INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

An Integrable Mixed-Signal Test System

by

Ara Hajjar, B. Eng. 1996

Department of Electrical Engineering

McGill University, Montréal

November 1998

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Engineering

Abstract

The growing need for integrable test solutions has prompted the creation of various test bus standards. A mixed-signal test core is an ideal complement to these standards. This work presents the design and implementation of an integrable test system. The design consists of two major components: a stimulus generator, and a waveform extractor.

A memory-based generator is used to construct the stimulus generation component. Such a circuit repeats a finite portion of an infinite-length PDM sequence in order to produce any arbitrary analog waveform. The circuitry is simple to design - it is comprised of a scan chain, and a 1-bit DAC; it is also area-efficient and robust (mostly digital design). Furthermore, since the analog signal is generated from a digital bit-stream, it is both stable and repeatable.

The extraction component of the test system focuses on the capture of steady-state type responses. A novel A/D algorithm is presented: the Multi-Pass technique. By taking advantage of repetitive waveforms, the Multi-Pass convertor achieves both area-efficiency and high-speed performance. A single on-chip comparator and sample-and-hold circuit is sufficient to extract analog waveforms. In addition, a novel, area-efficient, integrable, and highly-linear voltage reference design is presented.

Experimental results from two prototype boards serve to validate the proposed test system design. The first board implements the system using discrete components; the second makes use of a custom IC fabricated in a 0.5 μm CMOS process. The work presented in this thesis provides the groundwork for obtaining a practical and fully integrable mixed signal test system.

Résumé

Le besoin croissant en solutions de test intégrées fit naître divers standards de bus de test. Un noyau de test analogique-numérique forme un complément idéal à ces standards. Cette thèse présente le design et la réalisation d'un système de test intégrable. Le design est composé de deux parties majeures: un générateur de stimulus et un analyseur de signaux.

Un générateur à base de circuits de mémoire est utilisé pour la génération des stimuli. Un tel circuit reproduit périodiquement une portion d'une séquence PDM (Pulse Density Modulation) infinie afin de produire des signaux analogiques. Les circuits sont faciles de conception - ils consistent en une chaîne d'éléments mémoire et un convertisseur N/A à un bit; les circuits sont aussi compacts et robustes de par leur nature numérique. De plus, puisque les signaux analogiques proviennent d'une séquence numérique, ils sont stables et facilement reproduits.

L'analyseur de signaux se concentre sur l'acquisition des réponses en régime constant. Un nouvel algorithme A/N est présenté: la technique Multi-Pass. En prenant avantage des signaux répétitifs, le convertisseur Multi-Pass permet une réalisation compacte et haute-fréquence. Seul un comparateur et un circuit échantillonneur sont suffisants pour acquérir des signaux analogiques. De plus, une nouvelle référence de tension compacte et linéaire est présentée.

Des résultats expérimentaux provenant de deux circuits prototypes confirment la validité de l'approche proposée. Le premier prototype est constitué de composants commerciaux; le second utilise un circuit intégré à l'aide d'un procédé CMOS 0.5 µm. Cette thèse établit les fondations nécessaires pour un système de test pratique et intégrable.

Acknowledgments

First and foremost, I would like to thank my family for their love and constant support. The road has been difficult at times, and often filled with doubt, but their guidance and encouragement allowed me to overcome the many obstacles along the way.

Next, I would like to express my gratitude to Dr. Gordon Roberts. As a teacher, his enthusiasm and energy in the classroom made my choice of study very clear; as a supervisor, he honed my skills, and allowed me to realize the potential that I hold within. The many hours that we spent discussing our work have provided great insight into a broad range of topics. When things seemed to be confused and heading nowhere, I could always count on him to bring me back on track with a renewed optimism.

I would also like to extend a special thanks to Benoit Dufort for his part in designing and implementing the prototype IC. His help was invaluable both during the design process, and during the post-fabrication debugging process.

Next, I would like to thank my friends for helping me throughout the hard times. The fact that we were able to share our problems and our worries somehow helped to ease things.

Finally, I would like to acknowledge NSERC and MICRONET for their funding assistance, and CMC for providing the equipment in the lab, and for making available their IC fabrication resources.

Table of Contents

List of Figures

Chapter 1 - Introduction

1.1 - Motivation

The concept of "Design For Testability" is gaining more influence on microelectronics engineers today. With the ever increasing complexity of mixed-signal integrated circuits, it is imperative that test strategies be developed in conjunction with the design process. Furthermore, with the advent of System-On-Chip strategies that incorporate pre-designed core circuits, there is an even greater need for integrable test solutions [1]. Test bus standards such as the IEEE 1149.1 [2][3] and P1149.4 [4] have been proposed in order to provide a framework in which to access internal nodes. Though a mixed-signal test core would be an ideal complement to the proposed test bus standards, as yet, the biggest challenge still remains the functional testing of the analog portion of a mixed-signal design.

A functional measurement can be performed by exciting an analog circuit under test (CUT) with a signal source, and then extracting relevant parameters from the CUT's output response. Figure 1.1 shows a block diagram of the required test setup. Various stimulus signals (single tone, multi-tone, etc.) can be used to characterize the CUT

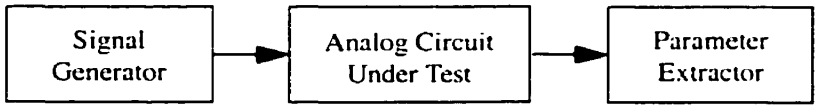

Figure 1.1 Requirements for a functional analog test.

according to a set of metrics (total harmonic distortion, inter-modulation distortion, etc.) as described in [5][6]. Thus, in order to construct a complete test core, two major, integrable components are required: a stimulus generator, and a parameter extractor. Furthermore, in order to produce practical test solutions, these components must be area-efficient and robust (insensitive to process variation), they must be able to function at high speeds (sampling rates), and they must be easy to connect to any existing on-chip circuitry (i.e. require little or no biasing consideration).

1.2 - Memory-Based Signal Generators

A prime candidate for the stimulus generator component of a test core is the memory-based signal generator presented by Hawrysh and Roberts [7]. Such a generator makes use of a delta-sigma modulator [8][9], implemented in software [10], in order to produce a Pulse Density Modulated (PDM) version of an arbitrary stimulus signal - the PDM encoded signal has a bus size of one bit. It then approximates the infinite-length bit-stream by repeating a finite portion of the PDM sequence, using a fixed-length scan chain looped back onto itself. Figure 1.2 shows a block diagram of the suggested memory-based signal generator. Although the finite-length approximation of the original PDM signal leads to a more practical circuit, there is a degradation in the stimulus signal's SNR figure. However, Dufort and Roberts [11] discuss various criteria that can be used to optimize the finite-length PDM sequence, in order to better approximate the desired stimulus signal.

The advantages of such RAM-based generators, in the context of built-in self test, are well documented in [12]. Of particular note are the fact that a simple 1-bit DAC is required at the output of the generator, the fact that very high clocking rates can be

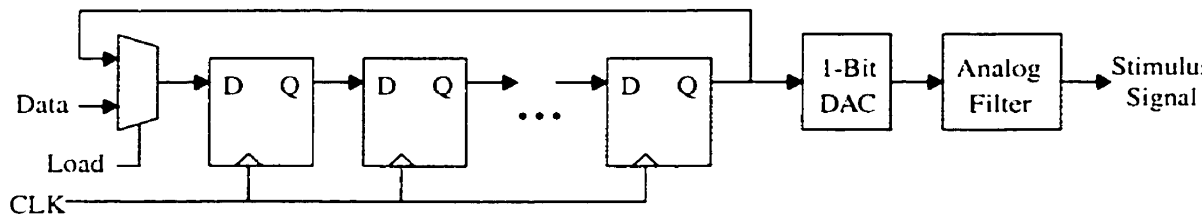

Figure 1.2 A memory-based signal generator.

achieved, and the fact that the design process is simple, and it leads to compact implementations. Furthermore, the analog signal is generated from a digital bit-stream, resulting in a signal that is both stable and repeatable.

1.3 - Parameter Extraction

The detection (or capture) component of a mixed-signal test core represents a more difficult challenge from the point of view of the broad range of possible CUT output responses. These responses can be categorized as being either steady state or transient. Both categories can provide useful information about a CUT - the choice of test depends upon the specific application of the CUT in question.

The work presented here focuses on detecting steady state type responses. Having thus narrowed the specifications of the parameter extraction component, the first solution that comes to mind is an on-chip network analyzer. A network analyzer would essentially capture all of the dynamic behaviour present at the test node (magnitude and phase response), as well as the DC bias level. The requirements for an on-chip network analyzer are shown in Figure 1.3, where an on-chip signal generator (RAM-based) excites the analog CUT, an A/D convertor is used to convert the CUT's output response into digital form, followed by a Fast Fourier Transform (FFT) block that computes the magnitude and phase response of the digitized signal.

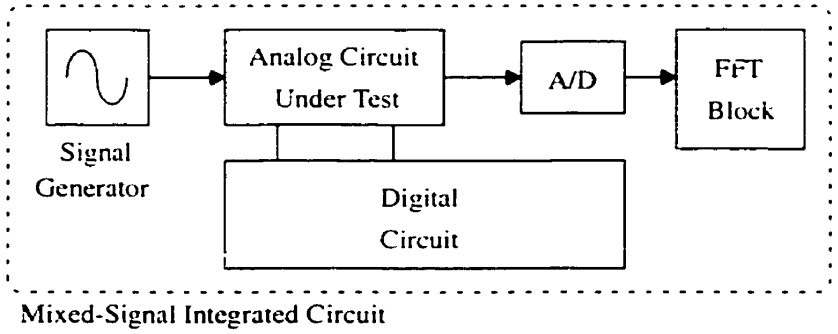

Figure 1.3 An on-chip network analysis approach to parameter extraction.

After a close examination of some good algorithms that are needed to implement the FFT block [13][14], as well as some efficient FFT architectures [15][16], one quickly concludes that it is extremely difficult to satisfy both the area efficiency and high speed operation requirements for a practical test core. The circuitry required to implement an FFT block includes N-bit multipliers (bus size N depends on the desired fractional accuracy of the measurement), that significantly slow down the system, and a RAM (to store partial computational results and to look-up the complex basis functions), that consumes a lot of area. Although it is possible to reduce the area requirements by employing recursive algorithms, the result is a decrease in speed - the tradeoff between area and speed work against the idea of building a network analyzer directly onto the chip.

An easier approach to parameter extraction is to simply move over the computational complexity either to an existing on-chip DSP, or to external software. Figure 1.4 shows the revised scheme for an integrable test core. An integrated A/D is still required in order to avoid distorting the CUT's response during tests at high speeds, and where high resolution is desired. The new extraction scheme also allows for more flexibility with respect to the analysis of the CUT's response; by implementing the FFT in software rather than in hardware, the user can now apply various pre- and post-processing operations, such as windowing and calibration, to all extracted waveforms.

Unfortunately, the A/D convertor is plagued with the same problem as the FFT block: an inherent tradeoff between area and speed [17][18]. On one end, a flash convertor

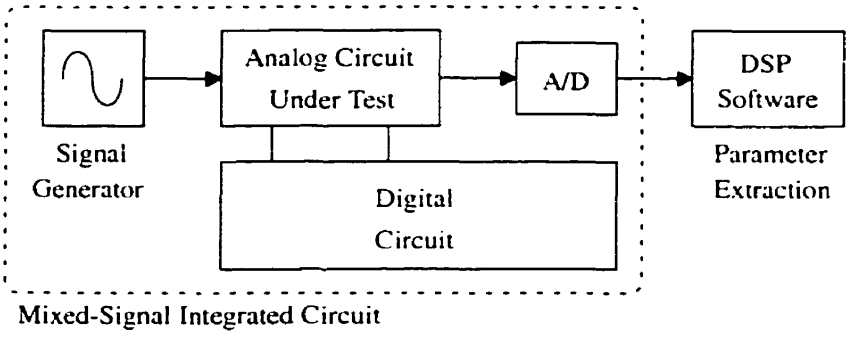

Figure 1.4 Revised parameter extraction scheme requiring less on-chip area.

can run at high speed, but consumes a lot of area; on the other end, a successive approximation convertor is area efficient, but runs at low speed. Lofstrom [19] presents an interesting approach to signal digitization: the Early Capture method - a similar circuit has been patented by Burns [20]. Early Capture takes advantage of the repetitive nature of a CUT's output response in order to decouple the A/D's conversion time from its sampling resolution. Consequently, it is possible to obtain timing resolutions equivalent to very high sampling rates using a slower, area efficient successive approximation convertor.

1.4 - Thesis Outline

The following thesis presents an alternative to the Early Capture method: the Multi-Pass technique. It will be shown that in some cases, it is possible to achieve a faster conversion with the proposed method. The Multi-Pass convertor also makes use of repetitive waveforms in order to obtain an area efficient implementation, without sacrificing high speed performance.

Chapter 2 provides a detailed look at both the Early Capture and Multi-Pass systems. Also presented is the derivation of a criterion for selecting the more optimal method for a given test.

Next, Chapter 3 describes the design details of a prototype IC that implements the proposed test core structure of Figure 1.4. Included are HSPICE simulations of both a bipolar and CMOS comparator, a sample-and-hold circuit, and the complete Multi-Pass convertor. In addition, an integrable, area-efficient, highly linear, and robust programmable voltage reference is presented.

Chapter 4 presents the experimental results obtained from two prototype boards: the first implements the test system using discrete components, and the second makes use of a prototype IC. Various timing and control issues are also discussed.

Finally, Chapter 5 concludes the thesis with a discussion of the experimental results, and some suggestions for future work.

Chapter 2 - Multi-Pass A/D Conversion

2.1 - Introduction

The A/D convertor shown in Figure 1.4 represents a difficult challenge in terms of implementation within the parameters set for the test core: area efficiency, high speed performance, and robustness. However, by focusing on the capture of steady-state responses, a unique advantage presents itself: repetitive waveforms.

As shown in Figure 2.1, a steady-state response will repeat its samples (or pattern) at every Unit Test Period (UTP) amount of time - a UTP is defined as,

$$UTP = N \cdot T_s \tag{2.1}$$

where N is the number of sample points, and T_s is the sampling period.

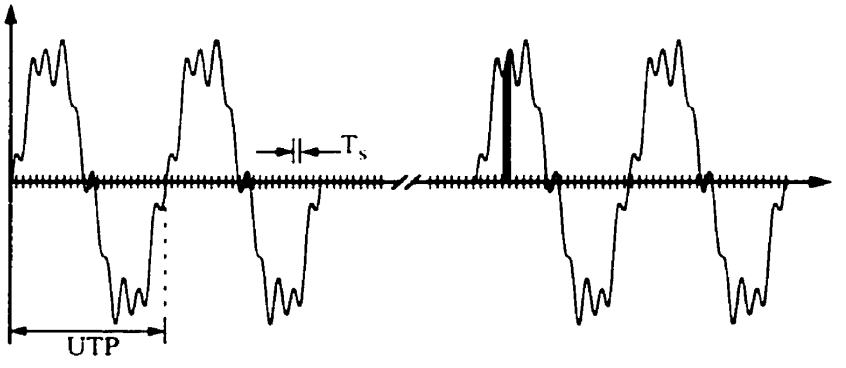

Figure 2.1 A steady-state, repetitive waveform.

Thus, the A/D convertor can sample a point, take as long as it needs to convert the sample into digital form, wait until the next sample point reoccurs, and continue its conversion process. Consequently, by taking advantage of the repetitive nature of a steady-state response, it is possible to decouple the extraction system's sampling resolution from the A/D's conversion time.

The following chapter provides a description of the Early Capture method [19], and the Multi-Pass technique - both systems deal with repetitive signals in order to reduce the on-chip area requirements. In addition, a criterion is presented in order to select the method that is more optimal to a given test.

2.2 - Analog Early Capture

The block diagram for the Analog Early Capture system is shown in Figure 2.2. It consists of a single on-chip comparator, an external D/A convertor (DAC), and a digital controller. The comparator has control over sample time (i.e. it has a built-in sample-and-hold function) allowing it to settle to its output, unaffected by the varying voltage levels (analog waveform) present at its input. The external DAC provides the reference levels with which to compare: the DAC's reference output is the only analog signal present off-chip. The digital controller is comprised of logic circuitry that configures the system as a successive approximation A/D convertor - essentially, the controller uses the external

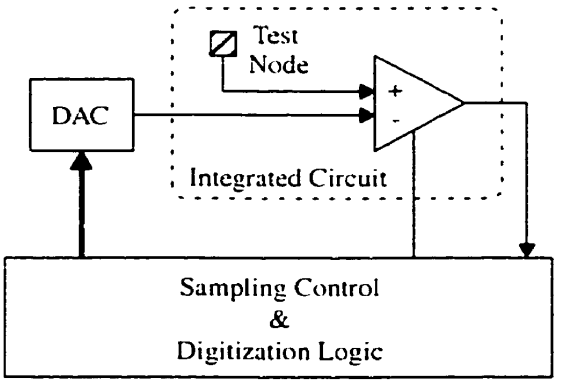

Figure 2.2 Block diagram of the Early Capture system.

DAC to perform a binary search on each of the analog samples, in order to extract the quantized data set.

Since the Early Capture system converts samples at a much slower rate than the desired, high-speed sampling rate, it must be synchronized with a repetitive waveform in order to avoid the effects of aliasing (associated with subsampling). As long as this condition is satisfied, the extracted data set will accurately represent the analog waveform.

2.3 - Multi-Pass A/D Conversion

The proposed Multi-Pass convertor consists of circuitry similar to that of the Early Capture system; however, the way in which it performs the A/D conversion differs. Figure 2.3 shows a block diagram of the Multi-Pass convertor. The sampling resolution T_S, the number of sample points N, and the amplitude resolution B (in bits) govern the timing of

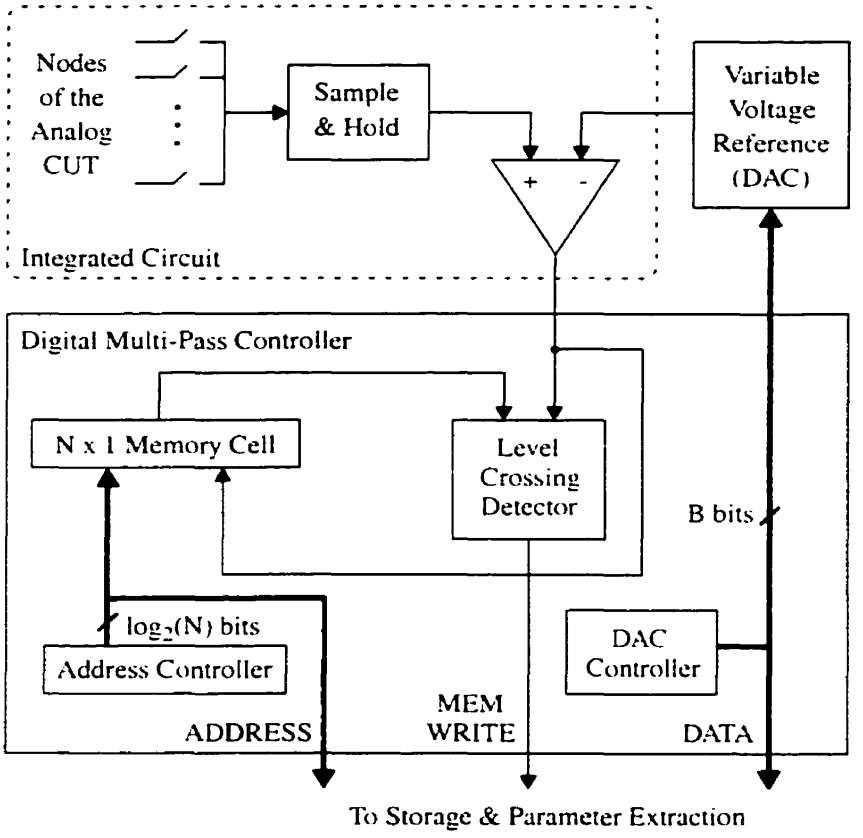

Figure 2.3 Block diagram of the Multi-Pass Convertor.

the Multi-Pass system. The on-chip comparator works in conjunction with the external reference source in order to make multiple comparison passes across the analog waveform. The digital controller sets the voltage reference with which comparisons are made; it starts with the highest level, and proceeds downward through all the possible 2^B levels. For each reference level, the controller monitors and records the state of the comparator's output throughout the unit test period. Every sample point is assigned a unique address in an Nx1 memory cell where the comparator's previous output is stored.

As the convertor proceeds from one reference level to the next, the controller compares the comparator's current output with its previous one, at each of the N sample points. The level at which the comparator's output changes state (i.e. the DAC output just passes under the analog waveform) marks the quantization value of the respective sample point (or sample address). Figure 2.4 illustrates the progression of the top-down comparison in time (for B=4).

During each pass, the comparator outputs a steady stream of 0's and 1's. However, in order to see the digitized signal forming at the output of the comparator, it is necessary to reorganize the information in the bit-stream according to each bit's respective sample

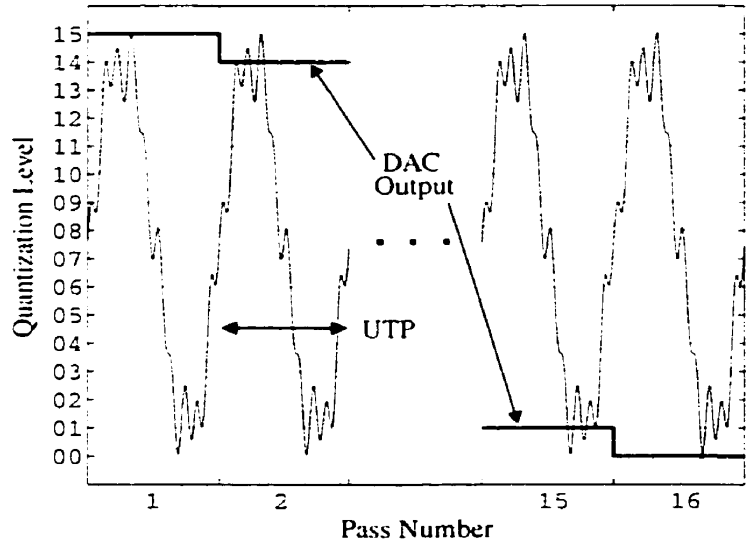

Figure 2.4 Multiple comparison passes along the UTP of a repetitive waveform.

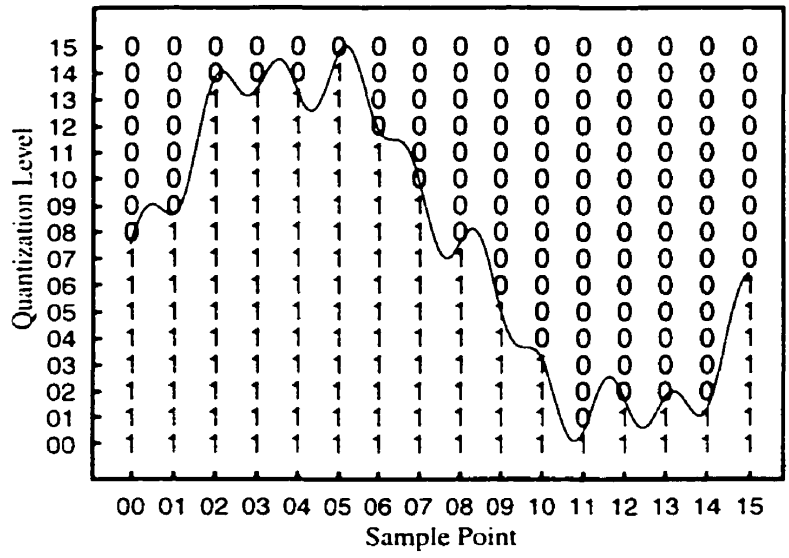

Figure 2.5 Reorganized comparator data showing the formation of the digitized waveform.

position and reference level. Figure 2.5 shows the result of reordering the comparator output for the waveform shown in Figure 2.4 and using N=16. It is interesting to note that all the bits that fall below the analog waveform are 1's. This makes sense because as the reference level passes below the analog waveform, a change of state occurs at the comparator's output.

Figure 2.6 shows a time-overlaid image of the UTP where the solid, horizontal lines represent the DAC's output for each of the 2^B passes. The dots indicate the closest reference levels under the analog waveform, corresponding to the transition levels of the comparator's output, and thus, the quantization values of the sample points.

Note that although the Multi-Pass controller is described here as a digital circuit, it can also be implemented in software. The digital nature of the controller allows it to be implemented either internally, with an existing on-chip DSP, or externally, with computer software (where the data is obtained from any common digital capture system).

Two important concerns arise regarding the limitations of the Multi-Pass convertor: sampling and amplitude resolution. From Figure 2.3, it is clear that these limitations will be due to the on-chip sample-and-hold circuit, and the comparator. The

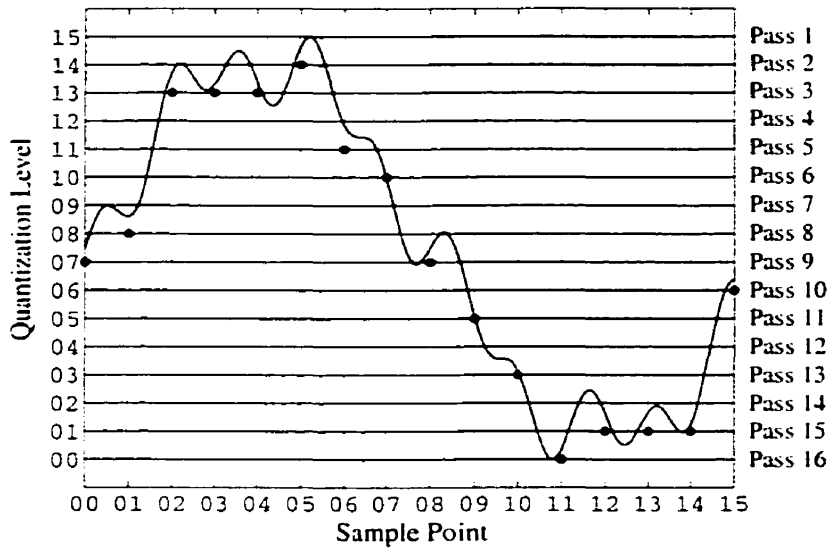

Figure 2.6 Time-overlaid image of the UTP showing the comparator's transition level for each sample point.

amplitude resolution will depend on the lowest voltage difference that the comparator is able to distinguish. Figure 2.6 suggests that the sampling resolution will be limited by the comparator's switching speed - this is implied by the fact that the comparator provides N comparison outputs in one pass (i.e. one UTP), and thus, sets the maximum sampling rate.

However, rather than capturing all N comparisons, for a given reference level, in one pass, it is possible to do so in say ρ passes by staggering the sampling instances. In other words, for each sample that is compared, the convertor skips over N/ρ samples (for that particular pass), but returns to compare those missing samples in the following passes. In this manner, all N samples are compared with the current reference level in ρ passes.

Although the result is a conversion that is less demanding on the comparator, it requires more comparison passes to complete - a total of $\rho \cdot 2^B$ passes are needed. It is important to note here that even though there is less demand on the comparator, the sample-and-hold circuit must still function at the desired, high speed sampling rate ($1/T_S$).

2.4 - Optimal Method Selection

Comparing the circuit setup shown in Figure 2.2 with the one in Figure 2.3, it is evident that the critical, analog circuitry for both the Multi-Pass and Early Capture methods is essentially the same. Consequently, in order to determine the method that is better suited to a given application, it is necessary to derive a selection criterion based on required conversion times rather than area efficiency.

It was shown in the previous section that $\rho \cdot 2^B$ passes are required to complete a conversion using the Multi-Pass method. Each pass must last the duration of a UTP in order to maintain sample coherency between successive passes. An additional setup time is required for the DAC to switch to the next reference level. Once again, this setup time must correspond to some multiple σ of a UTP. Since there are 2^B reference levels to switch to, the overall time devoted to DAC setup is $\sigma \cdot 2^B$ UTP's. The total conversion time T_{MP} for the Multi-Pass technique is thus given by,

$$\begin{aligned} T_{MP} &= (\rho \cdot 2^B\ UTP) + (\sigma \cdot 2^B\ UTP) \\ &= (\rho + \sigma) \cdot 2^B \cdot UTP \\ &= (\rho + \sigma) \cdot 2^B \cdot N \cdot T_s \end{aligned} \qquad (2.2)$$

As mentioned previously, the Early Capture method makes use of a successive approximation convertor to digitize each of the N samples of a UTP. Such a convertor has associated with it a certain decision time T_{dec} for each step of its binary search. Given that a binary search through 2^B levels requires at most $\log_2(2^B) = B$ steps, the conversion of one sample requires $B \cdot T_{dec}$ seconds (accounting for the worst case search scenario). Next, since the sample points of a repetitive signal reappear every $N \cdot T_s$ (UTP) seconds, the Early Capture system must wait until the completion of the corresponding UTP in which the conversion of a sample ends. In other words, the effective conversion time of one sample is τ UTP's, where τ is given by,

$$\tau = Ceil\left[\frac{B \cdot T_{dec}}{UTP}\right] = Ceil\left[\frac{B \cdot T_{dec}}{N \cdot T_s}\right] \quad (2.3)$$

and *Ceil* is defined as the next largest integer. The total time T_{EC} required to convert all N samples with the Early Capture method is then given by,

$$T_{EC} = N \cdot (\tau \cdot UTP) = N \cdot \tau \cdot (N \cdot T_s) = \tau \cdot N^2 \cdot T_s \quad (2.4)$$

A selection criterion η can be obtained by taking the ratio of Eqns (2.2) and (2.4),

$$\eta = \frac{T_{MP}}{T_{EC}} = \frac{(\rho + \sigma) \cdot 2^B \cdot N \cdot T_s}{\tau \cdot N^2 \cdot T_s} = \frac{(\rho + \sigma) \cdot 2^B}{\tau \cdot N} \quad (2.5)$$

where the two methods are equivalent when $\eta=1$, and the Multi-Pass technique is more optimal when $\eta<1$.

It is not unreasonable to assume that in most cases, the Multi-Pass convertor will be able to capture all N comparisons in one pass ($\rho=1$), that it will be able to switch to the next reference level in one UTP ($\sigma=1$), and that the Early Capture system will be able to convert a sample in one UTP ($\tau=1$). Using these (ρ, σ, τ) parameters, Figure 2.7 shows the regions where, given the test parameters B and N, one method produces a faster conversion than the other. The curve that separates the two regions represents the equivalency line $\eta=1$.

