1

CANADIAN THESES ON MICROFICHE

I.S.B.N.

THESES CANADIENNES SUR MICROFICHE

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Ottawa, Canada
K1A 0N4.

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Division Division des thèses canadiennes

Ottawa, Canada
K1A 0N4

59935

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

DIMITRIS CAELIS

Date of Birth — Date de naissance

23/8/57

Country of Birth — Lieu de naissance

GREECE

Permanent Address — Résidence fixe

61 GERMANOU

PATRAS GREECE

Title of Thesis — Titre de la thèse

"A MICROPROGRAMMABLE TEST PATTERN
GENERATOR"

University — Université

CARLETON

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

MASTER OF ENGINEERING

Year this degree conferred — Année d'obtention de ce grade

1982

Name of Supervisor — Nom du directeur de thèse

Prof. J. P. Knight

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date

20/8/82

Signature

Caelis

A MICRO-PROGRAMMABLE TEST PATTERN GENERATOR

by

© Dimitris G. Carelis, Dipl. Eng.

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

Department of Electronics
Faculty of Engineering
Carleton University
Ottawa, Canada

August 1982

The undersigned recommend to the Faculty of
Graduate Studies and Research, the acceptance of the thesis:

"A MICRO-PROGRAMMABLE TEST PATTERN GENERATOR"

submitted by Dimitris G. Carelis in partial fulfillment of the
requirements for the degree of Master of Engineering.

Thesis Supervisor,
J. P. Knight

Chairman,
Department of Electronics
A. R. Boothroyd

August 1982

I

ABSTRACT

This thesis describes a hardware system to generate functional test patterns for digital ICs in real time. The test patterns are generated from algorithmic descriptions which may be hand coded in the form of microcode or in the future, compiled from a high level algorithmic test pattern generation language such as ANGEL.

Algorithmic generation of test data offers the major advantage of data suppression in a useful form. Other methods employ a RAM to store the test vectors and then burst them to the device under test (DUT).

The proposed pattern generator uses five registers allowing the DUT pins to be partitioned into up to five groups. These registers provide the necessary control for timing variations, mask and input/output switching "on-the-fly".

A typical register was constructed using AMD-2900 bit-sliced devices.

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor, Professor John Knight for helping to define the direction of this work. His review and very valued constructive criticism are truly appreciated.

I would further like to thank Mr. Russ Fields, Director of Semiconductor Testing, Mitel Corporation, for helpful discussions and valuable technical advice. Without his assistance this project would not have been possible.

I wish also to thank Mitel Corporation for the financial assistance for this work.

The typing of Miss Diane Dodds also deserves credit.

TABLE OF CONTENTS

CHAPTER 1	INTEGRATED-CIRCUIT TESTING	1
	1.1 Introduction	1
	1.2 What Is Testing	1
	1.3 Test Types	2
	1.3.1 DC Parametric Tests	2
	1.3.2 AC Parametric Tests	2
	1.3.3 Functional Tests	3
	1.4 Automatic Test System	3
	1.5 DUT Interface (Pin Electronics)	6
	1.6 Testing Requirements	8
CHAPTER 2	EVOLUTION OF THE TEST PATTERN GENERATION PROCESS	11
	2.1 Introduction	11
	2.2 Evolution of Digital Word Generators	11
	2.3 RAM-Based Word Generators	12
	2.4 Recent Developments	12
	2.5 Impact of Microprocessors	17
	2.6 Incorporating Bit-Slice-Logic	17
	2.7 Introducing the RALU-Based Word Generator	20
CHAPTER 3	ALGORITHMIC FUNCTIONAL-TEST-PATTERN-GENERATION	24
	3.1 Introduction	24
	3.2 ANGEL Description	25
	3.3 General Architecture for Algorithmic Test Pattern Generation by Hardware On-The-Fly	27
	3.4 Generator Operation	30
	3.5 Bit-Sliced Implementation of a Register Module	31
	3.6 Realization of Bit-Slice RALU and Controller	31
	3.7 The 2901 RALU and the 2910 Sequencer	32
	3.7.1 The 2901 RALU-Slice	32
	3.7.2 The 2910 Sequencer	40
CHAPTER 4	HARDWARE IMPLEMENTATION OF ONE 8-BIT REGISTER MODULE	46
	4.1 Introduction	46
	4.2 Register Module Architecture	46
	4.3 Processing Section	49
	4.3.1 Combining Two 2901's	49
	4.3.2 Shift Linkage	51
	4.3.3 Description of the Various Shift Modes	51
	4.3.4 Eight-Bit Processing Section	56
	4.4 Control Section	58

4.4.1	Microprogram Control	58
4.4.2	DO and AC Loading of the 2910	59
4.4.3	Clock Generator-External Control	61
4.4.4	2910 Initialization	62
4.5	Pattern Memories	63
4.5.1	Pattern Memories	63
4.5.2	Loading of Memories	64
4.6	Timing Analysis	64
4.6.1	Control Section Timing	66
4.6.2	Processing Section Timing	73
4.6.3	Timing of Pattern Memories	81
4.7	Programming Example	81
CHAPTER 5	CONCLUDING MATERIAL	88
5.1	Summary	88
5.2	Conclusions and Recommendations	88
REFERENCES		93
APPENDIX A:	Wiring Diagrams	
APPENDIX B:	Wiring Diagram of a 16-bit Processing Section. Using Carry-Look-Ahead Logic	
APPENDIX C:	Device AC Specifications	

LIST OF SYMBOLS

DUT	Device Under Test
V_{CH}	High clamp voltage level
V_{CL}	Low clamp voltage level
V_{RH}	High reference level
V_{RL}	Low reference level
T_0	Basic test cycle
NRZ	Non-Return-to-Zero
RTZ	Return-To-Zero
RTO	Return-To-One
XOR	Exclusive OR
XNOR	Exclusive NOR
RALU	Register-Arithmetic-Logic-Unit
RM	Register Module
PLA	Programmable Logic array
PC	Microprogram Counter
I_{OL}	Output low current
I_{IL}	Input low current

CHAPTER 1

INTEGRATED CIRCUIT TESTING

1.1 Introduction

Today's movement toward VLSI devices has placed extreme demands on the test process.

As IC devices become more complex test departments must develop the attendant test equipment that fully and economically satisfies the increasing test requirements.

The rise in the amounts of test data that are needed to stimulate the inputs or to be compared with the expected output data of these complex devices dictates new techniques for the test pattern generation process.

This thesis focusses on the design and hardware implementation of a digital word and data generator as part of a general purpose test system.

1.2 What Is Testing?

Before discussing testing it is necessary to understand what the term testing means. At any time instant, any system or component, no matter how complex, can be regarded as a black box having a set of input and output terminals. All testing has to do, is to determine the correct operation of the box. This is done by applying stimuli on the input and observing the response on the output.

In testing logical functions the stimuli and the expected response, that are known as test vectors, are generated by the test system. Clearly, the more complex the box is, the more test vectors are needed for a satisfactory test and the more complex the test system becomes.

1.3 Test Types

IC testing is usually divided into two types: analog and digital. This thesis will be concerned with digital ICs only.

There are three types of tests commonly done on digital ICs.

1. DC parametric tests.
2. AC parametric tests.
3. Functional or logical tests.

1.3.1 DC Parametric Tests

DC tests verify specific parameters in terms of current and voltage, for example, the total current drawn by the DUT, voltage high/low levels, output drive currents, input noise thresholds, etc. DC tests are performed by forcing a current and measuring the resultant voltage or vice versa.

The important characteristics to be considered when performing a DC parametric test are accuracy and test time per parameter per device pin.

1.3.2 AC Parametric Tests

AC tests verify time-related parameters in terms of nanoseconds, for example propagation delays, set up and hold times, access time, rise and fall times, etc.

These tests are performed by measuring the time interval between two well predefined events. Important AC testing considerations are edge to edge timing and repeatability.

1.3.3 Functional Tests

Functional tests are the tests required to verify the correct performance of the DUT logical operations or functions with respect to the design specifications. As an example a functional test for an AND gate with inputs A, B must show if its output is $A \cdot B$ for any input set $\{A, B\}$.

Functional tests are performed by stimulating the inputs and comparing the device outputs to predicted data patterns.

Because of the fact that today's components often involve dynamic logic the test vectors should be generated and applied to the DUT at rates above the minimum frequency at which the device is specified to function properly (keep-alive frequency). Besides most of the AC parameters are verified by performing functional tests at high speed.

Furthermore, VLSI designs require extremely lengthy test patterns to be exercised thoroughly. The most important testing considerations include the efficiency of pattern generation, edge to edge timing control and input/output and mask switching. These considerations will be explained later.

1.4 Automatic Test System

A simplified block diagram of a general purpose test system is illustrated in Figure 1.1. Its basic components include a main processor,

test pattern generation modules, a timing generator, a format module, a voltage level control unit, analog forcing/measuring units for current-voltage measurements and a DUT interface, more often known as pin electronics. The function of the main processor is basically to house the software operating system and the test programs necessary to control the operations of the system components.

The timing generator provides programmable clock signals that are used to drive the test pattern generators, the format logic and the DUT interface circuitry.

The voltage level control circuits provide the high and low voltage levels to be driven to the DUT inputs and the high and low voltage levels that are used for comparison with the DUT outputs.

The analog forcing/measuring units are used for DC parametric tests by forcing a voltage and measuring the resulting current or vice versa. These units can also supply the voltage and current for the DUT operation by being connected to the DUT power supply pins. The pattern generation modules generate in real time, the test vectors to be driven through the format logic and the pin electronics circuitry to the DUT (Device Under Test).

The DUT interface is an important part of a test system. Most of the tester specifications are given at this part of the system. Examining a typical DUT interface one can understand the kind of signals required for an IC test.

FIGURE 1.1

Block diagram of a general purpose test system.

1.5 DUT Interface (Pin Electronics)

A typical one pin DUT interface is shown in Figure 1.2. Usually pin electronics are universal in nature, meaning that they can be connected to any kind of DUT pin (input, output, I/O, power or ground). Relay K_1 closes if the DUT pin is power or ground connecting a bias supply. Relay K_2 closes if the DUT pin is an input, output or I/O. K_3 closes in the output or I/O mode and connects a voltage comparator to the device pin. K_4 and K_5 are used to connect a DC measuring unit for DC parameter verification.

In the input mode the FET switch shown provides a low impedance path from the PIN DRIVER to the DUT pin, so that a high or low clamp voltage level (V_{CH} , V_{CL}) can be applied to the DUT pin, depending on the logic level of the PIN DRIVER input. In the output mode the FET switch is in high impedance allowing the DUT output level to be compared to the high or low reference level (V_{RH} , V_{RL}). In a "DON'T CARE" situation the comparator output is masked and the DUT output is ignored. In a "CARE" situation the MASK is "off" and the DUT output is compared to the expected value (V_{RH} or V_{RL}) at STROBE time. The MASK GATE is controlled by the MASK DATA input.

The DATA input after passing through the FORMAT MODULE is used to stimulate the DUT via the PIN DRIVER. The same input provides the DUT expected response.

The TRI DATA input controls the I/O SWITCHING through the FET SWITCH. This input is also formatted by passing through the FORMAT MODULE.

FIGURE 1.2

DUT-interface. Bits "DATA", "TRI DATA" and "MASK DATA" determine the DUT pin status. Clock pulses "DATA CLK", "TRI CLK" and "STROBE" along with the MODE used determine the exact time interval at which the status information is applied to the DUT pin.

