THE CONTROL OF A HIGH-SPEED, LOW-JITTER PULSE GENERATOR

by

Thomas A. Fitzpatrick

Submitted to the

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

in partial fulfillment of the requirements

for the degrees of

BACHELOR OF SCIENCE

and

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1986

Signature of Author Signature redacted
Department of Electrical Engineering and computer Science,
May 9, 1986

Certified by Signature redacted
Richard D. Thornton, Thesis Supervisor (Academic)

Certified by Signature redacted
Laszlo Dobos, Supervisor (Cooperating Company)

Accepted by Signature redacted
Arthur C. Smith, Chairman
Committee on Graduate Students

THE CONTROL OF A HIGH-SPEED, LOW-JITTER PULSE GENERATOR

by

Thomas A. Fitzpatrick

Submitted to the Department of Electrical Engineering and Computer Science on June 2, 1986 in partial fulfillment of the requirements for the degrees of Master of Science and Bachelor of Science.

Abstract

A microprocessor-based digital system was designed which would serve as the control portion of a high-speed, low-jitter pulse generator. Existing systems use analog techniques to generate the timebase and trigger-to-output delays, and are thus subject to a large amount of time jitter on the pulse output. The system presented uses digital techniques to stabilize a series of programmable oscillators to generate these timing signals. The variable periods of the oscillators allow time resolutions comparable to that of the currently available analog- based systems, while the digital circuitry allows for constant monitoring of these oscillators to reduce the jitter.

The system was tested and found to have 60 - 80 picoseconds of jitter on the leading edge of the output signal, which is comparable to currently available high performance pulse generators. The fact that this result did not meet the proposed specification is due in large part to the a number of noise sources present in the test implementation of the circuit. These noise sources were identified, and techniques for reducing or eliminating their effects were discussed.

Thesis Supervisor:	Professor Richard D. Thornton
Title:	Professor of Electrical Engineering and Computer Science

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

With the advancement of technology, today's engineer is faced with a problem. As device speeds increase, he must be able to verify that the systems or components he designs will work at top speeds. For emitter-coupled logic (ECL) components, this means frequencies of about 250 MHz, while, for gallium arsenide (GaAs) devices, the frequencies can be much higher. The easiest way for the engineer to test such systems is by repeatedly stimulating the system and observing the results. The programmable pulse generator is made for such applications.

To be most effective, the ideal pulse generator would need to

1. produce an output pulse at frequencies up to the device-under-test's (DUT) maximum frequency (which for our purposes will mean 250 MHz),

2. be programmable to allow testing with a wide range of input waveforms, and

3. have a trigger-to-pulse output jitter of aproximately 5 ps to allow for observation of a small portion of the device-under-test's (DUT) system response if desired.

The low jitter specification is necessary to allow a user to examine such characteristics as rise times with a high degree of accuracy. For example, if a user wishes to observe an output rising edge with a risetime of 50-100 ps, but this signal itself has 50 ps of jitter with respect to the trigger, this jitter is a significant portion of the time window being observed, and will make it difficult for the user to make very accurate measurements (see Fig. 1-1).

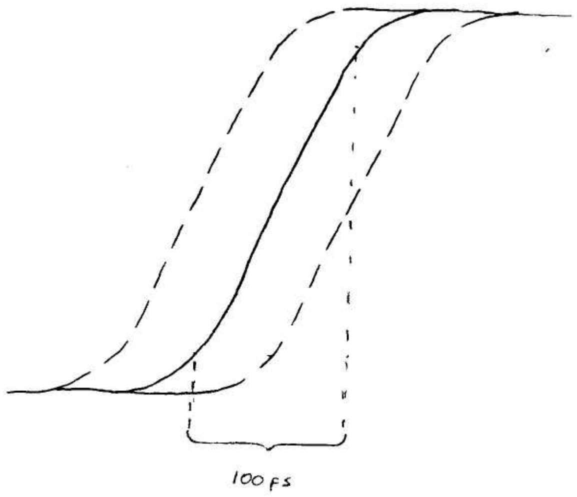


Figure 1-1: Jitter Reduces Measurement Accuracy

1.2 Proposed Methodology

Before discussing possible schemes for implementing our system, let us first examine some currently available pulse generators which define "the state of the art."

- Hewlett-Packard HP 8080A SO4 - Frequencies up to 1 GHz, square-wave output only, not programmable
- Hewlett-Packard HP 8161A - Frequencies up to 100 MHz, programmable output, 50 ps + 0.1% jitter
- Colby Instruments, Inc. PG 4000A - Frequencies up to 4 GHz, square-wave output, not programmable
- EH Electronics SPG 2000 - Frequencies up to 200 MHz, programmable output, 25 ps + 0.1%
- Tektronix PG502 - frequencies up to 250 MHz, not programmable, 25 ps + 0.1% jitter

We can see here that there is no commercially available pulse generator that meets our specifications for speed, stability, and programmablilty. The HP 8161A and the SPG 2000 come closest to meeting these goals, but unfortunately are unable to meet the stability specification of ~5 ps. This instability comes mainly from the fact that they have analog timebases which are highly susceptible to noise from sources such as power supplies, digital subsystems, and the like [5]. The obvious way around this would be to use a digital timebase, in which one would be working against a threshold, instead of linear gain stages, and which would therefore be easier to stabilize. Trigger-to-pulse delays could then be generated by simply counting periods of this stable clock.

Unfortunately, this scheme dictates that the delay would have to be some integer multiple of the clock frequency. Even at 1 GHz, then, the delay could only be specified to 1 nanosecond, which is not fine enough. In order to take set-up time measurements, for example, we would need to alter the spacing between two pulses (eg. the data and clock inputs to a flip-flop) gradually to see when the output fails. It would be convenient then to be able to specify delays with about 10 ps resolution. This would take a clock running faster than standard login components would be able to count.

The fastest clock speed that we would be able to count with standard components would be 250 MHz, using Motorola 10KH series ECL components. Using the above scheme, this would give us a minimum delay resolution of 4 ns. By allowing the clock frequency to change though, we can increase the resolution without having to count any faster than 250 MHz.

By using a programmable timer chip, developed at Tektronix, as the foundation, we were able to implement a timebase that can be set for anywhere from 125 MHz to 250 MHz with up to 13 bits of resolution (see section 3.5). This allows us to match the delay resolution of wholly analog systems, but since the oscillator is controlled digitally, we can constantly monitor its performance and change its inputs when necessary. This allows us the increased stability we need.

1.3 Perspective

Chapter 2 of this thesis deals with setting up the three programmable oscillators in the system to generate a pulse output with the desired parameters. It discusses the user interface system used in the test circuit, as well as the protocol the system uses for communicating with an output stage. It is here that the algorithms for calculating the oscillator inputs to achieve the desired pulse output are discussed. Chapter 3 consists of a detailed description of the hardware implementation of the system. It describes in detail the operation of the programmable timer chips, as well as the self-calibration system and the control system. The effects of noise on system stability are discussed in chapter 4, as are ways of reducing noise, and calculations to determine how much noise is acceptable before we can no longer meet the jitter specification. Chapter 5 describes a procedure for measuring the time-jitter of the system, evaluates the system performance, and draws conclusions.

Chapter 2

Control Tasks

2.1 Overview

A simplified block diagram of a pulse generator would consist of three basic blocks: user interface, output, and control. For the purposes of this thesis, we will concentrate on the control section, and simply provide a method of communicating with the user interface and the output stage. The control section must then be able to:

1. Get information from the user,
2. Provide appropriate signals to the output stage based on this information, and
3. Adjust itself to changing operating conditions.