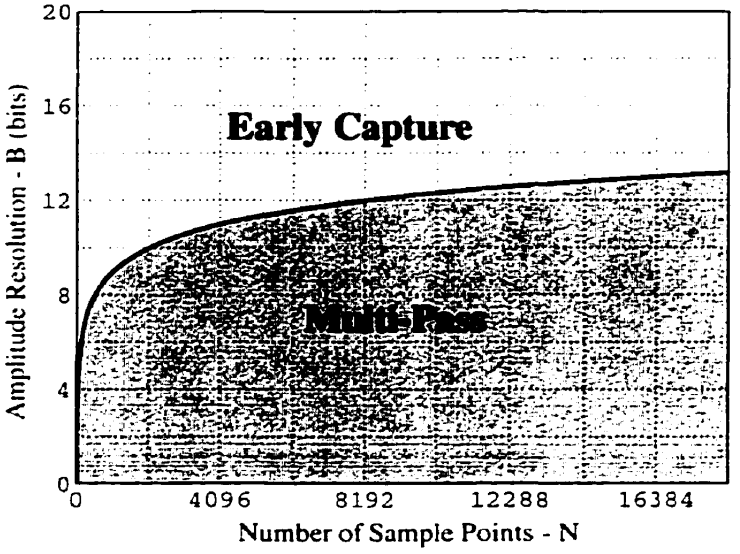

Figure 2.7 Optimal method regions for ρ=1, σ=1, τ=1.

2.5 - Conclusion

The Multi-Pass technique presented in this chapter makes use of a single on-chip comparator to extract analog waveforms. The on-chip circuitry is similar to that of the Early Capture method, however, the A/D algorithms differ. A selection criterion based on the total conversion times has been derived in order to choose the method that is more optimal. Once the parameters ρ, σ, τ, for a given system, have been specified (according to the limitations of the fabricated design), the test parameters B and N determine the scheme that is most advantageous. Furthermore, since both systems make use of a digital controller that can be implemented using DSP software (either on-chip or external), given the test inputs B and N, the controller can automatically switch to the more optimal scheme.

Chapter 3 - Test Core Design

3.1 - Introduction

The following chapter provides a detailed look at the design of a prototype test core. The objective is to integrate both a signal generator and a waveform extractor onto a single IC, in order to implement the test scheme shown in Figure 1.4.

As discussed in Chapter 1, the best candidate for the signal generation part is a memory-based generator - all that is needed is a simple daisy-chain of D-type flip-flops as shown in Figure 1.2. A 1-bit DAC at the output of the daisy-chain is not absolutely necessary - its purpose is to enhance the SNR of the PDM output. Consequently, in order to simplify any debugging that might be required after the first fabrication attempt, it was omitted from the design. On the other hand, an analog filter is required in order to remove the noise content of the PDM signal. Once again, in order to simplify the design, it was decided to make use of an external filter.

Figure 3.1 shows a block diagram of the prototype IC. The design includes the signal generator described above, the on-chip components of the Multi-Pass convertor, a PLL, and a few switches. Switch pair SW1 selects the analog input that is fed to the Multi-Pass convertor - it allows for three possible input configurations: the first consists of connecting the output of the daisy-chain to both a driving-point impedance and the switch pair SW2; the second consists of passing the PDM stream through a regular, two-port

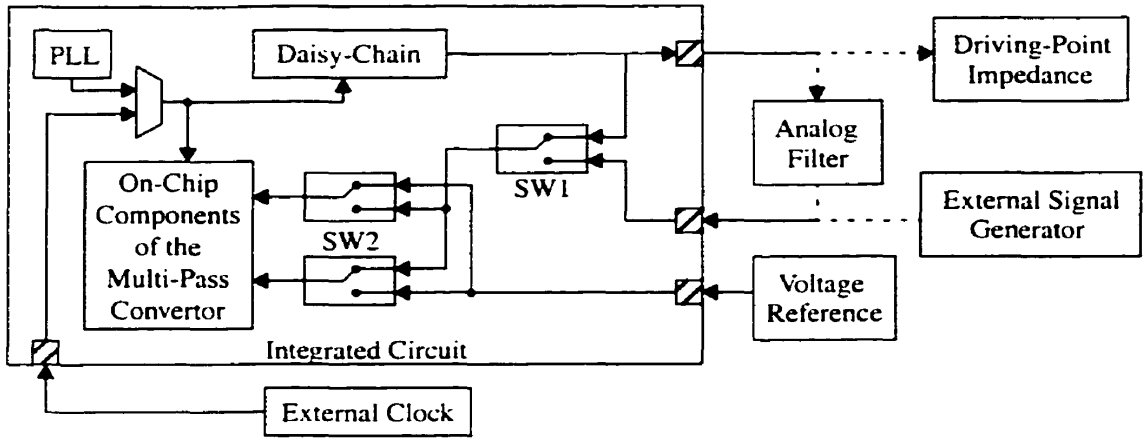

Figure 3.1 Block diagram of the prototype test core IC.

analog filter before connecting it to SW2; the third bypasses the on-chip generator altogether and connects an external signal source to SW2. These configurations make it possible to debug the signal generator and waveform extractor separately, as well as to combine them to form a complete test system.

Once the input configuration has been selected, switch pair SW2 selects the polarity of the comparison - an offset correction scheme, involving the swapping of the inputs to the comparator, is presented in Section 3.2.

An on-chip PLL provides the system clock with which the signal generator runs. Note that the Multi-Pass convertor uses the same clock signal in order to synchronize itself with the signal generator's output. In addition, an external clock input is also available in case the PLL fails to function.

The remainder of this chapter focuses on the on-chip components of the Multi-Pass convertor. A detailed analysis of the comparator is presented, including an offset correction scheme. Also shown is a simple sample-and-hold circuit that implements an input mapping function, allowing the capture of full-scale inputs. Next, a method of eliminating the effects of the sample-and-hold's non-linear transfer characteristic is presented. Finally, HSPICE simulations are used to verify the functionality of each component, and the complete Multi-Pass system.

3.2 - Comparator Design

The comparator is a critical component in both the Multi-Pass and Early Capture systems. As mentioned in Section 2.3, the upper limit to the A/D convertor's amplitude resolution is set by the smallest voltage difference that the comparator is able to distinguish. Furthermore, from the analysis shown in Section 2.4, it is clear that the switching characteristics of the comparator have a significant effect on the overall conversion time.

One way to describe a comparator is to say that it takes a voltage difference and amplifies it to power supply levels (TTL/CMOS levels). In this manner, the comparator can be treated as a differential amplifier, and it follows that the amplitude resolution will be equivalent to the amplifier's differential gain - this is evident from the fact that the gain sets the smallest voltage difference that can be amplified to digital levels. Therefore, all that is needed to build the comparator is an amplifier circuit with enough gain to meet the required amplitude resolution. It is also important to keep in mind that the faster the comparator, the shorter the conversion time (particularly for high speed applications).

In order to obtain the best compromise between gain and speed, multiple differential-pair gain stages can be used to implement an amplifier circuit. Figure 3.2 shows a bipolar version of such a design. Depending on the available power supply levels, it may be necessary to add level shifters at the output node in order to produce TTL compatible signals.

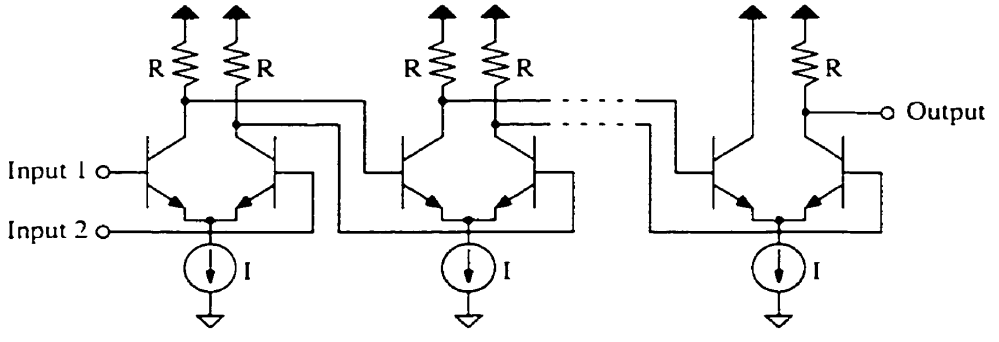

Figure 3.2 A high speed, bipolar comparator.

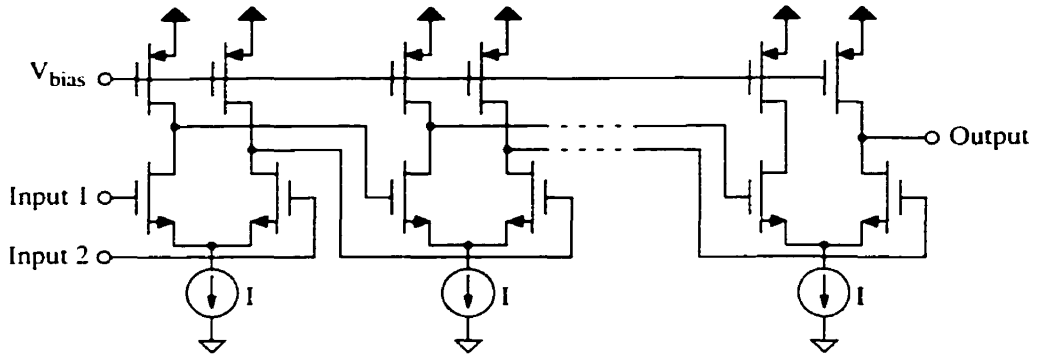

Figure 3.3 A high speed, CMOS comparator.

The CMOS version of the proposed comparator design is shown in Figure 3.3; note that in order to reduce the area requirements of the comparator, the load resistors R shown in Figure 3.2 are replaced by PMOS transistors. The p-devices' channel resistances, and hence the gain per stage, can be tuned via the bias voltage V_{bias}. This allows the option of lowering the gain per stage, and adding additional stages to compensate for the loss in the overall gain, in order to trade-off area (slight increase) for speed.

An important problem that arises with high resolution A/D convertors is input offset voltage. One method of reducing the effects of offset voltage is to use a double conversion scheme consisting of performing a complete conversion, swapping the inputs to the comparator (using switches), and then doing a second conversion. By taking the average of the two digitized waveforms, the effects of the input offset voltage are significantly reduced. Figure 3.4 shows the modified on-chip setup that allows the swapping of the comparator inputs. Unfortunately, the double conversion scheme takes twice the time to execute, and should only be used for tests that require accurate DC information.

3.3 - Comparator Simulation

In order to determine the maximum operating speeds of the proposed comparator designs, the following worst case switching scenario was simulated: the largest possible

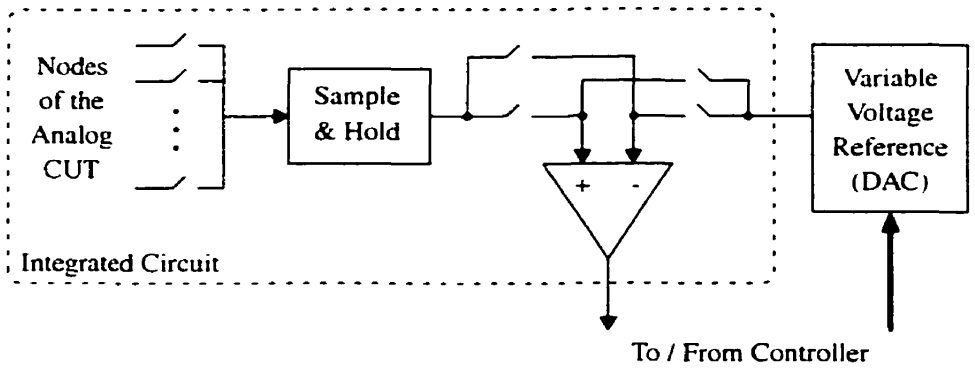

Figure 3.4 Modified on-chip circuit that allows swapping the inputs to the comparator.

input signal followed by the smallest possible input signal of opposite polarity. This sequence determines the time it takes the comparator to switch to an opposite output level, starting from a saturated state and using the smallest input to move over all the charge - it sets the minimum hold time.

Device models for a 0.8 µm BiCMOS process (Appendix A.1) were used to simulate the bipolar design shown in Figure 3.5. Refer to Appendix A.2 for the HSPICE netlist. It was found that five gain stages (including the output stage) are required to detect

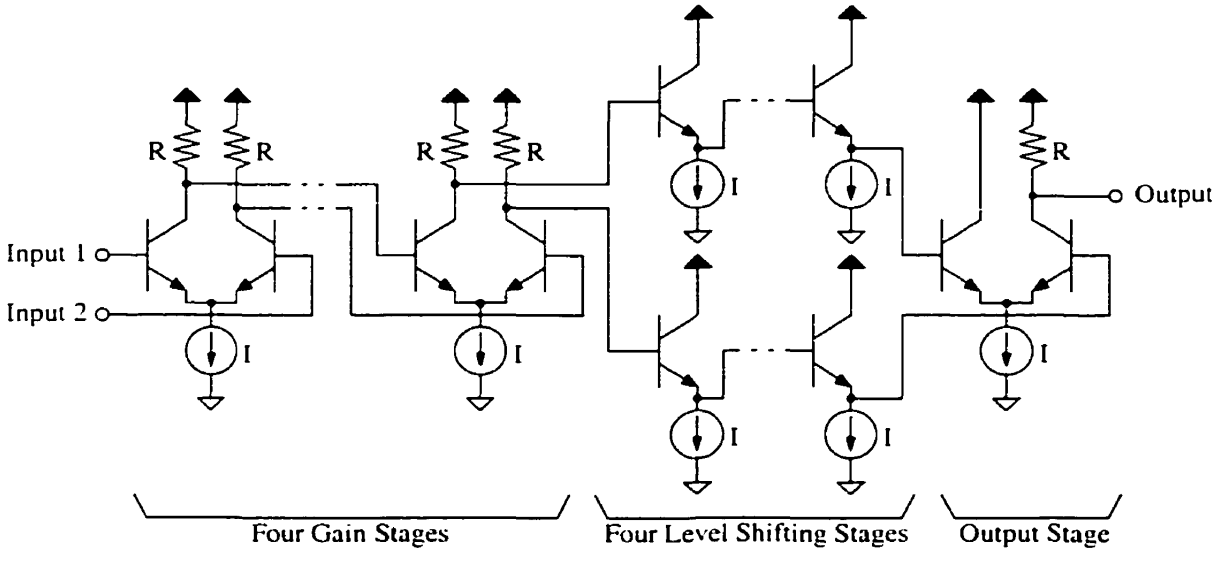

Figure 3.5 Simulated bipolar comparator circuit.

a minimum voltage difference of 2.5 mV. Note that four level shifting stages are needed in order to drop the output stage's input common-mode level - this is necessary for the output node to be able to produce TTL compatible signals. Figure 3.6 shows the results of simulation where the power supplies are 0 V and 5 V, and the input is defined as ranging from 1 V to 4 V. The figure is divided into three parts: the plot on top shows a full view of the input signals versus time; the second plot shows a magnification about the reference input in order to highlight the dynamic behaviour of the analog waveform; the last plot shows the output of the comparator as it tracks the reference crossing points of the input waveform. Note that the ratio of the input voltage range (3 V) to the smallest detectable voltage difference (2.5 mV) represents an amplitude resolution of 10 bits. Figure 3.6 also shows that the minimum hold time is 3.33 ns, corresponding to a maximum operating frequency of 300 MHz.

Next, device models for a 0.5 μm CMOS process (Appendix A.3) were used to simulate the CMOS design shown in Figure 3.7. Refer to Appendix A.4 for the HSPICE netlist. In this case, the power supplies are 0 V and 3.3 V, the input can range from 1.3 V to 3.3 V, and nine gain stages are needed to detect a minimum voltage difference of 1.5 mV. The circuit also includes three output buffer stages in order to drive the digital circuitry

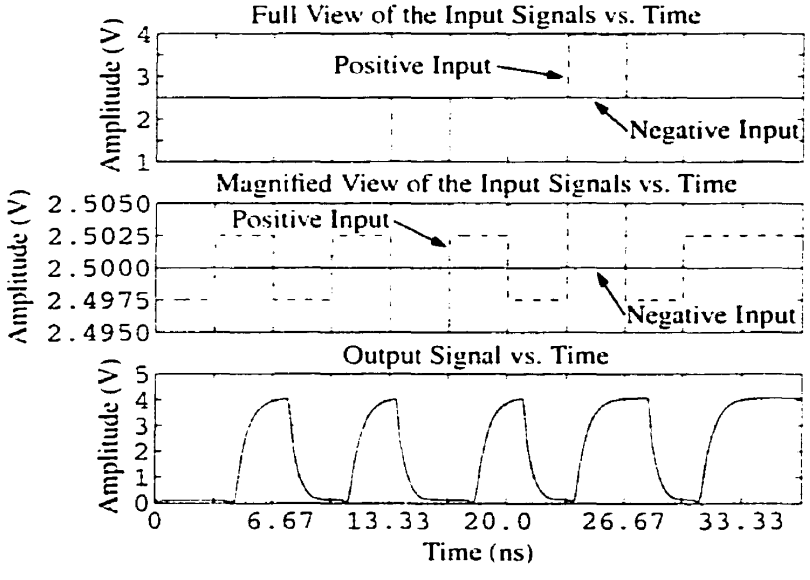

Figure 3.6 Simulation results for the bipolar comparator.

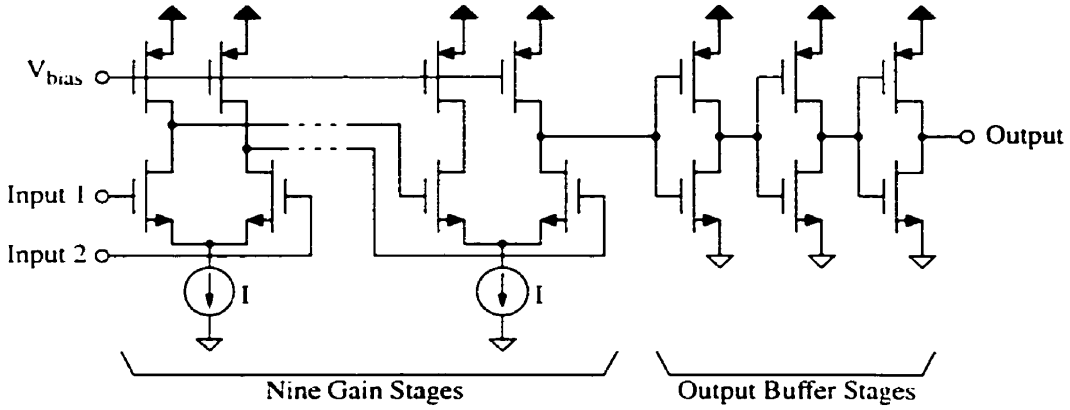

Figure 3.7 Simulated CMOS comparator circuit.

that follows: three different invertor stages are used - starting from a small size and growing towards the output - in order to minimize the effects on the comparator's speed performance (affected by capacitive loading). The simulation results in Figure 3.8 show that the circuit can operate at frequencies up to 667 MHz with an amplitude resolution of 10 bits.

It is important to note from these results that the greater the number of gain stages, the greater the delay in the output. Consequently, the output must be sampled with an

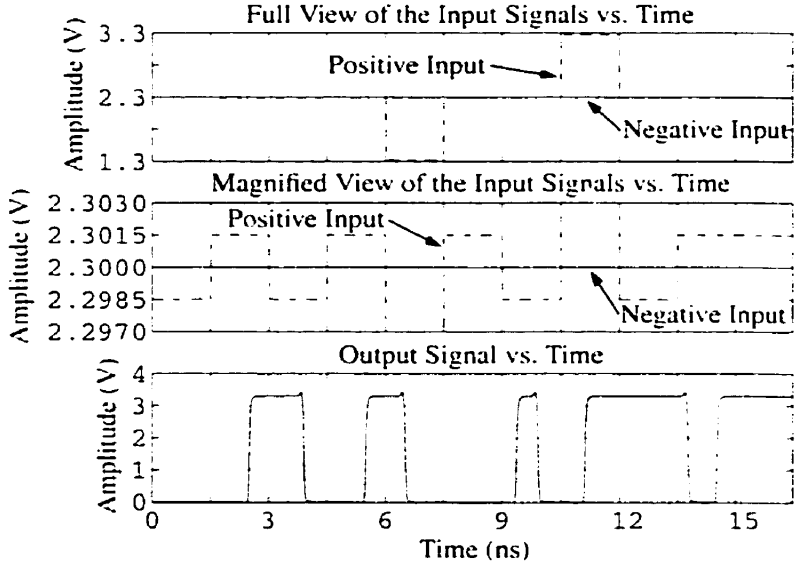

Figure 3.8 Simulation results for the CMOS comparator.

appropriately synchronized clock signal in order to avoid misinterpreting the comparison information.

3.4 - Sample-and-Hold Design

An important part of the Multi-Pass convertor is the on-chip sample-and-hold circuit. It is this circuit that dictates the effective sampling resolution of the system. Section 2.3 describes how the demand on the comparator can be reduced by performing the N comparisons, for a given reference level, in ρ passes rather than in one pass. However, although this option renders the comparator design more simple, the requirements for the sample-and-hold circuit increase. The longer the comparator needs to make a decision, the longer the sample-and-hold circuit must hold its value. Therefore, the challenge is to design a sample-and-hold circuit that can sample at very high rates (in the GHz range), and that can hold its value for roughly an order of magnitude longer than its sampling period.

A second requirement for the sample-and-hold circuit is that it map the V_{SS} to V_{DD} voltage range into the input range of the comparator. This will allow the Multi-Pass system to detect the full range of possible input signals. It will also remove the need for special input biasing circuitry, thereby satisfying one of the key properties for a practical test core.

Figure 3.9 shows a simple MOS circuit that is characterized by a very low leakage current while the input switch is turned off. The reason for choosing p-type devices is that

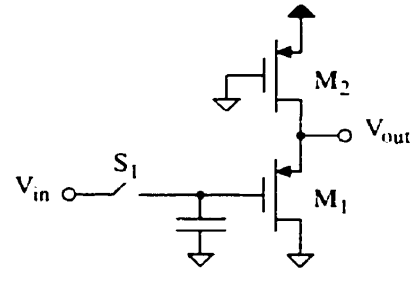

Figure 3.9 A simple sample-and-hold circuit.

n-type devices were used to design the differential-pairs - the output voltage range of the circuit in Figure 3.9 lies in the input voltage range of the proposed comparator designs.

Although the buffer circuit in Figure 3.9 produces an output in the desired range, it does not have a quasi-linear relationship between its input and output voltages across the full range of possible input signals. The reason is that as the input voltage V_{in} increases, transistor M_1 ultimately shuts off (when V_{in}-V_{out}>V_{th}), and the output voltage becomes V_{DD}. One way to solve this problem is to scale down the input to the buffer such that it falls in the buffer's quasi-linear region. This can be done by means of charge sharing between identical buffer circuits as shown in Figure 3.10. The sample-and-hold circuit shown in this figure requires two sampling periods to capture an input signal. During the first period, switches S_1 and S_3 are closed, and S_2 is opened - S_1 charges capacitor C_1 to the input voltage, and S_3 clears the charge on capacitor C_2. During the second period, S_2 is closed, and S_1 and S_3 are opened - S_2 divides the charge between capacitors C_1 and C_2.

It is important to note here that although the proposed sample-and-hold design solves the problem of the full-scale input voltage range, it also reduces the Multi-Pass system's amplitude resolution because of the down-scaling of the input signal. Furthermore, because the input/output transfer characteristic is neither linear nor has a unity slope, the on-chip circuit setup shown in Figure 3.4 can no longer be used - the extraction system would simply produce a digitized version of the distorted waveform. A simple fix to this problem is to pass the DAC's output through an identical sample-and-hold circuit before connecting it to the comparator, as shown in Figure 3.11. In this

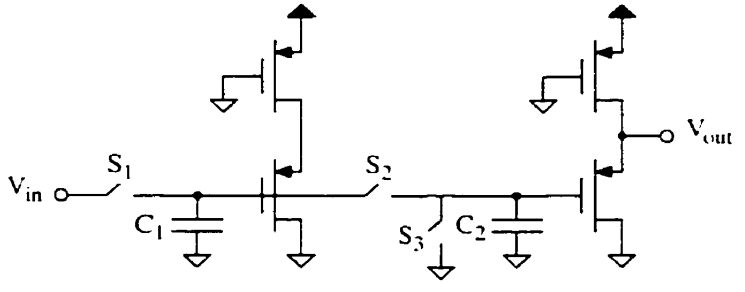

Figure 3.10 A sample-and-hold circuit with a full-scale input voltage range.

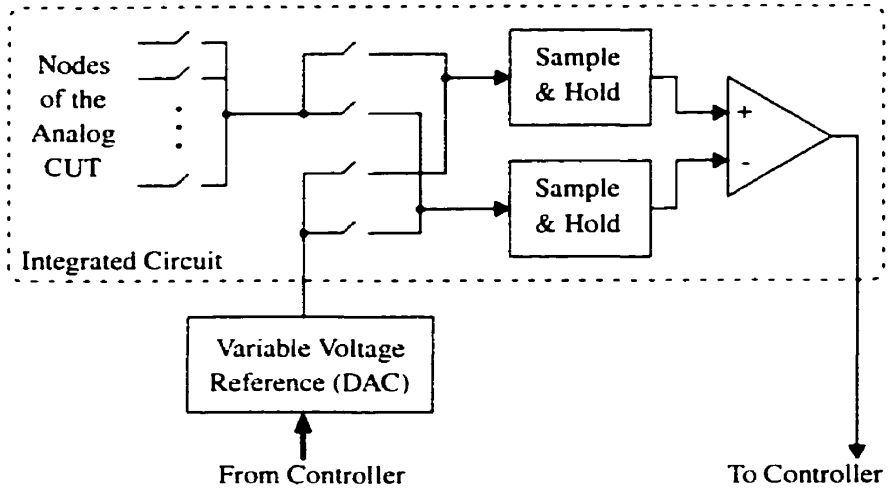

Figure 3.11 On-chip circuit setup incorporating the proposed sample-and-hold design, and the offset correction scheme.

manner, both the CUT's output signal and the DAC's reference signal will go through a similar mapping function.

Another modification shown in Figure 3.11 is the placement of the switches - they are now connected before the sample-and-hold circuits. Using the same input swapping scheme described in Section 3.2, the combined offset error from the comparator and sample-and-hold circuits can be eliminated.

3.5 - Sample-and-Hold Simulation

Device models for a 0.5 μm CMOS process were used to simulate the sample-and-hold circuit described in the previous section. Refer to Appendix A.5 for the HSPICE netlist. A staircase input was used in order to sweep through the full V_{SS} to V_{DD} range. Figure 3.12 shows the results of simulation where the power supplies are 0 V and 3.3 V, the input step size is 5 mV, and the sampling rate is 500 MHz. Note that although the results show a one-to-one mapping between the input and output voltages, the curve is not a straight line (i.e. the transfer characteristic is not linear). However, since both the analog waveform and the reference signal go through the same mapping function, and since the

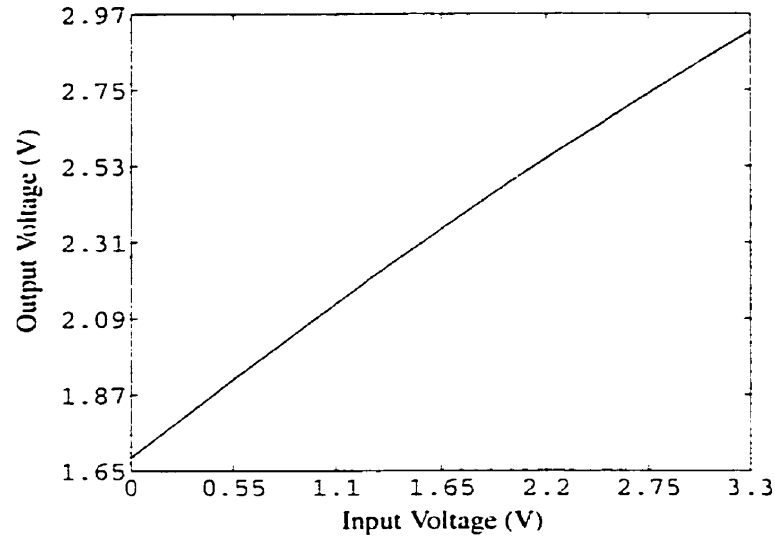

Figure 3.12 Input/output transfer characteristic of the proposed sample-and-hold circuit.

system only looks at the difference between the two, the linearity of the sample-and-hold circuit is not a major concern.

3.6 - Sample-and-Hold Controller Design

The issue of timing is the most critical in the Multi-Pass system. First, the sample-and-hold circuitry must receive two separate pulse signals in order to store an analog sample. Referring back to Figure 3.10, a pulse signal P_1 is needed to close the switches S_1 and S_3 (to charge C_1 and clear the charge on C_2), and a pulse signal P_2 is needed to close the switch S_2 (to divide the charge between C_1 and C_2). Next, the comparator must be allowed enough time to settle to its output. Finally, a means of tracking the position of the samples in time is required.

There is a simple solution to the timing problem; however, it is necessary to first define the read time T_R for one sample (i.e. the time it takes to sample a point and compare it to the reference level) as,

$$T_R = (W + 2) \cdot T_s \tag{3.1}$$

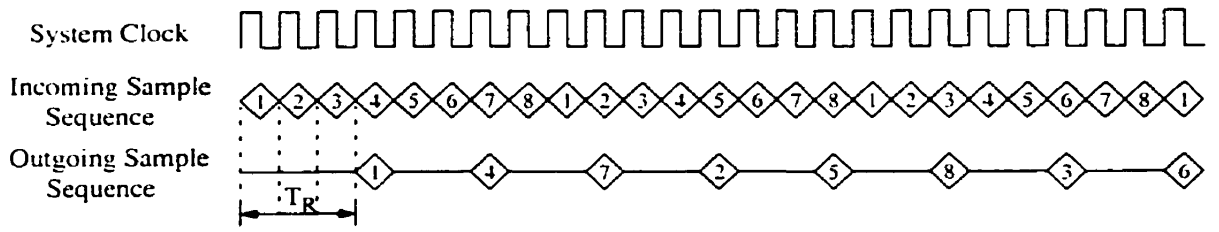

Figure 3.13 A synchronous sampling solution for the Multi-Pass convertor; output sample order shown for N=8, W=1.

where W is the number of wait states required for the comparator to settle to its output, and two sampling periods are required by the sample-and-hold circuit.