The FORMAT MODULE determines the proper time point and for how long (within the basic test period T_0) the test data will appear at the DUT pin. This is done by performing a logical function (e.g., AND, OR, XOR, etc.) between the test data (stimuli/response DATA or TRI DATA) and a clock pulse (DATA CLK or TRI CLK respectively) which comes from a timing generator. DATA CLK, TRI CLK as well as STROBE pulse have programmable rising and falling edges in the sense that they can be placed anywhere within the test period, T_0 . Formats usually available are normal or Non Return to Zero (NRZ), Return to Zero (RZ), Return to One (RTO), exclusive OR (XOR), etc. (Figure 1.3).

1.6 Testing Requirements

To comprehensively test a complex VLSI device with worst case timing the test system should satisfy the following requirements:

1. It should be able to supply a new logic high or low level to the "DATA" (stimuli/expected response), "TRI DATA" (I/O control), and "MASK - DATA" (compare/ignore) inputs of the DUT interface every T_0 time units. These three information bits determine the state of the DUT pin (input, output or don't care).

2. It should be able to set the application time of the stimuli, the time of switching from input to output and the time where the DUT response is examined, so that they occur anywhere within the T_0 test cycle. This translates into the ability to place the rising and falling edges of the "DATA CLK", "TRI CLK" and "STROBE" pulses in any portion relatively to each other or to the test cycle (T_0) start or end, with high resolution.

FIGURE 1.3

Test data waveforms as they may arrive at the DUT pin after being formatted by the format circuitry. T_0 is one test clock cycle.

3. It should have the ability to alter the position of rising and falling edges of "DATA CLK", "TRI CLK" and "STROBE" pulses "on-the-fly" from test cycle to test cycle.

4. It should have the ability to change the timing formats "on-the-fly" by loading a new value into the MODE LATCH.

Almost without exception present IC testers do not satisfy the requirements described above. They provide fixed timing, edge placement, MASK and I/O switching and format selection throughout a test sequence. Changes require stopping the test sequence. Such systems may need more than one uninterrupted test sequence to exercise a device thoroughly.

To achieve the requirements above one method is to employ a digital word generator providing the "DATA", "TRI DATA" and "MASK DATA" information bits and a timing generator generating a number of independently programmable clock channels. Furthermore, the digital word generator should provide additional information for controlling the timing generator and the formatting circuitry.

CHAPTER 2

EVOLUTION OF THE TEST PATTERN GENERATION PROCESS

2.1 Introduction

Efficient application of test patterns to the device under test (DUT) has been an important testing problem. A new scheme for calculating test patterns in real time is the major theme of this thesis. Test pattern generators provide the digital input stimuli for the DUT and supply the expected response for comparison with the actual output of the device.

After briefly discussing the present technique, a new approach to the subject will be introduced.

2.2 Evolution of Digital Word Generators

Pulse generators were the very first form of digital word generation instruments. For early requirements for digital testing, pulse generators allowed users to vary frequency, amplitude, width and rise and fall times for one channel of digital information. Due to their inability to produce more than one word per time period, they could not meet complex digital signal generation requirements. Other methods of pattern generation followed.

However, digital word generators did not truly emerge until the cost effective Random Access Memory (RAM) made its appearance in IC form.

2.3 RAM-Based Word Generators

RAM made the complex digital word generation process a lot easier, opening a new era in the testing field.

RAM based word generators have the architecture depicted in Figure 2.1.

Storing desired patterns of ones and zeroes in the memory, the generator shifts them out via the output register. By storing the appropriate data, any arbitrary digital waveform can be produced to stimulate the device under test. Changing the frequency of the clock generator determines the test pattern application rate, while controlling the address counter determines the order in which the stored data are applied to the DUT.

2.4 Recent Developments

Recent developments have increased the capabilities of the RAM-based word generator.

First, RAM fabrication technology increased both capacity and speed, allowing word generators to occupy less board space and operate at higher test rates.

Address control represents a second recent development in word generators. Additional logic permits users to produce a pattern burst or loop by programming a start address and an end address, as illustrated in Figure 2.2. These addresses are not necessarily the first and the last address of the pattern RAM. This way, users can address any group of test vectors as may be required by a specific test.

FIGURE 2.1

RAM-based word generators have a common architecture. The desired patterns of ones and zeroes in the memory are shifted via the output register to the D.U.T.

FIGURE 2.2

RAM-based pattern generator equipped with looping logic. A pattern burst or loop can be performed by programming the start and end addresses.

A third improvement overcame the fixed period limitation of the word generator. The generator of Figure 2.2 produces words at a rate directly related to the clock period T_0 . The generator cannot supply the same word for more than one clock period unless this particular word is stored in more than one memory location. For example, keeping the generator output unchanged for one thousand clock periods means wasting one thousand memory locations, but a generator incorporating a state-to-state programmable clock can do this very efficiently. Such generators allow users to program the time duration (any multiple of the currently used clock period) of each individual output state.

Figure 2.3 shows a generator incorporating this technique. The RAM consists of two sections, the test pattern RAM which contains the data to be driven to the DUT, and the timing RAM which contains the corresponding time duration of each test vector (i.e. how many times the test vector is to be repeated). In some designs, in addition to the timing information, this memory contains a few more bits to control other peripheral devices. After both memories have been loaded, a movement from address to address takes place only when the count down circuitry overflows. This technique is called "timing simulation" and makes generation of repetitive timing waveforms much easier. But even a generator employing the timing simulation technique cannot produce an output pulse signal with width less than the period of the reference oscillator. This is simply because the system cannot afford any higher clock rates. To overcome this, the word generator output can be formatted by format circuitry before being applied to the DUT as discussed in Chapter 1.

FIGURE 2.3

In this generator, the pattern memory contains the test data while the timing memory contains the time information; i.e. how many times each particular pattern is to be repeated.

2.5 Impact of Microprocessors

At this point it should be clear that the proper and efficient operation of word generators requires a lot of control information. Microprocessors provide the necessary control to increase word generator's flexibility and performance. Filling the memories with desired data, inserting, deleting and modifying test patterns are easy tasks for a generator controlled by a microprocessor.

While the standard LSI microprocessor can aid word generator operations, the high speed of the BIT-SLICE family seems to be even more powerful for this class of application. Due to its speed, the BIT-SLICE microprocessor cannot only supervise, but also exert real time control over the word generator. In other words, users may change the control parameters of the word generator to permit output pattern alteration in real time.

The characteristics of BIT-SLICE DEVICES is their high speed of operation compared to standard LSI microprocessors, and their ability to be cascaded together to form a processing section of any word length. Other properties of BIT-SLICE devices will be discussed in a later chapter.

2.6 Incorporating Bit-Slice-Logic

Figure 2.4 illustrates the block diagram of a RAM-based word generator incorporating Bit-Slice-Logic. Instead of an address counter and surrounding logic, another much more intelligent device is used. This device belongs to the Bit-Slice family and handles the whole address

7

FIGURE 2.4

As a part of a digital word generator, the BIT-SLICE sequencer allows pattern alteration "on-the-fly".

generation process. More specifically, its purpose is to determine the address of the test vector that is going to be applied to the DUT in the next clock cycle. It is called microprogram sequencer or controller because it is the basic unit of the control section in every microprogrammed machine. In addition to an instruction, it takes as input one or more addresses and one or more status conditions (flags) and from these, determines and sends to the memory the address of the next test vector. Along with replacing a large amount of previously used logic, microprogram controllers can supply enhanced test pattern generating features like conditional/unconditional jumping and subroutine nesting; all on the same chip.

As Figure 2.4 shows, the memory is again partitioned into two sections: the pattern memory and the sequencer control memory. The latter memory section contains the control information necessary for the sequencer operation. It is also partitioned into two subsections: instruction and operand. Common instruction types are: continue; jump; subroutine call/return; and loop. The operand subsection specifies things like where to jump, which subroutine to call, or how many times a particular group of instructions-patterns has to be repeated(loop). Since many of the sequencer instructions are conditional, there is a condition input driven by the condition bus. The external sequencer control inputs are used only for initialization or test program debugging.

Besides allowing features, such as conditional branching, looping, and subroutine nesting, and offering high speed operation, the bit-slice controller permits users to create a pattern by concatenating several different patterns that reside in non-contiguous locations in the pattern memory in real time.

A detailed description of such a device will follow in a later chapter.

2.7 Introducing the RALU-Based Word Generator

The Bit-Slice controller, endowed the RAM-based generators with new important features and made the digital waveform generation, necessary for testing complex devices, more efficient.

In this section, a new form of digital word generators will be introduced, the Register-Arithmetic-Logic-Unit based generator.

As an example, assume that a particular test requires generation of the sequence consisting of the following consecutive numbers:

00, 01, 02, ..., FF

Using a RAM-based generator, one has to store all these numbers, each in different locations in the pattern memory, and control the sequencer to select them one after another, from 00 to FF. Two hundred and fifty six memory locations are needed and the sequencer has to produce 256 different addresses.

The question that arises is: Is there not a better way to do this? Instead of storing all these numbers and using a large number of memory locations, cannot arithmetic logic be employed to produce them "on-the-fly"? Due to the Bit-Slice performance and flexibility the answer is: Yes, the RAM holding the test pattern sequences can be replaced by a bit-slice Register-Arithmetic-Logic-Unit which can easily generate such sequences using only one instruction.

RALUs are found inside every central processor unit (CPU). They consist of two parts, the bank of registers and the arithmetic logic unit. The registers, usually known as accumulators or working registers, serve as inputs to the ALU or as destination for its results. The ALU is the unit which actually performs the data manipulation. It has two inputs for operands and one output for the result. It usually operates on all the bits of a word in parallel and can perform all or part of the following operations:

<u>Arithmetic</u>	<u>Logical</u>
Add	OR
Complement	AND
Subtract	XOR
Increment	NAND
Decrement	NOR
	XNOR
	Complement

In some architectures one of the operands must always be in a special register (accumulator) and the result of the ALU operation is always transferred to this register. However, in more general designs, any two of the registers can contain the operands and the result of the ALU operation can be transferred to any one of them. Figure 2.5 illustrates an RALU-based word generator. Notice the elimination of the pattern memory. Its place has been taken by the bit-slice RALU. Test data (stimuli or expected response) are not stored in RAM, instead, test data are produced by the RALU on-the-fly using the control memory which controls both the sequencer and the RALU. The test data generation

FIGURE 2.5 RALU-based digital word generator can algorithmically generate test data on-the-fly.

procedure is very similar to how a computer fetches and executes an instruction. More specifically, the sequencer fetches the instruction (control data), while the RALU executes it and delivers the result (test data) and this happens in just one clock cycle. Test patterns are produced in real time according to some algorithm stored in the control memory.

The major advantage of this architecture is that executing the same instruction N-times may imply the generation of N different test vectors. One location of the control memory may be the source of a complete test pattern sequence. Two instructions are more than enough to generate the 256 different numbers of the example mentioned above. Besides the RALU-based generator has all the important features like timing simulation and reaches or approaches very closely the speed of the RAM-based generator. Little has been lost!

CHAPTER 3

ALGORITHMIC FUNCTIONAL-TEST-PATTERN-GENERATION3.1 Introduction

Effective testing of an LSI device requires the ability to handle a large amount of test data. The millions of test vectors needed for exercising up-to-date VLSI components may reside in large bulk memories but in this form they are not useful to serve different types of applications. Besides, creating entering or modifying the vector information is a very costly and time consuming task. To overcome this problem, users need suitable tools to reduce these large amounts of test data into a more useful and comprehensible form. Data compression techniques like signature analysis, that compresses a whole response sequence into one word, and random data generation, which generates test vectors instead of storing them, are not exact enough for current LSI testing. However, algorithmic test-vector generation techniques that are able to produce methodical test sequences, provide a potential solution to the problem of effectively creating and using large vector sets. For effective use, the test-vector generation algorithms have to be supported by some easy to use, high level language.