2.2 User Interface

To simplify the system development, it was necessary to provide for a method of allowing a user to control the system. To avoid having to design, build, and test a front-panel display unit, I used a serial interface subcircuit, which had already been designed at Tektronix, and for which driver software had already been written, to connect the system to a standard RS232-C compatible terminal. The user interface task was thus simplified to having to write software routines which allow the user to

1. Debug the hardware,
2. Take measurements on the system for self-calibration, and

3. Run the system.

The hardware debugging routines were developed as needed and were generally implemented as continuous loops executing writes to individual hardware registers. The performance of the registers and the parts of the circuit they drove could be checked and, if necessary, fixed in this way. Similar routines could be incorporated in the final product to provide hardware self-diagnostic capabilities.

2.3 Output Signals

In order to produce any kind of useful pulse output, a pulse generator must supply three important pieces of information:

1. A trigger signal with which to synchronize the pulse,
2. A delay from the trigger output to the beginning of the pulse, and
3. A pulse width.

The system uses three programmable oscillator/timer chips, developed at Tektronix, to synthesize these three signals. Please see sections 3.5 and 3.6 for a detailed explanation of the operation and drive requirements of this chip.

The trigger signal is generated either from the output of one timer chip (called the Repetition Rate Generator, or RRG) or from an external source by taking this source and logically ANDing it with an inverted version of itself, delayed by 1.2 nanoseconds. This 1.2 ns pulse then becomes the TRIG signal to both the outside world and the rest of the control system (see Fig. 2-1).

Another oscillator/timer (called the Leading Edge Delay generator, or LED) is used to produce the delay from TRIG to the pulse leading edge. The chip makes use of an internal burst counter to produce a signal, BTO, which is high for a specified number of oscillator periods. The third oscillator/timer (the Trailing Edge Delay

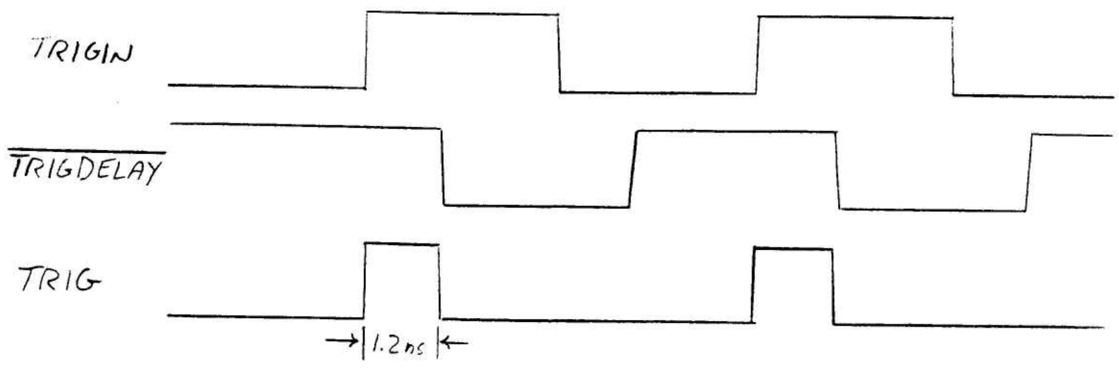


Figure 2-1: Generating the Trigger Signal

generator, or TED) is used similarly to produce a delay to the trailing edge of the pulse output. By combining these two signals, then, we can produce the desired pulse waveform.

2.4 Setting The Repetition Rate Generator

The user has the option of specifying either internal or external triggering. If external triggering is selected, he must supply a trigger input signal, with frequency of no more than 250 MHz (ie. period not less than 4 ns), which is high for at least 1.2 ns so that a useful TRIG signal can be generated (see section 2.3). In this case, the controller has only to configure the system to pass this EXTTRIG input signal to the TRIG output circuit produce the TRIG output.

If the user selects internal triggering, however, the controller must not only pass the RRG output to the TRIG circuit, but must also set up the RRG to produce the output frequency specified by the user. In order to achieve the desired frequency, we must calculate the correct drive currents for the oscillator inputs, as well as the appropriate number, if any, to load into the timer's on-chip divider.

To simplify this procedure, we can create a look-up table (as described in section 3.7) which contains the RRG output frequency for a given value of the input currents (part of such a table appears in Table 2-I). The frequency values in this table will range from less than 125 MHz to at least 250 MHz. Setting the currents then becomes the relatively straightforward task of finding the appropriate place in the table that the desired frequency falls.

Current	Frequency (MHz)
78	163.265
79	163.683
7A	164.103
7B	164.524
7C	164.949
7D	165.375
7E	165.802
7F	166.233
80	166.653
81	167.102
82	167.540
83	167.979
84	168.421
85	168.865
86	169.315
87	169.761

Table 2-I: Frequency vs. Current Look-up Table

For example, let us suppose that the user wishes to set a repetition rate of 165 MHz. Using Table 2-I, we see that 165 MHz lies between entries 7C (164.949 MHz) and 7D (165.375 MHz), so we therefore set the coarse value of the IPR input to 7C. We must now linearly interpolate to find the correct settings for the IPR fine and IPD inputs.

To set the IPD input, we must know which half of the interval the desired output lies. The middle frequency in this example is

$$\text{MIDFREQ} = (164.949 + 165.375) / 2 = 165.162$$

so the desired frequency is in the lower half of the interval and we can set the IPD input to 7C.

As stated in section 3.6, the range of the IPR fine register is such that 32 fine steps corresponds to one coarse step. We therefore interpolate between the bottom of the interval and MIDFREQ as follows:

$$F = \{[64*(165 - 164.949)/(165.162 - 164.949)] + 1\}/2 = 8.$$

Since the processor only does integer arithmetic operations, we have a convenient way of expressing the frequencies as integers (see Section 3.7). By multiplying by 64, adding 1, and dividing by 2, we effectively round-off the value to the nearest LSB, and eliminate truncation errors.

If the frequency specified by the user is not between 125 and 250 MHz, we must calculate the number by which to divide the oscillator output. We must also convert this frequency to a frequency in the relevant range so that we can use the look-up table. Fortunately, both these tasks can be done simultaneously by simply doubling the specified frequency until it becomes greater than 125 MHz. The number of doubling operations thus performed is the power of two that we need to divide the oscillator output by to get the desired frequency out. For example, if the user specified an output frequency of 21 MHz, we would simply double this 3 times to achieve a look-up frequency of 168 MHz. Once we set the oscillator output to close to 168 MHz, we need only to enable the on-chip divider and divide by 8 to achieve the necessary output.

2.5 Edge Delay Generation

The method used for generating the edge delay signals is similar to that used for generating the repetition rate in that a look-up table and linear interpolation are used. Instead of the oscillator frequency, however, the look-up table consists of the oscillator output period vs. the current setting (see Table 4-I), and the timer chip's burst counter is used instead of the divider

Once the LED and TED times are known, we can use the same algorithm to set the inputs for the two remaining timer chips. If the desired delay is between 4 and 8 nanoseconds, the interpolation is carried out exactly as it was for the RRG except that in this case, a higher index in the look-up table corresponds to a smaller value of the output period.

If the delay needed is larger that 8 ns, we divide this delay by a number, N, which will put it in the appropriate range. Once we set the oscillator inputs for this divided delay, we simply load the burst counter with N to achieve the desired delay output.

Chapter 3

Circuit Description

3.1 Overview

Once a general architecture had been decided, it became necessary to implement the system using real hardware. This chapter is intended to explain, in depth, the functionality of the hardware.

The system consisted of six major subsystems:

1. The processor subsystem, including the program memory and address decoding, which was used to execute the software and control the system,

2. The Serial Interface which was used to communicate with a terminal during system development,

3. The control registers which were used to hold information from the processor to correctly configure the system,

4. The timer chips which contained the programmable oscillators and were used to generate the appropriate timing signals for the output,

5. The "DAC Farm" which consisted of fifteen digital-to-analog converters, used to supply the appropriate drive to the timer chip oscillators, and

6. The diagnostic/calibration system which was used to monitor the performance of the programmable oscillators.

3.2 The Processor

The processor used was an Intel 80186 (see Fig. 3-1). This processor was chosen primarily because there was an abundance of support software already written for performing such tasks as allotting memory, configuring output ports, and communicating with a terminal via the serial interface unit, which was also conveniently available.