By setting W such that T_R represents a number of sampling periods that is relatively prime with respect to the number of sample points N, it is possible to obtain a reading on all of the N samples without the use of complex timing circuitry. Consider the example shown in Figure 3.13 where N=8 and W=1; as shown in the figure, three clock cycles are required to sample and compare the first sample point. At the instant the data for sample "1" is latched, the next available sample point (sample "4") is read and compared to the reference level. By following the sampling pattern outlined in Figure 3.13, the output data sequence is: "1", "4", "7", "2", "5", "8", "3", "6". Thus, although the samples are read in a shuffled manner, no samples are excluded, and the data can be obtained at regular (T_R) clocking intervals (i.e. synchronous data output).

The following subsections describe the various components that are need to implement the sample-and-hold controller.

3.6.1 - Non-Overlapping Pulses

The correct operation of the sample-and-hold circuit requires that the pulse signals P_1 and P_2 do not overlap; Figure 3.14 shows the desired waveforms where an overlap is avoided by delaying the rising edges. In order to produce such waveforms based on pulses originating from edge-triggered Flip-Flops (DP_1, DP_2), the circuit shown in Figure 3.15 can be used for each pulse signal. Note that the circuit includes both the signal P_x and its complement $\overline{P}_x$ (where x=1,2); complementary pulse signals are required for the

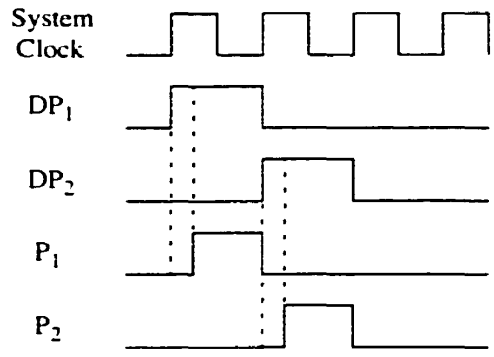

Figure 3.14 Non-overlapping pulse signals.

transmission gates that implement the switches in the sample-and-hold circuit. Due to an extra invertor stage required for P_x, the complementary pulses do not overlap in time (also necessary for correct operation). One way to alleviate this is to adjust the number and size of the invertors connected after the NAND gate, in order to delay $\overline{P}_x$ long enough to overlap with P_x; in this case, one of the invertors in the signal path of $\overline{P}_x$ is replaced by a smaller one (hivx2), in order to slow down the signal through that path. The digital circuit shown in Figure 3.15 makes use of the standard cells (h-cells) for the 0.5 μm CMOS process available in the CADENCE CAD tool.

3.6.2 - Pulse Generator with Wait States

As described previously, the Multi-Pass system requires a pulse generator that produces the two sampling pulses DP_1 and DP_2 (Figure 3.14), and allows the comparator enough time to settle to its output. Although simulation would show exactly how much time the comparator needs, there is always an uncertainty associated with fabrication.

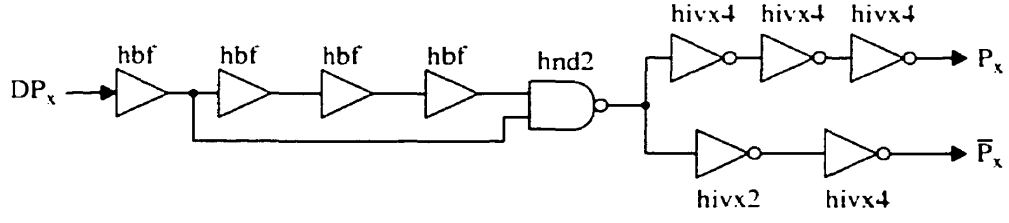

Figure 3.15 The circuit used to produce non-overlapping pulse signals.

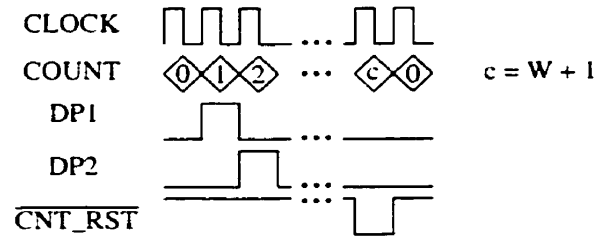

Figure 3.16 Timing diagram for the pulse generator.

Consequently, it is desirable to have a pulse generator that can accommodate a user-selectable number of wait states (W).

One method of building the pulse generator is based around a 4-bit counter, where each count represents a sampling period (T_s). Note that after using two periods (out of the 16 available with the counter) to sample and hold, there are 14 periods available for waiting. The timing diagram for the pulse generator is shown in Figure 3.16. The pulse DP_1 is generated when the count reaches one, and DP_2 is generated when the count reaches two. Then, the controller counts until it reaches W+1, at which time, it lowers the counter reset line $\overline{\text{CNT\_RST}}$, indicating that the counter should reset on the next rising-edge of the clock.

Figure 3.17 shows the circuit diagram of the pulse generator; once again, standard cells (h-cells) are used. Some simple logic is used to detect a count of zero and one, and then generate (after one clock delay) the pulse signals DP_1 and DP_2 respectively. In order to detect when the count has reached the user-selected number of wait states (W), a bit-by-bit XOR operation is required, followed by an OR gate (a match is indicated by lowering the output to '0'). The signal is then delayed one clock cycle in order to obtain the desired $\overline{\text{CNT\_RST}}$ pulse at W+1. The generator also includes a $\overline{\text{SYSTEM\_RST}}$ line that is connected to the main reset input of the Multi-Pass system.

3.6.3 - High-Speed Counter

A key component of the pulse generator is the 4-bit counter; it must be able to operate at the high-speed sampling rate of the system. Consequently, it is not possible to

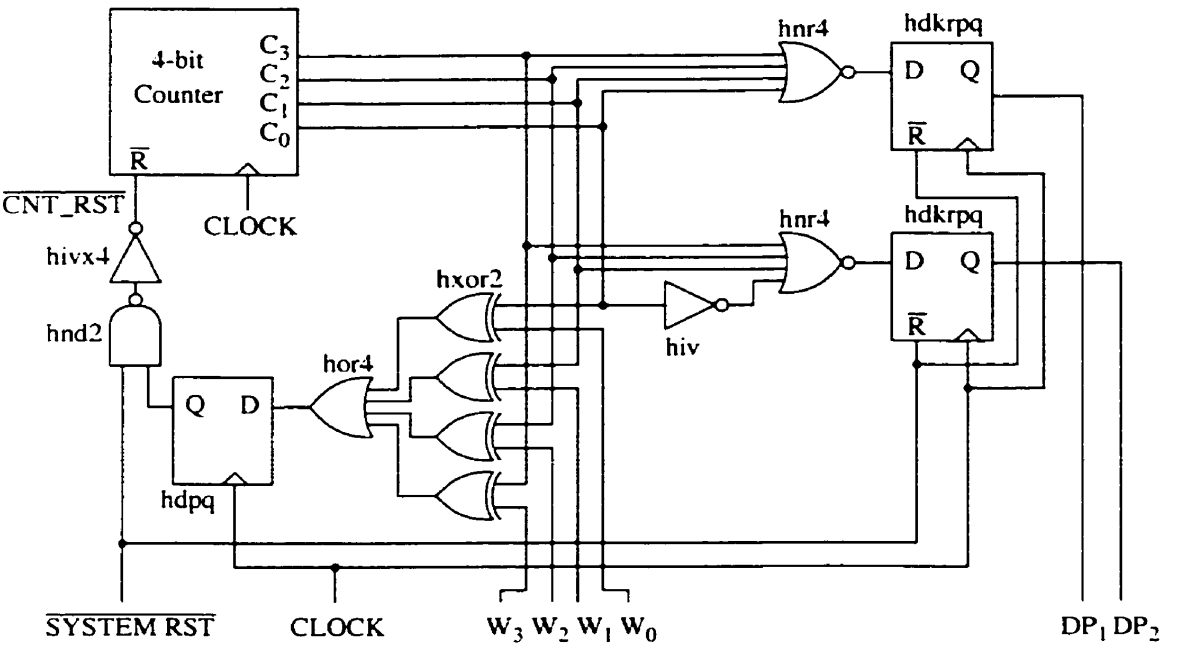

Figure 3.17 Circuit diagram of the pulse generator.

implement the counter using a ripple structure [21]. Instead, the counter must be implemented with a register and an adder as shown in Figure 3.18.

In order to obtain high-speed performance from the adder, a carry lookahead structure must be used [21]. The equations that describe such an adder (with a 4-bit bus size) can be written as,

$$x_0 = (a_0 \cdot b_0) + (a_0 + b_0) \cdot c_{in} \tag{3.2}$$

$$x_1 = (a_1 \cdot b_1) + (a_1 + b_1) \cdot (a_0 \cdot b_0) + (a_1 + b_1) \cdot (a_0 + b_0) \cdot c_{in} \tag{3.3}$$

$$\begin{aligned} x_2 = {} & (a_2 \cdot b_2) + (a_2 + b_2) \cdot (a_1 \cdot b_1) + (a_2 + b_2) \cdot (a_1 + b_1) \cdot (a_0 \cdot b_0) \\ & + (a_2 + b_2) \cdot (a_1 + b_1) \cdot (a_0 + b_0) \cdot c_{in} \end{aligned} \tag{3.4}$$

$$s_0 = (a_0 + b_0) \oplus (a_0 \cdot b_0) \oplus c_{in} \tag{3.5}$$

$$s_1 = (a_1 + b_1) \oplus (a_1 \cdot b_1) \oplus x_0 \tag{3.6}$$

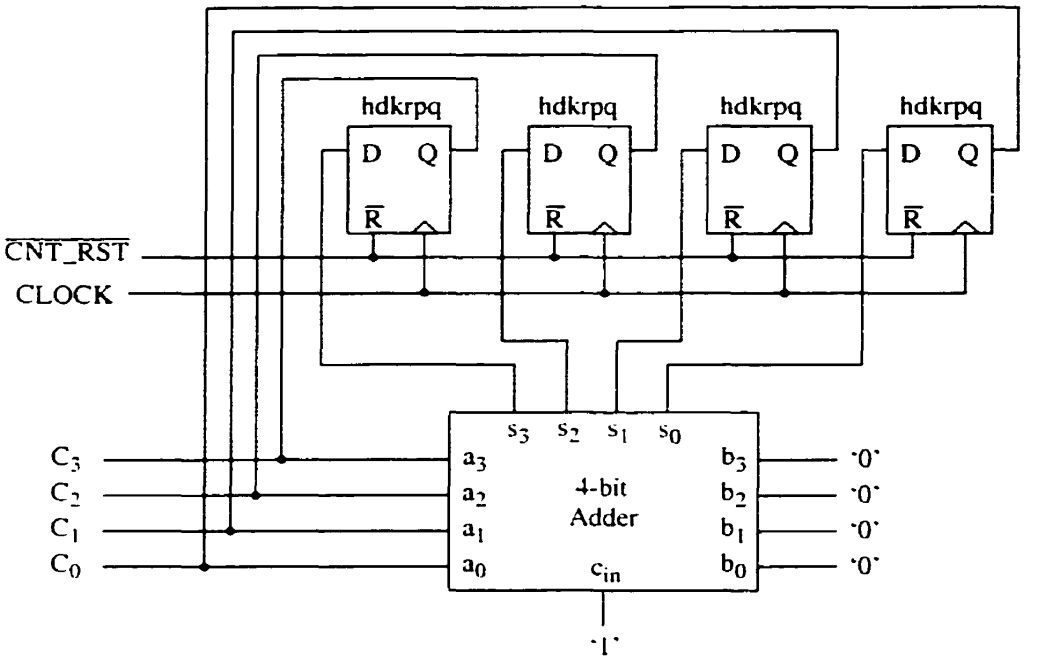

Figure 3.18 Circuit diagram of an adder-based counter.

$$s_2 = (a_2 + b_2) \oplus (a_2 \cdot b_2) \oplus x_1 \tag{3.7}$$

$$s_3 = (a_3 + b_3) \oplus (a_3 \cdot b_3) \oplus x_2 \tag{3.8}$$

Next, Eqns (3.2)-(3.8) can be simplified by substituting b_3=b_2=b_1=b_0=0 and c_{in}=1 (from Figure 3.18) to obtain the following,

$$x_0 = a_0 \tag{3.9}$$

$$x_1 = a_1 \cdot a_0 \tag{3.10}$$

$$x_2 = a_2 \cdot a_1 \cdot a_0 \tag{3.11}$$

$$s_0 = a_0 \cdot V_{DD} \tag{3.12}$$

$$s_1 = a_1 \oplus x_0 \tag{3.13}$$

$$s_2 = a_2 \oplus x_1 \tag{3.14}$$

$$s_3 = a_3 \oplus x_2 \tag{3.15}$$

where V_{DD} is the physical representation of a digital '1'.

In order to simplify the task of implementing the above equations in a physical layout, it is preferable to rewrite Eqns (3.9)-(3.11) in terms of the available standard cells (4-input AND), and in a more regular manner (i.e. a repeatable structure) as follows,

$$x_0 = a_0 \cdot V_{DD} \cdot V_{DD} \cdot V_{DD} \tag{3.16}$$

$$x_1 = a_1 \cdot a_0 \cdot V_{DD} \cdot V_{DD} \tag{3.17}$$

$$x_2 = a_2 \cdot a_1 \cdot a_0 \cdot V_{DD} \tag{3.18}$$

Thus, with the above equations, it is possible to elaborate on the 4-bit adder shown in Figure 3.18 using the circuit diagram in Figure 3.19. The increment by one adder shown in the figure is comprised of only two levels of logic, allowing it to operate at high speeds.

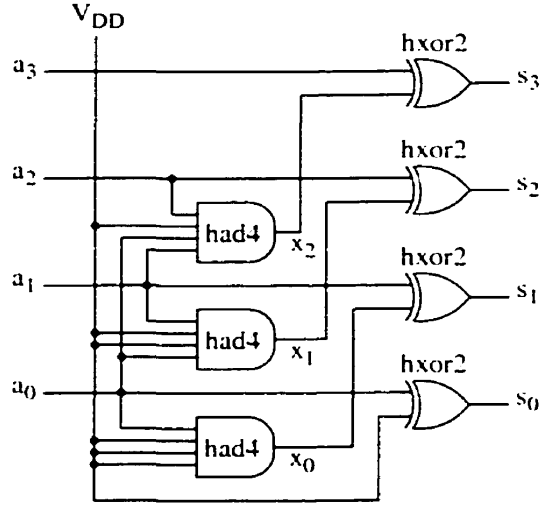

Figure 3.19 The simplified, increment by one, carry lookahead adder.

3.7 - Sample-and-Hold Controller Simulation

With the circuit description provided in the previous section, the sample-and-hold controller was physically laid out in a 0.5 μm CMOS technology, using the CADENCE CAD tool and the standard h-cell library. A transistor level netlist was then extracted with full parasitic capacitances. The HSPICE simulation results are shown in Figure 3.20, where a clocking rate of 500 MHz was used (the maximum operating rate of the sample-and-hold circuit in this technology). The results in the figure match the timing requirements shown in Figure 3.16 for a W=3 setting. Also note that as required, the sampling pulses P_1 and P_2 are non-overlapping with respect to each other, and are overlapping with respect to their complementary signals $\overline{P}_1$ and $\overline{P}_2$.

3.8 - Output Controller Design

As described in Section 3.6, the Multi-Pass system makes use of on-chip controller to obtain the various control signals necessary for the sample-and-hold circuit, and to

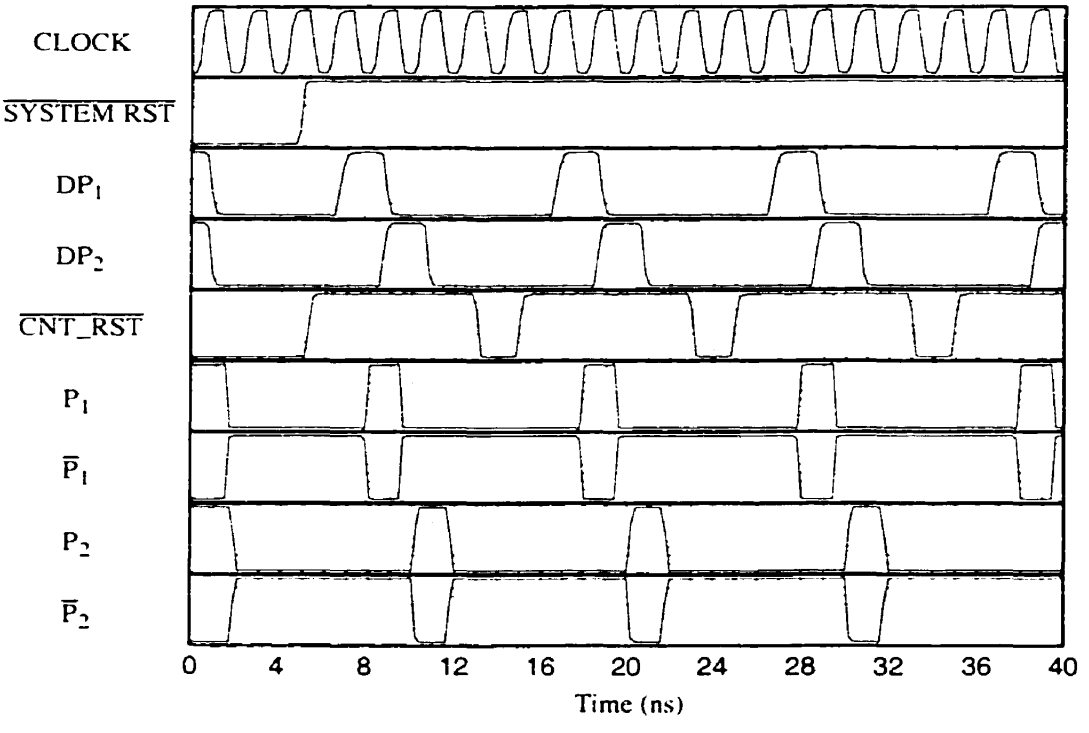

Figure 3.20 Simulation results for the sample-and-hold controller.

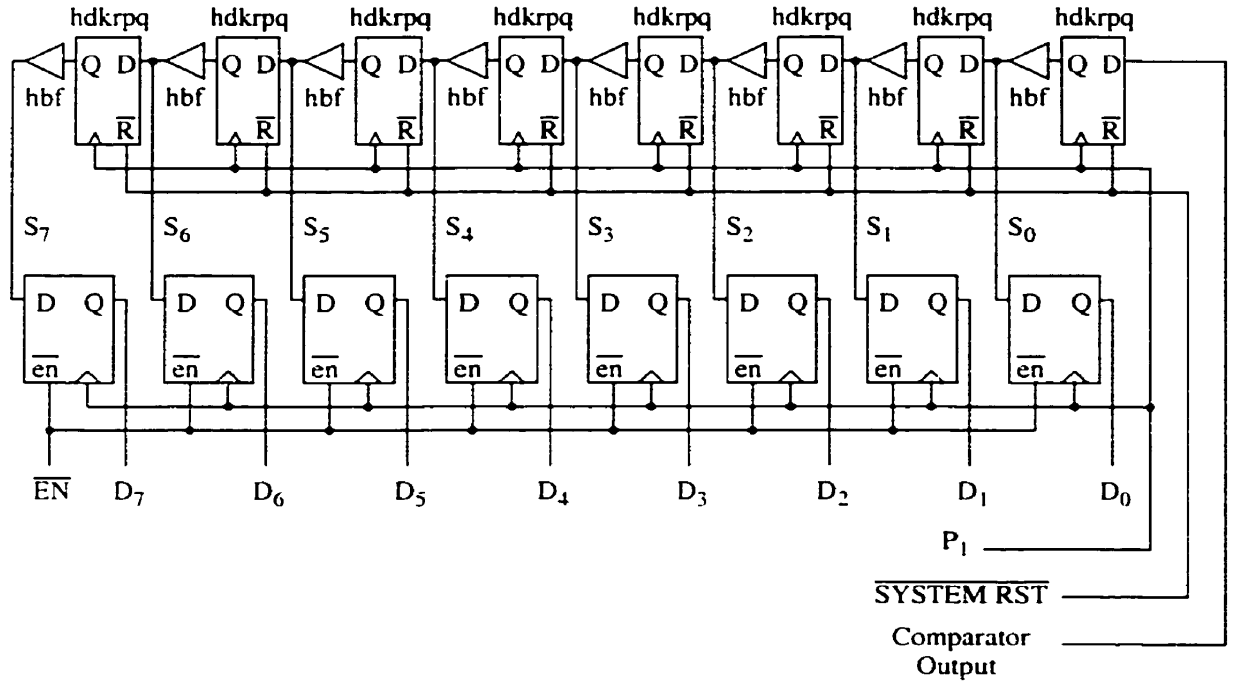

Figure 3.21 Circuit diagram of the serial-to-parallel convertor.

insure the coherency of the sample information between successive comparison passes. Consequently, since the external, Multi-Pass controller does not participate in the generation of the timing signals, it has no means of synchronizing with the comparator's output. Furthermore, the system is designed to be clocked at such a high rate that it is difficult to take the comparator's signal off-chip (using the standard bonding pads available in the CADENCE CAD tool), and to read and process the data using an external I/O device (e.g. HP E1451 I/O module). Therefore, what is required is an on-chip output controller that can slow down the output data rate, and that can provide a clock signal with which the external I/O device can synchronize.

3.8.1 - Serial-to-Parallel Conversion

The simplest way to slow down the comparator's output is to read the data in parallel, say eight bits at a time. This can be accomplished using the circuit shown in Figure 3.21, where a shift register is used to load the comparator data; once eight bits have been shifted in, the $\overline{EN}$ signal goes low indicating that the output register should load the

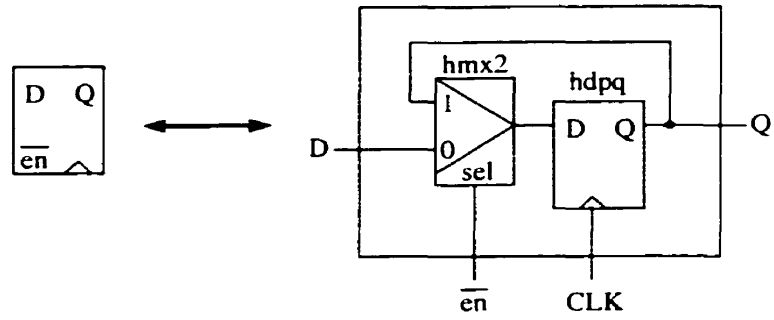

Figure 3.22 A D-type Flip-Flop with an active LOW enable.

next data set. Note that the parallel data could have been taken directly from the shift register outputs ($S_7..S_0$); however, since the data is shifting in rapidly, the shift register's output is changing at too high a rate, and hence, there would still be a need for a high-speed, external I/O device. Thus, an additional register is used to load the shift register's output once for every eight bits that are shifted in. Also note that since the analog waveform is sampled using the P_1 signal, a buffered version of P_1 can be used to clock the output controller.

The output register is comprised of a set of D-type Flip-Flops with an active LOW enable line. Such a device is not available as a standard h-cell; however, it can be implemented using a 2-input MUX as shown in Figure 3.22.

3.8.2 - Controller Timing Circuit

As with the sample-and-hold controller, the control signals required by the output controller can be generated from a 3-bit counter. A clock signal CLK_OUT (with a 50% duty cycle) can be produced using a toggle Flip-Flop as shown in Figure 3.23, where a low on the $\overline{\text{EN\_CLK}}$ line indicates that the Flip-Flop should toggle its output.

Figure 3.24 shows the timing diagram for the output controller. The $\overline{\text{EN}}$ signal is lowered when a count of zero is reached; similarly, the $\overline{\text{EN\_CLK}}$ signal is lowered when either a count of zero or four is reached. Thus, the output register loads on a count of one, and the CLK_OUT signal toggles on a count of one and five.

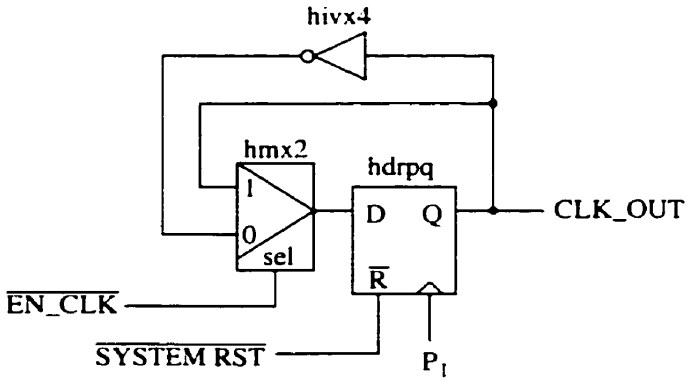

Figure 3.23 Circuit used to generate the CLK_OUT signal.

The circuit diagram for the output controller is shown in Figure 3.25, where the counter makes use of the high-speed design presented in Section 3.6.3. Once again, standard h-cells have been used to implement the circuit.

3.9 - Output Controller Simulation

The output controller was physically laid out in a 0.5 μm CMOS technology using components from the standard h-cell library. The layout was combined with the one for the sample-and-hold controller, in order to obtain the P_I signal that the output controller requires. A transistor level netlist was then extracted with full parasitic capacitances. The HSPICE simulation results are shown in Figure 3.26, where a clocking rate of 500 MHz and a W=3 setting were used. The results in the figure match the timing requirements

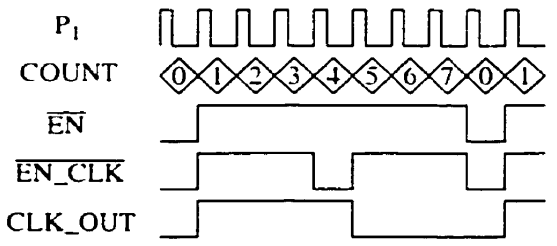

Figure 3.24 Timing diagram for the output controller.

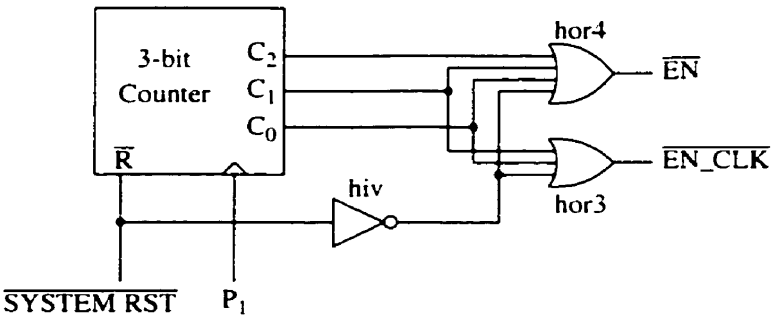

Figure 3.25 Timing circuit for the output controller.

shown in Figure 3.24. The CLK_OUT signal is synchronized with P_1, has a duty cycle of 50%, and has a frequency of $P_1/8$. The comparator output is represented by the sequence '10110010' (B2h - hexadecimal notation), followed by the sequence '01110101' (75h). The figure clearly shows that these two comparator sequences are properly shifted into the S-register, and once shifted in completely, are then loaded into the output register. Note

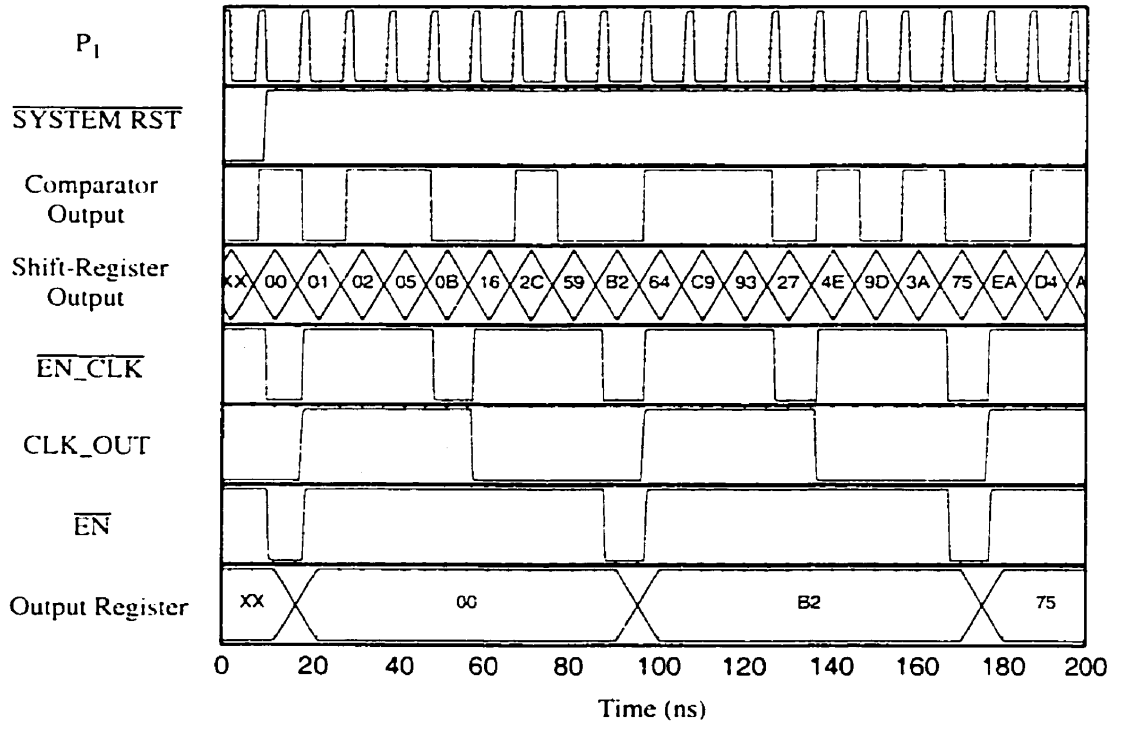

Figure 3.26 Simulation results for the output controller.

that as required, the output data ($D_7..D_0$) is held constant for eight cycles of P_1, resulting in a signal that is slow enough to read with the external I/O device.

3.10 - Multi-Pass System Simulation

The final step in the simulation process is to examine the behaviour of the complete Multi-Pass system. Using the circuit setup shown in Figure 3.11, the Multi-Pass convertor was physically laid out in a 0.5 μm CMOS technology. The on-chip components include two sample-and-hold circuits, two comparators (one for stand-alone use - not connected to the system, only to external pins), a sample-and-hold controller, an output controller, and the switch pairs SW1 and SW2 (refer to Figure 3.1). Figure 3.27 shows the layout view of the system obtained from the CADENCE CAD tool.