MITEL has developed ANGEL (Algorithmic Pattern Generation Language), a high level language offering a solution to the functional test pattern generation process. This chapter will present the basic structure of ANGEL and then describe a new architecture for algorithmic hardware generation of test patterns in real time, utilising many ANGEL features.

3.2 ANGEL Description

ANGEL is a high level language used to produce functional test patterns for integrated circuits and printed circuit boards. It simplifies the creation and entry of the large stimulus/response vectors required for device simulation and testing. This simplification is obtained by using short easy to understand high level algorithms to generate long pattern sets. Being close to user logic, ANGEL reduces test data into easy to use software modules and shortens the time required to produce test programs. The algorithmic description of test patterns allows a more complete coverage of possible device faults by making test programs easier to write and enhances test vector maintainability.

The language itself is block-structured, so that programming is quick, simple and relatively error-free. Including DO-WHILE and IF-THEN-ELSE statements and avoiding GO-TO statements contribute to the solid structure of the language.

ANGEL supports the following data types:

1. pins or testpoints
2. registers
3. local or global variables
4. constants in hex, octal, decimal or binary representations

The logic states supported are:

True	A logic value of 1
False	A logic value of 0
Don't Care	A logic value of X
High Z	A logic value of Z (high impedance)

The most important construct in ANGEL is the variable length register. A register is defined as an ordered group of pins or test-points (partial vector). Registers are completely isolated from each other. Modifying a register will not affect any other one, even if these registers contain common pins or test points.

Test patterns can be generated through any or all of the following:

Arithmetic calculation (e.g. addition, subtraction between registers)

Boolean calculation (e.g. OR, AND)

ANGEL function (e.g. often used patterns such as "all 1s")

User function (e.g. the user may write routines to produce its own patterns)

Other functions like Set/Clear, applying to pins or registers, Increment/Decrement, Shifts and Rotates, applying to registers, are also supported by the ANGEL system. Application of the generated test patterns is performed by using the "APPLY" statement, which acts on a special output register. This register cannot be used in any calculations or functions and cannot be changed or affected by any other statement. Its exclusive purpose is to apply test patterns to the device. The output register specifies the state of every device pin and the only pins that may change their state are the ones specified by the "APPLY" statement. Unspecified pins are left in their previous state.

Beyond the capability of functional test pattern generation, ANGEL supports timing functions such as CLOCK, which specifies the test rate and STROBE, which specifies the rate at which the device outputs are checked.

3.3 General Architecture for Algorithmic Test Pattern Generation by Hardware On-the-Fly

In this section a hardware is introduced based on ANGEL concepts.

In order to algorithmically generate a functional test pattern, it is very helpful to partition the DUT pins into registers (like ANGEL does) and dedicate a specific part of the pattern generator to each register. Registers contain a certain number of pins that might have some common function. For example, a register may consist of eight data pins, sixteen address pins, or four control pins. This partitioning allows users to write programs looking at groups of DUT pins rather than looking at all pins as a whole, it simplifies the required algorithms and gives the idea of a register organized architecture as depicted in Figure 3.1. The generator described here consists of five identical Register Modules (RM). Each register module is an autonomous data generator itself; that is, it can be connected to a particular group of output, input or I/O pins, apply a stimulus and/or observe a response. A RM is nothing else than a RALU-based digital word generator which has been introduced in the last chapter. The number "five" is reasonable, permitting the DUT-pins to be partitioned into up to five groups (registers). Another hardware restriction is the number of pins to be assigned to an individual RM. A maximum of twenty four pins is needed for things like address pins in some microprocessors but making all five RMs 24-bit wide does not seem to be the ideal solution as far as cost is concerned. As an initial attempt to optimize hardware cost versus performance, the following configuration was chosen:

5

FIGURE 3.1

This scheme for digital functional testing employs a word generator, consisting of five similar register modules operating in parallel.

	<u>Maximum Number of Pins</u>
RM #1	8
RM #2	16
RM #3	16
RM #4	20
RM #5	24

Note that the number appearing at the right column in the table above is the maximum number of pins that can be assigned to the corresponding RM. For example, RM #3 can handle any number of pins less than or equal to 16.

These five register modules can test devices with a maximum of $(8+16+16+20+24=84)$ pins (not counting power supply pins). However, this is an extreme case because the partition that is selected by the user and the generator configuration have to be in coincidence. This upper limit, being well beyond today's pin-outs, allows the generator of Figure 3.1 to be used for testing a variety of up-to-date components. Besides an extra register can be added for future applications.

In order to keep the various operations of all five RMs in synchronization, RMs are interconnected through a few FLAG lines. This way, every RM can at any time examine the FLAGS coming out from any other RM, itself included. FLAGS are the only form of communication between RMs. Data transfer between RMs is of much less importance and is not supported by the system hardware. Flag interconnection is enough to allow the efficient parallel operation of all five RMs, as well as the relatively easy programming of the system.

The input bus is used to load the programs into every RM. The FAIL signals coming out from the RMs are active when the DUT response does not match the expected one. Once activated, they cause the pattern generation (test) to stop and flag the main processor of the system to take care of the situation. The switching matrix or interface adaptor provides the necessary interface between the generator and the DUT.

3.4 Generator Operation

The operation of the generator described above is straightforward. First, the user has to group the DUT pins into five or fewer registers. Second, he must assign each of these registers to the proper RM. For instance, a register containing 16 pins cannot be assigned to RM #1 or RM #2 because they handle a maximum of 4 or 8 pins respectively. However, it can be assigned to any of RM #3, RM #4, or RM #5. This information, once entered, will be driven by the overall system control logic to the switching matrix and the proper connections between RMs and DUT pins will be established. This can also be done by cross connections on a probe card constructed especially to test a particular device. The next step is to load every RM with the algorithms (programs) required to generate the desired test patterns. Note that the information needed for the synchronization of the RMs must be included in the algorithms to be loaded. After loading the programs, the functional testing is ready to start. The Bit-Slice generator is ready to generate stimulus and expected response "on-the-fly".

It is worth mentioning once more that each RM acts on, and is connected to a specific predefined group of pins of the DUT. Also, note that a

pin cannot belong or be connected to more than one RM. This is a hardware restriction and does not apply to ANGEL.

The detailed hardware description of an RM is the subject of the next chapter that presents the prototype of an 8-bit register module as it has been implemented.

3.5 Bit-Sliced Implementation of a Register Module

Bit-sliced components are a natural way to implement the algorithmic pattern generator registers.

- They can be interconnected to make any register length (usually a multiple of four bits)
- The individual RALUs are powerful enough to implement any ANGEL function
- Since they are usually TTL or ECL they allow a very fast tester cycle time
- They support microprogrammable architectures which provide a highly ordered system structure.

3.6 Realization of Bit-Slice RALU and Controller

The heart of every processing section consists of an array of registers and a multifunction ALU. Design engineers could incorporate all this on a chip creating a universal building block usable in a variety of applications. However, this would require an excessively large pinout and the chip will not be really universal, because of the fixed data-path width. The solution is to design a device with a "narrow" (2 or 4 bits) data path and to allow any number of these devices to be interconnected, yielding data path widths in multiples of 2 or 4 bits. By bringing out appropriate

signals (carry in and carry out as well as both sides of the shift logic) a universal building block resulted called the Bit-Slice-RALU.

The same ideas apply to the sequencer or controller. Hardwired logic controllers decode an instruction and generate the various timing and control signals. The microprogrammed controller is constructed from a memory and a sequencer shown in Figure 3.2. Note that the architecture is very similar to the RAM-based word generator. This is not surprising because the terms timing waveform generator, controller and word generator describe the same form of digital instrumentation. Today's bit-slice sequencers have features such as looping, branching, subroutine calling and returning. These operations can be conditional or unconditional. Subroutine nesting is also a common feature.

3.7 The 2901 RALU and the 2910 Sequencer

In the design to follow, the 2901 RALU slice and the 2910 sequencer slice are used as the basic building blocks of the system. Both are members of the 2900 family, first produced by Advanced Micro-devices and now are second-sourced by a large number of semiconductor manufacturers. They are the most widely used bit-slice devices. This section illustrates their architecture and theory of operation. Also their relative advantages are discussed to justify their selection.

3.7.1 The 2901 RALU-Slice

The organization of the 2901 is shown in Figure 3.3. The circuit is a four-bit slice. Therefore, all data paths within the circuit are four bits wide. However, any number of 2901's can be interconnected to

FIGURE 3.2

Microprogrammed controller. The microprogram memory provides the hardware control signals and the information to determine the next address.

yield data paths of 4, 8, 12, 16 bits.

As Figure 3.3 shows the 2901 includes:

- A dual output port 16-word RAM
- A separate register (Q)
- A multifunction ALU
- A number of internal buses (A, F and Q-bus)
- A RAM-shifter to shift the position of bits entering RAM
- A Q-shifter to shift the position of bits entering Q register

The external connections include:

- A data input (D-bus)
- A three-state output (Y-bus)
- ALU status flags (sign, zero, carry, overflow)
- A number of control inputs (A-address, B-address, $I_0 \dots I_8$, $\overline{OE}$) and
- Several signals for interconnection to additional bit slices (CY_{in} , $\overline{G}$, $\overline{P}$, RAM_3 , RAM_0 , Q_3 , Q_0).

The two-port RAM enhances the 2901 performance allowing the simultaneous selection of any two words. The word selected by the A address appears on the A port, while the word selected by the B address appears on the B port. If a value is stored in the RAM, its location is determined by the B address. The A and B latches pass the data appearing on their inputs when the clock is high. When the clock is low the latches hold their prior inputs eliminating any race conditions when data are gated into the RAM. The clock serves also as the $RD/\overline{WR}$ signal for the RAM.

FIGURE 3.3

2901 organization.

The ALU R input is driven by a three-input multiplexer allowing the selection of the D Bus, the A register, or logic zeroes. The S input is driven by a four-input multiplexer allowing the selection of the A register, the B register, Q register or logic zeroes. The selection of the various R-S pairs as source operands is determined by the I_0 , I_1 and I_2 control inputs as illustrated in Figure 3.4.

The ALU performs three binary arithmetic and five logic functions. The function to be carried out is controlled by I_3 , I_4 , and I_5 inputs in Figure 3.5. The remaining I control inputs, I_6 , I_7 , I_8 , control the RAM shifter, the Q shifter, the Q register along with its multiplexer, and the Y-bus multiplexers. The definition of these inputs is shown in Figure 3.6. An oversimplified view of 2901 timing is shown in Figure 3.7. The 2901 timing considerations are discussed in the next chapter.

The major advantages of this chip are: first the two-port register array (RAM) includes 16 working registers ready to serve as the sources for the ALU operands or as the destination for the ALU result. Second, the ALU itself performs many arithmetic and logic functions and generates a complete status information. A third advantage is the very clear definition of the control signals ($I_0 \dots I_8$). Fourth, its excellent documentation and the large 2900 family of supporting devices have to be considered as advantages.

MICRO CODE				ALU SOURCE OPERANDS	
I ₂	I ₁	I ₀	Octal Code	R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	O	Q
L	H	H	3	O	B
H	L	L	4	O	A
H	L	H	5	O	B
H	H	L	6	D	Q
H	H	H	7	D	O

ALU Source Operand Control.

FIGURE 3.4 ALU source operand control.

