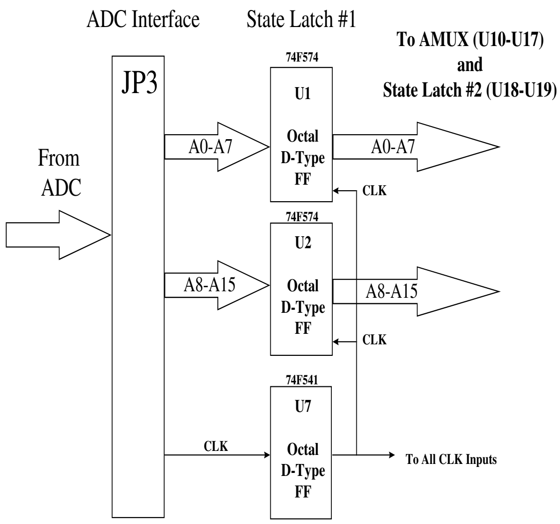


Figure 3.2: ADC Input Hardware

3.2 I/O Interface Circuits

3.2.1 ADC Interface Specification

The ADC interface consists of two 74F574 Octal D-Type Flip-Flops, U1 and U2, a 40 pin header-connector, JP3, and one 74F541 Octal Buffer, U7. U1 and U2 make up state latch #1 while U7 buffers the ADC *data ready* output used by the compensator for general purpose timing. As illustrated in Figure 3.2 ADC state inputs, A0-A15 are applied through JP3 to U1 and U2. JP3 CLK signal is buffered through U7 and latches the state inputs. The most significant bit(MSB) of the ADC should always be wired to bit A15. Any unused input lines should be tied to ground to prevent interference. U1 and U2 outputs are applied to AMUX inputs on U10-U17 devices and state latch #2 inputs on U18 and U19.

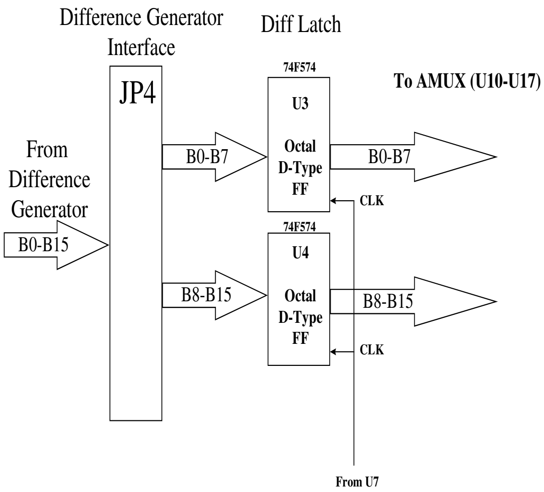


Figure 3.3: Difference Generator - Input Hardware

3.2.2 Difference Generator Interface Specification

The difference generator interface illustrated in Figure 3.3 consists of two 74F574 Octal D-Type Flip-Flops, U3 and U4, and a 40 pin header-connector, JP4. Difference generator inputs, B0-B15 shown in Figure 3.3 are applied to the diff latch made up of U3 and U4. CLK from U7 latches the 16-bit incoming difference data. The most significant bit(MSB) of the difference generator should always be wired to the MSB of the interface. Difference data is applied to JP4 in a two's-complement format. Any unused input lines should be tied to ground to prevent interference. Difference latch outputs are applied to one set of RAM AMUX inputs on U10-U17 devices.

3.2.3 External Interface Specification

Two data paths make up the external interface, program address and error data. Program address hardware is shown in Figure 3.4. It consists of a 40 pin header-connector, JP1 and two 74F541 Octal Buffers, U5 & U6. Error data hard-

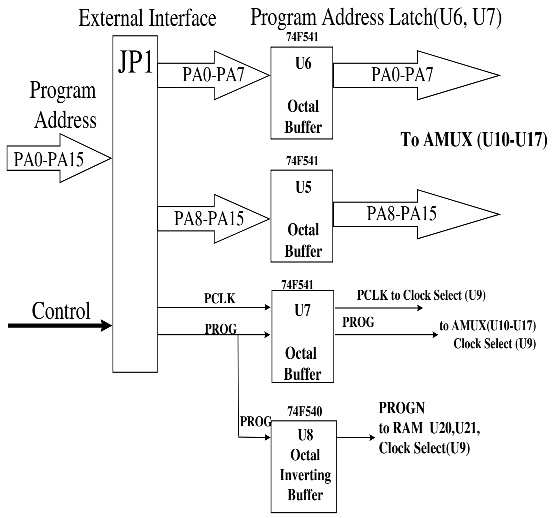


Figure 3.4: External Interface - Program Address Input Hardware

ware is shown in Figure 3.5. It utilizes a 40 pin header-connector, JP2 and two 74F245 Octal Bidirectional Transceivers, U20 & U21. This hardware permits two modes of operation, RAM program, and compensate mode.

In RAM program mode, an error table is downloaded into the 64k RAM by an external controller interfaced through the JP1 and JP2 shown in Figures 3.4 and 3.5. The controller could be a personal computer or IEEE-488 device. To select this mode the controller drives PROG-high and applies program address, PA0-PA15 to octal buffers U5 and U6. The buffered outputs propagate through the AMUX to RAM address inputs. The controller also applies error data to bidirectional transceivers U20 and U21. Transceiver outputs are enabled by PROGN-low with data direction hardwired to VCC. This places error data on the error bus and the inputs of the RAM devices. With PROG-high, program clock(PCLK) propagates through clock selection logic and becomes MCLK. Values placed on the error bus are stored in the RAM on the rising edge of MCLK.

Compensate mode is selected by the PROG-low signal. This is the desired

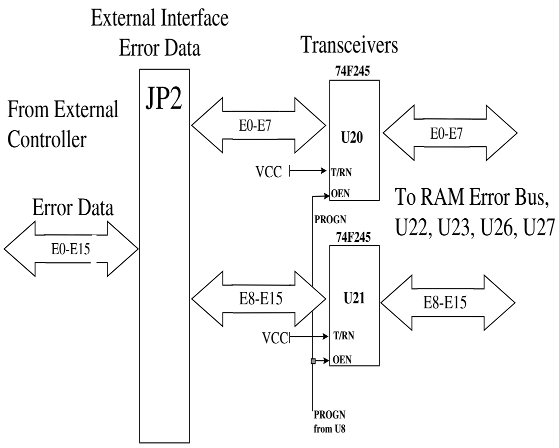


Figure 3.5: External Interface - Error Data Input Hardware

power-up state. To ensure a compensate mode power-up condition, the external interface must be preconditioned by an external controller prior to applying power. This ensures RAM and transceiver error bus contention is avoided. The conditioning disables the external interface inputs by maintaining PROG-low. In this mode, RAM timing is controlled by CLK. The bidirectional transceiver outputs are disabled by PROGN-low.

3.2.4 Output Hardware

Two 74F574 Octal D-Type Flip-Flops, U33 & U34, and a 60 pin header-connector make up the hardware required to deliver the compensated samples as shown in Figure 3.6. The output latch holds ALU results on CLK rising edge. The 16-bit latch output and CLK signal is available at JP5 as a twos complement value. JP5 odd-numbered pins carry the data. JP5 even pins adjacent to the data pins are grounded to provide a return path for twisted pair ribbon cables or logic analyzer probe return paths.

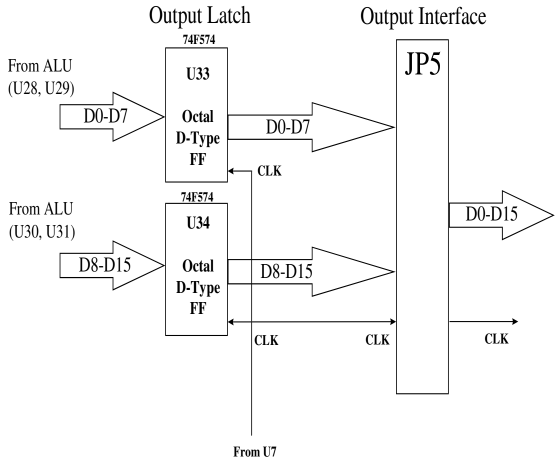


Figure 3.6: Output Interface - Compensator Data Output Hardware

3.3 Functional Circuit Specification

Circuit operation encompasses memory addressing, memory operation, ALU operation, and clock selection.

3.3.1 Memory Address Mux

3.3.1.1 Overview

The purpose of the address MUX is to select the desired address source based on mode of operation and switch configurations. Sources include ADC state, ADC difference, and program address. For program mode(PROG-high), program address is selected disabling the effect of configuration switches. Compensate mode requires assignment of AMUX inputs to state and difference inputs based on switch configuration settings. Figure 3.7 illustrates AMUX data flow and control. Inputs include the 16 state, 16 difference, 16 program address, eight configuration bits, and the PROG mode control line. AMUX outputs include 15 address bits, AD0-

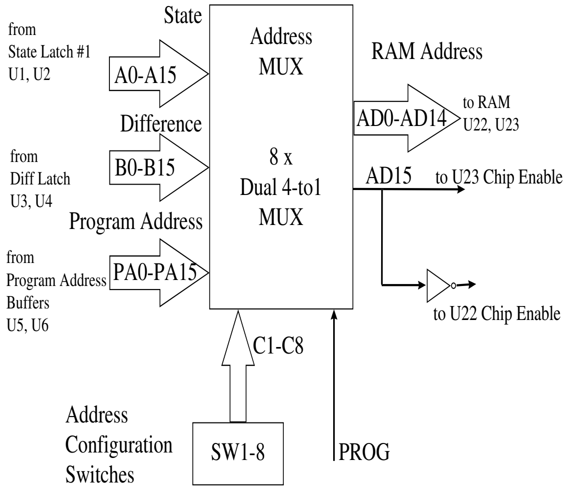


Figure 3.7: Address MUX Block Diagram

AD14, and one chip RAM enable. The compensator utilizes two 32k RAM devices to make one 64k RAM. AD15 selects the upper 32K of RAM when driving U23 chip enable low. A low complemented AD15(AD15N) enables RAM chip U22 selecting the lower 32K of memory.

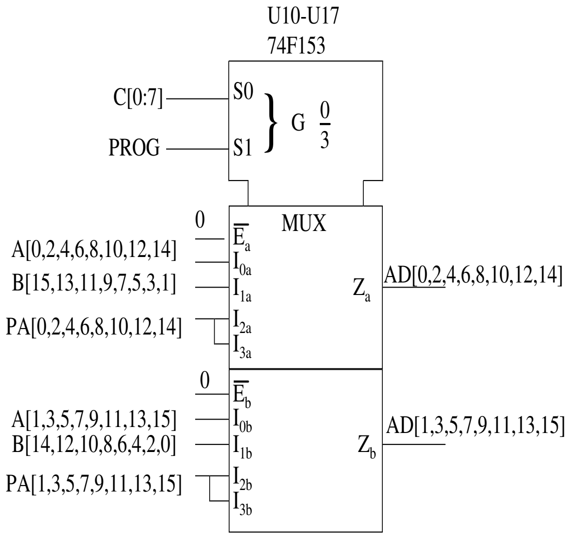


Figure 3.8: Address MUX

3.3.1.2 Hardware

Address MUX hardware includes eight 74F153 Dual 4-to-1 Mux devices, U10-U17, eight single-pole double-throw switches, SW1-SW8, and one 74F540 Octal Inverting Buffer, U8. Figure 3.8 illustrates how inputs are applied to MUX devices. ADC state inputs are applied in two groups to the I_{0a} and I_{0b} inputs. Similarly, difference estimate inputs are applied to I_{1a} and I_{1b}. Program addresses are applied to the remaining two sets of inputs, I_{2a}, I_{2b}, I_{3a}, and I_{3b}. AD0-AD14 AMUX outputs are applied to RAM address lines on U22 & U23. AD15 selects which RAM is enabled.

Recall that selectable addressing is a design requirement. Selection of the state and difference bits requires a PROG-low signal and some combination of configuration bits, C0-C7, selected at switches SW1-SW8. Configuration bits select state or difference bits in pairs. Note that not all state and difference bit combinations are possible due to the limit of the MUX select circuitry. Each configuration line selects the source for two RAM address lines. For example, C0-high selects difference bits B15 and B14 for RAM address AD0 and AD1, respectively. Similarly, C0-low selects state bits A0 and A1 for AD0 and AD1. Note the mapping configuration of the RAM address bits. RAM address line AD0 can be assigned to the state LSB(A0), difference MSB(B15), or program address LSB(PA0). Table 3.1 illustrates address selection options. The top row indicates RAM address inputs. The middle row gives address line options which are selected by configuration signals, C0-C7 given on the bottom row. Configuration signals are associated with configuration switches, SW1-SW8 which are manually selected by the operator.

LSB ← RAM ADDRESS WORD → MSB/CE

C-State	AD0/AD1	AD2/AD3	AD4/AD5	AD6/AD7	AD8/AD9	AD10/AD11	AD12/AD13	AD14/AD15
Low High	A0/A1 or B15/B14	A2/A3 or B13/B12	A4/A5 or B11/B10	A6/A7 or B9/B8	A8/A9 or B7/B6	A10/A11 or B5/B4	A12/A13 or B3/B2	A14/A15 or B1/B0
	C0	C1	C2	C3	C4	C5	C6	C7

Table 3.1: RAM Address Options

Table 3.2 shows configuration switch positions for selecting the desired state and difference bits for RAM addressing. Note that the number of states or differences could be as large as the value given in the table or smaller tables could be used. For example, a 128 state by 64 differences requires 7 state bits and 6 difference bits to address the error table. RAM address bits AD0-AD5 are assigned to the most significant difference inputs B15-B10, respectively. Bits AD15-AD8 are assigned to the state most significant bits A15-A8. This leaves two address lines, AD6 & AD7, to be arbitrarily assigned to either state or difference address inputs, A6 & A7 or B9 & B8. In this case, each error table value would be stored in four RAM locations to ensure that no matter which state or difference bits are mapped to the arbitrary address lines the correct error value will be retrieved. Note that the ADC state bits are wired to the Compensator state MSBs. In this example, only the eight state MSBs are used. Also, the six difference MSBs will be used.

It is important to note that the difference input may require fewer than sixteen bits to represent a full range of differences. Therefore, it is advisable to determine the maximum difference input and utilize only the bits containing actual difference information. In the case where ± 200 bits represents the full range of differences, 9 bits would be sufficient. These bits should be wired to the difference MSBs, B15-B7.

Configuration Switch Setting									
C0	0	0	0	0	0	0	0	0	1
C1	0	0	0	0	0	0	0	1	1
C2	0	0	0	0	0	0	1	1	1
C3	0	0	0	0	0	1	1	1	1
C4	0	0	0	0	1	1	1	1	1
C5	0	0	0	1	1	1	1	1	1
C6	0	0	1	1	1	1	1	1	1
C7	0	1	1	1	1	1	1	1	1
RAM Address Bit	RAM Address Assignment								
AD00	B15	B15	B15	B15	B15	B15	B15	B15	A0
AD01	B14	B14	B14	B14	B14	B14	B14	B14	A1
AD02	B13	B13	B13	B13	B13	B13	B13	A2	A2
AD03	B12	B12	B12	B12	B12	B12	B12	A3	A3
AD04	B11	B11	B11	B11	B11	B11	A4	A4	A4
AD05	B10	B10	B10	B10	B10	B10	A5	A5	A5
AD06	B9	B9	B9	B9	B9	A6	A6	A6	A6
AD07	B8	B8	B8	B8	B8	A7	A7	A7	A7
AD08	B7	B7	B7	B7	A8	A8	A8	A8	A8
AD09	B6	B6	B6	B6	A9	A9	A9	A9	A9
AD10	B5	B5	B5	A10	A10	A10	A10	A10	A10
AD11	B4	B4	B4	A11	A11	A11	A11	A11	A11
AD12	B3	B3	A12	A12	A12	A12	A12	A12	A12
AD13	B2	B2	A13	A13	A13	A13	A13	A13	A13
AD14	B1	A14	A14	A14	A14	A14	A14	A14	A14
AD15	B0	A15	A15	A15	A15	A15	A15	A15	A15
Number of Possible States									
States	0	4	16	64	256	1024	4096	16384	65536
Number of Possible Slopes									
Slopes	65536	16384	4096	1024	256	64	16	4	0

Table 3.2: RAM Address Selection Truth Table

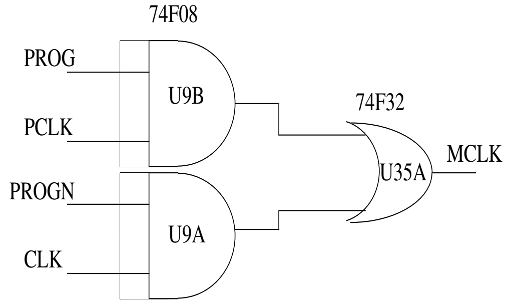


Figure 3.9: Clock Selection Logic

3.3.2 Clock Selection

Clock selection logic shown in Figure 3.9 consists of one 74F08 Quad 2-Input AND Gate and one 74F32 Quad 2-Input OR Gate. This logic selects which timing source will control RAM timing. In program mode(PROG-high) PCLK controls read & write timing of the RAM. In compensate mode(PROGN-high) CLK controls read timing.

3.3.3 IDT W73B586B RAM Configuration & Operation

A block diagram of the two IDT W73B586B RAMs used for compensator error table storage and look-up is shown in Figure 3.10. The 32k x 18 synchronous RAM devices are used to create a 64k x 16 error table vector space. The two RAM data LSBs are not used in this application due to the limitation of the 16-bit ALU. Chip selection is accomplished using AMUX output AD15 for U23 and AD15N(U8) for U22. There are only two possible RAM operations, read and write. Program mode is the only mode that allows write operations. Write operations

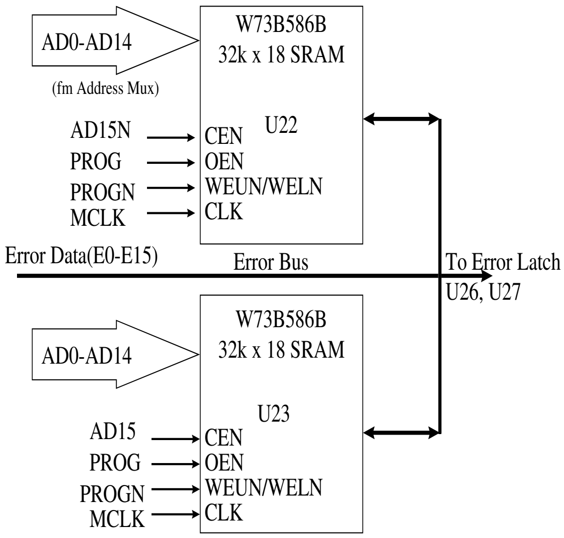


Figure 3.10: RAM Configuration Block Diagram

require disabling RAM outputs by applying PROG-high to OEN and enabling WEUN/WELN by bringing PROGN-low. Input data driven by transceivers onto the error bus is applied to the RAM data inputs, D2-D17. MCLK timing controlled by the external interface latches error data on the rising edge of MCLK.