A transistor level netlist was extracted with full parasitic capacitances; using the device models for the 0.5 μm CMOS process, the circuit was then simulated with

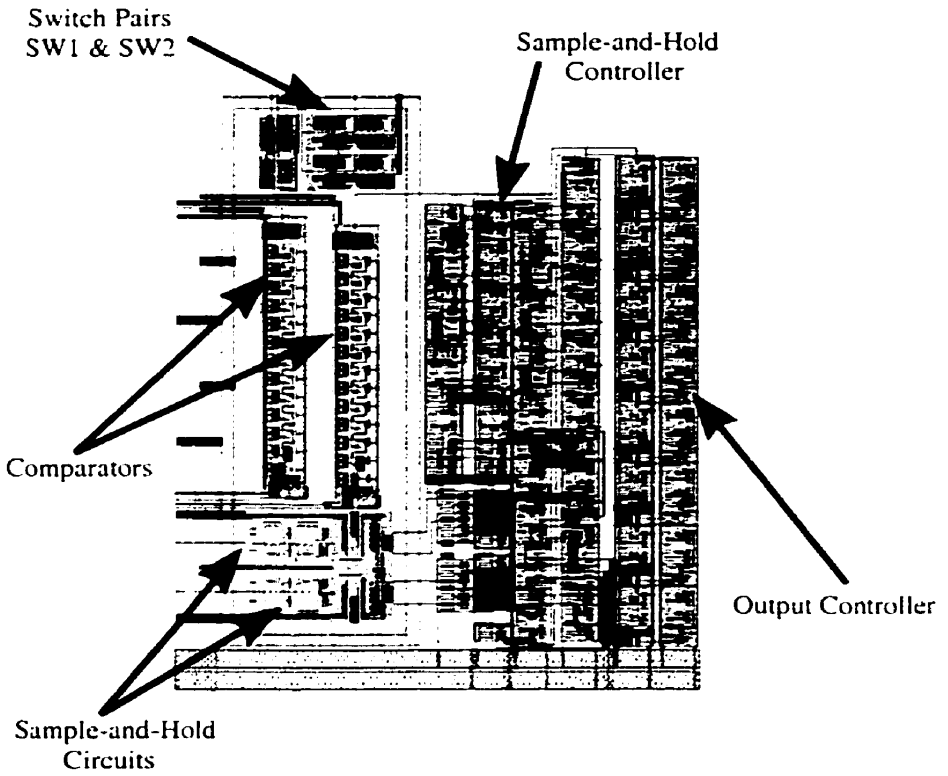

Figure 3.27 Layout view of the Multi-Pass convertor.

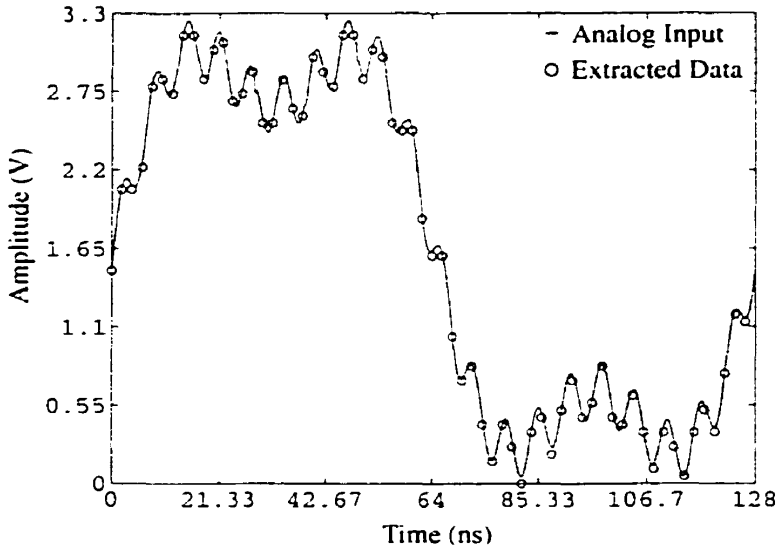

Figure 3.28 Simulation results for the Multi-Pass convertor where B=6, N=64, and F_s=500 MHz.

HSPICE. The results of simulation are shown in Figure 3.28, where the analog waveform is shown as a solid line, and the extracted data is shown as dots. The system is powered by a single 3.3V supply, and is clocked at a rate of 500 MHz. The input signal spans the full V_{SS} to V_{DD} range, and consists of a combination of three tones at the following frequencies: 7.81 MHz, 23.4 MHz, and 156 MHz. The extracted data set is obtained using the parameters B=6 and N=64.

Due to additional buffering stages that were needed at the output of the comparator, it was found that a W=3 setting is necessary to insure that the comparator has enough time to settle to its output. Referring back to Eqn (3.1), it is clear that five clock periods are required in order to compare a sample with the reference signal - this is also shown in the timing diagram of Figure 3.16, and in Figure 3.20. However, although sample information is taken at a rate of 100 MHz, it is interesting to note that after data processing, the effective sampling rate F_s is nonetheless 500 MHz.

The data processing step is performed by the external Multi-Pass controller, implemented here using a Matlab script file (see Appendix B.1). Essentially, the script file takes the shuffled comparator output (illustrated in Figure 3.13) from HSPICE, and

reorganizes the data such that the bits for each reference level fall into the correct sample positions. Next, the information for each sample point is examined in order to determine the level at which occurred a change of state in the comparator's output. Finally, quantization values are assigned, thus forming the digitized waveform.

Note that since the HSPICE netlist was extracted with full parasitics, and since it contains the transistor level representation of the digital circuits, the required simulation time is extremely long - a month's worth of simulation time was needed to construct the extracted waveform shown in Figure 3.28. Consequently, the parameters B and N were limited to 6 bits and 64 points respectively, in order to obtain results in a reasonable amount of time.

3.11 - Voltage Reference Design

The description of the Multi-Pass convertor presented in Section 2.3 shows that an external DAC is used to provide the reference signals with which to compare the analog waveform. Although such a DAC is sufficient for the proposed A/D algorithm, it is not convenient in the context of a fully integrable test core. A more desirable solution would be to integrate the DAC with the rest of the on-chip components of the Multi-Pass system.

Traditional D/A architectures [18] have associated with them two major difficulties: robustness and area efficiency. Note that in this case, high-speed performance is not absolutely necessary; Eqn (2.2) states that a slow DAC will simply increase the overall conversion time. Although a fast conversion is preferable, a slow one does not hinder waveform extraction at high speeds. The difficulty with robustness arises with the need for high amplitude resolutions, mainly due to component (transistor, resistor) matching limitations. Such limitations lead to DAC transfer characteristics that are non-linear, and hence, produce distortions in the extracted waveform.

It is possible to apply error correction schemes in order to improve the linearity of the DAC; however, this leads to the difficulty of area efficiency because of the additional error-correcting circuitry.

A simple solution presents itself upon a closer look at the definition of the DC component of a signal: the DC term consists of the average value of the signal. Consider applying the above definition to the most common digital signal: the clock (or square wave). The average value of a clock signal is simply its duty cycle multiplied by V_{DD}. Thus, the easiest way to obtain a programmable reference source is to build a circuit that can generate a clock signal with a variable duty cycle.

Given the maximum number of quantization levels 2^B that are required, the clock period must be set according to the following,

$$T_{CLK} = 2^B \cdot t_d \tag{3.19}$$

where t_d is any arbitrary time division (it will be shown later that t_d is related to the settling time of the reference signal). Essentially, Eqn (3.19) describes a clock period that is composed of 2^B time divisions t_d, in order to accommodate 2^B different duty cycles (or average values). Next, the quantized output voltage can be described by,

$$V_{out}(k,t) = Mean(V_{CLK}(k,t)) = \frac{1}{T_{CLK}} \int_{T_{CLK}} V_{CLK}(k,t)dt \tag{3.20}$$

and

$$\begin{aligned} V_{CLK}(k,t) &= V_{DD} \qquad n \cdot T_{CLK} \le t < (k \cdot t_d + n \cdot T_{CLK}) \\ V_{CLK}(k,t) &= V_{SS} \qquad (k \cdot t_d + n \cdot T_{CLK}) \le t < (n+1) \cdot T_{CLK} \end{aligned} \tag{3.21}$$

where n is an integer, t is time, and k is the quantization level (i.e. k=0, 1, 2, ... , 2^B-1). Figure 3.29 illustrates the quantization set for B=2, where by taking the average (or mean) of the V_{CLK} waveforms, a set of DC levels that span the V_{SS} to V_{DD} range can be obtained.

The next step is to find a means of obtaining the average value of the $V_{CLK}(k,t)$ waveforms. The simplest method is an RC lowpass filter as shown in Figure 3.30. Such a filter is robust in the sense that the magnitude of its transfer function is unity at DC, irrespective of the values chosen for R and C:

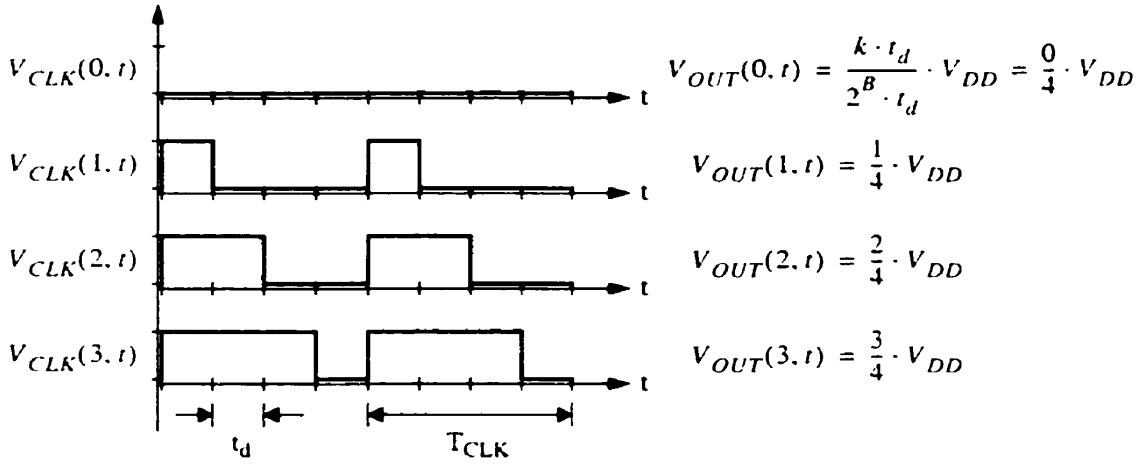

Figure 3.29 Quantization vector set for B=2.

$$H(s) = \frac{V_o(s)}{V_{CLK}(s)} = \frac{1}{1 + \frac{s}{1/RC}} \tag{3.22}$$

where at DC (s=0), the magnitude response is given by.

$$|H(0)| = 1 \tag{3.23}$$

The choice of R and C simply determine the cutoff frequency of the filter. Consider the Fourier Transform of the $V_{CLK}(k,t)$ signal given by the following equation [22].

$$V_{CLK}(k,s)\big|_{s=j\omega} = \sum_{n=-\infty}^{\infty} \frac{2V_{DD}}{n} \sin\left(n\pi \frac{k \cdot t_d}{T_{CLK}}\right) \delta\left(\omega - \frac{2\pi n}{T_{CLK}}\right) \tag{3.24}$$

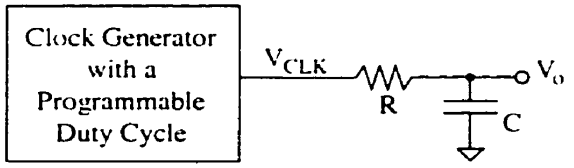

Figure 3.30 Block diagram of the programmable voltage source.

The above equation can be rewritten as,

$$V_{CLK}(k,s)\big|_{s=j\omega} = \sum_{n=-\infty}^{\infty} 2\pi V_{DD}\frac{k\cdot t_d}{T_{CLK}}\,\mathrm{sinc}\left(n\frac{k\cdot t_d}{T_{CLK}}\right)\delta\left(\omega-\frac{2\pi n}{T_{CLK}}\right) \tag{3.25}$$

where

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \tag{3.26}$$

Eqn (3.25) describes a frequency spectrum that consists of a set of scaled tones (or impulses) spaced at intervals of $1/T_{CLK}$ Hz, as illustrated in Figure 3.31. Consequently, in order to obtain a signal with only a DC component, all tones must be filtered out by the RC circuit. More specifically, in order to obtain an amplitude resolution of B bits, the filter must satisfy the following attenuation (in dB) condition.

$$\begin{aligned} |H(s)|_{s=j2\pi/T_{CLK}} &\le 20\cdot\log_{10}(2^B) \\ &\le 6.02\cdot B \end{aligned} \tag{3.27}$$

in order to insure that the $f=1/T_{CLK}$ component (largest tone) of the output signal is small enough such that the output does not reach the adjacent quantization levels.

Next, the circuit must be given enough time to settle to its output. The worst case settling time occurs when the capacitor starts with a charge of zero, and must be charged up to the maximum possible voltage level V_{max} in the quantization set:

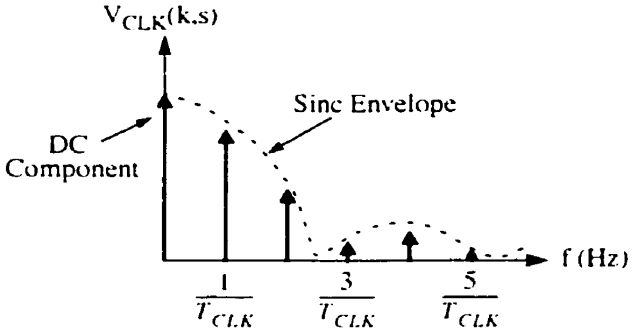

Figure 3.31 Frequency spectrum of the V_{CLK} signal.

$$V_{max} = \frac{2^B - 1}{2^B} \cdot V_{DD} \tag{3.28}$$

The output voltage $V_o(t)$ can be described by the following equation (single time constant circuit),

$$V_o(t) = V_{max} \cdot (1 - e^{-t/\tau}) \tag{3.29}$$

where

$$\tau = RC \tag{3.30}$$

For an amplitude resolution of B bits, the output voltage must be allowed to settle to within at least half an LSB ($V_{DD}/2^B$) of its final value:

$$V_o = V_{max} - \frac{1}{2} \cdot \frac{V_{DD}}{2^B} = V_{max} - \frac{V_{DD}}{2^{B+1}} \tag{3.31}$$

Using Eqns (3.30) and (3.31) to replace the V_o and τ terms in Eqn (3.29), it is possible to obtain the following solution for the required minimum settling time T_{min},

$$T_{min} = RC \cdot \ln\left(\frac{V_{max} \cdot 2^{B+1}}{V_{DD}}\right) \tag{3.32}$$

Eqn (3.32) can be further simplified by replacing the V_{max} term with the expression in Eqn (3.28) to obtain,

$$\begin{aligned} T_{min} &= RC \cdot \ln\left(\frac{2^{B+1}}{V_{DD}} \cdot \frac{2^B - 1}{2^B} \cdot V_{DD}\right) \\ &= RC \cdot \ln(2^{B+1} - 2) \end{aligned} \tag{3.33}$$

Intuitively, Eqn (3.33) can be interpreted as describing the minimum number of time constants (τ=RC) that must be allowed to pass before the reference output can be used. It is possible to use additional RC stages in order to reduce the minimum settling time - Eqn (3.33) must be adjusted accordingly. Note that it may be necessary to wait for a few more time constants than the one indicated by T_{min}, especially if the system is being operated

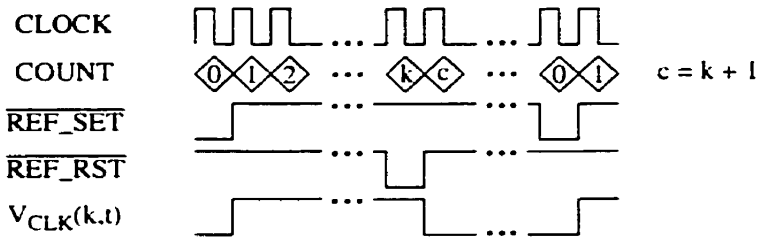

Figure 3.32 Timing diagram for the programmable clock generator.

with an amplitude resolution that is near the specified limits of the comparator - an error of half an LSB may not be enough to guarantee a correct comparison.

The final step in the design of the programmable reference source is the implementation of the clock generator. The most efficient structure for the clock generator is based around a B-bit counter. Figure 3.32 shows the timing diagram of the required circuit, where the signals $\overline{\text{REF\_SET}}$ and $\overline{\text{REF\_RST}}$ indicate when the V_{CLK} signal should go high and low respectively. The circuit diagram for the programmable clock generator is shown in Figure 3.33. It is important to note here that the frequency of the CLOCK signal is equivalent to $1/t_d$, and thus, according to Eqn (3.19), specifies the period T_{CLK} of the V_{CLK} signal. Consequently, a higher clocking rate for the counter results in a higher cutoff

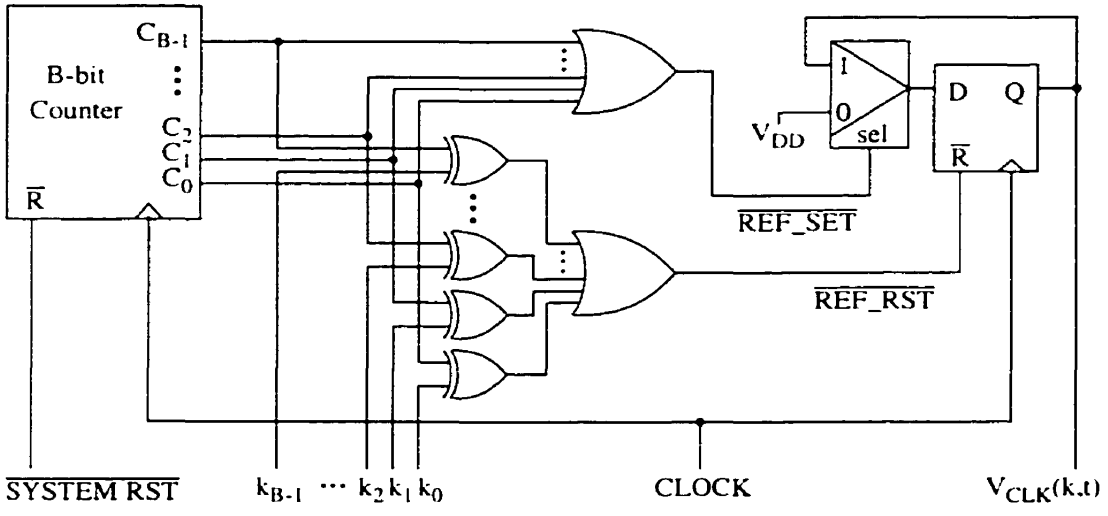

Figure 3.33 Circuit diagram of the programmable clock generator.

frequency for the RC filter, and therefore, a lower settling time T_{min}. Furthermore, a higher cutoff frequency for the lowpass filter results in a reduction of the values of R and C, and thus, produces a more area efficient design.

In summary, the proposed programmable voltage source is area efficient and robust. Since the settling time of the reference output does not hinder waveform extraction at high speeds, the design trades off area for time - the higher the amplitude resolution, the longer the settling time. The voltage source is comprised almost entirely of digital circuitry, making it robust. In fact, the only analog components are a resistor and a capacitor that are used in such a manner that their precision does not affect the amplitude resolution of the reference signal. However, although the dependency on component matching (traditional D/A designs) has been eliminated, the proposed design's performance depends heavily on the amount of jitter present on the CLOCK waveform. The linearity of the voltage source is determined by the matching of the time divisions t_d, as well as the consistency of the digital output HIGH level. This said, the design's source of limitations are more easily overcome than the ones for traditional D/A architectures.

As an alternative to the voltage source presented here, a $\Delta\Sigma$-type generator could be used to produce a PDM encoded DC signal. Essentially, the 1-bit PDM stream would consist of a DC term and some additional, high frequency noise. In a typical $\Delta\Sigma$ modulator (e.g. 2^{nd} order), the noise power rises gradually from f=0 (DC) to $f=F_s/2$ - this leads to a lowpass filter specification that is more relaxed than the one presented in this section. Consequently, a higher RC cutoff frequency can be used to reduce the output settling time of the voltage source. However, although a shorter settling time would be obtained, a $\Delta\Sigma$-type generator would require more area to implement.

3.12 - Conclusion

The test core design presented in this chapter shows the integration of a memory-based signal generator and an analog waveform extractor onto a single IC. Figure 3.1 shows the various configurations that are possible (using switches) once the circuit has

been fabricated; these different connections allow for the possibility of testing the IC components separately, in the event of partial failures after fabrication.

A detailed description of the on-chip components of the Multi-Pass system was presented. Included in the description were the comparator design, with a means of eliminating offset error (input-swapping scheme); the sample-and-hold circuit, that incorporates a mapping function to allow for the sampling of full-scale inputs; the sample-and-hold controller, that is at the heart of the Multi-Pass system with its generation of the key control signals; and the output controller, that eases the task of interfacing with the external digital controller. In addition, a simple, area efficient, highly linear, and robust programmable voltage source was presented.

With all of the above components designed and laid out, the prototype test core was sent for fabrication in a 0.5 μm CMOS process. The system is self-contained in the sense that no complex, external control signals are required, nor is there a need for special biasing circuitry. All that is needed to test the design is a digital I/O card (synchronized with the CLK_OUT signal), to load the RAM and to record the comparator data; a Matlab script file, to generate the PDM stream and to interpret the comparator data; an external DAC, to provide the reference signals; and a filter, to remove the noise present in the PDM stream.

Therefore, although the Multi-Pass system makes use of a fairly complex timing mechanism to capture data, the fact that it is all contained within the on-chip components makes it very easy to work with, and thus fulfills the requirements for a practical test solution.

Chapter 4 - Experimental Results

4.1 - Introduction

Experimental data are an important part of determining the feasibility of the proposed mixed-signal test system. Many practical issues become apparent only during the actual implementation stage - such issues include timing and control (of external components), data capture and processing, and system characterization (calibration). Furthermore, since the simulation time required for the complete Multi-Pass convertor is extremely long, the only way to truly verify its functionality is through experimental means.

The following chapter presents the results obtained from two prototype boards: the first implements the test system using discrete components, and the second makes use of the prototype IC described in Chapter 3.

4.2 - Implementation with Discrete Components

The test system shown in Figure 3.1 can be implemented (for verification purposes) using discrete components. Figures 4.1 and 4.2 show the picture and the block diagram of the experimental board respectively. The design can be divided into two parts: the stimulus generator, and the Multi-Pass convertor. Each of these two parts is comprised of both software and hardware components.

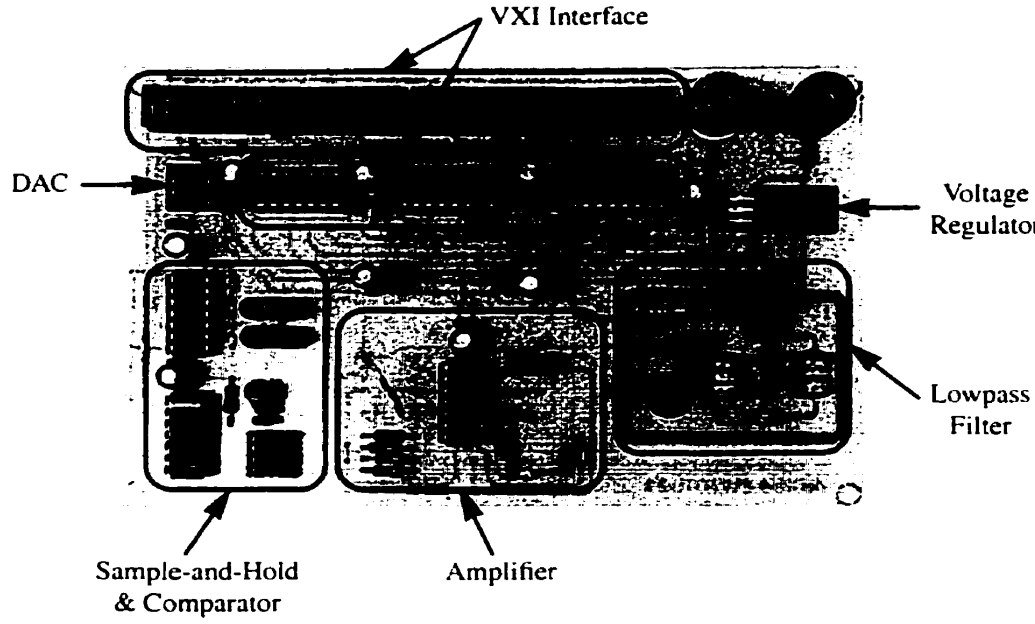

Figure 4.1 PCB board used to implement the proposed test system.

The stimulus generator begins with an arbitrary discrete-time waveform that is then passed through a software implemented ΔΣ modulator. The resulting 1-bit PDM stream is fed directly into a lowpass elliptic filter, consisting of passive (RLC)

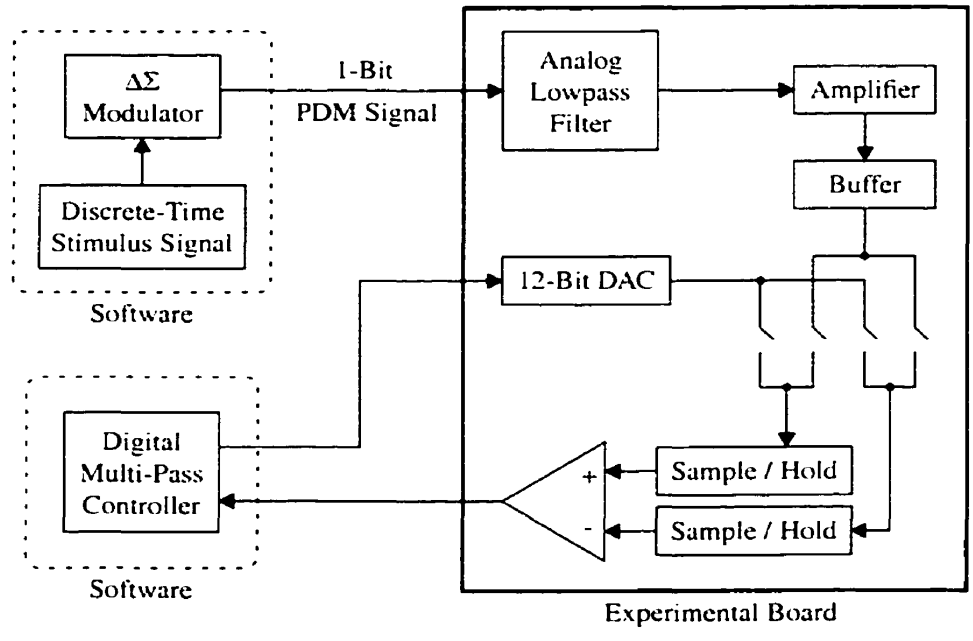

Figure 4.2 Block diagram of the prototype board that implements the test system using discrete components.

components. The filtered signal is then sent to an amplifier followed by a buffer: the amplifier is needed in order to obtain near full-scale output signals (the board is powered by a single 5 V supply).

The Multi-Pass section receives the buffered stimulus signal which is then passed through the input-swapping switches (offset error correction scheme). A 12-bit DAC is used to provide the reference levels; its output is also passed through the switches. Next, the reference and stimulus signals are fed into separate sample-and-hold blocks before connecting to the comparator (as described in Section 3.4). The comparator's output is then sent to the software implemented digital controller. The data for each level is stored, and later processed according to the Multi-Pass algorithm, in order to produce a digitized version of the analog waveform.

The interface between the software and hardware components is provided by an HP 75000 Series VXIbus system. Specifically, the HP E1451 Pattern I/O module is used both to output the PDM stream, and to capture the comparator data. Thus, since the stimulus signals are generated digitally by the same device that captures the comparator data, the Multi-Pass extraction system is automatically synchronized with the analog waveforms.

4.2.1 - Signal Generation

As mentioned previously, the desired discrete-time stimulus signal is passed through a $\Delta\Sigma$ modulator that converts the multi-bit signal (32-bit floating point format) into a 1-bit PDM stream. This eliminates the need for a D/A convertor before the reconstruction filter. For the experimental setup presented here, a 6^{th} order lowpass $\Delta\Sigma$ modulator with an OSR of 32 is used; the modulator is implemented by means of a Simulink model file as shown in Figure 4.3. The modulator coefficients A(1),....A(7), and B(1),....B(6) were obtained from the DSMOD CAD tool [10], and are listed in Table 4.1.

The PDM bit-stream is output at a clocking rate of 2.286 MHz, resulting in a signal band of about 34 kHz. The frequency spectrum of a sample PDM encoded stimulus signal

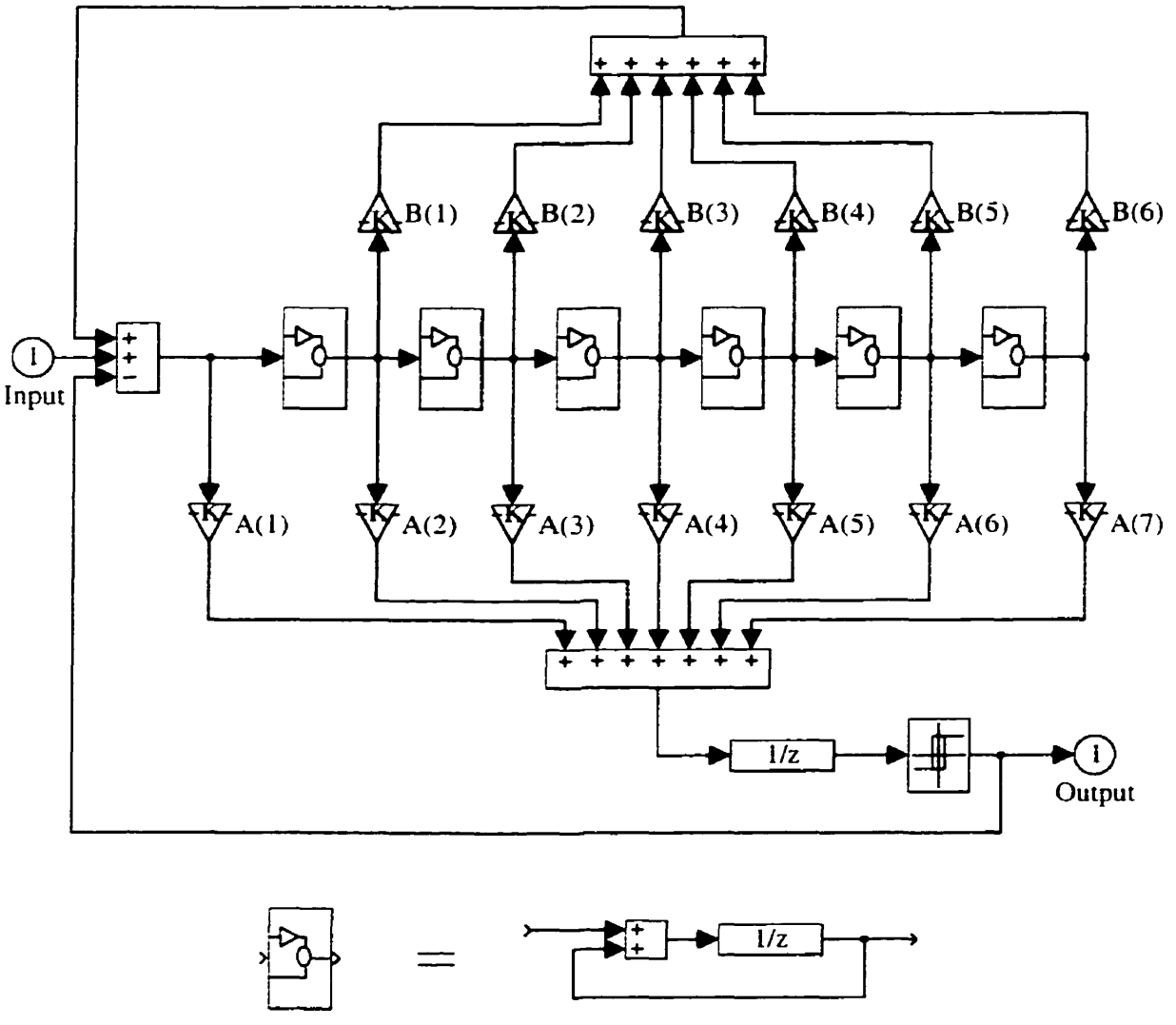

Figure 4.3 Simulink model of a 6th order lowpass ΔΣ modulator.

is shown in Figure 4.4: the signal contains a sine wave with an amplitude of 0.25 at 17.8 kHz, and some additional out-of-band, high-frequency noise.