MICRO CODE				ALU Function	Symbol
I ₅	I ₄	I ₃	Octal Code		
L	L	L	0	R Plus S	$R + S$
L	L	H	1	S Minus R	$S - R$
L	H	L	2	R Minus S	$R - S$
L	H	H	3	R OR S	$R \vee S$
H	L	L	4	R AND S	$R \wedge S$
H	L	H	5	R AND S	$\overline{R \wedge S}$
H	H	L	6	R EX-OR S	$R \vee S$
H	H	H	7	R EX-NOR S	$\overline{R \vee S}$

ALU Function Control.

FIGURE 3.5 ALU function control.

MICRO CODE			RAM FUNCTION		Q-REG. FUNCTION		Y OUTPUT	RAM SHIFTER		Q SHIFTER	
I ₈	I ₇	I ₆	Shift	Load	Shift	Load		RAM ₀	RAM ₃	Q ₀	Q ₃
L	L	L	X	NONE	NONE	F → Q	F	X	X	X	X
L	L	H	X	NONE	X	NONE	F	X	X	X	X
L	H	L	NONE	F → B	X	NONE	A	X	X	X	X
L	H	H	NONE	F → B	X	NONE	F	X	X	X	X
H	L	L	DOWN	F/2 → B	DOWN	Q/2 → Q	F	F ₀	IN ₃	Q ₀	IN ₃
H	L	H	DOWN	F/2 → B	X	NONE	F	F ₀	IN ₃	Q ₀	X
H	H	L	UP	2F → B	UP	2Q → Q	F	IN ₀	F ₃	IN ₀	Q ₃
H	H	H	UP	2F → B	X	NONE	F	IN ₀	F ₃	X	Q ₃

X = Don't care. Electrically, the shift pin is a TTL input internally connected to a three state output which is in the high-impedance state.

B = Register Addressed by B inputs.

Up is toward MSB, Down is toward LSB.

ALU Destination Control.

FIGURE 3.6 ALU shift and destination control.

(From AMD 2901 manual)

FIGURE 3.7

An oversimplified consideration of 2901 timing.

3.7.2 The 2910 Sequencer

Although the 2910 sequencer is considered a member of the 2900 BIT-SLICE family, it is not a bit-slice device. It is 12-bit wide and includes the entire sequencing function in itself. The 2910 can address up to 4K of memory and is equipped with a very powerful instruction set.

The structure of the 2910 is depicted in Figure 3.8. It contains a four-input multiplexer, a register/counter, a 5-word stack, a stack pointer, a microprogram counter μPC , and an instruction PLA (Programmable Logic Array).

The multiplexer is used to select either the direct input D_i , register/counter, the top value of the stack or microprogram counter as the source of the next microinstruction address.

The register/counter is used to keep track of loop interactions. Also the register/counter can be loaded with a microinstruction address that is going to be used as a branch address.

The stack is used to provide the return address linkage when microsubroutines or loops are executed. The stack pointer always points to the last word written in the stack. The stack is used in ways which are unfamiliar to conventional programmers, such as holding the starting address in a loop.

Provided that carry in (CI) is high, the microprogram counter μPC is used for sequential execution. If CI is low, the μPC is not incremented allowing the same microinstruction to be executed any number of times.

FIGURE 3.8

The 2910 architecture.

(From AMD 2910 manual)

The instruction PLA takes as inputs the 4-bit instruction $I_3 I_2 I_1 I_0$, the Condition Code ($\overline{CC}$), and Condition Code Enable ($\overline{CCEN}$) inputs and the internal zero detect for the register counter. The PLA decodes the instruction and provides the internal controls that correspond to the next address control logic as shown in Figure 3.8. The $\overline{PL}$, $\overline{MAP}$ and $\overline{VECT}$ outputs-only one of them can be active at a time-are used to select one of three possible sources to drive the D_i inputs. Table 3.1 shows all pin functions.

The four instruction bits $I_0 I_1 I_2 I_3$ provide 16 basic instructions the function of which is shown in Table 3.2. Figure 3.9 shows examples of how each instruction is executed.

The CP input is the clock whose rising edge controls all the storage devices within the 2910. When the clock rises, also depending on the value of the control inputs, an incremented value is stored in the μPC , a value is stored in the register/counter, the register/counter is decremented, and/or a value is stored in the stack. Although the 2910 is not cascadable, its universal nature and distinctive characteristics contributed to its wide use. The 2910 major advantages are:

- First, it is 12-bit wide and thus capable of addressing up to 4K microinstructions.
- Second, it has a powerful instruction set including many very useful branch operations.
- Third, the internal loop counter/register, eliminating some additional logic.
- Fourth, it is a fast TTL microsequencer.
- Fifth, it is supported by an excellent documentation.

Abbreviation	Name	Function
D_i	Direct Input Bit i	Direct input to register/counter and multiplexer. D_0 is LSB
I_i	Instruction Bit i	Selects one-of-sixteen instructions for the Am2910
$\overline{CC}$	Condition Code	Used as test criterion. Pass test is a LOW on $\overline{CC}$
$\overline{CCEN}$	Condition Code Enable	Whenever the signal is HIGH, $\overline{CC}$ is ignored and the part operates as though $\overline{CC}$ were true (LOW)
CI	Carry-In	Low order carry input to incrementer for microprogram counter
$\overline{RLD}$	Register Load	When LOW forces loading of register/counter regardless of instruction or condition
$\overline{OE}$	Output Enable	Three-state control of Y_i outputs
CP	Clock Pulse	Triggers all internal state changes at LOW-to-HIGH edge
VCC	+5 Volts	
GND	Ground	
Y_i	Microprogram Address Bit i	Address to microprogram memory. Y_0 is LSB, Y_{11} is MSB
$\overline{FULL}$	Full	Indicates that five items are on the stack
$\overline{PL}$	Pipeline Address Enable	Can select #1 source (usually Pipeline Register) as direct input source
$\overline{MAP}$	Map Address Enable	Can select #2 source (usually Mapping PROM or PLA) as direct input source
$\overline{VECT}$	Vector Address Enable	Can select #3 source (for example, Interrupt Starting Address) as direct input source

TABLE 3.1 The 2910 pin definitions

(From AMD-2910 manual)

HEX 13-10	MNEMONIC	NAME	REG/ CNTR CON- TENTS	FAIL		PASS		REG/ CNTR	ENABLE
				Y	STACK	Y	STACK		
0	JZ	JUMP ZERO	X	0	CLEAR	0	CLEAR	HOLD	PL
1	CJS	COND JSB PL	X	PC	HOLD	0	PUSH	HOLD	PL
2	JMAP	JUMP MAP	X	D	HOLD	0	HOLD	HOLD	MAP
3	CJP	COND JUMP PL	X	PC	HOLD	0	HOLD	HOLD	PL
4	PUSH	PUSH/COND LD CNTR	X	PC	PUSH	PC	PUSH	Note 1	PL
5	JSRP	COND JSB R/PL	X	R	PUSH	0	PUSH	HOLD	PL
6	CJV	COND JUMP VECTOR	X	PC	HOLD	0	HOLD	HOLD	VECT
7	JRP	COND JUMP R/PL	X	R	HOLD	0	HOLD	HOLD	PL
8	RFCT	REPEAT LOOP, CNTR $\neq$ 0	$\neq$ 0	F	HOLD	F	HOLD	DEC	PL
			-0	PC	POP	PC	POP	HOLD	PL
9	RPCT	REPEAT PL, CNTR $\neq$ 0	$\neq$ 0	D	HOLD	D	HOLD	DEC	PL
			-0	PC	HOLD	PC	HOLD	HOLD	PL
A	CRTN	COND RTN	X	PC	HOLD	F	POP	HOLD	PL
B	CAPP	COND JUMP PL & POP	X	PC	HOLD	D	POP	HOLD	PL
C	LDCT	LD CNTR & CONTINUE	X	PC	HOLD	PC	HOLD	LOAD	PL
D	LOOP	TEST END LOOP	X	F	HOLD	PC	POP	HOLD	PL
E	CONT	CONTINUE	X	PC	HOLD	PC	HOLD	HOLD	PL
			$\neq$ 0	F	HOLD	PC	POP	DEC	PL
F	TW8	THREE-WAY BRANCH	-0	D	POP	PC	POP	HOLD	PL

Note 1. If $\overline{CCEN}$ = LOW and $\overline{CC}$ = HIGH, hold; else load X = Don't Care

TABLE 3.2 The 2910 instruction set.

(From AMD 2910 manual)

FIGURE 3.9

2910 Execution examples.

(From AMD 2910 manual)

CHAPTER 4

HARDWARE IMPLEMENTATION OF ONE 8-BIT REGISTER MODULE

4.1 Introduction

This chapter describes the detailed hardware implementation of an 8-bit ANGEL Register Module. An effort has been made to avoid disabling any features offered by the bit-slice components used and to keep the hardware cost as low as possible.

The following list represents the main design goals:

1. Microprogrammed architecture
2. Very fast (200 ns maximum cycle time)
3. A large number of data operations (arithmetic and logical)
4. Single bit operations
5. Variable word-length operations especially shift and rotate operations that can be performed on any number of bits (up to 8) including double shifts and rotates
6. I/O, MASK and MODE switching as well as timing alteration "on-the-fly"
7. Microprocessor-control for memory loading and system operation supervision

4.2 Register Module Architecture

Figure 4.1 shows the Register Module block diagram. Its heart is a 2901 array which under appropriate control produces the desired data (8-bits) and sends them out through the Y-bus. For the control section

the 2910 sequencer is used. The control information required by the RALU and the sequencer and certain other devices, is stored in a RAM. This RAM is called microprogram memory because it holds microinstructions, rather than test data. The "TRI" and "MASK" memories (both 8-bits wide) can supply the pin electronics circuits with "three-state" and "mask" data patterns. Such patterns can also be supplied by the 2901 array. The data stored in the "MODE" memory (40-bits) drive the format circuitry to select the desired data mode (Normal, Return-To-Zero etc.). The information held in the "CLOCK" memory (40 bits) is used to control the binary generators which generate the "TRICKL", "DATA CLK" and "STROBE" pulses so that a number of different clock types can be applied to each pin. The address for "TRI", "MASK", "MODE" and "CLOCK" memories can be supplied by 4 different sources, as shown in the diagram.

The pipeline and status registers are used to implement a form of the so-called pipelining technique. Pipelining reduces the cycle time of the system by allowing the control section (2910, microprogram RAM) and the processing section (2901 array) to operate in parallel, so that the next microinstruction is being addressed and fetched in parallel with the control activities of the current microinstruction.

The condition-code multiplexer (CC MUX) selects and feeds the sequencer with two groups of conditions-flags. The first of these groups originates from the Register Module itself while the second group includes the status conditions-flags of the other Register Modules.

The block diagram of the system shows also an interface to a microprocessor. The basic functions of the microprocessor are system initialization and memory loading. Another function of the microprocessor is to "HALT" the system whenever a DUT failure is detected or when a specific address of the microprogram memory is reached. The decoder, shown in the upper right corner of the diagram is a part of the microprocessor interface. It allows the microprocessor to READ the memories as well as to WRITE into them. The decoder is exclusively used to load the system memories and does not interfere when the system is in the "RUN" mode.

The description of the various operations and wiring diagrams of each block are the subject of the following sections.

4.3 Processing Section

4.3.1 Combining Two 2901's

To form an 8-bit RALU two 2901's have to be cascaded as shown in Figure 4.2. The control signals for the I, A, B and CP inputs are fed in parallel to both slices. The F=0 outputs of the 2901's are connected together and to a pull-up register. This signal (Z) indicates whether the result of the combined ALUs is zero. All other conditions (Flags) are taken from the most significant slice. These are: Carry (CY_{out}) overflow (OVR), and the MSB of the ALU (S) which also represents the sign of the ALU result in two's complement notation.