Only compensate mode allows reading of RAM contents onto the error bus. PROG-low enables compensate mode reading of the RAM under CLK timing control. Addressed memory cell contents is strobed on the rising edge of CLK. Output data is valid 9ns later, and applied to the ALU error latch inputs, U26 & U27, and tri-stated transceiver outputs, U20 & U21. Currently, bidirectional transceiver operation is not enabled. Truth Table 3.3, summarizes the active states of U22 and U23.

3.3.4 16-bit ALU

The 16-bit ALU consists of 5 devices, U28-U32. U28-U30 are 74F381 4-Bit ALUs with Carry Look-Ahead(CLA) outputs. U31 is a 74F382 4-bit ALU

Compensator Mode	RAM Operation	MCLK	AD15 CEN
Program	U22-Write	$\uparrow$	H
Program	U23-Write	$\uparrow$	L
Compensate	U22-Read	$\uparrow$	H
Compensate	U23-Read	$\uparrow$	L

Table 3.3: RAM Control Truth Table

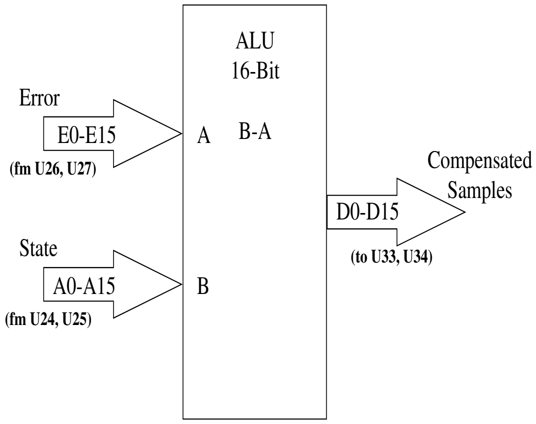


Figure 3.11: Arithmetic Logic Unit Block Diagram

with no carry look ahead and U32 is a 74F182 Carry Look-Ahead device. The four 4-bit devices and the CLA combine to form a 21.4ns subtractor circuit. The configuration shown as a block diagram in Figure 3.11 is detailed in [10, 11]. The ALU subtracts the error term applied by error latches, U26 & U27, from the ADC state latched by state latch #3, U24 & U25. Note U24 & U25 state inputs driven by state latch #2, is delayed by one clock cycle. This pipeline delay is required to wait for RAM access time. The ALU output is applied to output latches, U33 & U34 and available at JP5 along with the CLK signal. Note CLK rising edge leads ALU results by approximately 5ns due to latch propagation delay.

The compensator circuits require additional supporting hardware in order to interface to the ADC Testbed. The following chapter illustrates the remaining hardware required for interfacing an ADC to the compensator through a difference generator circuit, as well.

CHAPTER 4

ADC Compensator Application

This chapter illustrates the application of the ADC Compensator to an off-the-shelf ADC. It presents the design of additional hardware required to support the application of an Analog Devices AD9027 ADC. To complete this chapter, compensator setup and operating procedures are given.

4.1 Hardware Overview

The first requirement is to select a suitable ADC to compensate. The Analog Devices AD9027 12-bit monolithic ADC was selected based on previous compensation success in software. Additionally, it is well suited to the operating frequency of the compensator. It will operate at 20.48 million-samples-per-second (MSPS) over an input range of two volts peak-to-peak.

Several additional circuits required to support the ADC interface to the compensator include a Difference Generator, ECL-to-TTL Translator, and two connector interface boards. A PC Interface circuit will be used to interface a personal computer to the compensator external interface. Each of these circuits will be discussed following a review of how these components interconnect.

The block diagram shown in Figure 4.1 gives two input data paths applied to ADC compensator inputs. Ouput data is shown at Compensator JP5. Data path #1 beginning on the left of the figure shows two 12-bit Analog Devices AD9027 ADCs. The ADC testbed drives ADC inputs not shown in this figure. ADC outputs are wired through a wiring interface board (IXBD) and an ECL-to-TTL Translator board to the Difference Generator inputs. IXBDs provide the required interconnect between the two boards. The ECL-to-TTL Translator Board converts ADC outputs from differential ECL voltage levels to single ended TTL values required by the Difference Generator. The Dual-Sample Difference Generator calculates the difference between ADC #1 and ADC #2 previously illustrated in Chapter 2. Difference Generator outputs, state & difference are applied to Compensator inputs.

A second input data path provides a means for downloading error tables into compensator RAM. It uses a PC to control the transfer of error tables from the ADC Testbed workstation to the compensator. This is accomplished by connecting the PC parallel-port to PC Interface board. PC Interface board outputs are connected to Compensator External Interface inputs. The PC Inteface board organizes the RAM address and error value data prior to transferring it into RAM. Supporting circuits will be detailed beginning with data path #1 followed by data path #2.

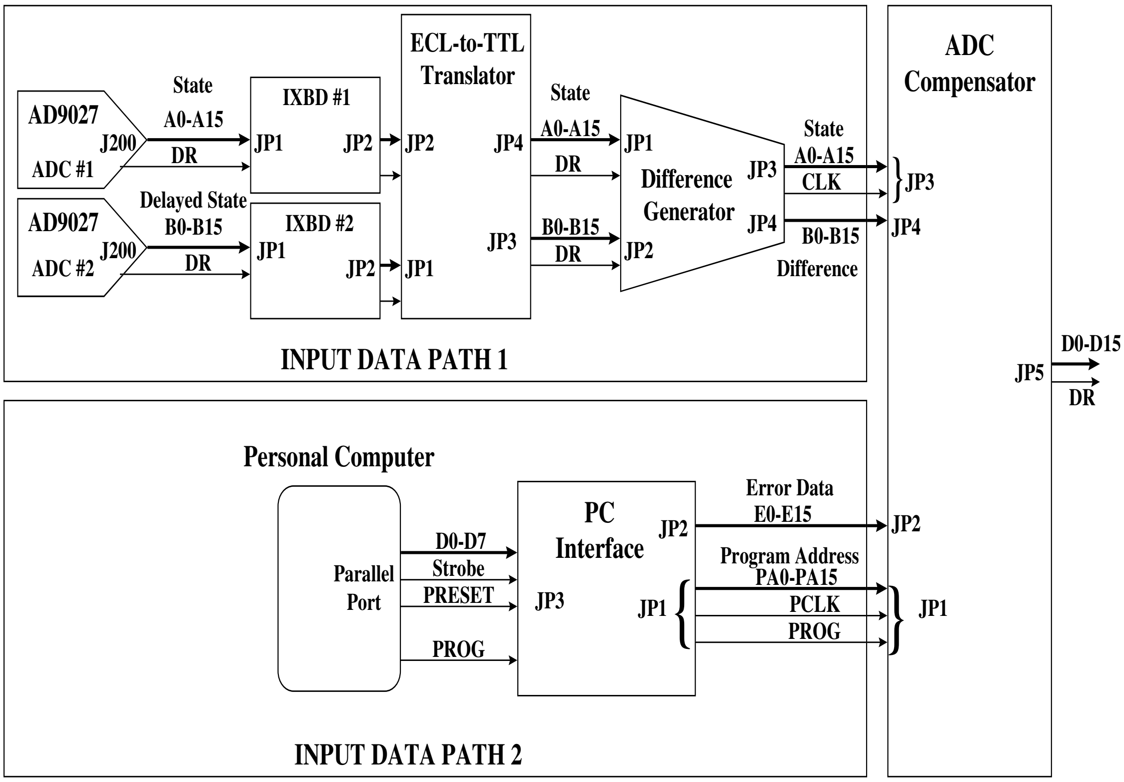


Figure 4.1: Application Support Circuitry Block Diagram

4.2 Data Path #1 Circuits

4.2.1 Analog Devices AD9027 12-bit ADC

The Analog Devices AD9027 12-bit, 31 MSPS Monolithic ADC is mounted on a AD9026/AD9027 Evaluation Board. Figure 4.2 shows the ADC input and output signal distribution. Two testbed signal generators supply the sinusoidal input signals. A single analog input is driven by a Hewlett Packard HP8645A Agile Signal Generator through a set of precision low-pass filters, an Anzac THV50 power divider, and dither combining circuitry. At the divider, the signal is sent to each ADC. ADC #2 receives a delayed version of the input signal. The delayed signal is sampled by ADC #2 synchronously with the normal version seen at ADC #1. Optimum signal delay is determined to be 600-1000ps using a technique illustrated by Kennedy [7]. The implementation of the signal delay requires inserting a six inch piece of coaxial cable into the input line of ADC #2. The sinusoidal clock is supplied by a Hewlett Packard HP8642A Signal Generator through an Anzac THV50 power divider to the ECLEN SMA port on the AD9027 evaluation board.

Evaluation board outputs are differential emitter-coupled-logic (ECL) and must be converted to single ended TTL levels to be used by the difference generator. The ECL-to-TTL translator board performs the required level conversion and is interfaced to the ADCs through IXBD #1 and IXBD #2. An input requirement of the Difference Generator is that the ADC outputs must be wired to the Difference Generator most significant bits. Therefore the IXBDs align the ADC bits 0-11 with Difference Generator inputs 4-15.

A data-ready (DR) signal accompanies each 12-bit value for external latching purposes. ADC I/O details are given with IXBD I/O in Appendix F. AD9027 Evaluation Board details including timing are given by [1] and [2].

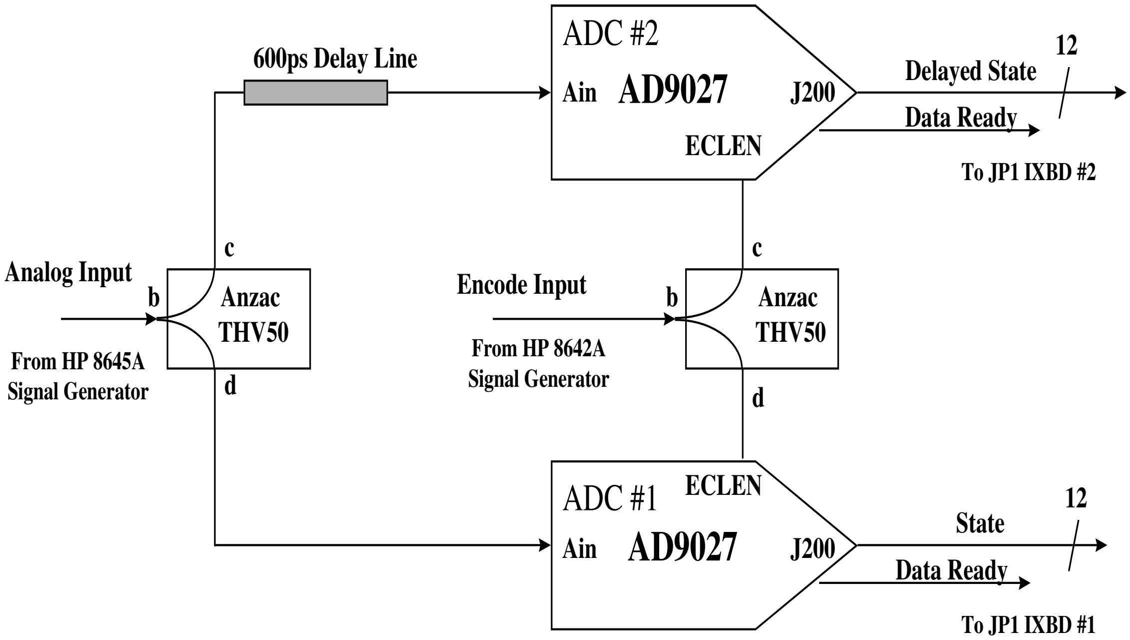


Figure 4.2: ADC I/O Signal Distribution

4.2.2 ECL-to-TTL Translator

The ECL-to-TTL translator block diagram is shown in Figure 4.3. Six F100325 Hex Low-Power ECL-to-TTL translator devices, U1-U6, provide the necessary level conversion for single ended TTL. A 74F541 Octal Buffer, U7, is used to delay the data ready signal by 2-5ns for Difference Generator latch setup time purposes. Interfacing is accomplished using four 40-pin header-connectors, JP1-JP4. JP2 and JP1 accept ADC #1 & #2 outputs from IXBD #1 & #2, respectively. The 50 Ohm differential inputs are terminated into 100 Ohm resistors at U1-U6 inputs. Schematic details and physical layout are given in Appendix C. ADC outputs pass through the ECL-to-TTL translator delayed by 2-5 nsec and are applied to the Difference Generator at JP3 and JP4.

4.2.3 Dual-Sample Difference Generator

Figure 4.4 shows a dual-sample difference generator block diagram, previously presented in Section 2.3.1.1. It consist of four sets of two 74F574 D-Type flip-flops, four 40-pin header-connectors, and a 16-bit ALU. A data ready(DR) rising edge causes STATE & Delayed STATE offset binary inputs to be latched by the difference generator U1-U4. Latch outputs propagate through ALU circuitry yielding a twos-complement difference result at the output, JP4. The ALU is configured to subtract ADC #2 from ADC #1 outputs. It consists of 5 devices, U5-U9. Devices U6,U8,U9 are 74F381 4-Bit ALUs with Carry Look-Ahead(CLA) outputs. U5 is a 74F382 4-bit ALU with no carry look ahead and U7 is a 74F182 Carry Look-Ahead device. The five devices combine to form a 21.4ns ALU used as a digital subtractor. ALU details are given by [10, 11]. The next DR rising edge latches the difference in U12 & U13 along with the associated STATE value at U10 & U11. State and associated difference are applied to Compensator JP3 & JP4 inputs along with DR. Schematic details, layout, timing, and power are given

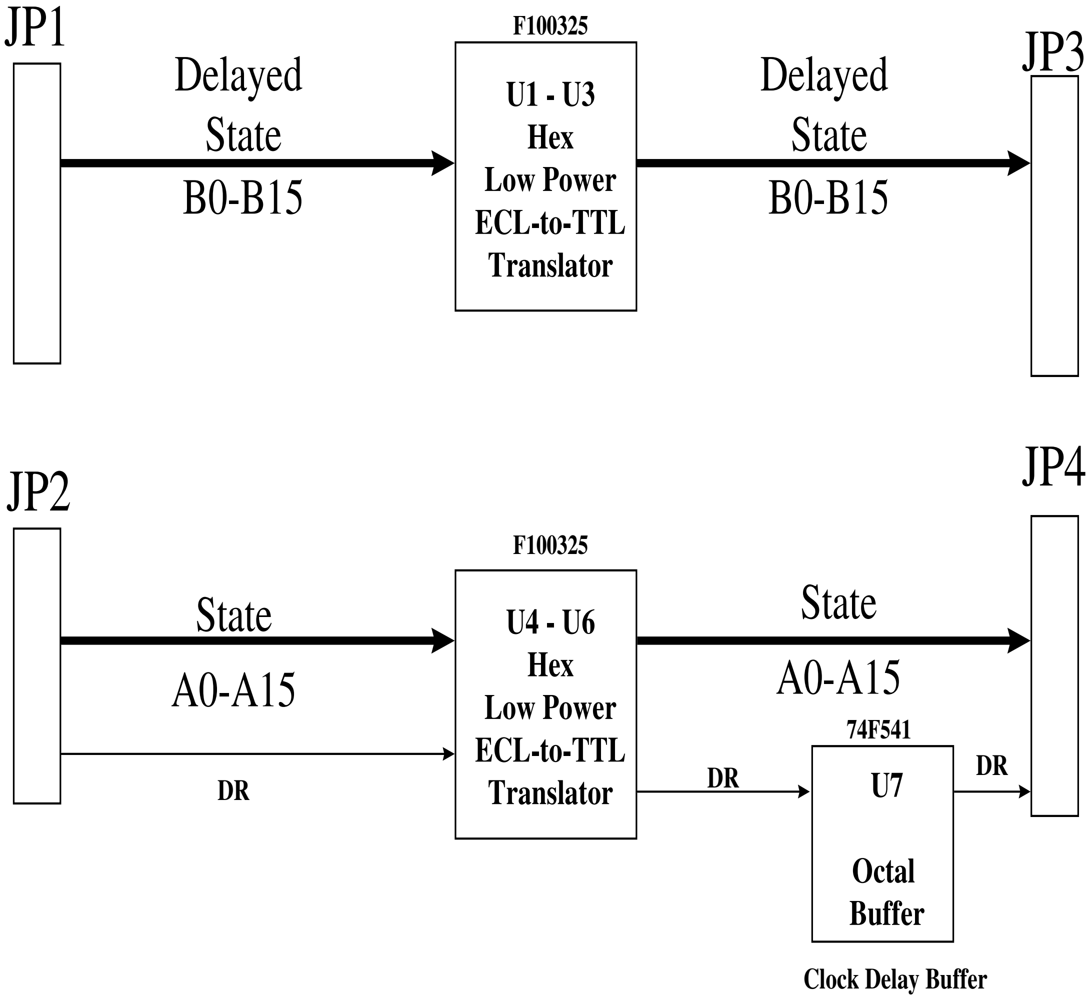


Figure 4.3: ECL-to-TTL Translator Block Diagram

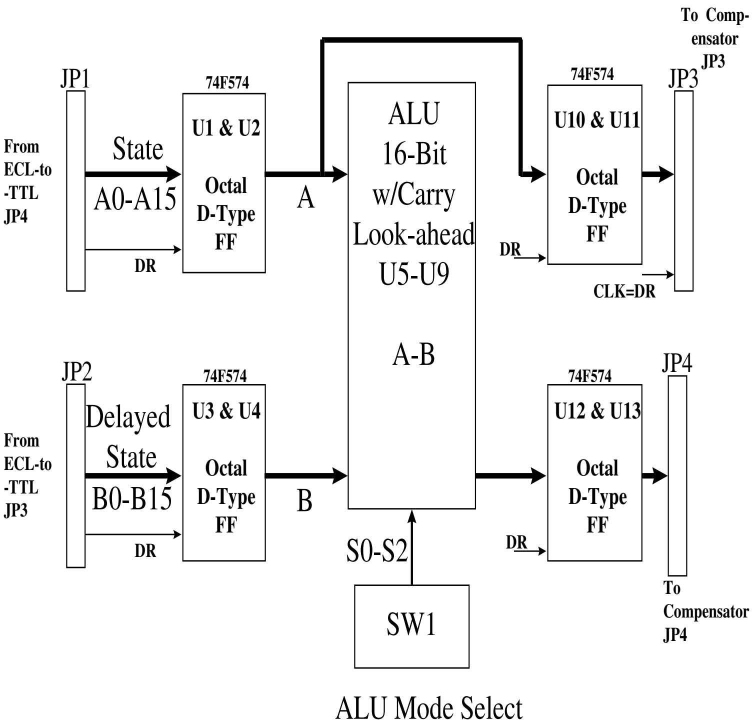


Figure 4.4: Difference Generator Block Diagram

in Appendix B.