4.2.2 - Analog Filter and Amplifier

The analog filter plays a dual role in the system presented here. First, it serves as a reconstruction filter that eliminates the images of the Nyquist band (digital signal); second, it removes the unwanted, high-frequency noise present in the PDM encoded signal. A 7th order lowpass elliptic filter is used; the LC-ladder implementation of the filter

Table 4.1 - Gain coefficients of the 6th order ΔΣ modulator.

Coefficient	Value	Coefficient	Value
A(1)	7.87706e-01	B(1)	-1.31358e-02
A(2)	1.08893e+00	B(2)	-1.31780e-02
A(3)	3.77668e-01	B(3)	-8.43883e-05
A(4)	8.84964e-02	B(4)	-4.22426e-05
A(5)	1.32240e-02	B(5)	-5.80774e-08
A(6)	1.22636e-03	B(6)	-1.93591e-08
A(7)	5.43263e-05		

is shown in Figure 4.5. The filter is designed with a passband edge at 25 kHz, and a 60 dB attenuation level at 34 kHz (for a 10-bit spurious-free dynamic range).

The out-of-band noise content of the 1-bit PDM signal makes it impossible to obtain a waveform with a full range voltage swing - this is due to the fact that a lot of power goes into the noise portion of the signal. Consequently, an amplifier is required; however, the gain of the amplifier is limited by the op-amp distortion levels. For the experimental board presented here, it was found that a gain of two yielded acceptable distortion levels; note that this gain merely compensates for the filter's passband

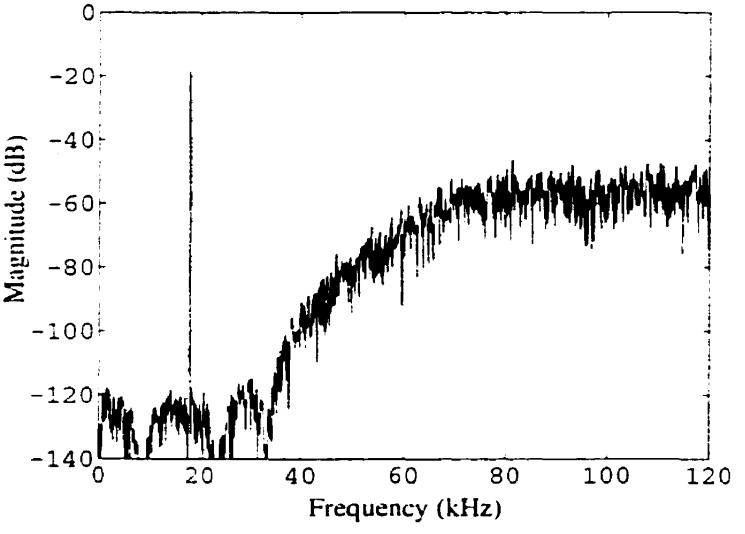

Figure 4.4 Frequency spectrum of a PDM encoded signal.

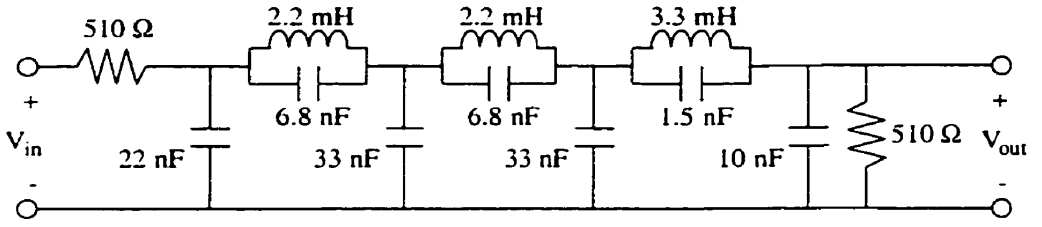

Figure 4.5 Circuit diagram of the 7th order lowpass LC-ladder filter.

magnitude response of -6 dB. The circuit diagram of the amplifier is shown in Figure 4.6; it consists of a non-inverting gain stage, followed by a unity-gain buffer stage.

Figure 4.7 shows the magnitude response of the combined filter and amplifier, measured with an HP 3588A spectrum analyzer. As mentioned above, the amplifier provides the gain that is needed to obtain a 0 dB magnitude response in the passband. Figure 4.8 shows a filtered and amplified PDM signal, as seen with a Tektronix 2235 oscilloscope set to 0.5 V/div and 10 μs/div; the PDM stream is the same as the one used in Figure 4.4.

4.2.3 - Waveform Extraction

In a synchronous capture system, where one wants to avoid the complication of delays between successive comparison passes, it is very important that the number of sample points N be chosen appropriately, relative to the spectral content of the stimulus signal. N must represent an integer number of periods for each of the tones that comprise the stimulus signal. If this condition is not satisfied, the sample points will not repeat every

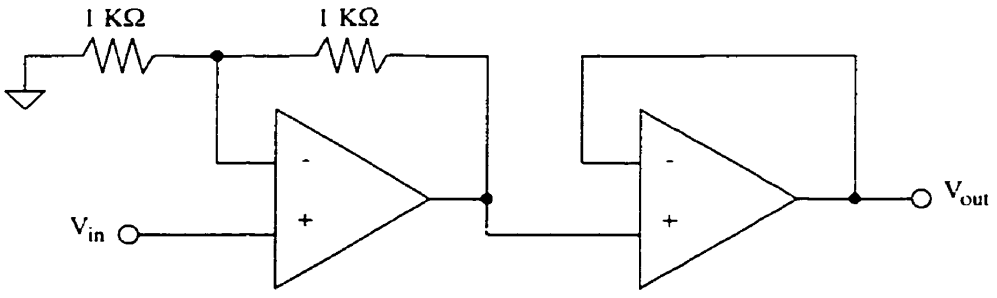

Figure 4.6 Circuit diagram of the amplifier (voltage gain = 2).

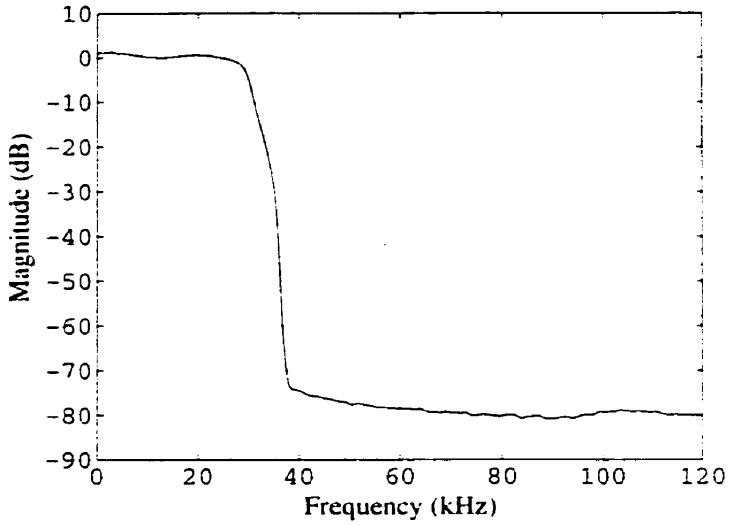

Figure 4.7 Measured magnitude response of the combined lowpass filter and amplifier.

N number of points, and thus, a significant amount of distortion will appear in the extracted waveform. Therefore, the first step in capturing a waveform is to select a correct number of sample points.

In the case of the oversampled stimulus generator presented here, a large number of points is required in order to obtain a reasonable frequency resolution in the available signal band. The experimental setup was limited to a maximum of 8192 sample points.

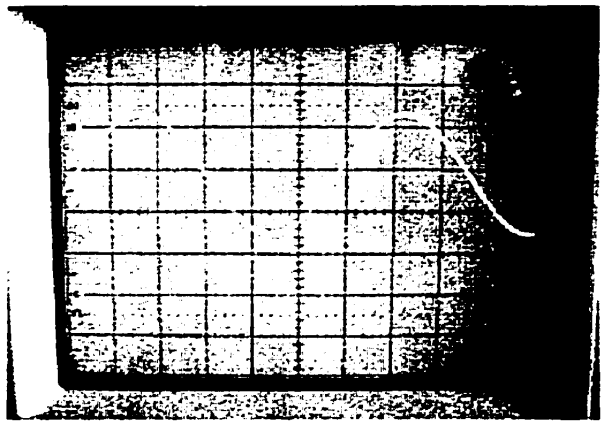

Figure 4.8 A sample analog stimulus signal seen at the output of the amplifier (oscilloscope settings: 0.5 V/div and 10 μs/div).

Using the cutoff frequency shown in Figure 4.7, an effective sampling rate of 2.286 MHz, and the maximum for N, one concludes that roughly 90 frequency bins are available for use. Consequently, in order to make use of all the available bins, and to satisfy the repeatability condition stated above, N was set to 8192 points for all of the captured waveforms that follow.

Next, a VXI program was written in C (see Appendix C.1) to read the PDM stimulus vector generated by Matlab (Section 4.2.1), to set up the necessary control signals required by the experimental board (i.e. clock, DAC configuration data, comparator input polarity, etc.), and to record and process the output of the comparator for all N sample points. Figure 4.9 shows a waveform captured at various amplitude resolution settings. The waveform represents the fundamental frequency F_0 (i.e. F_0/F_s = 1/8192) for this system. Having verified the functionality of the capture system, the next step is to determine the system's linearity. To do so, a stimulus signal containing a single sine wave at 17.8 kHz was used (the signal is the same as the one in Figure 4.4) as an input to the Multi-Pass A/D convertor; Figure 4.10 shows the spectral plot of the extracted

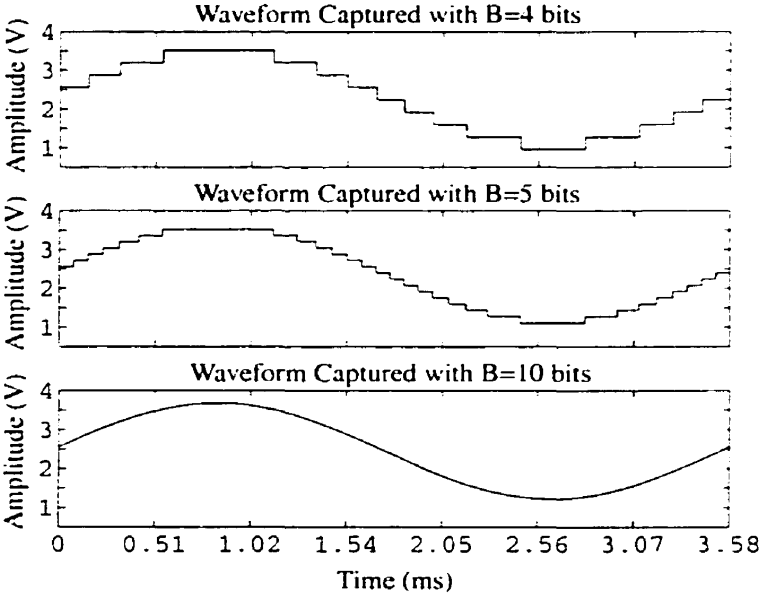

Figure 4.9 A set of waveforms captured with different amplitude resolution settings (B=4, 5, 10).

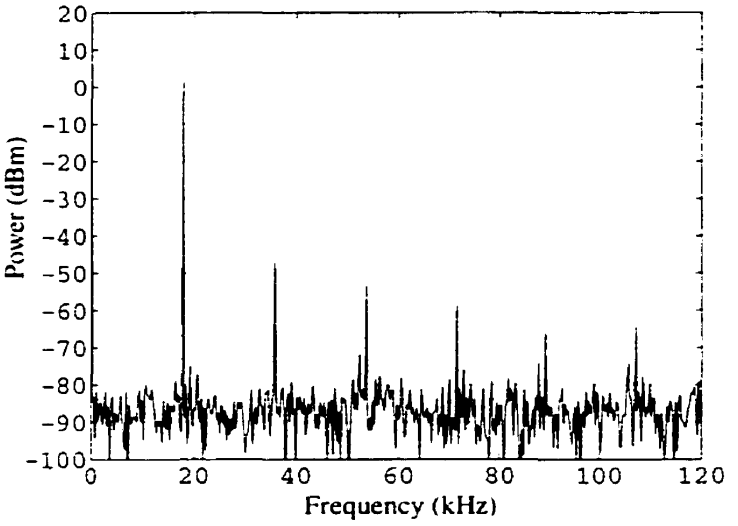

Figure 4.10 Frequency spectrum of a captured sine wave (f=17.8 kHz; B=10 bits).

waveform. The data was extracted using an amplitude resolution setting of 10 bits - the total harmonic distortion (THD) is -47.5 dB.

4.2.4 - The Effects of Clock Jitter

The Multi-Pass algorithm proposed in Chapter 2 makes use of multiple comparison passes along the UTP of an analog waveform, in order to determine the quantization values of the N sample points. For each point, the technique must analyze the comparator data from successive passes in order to find its correct quantization value. Consequently, sample coherency between the multiple passes is paramount to the successful digitization of the analog waveform - the technique is particularly sensitive to the effects of clock jitter.

It is possible to model the effects of jitter on the extraction process by applying a random position offset to each reference level's comparator data set, as illustrated in Figure 4.11. Clearly, this is a simplified representation of jitter - a more complex and time-consuming (simulation) approach would be to model a distinct, random position shift for each of the comparator data points (rather than a group position shift for a given reference level). A Matlab script file (see Appendix B.2) was used to simulate the digitization of a

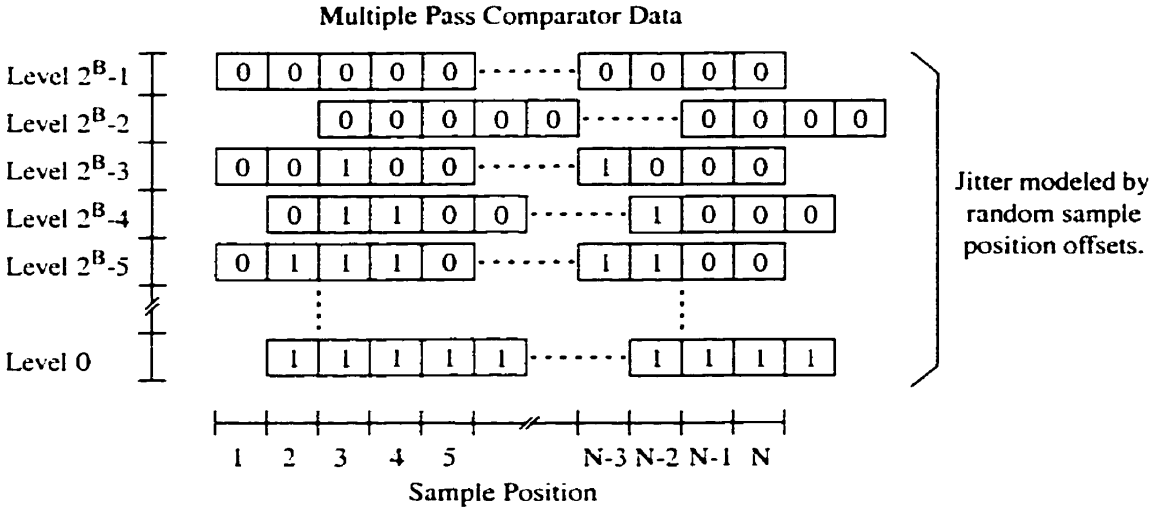

Figure 4.11 Modeling the effects of clock jitter by randomly offsetting the starting position of the comparator data for each reference level.

sine wave using the Multi-Pass algorithm: for each comparison pass (or reference level), the data starting position was offset by a random amount (the offset was limited to three sample positions). Note that any data point that is shifted out of the valid position range is simply wrapped around to the beginning of the range. Figure 4.12 shows the spectral plot of the distorted waveform, obtained from an input analog signal similar to the one used in the previous subsection. A significant amount of distortion is visible in the figure.

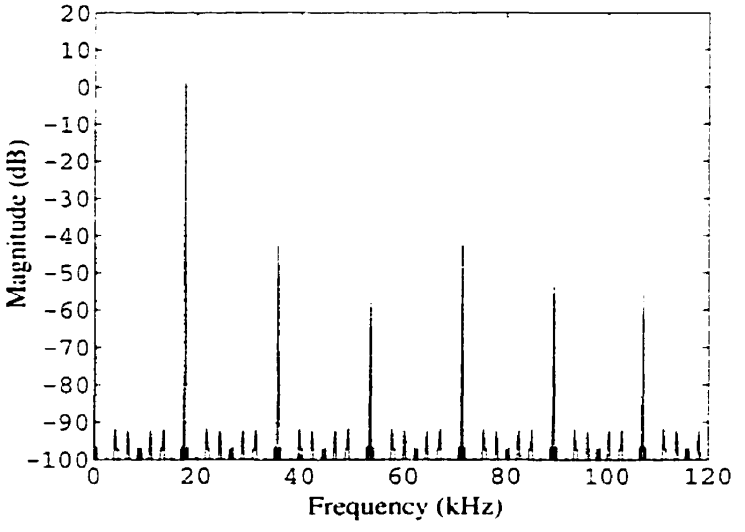

Figure 4.12 Frequency spectrum of a sine wave captured using a Multi-Pass convertor with clock jitter (f=17.8 kHz; F_s=2.286 MHz).

comparable to the one seen in Figure 4.10. However, it is important to keep in mind that clock jitter is only one possible cause of distortion; there is also the fact that both the reference signal and the analog waveform pass through similar switches, and thus, the matching accuracy of these switches also contributes to distortion.

4.2.5 - System Calibration

In order to correctly interpret the extracted data, it is essential to perform a calibration step. Such a step consists of measuring the system's magnitude and phase response for each of the available frequency bins in the signal band. The measured data for each bin is then referred back to its respective discrete-time input signal. It is important to note here that the magnitude response represents a relationship between a set of voltages (measured data) and a set of numbers (stimulus amplitudes). Although calibration is a lengthy process, it need only be done once for a given system. The data can then be used to compute an inverse function that is stored, and later applied to all extracted waveforms.

During calibration, the highest possible amplitude resolution setting should be used. For the experimental board presented here, the comparator can detect voltage differences down to about 5 mV. This limits the system's amplitude resolution to 10 bits (corresponding to the ratio of 5 mV to the supply range of 5 V).

Figure 4.13 shows the magnitude and phase response of the prototype board. The calibration set shown in the figure includes data for frequency bins ranging from 1 to 64. For each bin, a PDM stream containing a single sine wave was used to excite the system. The stimulus amplitude A_{STIM} was set to 0.5 in all cases.

The measured frequency response can be used to write the system's impulse response h[n], in closed-form, as follows,

$$h[n] = \sum_{k=0}^{N-1} A_k \cos\left(\frac{2\pi k}{N} n + \Phi_k\right) u[n] \qquad (4.1)$$

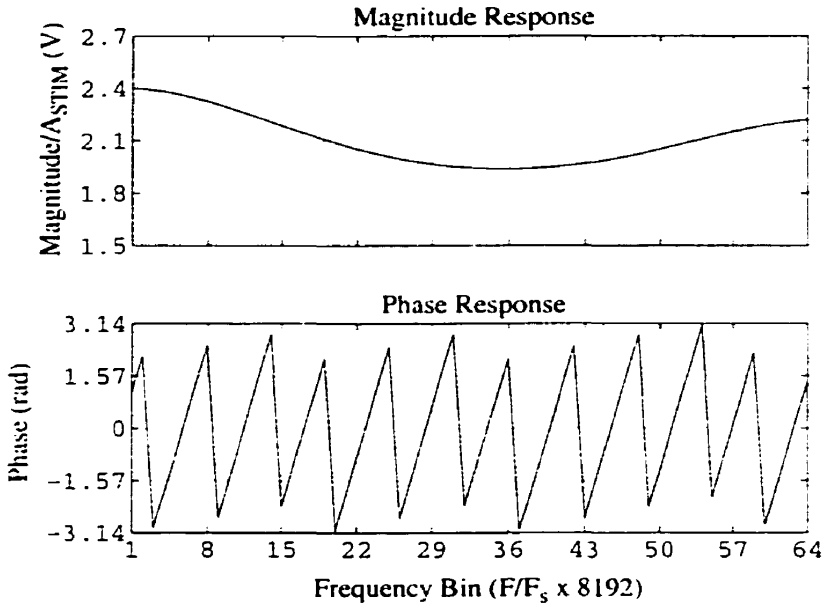

Figure 4.13 Closed-loop transfer function of the experimental board (calibration set includes bins 1 through 64).

where k is the frequency bin, N is the number of sample points, and A_k and Φ_k are the measured amplitude and phase of the k^{th} bin respectively. The system's inverse function can then be obtained by finding a function $h^{-1}[n]$ that satisfies the following condition,

$$h[n] \otimes h^{-1}[n] = \delta[n] \tag{4.2}$$

where $\otimes$ denotes an N-point circular convolution. It can be shown that $h^{-1}[n]$ is given by,

$$h^{-1}[n] = \frac{2}{N} \cdot \sum_{k=0}^{N-1} \frac{1}{A_k} \cos\left(\frac{2\pi k}{N} n - \Phi_k\right) u[n] \tag{4.3}$$

Thus, using the magnitude and phase information from Figure 4.13, and Eqn (4.3), it is possible to construct the compensation vector shown in Figure 4.14. Essentially, by convolving (circular convolution) the vector with the captured data set, one can compensate for the effects of the stimulus generator and the Multi-Pass convertor.

Note that the time-domain approach to obtaining an inverse function, as opposed to a frequency-domain approach, has two advantages. First, it directly produces the time-domain vector needed for the convolution operation (no inverse FFT involved). Second, in

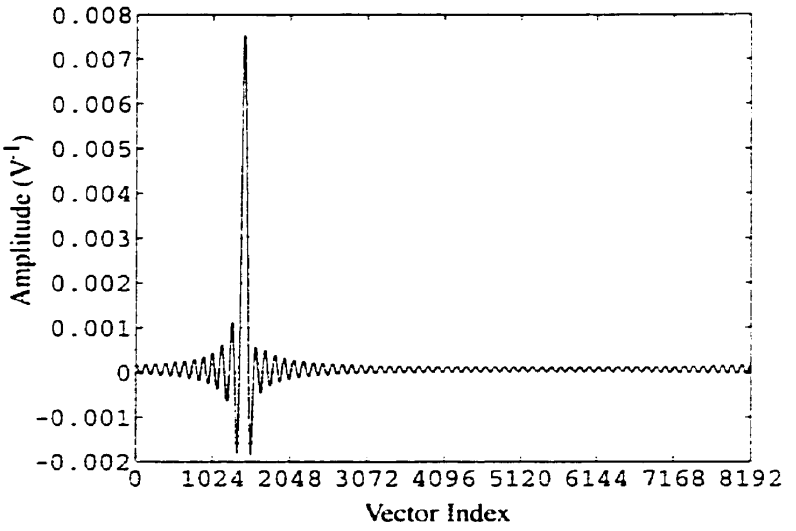

Figure 4.14 System compensation vector (8192 points).

the case of a signal band that is smaller than the Nyquist band (e.g. an oversampled system), where the system's transfer function cannot be measured for all N frequency bins (nor is it required to do so), one can simply limit the summation of Eqn (4.3) to the data measured in the band of interest. It is more difficult to handle such a case with the frequency-domain approach; the difficulty lies in finding a set of magnitude and phase values for the unknown frequency region, such that the inverse FFT leads to a real, N-point vector.

In order to demonstrate the results of calibration, a stimulus waveform similar to the one in Figure 3.28 was generated, and then captured. The waveform consisted of a combination of sine waves located at the following frequencies: 279 Hz (bin 1), 837 Hz (bin 3), and 5.58 kHz (bin 20). Figure 4.15 shows the signal at the output of the amplifier as seen with an oscilloscope. Figure 4.16 shows the captured waveform, where an amplitude resolution of 10 bits was used. In order to see the difference between the captured data before and after compensation, it is necessary to look at a magnified view of the extracted waveform. Figure 4.17 shows the superposition of the captured data over the ideal, discrete-time signal - the extracted data has been shifted and scaled in order to obtain the best fit. Some small, but important differences are noticeable; these differences are of course due to the frequency dependent gain and phase shift of the system. Figure

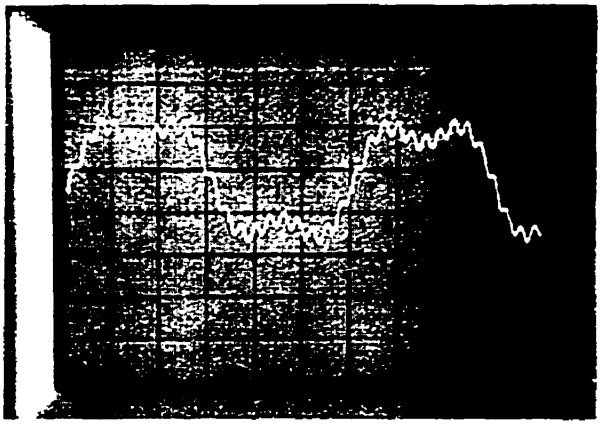

Figure 4.15 A three-tone stimulus signal seen at the output of the amplifier (oscilloscope settings: 0.5 V/div and 0.5 ms/div).

4.18 shows the ideal curve superimposed onto the compensated waveform; the vector shown in Figure 4.14 was applied directly to the captured waveform. The differences are of the order of 10^{-3}, too small to be seen at the current magnification; Figure 4.19 shows the difference between the two curves; note that the difference from the ideal corresponds to the quantization error made with a resolution of 10 bits.

In summary, the compensation vector obtained from the calibration step can be used to eliminate the magnitude and phase errors caused by the signal generator and the

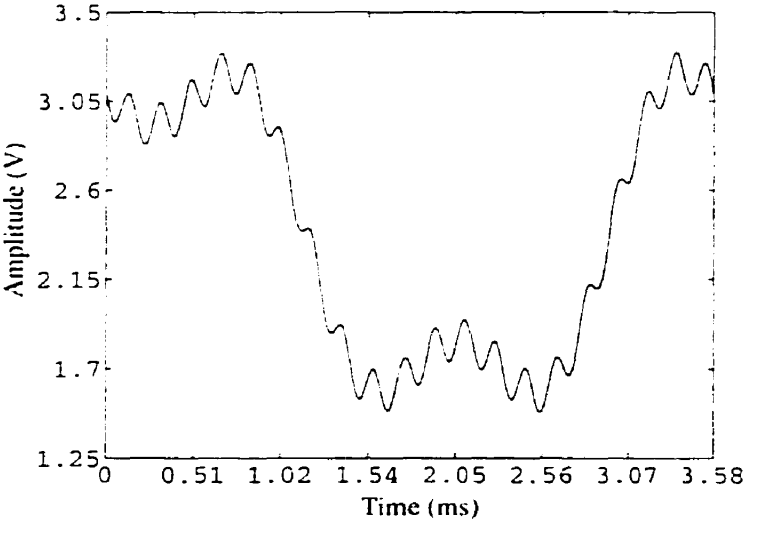

Figure 4.16 Captured three-tone waveform (B=10 bits).

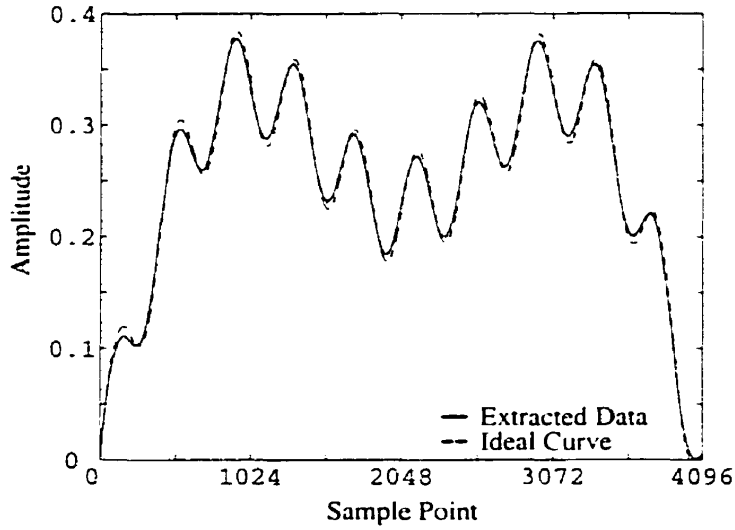

Figure 4.17 Comparison of the captured waveform with the ideal, discrete-time curve.

waveform extractor. Therefore, when a circuit under test is placed inside the loop, its transfer function can be properly isolated from that of the test system.