For carry propagation, the carry-out (C_{n+4}) of the least significant slice is connected to the carry-in (C_n) of the most significant

FIGURE 4.2 2901 interconnections for expansion.

slice (ripple-carry). An alternative option for carry interconnection is the use of a carry-look ahead generator which generates the carry inputs considerably faster if the number of cascaded slices is three or more. (To demonstrate the carry-look ahead option, the detailed connection diagram of a 16-bit RALU, using the 2902 lookahead carry generator is shown in Appendix B). Left and right shifts between devices take place over bidirectional lines; that is the Q_3 and RAM_3 pins of the least significant slice are connected to the Q_0 and RAM_0 pins of the most significant slice respectively.

4.3.2 Shift Linkage

Figure 4.3 shows the 2901 array and the external logic to provide the necessary shift linkage. The connection shown holds for any number of cascaded slices. At the ends of the array, two dual multiplexers (RMUX and LMUX) are used to select the new shift inputs to the registers. The instruction bit I_7 is used to select whether the right multiplexer (RMUX) or the left multiplexer (LMUX) is active. The Y-multiplexer (YMUX) which is driven by the Y bus and the decoder/demultiplexer (DECODER/DEMUX) which can drive the RALU D but have their own control inputs coming from the microinstruction field (MIF). The DECODER/DEMUX can be used as an input to the 2901 for performing a bit set, bit clear, bit toggle or bit test operation.

4.3.3 Description of the Various Shift Modes

Working as a data generator the RALU array which here is assumed to be N-bits wide (N/4 slices), must be able to operate with words consisting

FIGURE 4.3 Shift linkage logic surrounding the 2901 array.

of any number M of bits where $1 \leq M \leq N$. Although in most cases this can be done by forcing to zero or ignoring the unused bits, a special treatment is required when shift and rotate operations are performed. The shift modes will be examined in detail here. Referring to the 2901 block diagram will help in understanding the various shift modes.

I) N-bit Up/Down Shift Zero Fill

RAM shift: The desired RAM register is passed without change through the ALU and the F bus to the RAM shifter, where it is shifted and then stored to the RAM register indicated by the B address.

Q shift: The value in the Q register is driven to the Q shifter through the Q bus and then, after shifting, back to the Q register.

RAM and Q registers can both be shifted simultaneously in the same cycle depending on the value of I_8 and I_7 control bits. One cycle is needed.

II) N-bit Up/Down Shift, One's Fill

III) N-bit Down Rotate

The shifted out LSB is driven through the LMUX into the most significant position. RAM and Q registers can both rotate simultaneously in the same manner. One cycle is needed.

IV) N-bit Up Rotate

up

$$I_7 S_0 S_1 = 110_2$$

RAM rotate: The RAM register, to be rotated, is driven to the RAM shifter through the ALU and Fbus. The MSB follows a different path. It passes through the ALU and F bus to the external Y bus, through the YMUX and RMUX, into the RAM_0 of the RAM shifter and takes the least significant position.

Q rotate: The shifted out MSB is driven directly through the RMUX to the least significant position. The RAM and Q registers can be rotated simultaneously in the same manner. One cycle is needed.

V) Double Length Down Rotate

down

$$I_7 S_0 S_1 = 011_2$$

The Q register LSB is driven into the RAM MSB and the RAM LSB is driven into the Q register MSB. Again one cycle is needed.

VI) Double Length Up Rotate

up

$$I_7 S_0 S_1 = 111_2$$

The RAM MSB is driven into the Q LSB (following the path ALU-FBUS-YBUS-YMUX-RMUX) and the Q register MSB is driven into the RAM LSB. One cycle is needed.

VII) M-bit Shift Down Zero Fill

This is similar to the N-bit shift down except that the M+1 bit has to be cleared when the RAM register passes through the ALU. Clearing the M+1 bit can be easily done by using the DECODER/DEMUX. This shift operation takes one cycle but only the RAM registers can be shifted this way.

VIII) M-bit Shift Down "One Fill"

Same as above but in this case the M+1 bit has to be set to one.

IX) M-bit Up Rotate

The register which is going to be rotated is sent unchanged to the X bus then the YMUX selects the Mth bit and drives it through the RMUX to the shifter I/O pin (Q_0 or RAM_0). The RAM or Q registers can be rotated this way but not both in the same cycle. One cycle is needed.

X) M-bit Down Rotate

This operation is slower and is the only one that needs two cycles to be performed. In the first cycle the $M+1$ bit is cleared (set to zero) and the whole N -bit word is shifted down (right). The shifted out LSB is latched in the D Flip-Flop. The DECODER/DEMUX outputs, which in this case drive the RALU D bus, are all zeroes except the M th bit which takes the value latched in the Flip-Flop (shifted out LSB). Finally in the second cycle an OR operation is performed between the already shifted down register and the value appearing on the D bus (DECODER/DEMUX output). Only RAM registers can be shifted this way.

Table 4.1 illustrates the various shift modes.

4.3.4 Eight-bit Processing Section

The complete connection diagram of the 8-bit processing section is shown in Appendix A, Figure A.1. U_{71} and U_{82} are the two 2901's. U_{70} and U_{80} are the L-MUX and R-MUX respectively. U_{82} is the Y-MUX. The 8-bit DECODER/DEMUX is the 2921 (U_{72}) whose output is applied to the D-bus to implement all the Bit Functions when BF (Bit Function) line is high. These include bit set, bit clear, bit toggle and bit test. When BF is low the 2921 is used as a demultiplexer. The 2921 can be three-stated by the $\overline{DBS}$ (D-Bus Source) signal allowing the μ Instruction field to drive the D bus.

CODE			SOURCE OF NEW DATA				SHIFT	TYPE
I_7	S_1	S_0	RAM_0	RAM_N	Q_0	Q_N		
1	0	0	0	F_{N-1}	0	Q_{N-1}	UP	ZERO
1	0	1	1	F_{N-1}	1	Q_{N-1}		ONE
1	1	0	F_X	(F_{N-1})	Q_N	Q_{N-1}		ROTATE
1	1	1	Q_N	F_{N-1}	F_X	Q_{N-1}		DOUBLE LENGTH ROTATE
0	0	0	F_1	0	Q_1	0	DOWN	ZERO
0	0	1	F_1	1	Q_1	1		ONE
0	1	0	F_1	F_0	Q_1	Q_0		ROTATE
0	1	1	F_1	Q_0	Q_1	F_0		DOUBLE LENGTH ROTATE

TABLE 4.1

Illustration of the various shift modes.

Note that $N \neq X \neq 0$.

4.4 Control Section

4.4.1 Microprogram Control

The detailed connection diagram of the control section is shown in Figures A.2, A.3 and A.4 (Appendix A). In this design a 1K by 76 microprogram memory is required, which is constructed using 2114 (1Kx4) RAM's. Nineteen chips (U_{10-17} , U_{20-27} and U_{34-36}) are placed in parallel with common address lines. The required 10-bit address ($2^{10}=1024$) is supplied by the $Y_0...Y_9$ outputs of the 2910 sequencer, shown as U_{33} . Y_{10} and Y_{11} , which are not used, are left floating, while the corresponding D_{10} and D_{11} 2910-inputs are tied to ground since the only source for the D-inputs is the pipeline register, the output signals $\overline{PL}$, $\overline{VECT}$ and $\overline{MAP}$ are not used (floating). The $\overline{FULL}$ output is also left floating. The CI and $\overline{RLD}$ inputs are connected to V_{cc} through a pull-up resistor.

An 8-input condition code multiplexer function is provided by using a 2922 (U_{32}). This device allows one of eight inputs to be tested. The status and condition register consists of a 74LS 377 8-bit edge triggered D-type register shown as U_{31} . One of the condition inputs is grounded serving as permanent "true" or "false" condition depending on the 2922 polarity control signal (POLM).

The pipeline register consists of U_{40} , U_{41} , U_{42} , U_{43} , U_{44} , U_{45} , U_{46} , U_{52} , U_{53} , U_{54} , U_{55} and U_{56} . All of these are positive edge triggered D-type registers. Two 74LS374 are used for U_{41} and U_{42} which provide the 2910 instruction ($SI_0...SI_3$) and 2910 operand ($SO_0...SD_9$). They have outputs that can be tristated and allow the external microprocessor-control of the 2910 through U_{50} and U_{51} . Note that the microinstruction bits

9 through 16 are shared by more than one device. More specifically, they are used as the eight least significant bits of the 2910 operand ($SD_7 \dots SD_0$), as the eight-bit RALU operand ($D_7 \dots D_0$) and as one of four possible address sources for "TRI", "MASK", "MODE" and "CLOCK" memories ($MD_7 \dots MD_0$). A 74LS175 was selected for U_{45} because both a true and complement output were desired for the I_7 and DBS microcode bits. U_{54} is a 74LS273 latch. This device has a master Reset input which is active (low) during memory loading. Microinstruction bits 21, 22, 23 and 24 are not pipelined because their values are latched when the clock rises into the internal latches of the 2922. Microinstruction bits 50, 51, 52, 53, 54, 55 and 56 are not pipelined because their control activities are restricted to a short time interval during clock rise when they are stable.

4.4.2 DC and AC Loading of the 2910

In addition to the nineteen 2114's the 2910 Y-outputs also drive one 74LS374 (U_{61}) and each of them drives one input of the 74S86 gates (U_{47} , U_{57} and U_{66}). Therefore the DC and AC loading of the 2910 has to be investigated. Each address line carries nineteen times the load presented by one 2114, the load presented by one 74LS374 input and one 74S86 input. From Table 4.2 it can be seen that the I_{OL} given for the 2910 is much greater than the total I_{IL} of the devices that are connected to its outputs. Therefore no buffer drivers are required.

Unlike the DC loading the AC loading is significant with 110pf on each 2910 address line.

Device	DC loading mA (I_{IL})	AC loading pf
19x2114	19x.01=.19mA	19x5=95 pf
74LS374	.4 mA	5 pf
74S86	2 mA	5 pf
74S245	high impedance	5 pf
Total	2.59 mA	110 pf
2910	$I_{OL} = 12 \text{ mA}$	$C_L = 50 \text{ pf}$

Table 4.2 Worst-Case DC and AC Characteristics of the 2910 and the Devices Connected to its Y Outputs.

The 2910 performance is degraded in accordance with the increased capacitive load by 0.1 ns per pf over 50 pf 110 pf -- 50 pf = 60 pf.

$$0.1 \text{ ns/pf} \times 60 \text{ pf} = 6 \text{ ns}$$

This delay is added to the 2910 propagation time. However, there is no way to avoid this delay. Any buffer drivers placed at the 2910 Y outputs, would remove the 6 ns delay but they would add a propagation delay of their own of a greater or at least equivalent magnitude.

4.4.3 Clock Generator-External Control

The 2925 (U_{61}) is used as the system clock generator. It is a single-chip general purpose clock generator and has many useful features. The 2925 generates four different simultaneous clock output waveforms C_1 , C_2 , C_3 and C_4 . Also a buffered oscillator output F_0 is provided for external timing. From these only one clock phase C_3 is used which is delivered to the clocked devices through the 74LS244 clock driver (U_{62}) as shown in Figure A.4. One of the eight clock cycle lengths can be selected by controlling the L_1 , L_2 and L_3 cycle length control inputs. These inputs are connected to switches 1, 2 and 3 respectively of U_{60A} (DIP switch).

Other control functions are: Run/Halt, Single-Step and Ready/Wait controls.