This completes the discussion of the ADC interface circuitry. In summary, two ADCs are driven by the ADC Testbed. The converter to be compensated generates samples of a spectrally pure sinusoidal input signal while the second ADC samples a delayed input signal. The ECL-to-TTL Translator performs the logic level shifting and supplies the ADC outputs to the Difference Generator. The Difference Generator determines the difference between these two sample streams and supplies the ADC State and difference to the compensator. These inputs are used to address an error term in the compensator RAM which is subtracted from the State input. The result is a compensated sample output from the compensator. The remaining components perform the compensator RAM loading. This is accomplished by the PC Interface circuitry.

4.3 Input Data Path #2 Circuits

This section addresses the transfer of an error table to compensator RAM. First take a look at an actual error table. The table format is N-rows on the state axis by M-columns on the difference axis. Typically, N & M are on the order of 128 by 64, respectively. Note, 128 values are used to represent a possible 4096 states for this 12-bit converter. Similarly 64 difference values are used to represent thousands of possible slopes. This is permitted because PPC uses smooth functions to represent error terms. Smooth functions allow interpolation between indexes. For example, using 128 rows to represent 4096 states implies that each table row will represent 32 states. Similarly, each column may represent a range of differences. One important difference between state and slope representation is that the slope address can be positive or negative.

Error table values are represented by 16-bit numbers. The table is stored in an ASCII file created by ADC calibration routines. ADC calibration theory is beyond the scope of this work but a procedure will be included at the end of this chapter giving the steps required to generate an error table.

Given a 64k x 16 RAM for error table storage, the error table must be converted into a more suitable format to be loaded into RAM and used by the compensator hardware. This chapter illustrates the hardware and software routines necessary to move the error table from a table of 64-bit values into a 64kx16 RAM.

4.3.1 PC Interface

The PC interface consists of a personal computer(PC), a PC Interface circuit and two software routines. A networked PC provides the link between the ADC Testbed workstation and the compensator. The workstation generates the error-tables as a result of ADC calibration. Error tables are reformatted for downloading and transferred to the PC. The PC parallel-port is used as a unidirectional downlink to the compensator RAM using the PC Interface circuit. First, consider the error table formatting software.

A Matlab script running on the workstation loads the desired error table. Hardmap.m, given in Appendix E, performs the following operations on the table:

- Scale all values by a factor equivalent to the ADC hardware. For example, a 12-bit ADC is wired to the twelve MSBs of the Compensator. Therefore the error values must be shifted by four bits. Additionally, negative values are converted to two's complement to conform with ALU input requirements.

- Truncate all error values to conform to Compensator hardware. For the 12-bit case where the error term was shifted by four bits, the remaining fractional part of the error term is discarded. Thus, the four bits hold the fractional part of the error term.

- Generate a RAM address array ranging from 0-to-65535.

- Loop through each of the above RAM addresses and determine the state and difference index. Indexes point to the appropriate error table value to be associated with the address.

  - State index is based on error table size. For example, 128 state values would require 7-bits. Therefore the seven MSBs of the RAM address will determine the state index.
  - Similarly, difference index is determined by the number of differences represented. For a 64 difference table, 6 LSBs of the address are used. Additionally, the difference bits are flipped left to right in order to match the Address MUX hardware. The negative difference index values are converted to two's complement to match the values generated by the Difference Generator.

- State and difference indexes access the appropriate error table value which is written to the loadertable.mat file along with the associated address.
- The loadertable.mat file is transferred to the PC prior to downloading. A C-program called loader.exe running on the PC reads the loadertable file and orchestrates the control sequence required to stuff the RAM.

4.3.1.1 PC Interface Hardware

The hardware required for the downloading operation is shown in the block diagram Figure 4.5. The PC parallel-port is connected to the PC Interface board JP3. PC Interface outputs connect to the compensator external interface, JP1 & JP2. PC Interface JP1 and JP2 supply RAM address values and error values through the compensator JP1 & JP2 interfaces, respectively.

The PC Interface hardware consists of four 74F373 Octal Latches, a 5-bit serial shift register, and 12 buffers implemented using two Altera EP910D-40 EEPROM devices. The input connector JP3 is a 25-Pin D-Type connector. The PC Interface schematic, layout, and timing are illustrated in Appendix D.

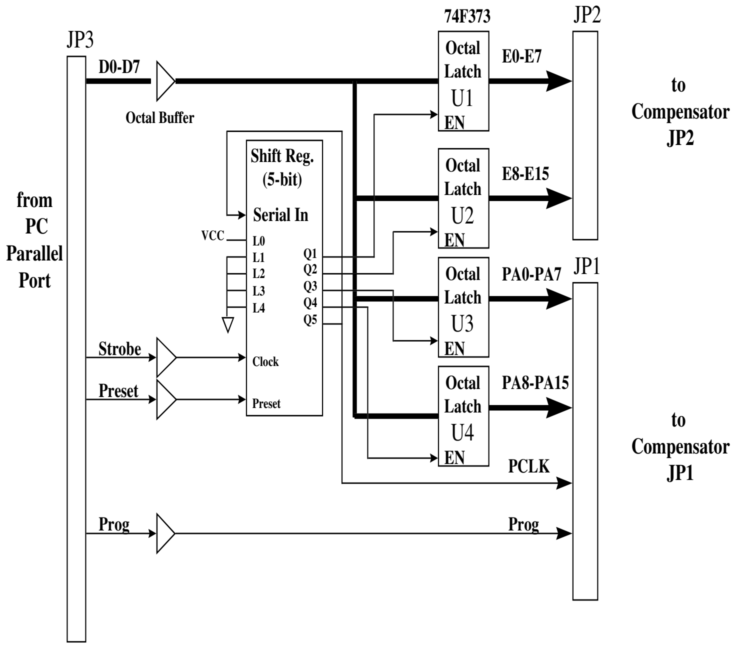


Figure 4.5: PC Interface Block Diagram

4.3.1.2 PC Interface Operation

The loader routine controls the PC parallel-port and PC Interface circuit for the purpose of transferring an error table into compensator RAM. Source code is given in Appendix E. The following discussion illustrates the steps performed by the loader routine.

The loader routine receives an ASCII file containing two arrays, RAM address and error values. RAM address consists of 16-bit decimal values ranging from 0 to 65,535. The second array holds the error values, also formatted as 16-bit decimal values.

The parallel-port is limited to an 8-bit data register and 8 control bits. Three control bits are used to transfer the error table, STROBE, PROG, and PRESET. Downloading the two 16-bit words, is accomplished by dividing each word into bytes. Each byte is downloaded through the parallel-port, reassembled by the PC interface circuit and latched into RAM.

- The first step is to place the compensator in program mode by applying a logic high to PC Interface PROG control line. This is accomplished by driving PC parallel-port control register bit C3 with a logic low. C3 output is complimented placing a high on PC Interface PROG input line. PROG-high signal propagates to the compensator and commands the following operations. First it tristates RAM outputs and enables external interface transceiver outputs. Second, it selects program address bits, PA0-PA15, at the AMUX. Lastly, it allows PC Interface PCLK to command RAM timing for writing error data to RAM cells.

- Next, the PC Interface board hardware is initialized by presetting the 5-bit shift register to binary 10000. This is accomplished by pulling control register bit C1(PRESET) to a high and toggling control register bit C0(STROBE).

The shift register selects a latch based on its 5 outputs. For example, a high on the Q1 output places the first latch in a transparent mode, or allows data on the input bus to propagate through the latch to its outputs. When the next STROBE occurs, the data on the latch input is held at the output until the latch is enabled again. For each STROBE received, the high output shifts to the next significant output.

- Each RAM address and error value is read from the error table file and parsed into four bytes.

- The least significant error value byte is written to the parallel-port data register. This byte is applied to all PC Interface board latch inputs but only the selected latch transmits the valid data. In this case, U1 outputs E0-E7 to JP2.

- STROBE is toggled to latch the byte and causing the latch enable high bit to shift to latch U2. The first valid byte of data, E0-E7 is applied to the RAM error bus through external interface transceiver, U20.

- Error value MSByte is loaded into the data register and latched in U2. The entire error word is now applied to the error bus. Next the appropriate address must be applied through the remaining two latches, U3 &U4.

- The third and fourth write cycles latch the RAM address LSByte and MS-Byte. Once the data and address applied to the RAM meet required setup times, the data is ready to be stored.

- The fifth and final STROBE places a high on shift register Q5 output, triggering two important events. First a PCLK signal is sent to the Compensator to latch the error values into RAM. Second the Q5 output is applied to the shift register serial input preparing the circuitry for the next address word and error value. This

- The loader routine loops through all error values and addresses. Normally, the entire RAM is initialized to some value.
- Once the RAM is stuffed, control of the RAM must be returned to the ADC. To do this PROG signal is pulled low by placing a high in control register bit C3. The Compensator is now in compensate mode and running.

The following section gives the procedures for ADC calibration and error table downloading.

4.4 ADC Calibration Procedure

Calibration requires a particular hardware setup and the execution of Matlab scripts to generate an ADC error table. Once the error table is generated, the operator must verify its effectiveness. The following procedure gives the reader steps to follow for connecting various hardware components and the sequence for software execution.

4.4.1 Hardware Setup

For connecting the hardware, refer to Figure 4.6 and Figure 4.2.

1. Connect HP16500B Logic Analyzer to Difference Generator outputs JP3 & JP4. This may require disconnecting the ribbon cables between the Difference Generator and Compensator.
2. Connect clock and analog input signal generators to the appropriate ADC inputs.

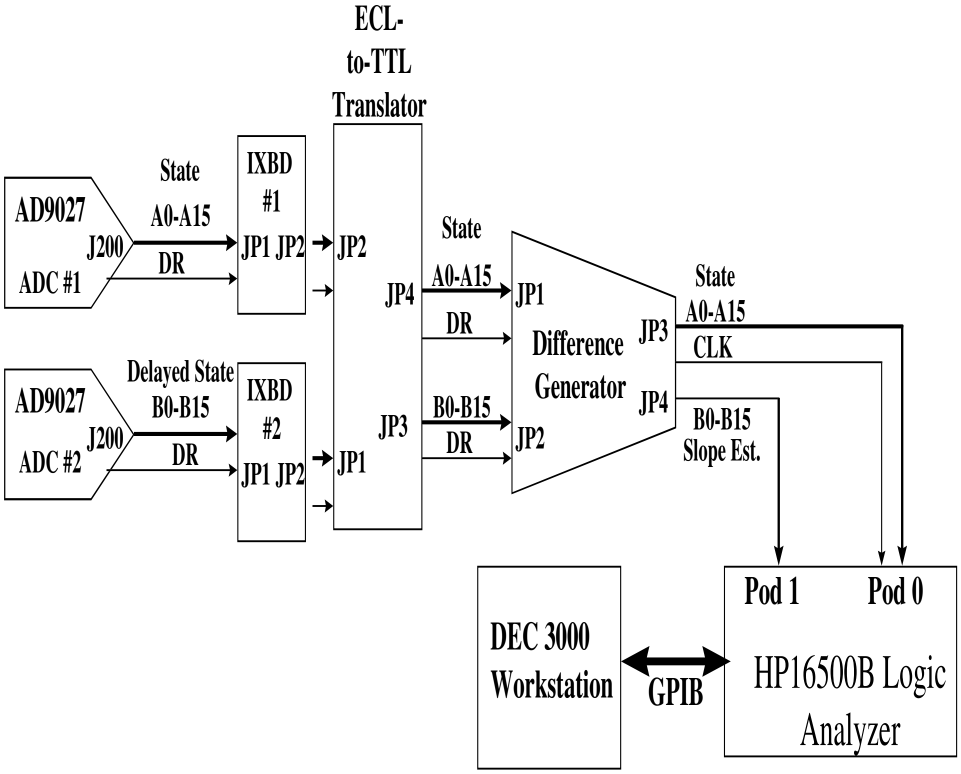


Figure 4.6: Calibration Hardware Setup Block Diagram

4.4.2 Software Execution

From DEC 3000 workstation:

1. Determine the operating frequency of the ADC, desired frequency band to compensate, and the input amplitude range to compensate. These values must be inserted into the adc_cal.m script.

2. The user is also required to determine error table size. Typical values are 128 states by 64 differences, or 7-by-6 bits, respectively.

3. Within Matlab: execute adc_cal.m script. This routine selects the appropriate default equipment setup for calibrating the AD9027. The user is prompted to change low-pass filters when required. The result is an ADC error table, called table.

4. To verify error table effectiveness execute Matlab script: comp.m. This routine collects a buffer of samples and generates harmonic distortion comparison plots showing compensated versus non-compensated samples. These plots are explained in Chapter 5.

4.5 ADC Compensation Procedure

4.5.1 Compensation Hardware Setup

1. ADC inputs are setup identically to the ADC calibration input setup.

2. Disconnect HP16500B Logic Analyzer from Difference Generator JP3, JP4.

3. Re-connect ribbon cables between Difference Generator and Compensator.

4. Connect HP16500B Logic Analyzer PODs 0 & 1 to Compensator JP5. Connect Compensator clock pin JP5-P39 to POD 0, CLK.

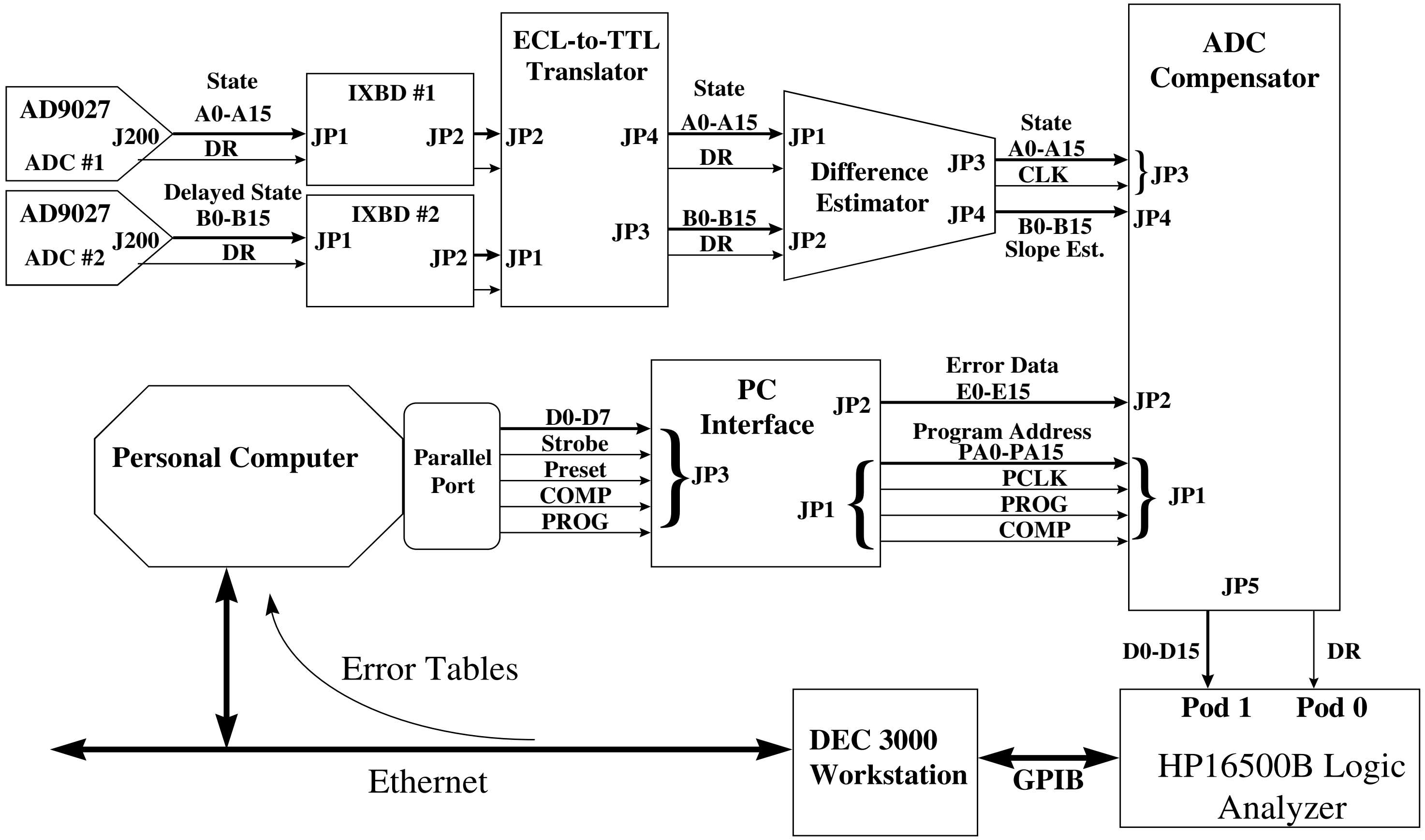


Figure 4.7: Compensation Hardware Setup Block Diagram

5. Connect ribbon cable between the personal computer parallel-port and the PC Interface Board JP3.

6. PC Interface board JP1 & JP2 should be connected to Compensator JP1 & JP2.

7. Compensator address configuration switches (SW1-SW8) must be configured according to the error table to be downloaded. For example, a 128-by-64 table requires SW1 through SW4 be closed, while SW6-SW8 is set to open. SW5 is a don't care.

8. If an error table is loaded into RAM, compensated samples will be output at JP5. If uncompensated samples are desired, download zeros.mat table. This table initializes all RAM cells to zeros. This applies zeros to one half of the ALU input. The ALU output becomes the ADC value applied to the compensator input.

4.5.2 Compensation Procedure

1. From Workstation within Matlab: Execute Hardware Mapping Script using *table* and *max-difference* as input arguments. syntax: hardtable(table, max-difference). The error table result will be a file called *hardtable.mat.*

2. From Personal Computer: FTP hardtable.mat to compensator directory.

3. Double click the loader icon from the power tools window. At prompt for error table file name, enter hardtable.mat. Reformatted error table will download through the PC Interface. When complete, PC beeps several times and displays a "Download Complete" Message.

4. At workstation, within Matlab: Select ADC Setup #4, i.e. select_adcsetup(4).

5. At workstation, within Matlab: Collect samples, syntax: state = fetch_cachesamples (number_samples). Typical number of samples is 16,384 samples.

This procedure will place a buffer of samples into Matlab. Use Matlab script plotspec.m to display a harmonic distortion plot.