4.2.6 - Comparator Offset

As seen in Section 3.2, the offset error due to the comparator can be eliminated by a simple, double conversion procedure. The procedure consists of performing a complete

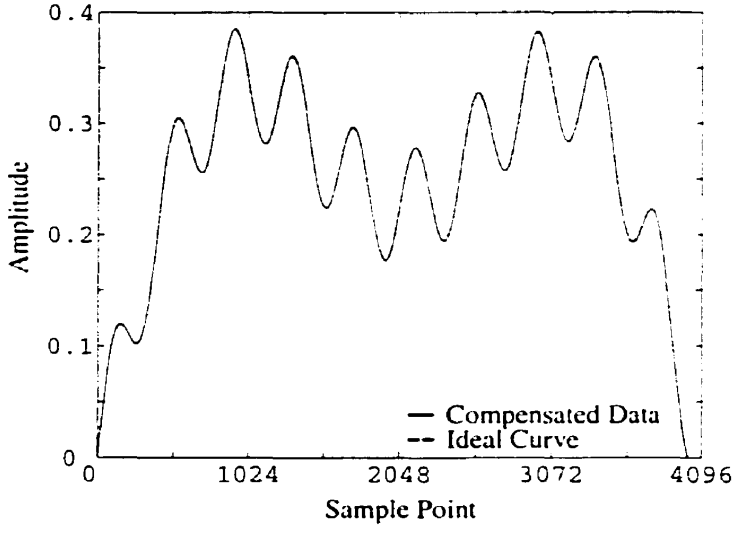

Figure 4.18 Comparison of the compensated waveform with the ideal, discrete-time curve - the differences between the two are of the order of 10^{-3}.

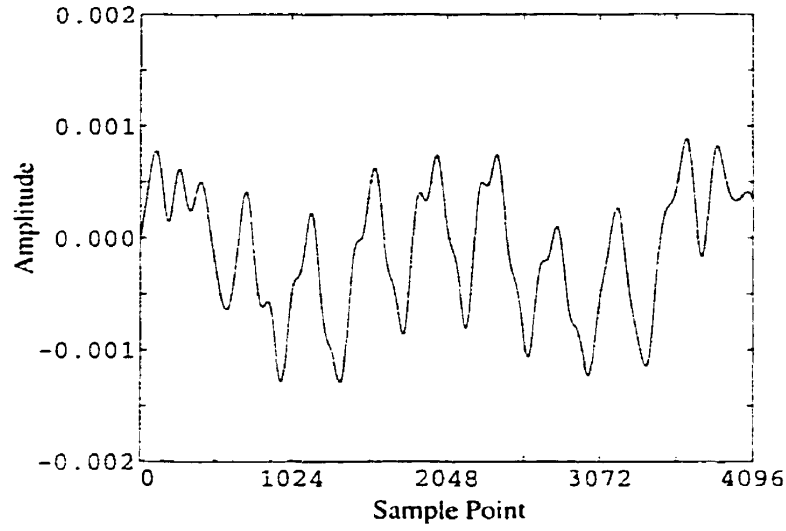

Figure 4.19 Error vector obtained by taking the difference between the compensated data, and the ideal curve.

conversion, swapping the inputs to the comparator, performing a second conversion, and then averaging the two digitized waveforms. Since an offset error is added at opposite polarities, it is effectively cancelled out by the averaging operation.

However, as will be discussed in the next section, the experimental setup was plagued with high levels of noise. Consequently, although it was possible to average out the offset error in the captured data sets, it was not possible to get an accurate DC measurement of the analog signal using a multimeter. Note that because of the common-mode nature of the noise, and the differential nature of the extraction process, most of the noise components were cancelled out during waveform capture.

Thus, in order to verify that the double conversion scheme is in fact extracting information about the offset voltage, an alternative method can be used, relying on a comparison with the known specifications of the discrete comparator. Essentially, rather than taking an average of the waveforms, one can take the difference. Let $V_{CAP1}[n]$ and $V_{CAP2}[n]$ represent the extracted waveforms where opposite input polarities have been used; the following equations can be written,

$$V_{CAP1}[n] = V_{an}[n] + V_{ofs} \tag{4.4}$$

$$V_{CAP2}[n] = V_{an}[n] - V_{ofs} \tag{4.5}$$

where $V_{an}[n]$ is the correct extracted version of the analog waveform. In order to isolate the V_{ofs} term, the difference voltage $V_{DIF}[n]$ can be computed:

$$V_{DIF}[n] = \frac{V_{CAP1}[n] - V_{CAP2}[n]}{2} \equiv V_{ofs} \tag{4.6}$$

Note that the above equation states that $V_{DIF}[n]$ is approximately equal to V_{ofs}; in fact, the accuracy of the approximation depends on the quantization error (amplitude resolution) introduced by the capture system. Consider the best resolution of the experimental board: 10 bits of amplitude resolution from a single 5V supply - the best resolution of the ADC is thus 4.88 mV. Next, an LM319A comparator was used with the board presented here, and its specifications state that the typical input offset voltage is 0.5 mV. Clearly, the capture system does not have enough resolution to measure such a small voltage level.

However, by taking the average over many samples, it is possible to obtain a better measurement of the offset voltage. Using the captured waveform of Figure 4.9 (for B=10 bits), and averaging the $V_{DIF}[n]$ waveform over all its sample points (8192 points), an offset voltage of 0.34 mV was obtained. This number compares well with the specified offset voltage of the LM319A comparator.

Therefore, although the common-mode noise did not allow for a measured verification of the capture system's ability to isolate the comparator's offset voltage, the alternative $V_{DIF}[n]$ approach did allow a verification with the device's specifications.

4.3 - Implementation with a Custom IC

As mentioned previously, the test core design described in Chapter 3 was sent for fabrication in a 0.5 µm CMOS process. The micrograph of the IC is shown in Figure 4.20; the complete design required an area of 3 mm^2 (including I/O pads), where the on-chip components of the Multi-Pass system occupy an area of 0.25 mm^2.

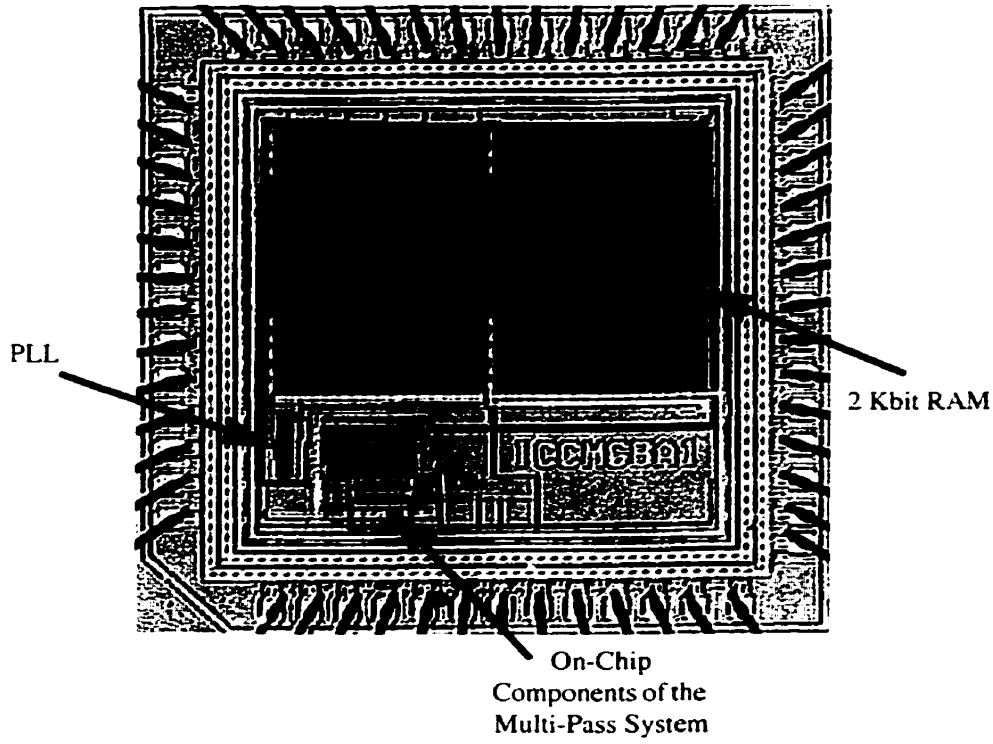

Figure 4.20 Micrograph of the prototype IC.

In order to test the prototype chip, a PCB board was then constructed to provide the necessary interface components. Figure 4.21 shows a picture of the experimental board that consists of the VXI interface, two voltage regulators (3.3 V and 5 V), a digital buffer (to convert TTL level signals from the VXI into CMOS level ones), a 12-bit DAC, a crystal oscillator (to provide an external clock), and a few DIP switches (to set the number of wait-states W, and to control the SW1 switch - see Figure 3.1).

The following subsections describe the steps that were taken to verify the functionality of the fabricated IC.

4.3.1 - The PLL

The first step in the debugging process involves the PLL circuit; it is a key component of the design since it provides the high speed clock that runs the entire IC. Essentially, the PLL makes use of a frequency divider (a factor of 128) in its feedback loop

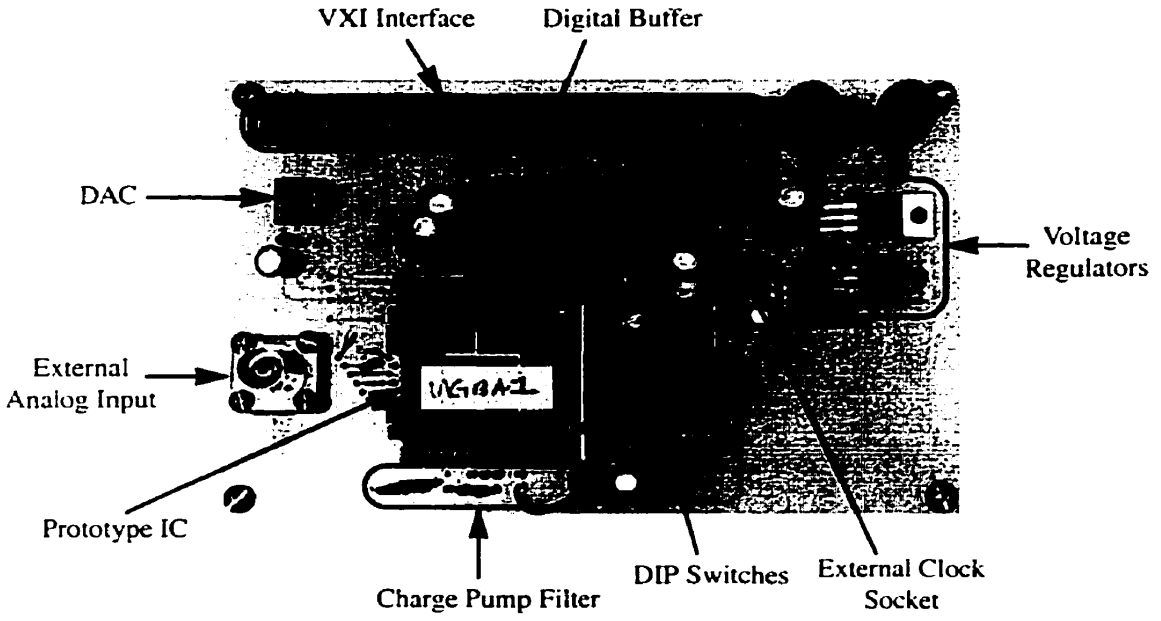

Figure 4.21 PCB board used to test the prototype IC.

to synchronize with a slower, off-chip clock (crystal oscillator). For a system clock frequency of 500 MHz, an external clock of roughly 4 MHz is required.

Since the design is intended to be clocked at a rate higher than the standard I/O pads allow, no direct taps of the PLL's output were taken off-chip. Consequently, in order to verify that the PLL is indeed operating correctly, it is necessary to examine an output waveform that is generated from the system clock. The CLK_OUT signal from the Multi-Pass system's output controller is an ideal candidate.

As described in Section 3.8, the frequency of the CLK_OUT signal depends on the system clock rate (or F_S), and the wait-state setting (W). Eqn (3.1) indicates that the frequency of the pulse signal P_1 is W+2 times slower than the system clock. Since the P_1 signal clocks the output controller, and since CLK_OUT is eight times slower than P_1, the frequency of CLK_OUT (denoted F_{CO}) should thus be observed to be,

$$F_{CO} = \frac{F_s}{8(W+2)} \tag{4.7}$$

The first attempt to run the PLL circuit proved unsuccessful; a few other IC's from the fabricated batch were then plugged into the board in order to check whether the problem was common to all of the prototype chips - the results were the same: no CLK_OUT signal was observable.

The next step was to rule out the possibility that the output controller was at fault. Using an external, 100 MHz clock signal, and a W=3 setting, the correct 2.5 MHz CLK_OUT signal was observed.

At this point, it seemed that the PLL itself was not functioning. Unfortunately, there weren't many circuit taps available for extensive debugging; the off-chip connections consisted of an RC lowpass filter, and a bias resistor, both for the charge pump section of the PLL. By setting up various DC conditions on these pins, it became clear that the ring oscillator's (used to implement the VCO) range of frequency operation was nowhere near the simulated range. Instead of operating around 500 MHz, the VCO was operating around 200 MHz.

The problem was further increased because of high levels of noise present on the PCB board. Due to the single-ended nature of the PLL design, the off-chip filter propagated the noise into the PLL loop, leading to an unstable circuit. A differential design would have been preferable, but it was not possible to implement one because of limited design time.

In short, the PLL circuit was not designed with enough flexibility (time limitations) to work with a VCO frequency range different from the one in simulation, nor was there a way for it to cope with the presence of noise.

4.3.2 - The Signal Generator

As described in Chapter 3, the on-chip signal generator consists of a RAM connected in a daisy chain configuration. The design can be verified in two steps: a shift-in/shift-out pattern match at low speed, followed by a looped test at high speed.

The first part of the RAM test was successful; a data stream was shifted into the daisy chain, and a correct pattern match was observed at the output. In addition, the simulated frequency spectrum of a test stimulus vector (Figure 4.4) was visible on the spectrum analyzer.

However, the second part of the RAM test failed; at high clocking rates (40 MHz and above), the preloaded data stream simply became corrupted, and then disappeared. Many attempts to load and loop various bit-streams at various clocking rates were made (on most of the IC's in the fabricated batch). In the end, it seemed as though the only way to explain the loss of data was that the setup and hold times for the D-type Flip-Flops in the chain were not sufficient.

Given that the technology is a high speed one, the problem could be due to loading effects; in other words, at high clocking rates, a Flip-Flop may not be able to drive another one without a buffer at its output. This argument is reinforced by the fact that the Multi-Pass system's output controller does work at high speeds, and it includes buffers at the output of its Flip-Flops.

Therefore, from the observed behaviour of the RAM, it is safe to say that the HSPICE models for the 0.5 μm CMOS process do not accurately represent the dynamic behaviour of the devices (including capacitive loading, whether it be parasitic, gate, or other). This is especially true given the fact that the RAM was constructed entirely with components from the standard h-cell library (provided with the technology files).

4.3.3 - The Multi-Pass Convertor

Although the on-chip signal generator did not function as expected, there was still the possibility to test the Multi-Pass system using an external, analog stimulus signal. An HP E1445A arbitrary function generator was used to generate a variety of input waveforms; a script file similar to the one in Appendix B.1 was then used to obtain the extracted data set. Note that in all cases, an external, 100 MHz clock signal was used as the system clock.

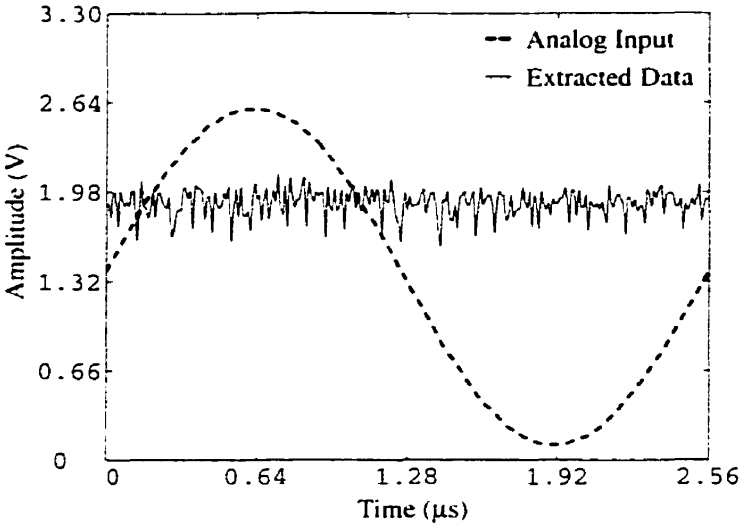

Figure 4.22 Extracted data obtained with a sine wave input (f=391 kHz, F_s=100 MHz, B=8 bits, N=256 points).

The captured results were all incomprehensible; there were no visible relationships to the analog inputs. Figure 4.22 shows an example of the kind of results that were obtained. In order to isolate the source of error, the following were considered: noise, sample-and-hold failure, comparator failure.

Given the complex nature of the Multi-Pass convertor's extraction process, it is preferable to work with a DC input signal for debugging purposes. Referring back to Figure 2.5, in the ideal case, the bits above the DC input level should all be zero, and the bits below the DC level should all be ones (i.e. a change of state occurs on the same level for all sample points). Consequently, the data stream coming out of the comparator should look like a perfect step response; anything different than a step would indicate the presence of noise in the system. Indeed, the observed comparator output consisted of a step with a noisy transition region. Figures 4.23 and 4.24 show the extraction results obtained for input DC levels of 0.75 V and 2.25 V respectively. It is clear from these figures that there is definitely noise present in the experimental setup; also, on a more positive note, the sample-and-hold and the comparator seem to be operating correctly - the sample-and-hold circuit is able to track the DC level.

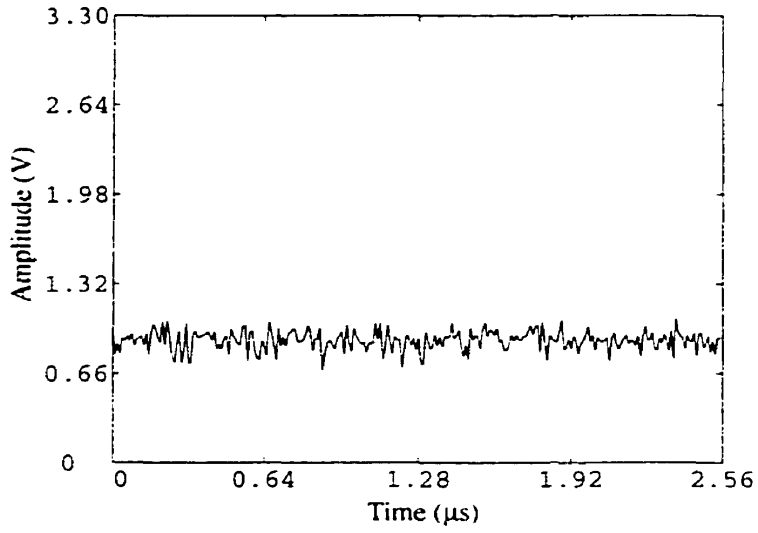

Figure 4.23 Extracted data obtained with a DC input of 0.75 V (F_s=100 MHz, B=8 bits, N=256 points).

A few attempts were made to reduce the system noise level. Pi filters were placed at the power supply inputs, but the noise still persisted. The board was then powered by a UPS battery - the results were the same. Due to extensive building renovations, all of the equipment in the lab was affected by high levels of noise on the power lines. Furthermore, because the prototype IC required an interface to the VXI system as well as a computer

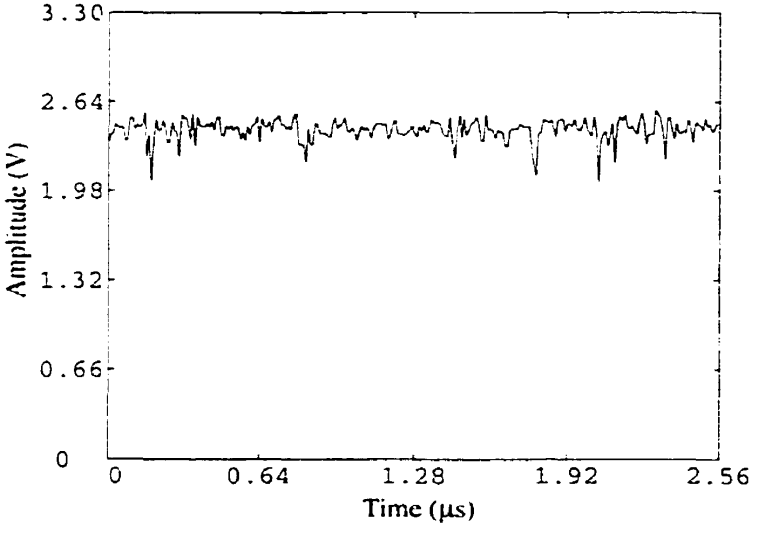

Figure 4.24 Extracted data obtained with a DC input of 2.25 V (F_s=100 MHz, B=8 bits, N=256 points).

terminal, it became clear that the noise was entering the board from many directions. In addition, there was no practical (and immediately feasible) way to connect all of the necessary equipment to an isolated power source.

Nonetheless, the presence of noise did not explain why the convertor was not able to extract even a simple sine wave; the results should have simply been a noisy, digitized sine wave. Given the observations made with the RAM, it is possible that once again, the dynamics of the sample-and-hold circuit were not correctly simulated. This would explain its operation at DC, whereas at the high speeds for which it was designed to operate, the circuit would fail - especially if capacitive effects were not modeled correctly, since capacitances are at the heart of the sample-and-hold circuit. At this point, it would have been convenient to bypass the sample-and-hold circuit, in order to verify the correct operation of the comparator (and confirm that the sample-and-hold is at fault); however, due to time constraints, this option was not designed into the prototype IC. On the other hand, a stand-alone comparator was integrated with the design; unfortunately, it was not connected to an on-chip sampling controller, and thus, requires an external one. It is also important to note here that A/D conversion is possible without a sample-and-hold circuit, provided that the sampling period T_s is much longer than the settling time of the comparator (1.5 ns - see Section 3.3).

In order to make use of the on-chip, stand-alone comparator, it is necessary to build an external sampling controller. This can be done quickly by programming an FPGA to monitor and control the extraction process. Figure 4.25 shows a flowchart that represents the operation of the FPGA controller. Appendix D contains a more detailed description of the controller's structure, in the form of modular VHDL codes. The VHDL description can be synthesized using the Synopsis CAD tool, and then compiled for an XC4010 FPGA using the XACT place-and-route software.

Using a VXI program similar to the one in Appendix C.1, the comparator data for a test sine wave was captured. The stored comparator information was then processed by a C program (Appendix C.2) to obtain the digitized waveform shown in Figure 4.26. The figure compares the extracted waveform to the analog input (captured using an HP

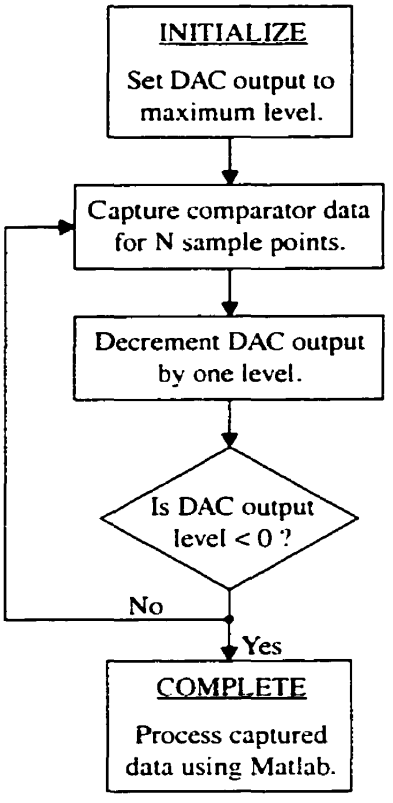

Figure 4.25 Flowchart for the FPGA sampling controller.

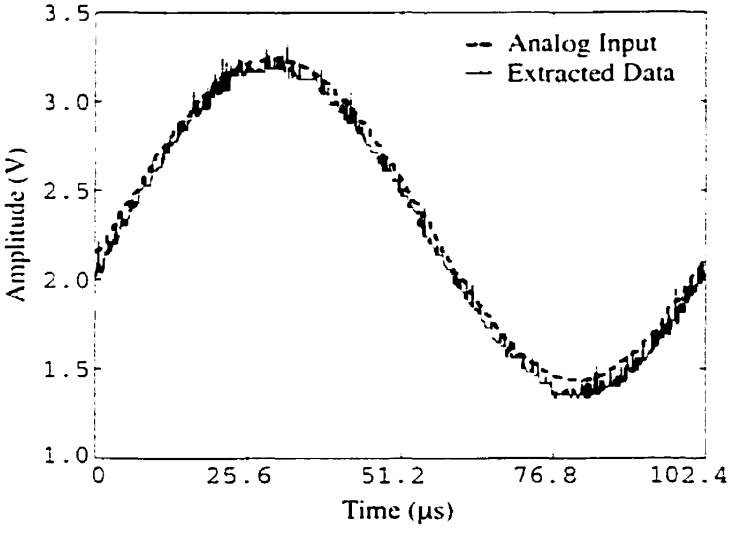

Figure 4.26 Waveform captured using the integrated, stand-alone comparator (f=9.77 kHz, F_s=10 MHz, B=10 bits, N=1024 points).

54510A digitizing oscilloscope). As expected, a fair amount of noise is present on the extracted data set, but the data does closely follow the analog input. Thus, from these results, one can conclude that the sample-and-hold circuit is the faulty part of the Multi-Pass convertor.

4.4 - Conclusion

To summarize, two experimental boards have been constructed and tested in order to verify the feasibility of the proposed test system. The first board was implemented using discrete components, and the second, using a custom IC.

The discrete board consisted of both software and hardware components: most of the complex computations were done in software in order to minimize the complexity of the hardware. It was shown that a complete test system (i.e. a signal generator, and a waveform extractor) can be constructed using a few, simple components. The programmability of the system was demonstrated in terms of generating arbitrary stimulus signals, as well as extracting waveforms at various amplitude resolutions and number of sample points. Finally, a calibration procedure was outlined in order to compute the system's inverse transfer function; such a transfer function can be used to eliminate the effects of the test system, and thus, isolate the circuit under test's output response.

The second board, implemented with a custom IC, displayed a number of component failures. Essentially, the key, high-speed components (PLL, RAM, and sample-and-hold) did not function. By applying various circuit conditions to the prototype IC, the debugging procedure led to the conclusion that the HSPICE models for the given process do not capture the correct dynamic behaviour of the devices. Due to building renovations, the noise from the power lines affected the equipment in the lab, resulting in additional design verification problems. On the other hand, the flexibility (configuration) and redundancy that was designed into the prototype IC allowed some partial results to be obtained (stand-alone comparator).

Therefore, the experimental results confirm that the Multi-Pass algorithm is a valid one, and that the RAM-based generator can provide any desired stimulus signal in an area-efficient manner. Thus, the proposed test system does meet the requirements for a practical test solution.

Chapter 5 - Conclusion

5.1 - Thesis Review

With the increasing complexity of mixed-signal integrated circuits, there is a growing need for integrable test solutions. Although a mixed-signal test core would be an ideal complement to existing test bus standards, as yet, the biggest challenge still remains the functional testing of the analog portion of a mixed-signal design.

A complete test core requires two major components: a stimulus generator, and a parameter extractor. In order to produce practical test solutions, these components must be area-efficient and robust, they must be able to function at high speeds, and they must be easy to connect to any existing on-chip circuitry.

A prime candidate for the stimulus generation component is a RAM-based signal generator. Such a generator repeats a finite portion of a PDM sequence in order to produce any arbitrary analog waveform. RAM-based generators are area-efficient, simple to design (a scan chain, and a 1-bit DAC), robust (mostly digital design), and can run at high speeds. Furthermore, since the analog signal is generated from a digital bit-stream, it is both stable and repeatable.

The detection (or capture) component of a mixed-signal test core represents a more difficult challenge because of the broad range of possible CUT output responses. In order to narrow the specifications of the parameter extractor, the work presented here focused on

steady-state type responses. For these kinds of responses, an ideal solution would be to design an on-chip network analyzer. However, a closer examination of the hardware requirements (FFT block) revealed that such an approach does not fall into the guidelines of a practical test solution. Consequently, it was decided that the on-chip component for parameter extraction would consist of an A/D convertor, and thus, the computational complexity would be moved over to either an existing on-chip DSP, or to external software.

Unfortunately, traditional A/D structures have an inherent tradeoff between area and speed; this works against the goal of obtaining both high speed performance and area efficiency. In Chapter 2, the Early Capture method was presented, and it was shown that by taking advantage of repetitive waveforms, it is possible to decouple a capture system's sampling resolution from the A/D's conversion time. In other words, repetitive waveforms eliminate the tradeoff between area and speed - the overall conversion time is increased, but there is no decrease in sampling resolution.

Next, an alternative A/D algorithm was presented: the Multi-Pass technique. it was shown that the Multi-Pass convertor makes use of the same on-chip components (sample-and-hold, and comparator) as the Early Capture system; however, the way in which it performs the conversion from analog to digital differs. Essentially, the Early Capture system makes use of a binary search on each sample, whereas the Multi-Pass convertor makes multiple comparison passes along the UTP of a signal to determine the level-crossing points for each sample. Given the test parameters B and N, a selection criterion was then derived to obtain the more optimal method.

Chapter 3 discussed the design details of a prototype test core that was fabricated in a 0.5 μm CMOS process. Included in the description were a high-speed comparator design, with a simple means of correcting its offset error (input swapping switches); a sample-and-hold design, with a V_{SS} to V_{DD} input voltage range (no biasing considerations required); a sample-and-hold controller, with a user selectable number of wait-states (comparator settling time); and an output controller, with a serial-to-parallel convertor (to slow down the output data rate), and an output synchronization clock generator (external

triggering). Also included in the design were a PLL, to generate the 500 MHz system clock, and a RAM, to implement the memory-based signal generator. In addition, a novel, integrable, highly-linear, and area-efficient voltage reference design was presented.

Chapter 4 presented the results that were obtained from two experimental boards. The first implemented the proposed test system using discrete components, the second made use of a custom IC. The discrete board consisted of both software and hardware components; the complex computations were done in software (PDM encoding, Multi-Pass data processing) in order to reduce the complexity of the hardware. The programmability of the system was demonstrated; the system's capability of generating arbitrary stimulus signals, as well as its ability to extract waveforms at various amplitude and sampling resolutions were shown. A calibration procedure was then outlined in order to compute the system's inverse transfer function; such a transfer function is needed to remove the effects of the test system on the captured response, thereby isolating the desired CUT output response.