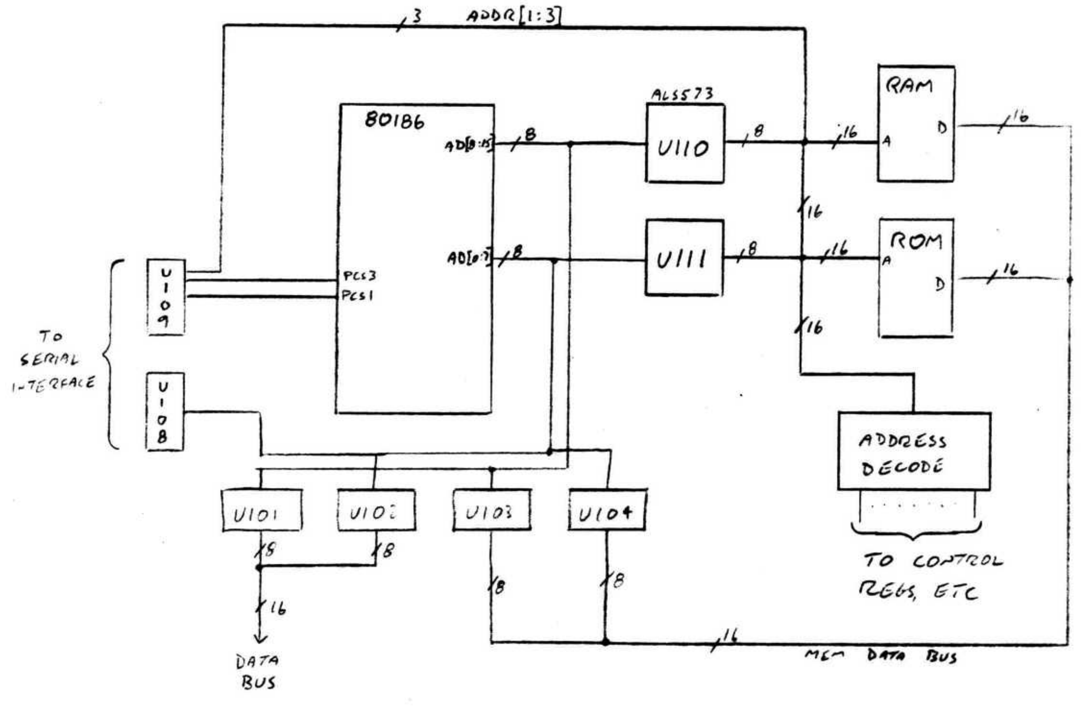


Figure 3-1: The Processor and Memory

One major advantage of this processor is its Peripheral Chip Select (PCS) capabilities. These PCS lines allowed us to partition the hardware into separate areas of memory, and thus to simplify the software which drove this hardware.

Two ALS573 latches (U110 and U111) were used in conjunction with the processor's Address Latch Enable (ALE) output to demultiplex the address and data lines from the 80186. A group of four ALS245 bidirectional buffers (U101 - U104) were used to boost the drive capabilities of these lines so they could drive the rest of the circuit. Another pair of buffers served a similar purpose in the serial interface. A small amount of combinational logic was used to enable these buffers when the appropriate address space was selected.

The program memory consisted of two eight-bit wide ROMs with up to 64 Kbytes of address space to hold up to 128 Kbytes of code. There was also 128 Kbytes of RAM space to be used for performing calculations. The processor's Upper and Lower Chip Select lines (UCSL and LCSL, respectively) were used, along with the Bus High Enable (BHEL) and least significant address bit (A0) and some combinational logic, to select the appropriate memory chip(s).

3.3 The Serial Interface

The Serial Interface, as explained in section 3.3 allowed the processor to communicate with up to two different devices which were selected by the 80186's PCS1 and PCS3 lines. We used a standard RS232 terminal to develop and test the system. It would be possible, in the future, to create a front-panel which could make the final production user-interface software somewhat simpler.

3.4 The Control Registers

There were two control registers which control the modes for the timer chips. The first, called the the Edge Delay Control Register (edctrl), controled the Leading and Trailing Edge Delay circuits (LED and TED, respectively). The other, called the Repetition Rate Generator Control Register (rrgctrl) controled the Repetition Rate Generator (RRG), the diagnostic source selector, the trigger output control, and some of the TED circuit. There was also a trigger input control register, U390, which provided the control signals for the external trigger input circuit.

The OUTSEL bit of the trigger output control allowed the user to select between an external trigger source or the output of the RRG circuit for the trigger output. The OUTDIS bit disabled the trigger output when the source was being changed, or when calibration was being performed.

3.5 The Timer Chips

The heart of the system was the three programmable oscillator/timer chips developed at Tektronix. Each chip contains:

1. A programmable oscillator with current control inputs,
2. A 2 ^ n divide-by circuit
3. A 14-bit programmable burst counter

3.5.1 The Oscillator

The on-chip oscillator is fully programmable from 125 MHz to 250 MHz depending on the value of the currents supplied to the Input Duration (ID) and Input Reset (IR) inputs, with ID defining the time that the output is high, and IR defining the time that it is low. These times decrease as the current to the corresponding input increases, so the output frequency is an increasing function of the input currents. The maximum current allowed on each of these inputs is 4.1 mA.

The oscillator also has five enable inputs, GI1 through GI5, which turn on the oscillator. Inputs GI1 through GI4 are driven by the divider and burst counter circuits, while GI5 is the logical OR of the chip-inputs G1 and G2. A pulse on any of these lines of less than 60% of the programmed period of the oscillator will put the oscillator in one-shot mode, where the oscillator output will go high for one period of the divided output minus one half the programmed oscillator period (see

Fig. 3-2). Normally, however, the oscillator is used either in burst mode (when generating a delay) or in free-run mode with G2 held high (when running calibration).

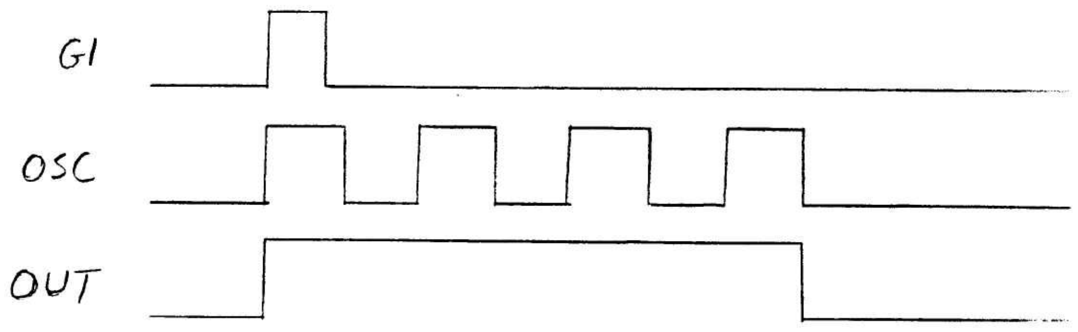


Figure 3-2: Divided-by-four Oneshot Output

3.5.2 The On-Chip Divider

The divider subcircuit (see Fig. 3-3) is implemented using an 8-bit shift register and eight toggle flip-flops, with the output of each toggle stage being used to clock the next highest stage. The least significant bit of the counter is clocked from the output of the oscillator itself. The shift register is loaded serially from the DCIN (Divide Counter INput) pin, which is connected to the least significant bit of the data bus (DATA0), with each rising edge of the DCLK (Divider CLocK) line. By connecting the appropriate select line from the address decoder to DCLK, then, we can write to this register one bit at a time by writing to the divider's address (eg. LEDDIV for the LED divider). We can read the contents of the shift register by shifting the data out from the most significant bit of the shift register to the BDLT pin. By connecting this pin back to DCIN through a tri-state buffer which is only enabled on a read, we were able to do a non-destructive read of the shift register contents.