CHAPTER 5

Application Results

This chapter presents the AD9027 application results. Dynamic range measurement will determine the effectiveness of both the software and hardware compensation techniques. To begin, uncompensated samples are collected and dynamic range is measured for a single input frequency and amplitude. Then software compensation is applied to the samples to generate baseline dynamic range improvement results. These results represent the best that the hardware compensator can accomplish. Finally, the dynamic range improvements using the hardware compensator are measured and compared with software results. This comparison shows how closely the hardware compensator resembles software methods used at the University of Maine.

5.1 Dynamic Range of Uncompensated Samples

Figure 5.1 shows a harmonic distortion plot of ADC samples collected from the AD9027. This figure was generated using Matlab's Fast-Fourier-Transform function. Frequency is plotted along the horizontal axis while amplitude is shown along the vertical axis in dB with respect to a fullscale sinewave. The fundamental and low-order harmonics are numbered. Dynamic range is defined as the amplitude difference between the fundamental and the largest harmonic. Designer's try to maintain distortion levels below $6n\ dBFS$ where n is the converter resolution in bits. For example, the AD9027 should show at least 72 dB of dynamic range across its full operating input and frequency range. The desired effect of compensation is to reduce harmonic distortion. In Figure 5.1 the largest harmonic is number two at -82 dBFS while the fundamental is about -3 dBFS, giving 79 dB of dynamic range. Figure 5.1 shows the harmonic distortion of the test signal at a frequency

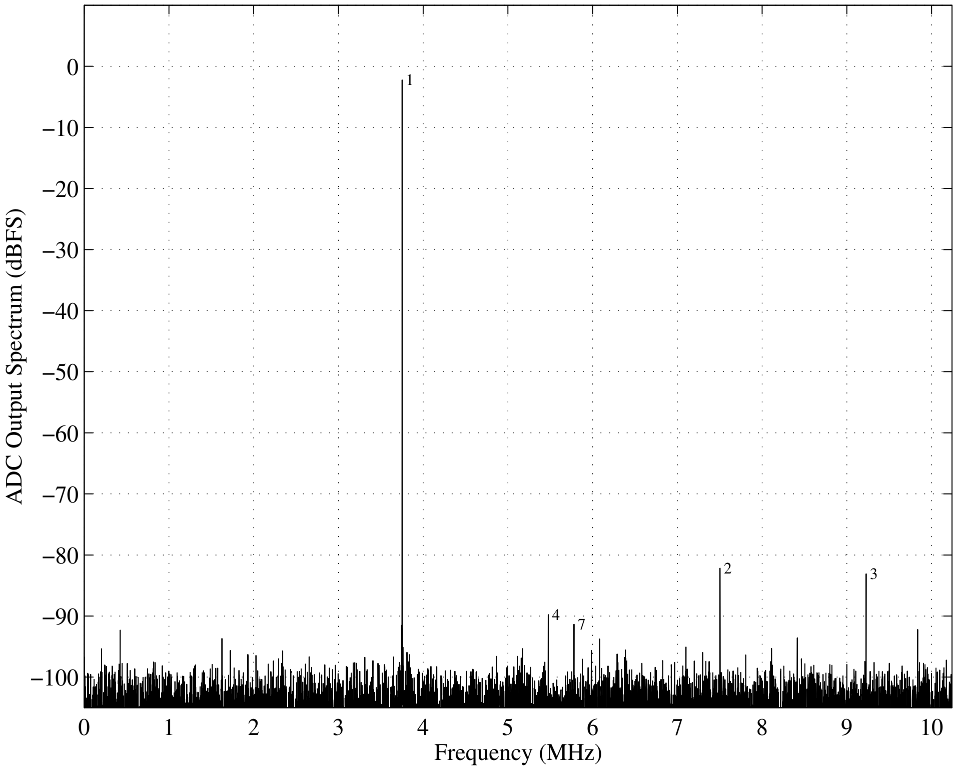


Figure 5.1: Harmonic Distortion Plot of Uncompensated Samples

of 3.751250 MHz and 90% fullscale input. The ADC is operated at 20.48 MSPS. The first 10 harmonics are labeled if greater than -95 dBFS. Harmonics 2-4, and 7 meet the criteria for labeling.

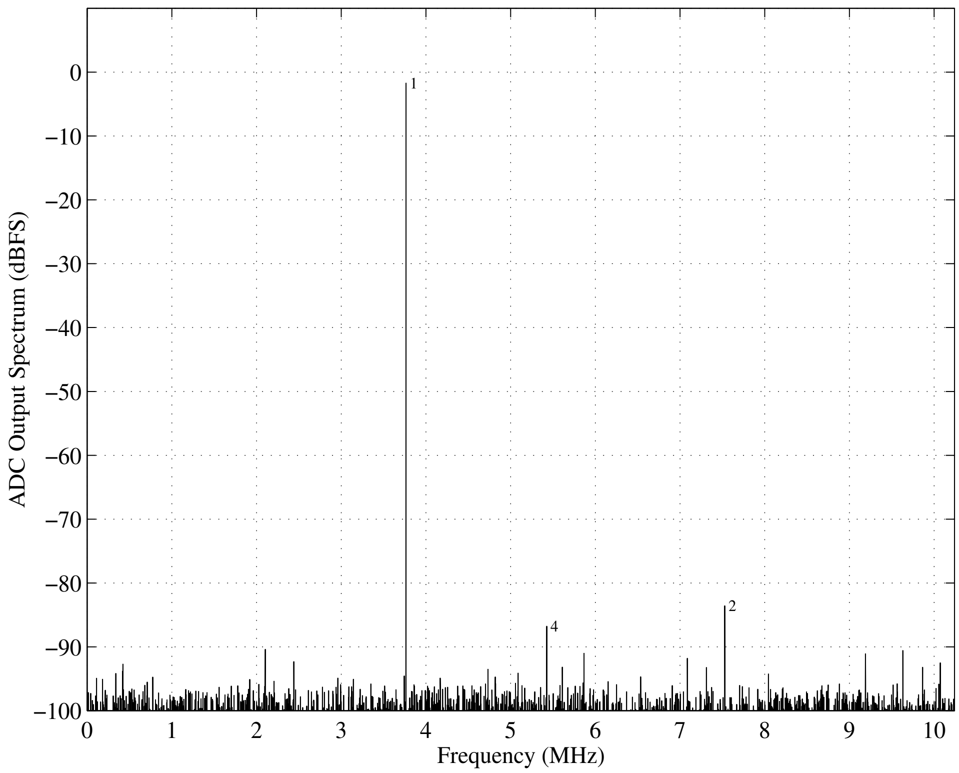


Figure 5.2: Harmonic Distortion of Software Compensated Result Samples

5.2 Baseline Compensation in Software

To determine improvements to dynamic range a buffer of samples is collected and compensated using traditional software methods. The results of software compensation show the dynamic range to be 82 dB, an improvement of 3 dB, for this signal. Note only two harmonics remain larger than -90 dBFS.

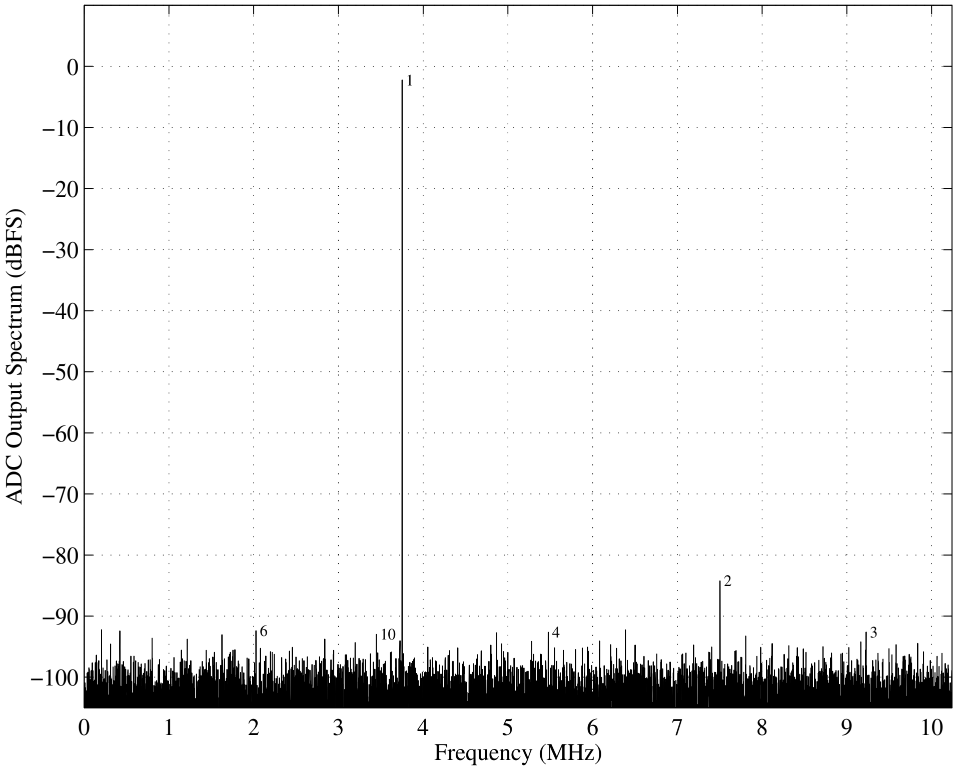


Figure 5.3: Harmonic Distortion of Hardware Compensated Samples

5.2.1 Hardware Compensation

By comparison with software compensation, similar results are shown in Figure 5.3 for the hardware compensator. Dynamic range results show 3 dB improvement and suppression of the largest harmonics similar to the software results.

5.3 Conclusions

This thesis has illustrated the design and implementation of a Phase-Plane Compensator circuit for Analog-to-Digital Converters. Setup and operating procedures were given based on current methods of ADC compensation using the University of Maine ADC testbed and software routines. Application of the hardware compensator showed how it could be used to compensate one off-the-shelf ADC. The results of this application show the design and implementation of the ADC compensator hardware to be sound. Additionally, ADC compensation in hardware shows similar results to software methods. One remarkable result is that software techniques operate on double precision error values while the hardware compensator is limited to 16-bit word lengths. This application shows that only 4 bits appended to the 12-bit sample appear to be as effective as the double-precision word used by the software compensation technique. This is interesting because ADC word length is a limiting factor for hardware applications.

Future work could investigate the optimum word length for ADC hardware compensators. Error checking should be incorporated into the error-table downloading routine. An alternate method for downloading error-tables using the IEEE-488 infrastructure of the ADC Testbed is also desired. Additional work could focus on expanding the compensator design to operate at faster data rates in the VHF region of the electromagnetic spectrum. And finally, the total project could be integrated into one or two programmable logic arrays.

References

[1] Analog Devices, Greensboro, North Carolina. *Analog Devices AD9027 Datasheet*, 1993.

[2] Analog Devices, Greensboro, North Carolina. *Analog Devices AD9026/27 Evaluation Board Datasheet*, 1994.

[3] D.M. Hummels, F.H. Irons, R. Cook, and I. Papantonopolous. Dynamic characterization of ADCs using a non-iterative procedure. In *ISCAS 94 vol 2*, pages 5–8, London, U.K., June 1994.

[4] D.M. Hummels, J. McDonald, and F.H. Irons. Distortion compensation for time-interleaved ADCs. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, Brussels, June 1996.

[5] F.H. Irons, D.M. Hummels, and I.N. Papantonopolous. ADC error diagnosis. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, Brussels, June 1996.

[6] F.H. Irons and T.A. Rebold. Characterization of high frequency analogto-digital converters for spectral applications, lincoln laboratory project report AST2. Technical report, MIT Lincoln Laboratory, Lexington, Massachusettes, November 1986.

[7] S.P. Kennedy. Improved ADC phase plane error compensation. Master's thesis, University of Maine, Orono, ME, May 1992.

[8] J.H. Larrabee, D.M. Hummels, and F.H. Irons. ADC compensation using a sinewave histogram method. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, Ottawa, May 1997.

[9] T.A. Rebold. Dynamic error correction method for high-speed analog-to-digital converters. Master's thesis, MIT, Cambridge, MA, May 1987.

[10] National Semiconductor. *Fast Logic Applicaitons Handbook*. National Semiconductor Inc., Santa Clara, CA, 1990.

[11] National Semiconductor. *National Advanced Bipolar Logic Databook*. National Semiconductor Inc., Santa Clara, CA, 1995.

APPENDIX A

ADC Compensator

The compensator details are illustrated in this appendix. Figures A.1 through A.5 give the schematic diagrams detailing the compensator. Physical layout of the compensator devices are given in Figures A.7 and A.6. Device cross-reference is given in Table A.1 and A.2. Compensator timing is given in Table A.3 and A.4 and Figures A.8 and A.9.