The clock generator receives a wait request whenever the comparator output ($\overline{SA}=\mu PA$) or the $\overline{PFHALT}$ signals are activated (low). Figure A.4.

The comparator consists of U_{47} , U_{57} , U_{66} , U_{65} , U_{64} and U_{63} . Its output becomes active when the 2910 address (SA_{0-9}) and the address

supplied by the microprocessor (μPA_{0-9}) are equal, provided that the switch 8 of U_{60A} is open. This means that the microprocessor can "HALT" the system at any microprogram address (e.g. end of program).

The microprocessor can also cause the machine to single-step through the microprogram by activating the μPSS signal while keeping the $\mu PHALT$ signal low. The rising edge of μPSS produces a narrow pulse which is applied through a monostable to the $\overline{READY}$ input of the 2925. This causes the 2925 to generate one clock pulse. Note that when $\overline{READY}$ is low while $\overline{WAITREQ}$ is also low the 2925 is free running. The minimum time duration of the $\overline{READY}$ pulse T_D , is 25 ns (from the 2910 manual) and if only one clock pulse is to be generated the $\overline{READY}$ signal must be shorter than two oscillator cycles. More specifically, the crystal used oscillates at 12.5 MHz ($1/f=80ns$) thus the monostable has to be adjusted to produce a $\overline{READY}$ pulse of duration $25ns < T_D < 160ns$ $T_D=100 ns$ was the selected value.

4.4.4 2910 Initialization

Initializing the 2910 at power-up is accomplished under microprocessor control. As shown in the wiring diagram Figure A.3, the microprocessor can supply the 2910 instruction and operand through U_{50} and U_{51} by activating the JAM signal (Figure A.3). For 2910 initialization the following steps are needed: 1) the 2925 is caused to "WAIT" by activating the $\overline{PFHALT}$ signal and JAM goes high; 2) using the μPSS (single step) signal the microprocessor causes the 2925 to generate two clock cycles. The first clock cycle loads the desired 2910 instruction and operand into the U_{50} and U_{51} . The second clock cycle causes the 2910 instruction

execution; 3) after disabling the $\overline{\text{PFHALT}}$ and JAM signals ($\overline{\text{PFHALT}}$ =high, JAM=low) activating the μPSS signal once more puts the system into the RUN mode.

4.5 Pattern Memories

4.5.1 Pattern Memories

As mentioned earlier, patterns for "TRI", "MASK" and "MODE" data as well as "CLOCK" data for timing control are stored in four pattern memories (Figure 4.1). Under microprogram control, the address for each one of the pattern memories can be selected among the four following sources:

- 1) A address. This address is the 2910 latched output. It may be selected when a subroutine is called which requires its own "TRI", "MASK", "MODE" patterns and timing.
- 2) B address. This address comes from the RALU output. It allows users to generate addresses with the flexibility of an RALU.
- 3) C address. It comes from a microinstruction field which is exclusively used to supply this address.
- 4) D address. This address is also coming from a microinstruction field which is also used to supply the operand for the 2910 and RALU.

The operation of the pattern memories is very simple. At the start of a clock cycle the selected address propagates through the transparent address latches and is presented to the memory address bus. At the end of the cycle (start of the next one) the memory output data are latched into the output latches. Both, address and data latches are enabled or disabled from receiving new values by the microcode. This means that an

address value may be latched into the address latch at the Nth clock cycle, while the corresponding addressed data may be enabled to appear on the output of the (N+K)th clock cycle.

4.5.2 Loading of Memories

Memory loading is done through the microprocessor which loads the desired data byte by byte when the system is halted. When the system is in RUN mode, all memories are enabled (chip select lines active). When the system is in memory load mode, a microprocessor-memory interface decoder board is used. Its block diagram is shown in Figure 4.4.

The microprocessor supplies the decoder board with 5 bits of data, 10 bits of address, 5 control bits which may be considered as five additional address bits, a $\overline{RD/\overline{WR}}$ and a $\overline{MEMORY\ LOAD\ ENABLE}$ signal.

The 1 of 19 decoder, when enabled by the $\overline{MEMORY\ LOAD\ ENABLE}$ signal, activates one octal transceiver at a time as well as the corresponding chip select ($\overline{CS}_x$) signal. The direction of data flow is directly controlled by the microprocessor $\overline{RD/\overline{WR}}$ signal. The $\overline{MEMORY\ LOAD\ ENABLE}$ signal is also sent to the register module and pattern memory boards to disable the other address sources and to ensure that the pattern memory address latches are in the transparent mode.

The detailed wiring diagram of the DECODER BOARD is shown in Figure A.6 and A.7. The board layout is shown in Figure A.8.

4.6 Timing Analysis

The minimum clock period that can be used is determined by the component delays along the longest or critical data path. The pipelining

FIGURE 4.4 Block diagram of the decoder that provides the microprocessor-Register Module memories interface. Through this decoder the microprocessor can read from or write into the Register Module memories.

technique used, employs two registers: 1) the pipeline register which holds the current microinstruction and 2) the status register which holds the status conditions generated by the last microinstruction. The pipeline and status registers are triggered by the same clock that triggers the control and processing section. Pipelining allows the separate timing examination of the control and processing section.

All delays have been calculated using worst case times (maximum propagation times, minimum set-up times) over commercial operating range and $\pm 5\%$ power supply voltage variations.

In each case, delays are computed from the LOW-to-HIGH transition of a clock through an entire microcycle to the next LOW-to-HIGH transition of a clock.

4.6.1 Control Section Timing

To determine the longest data path through the control section the delays along four different signal paths have been calculated. These data paths are shown in Figures 4.5, 4.6, 4.7 and 4.8. The AC specifications of the devices are found in Appendix C.

PATH 1		
DEVICE	SIGNAL PATH	DELAY ns
Status register 74LS377	CP-to-Q	25
Cond. Cod MUX 2922	D-to-Y	22
Sequencer 2910	$\overline{CC}$ -to-Y	43+6
Memory 2114	Addr.-to-Out.	75
Cond. Code MUX 2922	I set-up	11

SUM 182 ns

FIGURE 4.5

Control section DATA PATH 1.

PATH 2

DEVICE	SIGNAL PATH	DELAY ns
Cond. Code MUX 2922	CP-to-Y	40
Sequencer 2910	$\overline{CC}$ -to-Y	43+6
Memory 2114	Addr.-to-out.	75
Cond. Code MUX 2922	I set-up	11
SUM		175 ns

PATH 3

DEVICE	SIGNAL PATH	DELAY ns
Pipeline Reg. 74LS374	CP-to-0	25
Sequencer 2910	I-to-Y	70+6
Memory 2114	Addr.-to-Out.	75
Cond. Code MUX 2922	I set-up	11
SUM		187 ns

FIGURE 4.6

Control section DATA PATH 2.

FIGURE 4.7

Control section DATA PATH 3.

PATH 4

DEVICE	SIGNAL PATH	DELAY ns
Sequencer 2910	CP-to-Y	125+6
Memory 2114	Addr.-to-Out.	75
Cond. Code MUX 2922	I set-up	11
SUM		217 ns

The longest data path is PATH 4 which is also the system critical path as it will be shown after examining the timing of the processing section. Calculations for PATH 4 show a relatively long propagation delay of 125 ns through the 2910, namely from CP to Y. This delay holds only when the current 2910 instruction $I_3I_2I_1I_0$ is one of 8, 9 or 15 and the previous one was instruction 4 or 12. The explanation is simple by recalling that instructions 4 and 12 can load a new value into the 2910 internal register/counter while instructions 8, 9 and 15 are conditional on the counter and the register/counter loading is controlled by the rising clock edge. Therefore, there is a long signal path within the 2910 chip running from CP through the register/counter, the zero detector, the instruction PLA, the multiplexer and the output drivers to the Y output. A solution to reduce the critical path delay might be to disable instructions 4 and 12 of the 2910, but these instructions are extremely useful. Since a faster version of the 2910 is not yet available, the only way to improve the system speed is to use faster memories and/or faster devices to replace the 2922 condition code multiplexer.

PATH 4

FIGURE 4.8

Control section DATA PATH 4

4.6.2 Processing Section Timing

Here also a number of long data paths are examined. The propagation and set-up times of the devices, through which these data paths run, are found in Appendix C. The data paths are shown in Figures 4.9, 4.10, 4.11, 4.12, and 4.13.

PATH 1

DEVICE	SIGNAL PATH	DELAY ns
Pipel. Reg. 74LS377	CP-to-Q	25
RALU 2901	A, B-to- C_{N+4}	59
RALU 2901	C_N -to-Y	30
Outp. Reg. 74LS377	D set-up	10
SUM		124 ns

PATH 2

DEVICE	SIGNAL PATH	DELAY ns
Pipel. Reg. 74LS175	CP-to-Q	25
Decoder/Demux 2921	OE-to-X	38
RALU 2901	D-to- C_{N+4}	40
RALU 2901	C_N -to-Y	30
Outp. Reg. 74LS371	D set-up	10
SUM		142 ns

FIGURE 4.9 Processing section DATA PATH 1.

PATH 2

FIGURE 4.10 Processing section DATA PATH 2.

PATH 3

DEVICE	SIGNAL PATH	DELAY ns
Pipe1. Reg. 74LS175	CP-to-Q	25
Decoder/DeMUX 2921	OE-to-Y	38
RALU 2901	D-to- C_{N+4}	40
RALU 2901	C_N -to-F=0	37
Status Reg. 74LS377	D set-up	10
SUM		150 ns

PATH 4

DEVICE	SIGNAL PATH	DELAY ns
Pipe1. Reg. 74LS175	CP-to-Q	25
Decoder/Demux 2921	OE-to-Y	38
RALU 2901	D-to- C_{N+4}	40
RALU 2901	C_N -to-Y	30
Y MUX 74S151	I-to-Z	12
R MUX 74S253	I-to-Y	9
RALU 2901	RAM ₀ , Q ₀	16
SUM		169 ns

PATH 3

FIGURE 4.11 Processing section DATA PATH 3.

PATH 4

FIGURE 4.12 Processing section DATA PATH 4.

PATH 5

DEVICE	SIGNAL PATH	DELAY ns
Pipel. Reg. 74LS174	CP-to-Q	25
OR gate 74LS32	Inp.-to-Out.	15
Decoder/Demux	E-to-Y	38
RALU 2901	D-to- C_{N+4}	40
RALU 2901	C_N -to-Y	30
Y MUX 74S151	I-to-Z	12
R MUX 74S253	I-to-Y	9
RALU 2901	RAM ₀ , Q ₀ setup	16

SUM 185 ns

PATH 5

FIGURE 4.13 Processing section DATA PATH 5.

4.6.3 Timing of Pattern Memories

The system pattern memories were not implemented in hardware as part of this thesis. However, timing requirements are given below. Figure 4.14 shows a signal path running through the pattern memory. The delays along each device of this path and the total one are:

Pipel. Reg. 74LS377	D-to-Q	25 ns
MUX 74S253	I-to-Y	9 ns
Transp. Latch 74LS375	D-to-Q	27 ns
Memory	A-to-D	max taccess
Out. Reg. 74LS377	D set-up	10 ns

$$\text{Total Delay} = 71 + t_{\text{access}} \text{ ns}$$

To avoid deterioration of the system speed this total delay has to be less than the total delay along the critical path of the system

$$71 + t_{\text{access}} \leq 211 \text{ ns}$$

$$\text{or } t_{\text{access}} \leq 140 \text{ ns}$$

Therefore, any memories with maximum access time less than 140 ns may be used for the pattern memory hardware implementation.