The divide-by number, N, which must be an integer power of 2, was set by loading the shift register with the value 256 - N. If, for example, we wished to divide the output by 4, we would load the shift register with

256 - 4 = 252

or, in binary,

100000000 - 000000100 = 11111100

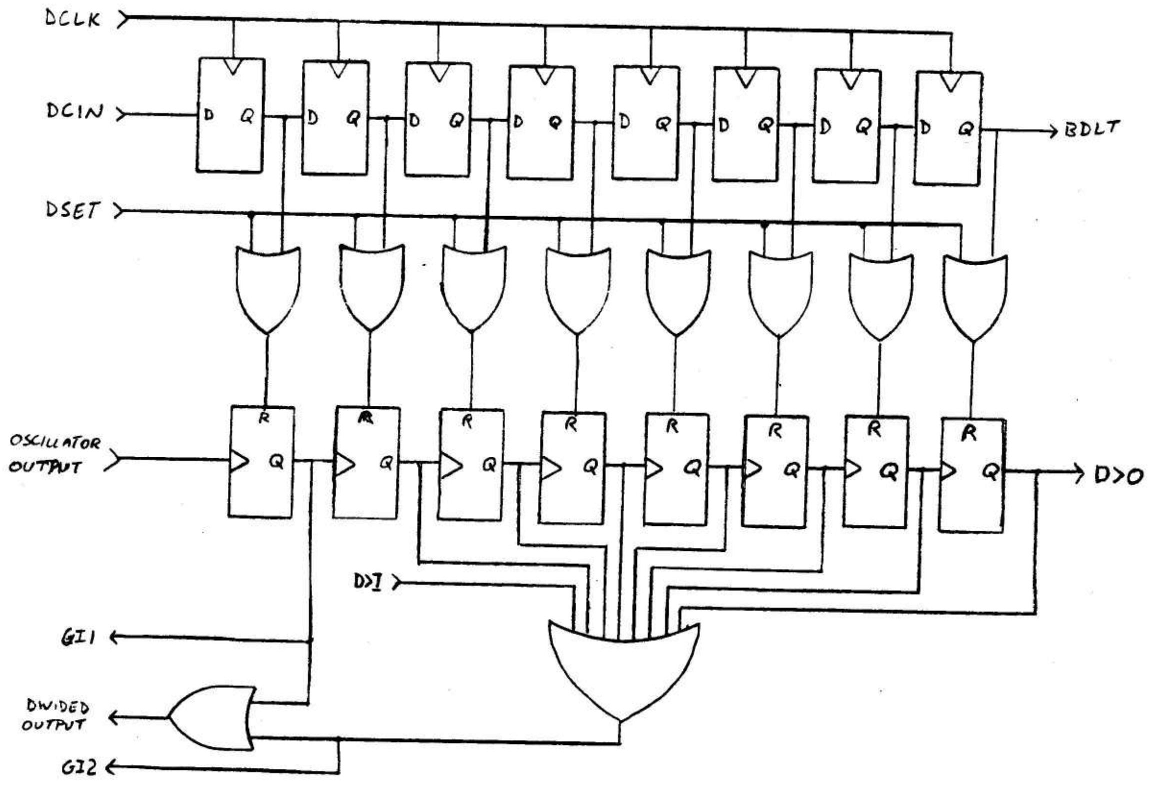


Figure 3-3: Divider Subcircuit

The outputs of this shift register are each logically ORed with the DSET control line and run into the RESET input of each of the toggle flip-flops. Bringing DSET low, then, will enable only the least significant n bits of the divider.

The TTL signals D>O and D>I can be used to expand this 8-bit divider to any length, allowing us to produce any output frequency from 250 MHz down to essentially a DC output. The D>O output was also used to divide the oscillator output by 256 and provide a TTL compatible signal for use in the calibration subsystem (see section 3.7).

By dividing down the output of the RRG subsystem, then, we were able to get output frequencies from 250 MHz down to

125 MHz / 256 = 490 KHz

3.5.3 The Burst Counter

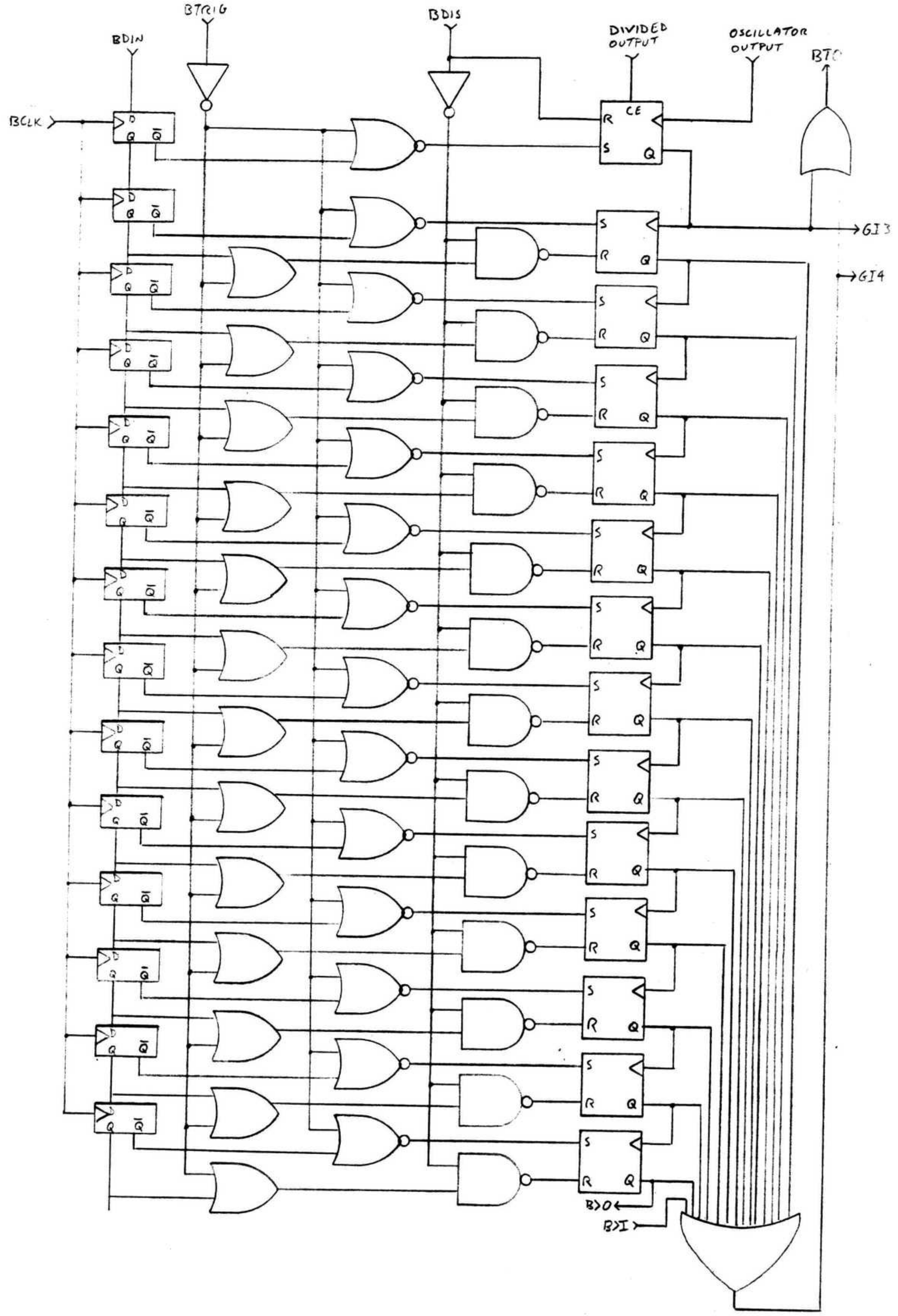
The burst counter (see Fig. 3-4) is implemented with a string of fourteen cascaded toggle flip-flops, and is clocked from the output of the oscillator when the divider outputs are all zero. This counter is loaded via a 14-bit shift register which is controlled similarly to the divider shift register, with BCLK and BCIN behaving analogously to DCLK and DCIN. With BCLK and DCLK decoded to be different addresses, we could run DATA0 to BCIN as well as DCIN, since only one shift register would be clocked at a time. Similarly, since the most significant bit of this register was also connected to BDLT, we could use the same tri-state buffer as with the divider to allow non-destructive reads of the burst counter.