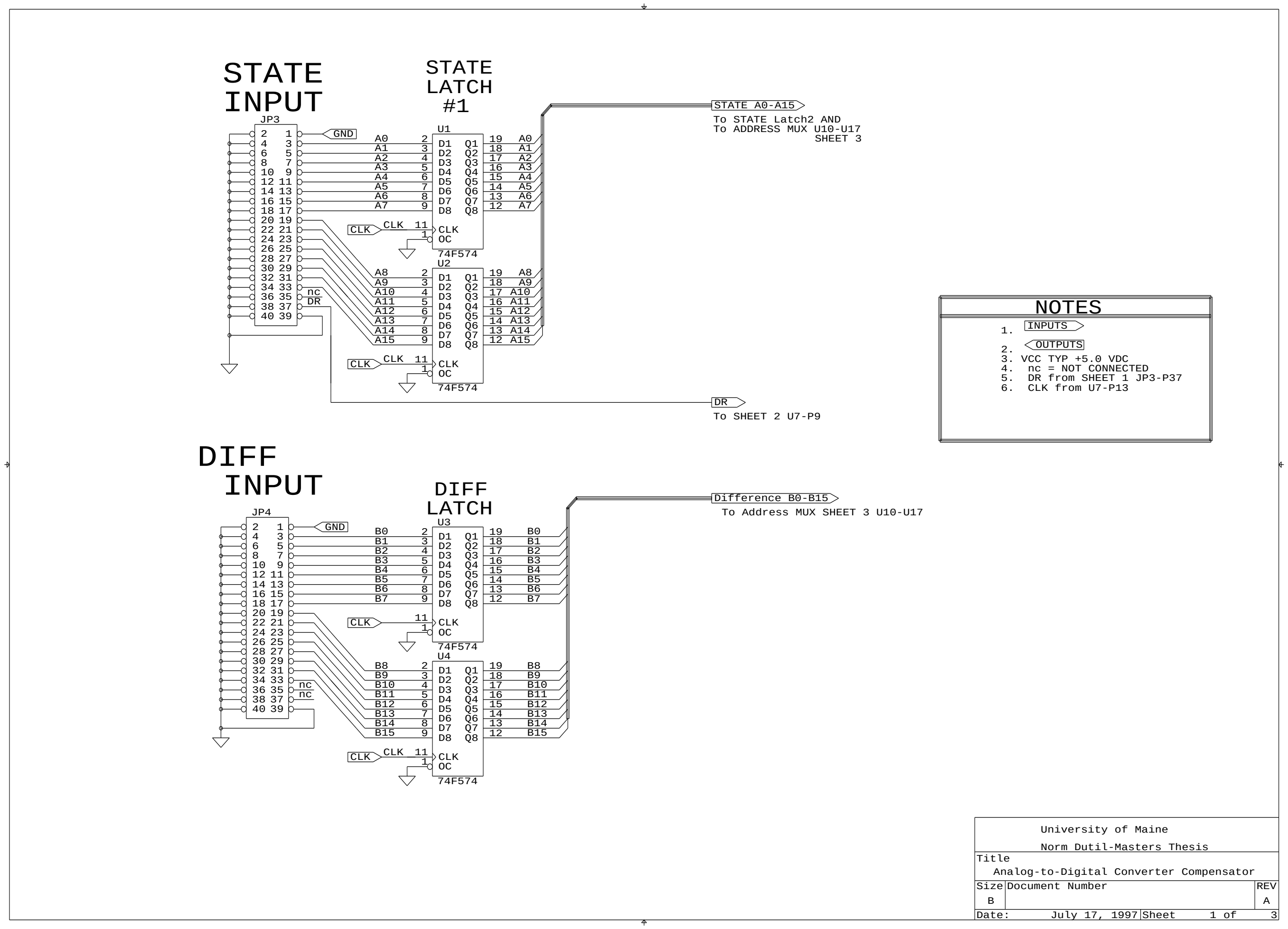


Figure A.1: Compensator Schematic Sheet 1 of 5

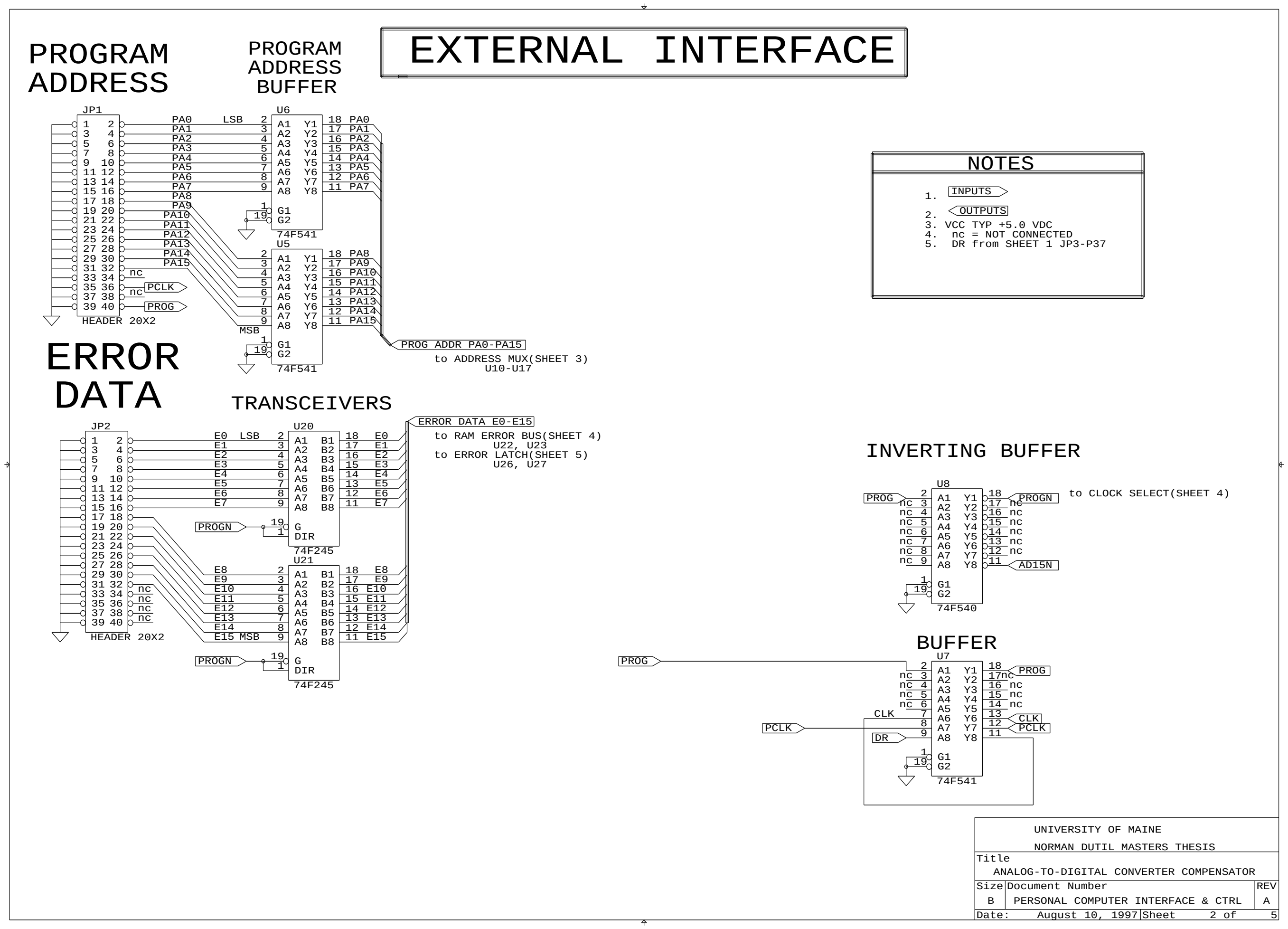


Figure A.2: Compensator Schematic Sheet 2 of 5

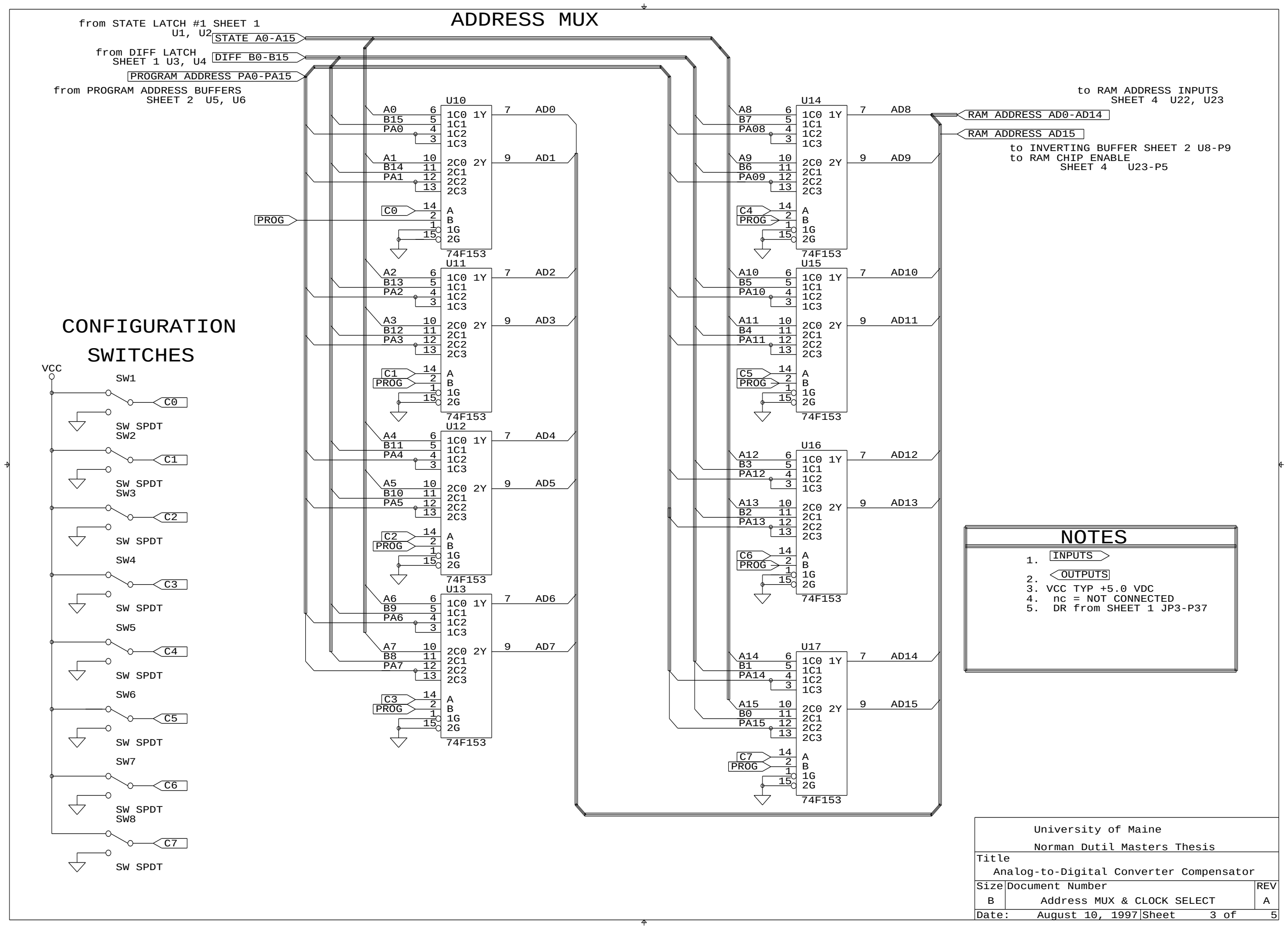


Figure A.3: Compensator Schematic Sheet 3 of 5

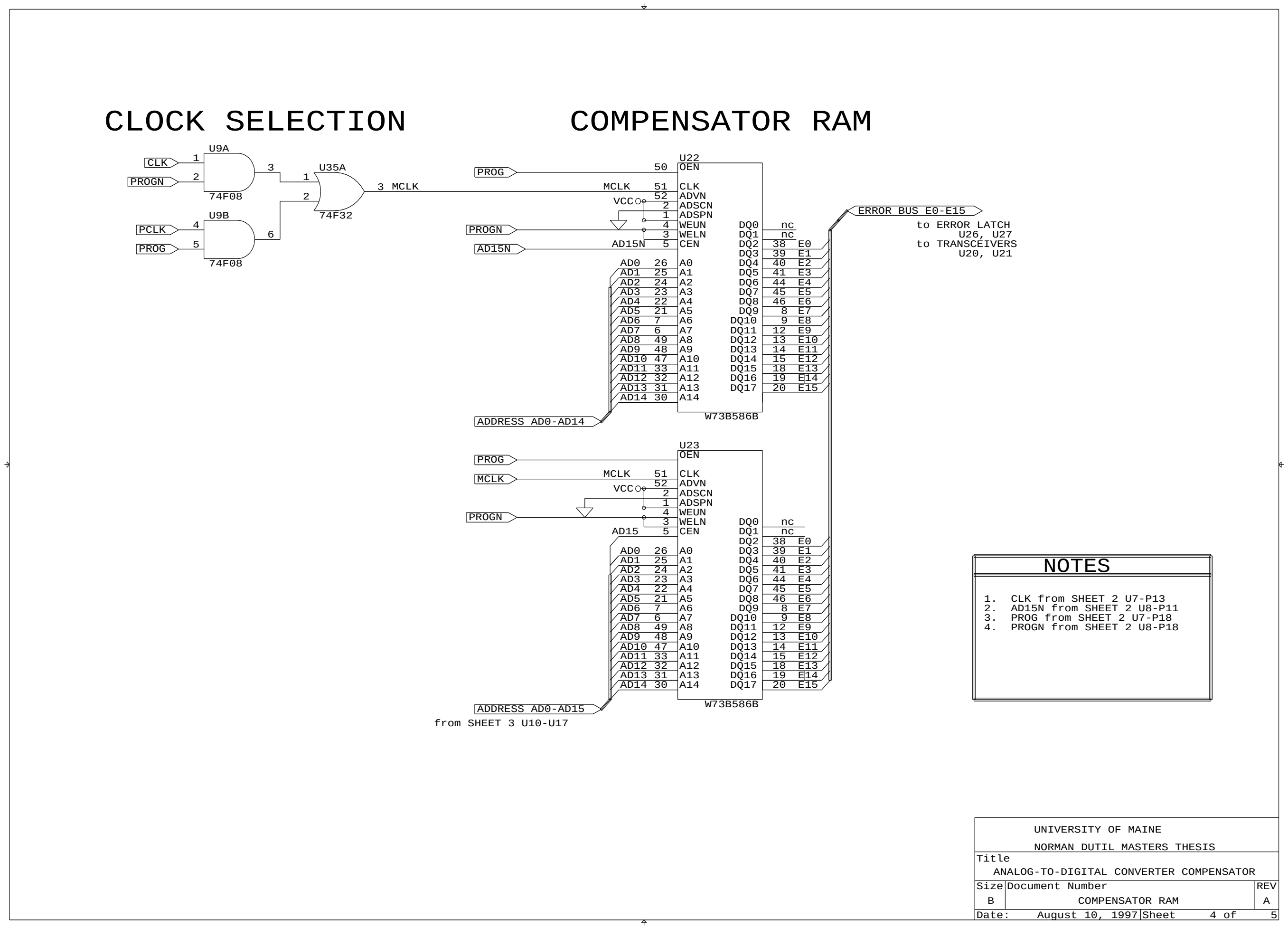


Figure A.4: Compensator Schematic Sheet 4 of 5

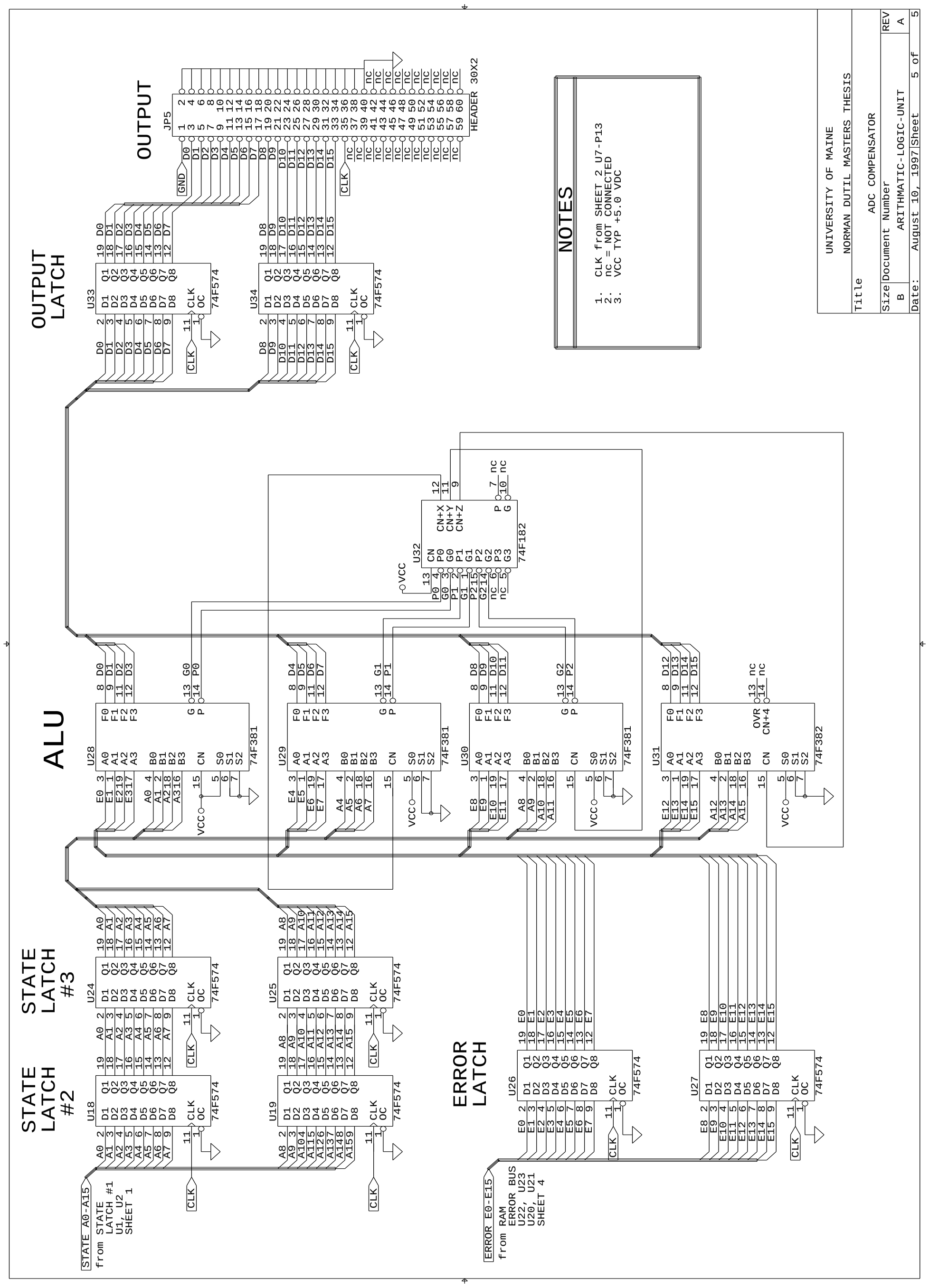


Figure A.5: Compensator Schematic Sheet 5 of 5

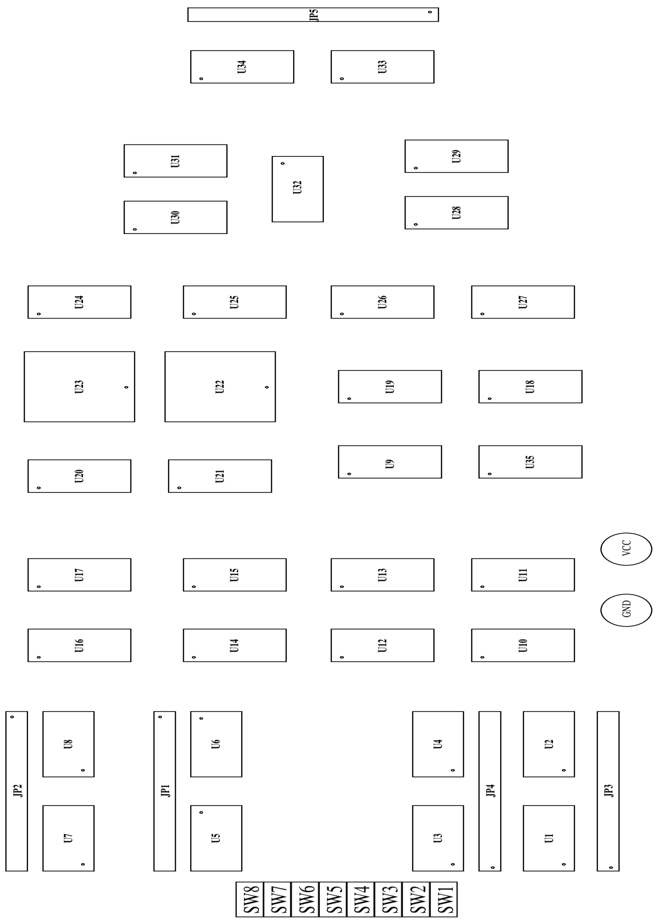


Figure A.6: Compensator Layout - Top View

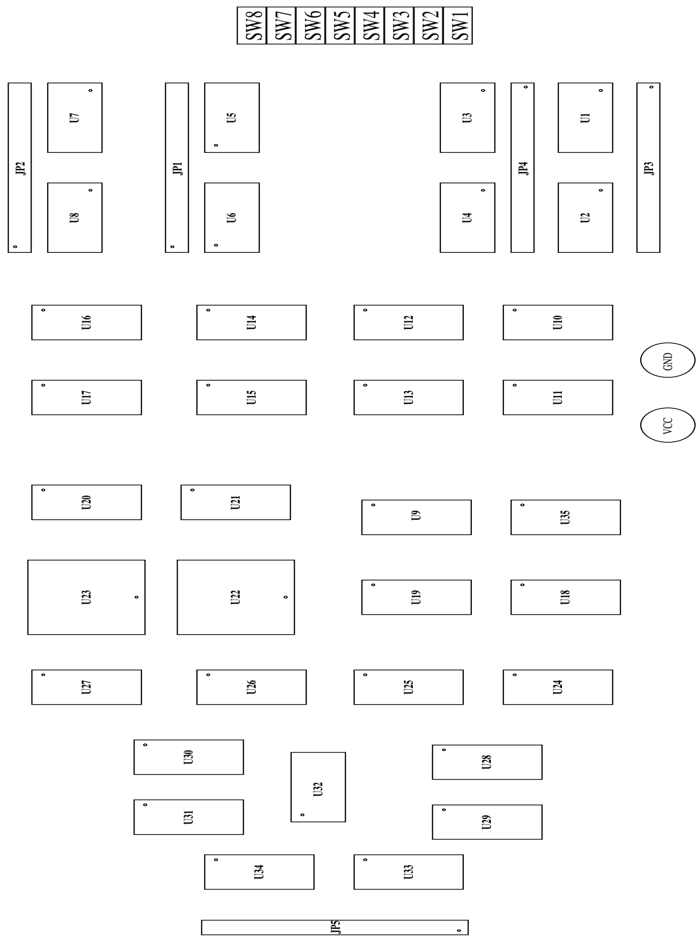


Figure A.7: Compensator Layout-Bottom View

UNIT ID	DEVICE	DESCRIPTION
U1	74F574PC	OCTAL D-TYPE FLIP-FLOP
U2	74F574PC	OCTAL D-TYPE FLIP-FLOP
U3	74F574PC	OCTAL D-TYPE FLIP-FLOP
U4	74F574PC	OCTAL D-TYPE FLIP-FLOP
U5	74F541PC	OCTAL BUFFER/LINE DRIVER
U6	74F541PC	OCTAL BUFFER/LINE DRIVER
U7	74F541PC	OCTAL BUFFER/LINE DRIVER
U8	74F540PC	OCTAL BUFFER/LINE DRIVER
U9	74F08PC	QUAD 2-INPUT AND GATE
U10	74F153PC	DUAL 4-INPUT MUX
U11	74F153PC	DUAL 4-INPUT MUX
U12	74F153PC	DUAL 4-INPUT MUX
U13	74F153PC	DUAL 4-INPUT MUX
U14	74F153PC	DUAL 4-INPUT MUX
U15	74F153PC	DUAL 4-INPUT MUX