4.7 Programming Example

The programming of the system is done at the micro-level. The microinstruction format is found in Appendix A (pages A.14, A.15), where the functions and control activities of each individual microinstruction bit or groups of bits are provided.

FIGURE 4.14

Pattern memory data path.

An example of an hypothetical test pattern is shown in Figure 4.15a. The corresponding ANGEL program is shown in Figure 4.15b. The actual microprogram used to generate the data sequence of Figure 4.15a which is generated by the 2901 array is illustrated in Figure 4.16. Note that only the microinstruction bits that are used, are shown. The rest of them are used to control the pattern memory section which is not included in the prototype hardware.

The operation of each microinstruction is briefly described below.

Microinstruction 00

The 2910 performs a "push/conditional load counter" instruction. The next address (01 hex) is pushed onto the stack and since the CC MUX is controlled to select a permanent true condition the counter is loaded with the value supplied by the microinstruction. This way a loop is initialized. The RALU passes the value supplied by the DECODER/DEMUX to the output and also stores it in its R_0 register.

Microinstruction 01

The 2910 executes a "repeat loop, counter $\neq 0$ " instruction. The counter is decremented and the next address is supplied by the top of the stack (=01 hex). Thus, the same microinstruction is executed until the counter reaches a zero value. Each time this microinstruction is executed the RALU performs the operation: $R_0 = R_0 + 1$.

Microinstruction 02

The 2910 performs a "push/conditional load counter" exactly the same way as in microinstruction 00. The RALU performs a BIT CLEAR function. This is done by performing a " R_0 AND D_{INPUT} " operation. Of course, the DECODER/DEMUX is controlled to drive the RALU D-inputs with

ANGEL PROGRAM

program: EXAMPLE

declarations:

.....

D₀ is pin X₀

D₁ is pin X₁

D₂ is pin X₁

D₃ is pin X₃

D₄ is pin X₄

D₅ is pin X₅

.....

register DATA consists of D₄D₃D₂D₁D₀

.....

end declarations

set DATA=01_{hex}

apply DATA

do t=1, 14

increment DATA

apply DATA

end do

clear D₃ of DATA register

apply DATA

do t=16, 20

rotate DATA one position up

apply DATA

end do

set data to 02_{hex}

apply DATA

do t=22, 26

rotate DATA one position down

apply DATA

end do

do t=27, 30

shift DATA up zero fill

apply DATA

end do

⋮

"DATA"						
t	D ₄	D ₃	D ₂	D ₁	D ₀	
0	0	0	0	0	1	] SET
1	0	0	0	1	0	
2	0	0	0	1	1	
.....						] INCREMENT
13	0	1	1	1	0	
14	0	1	1	1	1	
15	0	0	1	1	1	] BIT CLEAR
16	0	1	1	1	0	
17	1	1	1	0	0	
18	1	1	0	0	1	] ROTATE ONE POSITION UP
19	1	0	0	1	1	
20	0	0	1	1	1	
21	0	0	0	1	0	] SET
22*	0	0	0	0	1 *	
23*	1	0	0	0	0 *	
24*	0	1	0	0	0 *	] ROTATE ONE POSITION DOWN
25*	0	0	1	0	0 *	
26*	0	0	0	1	0 *	
27	0	0	1	0	0	] SHIFT UP ZERO FILL
28	0	1	0	0	0	
29	1	0	0	0	0	
30	0	0	0	0	0	

FIGURE 4.15.a

Hypothetical test sequence.
The asterix indicates a
double clock cycle.

FIGURE 4.15.b

Corresponding ANGEL program.

the proper value. The result appears at the output and before being stored in R_0 it is shifted up.

Microinstruction 03

A one microinstruction loop is preformed as in microinstruction 01. The RALU performs a 5-bit up rotate.

Microinstruction 04

A loop is initialized as in microinstructions 00 and 02. The decoder supplies the RALU D-inputs with a new value (=02 hex) which is driven to the output and after a down shift is stored in R_0 .

Microinstructions 05, 06

The combination of these two microinstructions causes the five least significant bits of the R_0 register to rotate one position down as it has been described in section 4.3.3. These instructions are executed a number of times as required.

Microinstruction 07

The RALU performs a shift up with zero fill. The 2910 performs a conditional jump to the address supplied by the microinstruction or to the address whose value is contained in the register/counter depending on the condition. The microinstruction branch field contains the value 07 hex and the 2910 register/counter contains a zero value. Therefore, when the condition fails the same microinstruction (07 hex) is repeated. When the condition is satisfied the control is transferred to microinstruction 00.

One point of interest in this example is that only the R_0 register of the RALU has been used as the source of the ALU operand or as the destination for the results. By combining any of the 17 available

registers ($R_{0-15} + Q$), greater flexibility may be obtained.

Also in pipelined architectures, one has to pay attention to conditional operations because the currently tested condition is not the condition generated by the current microinstruction but the condition produced from the previous cycle. If the conditions of the previous cycle cannot be used, one cycle will be lost. Furthermore, the programmer must take advantage of the large control information contained in each microinstruction by performing as many operations as possible in each one instruction. This results in a minimum number of micro-cycles.

CHAPTER 5

CONCLUDING MATERIAL5.1 Summary

The concept of using an algorithmic test pattern generator to produce the test data in real time has been examined. Such a generator to provide the digital signals for eight DUT pins has been prototyped and tested. For the hardware implementation of ANGEL features an architecture with high pin-out capability has been proposed. This architecture employs five similar pattern generators that operate in parallel. Each generator executes its own set of instructions (micro-code) that express the real time step-by-step procedure to generate the DUT relevant test vectors.

The evolution of other methods which use a RAM to store the stimuli/expected response data before applying them to the DUT is also reported.

Microprogrammable architecture supported by bit-slice logic was a natural way of the design hardware implementation. The generator features and operation are explained in detail. The wiring diagrams are found in the appendixes. The system timing has been analyzed by examining a fair number of data paths. Finally, the operation of the generator has been demonstrated.

5.2 Conclusions and Recommendations

The implemented pattern generator offers many desirable features such as a large number of arithmetic and logical functions, single bit functions and variable length shift and rotate operations. The idea of

22
OF / DE

replacing the pattern RAM by a microcode-controlled RALU has been proven feasible.

The heart of the generator is the processing section which consists of an array of 2901's and shift linkage logic. The processing section generates an 8-bit test pattern and for its operation requires 26 microcode bits (excluding the 8-bit 2901-operand which is shared with the 2910 operand). However, a 26-bit microinstruction can produce a large number of 8-bit test data. An example is the "increment" microinstruction which can produce $2^8=256$ different 8-bit test data and repeat this sequence any number of times. Therefore thousands of test patterns can be generated by a single microinstruction.

Furthermore a microcoded algorithm, consisting of a rather small number of microinstructions, can produce a long test pattern sequence. Since an algorithmically generated sequence depends on the initial values stored in the RALU registers, by changing these initial values the same algorithm can produce different pattern sequences. In other words, a microsubroutine can be executed several times during a test sequence with different results.

The proposed register-organized generator promises an effective operation. Since usually only a few changes occur from pattern to pattern, a register may idle (perform a one-microinstruction loop) for a number of cycles until it is flagged by another register to continue execution. The result is the efficient use of the microprogram memories.

The RALU output-bus is routed to three destinations: the DATA latch, the TRI latch and the MODE latch. Since the RALU can supply only one word per machine cycle the TRI and MODE latches are also fed from the corresponding pattern memories. A RALU with more than one output bus could serve this purpose better. The drawback of such a solution is that RALUs with two output buses require many control inputs (microcode bits). However such an alternative should not be excluded.

The microprogrammed architecture used, resulted in a structured and easy to debug design. Microinstruction pipelining allows the system to operate at satisfactory speed but requires close attention when conditional branching operations are performed. One must remember that a conditional branching operation tests conditions generated in the previous machine cycle, not the current cycle.

Although higher speed is desirable, it is hard to be achieved. The delays along the longest data path of the processing section can be improved by using a faster 2901 version (2901C) and other faster devices along this path. To improve the longest path of the control section (system critical path) faster microcode memories must be used. However, these changes will not result in any significant speed increase. 5MHz may be considered as a recommended clock frequency for this design. A two-level pipeline technique can provide the maximum possible speed but it also causes a further complication of the microprogram because the fetching of an instruction occurs two cycles ahead of its execution and therefore is not recommended.

Modification of the microinstruction format should be considered. For instance, microcode bits 2, 3 and 4 (C, B, A) are used to control both, the DECODER/DEMUX (U_{72}) and the YMUX (U_{82}). Since in some cases these devices are active simultaneously in the same cycle and may require different control inputs, the use of two separated microcode fields is recommended. Although three additional microcode bits are needed the processing section flexibility is enhanced.

Another consideration is to separate the microinstruction field (7, 8, ... 16) that provides an operand to the RALU and the sequencer. Using a common field, only one operand can be supplied in one cycle. By separating this microcode field a higher degree of flexibility is achieved but it should be weighed against the eight additional control bits that are needed.

Increasing the width of the microinstruction is an easy way to provide many parallel operations in the expense of an additional microcode memory. However, a wide microinstruction is possible to specify a number of redundant operations. Microinstruction encoding reduces the possibility of specifying operations that are not needed. Employing some degree of encoding a number of microinstruction bits is saved and programming is simplified. However, microinstruction encoding has drawbacks. It requires extra decoding logic which introduces its own delay. After carefully determining any redundant operations a degree of encoding is recommended but should not create a new critical data path.

Finally the proposed generator requires a large amount of hardware. Although the hardware needed is justified by the ability of the system to satisfy current VLSI demands like "on-the-fly" timing variation, format selection as well as MASK and I/O switching, further work may reduce it and keep the system performance in an acceptable level.

REFERENCES

1. Glenford, J. Myers, "Digital system design with LSI bit-slice logic", John Wiley and Sons Inc., 1980.
2. John Mick and Jim Brick, "Bit-slice microprocessor design", McGraw-Hill Book Co., 1981.
3. James T. Healy, "Automatic testing and evaluation of digital integrated circuits", Reston Publishing Co., Inc., 1981.
4. Donnamie E. White, "Bit-slice design: controllers and ALUs", Garland Publishing Inc., 1981.
5. Brad Snoultten and John Peacock, MITEL Corporation, "ANGEL-Algorithmic pattern generation system", IEEE Test Conference, 1981.
6. Daniel P. Siewiorek and Larry Kwok-Woom Lai, "Testing of digital systems", Proceedings of the IEEE, Vol.69, No.10, pp.1321-1333, October 1981.
7. Advanced Micro Devices, "Bipolar microprocessor logic and interface data book", Advanced Micro Devices Inc., 1981.
8. Fairchild, "TTL Data book", Fairchild Corporation, 1978.
9. Intel, "Component data catalog", Intel Corporation, 1980.
10. Peter Hansen, "Functional and in-circuit testing", Electronics, Vol.54, No.8, pp.189-195, April 1981.
11. Stanley P. Kubota, "Digital word and data generation", Electronics test, pp.30-42, May 1981.

APPENDIX A

WIRING DIAGRAMS

FIGURE A.1
Processing section
wiring diagram.

Processing Section Control Signals (Inputs)

The following signals listed below are used to control the processing section and are supplied by the microinstruction.