A pulse of 1.1 to 1.8 ns on the BTGR input will trigger the start of a burst (see Fig. 3-5) by loading the counter with the desired number of periods in the burst. The BTO output is the logical OR of all the bits of the burst counter, and is therefore high when a burst is being counted. It was this signal that was used to produce the delay signals for the leading or trailing edge of the output pulse. By combining the divider and the burst counter, then, we could produce a delay of as small as 4 ns and as large as

8 ns * 256 * 16384 = 33.5 ms

from either the LED or TED subsystems (the burst counter of the RRG subsystem is never used).

Figure 3-4: Burst Counter Subcircuit



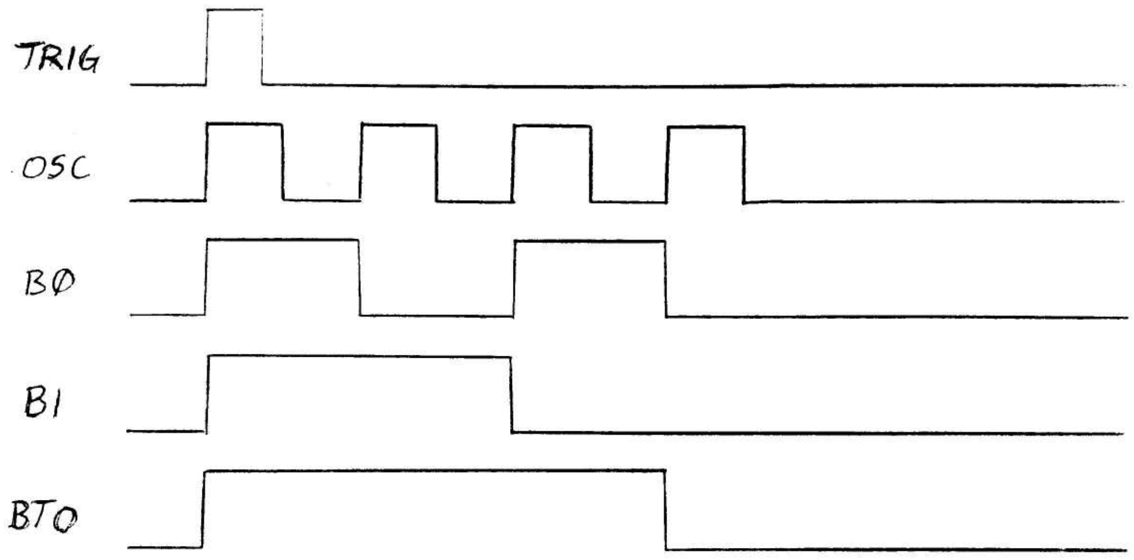


Figure 3-5: Burst Counter Output Timing

3.6 The "DAC Farm"

To set the values for the current inputs of the oscillators, we used an arrangement of five 8-bit digital-to-analog converters (Advanced Micro Devices, Inc. DAC-08's) to generate the ID and IR signals for each oscillator: three to generate IR and two to generate ID (see Fig. 3-6).

The BASE register was used to drive two DACs, one for each input, and to set the oscillator to a base frequency of just under 125 MHz. Two other DACs, IRC (Input Reset Coarse) and IDC (Input Duration Coarse) were connected to these BASE DACs to actually set the frequency between 125 and 250 MHz, and a third DAC (FINE) was connected to the IR input as a fine adjustment to improve the resolution.

The range of the BASE DACswais 3.1 mA, which is guaranteed to set the oscillator to at least 125 MHz. The range of the IRC and IDC DACs was 4.1 mA. Since it takes at least 2.0 mA to cover the 125-250 MHz range, this gave us a

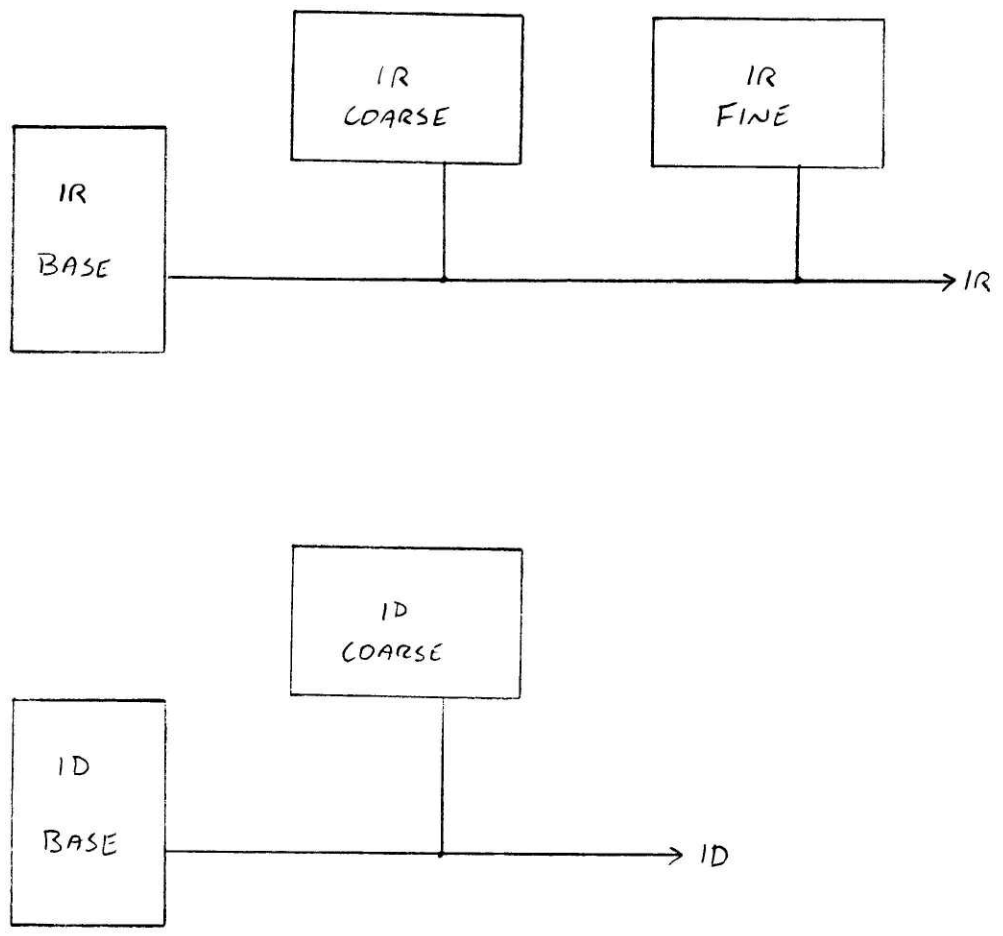


Figure 3-6: DAC Current Drivers

coarse resolution of 7 bits. Setting the FINE DAC's range to 0.12 mA (ie. approximately 4.1/32) gave us another 5 bits of resolution on the IR input.

Thus, we could step through all the frequency settings, keeping approximately a 50% duty cycle using the following steps:

1. Set the IDC, IRC, and FINE registers to 0

2. Set the BASE registers to yield a frequency of just under 125 MHz

3. Increment the FINE register through one coarse step (32 settings)
4. Set the FINE register to 0 and increment the IDC register
5. Increment the FINE register through one coarse step
6. Increment the IRC register
7. Repeat from step 3.

This scheme then gave us

$$\log(128) + \log(2 * 32) = 13$$

bits of resolution for the oscillator frequencies.