Table A.1: Compensator Device Cross-Reference Part 1

U16	74F153PC	DUAL 4-INPUT MUX
U17	74F153PC	DUAL 4-INPUT MUX
U18	74F574PC	OCTAL D-TYPE FLIP-FLOP
U19	74F574PC	OCTAL D-TYPE FLIP-FLOP
U20	74F245PC	OCTAL BIDIRECTIONAL TRANSCEIVER
U21	74F245PC	OCTAL BIDIRECTIONAL TRANSCEIVER
U22	W73B586B	32K X 18 SYNCHRONOUS BURST CACHE RAM
U23	W73B586B	32K X 18 SYNCHRONOUS BURST CACHE RAM
U24	74F574PC	OCTAL D-TYPE FLIP-FLOP
U25	74F574PC	OCTAL D-TYPE FLIP-FLOP
U26	74F574PC	OCTAL D-TYPE FLIP-FLOP
U27	74F574PC	OCTAL D-TYPE FLIP-FLOP
U28	74F381PC	4-BIT ALU
U29	74F381PC	4-BIT ALU
U30	74F381PC	4-BIT ALU
U31	74F382PC	4-BIT ALU
U32	74F182PC	CARRY LOOKAHEAD GENERATOR
U33	74F574PC	OCTAL D-TYPE FLIP-FLOP
U34	74F574PC	OCTAL D-TYPE FLIP-FLOP
U35	74F32PC	QUAD 2-INPUT OR

Table A.2: Compensator Device Cross-Reference Part 2

Spec	Description	Min Value	Max Value	Units
tsl	74F574 D-FF Setup Time	5		ns
thl	74F574 D-FF Hold Time	5		ns
tpl	74F574 D-FF Propagation Delay	3.5		ns
tcdv	mclk to Data Valid	9		ns
t_cycle	Total Cycle Time	40		ns
c2c_skew	Clock-to-Clock Skew	1.4		ns
tpd_ALU	ALU Propagation Delay	20.4		ns
tdv_ALU	ALU Data Valid	18		ns
t_latency	Total Pipeline Delay	4		cycles

Table A.3: Compensator Read-Mode Timing Specifications

Spec	Description	Min Value	Max Value	Units
tse	Memory Setup	2.5		ns
the	Memory Hold	0.5		ns
pad	Program Address Delay	4.5		ns
tprogs	Program Setup	1		ms
t-cycle	Total Cycle	40		ns
c2c-skew	pclk-to-mclk skew	1.4		ns

Table A.4: Compensator Program-Mode Timing Specifications

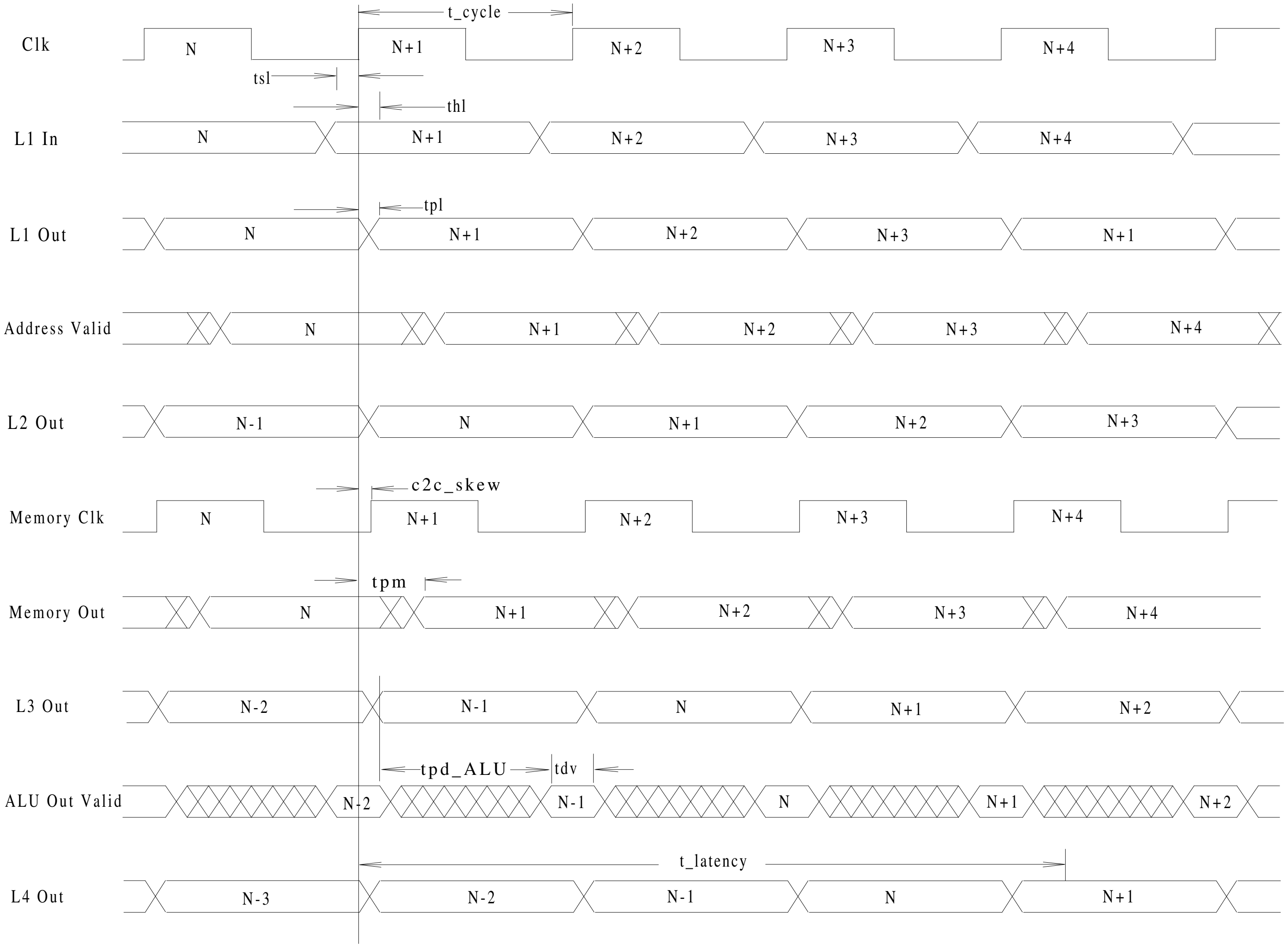


Figure A.8: Compensator Read-Mode Timing

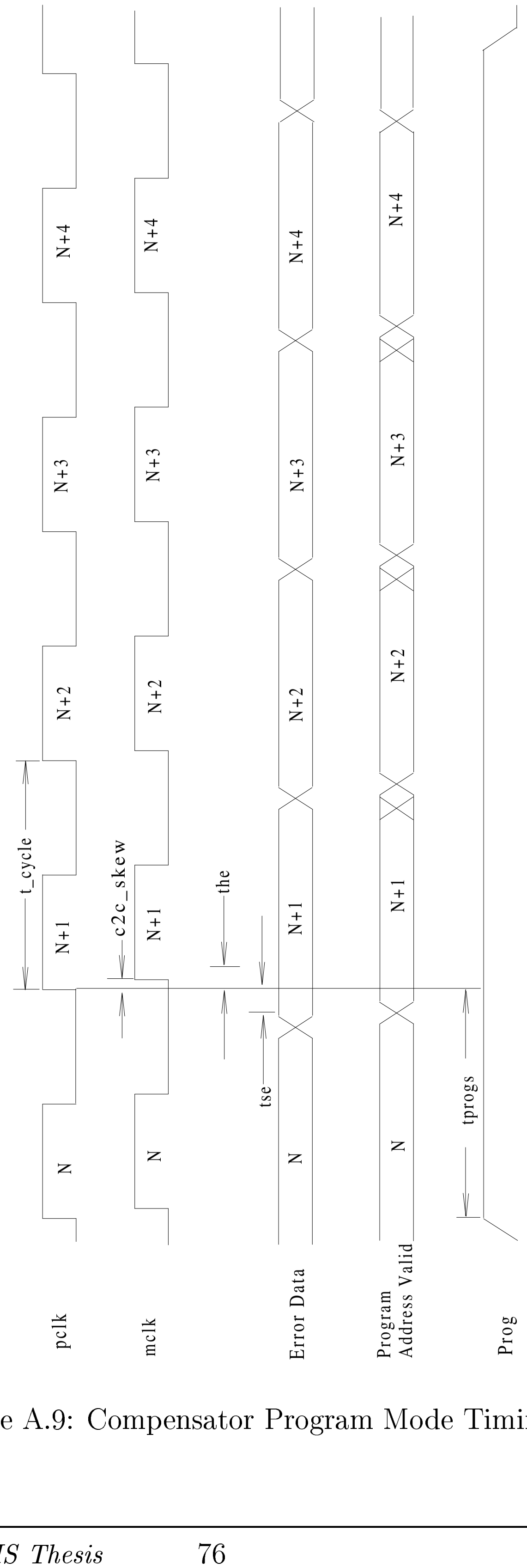


Figure A.9: Compensator Program Mode Timing

APPENDIX B

Difference Generator

The Difference Generator details are illustrated in this appendix. Figure B.1 gives the schematic diagrams detailing the Difference Generator. Physical layout of the Difference Generator devices are given in Figures B.2. Device cross-reference is given in Table B.1. Difference Generator timing is given in Table B.2 and Figure B.3.

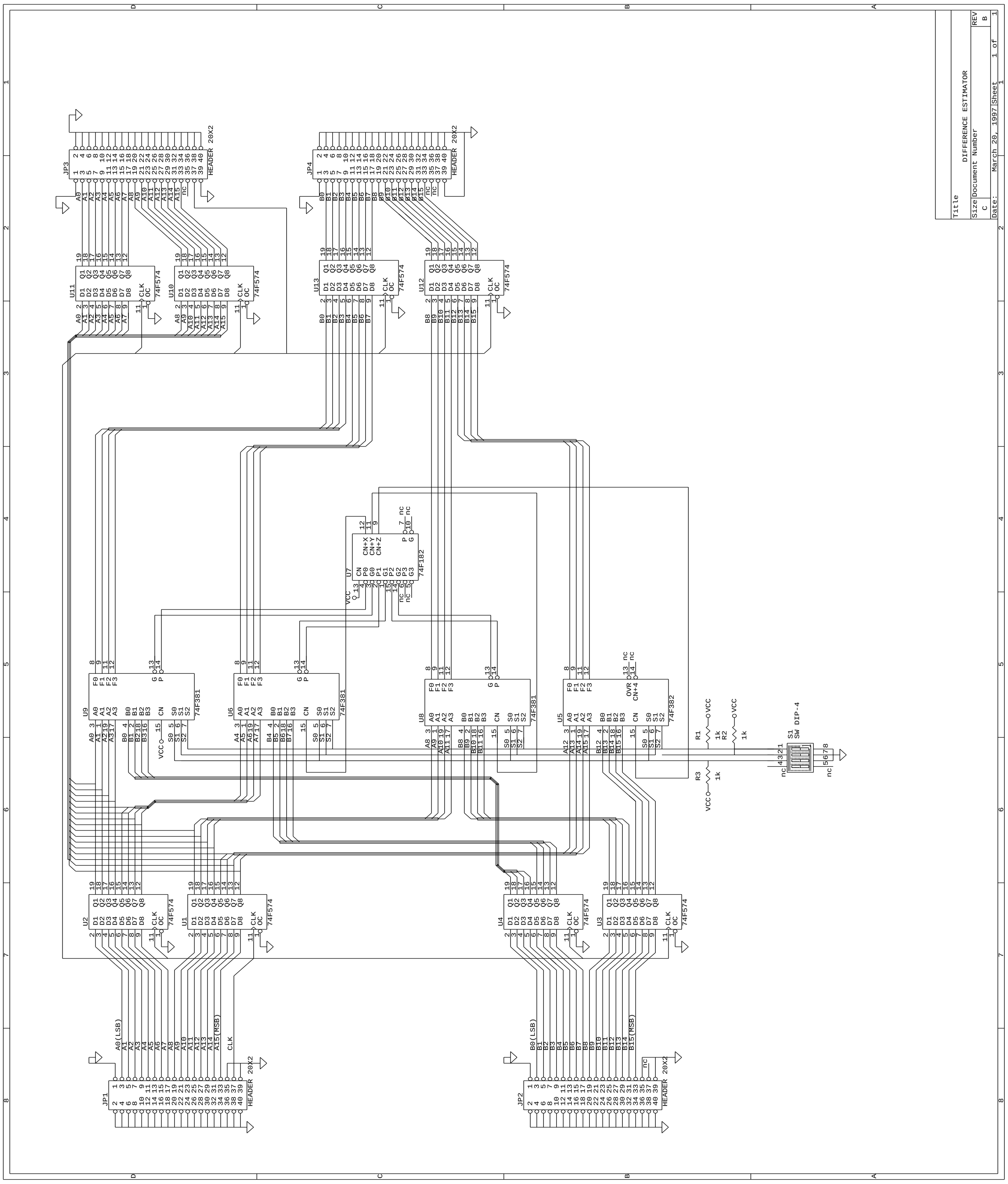
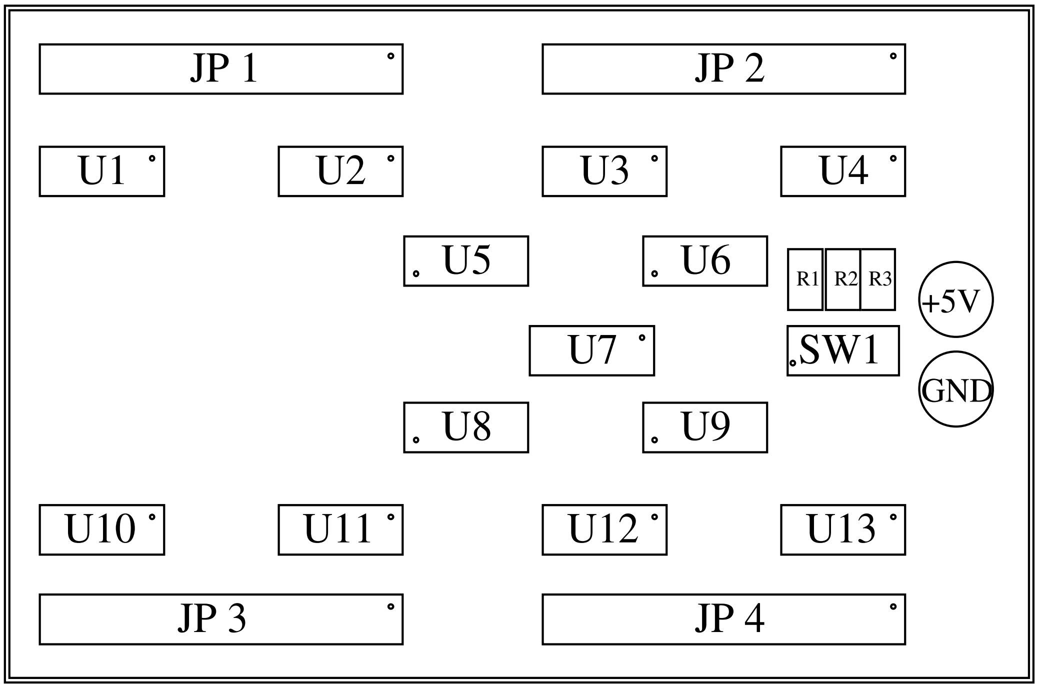
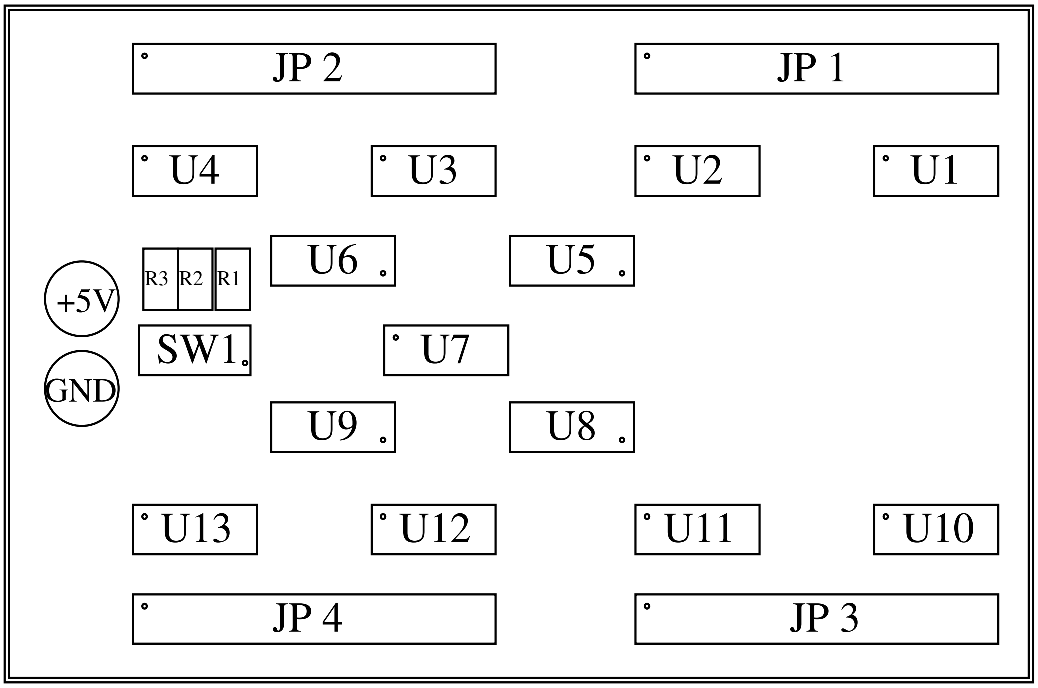


Figure B.1: Difference Generator Schematic



Difference Generator Top View



Difference Generator Bottom View

Figure B.2: Difference Generator Unit Layout

UNIT ID	DEVICE	DESCRIPTION
U1	74F574PC	OCTAL D-TYPE FLIP-FLOP
U2	74F574PC	OCTAL D-TYPE FLIP-FLOP
U3	74F574PC	OCTAL D-TYPE FLIP-FLOP
U4	74F574PC	OCTAL D-TYPE FLIP-FLOP
U5	74F382PC	4-BIT ALU
U6	74F381PC	4-BIT ALU
U7	74F182PC	CARRY LOOKAHEAD GENERATOR
U8	74F381PC	4-BIT ALU
U9	74F381PC	4-BIT ALU
U10	74F574PC	OCTAL D-TYPE FLIP-FLOP
U11	74F574PC	OCTAL D-TYPE FLIP-FLOP
U12	74F574PC	OCTAL D-TYPE FLIP-FLOP
U13	74F574PC	OCTAL D-TYPE FLIP-FLOP
R1-R3	1K OHM	RESISTOR

Table B.1: Difference Generator Device Cross-Reference

Spec	Description	Min Value	Max Value	Units
tcycle	Period	40		ns
ts	74F574 Setup	5		ns
th	74F574 Hold	5		ns
tp	74F574 Prop Delay	5		ns
tpALU	ALU Prop Delay	20.4		ns
t-cycle	Total Cycle	40		ns
tpdo	Output Prop Delay	5		ns