The second board, implemented with a custom IC, did not function as expected; the critical, high-speed components such as the PLL, the RAM, and the sample-and-hold circuit were not operational. The debugging procedure revealed that the HSPICE models used in simulation did not capture the true dynamic behaviour of the devices. In addition, the presence of high levels of noise in the test environment made it clear that some improvements need to be made. However, the flexibility and redundancy that were designed into the prototype IC allowed some partial results to be obtained.

In summary, the overall simulation and experimental results serve to validate the Multi-Pass A/D algorithm; they also demonstrate that a RAM-based generator can provide any desired stimulus signal in an area-efficient manner. As required, the proposed test system meets the requirements for a practical test solution. Furthermore, the combination of the Multi-Pass convertor, and the RAM-based generator takes us one step closer to obtaining a fully integrable mixed-signal test core.

5.2 - Future Work

The experimental results obtained with the prototype IC suggest that there is still some work to be done in order to improve the proposed test core design. First, it is clear that a more flexible and robust PLL circuit must be employed. The circuit must be flexible with respect to process variations; for example, if the simulated operating frequency of the VCO differs from the one after fabrication, there should be some way of dealing with the problem. This can be done by adding taps at the output of the invertor stages, such that the number of invertors that make up the ring oscillator (VCO) can be varied - i.e. the change in the delay per stage can be compensated by adding or removing invertors in the loop. Evidently, the design must also include a means of determining the operating frequency of the VCO; this can be done by adding an extra ring oscillator with off-chip circuit taps for calibration purposes. Next, the PLL must be designed to be robust - it must be insensitive to noise. The circuit's noise immunity can be enhanced by using a differential design, and by moving the lowpass filter onto the IC.

The PLL was not the only component that was affected by noise; the Multi-Pass system's performance can also be improved by addressing the noise problems that were observed during experimentation. An immediate step that should be taken is to integrate the proposed voltage reference with the rest of the on-chip components of the convertor. The result would be less external components, and thus, less sources of noise. The Multi-Pass system can also be implemented in a differential manner; it is not difficult to extend the algorithm for use with a differential comparator. This approach would help to further eliminate the effects of common-mode noise. In addition, since the proposed voltage reference makes use of a 1-bit digital signal to produce an analog output, it can be easily converted into a differential design; by taking an inverted version of the digital output, a pair of complementary analog signals can be generated.

Finally, the issues pertaining to the RAM-based generator's reconstruction filter, and the Multi-Pass system's anti-aliasing filter need to be addressed; this is a necessary step in obtaining a fully integrable test core.

In conclusion, this thesis has presented the merger of some key ideas that have helped to provide the groundwork for a practical test system. There is still room for improvement, and the work should be continued with the following in mind: flexibility, robustness, and simplicity.

References

[1] A. Susin and G. W. Roberts, "Signal Generator Cores for Mixed-Signal Systems-On-Chip," *Proceedings of SBMICRO*, Curitiba, Brazil, August 1998.

[2] IEEE Standard 1149.1-1993a: "IEEE Standard Test Access Port and Boundary Scan Architecture," IEEE, 345 East 47th St., New York, NY, October 1993.

[3] K. P. Parker, "The Boundary-Scan Handbook," Kluwer Academic Publishers, Norwell, MA, September 1992.

[4] IEEE P1149.4 Mixed-Signal Test Bus Working Group, *http://stdsbbs.ieee.org/groups/1149/4/index.html.*

[5] M. F. Toner and G. W. Roberts, "A BIST Scheme for an SNR, Gain Tracking, and Frequency Response Test of a Sigma-Delta ADC," *IEEE Transactions on Circuits and Systems -- II: Analog and Digital Signal Processing*, vol. 41, no. 12, pp. 1-15, January 1995.

[6] G. W. Roberts, "Improving the Testability of Mixed-Signal Integrated Circuits," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 214-221, Santa Clara, May 1997.

[7] E. M. Hawrysh and G. W. Roberts, "An Integration of Memory-Based Analog Signal Generation into Current DFT Architectures," *Proceedings of the IEEE International Test Conference*, pp. 528-537, Washington D.C., October 1996.

[8] X. Haurie and G. W. Roberts, "Arbitrary-Precision Signal Generation for Bandlimited Mixed-Signal Testing," *Proceedings of the IEEE International Test Conference*, pp. 78-86, Washington D.C., October 1995.

[9] B.R. Veillette and G.W. Roberts, "Delta-Sigma Oscillators: Versatile Building Blocks," *International Journal of Circuit Theory and Applications*, vol. 25, pp. 407-418, 1997.

[10] X. Haurie and G. W. Roberts, "A Design, Simulation and Synthesis Tool For Delta-Sigma-Modulator-Based Signal Sources," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 715-718, Atlanta, Georgia, May 1996.

[11] B. Dufort and G.W. Roberts, "Signal Generation Using Periodic Single and Multi Bit Sigma-Delta Modulated Streams," *Proceedings of the IEEE International Test Conference*, pp. 396-405, Washington D.C., November 1997.

[12] B. Dufort and G.W. Roberts, "On-Chip Analog Signal Generator for Mixed-Signal Built-In Self-Test," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 549-552, Santa Clara, May 1998.

[13] N.C. Hu and O.K. Ersoy, "Fast Computation of Real Discrete Fourier Transform for any Number of Data Points," *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 1280-1292, November 1991.

[14] A.R. Varkonyi-Koczy, "A Recursive Fast Fourier Transformation Algorithm," *IEEE Transactions on Circuits and Systems -- II: Analog & Digital Signal Processing*, vol. 42, pp. 614-616, September 1995.

[15] I.R. Mactaggart and M.A. Jack, "A Single Chip Radix-2 FFT Butterfly Architecture Using Parallel Data Distributed Arithmetic," *IEEE Journal of Solid-State Circuits*, vol. 19, pp. 368-373, June 1984.

[16] E. Bidet, D. Castelain, and C. Joanblanq, "A Fast Single-Chip Implementation of 8192 Complex Point FFT," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 300-305, March 1995.

[17] A. B. Grebene, "Bipolar and MOS Analog Integrated Circuit Design," John Wiley & Sons, New York, 1984.

[18] B. Razavi, "Principles of Data Conversion System Design," IEEE Press, New York, 1995.

[19] K. Lofstrom, "Early Capture for Boundary Scan Timing Measurements." *Proceedings of the IEEE International Test Conference*, pp. 417-422, Washington, D.C., October 1996.

[20] M. A. Burns, "Undersampling Digitizer with a Sampling Circuit Positioned on an Integrated Circuit," US Patent 5578935, Texas, November 1996.

[21] J. P. Hayes, "Introduction to Digital Logic Design," Addison-Wesley Publishing Company, New York, 1993.

[22] A. V. Oppenheim, A. S. Willsky, and I. T. Young, "Signals and Systems," Prentice Hall, New Jersey, 1983.

Appendix A - HSPICE Netlists

A.1 - NPN Model for the 0.8 μm BiCMOS Process

```
* Canadian Microelectronics Corporation
* BiCMOS Design Kit V1.0 for Cadence Analog Artist
* August 23, 1995
*
* 0.8-micron BiCMOS HSPICE library for 5V bipolar models
*
* You should have the following options set in order to suppress the
* printing of the model parameters, to make the nominal temperature
* compatible with SPICE, and to make HSPICE use the same device models
* as SPICE 2G.6:
*
*    .OPTIONS NOMOD TNOM=27 DCAP=1
*
*********************************************************************
**
*
*    1X NPN WITH SINGLE BASE CONTACT CONTACTED WITH METAL 1
.MODEL NN51111X NPN
+         IS= 3.10E-18         BF= 1.03E+02         NF= 1.00E+00
+        VAF= 1.18E+02        IKF= 5.36E-03        ISE= 4.57E-18
+         NE= 1.50E+00    ISS= 1.00E-17
+         BR= 1.00E+00         NR= 1.00E+00        VAR= 5.50E+00
+        IKR= 0.00E+00        ISC= 0.00E+00         NC= 2.00E+00
+         RB= 5.76E+02        IRB= 0.00E+00        RBM= 5.76E+02
+         RE= 1.20E+01         RC= 5.94E+01
+        CJE= 1.50E-14        VJE= 8.00E-01        MJE= 2.67E-01
+         TF= 1.10E-11        XTF= 7.36E+02        VTF= 1.31E+00
+        ITF= 9.38E-02        PTF= 2.00E+01
+        CJC= 1.58E-14        VJC= 7.10E-01        MJC= 3.97E-01
+       XCJC= 5.10E-01         TR= 4.00E-09
+        CJS= 3.44E-14        VJS= 3.70E-01        MJS= 1.34E-01
+        XTB= 1.62E+00         EG= 1.11E+00        XTI= 5.82E+00
+         KF= 4.27E-09         AF= 2.07E+00         FC= 8.00E-01
**
**
```

A.2 - Bipolar Comparator Netlist

```
*** BiCMOS Comparator Simulation With Delta_Min = 0.0025 V ***
***              Using 0.8 um Process Models                 ***

.inc "NPN_0p8.inc"

*** Single 5V Power Supply ***

V1    VCC 0 5.0V

*** Comparator Input Waveforms ***

VDC1 VDC 0 2.5V
VI1 In1 VDC DC 0V
VI2 In2 VDC PWL
+0            -0.0025
+3.33e-9 -0.0025
+3.331e-9     0.0025
+6.67e-9  0.0025
+6.671e-9    -0.0025
+10.00e-9 -0.0025
+10.001e-9    0.0025
+13.33e-9  0.0025
+13.331e-9   -1.5000
+16.67e-9    -1.5000
+16.671e-9    0.0025
+20.00e-9  0.0025
+20.001e-9   -0.0025
+23.33e-9 -0.0025
+23.331e-9    1.5000
+26.67e-9     1.5000
+26.671e-9   -0.0025
+30.00e-9    -0.0025
+30.001e-9    0.0025

*** Current Mirror Reference ***

R1 VCC 3 10k
Q1 3 3 0 0 NN51111X

*** Gain Stage 1 ***

Q2 Out1 In1 2 0 NN51111X
Q3 Out2 In2 2 0 NN51111X
R2 VCC Out1  1k
R3 VCC Out2  1k
Q4 2 3 0 0 NN51111X

*** Gain Stage 2 ***

Q5 Out3 Out1 4 0 NN51111X
Q6 Out4 Out2 4 0 NN51111X
```

```
R4 VCC Out3  1k
R5 VCC Out4  1k
Q7 4 3 0 0 NN51111X

*** Gain Stage 3 ***

Q8 Out5 Out3 6 0 NN51111X
Q9 Out6 Out4 6 0 NN51111X
R6 VCC Out5  1k
R7 VCC Out6  1k
Q10 6 3 0 0 NN51111X

*** Gain Stage 4 ***

Q11 Out7 Out5 8 0 NN51111X
Q12 Out8 Out6 8 0 NN51111X
R8 VCC Out7  1k
R9 VCC Out8  1k
Q13 8 3 0 0 NN51111X

*** Level Shifter 1 ***

Q20  VCC Out7 Out13 0 NN51111X
Q21 Out13 3 0 0 NN51111X
Q22  VCC Out8 Out14 0 NN51111X
Q23 Out14 3 0 0 NN51111X

*** Level Shifter 2 ***

Q24  VCC Out13 Out15 0 NN51111X
Q25 Out15 3 0 0 NN51111X
Q26  VCC Out14 Out16 0 NN51111X
Q27 Out16 3 0 0 NN51111X

*** Level Shifter 3 ***

Q28  VCC Out15 Out17 0 NN51111X
Q29 Out17 3 0 0 NN51111X
Q30  VCC Out16 Out18 0 NN51111X
Q31 Out18 3 0 0 NN51111X

*** Level Shifter 4 ***

Q32  VCC Out17 Out19 0 NN51111X
Q33 Out19 3 0 0 NN51111X
Q34  VCC Out18 Out20 0 NN51111X
Q35 Out20 3 0 0 NN51111X

*** Gain Stage 5 ***

Q36 Out21 Out19 15 0 NN51111X
Q37 VCC Out20 15 0 NN51111X
R14 VCC Out21  13k
```

```
Qb9 15 3 0 0 NN51111X

*** Level Shifter 5 ***

Q38 VCC Out21 Out23 0 NN51111X
R15 Out23 0 10k

*** HSPICE Analysis ***

.op
.tran 0.02e-9 38e-9
.print V(In1) V(In2) V(Out23)

.options nomod tnom=27 dcap=1
.options INGOLD=2 NUMDGT=10
.option post
.end
```

A.3 - Device Models for the 0.5 μm CMOS Process

```
*N5BO SPICE BSIM1 (Berkeley Level 4; HSPICE Level 13) PARAMETERS

*NM1 PM1 DU1 DU2 ML1 ML2
*
*PROCESS=HP
*RUN=n5bo
*WAFER=42
*Gate-oxide thickness= 96 angstroms
*DATE=1-Feb-1996
*
*NMOS PARAMETERS
*

.MODEL CMOSN nmos level=13
+vfb0=-7.05628E-01,lvfb=-3.86432E-02, wvfb=4.98790E-02
+phi0=8.41845E-01, lphi=0.00000E+00, wphi=0.00000E+00
+k1=7.76570E-01,lk1=-7.65089E-04,wk1=-4.83494E-02
+k2=2.66993E-02, lk2=4.57480E-02,wk2=-2.58917E-02
+eta0=-1.94480E-03, leta=1.74351E-02,weta=-5.08914E-03
+muz=5.75297E+02,d10=1.70587E-001,dw0=4.75746E-001
+u00=3.30513E-01, lu0=9.75110E-02,wu0=-8.58678E-02
+u1=3.26384E-02, lu1=2.94349E-02,wu1=-1.38002E-02
+x2m=9.73293E+00,lx2m=-5.62944E+00, wx2m=6.55955E+00
+x2e=4.37180E-04,lx2e=-3.07010E-03, wx2e=8.94355E-04
+x3e=-5.05012E-05,lx3e=-1.68530E-03,wx3e=-1.42701E-03
+x2u0=-1.11542E-02,lx2u0=-9.58423E-04, wx2u0=4.61645E-03
+x2u1=-1.04401E-03, lx2u1=1.29001E-03,wx2u1=-7.10095E-04
+mus=6.92716E+02,lms=-5.21760E+01, wms=7.00912E+00
+x2ms=-6.41307E-02, lx2ms=1.37809E+00, wx2ms=4.15455E+00
+x3ms=8.86387E+00, lx3ms=2.06021E+00,wx3ms=-6.19817E+00
+x3u1=9.02467E-03, lx3u1=2.06380E-04,wx3u1=-5.20218E-03
+toxm=9.60000E-003, tempm=2.70000E+01, vddm=5.00000E+00
```

```
+cgdom=3.60204E-010,cgsom=3.60204E-010,cgbom=4.37925E-010
+xpart=1.00000E+000,dum1=0.00000E+000,dum2=0.00000E+000
+n0=1.00000E+000,ln0=0.00000E+000,wn0=0.00000E+000
+nb0=0.00000E+000,lnb=0.00000E+000,wnb=0.00000E+000
+nd0=0.00000E+000,lnd=0.00000E+000,wnd=0.00000E+000
*
*N+ diffusion::
*
+rshm=2.1,     cjm=3.500000e-04,     cjw=2.900000e-10,
+ijs=1e-08,     pj=0.8
+pjw=0.8,     mj0=0.44,     mjw=0.26,     wdf=0,     ds=0

* Gate Oxide Thickness is 96 Angstroms
*
*
*PMOS PARAMETERS
.MODEL CMOSP pmos level=13
+vfb0=-2.02610E-01, lvfb=3.59493E-02,wvfb=-1.10651E-01
+phi0=8.25364E-01, lphi=0.00000E+00, wphi=0.00000E+00
+k1=3.54162E-01,lk1=-6.88193E-02, wk1=1.52476E-01
+k2=-4.51065E-02, lk2=9.41324E-03, wk2=3.52243E-02
+eta0=-1.07507E-02, leta=1.96344E-02,weta=-3.51067E-04
+muz=1.37992E+02,dl0=1.92169E-001,dw0=4.68470E-001
+u00=1.89331E-01, lu0=6.30898E-02,wu0=-6.38388E-02
+u1=1.31710E-02, lu1=1.44096E-02, wu1=6.92372E-04
+x2m=6.57709E+00,lx2m=-1.56096E+00, wx2m=1.13564E+00
+x2e=4.68478E-05,lx2e=-1.09352E-03,wx2e=-1.53111E-04
+x3e=7.76679E-04,lx3e=-1.97213E-04,wx3e=-1.12034E-03
+x2u0=8.71439E-03,lx2u0=-1.92306E-03, wx2u0=1.86243E-03
+x2u1=5.98941E-04, lx2u1=4.54922E-04, wx2u1=3.11794E-04
+mus=1.49460E+02, lms=1.36152E+01, wms=3.55246E+00
+x2ms=6.37235E+00,lx2ms=-6.63305E-01, wx2ms=2.25929E+00
+x3ms=-1.21135E-02, lx3ms=1.92973E+00, wx3ms=1.00182E+00
+x3u1=-1.16599E-03,lx3u1=-5.08278E-04, wx3u1=9.56791E-04
+toxm=9.60000E-003, tempm=2.70000E+01, vddm=5.00000E+00
+cgdom=4.18427E-010,cgsom=4.18427E-010,cgbom=4.33943E-010
+xpart=1.00000E+000,dum1=0.00000E+000,dum2=0.00000E+000
+n0=1.00000E+000,ln0=0.00000E+000,wn0=0.00000E+000
+nb0=0.00000E+000,lnb=0.00000E+000,wnb=0.00000E+000
+nd0=0.00000E+000,lnd=0.00000E+000,wnd=0.00000E+000
*
*P+ diffusion::
*
+rshm=2,     cjm=9.452900e-04,     cjw=2.458300e-10,
+ijs=1e-08,     pj=0.85
+pjw=0.85,     mj0=0.439735,     mjw=0.237251,     wdf=0,     ds=0
```

A.4 - CMOS Comparator Netlist

```
*** CMOS Comparator Simulation With Delta_Min = 0.0015 V ***
***              Using 0.5 um Process Models                ***

.inc "CMOS_0p5.inc"

.global VDD
.global NB
.global VRB

*** Single 3.3V Power Supply ***

VSUP1 VDD 0 DC 3.3

*** Comparator Input Waveforms ***

VDC NDC 0 DC 2.3
V1 IN1 NDC DC 0
V2 IN2 NDC DC 0 PWL
+00.000e-9  -0.0015
+01.500e-9  -0.0015
+01.501e-9   0.0015
+03.000e-9   0.0015
+03.001e-9  -0.0015
+04.500e-9  -0.0015
+04.501e-9   0.0015
+06.000e-9   0.0015
+06.001e-9  -1.0000
+07.500e-9  -1.0000
+07.501e-9   0.0015
+09.000e-9   0.0015
+09.001e-9  -0.0015
+10.500e-9  -0.0015
+10.501e-9   1.0000
+12.000e-9   1.0000
+12.001e-9  -0.0015
+13.500e-9  -0.0015
+13.501e-9   0.0015
+15.000e-9   0.0015

*** Current Mirror Reference ***

VSUP2 VCB 0 DC 2.8

MB1 NB  NB VCB VDD CMOSP L=0.6e-6 W=4e-6
MB3 NB  NB  0   0  CMOSN L=0.6e-6 W=4e-6

*** Gain Stage 1 ***

MFL1 OUT1  0   VDD  VDD CMOSP L=0.6e-6 W=1e-6
MFL2 OUT2  0   VDD  VDD CMOSP L=0.6e-6 W=1e-6
MF1  OUT1 IN1  NF1   0  CMOSN L=0.6e-6 W=2e-6
```

```
MF2  OUT2 IN2  NF1   0  CMOSN L=0.6e-6 W=2e-6
MFSB NF1  NB    0    0  CMOSN L=0.6e-6 W=4.5e-6

*** Gain Stage Subcircuit ***

.subckt gain_stage 1 2 3 4
ML1  VDD   0   3   VDD CMOSP L=0.6e-6 W=1e-6
ML2  VDD   0   4   VDD CMOSP L=0.6e-6 W=1e-6
M1   3     1   N1   0  CMOSN L=0.6e-6 W=2e-6
M2   4     2   N1   0  CMOSN L=0.6e-6 W=2e-6
MSB  N1   NB   0    0  CMOSN L=0.6e-6 W=4.5e-6
.ends gain_stage

*** Gain Stages 2 to 9 ***

X2 Out1 Out2 Out3 Out4  gain_stage
X3 Out3 Out4 Out5 Out6  gain_stage
X4 Out5 Out6 Out7 Out8  gain_stage
X5 Out7 Out8 Out9 Out10 gain_stage
X6 Out9 Out10 Out11 Out12 gain_stage
X7 Out11 Out12 Out13 Out14 gain_stage
X8 Out13 Out14 Out15 Out16 gain_stage
X9 Out15 Out16 Out17 Out18 gain_stage

*** Buffer Stage 1 ***

MI1a VDD   Out18 OutI1  VDD CMOSP L=0.6e-6 W=1.2e-6
MI1b OutI1 Out18   0     0  CMOSN L=0.6e-6 W=0.8e-6

*** Buffer Stage 2 ***

MI2a VDD   OutI1 OutI2  VDD CMOSP L=0.6e-6 W=1.6e-6
MI2b OutI2 OutI1   0     0  CMOSN L=0.6e-6 W=1e-6

*** Buffer Stage 3 ***

MI3a VDD   OutI2 OutI3  VDD CMOSP L=0.6e-6 W=3.2e-6
MI3b OutI3 OutI2   0     0  CMOSN L=0.6e-6 W=2e-6

*** HSPICE Analysis ***

.op
.tran 0.01ns 18ns

.options INGOLD=2 NUMDGT=10
.option post
.print TRAN V(In1) V(In2) V(OutI3)
.end
```

A.5 - Sample-and-Hold Netlist

```
*** SAMPLE-AND-HOLD DESIGN With Step_Inc = 0.005 V ***
***          Using 0.5 um Process Models            ***

.inc "CMOS_0p5.inc"

.global VDD

*** Single 3.3V Power Supply ***

VSUP  VDD 0 DC 3.3

*** Sample-and-Hold Input Waveform ***

V1 NDC  0  DC 3.3V
V2 IN  NDC DC 0 PWL
+0.500e-9  -0.000
+4.500e-9  -0.000
+4.501e-9  -0.005
+8.500e-9  -0.005
+8.501e-9  -0.010
+12.500e-9  -0.010
+12.501e-9  -0.015
+16.500e-9  -0.015
+16.501e-9  -0.020
+20.500e-9  -0.020
+20.501e-9  -0.025
+24.500e-9  -0.025

*** Most of the piece-wise linear sequence is omitted ***
*** here for the sake of brevity.                      ***

+2620.501e-9  -3.275
+2624.500e-9  -3.275
+2624.501e-9  -3.280
+2628.500e-9  -3.280
+2628.501e-9  -3.285
+2632.500e-9  -3.285
+2632.501e-9  -3.290
+2636.500e-9  -3.290
+2636.501e-9  -3.295
+2640.500e-9  -3.295
+2640.501e-9  -3.300
+2644.500e-9  -3.300

*** Sample-and-Hold Control Signals ***

VP1 P1 0 DC 3.3V PULSE 0V 3.3V 1.00e-9 1e-12 1e-12 1.75e-9 4e-9
VP2 P2 0 DC 3.3V PULSE 0V 3.3V 3.00e-9 1e-12 1e-12 1.75e-9 4e-9

VNP1 NP1 0 DC 3.3V PULSE 3.3V 0V 1.00e-9 1e-12 1e-12 1.75e-9 4e-9
VNP2 NP2 0 DC 3.3V PULSE 3.3V 0V 3.00e-9 1e-12 1e-12 1.75e-9 4e-9
```

```
*** Sample-and-Hold Circuit ***

MS1a   IN   NP1   OUT1 VDD CMOSP L=0.6u W=20u
MS1b   IN    P1   OUT1   0  CMOSN L=0.6u W=20u
C1    OUT1   0    0.4e-12
MC1a    1    0    VDD  VDD CMOSP L=0.6u W=1u
MC1b    1  OUT1   0    VDD CMOSP L=0.6u W=4u

MS2a OUT1 NP2   OUT2 VDD CMOSP L=0.6u W=20u
MS2b OUT1  P2   OUT2   0  CMOSN L=0.6u W=20u
MS3a  0    P1   OUT2   0  CMOSN L=0.6u W=20u
MS3b  0   NP1   OUT2 VDD CMOSP L=0.6u W=20u
C2   OUT2  0    0.4e-12
MC2a  OUT  0    VDD  VDD CMOSP L=0.6u W=1u
MC2b  OUT OUT2   0    VDD CMOSP L=0.6u W=4u

*** HSPICE Analysis ***

.op
.tran 0.025ns 2644ns

.options INGOLD=2 NUMDGT=10
.option post
.print TRAN V(IN) V(OUT1) V(OUT2) V(OUT)
.end
```

Appendix B - Matlab Script Files

B.1 - Multi-Pass Controller Script File

```
%%%          EXTRACTED WAVEFORM BUILDER          %%%

% Load comparator data obtained from simulation. %

load DataStream

% Construct a position vector based on the number of clock %
% cycles required to read one sample. The position vector  %
% models the sample skipping behaviour of the read/compare %
% operation for each level.                                %

n = [ 0:63 ];
Sample_Cycles = 5;
Position_Vector=rem( n * Sample_Cycles , 64 );

% Use the position vector to reorganize the data according %
% to the correct sample positions.                         %

n = 1:64:length( DataStream );
bins = zeros( length(n) , 64 );
for i = 1:64
  ind = find( Position_Vector == ( i-1 ) );
  bins(:,i) = DataStream( 1 , n + ( ind-1 ) )';
end

% Determine, for each sample point, the level number at which %
% a change of state occured in the comparator's output.       %

levelnum = ones(1,64) * 64;
for i = 1:64
  ind = find( bins(:,i) == 0 );
  if length( ind ) > 0
    levelnum(1,i) = ind( 1 );
  end
end

% Assign a quantization value to each level number, and then %
% to each sample point in order to construct the digized     %
% waveform.                                                  %

n = 1:64;
```

```
qlevel = 3.3 - ( 3.3 / 64 * n );
QWaveform=qlevel(levelnum);

% Save digitized waveform. %

save Extracted QWaveform
```

B.2 - Clock Jitter Simulation Script File

```
%%%       CLOCK JITTER SIMULATOR       %%%

% Set the amplitude resolution and number of %
% sampling points.                           %

M=10;
N=8192;

% Create sine wave. %

n=1:N;
wave=(0.75*sin(2*pi*64/N*n)+1)./2;

% Create jitter offset vector. %

max_levels=2^M;
jit=fix(randn(1,max_levels));
jit=jit+abs(min(jit));

% Convert waveform. %

comp_out=zeros(1,N);
dl=1/max_levels;
data=ones(max_levels,N);
for i=1:max_levels;
  level=1-dl*i;
  jwave=wave(rem(n+jit(i),N)+1);
  tmp=sign(jwave-level);
  tmp=sign(tmp+0.25);
  data(i,:)=tmp./2+0.5;
  fprintf('Current Extraction Level = %i\n',i);
  drawnow;
end

bins=zeros(1,N);

for i=1:N
  tmp=find(data(:,i)>0);
  bins(i)=tmp(1);
end
```

Appendix C - C Code

C.1 - VXI Control Program for the Discrete Board

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "ieeeio.h"

#define MAX_RES          1024
#define MAX_N            8192
#define MAX_DAC_LEVELS   4096
#define SAMPLE_CYCLES      13

FILE *input, *output, *temp;

/* Procedure wait_for_vxi stops program execution until the VXI bus
   completes its current RUN cycle.                                  */

void wait_for_vxi ()
{
  char status[256],command[256];

  strcpy(status,"");

  /* Wait until the status register is set to +256 indicating that the
     device has stopped and the sequence is complete.                  */

  while ( strcmp(status,"+256\n") != 0 )
    {
      ieeewt("OUTPUT 0917;STAT:OPER:COND?\nENTER 0917\n");
      ieeerd(status);

    }
}

/* Procedure clear_errors clears the error buffer of the VXI system. */

void clear_errors()
{

  int i=1;
  int d=2;
  char response[128];
```

```
  do {
    ieeeprtf("output 0917;system:error?\n");

    ieeeprtf("enter 0917\n");
    while(state()==0)
      {
      i=d;
      d=i;
      }

    ieeerd(response);

  } while(!strstr(response,"No error"));

  printf("\n\n*** Cleared All Errors From Previous Users. ***\n\n");
}

/* Procedure report_errors lists the VXI errors that occurred during
   program execution.                                                */

void report_errors()
{

  int i=1;
  int d=2;
  char response[128];

  do {
    ieeeprtf("output 0917;system:error?\n");

    ieeeprtf("enter 0917\n");
    while(state()==0)
      {
      i=d;
      d=i;
      }

    ieeerd(response);
    printf("\n     Error Check: %s\n",response);

  } while(!strstr(response,"No error"));
}

/* Procedure num2str converts a number into a string. */

void num2str( number,strnum )
     long number;
     char strnum[32];
{
  long dum;
  int i,count=0,temp,temp2;
  char tempstr[32];
```

```
  dum = number;
  strcpy(tempstr,"");

  while (dum>0)
    {
      temp2=dum/10;
      temp = dum - temp2*10;
      tempstr[count] = 48 + temp;
      dum=dum / 10;
      count++;
    }
  for (i=0; i<count; i++)
     strnum[i] = tempstr[count-i-1];
  strnum[count]='\0';
  if (count==0)
    strcpy(strnum,"0");

}

/* Procedure num2bin converts a number into its bnary form. */

void num2bin( number, strnum, pad )
     long number,pad;
     char strnum[32];
{
  int i;
  long q,r;
  char tempstr[32];

  q=number;
  r=0;
  i=0;

  while( q > 0 )
    {
      r = q % 2;
      q = q / 2;
      if ( r == 1 )
        strnum[i] = '1';
      else
      strnum[i] = '0';
      i++;
    }
  for( i=i; i<pad; i++ )
    strnum[i] = '0';
  strnum[pad] = '\0';

  strcpy(tempstr,strnum);
  for ( i=0; i<pad; i++ )
    strnum[i]=tempstr[pad-1-i];
}