3.7 The Diagnostic/Calibration System

The calibration system (see Fig. 3-7) consisted mainly of two counters - a gate counter and an event counter. The gate counter was simply the TMR1 counter of the 80186 driven by a TTL compatible, divided-by-256 version of either the 200 MHz reference oscillator or one of the programmable oscillators. By loading the TMR1 MXB register with the appropriate number, we could produce a signal on the TMR1OUT pin of the 80186 which was high for a known amount of time. The event counter was a 24-bit counter, formed by cascading an 8-bit ECL counter with the 16-bit TMR0 counter on the 80186. This allowed us to count the number of periods of either the reference oscillator or one of the programmable oscillators while the GATE signal was high.

Using these two counters, then, we were easily able to calculate the frequency or period of any of the three programmable oscillators. For example, to calculate the frequency of the RRG oscillator, we would select the RRG as the source for the event counter via DIAGSEL field of the rrgctrl register, and the reference oscillator as the source for the TMR1 of the 80186 via the TMRSEL port. By loading the

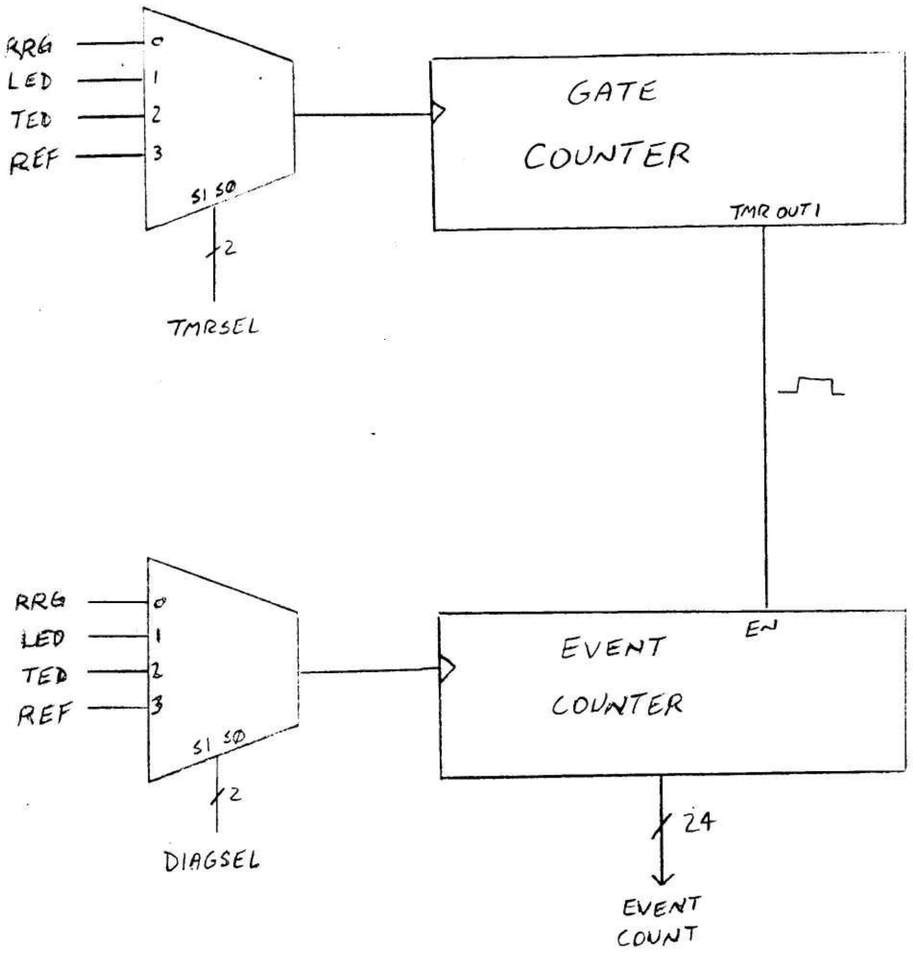


Figure 3-7: Calibration System Block Diagram

TMR1 MXB register with 12500, we got a signal on the TMR OUT1 pin of the 80186 which was high for

256 * 5 ns * 12500 = 16 ms

Thus, the number recorded by the event counter was 16 times the frequency of the RRG oscillator in kilohertz.

If, however, we wished to know the period of the LED oscillator, we would select the reference oscillator as the event counter source, and the LED as the source for TMR1. By loading the TMR1 MXB register with 5000, we knew that the LED period (in nanoseconds) was related to the event count (evcnt) by

evcnt = (5000 * 256 / 5 ns) * period

and the number recorded by the event counter was then simply the period of the selected oscillator in picoseconds times 256.

We used these methods to formulate look-up tables which simplified the task of calculating the DAC inputs to achieve a given frequency output from the oscillators. By taking a frequency measurement for each setting of the IPR and IPD registers, we got the transfer curve of frequency vs. current (see table 2-I) which we used to find the correct settings for the DACs as discussed in section 2.4.

Chapter 4

Critical Performance Issues

4.1 Overview

Throughout the development of this system, we have been concerned with two critical areas of performance: repetition rate and time-jitter. As technology improves and device speeds become faster, the term "useful," for a pulse generator, will be defined in terms of these criteria. A pulse generator will have to perform well in both these areas in order to test a circuit at the limits of its performance.

When testing a high performance circuit or chip, it is desirable to be able to do two things:

1. Drive the DUT at its normal operating speed, and
2. Observe the behavior of the DUT at these speeds.

Currently, high-speed Emitter-Coupled Logic (ECL) chips are designed to run at speeds of up to 250 MHz, while in other technologies, such as Gallium Arsenide, devices run even faster [5]. We therefore need a system which can supply a stimulus signal at such frequencies. In order to observe such a fast signal, we would need to use a very fast measuring device, such as an oscilloscope. Even the fastest oscilloscopes, however, are limited to bandwidths of about 400 - 500 MHz. To look at signals faster than this or to increase the resolution with which we can view a signal, it is often convenient to use techniques such as sampling.

The technique of sampling a periodic signal consists of recording the value of that signal at a given time relative to a fixed trigger time (see Fig. 4-1). For each period of the waveform, if we take a sample at a different time with respect to the

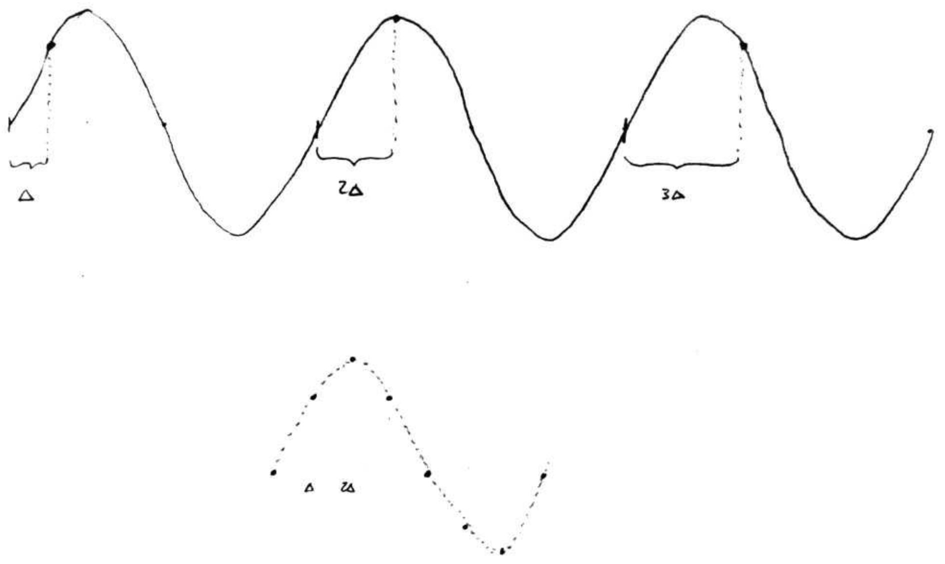


Figure 4-1: Sampling a Periodic Signal

trigger, we can eventually find the value of the signal at virtually any point in the period [1]. In order to accurately recreate this signal, then, it is extremely important that the value of a given sample actually be the value of the signal at that time. Moreover, if the event we are examining is of very short duration, such as the rising edge of an output signal, we want to be able to resolve very small amounts of time between samples [4].