Table B.2: Difference Generator Timing Specifications

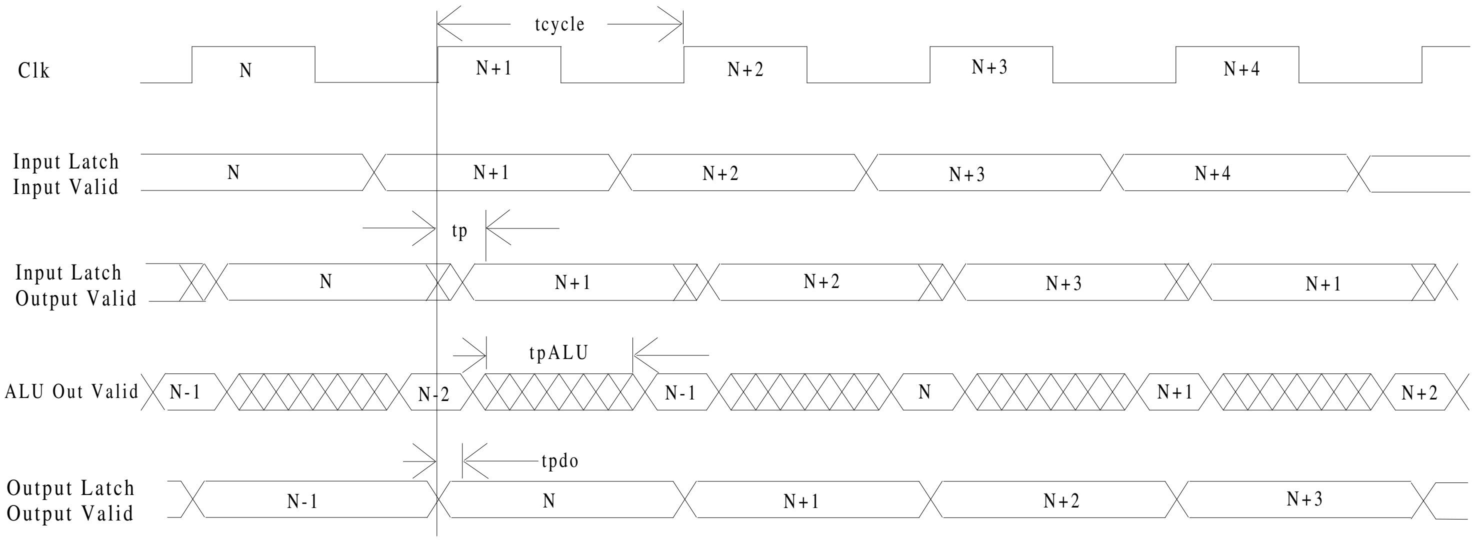


Figure B.3: Difference Generator Timing

Input	Value	Connection
Power VCC	+5.0 Volt 0.5 Amp	BJ01
Ground	0.0 Volt	BJ02

Table B.3: Difference Generator Power & Ground Inputs

APPENDIX C

ECL-to-TTL Translator

The ECL-to-TTL Translator details are illustrated in this appendix. Figure C.1 gives the schematic diagrams detailing the ECL-to-TTL Translator. Physical layout of the ECL-to-TTL Translator devices are given in Figures C.2. Device cross-reference is given in Table C.1. ECL-to-TTL Translator power is given in Table C.2. Timing diagrams are not given for this circuit. The total effect of this circuit is to delay all signals by 2-4 ns.

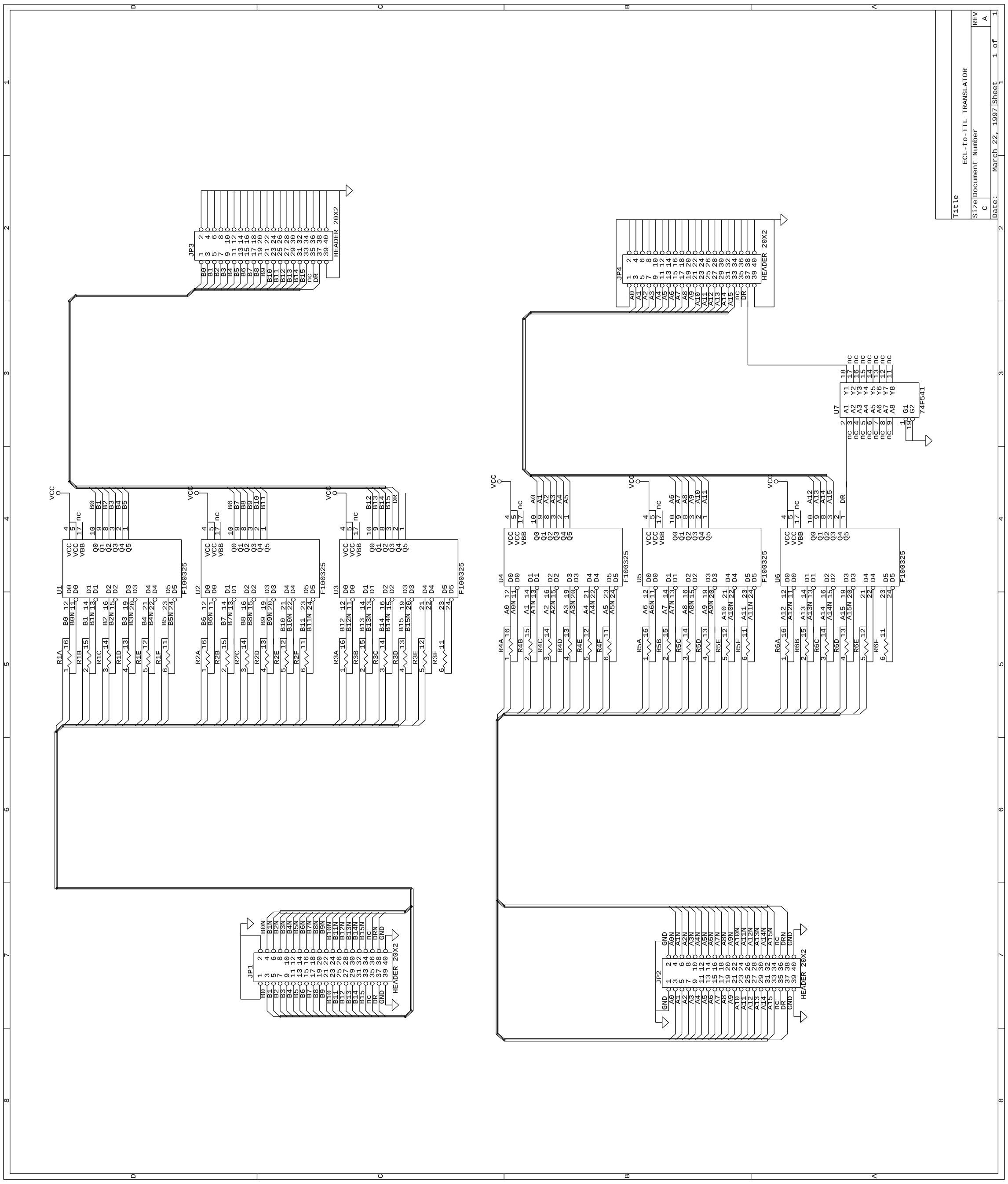


Figure C.1: ECL-to-TTL Translator Schematic

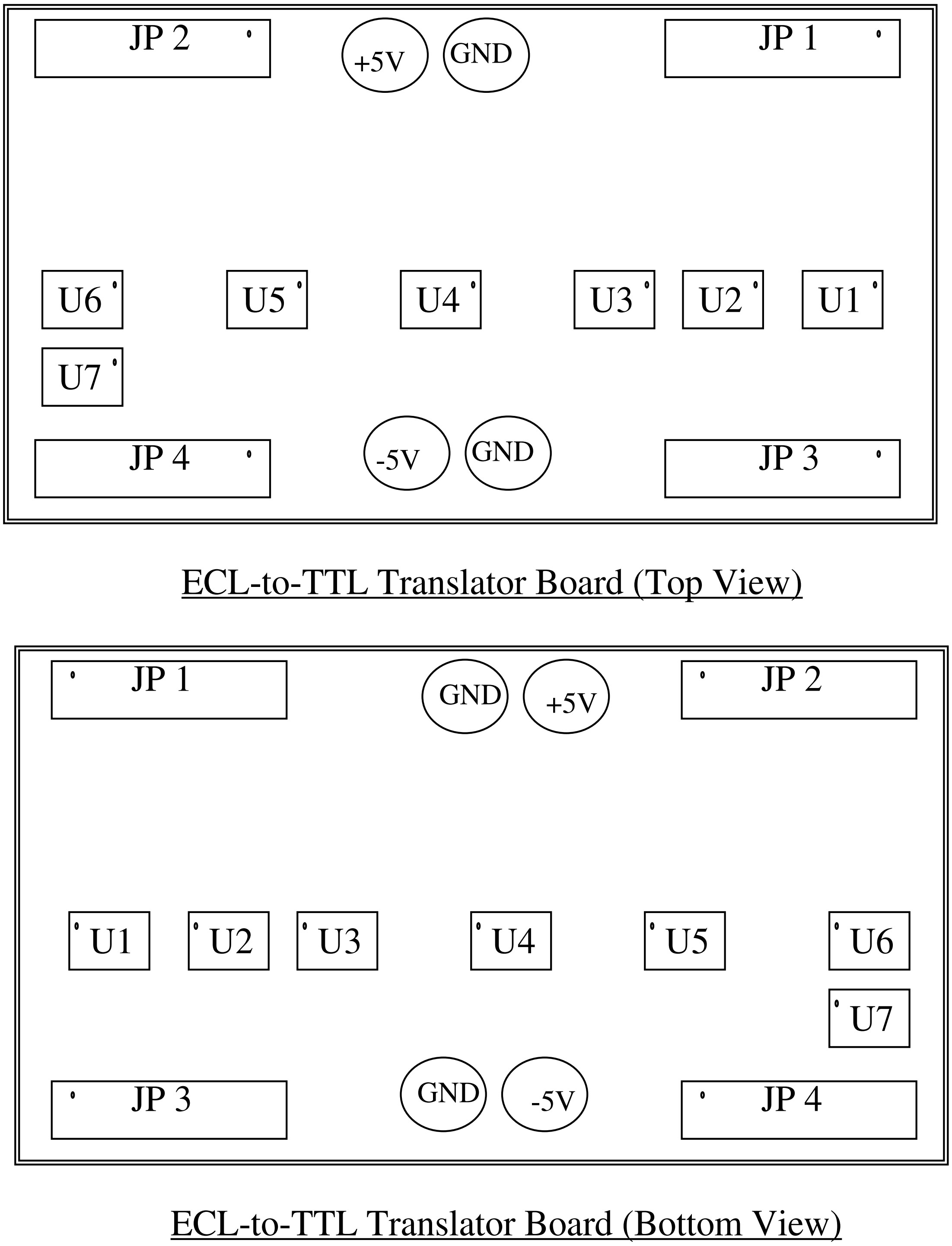


Figure C.2: ECL-to-TTL Translator Layout

UNIT ID	DEVICE	DESCRIPTION
U1	100325	HEX ECL-TO-TTL TRANSLATOR
U2	100325	HEX ECL-TO-TTL TRANSLATOR
U3	100325	HEX ECL-TO-TTL TRANSLATOR
U4	100325	HEX ECL-TO-TTL TRANSLATOR
U5	100325	HEX ECL-TO-TTL TRANSLATOR
U6	100325	HEX ECL-TO-TTL TRANSLATOR
U7	74f541	OCTAL BUFFER/LINE DRIVER
R1-R6	100 OHM RESISTOR 8 PACK	

Table C.1: ECL-to-TTL Translator Device Cross-Reference

Input	Value	Connection
Power VEE	-5.2 Volt 0.2 Amp	BJ01
Power VCC	+5.0 Volt 0.3 Amp	BJ02
Ground	0.0 Volt	BJ03

Table C.2: ECL-to-TTL Translator Power & Ground Inputs

APPENDIX D

PC Interface

The PC Interface details are illustrated in this appendix. Figure D.1 gives the schematic diagrams detailing the PC Interface. Physical layout of the PC Interface devices are given in Figure D.2. Device cross-reference is given in Table D.1. PC Interface timing is given in Table D.2 and Figure D.3.

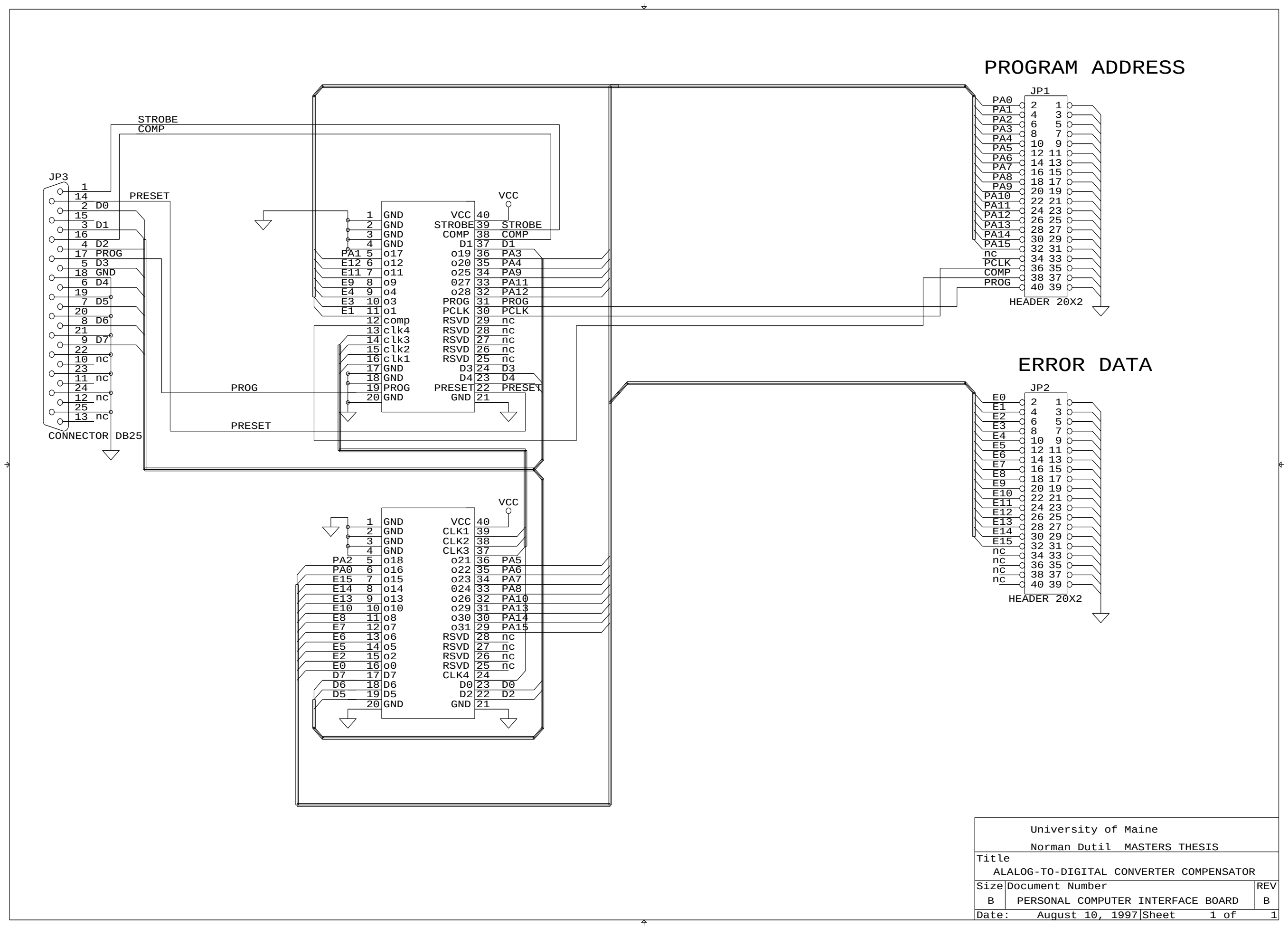
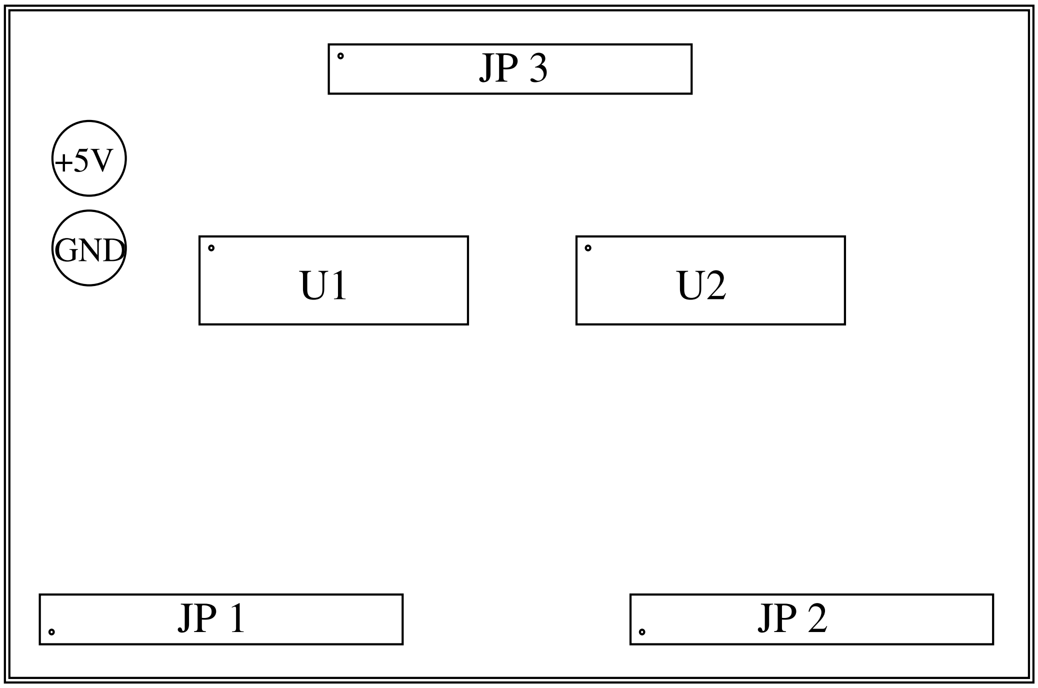
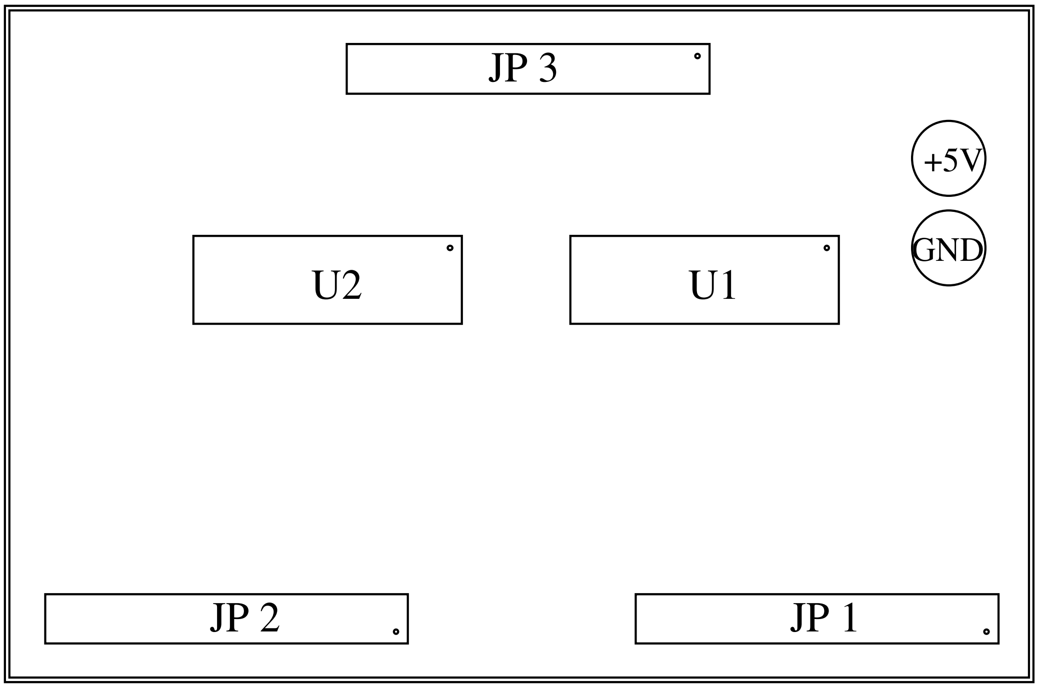