/* Procedure Set_DAC generates the necessary control signals that
```

```
   are required to set the DAC output to the desired level.        */

void Set_DAC(current_level)
     int current_level;
{
  char commandline[256],bit_stream[256],str[32],last_bit;
  int i;

  ieeewt("OUTPUT 0917;TIM:CYCLE dac_clock\n");
  strcpy(commandline,"OUTPUT 0917;TIM:CYCL:CONT1:WAV 0");
  num2bin(current_level,str,12);
  strcpy(bit_stream,str);
  strcat(bit_stream,"000000000000");
  /* printf("<%s> [%i]\n",bit_stream,strlen(bit_stream)); */
  last_bit = '0';
  for (i=1; i<=strlen(bit_stream); i++)
    {
      if (bit_stream[i-1] != last_bit)
        {
          num2str(i*4-2,str);
          strcat(commandline,",");
          strcat(commandline,str);
          last_bit = bit_stream[i-1];
        }
    }
  strcat(commandline,"\n");
  ieeewt(commandline);
  ieeewt("OUTPUT 0917;SEQ dac\n");
  /* ieeewt("OUTPUT 0917;TIM:CYCL dac_clock\n"); */
  ieeewt("OUTPUT 0917;RUN\n");
  wait_for_vxi();
}

/* Procedure power calculates the number "base^exp". */

int power( base, exp )
  int base,exp;
{
  int mult=1;
  int i;

  for (i=1; i<=exp; i++)
    mult*=base;
  return(mult);
}

/* Procedure process uses the comparator data to construct the
   digitized waveform. It begins by reshuffling the data according
   to the number of sampling cycles required for each point, and
   then determines at which level a change of state occurred, in
   order to assign the appropriate quantization value.             */

process()
```

```
{

  int i,num_levels,scan_up,byte;
  int N,dig,len,level,dlevel;
  int data[MAX_N],comp_out[MAX_N];

  /* Open Data I/O Files. */

    if ( ( input = fopen("vxi.tmp","r") ) == NULL )
      {
      printf("\n\n    Error Opening Input File.    \n\n");
      return(0);
      }
    if ( ( output = fopen("vxioutput","w") ) == NULL )
      {
      printf("\n\n    Error Creating Output File.    \n\n");
      return(0);
      }

  /* Get Header Info From Input File. */

    fscanf(input,"%i %i %i\n",&N,&num_levels,&scan_up);

  /* Initialize Working Variables. */

    level=MAX_DAC_LEVELS;
    dlevel=MAX_DAC_LEVELS/num_levels;
    for(i=0; i<MAX_N; i++)
      data[i]=level;
    if(scan_up==0)
      {
      for(i=0; i<MAX_N; i++)
        comp_out[i]=1;
      }
    else
      {
      for(i=0; i<MAX_N; i++)
        comp_out[i]=0;
      }

  /* Multi-Pass Data Processing Loop. */

    printf("\n   Processing Comparator Data ... ");

    while( (byte=fgetc(input)) != EOF )
      {
      level-=dlevel;
      dig=fgetc(input)-48;
      len=0;
      for(i=dig-1; i>=0; i--)
        len+=power(10,i)*(fgetc(input)-48);
      for(i=0; i<len-MAX_N; i++)
        byte=fgetc(input);
```

```
      for(i=0; i<MAX_N; i++)
        {
          byte=fgetc(input);
          dig=byte % 2;
          if(comp_out[i]!=dig)
            {
            comp_out[i]=dig;
            data[i]=level;
            }
        }
      byte=fgetc(input);     /* New Line Character */
      }
    printf("complete.\n");

  /* Reshuffle Sample Points According to SAMPLE_CYCLES. */

    for(i=0; i<MAX_N; i++)
      comp_out[i]=data[i];
    for(i=0; i<MAX_N; i++)
      data[(i*SAMPLE_CYCLES) % MAX_N]=comp_out[i];

  /* Generate Output File. */

    printf("\n   Generating Output File ... ");

    fprintf(output,"%% %i %i %i %%\n",MAX_N,num_levels,scan_up);
    for(i=0; i<MAX_N; i++)
      fprintf(output,"%i\n",data[i]);
    fclose(output);
    printf("complete.\n");
}

/* Procedure main gets the test parameters from the user, loads the
   PDM vector obtained from Matlab, and sets up the necessary
   control waveforms on the VXI bus. It then runs the capture cycle
   and records the output of the comparator for post-processing.    */

main ()
{

  char vxi_temp[512000],vxi_string[512000],commandline[256];
  char line[256],str[32];
  int i,pdm_length=16500,pdm_needed=16500,byte;
  int amp_res,num_levels,dlevel,level,scan_up=0;

  printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

  /* Open Data I/O Files. */

    if ( ( input = fopen("vxiinput","r") ) == NULL )
      {
        printf("\n\n    Error Opening Input File.    \n\n");
        return(0);
```

```
    }

  if ( ( temp = fopen("vxi.tmp","w") ) == NULL )
    {
      printf("\n\n    Error Creating Temporary File.    \n\n");
      return(0);
    }

/* Initialize VXI Bus */

  if ( ieeeinit() == -1 )
    {
      printf("\n\n    Error Initializing VXI System.    \n\n");
      return(0);
    }
  else
    {
    ieeewt("HELLO\n");
    ieeerd( vxi_string );
    printf("\n\n%s\n\n", vxi_string);
    }

  clear_errors();

/* Get Test Parameters From User. */

  printf("\n   Enter Amplitude Resolution (MAX 10 bits) : ");
  scanf("%i",&amp_res);
  if(amp_res<1)
    amp_res=1;
  else if(amp_res>10)
    amp_res=10;
  printf("\n   Amplitude Resolution Set To %i bits.\n",amp_res);
  num_levels=power(2,amp_res);
  dlevel=MAX_DAC_LEVELS/num_levels;

  printf("\n   Enter Polarity Scheme (0/1) : ");
  scanf("%i",&scan_up);
  if ((scan_up!=0) & (scan_up!=1))
    scan_up=0;
  printf("\n   Polarity Scheme Set To %i.\n",scan_up);

/* Scan Input File and Create PDM Command in TEMP File. */

  printf("\n   Total PDM Vectors Required = %i\n",pdm_needed);
  printf("\n   Scanning Input File ... ");

  pdm_length=0;
  fscanf(input,"%s\n",line);
  while ((strcmp(line,"END") != 0) & (pdm_length < pdm_needed))
    {
      pdm_length++;
      fprintf(temp,",%s",line);
```

```
        fscanf(input,"%s\n",line);
      }
    fclose(input); fclose(temp);
    printf("complete.\n");
    printf("\n   Total PDM Vectors Found = %i\n",pdm_length);

    if (pdm_length<pdm_needed)
      {
      printf("\n   Not Enough PDM Vectors. Conversion Not
Possible...\n\n");
      return;
      }

    if ( ( temp = fopen("vxi.tmp","r") ) == NULL )
      {
        printf("\n\n    Error Opening Temporary File.    \n\n");
        return(0);
      }
    fscanf(temp,"%s",vxi_string); fclose(temp);

    /*
    printf("\n   Reading Input File ... ");
    for(i=0; i<33007; i++)
      {
      byte=fgetc(input);
      vxi_string[i]=byte;
      }
    vxi_string[33007]='\0';
    printf("complete.\n");
    */

  /* Reset Digital I/O Card and Define Sequences */

    ieeewt("OUTPUT 0917;*RST;FORM HEX\n");
    ieeewt("OUTPUT 0917;FORM HEX\n");
    ieeewt("OUTPUT 0917;TIM:CONT ON\n");
    ieeewt("OUTPUT 0917;SEQ:DEL:ALL\n");

    /* ieeewt("OUTPUT 0917;SEQ:DEF main,16500\n"); */
    strcpy(commandline,"OUTPUT 0917;SEQ:DEF main,");
    num2str(pdm_needed,str);
    strcat(commandline,str);
    strcat(commandline,"\n");
    ieeewt(commandline);

    ieeewt("OUTPUT 0917;SEQ:DEF dac,4\n");

  /* Define Pin Groups */

    ieeewt("OUTPUT 0917;GRO:DEL:ALL\n");
    ieeewt("OUTPUT 0917;SEQ main\n");
    ieeewt("OUTPUT 0917;GRO:DEF data_in, (@1(0))\n");
    ieeewt("OUTPUT 0917;GRO:DEF pdm_out, (@1(3,2))\n");
```

```
  ieeewt("OUTPUT 0917;GRO data_in\n");
  ieeewt("OUTPUT 0917;GRO:MODE RESP\n");
  ieeewt("OUTPUT 0917;RESP:CLOC:SOUR INT1\n");
  ieeewt("OUTPUT 0917;RESP:COMP 0\n");
  ieeewt("OUTPUT 0917;GRO pdm_out\n");
  ieeewt("OUTPUT 0917;GRO:MODE STIM\n");
  ieeewt("OUTPUT 0917;STIM:CLOC:SOUR INT1\n");

/* Create Timing Cycles */

  ieeewt("OUTPUT 0917;TIM:RES 3.125E-008\n");
  ieeewt("OUTPUT 0917;TIM:CYCL:DEL:ALL\n");
  ieeewt("OUTPUT 0917;TIM:CYCLE:DEF main_clock,182\n");
  ieeewt("OUTPUT 0917;TIM:CYCLE:DEF dac_clock,100\n");

/* Associate Sequences with Timing Cycles */

  ieeewt("OUTPUT 0917;SEQ MAIN\n");
  /* ieeewt("OUTPUT 0917;TIM:CYCL:SEQ:REP 0,16500,MAIN_CLOCK\n"); */
  strcpy(commandline,"OUTPUT 0917;TIM:CYCL:SEQ:REP 0,");
  num2str(pdm_needed,str);
  strcat(commandline,str);
  strcat(commandline,",MAIN_CLOCK\n");
  ieeewt(commandline);

  ieeewt("OUTPUT 0917;SEQ DAC\n");
  ieeewt("OUTPUT 0917;TIM:CYCL:SEQ:REP 0,4,DAC_CLOCK\n");

/* Send PDM Data. */

  printf("\n   Sending PDM Vectors ... ");

  ieeewt("OUTPUT 0917;SEQ main\n");
  ieeewt("OUTPUT 0917;GRO pdm_out\n");
  strcpy(vxi_temp,"OUTPUT 0917;STIM:PATT:SEQ:PART 0");
  /*  strcpy(vxi_temp,"OUTPUT 0917;STIM:PATT:SEQ:FULL "); */
  strcat(vxi_temp,vxi_string);
  strcat(vxi_temp,"\n");
  ieeewt(vxi_temp);
  printf("complete.\n");

  /*    printf("\n   Verifying PDM Vectors ... ");
  strcpy(vxi_string,"");
  ieeewt("OUTPUT 0917;STIM:PATT:SEQ:PART? 0,2048\n");
  ieeewt("ENTER 0917\n");
  ieeerd(vxi_string);
  printf("\n\n%s",vxi_string);
  return;    */

/* Setup Control Waveforms for Timing Cycle MAIN_CLOCK */

  ieeewt("OUTPUT 0917;TIM:CYCLE main_clock\n");
  ieeewt("OUTPUT 0917;TIM:CONT:STAT ON\n");
```

```
    ieeewt("OUTPUT 0917;TIM:CYCL:CONT0:WAV 1\n");
    ieeewt("OUTPUT 0917;TIM:CYCL:CONT1:WAV 0\n");
    ieeewt("OUTPUT 0917;TIM:CYCL:CONT2:WAV 1\n");

    ieeewt("OUTPUT 0917;TIM:CYCL:CONT4:WAV
0,7,14,21,28,35,42,49,56,63,70,77,84,91,98,105,112,119,126,133,140,147,
154,161,168,175\n");
    ieeewt("OUTPUT 0917;TIM:CYCL:CONT5:WAV 0,14\n");

    if( scan_up==0)
      {
      ieeewt("OUTPUT 0917;TIM:CYCL:CONT6:WAV 1\n");
      ieeewt("OUTPUT 0917;TIM:CYCL:CONT7:WAV 0\n");
      }
    else
      {
      ieeewt("OUTPUT 0917;TIM:CYCL:CONT6:WAV 0\n");
      ieeewt("OUTPUT 0917;TIM:CYCL:CONT7:WAV 1\n");
      }

  /* Setup Control Waveforms for Timing Cycle DAC_CLOCK */

    ieeewt("OUTPUT 0917;TIM:CYCLE dac_clock\n");
    ieeewt("OUTPUT 0917;TIM:CONT:STAT ON\n");

    ieeewt("OUTPUT 0917;TIM:CYCL:CONT0:WAV
1,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,5
0,52,54,56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96\
n");

    ieeewt("OUTPUT 0917;TIM:CYCL:CONT2:WAV 1,2,98\n");

    ieeewt("OUTPUT 0917;TIM:CYCL:CONT4:WAV 0\n");
    ieeewt("OUTPUT 0917;TIM:CYCL:CONT5:WAV 0\n");

    ieeewt("OUTPUT 0917;TIM:CYCL:CONT6:WAV 0\n");
    ieeewt("OUTPUT 0917;TIM:CYCL:CONT7:WAV 0\n");

  /* Multi-Pass Capture Sequence. */

    if ( ( temp = fopen("vxi.tmp","w") ) == NULL )
      {
        printf("\n\n    Error Creating Temporary File.    \n\n");
        return(0);
      }
    fprintf(temp,"%i %i %i\n",MAX_N,num_levels,scan_up);
    printf("\n");
    for( level=(num_levels-1)*dlevel; level>=0; level-=dlevel )
      {
      Set_DAC(level);

      printf("     Scanning Level :  %4i",level);
```

```
      printf("          Run"); fflush(stdout);
      ieeewt("OUTPUT 0917;SEQ main\n");
      ieeewt("OUTPUT 0917;RUN\n");
      wait_for_vxi();

      printf(" - Read");  fflush(stdout);
      /*    strcpy(vxi_string,"");      */
      ieeewt("OUTPUT 0917;GRO DATA_IN\n");
      ieeewt("OUTPUT 0917;RESP:PATT:SEQ:FULL?\n");
      ieeewt("ENTER 0917\n");
      ieeerd(vxi_string);

      printf(" - Write\n");  fflush(stdout);
      fprintf(temp,"%s",vxi_string);
      }
    fclose(temp);

    process();

    report_errors();
}
```

C.2 - Multi-Pass Data Processing Program

```
#include <stdio.h>
#include <string.h>

#define MAX_N           1024
#define MAX_DAC_LEVELS   256
#define SAMPLE_CYCLES      1

FILE *input, *output;

/* Procedure power calculates the number "base^exp". */

int power( base, exp )
  int base,exp;
{
  int mult=1;
  int i;

  for (i=1; i<=exp; i++)
    mult*=base;
  return(mult);
}

/* Procedure main uses the comparator data to construct the
   digitized waveform according to the Multi-Pass algorithm.  */

main ( argc, argv )
 int argc;
```

```
 char *argv[];
{

  int i,j,byte,level,dlevel;
  int num_levels=MAX_DAC_LEVELS;
  int N;
  int data[MAX_N],comp_out[MAX_N];

  /* Get Commandline Arguments - Number of Sample Points. */

    if (argc<2)
      {
        printf("\n\nNot enough arguments..\n\n");
        exit(0);
      }
    N=atoi(argv[1]);
    printf("\nM=[%i]  N=[%i]\n\n",num_levels,N);

  /* Open Data I/O Files. */

    if ( ( input = fopen("data.out","r") ) == NULL )
      {
      printf("\n\n    Error Opening Input File.    \n\n");
      return(0);
      }
    if ( ( output = fopen("process.dat","w") ) == NULL )
      {
      printf("\n\n    Error Creating Output File.    \n\n");
      return(0);
      }

  /* Initialize Working Variables. */

    level=MAX_DAC_LEVELS;
    dlevel=MAX_DAC_LEVELS/num_levels;
    for(i=0; i<MAX_N; i++)
      data[i]=level;
    for(i=0; i<MAX_N; i++)
      comp_out[i]=0;

  /* Multi-Pass Data Processing Loop. */

    for(i=0; i<800; i++)
      fscanf(input,"%i",&byte);

    for(j=0; j<num_levels; j++)
      {
      level-=dlevel;
      for(i=0; i<N; i++)
        {
          fscanf(input,"%i",&byte);
          if((comp_out[i]!=byte) & (data[i]==MAX_DAC_LEVELS))
            {
```

```
            comp_out[i]=byte;
            data[i]=level;
            }
        }
      }

  /* Reshuffle Sample Points According to SAMPLE_CYCLES. */

    for(i=0; i<N; i++)
      comp_out[i]=data[i];
    for(i=0; i<N; i++)
      {
      data[(i*SAMPLE_CYCLES) % N]=comp_out[i];
      }

  /* Generate Output File. */

    for(i=0; i<N; i++)
      fprintf(output,"%f\n",(float)(data[i])*3.3/num_levels);
    fclose(output);
}
```

Appendix D - VHDL Code

D.1 - blocks.vhd

```
----------------------------------------
--  Parallel Load Register With Reset  --
----------------------------------------

entity reg is
  generic( size : natural );
  port( clock, not_reset : bit;
        input : bit_vector(size-1 downto 0);
        output : out bit_vector(size-1 downto 0) );
  -- pragma template
end reg;

architecture struct of reg is
begin
  process( clock )
  begin
    if clock'event and clock='1' then
      if not_reset='0' then
      ZERO_LOOP: for i in 0 to size-1 loop
        output(i) <= '0';
      end loop ZERO_LOOP;
      else
      output <= input;
      end if;
    end if;
  end process;
end struct;

--------------------------------------------------
--  Parallel Load Register With Enable and Reset  --
--------------------------------------------------

entity reg_en is
  generic( size : natural );
  port( clock, not_reset, not_enable : bit;
        input : bit_vector(size-1 downto 0);
        output : out bit_vector(size-1 downto 0) );
  -- pragma template
end reg_en;

architecture struct of reg_en is
```

```
begin
  process( clock )
  begin
    if clock'event and clock='1' then
      if not_reset='0' then
      ZERO_LOOP: for i in 0 to size-1 loop
        output(i) <= '0';
      end loop ZERO_LOOP;
      else
      if not_enable='0' then
        output <= input;
      end if;
      end if;
    end if;
  end process;
end struct;

----------------------------------------
--  Parallel Load Shift Left Register  --
----------------------------------------

entity shift_reg_left is
  generic( size : natural );
  port( clock : bit;
        notshift_load : bit;
        ser_in : bit;
        input : bit_vector(size-1 downto 0);
        ser_out : out bit;
        output : out bit_vector(size-1 downto 0) );
  -- pragma template
end shift_reg_left;

architecture struct of shift_reg_left is
  signal temp : bit_vector(size-1 downto 0);
begin
  process( clock )
  begin
    if clock'event and clock='1' then
      if notshift_load='1' then
        temp <= input;
      else
      SHIFT_LOOP: for i in 1 to size-1 loop
        temp(i) <= temp(i-1);
      end loop SHIFT_LOOP;
      temp(0) <= ser_in;
      end if;
    end if;
  end process;
  output <= temp;
  ser_out <= temp(size-1);
end struct;
```

```
------------------------------
--  D-Flip Flop With Reset  --
------------------------------

entity dff is
  port( clock, not_reset, input : bit;
        output : out bit );
end dff;

architecture struct of dff is
begin
  process( clock )
  begin
    if clock'event and clock='1' then
      if not_reset='0' then
      output <= '0';
      else
      output <= input;
      end if;
    end if;
  end process;
end struct;

---------------------------------
--  D-Flip Flop Without Reset  --
---------------------------------

entity dff_norst is
  port( clock, input : bit;
        output : out bit );
end dff_norst;

architecture struct of dff_norst is
begin
  process( clock )
  begin
    if clock'event and clock='1' then
      output <= input;
    end if;
  end process;
end struct;

----------------------------------------
--  D-Flip Flop With Enable And Reset  --
----------------------------------------

entity dff_en is
  port( clock, not_reset, not_enable, input : bit;
        output : out bit );
end dff_en;

architecture struct of dff_en is
begin
```

```
  process( clock )
  begin
    if clock'event and clock='1' then
      if not_reset='0' then
        output <= '0';
      else
        if not_enable='0' then
          output <= input;
      end if;
      end if;
    end if;
  end process;
end struct;

---------------
--  Counter  --
---------------

use work.bv_arithmetic.all;

entity counter is
  generic( size : natural );
  port( clock, not_reset : bit;
        add_vec : bit_vector(size-1 downto 0);
        count : out bit_vector(size-1 downto 0) );
  -- pragma template
end counter;

architecture struct of counter is
  component reg
    generic( size : natural );
    port( clock, not_reset : bit;
          input : bit_vector(size-1 downto 0);
          output : out bit_vector(size-1 downto 0) );
  end component;
  signal sum, temp : bit_vector(size-1 downto 0);
begin
  sum <= temp + add_vec;
  REG1: reg
    generic map( size )
    port map( clock, not_reset, sum, temp );
  count <= temp;
end;

---------------------------
--  Counter With Enable  --
---------------------------

use work.bv_arithmetic.all;

entity counter_en is
  generic( size : natural );
  port( clock, not_reset, not_enable : bit;
```

```
        add_vec : bit_vector(size-1 downto 0);
        count : out bit_vector(size-1 downto 0) );
  -- pragma template
end counter_en;

architecture struct of counter_en is
  component reg_en
    generic( size : natural );
    port( clock, not_reset, not_enable : bit;
          input : bit_vector(size-1 downto 0);
          output : out bit_vector(size-1 downto 0) );
  end component;
  signal sum, temp : bit_vector(size-1 downto 0);
begin
  sum <= temp + add_vec;
  REG1: reg_en
    generic map( size )
    port map( clock, not_reset, not_enable, sum, temp );
  count <= temp;
end;
```

D.2 - comp_blocks.vhd

```
package components is
  component reg
    generic( size : natural );
    port( clock, not_reset : bit;
          input : bit_vector(size-1 downto 0);
          output : out bit_vector(size-1 downto 0) );
  end component;
  component reg_en
    generic( size : natural );
    port( clock, not_reset, not_enable : bit;
          input : bit_vector(size-1 downto 0);
          output : out bit_vector(size-1 downto 0) );
  end component;
  component shift_reg_left
    generic( size : natural );
    port( clock : bit;
          notshift_load : bit;
          ser_in : bit;
          input : bit_vector(size-1 downto 0);
          ser_out : out bit;
          output : out bit_vector(size-1 downto 0) );
  end component;
  component dff
    port( clock, not_reset, input : bit;
          output : out bit );
  end component;
  component dff_norst
    port( clock, input : bit;
          output : out bit );
```

```
  end component;
  component dff_en
    port( clock, not_reset, not_enable, input : bit;
          output : out bit );
  end component;
  component counter
    generic( size : natural );
    port( clock, not_reset : bit;
          add_vec : bit_vector(size-1 downto 0);
          count : out bit_vector(size-1 downto 0) );
  end component;
  component counter_en
    generic( size : natural );
    port( clock, not_reset, not_enable : bit;
          add_vec : bit_vector(size-1 downto 0);
          count : out bit_vector(size-1 downto 0) );
  end component;
end components;

package body components is
end components;
```

D.3 - DAC_CTRL.vhd

```
use work.components.all, work.bv_arithmetic.all;
entity DAC_CTRL is
  generic( M : natural := 8 );
  port( XCLOCK, nRESET : bit;
        DATA_IN : bit;
        TRG_OUT : out bit;
        DATA_OUT : out bit_vector(7 downto 0);
        DCLK, nCS_LD, DIN : out bit );
end DAC_CTRL;

architecture struct of DAC_CTRL is

  signal DAC_one, DAC_CNT : bit_vector(11 downto 0);
  signal N_CNT, N_one, NXT_LEV : bit_vector(11 downto 0);
  signal minus_one, LEV_CNT : bit_vector(M-1 downto 0);

  signal n5, n6, n7, n9, n10, n12, n13, n14, n15 : bit;
  signal n16, n17, n18, n19, n20, n21, n22, n23, n24 : bit;
  signal MCLK, nMCLK, nTRG_EN, nMRST, DAC_SEQ, nRSH_RLD : bit;
  signal nCNT_RST, N_DET, nUPDATE, nDAC_STOP, DAC_DET, DCLK_EN : bit;

  signal n25 : bit_vector(2 downto 0);
  signal DATA_zero : bit_vector(7 downto 0);
  signal n26, n27, n28, nTRGD, Dzero : bit;

begin
```

```
-------------------------------------------------
-- N setting depends on M and VXI capture limits --
-------------------------------------------------
with N_CNT select
  N_DET <= '1' when "001000000000",
           '0' when others;
-------------------------------------------------

nMCLK <= not MCLK;

DIV2_XCLK: dff_norst
  port map( XCLOCK, nMCLK, MCLK );

GEN_nMRST: dff_norst
  port map( MCLK, nRESET, nMRST);

nTRG_EN <= DAC_SEQ;

DAC_one <= ext("1",12);
DAC_COUNTER: counter
  generic map( 12 )
  port map( nMCLK, nCNT_RST, DAC_one, DAC_CNT );

n5 <= nMRST and n7;

n7 <= not nTRG_EN;

N_one <= ext("1",12);
N_COUNTER: counter
  generic map( 12 )
  port map( MCLK, n5, N_one, N_CNT );

n6 <= nUPDATE and nDAC_STOP;

n9 <= not nCNT_RST;

GEN_nCNT_RST: dff_en
  port map( MCLK, nMRST, n6, n9, nCNT_RST );

DAC_SEQ <= n9 nand nUPDATE;

with DAC_CNT select
  nDAC_STOP <= '0' when "111111111111",
               '1' when others;

with DAC_CNT select
  DAC_DET <= '1' when "000000011001",
             '0' when others;

DAC_REG: shift_reg_left
  generic map( 12 )
  port map( MCLK, nRSH_RLD, n10, NXT_LEV, n10, OPEN );
```

```
DIN <= n10;

minus_one <= sxt("1",M);
LEVEL_COUNTER: counter_en
  generic map( M )
  port map( MCLK, nMRST, nUPDATE, minus_one, LEV_CNT );

NXT_LEV(11 downto 12-M) <= LEV_CNT;
NXT_LEV(11-M downto 0) <= ext("0",12-M);

DELAY3: dff_norst
  port map( MCLK, N_DET, n12 );

n13 <= not N12;

DELAY4: dff_norst
  port map( MCLK, nMRST, n14 );

n15 <= not n14;

n16 <= N_DET and n13;

n17 <= nMRST and n15;

n18 <= n16 or n17;

GEN_nRSH_RLD: dff_norst
  port map( MCLK, n18, nRSH_RLD );

nUPDATE <= not n18;

n19 <= nRSH_RLD nor DAC_DET;

n20 <= not n21;

nCS_LD <= n20;

DELAY5: dff_en
  port map( nMCLK, nMRST, n19, n20, n21 );

n22 <= not n23;

DELAY6: dff_en
  port map( MCLK, nMRST, n19, n22, n23 );

DELAY7: dff_norst
  port map( nMCLK, n21, n24 );

DCLK_EN <= n23 and n24;

DCLK <= nMCLK and DCLK_EN;
```

```
  n25 <= N_CNT(2 downto 0);
  with n25 select
    n26 <= '0' when "000",
           '1' when others;

  n27 <= n26 nor nTRGD;

  GEN_nTRGD: dff_norst
    port map( MCLK, nTRG_EN, nTRGD );

  GEN_n28: dff_norst
    port map( MCLK, nMRST, n28 );

  GEN_TRG_OUT: dff
    port map( nMCLK, n28, n27, TRG_OUT );

  Dzero <= '0';
  DATA_zero <= "00000000";
  GEN_DATA_OUT: shift_reg_left
    generic map( 8 )
    port map( MCLK, Dzero, DATA_IN, DATA_zero, open, DATA_OUT );

end struct;
```

D.4 - DAC_CTRL.script

```
/***      SYNOPSYS DESIGN COMPILER (DC_SHELL) SCRIPT FILE      ***/

bus_naming_style = "%s<%d>"
bus_dimension_separator_style = "><"
bus_inference_style = "%s<%d>"

search_path = { ./ /usr/vlsi1/xact/synopsys/libraries/syn/
/usr/vlsi1/synopsys/libraries/syn }
link_library = {xprim_4010-6.db xprim_4000-6.db xfpga_4000-6.db
xgen_4000.db xio_4000-6.db xdc_4000-6.db}
target_library = {xprim_4010-6.db xprim_4000-6.db xfpga_4000-6.db
xgen_4000.db xio_4000-6.db xdc_4000-6.db}
symbol_library = xc4000.sdb
define_design_lib xblox_4000 -path
/usr/vlsi1/xact/synopsys/libraries/dw/lib/fpga/xc4000
synthetic_library = {xblox_4000.sldb standard.sldb}

read -f vhdl bvarithmetic.vhd   /* Included in the Synopsys library
files. */
read -f vhdl blocks.vhd
read -f vhdl comp_blocks.vhd
read -f vhdl DAC_CTRL.vhd

TOP = DAC_CTRL
CLOCK = XCLOCK
```

```
current_design TOP

uniquify

set_port_is_pad find(design, TOP )
set_pad_type -clock find(port, CLOCK )
insert_pads

check_design

compile_fix_multiple_port_nets = true

set_operating_conditions WCCOM
set_wire_load "4010-6_avg"
create_clock -name clock -period 50 { CLOCK }

max_area 350

compile -boundary_optimization -ungroup_all

report -area -timing
report_fpga

write -format db -hierarchy -output  TOP + ".db"

set_attribute find(design, TOP ) "part" -type string "4010PG191-6"

set_attribute { "XCLOCK" }  "pad location" -type string "C3"
set_attribute { "DCLK" }  "pad location" -type string "B17"
set_attribute { "DIN" }  "pad location" -type string "R1"
set_attribute { "nCS_LD" }  "pad location" -type string "U16"

set_attribute { "DATA_OUT<0>" }  "pad location" -type string "V16"
set_attribute { "DATA_OUT<1>" }  "pad location" -type string "V15"
set_attribute { "DATA_OUT<2>" }  "pad location" -type string "V14"
set_attribute { "DATA_OUT<3>" }  "pad location" -type string "V13"
set_attribute { "DATA_OUT<4>" }  "pad location" -type string "U15"
set_attribute { "DATA_OUT<5>" }  "pad location" -type string "U14"
set_attribute { "DATA_OUT<6>" }  "pad location" -type string "U13"
set_attribute { "DATA_OUT<7>" }  "pad location" -type string "U11"

set_attribute { "nRESET" }  "pad location" -type string "R17"
set_attribute { "DATA_IN" }  "pad location" -type string "U5"
set_attribute { "TRG_OUT" }  "pad location" -type string "A11"

replace_fpga

xnfout_library_version = "2.0.0"
xnfout_constraints_per_endpoint = 4

write -format xnf -hierarchy -output TOP + ".sxnf"

exit
```