For example, if we have a signal we wish to sample at a time T after the trigger, but instead sample it at time T+j, the voltage we see could be substantially different from the actual voltage of the signal at time T. Thus, any time jitter of the pulse out with respect to the trigger will appear as noise on the scope (see Fig 4-2).

4.2 Factors Affecting Performance

In order to achieve the repetition rate goal of 250 MHz, we used the programmable timer chip discussed in section 3.5, and some ECL circuitry designed to run at this frequency. In order to keep the jitter as small as possible, though, we needed to take into account many other factors.

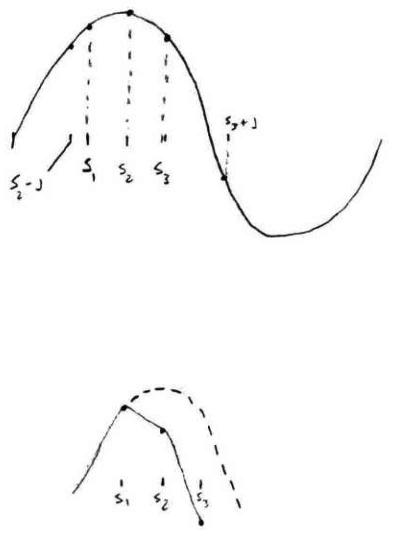


Figure 4-2: Time Jitter Affects a Sampled Signal

The relatively large amounts of jitter present in the devices listed in section 1.1 are due primarily to the analog manner in which they generate the trigger-to-output delay [5]. To reduce this problem, we have implemented the delay generation units digitally (see section 3.5), with a relatively small amount of easily controlled analog circuitry to improve the resolution.

Although there are still a number of factors which contribute to jitter, and we shall now examine how they can be controlled. But first, to discover what these factors are, let us briefly examine what is involved in generating a pulse with this system.

1. A trigger signal is generated which not only goes to the outside world, but also signals the two edge delay generators to begin counting their specified bursts.

2. The edge delay generators generate signals telling the output stage when to start and end the pulse.

3. The output stage converts these signals into a pulse output.

If this is the case, we can get jitter caused by noise on the internal trigger line, noise in the output stage, or by variations in the frequency of the programmable oscillators. Noise in the output stage is beyond the scope of this thesis and will be assumed to be negligible. Noise on the trigger line can be reduced by laying the board out in such a way as to minimize cross-coupling between the trigger and the rest of the circuit (see section 4.3). By far the most significant contribution to the jitter is caused by frequency instability in the oscillators.

The oscillators require a voltage reference, VREF, of -5.0 volts which must be kept steady in order to keep the frequency stable. As long as there is less than 2 millivolts of ripple on the reference voltage input, the oscillator frequency is guaranteed to be stable to within 0.01%. To set this voltage we used a LM145 -5 volt regulator which is specified to have only 150 microvolts of output noise and a maximum change in output voltage of 15 millivolts per volt of input change. Therefore, if the power supply is stable to within 1 tenth of a volt, the reference voltage will be well within the 2 mv tolerance. In a future version of this system, a better regulator could be used which would allow more power supply noise and still stay within this tolerance. The problem of cross-coupled noise on this reference line will be discussed in section 4.3.

Even if the oscillator inputs are completely stable, it is still possible to get drift in the oscillator frequency due to temperature. Since we use look-up tables to set the IR and ID drive currents to achieve a desired frequency, we can eliminate this long-term variation by recalculating the look-up tables. In the present software, this recalibration is done when the user wishes to change a pulse parameter, but it could be done at other times. Since the trigger output is disabled during recalibration, the most convenient time to perform this function would be when the measuring device is not looking for a signal (ie. undergoing its own recalibration or some similar function). It would be beneficial, therefore, to allow for an external signal to cause the pulse generator to undergo recalibration.

Since we now have an oscillator whose output frequency is stable enough for given drive currents, we now have only the problem of noise affecting the ID and IR inputs of the oscillators. Let us first examine the effects of such noise. Since any noise will be small relative to the actual value of the currents, we can treat the period vs. current table as being linear in the region of interest. We will concern ourselves with the area of the table corresponding to small input currents, and thus any noise will have a larger effect. For a BASE current value of .908 mA (ie. a register value of 4B) we get the first four entries of a look-up table as in Table 4-I.

Register Value	Current (mA)	Period (ps)
00	0.908	8001.6
01	0.924	7730.7
02	0.940	7469.4
03	0.956	7221.5

Table 4-I: Period vs. Input Current

The Slope of this curve between points 00 and 01 is

(8001.6 - 7730.7) / (908 - 924) = -16.93 ps/μA

for common mode noise present on both the IR and ID inputs. This means that, for each μA of noise on one of the lines, the period will change by 8.47 ps. The power supply sensitivity of the DAC08 digital-to-analog converter is, at worst case, .01% of output change per 1% change in power supply voltage. A 0.1 volt change in the power supply voltage will thus create a common mode period change of

[(0.1 / 15) * 100] * (924 * .0001) * 16.93 = 1.04 ps

which is within the desired tolerance. At the other end of the look-up table, we get the values of Table 4-II. Between points AB and AC, this part of the curve has a slope of

(3998.3 - 4033.5) / (3660 - 3644) = -2.2 ps/μA

and we can again calculate the period change for a 0.1 volt change in the power supply as

Register Value	Current (mA)	Period (ps)
A9	3.612	4105.3
AA	3.628	4069.2
AB	3.644	4033.5
AC	3.660	3998.3

Table 4-II: Period vs. Input Current

$$[(0.1 / 15) * 100] * (3660 * .0001) * 2.20 = 0.54 \text{ ps.}$$

We see here that the decrease in the magnitude of the slope of the period vs. current curve more than makes up for the increased magnitude of the current variation.

4.3 Layout Considerations

Even if the voltage supplied by the power supply is perfectly stable, we can still get effective power supply noise due to common impedance coupling on the ground plane [3]. This effect is particularly troublesome in the case of TTL circuits which induce enormous current spikes during gate transitions. The best way to deal with problems of this sort, then, would be to separate the system into modules according to power supply needs [2] like so:

1. TTL components, including the processor, memories, and registers,
2. ECL components, including the reference oscillator,
3. Digital-to-analog converters,
4. Analog components, including trigger input and output circuitry, and
5. Programmable oscillators

We can reduce the coupling between these modules by dividing the ground plane, as shown in figure 4-3, and effectively instituting a multipoint ground system even though all power supplies are referenced to the same ground [2].The current requirements of the TTL components are met by adding a second ground path back to the power supply, besides the one used by the rest of the circuit. Bypass capacitors are used in the vicinity of each digital component, including ECL and the programmable oscillators, to reduce the ground coupling even further.