Figure D.1: PC Interface Schematic



Top View



Bottom View

Figure D.2: PC Interface Unit Layout

UNIT ID	DEVICE	DESCRIPTION
U1	EP910DG40	ALTERA
U2	EP910DG40	ALTERA

Table D.1: Personal Computer Interface Device Cross-Reference

Spec	Description	Min Value	Max Value	Units
Strobe	Period	50		ms
Preset	Period	50		ms
PCLK	Period	250		ms

Table D.2: PC Interface Timing Specifications

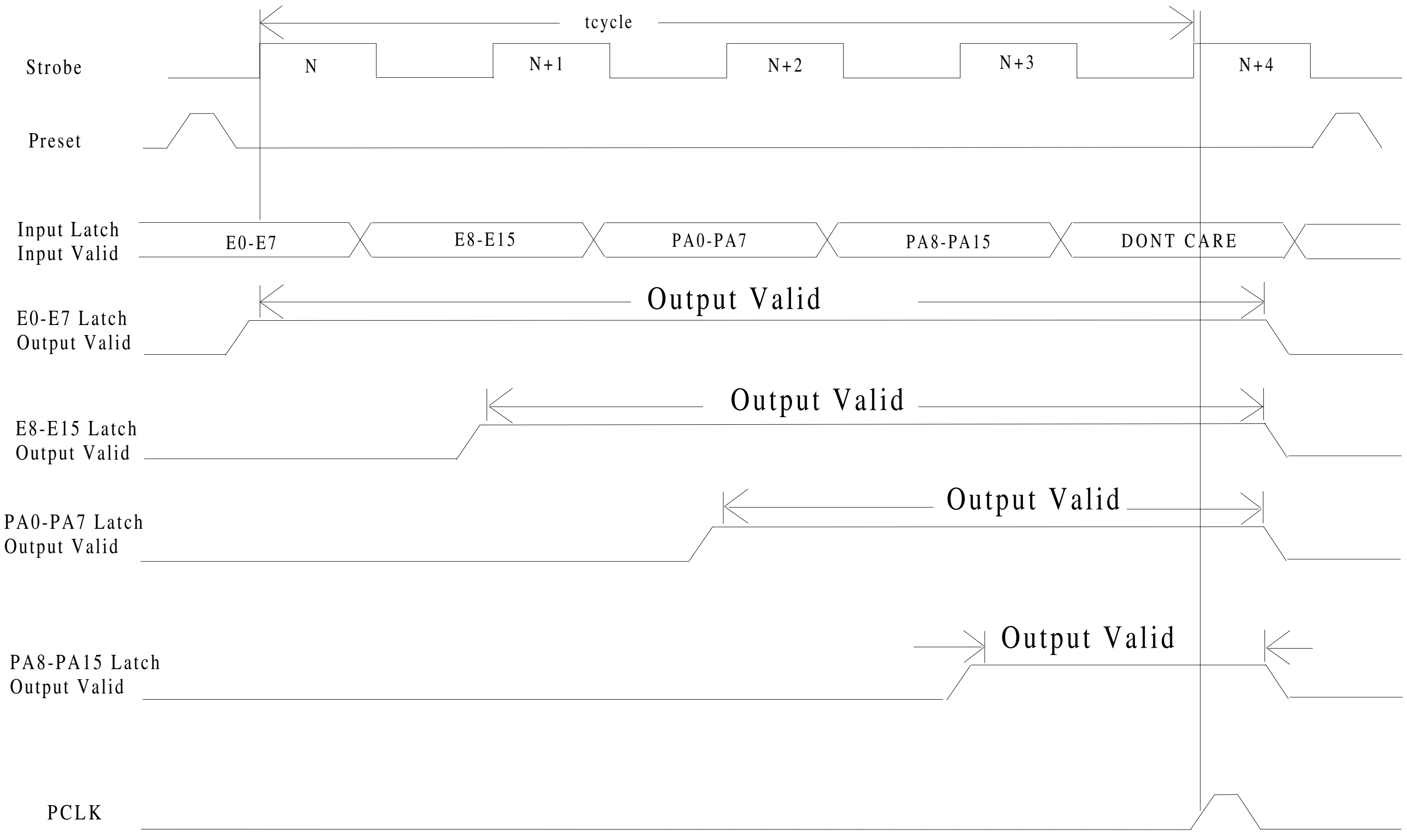


Figure D.3: PC Interface Timing

APPENDIX E

Compensator Software

```
/*****************************************************************************/
/*   Source Code:  loader.cpp                                                */
/*   Executable :  loader.exe                                                */
/*   Author:        Norm Dutil 4/21/96                                       */
/*                                                                           */
/*   Purpose:   loader.exe loads an error table into compensator RAM.        */
/*              The error table must be pre-formatted by Matlab              */
/*              script HARDMAP.m.                                            */
/*                                                                           */
/*   INPUT:     ASCII file containing two columns.                           */
/*              data.  Column 1 contains the RAM address for a error value   */
/*              stored in column 2.                                          */
/*                                                                           */
/*   OUTPUT:    Audible tone from PC indicating error table was transferred. */
/*                                                                           */
/*                                                                           */
/*   Overview:                                                               */
/*              Each line of the preformatted error table(file)              */
/*              is read and parsed into four bytes. The first byte is        */
/*              loaded into the parallel-port data register, which is        */
/*              tied directly to the 25-pin printer port on the rear         */
/*              of the PC.  This port is connected to the PC Interface       */
/*              circuit board which latches each byte placed in the data     */
/*              register.  Three control lines                               */
/*              Parallel port control register bits C0, C1 and C3 facilitate */
/*              the data transfer.  C0 clocks the data into PC Interface latches*/
/*              C1 presets the PC Interface shift-register while C3 places all */
/*              Compensator hardware in Program Mode.                        */
/*                                                                           */
/*****************************************************************************/


#include <stdio.h>
#include <process.h>
#include <stdlib.h>
#include <bios.h>
#include <math.h>
#include <dos.h>
#include <string.h>
#include <conio.h>

#define LPT1_LOOKUP 0x408  // LPT1 ADDRESS
#define MAX_ADDRESS 65535  // LARGEST SRAM ADDRESS MINUS 1, FOR 64K MEMORY
void main(void)
{
FILE *infile;

long i;

long      data,
          address;
int control,
    p_port,
    j,
    outword[5];

char *stuff,
     table_fn[100],
     temp[40];
```

```
p_port   = *(int far *)LPT1_LOOKUP;
control  = p_port+2;

clrscr();
printf("Enter Error Table File Name \n");
scanf("%s",table_fn);
printf("\n File : %s is Downloading to Compensator RAM.\n",table_fn);
infile = fopen(table_fn,"r");
disable();  // DISABLE INTERRUPTS WHILE I EXECUTE
 outportb(control,0x00);  //Program line High Bit C3(complemented output)

for (i=0;i<MAX_ADDRESS+1;i++)
 {
 stuff = fgets(temp,40,infile);   // READ IN DATA A LINE AT A TIME AS A STRING
 sscanf(stuff,"%ld %ld\n",&address,&data); //WEED OUT THE ADDRESS AND DATA
 data = data&0xffff;
 address = address&0xffff;
 // SEPARATE THE ADDRESS & DATA INTO TWO BYTES EACH
outword[0] =data&0xff;                //DATA_LSB
outword[1] =(data/256)&0xff;          //DATA_MSB
outword[2]  =address&0xff;            //ADDRESS_LSB
outword[3]  =(address/256)&0xff;      //ADDRESS_MSB


// INITIALIZE COMPENSATOR SHIFT REGISTER
  outportb(control,0x03); //clk low & pset low    CLOCK IS BIT C0
  outportb(control,0x01); //clk low & pset high   PSET  IS BIT C1
  outportb(control,0x00); //clk high& pset high
  outportb(control,0x03); //clk low & pset low

// SEND DATA TO PRINTER PORT & CLOCK
  for (j=0;j<4;j++)
  {
    outportb(p_port,outword[j]);
//DATA TO PRINTER PORT
    outportb(control,0x03);                  //clk low & pset low
    outportb(control,0x02);                  //clk high & pset low
    outportb(control,0x03);                  //clk low & pset low
  }
//FINAL CLOCK TO WRITE DATA OUT OF PC INTERFACE Board
 outportb(control,0x03); //clk low & pset low
 outportb(control,0x02); //clk high & pset low
 outportb(control,0x03); //clk low & pset low
 }
enable();
fclose(infile);
outportb(control,0xc);  //Pull Program line low
printf("\n\n\n Compensator RAM Loading is Complete \a\a\a \n ");
}
```

```
%% Script Code:  hardmap.m
%%
%% Purpose:  Translate error table from logical addressing to physical
%%           addressing and format 64-bit error data into 16-bit error
%%           word to store in compensator RAM.
%% Inputs:      table = error table with N state rows by M slope cols
%%              max_slope   = Max slope determined by Difference estimator
%% Output:   Compensator Formatted Error Table File.
%%
%% CALL  hardmap(table,maxslope);
 function  hardmap(table,maxslope);

%% Format error data

[rows,cols]=size(table);        % Determine size of input error table.
N = log2(rows);                 % Number of Error Table State Bits.
M = log2(cols);                 % Number of Error Table Slope Bits.
table = round(table * 16);      % Scale error data by 4 lsbs.
table = table + (table < 0)*2^16; % Take Twos Complement of Negative values

%
%%  Format RAM Address
%

addr = 0:(64*1024-1)';                  % RAM Address Array

%
%% Generate Error Table Index Values based on each address in addr array.
%

state = floor(addr/(2^(16-N)));         % State values from address word
bits = dec2bin(addr,16);                % Slope values from address word
slope = bits(:,16:-1:16-M+1)*(2 .^[M-1:-1:0]');
slope = (slope - (2^M)*bits(:,16))';    % Convert negative slope values to twos complement
state_index = state+1;
slope_index = slope+(2^(M-1))+1;
error_vec = zeros(size(addr));

%
%% Loop through all addresses and select the value associated with state and slope index.
%

for k = 1:length(error_vec)
  error_vec(k) = table(state_index(k), slope_index(k));
end; % for k

%
%% Write out the results.
%

fid = fopen('loadertable.mat','w');  % File to store address and error vectors
for m = 1:length(addr)
 fprintf(fid,'%8.0f %5.0f \n', addr(m), error_vec(m));
end;
fclose(fid);
```

APPENDIX F

Analog Devices AD9027

This appendix gives connectivity details for the Analog Devices AD9027 12-bit ADC. Inputs are given in Table F.1. Digital outputs are given in Table F.2 and F.3. Timing and evaluation board specifics are given in [1, 2].

Input	Value	Connection
Power VEE	-5.2 Volt 0.75 Amp	BJ101
Power VCC	+5.0 Volt 0.25 Amp	BJ100
Ground	0.0 Volt	BJ104
Analog	2.0 Volt peak-to-peak	AIN
Encode	1.0 Volt peak-to-peak minimum	ECLEN

Table F.1: Analog Devices AD9027 Evaluation Board Inputs

From ADC #1	To IXBD #1	Signal	To IXBD #1	Signal	To ECL-TO-TTL Translator
J200 P1	JP1-P1	GND	JP2-P1	GND	JP2-P1
J200 P2	JP1-P2	DRB	JP2-P38	ADRN	JP2-P38
J200 P3	JP1-P3	D4B	JP2-P20	A4N	JP2-P20
J200 P4	JP1-P4	D5B	JP2-P22	A5N	JP2-P22
J200 P5	JP1-P5	D6B	JP2-P24	A6N	JP2-P24
J200 P6	JP1-P6	D7B	JP2-P26	A7N	JP2-P26
J200 P7	JP1-P7	D8B	JP2-P28	A8N	JP2-P28
J200 P8	JP1-P8	D9B	JP2-P30	A9N	JP2-P30
J200 P9	JP1-P9	D10B	JP2-P32	A10N	JP2-P32
J200 P10	JP1-P10	D11B	JP2-P34	A11N	JP2-P34
J200 P11			NC		
J200 P12			NC		
J200 P13			NC		
J200 P14			NC		
J200 P15	JP1-P15	D0B	JP2-P12	A0N	JP2-P12
J200 P16	JP1-P16	D1B	JP2-P14	A1N	JP2-P14
J200 P17	JP1-P17	D2B	JP2-P16	A2N	JP2-P16
J200 P18	JP1-P18	D3B	JP2-P18	A3N	JP2-P18
J200 P19	JP1-P19	GND	JP2-P2	GND	JP2-P2
J200 P20		NC			
J200 P21	JP1-P21	DR	JP2-P37	ADR	JP2-P37
J200 P22	JP1-P22	D4	JP2-P19	A4	JP2-P19
J200 P23	JP1-P23	D5	JP2-P21	A5	JP2-P21
J200 P24	JP1-P24	D6	JP2-P23	A6	JP2-P23
J200 P25	JP1-P25	D7	JP2-P25	A7	JP2-P25
J200 P26	JP1-P26	D8	JP2-P27	A8	JP2-P27
J200 P27	JP1-P27	D9	JP2-P29	A9	JP2-P29
J200 P28	JP1-P28	D10	JP2-P31	A10	JP2-P31
J200 P29	JP1-P29	D11	JP2-P33	A11	JP2-P33
J200 P30		NC			
J200 P31		NC			
J200 P32		NC			
J200 P33		NC			
J200 P34	JP1-P34	D0	JP2-P11	A0	JP2-P11
J200 P35	JP1-P35	D1	JP2-P13	A1	JP2-P13
J200 P36	JP1-P36	D2	JP2-P15	A2	JP2-P15
J200 P37	JP1-P37	D3	JP2-P17	A3	JP2-P17

Table F.2: ADC #1 Pin-Out

From ADC #2	To IXBD #2	Signal	To IXBD #2	Signal	To ECL-TO-TTL Translator
J200 P1	JP1-P1	GND	JP2-P1	GND	JP1-P1
J200 P2	JP1-P2	DRB	JP2-P38	BDRN	JP1-P38
J200 P3	JP1-P3	D4B	JP2-P20	B4N	JP1-P20
J200 P4	JP1-P4	D5B	JP2-P22	B5N	JP1-P22
J200 P5	JP1-P5	D6B	JP2-P24	B6N	JP1-P24
J200 P6	JP1-P6	D7B	JP2-P26	B7N	JP1-P26
J200 P7	JP1-P7	D8B	JP2-P28	B8N	JP1-P28
J200 P8	JP1-P8	D9B	JP2-P30	B9N	JP1-P30
J200 P9	JP1-P9	D10B	JP2-P32	B10N	JP1-P32
J200 P10	JP1-P10	D11B	JP2-P34	B11N	JP1-P34
J200 P11			NC		
J200 P12			NC		
J200 P13			NC		
J200 P14			NC		
J200 P15	JP1-P15	D0B	JP2-P12	B0N	JP1-P12
J200 P16	JP1-P16	D1B	JP2-P14	B1N	JP1-P14
J200 P17	JP1-P17	D2B	JP2-P16	B2N	JP1-P16
J200 P18	JP1-P18	D3B	JP2-P18	B3N	JP1-P18
J200 P19	JP1-P19	GND	JP2-P2	GND	JP1-P2
J200 P20		NC			
J200 P21	JP1-P21	DR	JP2-P37	BDR	JP1-P37
J200 P22	JP1-P22	D4	JP2-P19	B4	JP1-P19
J200 P23	JP1-P23	D5	JP2-P21	B5	JP1-P21
J200 P24	JP1-P24	D6	JP2-P23	B6	JP1-P23
J200 P25	JP1-P25	D7	JP2-P25	B7	JP1-P25
J200 P26	JP1-P26	D8	JP2-P27	B8	JP1-P27
J200 P27	JP1-P27	D9	JP2-P29	B9	JP1-P29
J200 P28	JP1-P28	D10	JP2-P31	B10	JP1-P31
J200 P29	JP1-P29	D11	JP2-P33	B11	JP1-P33
J200 P30		NC			
J200 P31		NC			
J200 P32		NC			
J200 P33		NC			
J200 P34	JP1-P34	D0	JP2-P11	B0	JP1-P11
J200 P35	JP1-P35	D1	JP2-P13	B1	JP1-P13
J200 P36	JP1-P36	D2	JP2-P15	B2	JP1-P15
J200 P37	JP1-P37	D3	JP2-P17	B3	JP1-P17

Table F.3: ADC #2 Pin-Out

Biography of the Author

Norman Dutil was born in Metheun, Massachusetts on May 17, 1965. He received his high school education from Saint Dominic Regional High School in Lewiston, Maine in 1983.

He served as a Firecontrolman in the U.S. Navy from 1983 to 1989. In the fall of 1989, he entered the University of Maine where he was selected as the third Castle Student and obtained his Bachelor of Science degree in Electrical Engineering in 1994.

In June 1994, he enrolled for graduate study in Electrical Engineering at the University of Maine and served as Research Assistant. In February 1996 he accepted a position with National Semiconductor of Santa Clara, CA. His current interests involve device level test engineering. He is a member of IEEE and his interests include hiking and travel. He is a candidate for the Master of Science degree in Electrical Engineering from the University of Maine in August 1997.