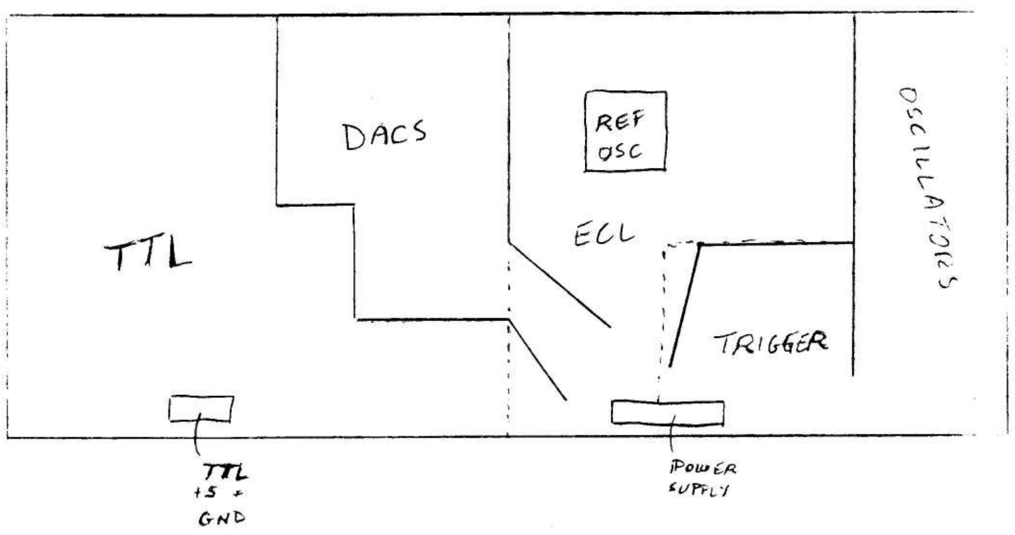


Figure 4-3: Test Circuit Board Layout

With power supply noise thus reduced, the other major contribution to the output jitter is capacitive and inductive coupling on important signal lines like the internal trigger line, the IR and ID inputs of the oscillators, and the VREF line to the oscillators. For the test circuit, the board was laid out as in figure 4-3, where the main concerns were separating the circuit by power supply needs, as discussed earlier, and keeping the trigger line as short as possible. By keeping the trigger line short, we not only reduce the opportunities for coupling (and thus for jitter) but we also keep the skew as small as possible between the times that each edge delay generator sees the trigger signal.

The biggest problem with this arrangement, however, is the fact that the reference oscillator was placed between the "DAC farm" and the programmable oscillators, causing a large amount of noise on the IR and ID inputs to the oscillators. It is this noise that is believed to be the main factor in the amount of jitter seen in the test circuit (see section 5.2), although the noise could be reduced in a future version by one of two ways.

The first method would be to rearrange the board so that the "DAC farm" is nex to the oscillators (perhaps as in figure 4-4). This would not only remove the reference oscillator as a noise source to the oscillator inputs, but would also shorten these lines and make them less susceptible to cross-coupling noise.

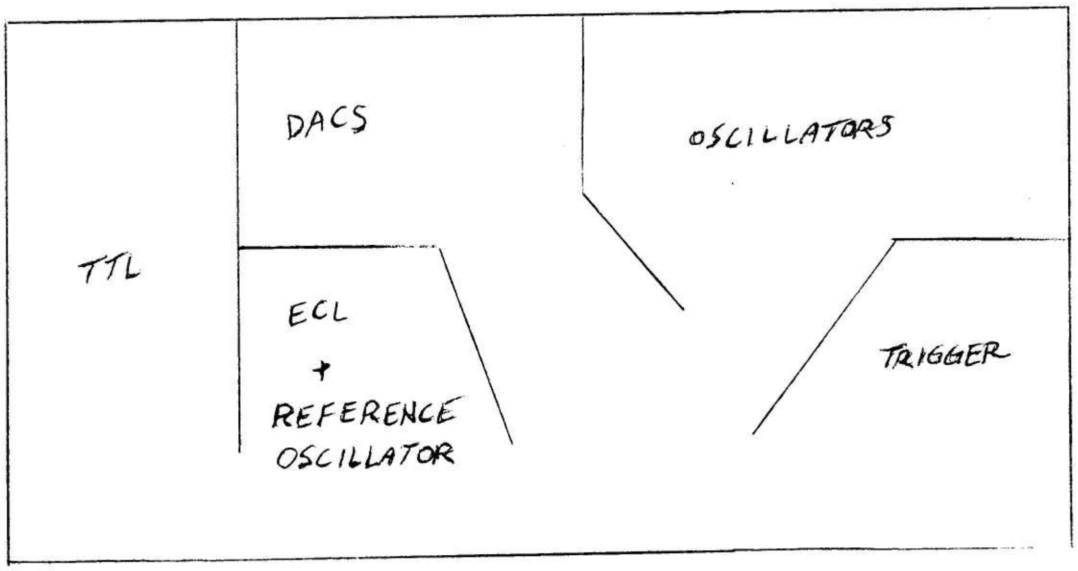


Figure 4-4: Possible Board Layout to Reduce Noise

Another method which would reduce the effect of noise on the IR and ID lines would be to keep the ground plane between them and any oscillating lines, although the large number of such lines would make strict adherence to this constraint difficult. It might be possible to use two layers of the board for signals which must be quiet and separate these from the other signal layers by the power and ground planes, but this would make the task of circuit board layout significantly more difficult.

The test circuit consisted of six signal layers, the ground plane, and a voltage plane which was used to supply the appropriate voltages to the different parts of the circuit (ie. +5 for TTL, -5 for ECL, and +15 and -15 for the DACs). Subject to the given component placement, however, the autorouting program we used was still only able to achieve a 90% completion rate. It is possible that this could be improved by laying the components out in a different pattern, but it is also likely that imposing such constraints as mentioned above will significantly reduce this completion rate.

Chapter 5

Results and Conclusions

5.1 Test Procedure

To test the jitter performance of the system, we would need to supply a signal to the trigger input of the system and observe the trigger output. By seeing how much jitter is present on the trigger output, we would know how much jitter is present in the trigger selection circuitry itself. Although this does not affect the trigger-to-pulse jitter, we can use this as a reference to which we can compare the pulse output and thus measure the jitter of the system. Using this signal as the trigger source, we can generate a pulse output and examine the leading edge delay signal. By measuring the jitter on this signal, and subtracting the jitter due to the trigger selection, we can get a fairly accurate measure of the jitter performance of the system.

5.2 Results

Unfortunately, we were unable to implement the external trigger input circuitry, and so had no absolute reference from which to calculate the system jitter. However, by generating an internal trigger, we could look at the edge delay signal and approximately measure the jitter in this manner. When the output of the timer chip was examined with a scope, it was estimated that the system had a time-jitter of 60 - 80 ps. This is outside the range we were hoping for, but a significant part of this jitter was due to the large amounts of cross-coupled noise on the IR and ID inputs to the programmable oscillators, as discussed in section 4.3.

The fact that the power supply voltages exceeded the 0.1 volt specification discussed in section 4.2 also made a significant contribution to the jitter.

5.3 Conclusions

The purpose of this thesis was to design the control portion of a high-performance pulse generator which would run at frequencies of up to 250 MHz, and have output jitter of approximately 5 ps. A novel approach was implemented which used both analog and digital techniques to achieve a timebase that is comparable to state-of-the-art pulse generators in terms of speed, and which could be made to be an order of magnitude better in terms of stability. Moreover, by implementing certain parts of the system, such as the oscillators and the calibration system, in faster technologies, the pulse generator could be made to run even faster without significantly altering the present design.

Although the test system failed to exceed current stability standards, it nonetheless managed to meet these standards. Since there were many factors identified which contributed to jitter of the system, and procedures suggested which would lessen the effects of these problems, it is believed that, with proper care taken in printed circuit board layout and signal routing, as well as some minor software enhancements, the system can made to meet the proposed specifications of less than 5 ps of jitter, or at least to have less jitter than current state-of-the-art programmable pulse generators.

References

[1] Chris Everett.
1 GHz Digital Scope Handles Repetitive High-Speed Logic Signals.
EDN , November 15, 1984.

[2] F. F. Mazda.
Electronics Engineer's Reference Book 5th edition.
Butterworth & Co, Ltd., London, 1983.

[3] Henry W. Ott.
Noise Reduction Techniques in Electronic Systems.
John Wiley & Sons, New York, 1976.

[4] Art Porter.
Sampling Sees Skinny Signals.
Electronics Week , January 7, 1985.

[5] Charles H. Small.
Benchtop Pulse Generators Address High-Speed Applications.
EDN , July 25, 1985.