$I_8 \dots I_0$	2901 instruction
$A_3 A_2 A_1 A_0$	2901 A address
$B_3 B_2 B_1 B_0$	2901 B address
CY_{in}	2901 carry input
$D_7 \dots D_0$	2901 D-bus
$S_1 S_0$	Along with I_7 these signals control the U_{70} and U_{80} shift-multiplexers
CBA	control of 2921 (U_{72}) and 74S151 (U_{82})
POL	2921 (U_{72}) polarity control
BF	(Bit Function) when high the 2921 operates as decoder. When low the 2921 operates as demultiplexer
DBS	(D-Bus Source) determines whether the microinstruction or the 2921 output drive the D-bus
DAT. EN	Enables the data latch (U_{74}) to receive a new value from the 2901 Y-bus
TRI.S	(Tristate Source) determines the data source for the tristate-data-latch
MSK.S	(Mask Source) determines the data source for the mask-data-latch
ADDREN	(Address Enable) enables the address latch (U_{84}) to receive a new value from the 2901 Y-bus

Processing Section Output Signals

The processing section outputs are:

CY _{out}	(Carry Out)	
OVR	(Overflow)	Status-conditions (FLAGS) generated
S	(Sign)	by the RALU
Z	(Zero)	
DAT ₇ ... DAT ₀	(DATA) Stimulus/expected response used to drive the pin electronics circuitry.	
TT ₇ ... TT ₀	These are inputs to-Tristate data latch	
TM ₇ ... TM ₀	These are inputs to-Mask data latch	
MB ₇ ... MB ₀	One of the four possible sources that can be the address to "TRISTATE", "MASK", "MODE" and "CLOCK" memories	

FIGURE A.2
Control section
wiring diagram.

FIGURE A.3 Control section wiring diagram.
(Pipeline register)

**SINGLE STEP
PULDDOWN
SWITCH**

FIGURE A.5 Board layout.

CONNECTOR A			
1	1	2	2
3	3	4	4
5	5	6	6
7	7	8	8
9	9	10	10
11	11	12	12
13	13	14	14
15	15	16	16
17	17	18	18
19	19	20	20
21	21	22	22
23	23	24	24
25	25	26	26
27	27	28	28
29	29	30	30
31	31	32	32
33	33	34	34
35	35	36	36
37	37	38	38
39	39	40	40
41	41	42	42
43	43	44	44
45	45	46	46
47	47	48	48
49	49	50	50
51	51	52	52
53	53	54	54
55	55	56	56
57	57	58	58
59	59	60	60
61	61	62	62
63	63	64	64
65	65	66	66
67	67	68	68
69	69	70	70
71	71	72	72
73	73	74	74
75	75	76	76
77	CS ₀	78	CS ₁
79	CS ₂	80	CS ₃
81	CS ₄	82	CS ₅
83	CS ₆	84	CS ₇
85	CS ₈	86	CS ₉
87	MSKEN	88	SA ₀
89	SA ₁	90	SA ₂
91	SA ₃	92	SA ₄
93	SA ₅	94	SA ₆
95	SA ₇	96	SA ₈
97	SA ₉	98	LOADEN
99		100	GND

CONNECTOR B			
1	READY	2	WAITREQ
3	JAM	4	I ₀
5	I ₁	6	I ₂
7	I ₃	8	I ₄
9	O ₁	10	O ₂
11	O ₃	12	O ₄
13	D ₅	14	D ₆
15	D ₇	16	D ₈
17	D ₉	18	A ₁₁
19	A ₁	20	A ₂
21	A ₃	22	A ₄
23	A ₅	24	A ₆
25	A ₇	26	A ₈
27	A ₉	28	M ₀
29	M ₁	30	M ₂
31	M ₃	32	M ₄
33	M ₅	34	M ₆
35	M ₇	36	ALE1
37	ALE2	38	ALE3
39	ALE4	40	TRIS
41	MSWS	42	TRIN
43	MSKEN	44	MODEN
45	CLKEN	46	
47		48	
49		50	GND

TABLE A.3, A.4 Edge connections.

FIGURE A.6
Decoder wiring diagram.

FIGURE A.7
Decoder wiring diagram.
(continued)

FIGURE A.8 Decoder board layout.

CONNECTOR A			
1	1	2	2
3	3	4	4
5	5	6	6
7	7	8	8
9	9	10	10
11	11	12	12
13	13	14	14
15	15	16	16
17	17	18	18
19	19	20	20
21	21	22	22
23	23	24	24
25	25	26	26
27	27	28	28
29	29	30	30
31	31	32	32
33	33	34	34
35	35	36	36
37	37	38	38
39	39	40	40
41	41	42	42
43	43	44	44
45	45	46	46
47	47	48	48
49	49	50	50
51	51	52	52
53	53	54	54
55	55	56	56
57	57	58	58
59	59	60	60
61	61	62	62
63	63	64	64
65	65	66	66
67	67	68	68
69	69	70	70
71	71	72	72
73	73	74	74
75	75	76	76
77	CS ₀	78	CS ₁
79	CS ₂	80	CS ₃
81	CS ₄	82	CS ₅
83	CS ₆	84	CS ₇
85	CS ₈	86	CS ₉
87	WR _{EN}	88	SA ₀
89	SA ₁	90	SA ₂
91	SA ₃	92	SA ₄
93	SA ₅	94	SA ₆
95	SA ₇	96	SA ₈
97	SA ₉	98	LOAD _{EN}
99		100	GND

CONNECTOR B			
1	AD ₀	2	AD ₁
3	AD ₂	4	AD ₃
5	AD ₄	6	AD ₅
7	AD ₆	8	AD ₇
9	AD ₈	10	AD ₉
11	MS ₀	12	MS ₁
13	MS ₂	14	MS ₃
15	MS ₄	16	DAT ₀
17	DAT ₁	18	DAT ₂
19	DAT ₃	20	DAT ₄
21	DAT ₅	22	DAT ₆
23	DAT ₇	24	RD/WR
25		26	MEMLOAD _{EN}
27		28	
29		30	
31		32	
33		34	
35		36	
37		38	
39		40	
41		42	
43		44	
45		46	
47		48	
49		50	GND

TABLE A.1, A.2 Edge connections.

Microinstruction Format

As mentioned earlier each microinstruction is 76 bits in length.

The definition of these bits is listed below:

1	SYNC	This is a micro-software controlled flag and is driven to the other Register Modules for synchronization purposes.
2	C	These bits control the U ₇₂ and U ₈₂ of the processing section.
3	B	
4	A	
5	POL	Controls the polarity of U ₇₂ outputs.
6	BF	(Bit Function) Determines if the U ₇₂ is used as decoder or demultiplexer.
7	SD ₉ - -	These bits form a shared microinstruction field: 1) they provide the branch address for the 2910 (sequencer data); 2) they feed the 2901 array D-inputs; 3) they provide an address source for the pattern memories.
8	SD ₈ - -	
9	SD ₇ - D ₇ - MD ₇	
10	SD ₆ - D ₆ - MD ₆	
11	SD ₅ - D ₅ - MD ₅	
12	SD ₄ - D ₄ - MD ₄	
13	SD ₃ - D ₃ - MD ₃	
14	SD ₂ - D ₂ - MD ₂	
15	SD ₁ - D ₁ - MD ₁	
16	SD ₀ - D ₀ - MD ₀	
17	SI ₃	These bits are connected to the I ₃₋₀ inputs of the 2910 to control the sequencing of the microprogram.
18	SI ₂	
19	SI ₁	
20	SI ₀	
21	CM	These bits are used to control the 2922 which feeds the 2910 with the selected condition.
22	BM	
23	AM	
24	POLM	

25	D8S	This bit enables either the 2921 (U ₇₂) or the 74S151 (U ₈₂) onto the 2901 array D-bus.
26	I ₈	These bits are the instruction for the two 2901's. Bit I ₇ is used to activate one of the U ₇₀ or U ₈₀ .
27	I ₇	
28	I ₆	
29	I ₅	
30	I ₄	
31	I ₃	
32	I ₂	
33	I ₁	
34	I ₀	
35	S _L	These bits control the U ₇₀ and U ₈₀ shift multiplexers.
36	S ₀	
37	B ₃	2901 B-address
38	B ₂	
39	B ₁	
40	B ₀	
41	A ₃	2901 A-address
42	A ₂	
43	A ₁	
44	A ₀	
45	CY _{IN}	Carry input to the 2901 array.
46	TRIS	(TRI-data-Source). Selects the RALU D-bus or the "TRI" pattern memory to load the TRI-latch.
47	MSKS	(Mask-data-Source). Enables the RALU D-bus or the "MASK" pattern memory to load the MASK-latch.
48	N.C.	These bits are not connected.
49	N.C.	

50	DATEN	Enables the Data latch U ₇₀ to receive new data.
51	TRIEN	Enables the TRI-latch to receive new data.
52	MSKEN	Enables the MASK-latch to receive new data.
53	MODEN	Enables the MODE-latch to receive new data.
54	CLKEN	Enables the CLOCK-latch to receive new data.
55	ADDREN	Enables the ADDRESS latch (U ₈₄) to receive new data.
56	N.C.	Not connected.
57	M ₇	These bits control the address multiplexers to select the address for the pattern memories ("TRI", "MASK", "MODE" and "CLOCK").
58	M ₆	
59	M ₅	
60	M ₄	
61	M ₃	
62	M ₂	
63	M ₁	
64	M ₀	
65	MC ₇	These bits provide an address source for the pattern memories.
66	MC ₆	
67	MC ₅	
68	MC ₄	
69	MC ₃	
70	MC ₂	
71	MC ₁	
72	MC ₀	
73	ALE1	These bits enable the address latches which feed the address to the pattern memories, to receive a new value.
74	ALE2	
75	ALE3	
76	ALE4	

APPENDIX B

WIRING DIAGRAM OF A 16-BIT PROCESSING
SECTION USING CARRY-LOOK-AHEAD LOGIC

FIGURE B.1 Wiring diagram of a 16-bit processing section using carry-look-ahead logic.

APPENDIX C

DEVICE AC SPECIFICATIONS

I. Am2901B Guaranteed Commercial Range Performance

The tables below specify the guaranteed performance of the Am2901B over the commercial operating range of 0°C to +70°C, with V_{CC} from 4.75V to 5.25V. All data are in ns, with inputs switching between 0V and 3V at 1V/ns and measurements made at 1.5V. All outputs have maximum DC load.

This data applies to the following part numbers: Am2901BPC
Am2901BDC

A. Cycle Time and Clock Characteristics.

Read-Modify-Write Cycle (from selection of A, B registers to end of cycle.)	60ns
Maximum Clock Frequency to shift Q (50% duty cycle, $I = 432$ or 632)	16MHz
Minimum Clock LOW Time	30ns
Minimum Clock HIGH Time	30ns
Minimum Clock Period	60ns

B. Combinational Propagation Delays. $C_L = 50pF$

To Output From Input	Y	F3	Cn+4	$\bar{G}, \bar{P}$	F=0	OVR	RAM0 RAM3	Q0 Q3
A, B Address	60	61	59	50	70	67	71	-
D	38	38	40	33	48	44	45	-
Cn	30	29	20	-	37	29	38	-
I012	50	47	45	45	56	53	57	-
I345	51	52	52	45	60	49	53	-
I678	28	-	-	-	-	-	35	35
A Bypass ALU ($I = 2XX$)	37	-	-	-	-	-	-	-
Clock	49	48	47	37	58	55	59	29

C. Set-up and Hold Times Relative to Clock (CP) Input.

Input	CP: Set-up Time Before H → L	Hold Time After H → L	Set-up Time Before L → H	Hold Time After L → H
A, B Source Address	20	0 (Note 3)	69 (Note 4)	0
B Destination Address	15	Do Not Change	-	0
D	-	-	51	0
Cn	-	-	39	0
I012	-	-	56	0
I345	-	-	55	0
I678	11	Do Not Change	-	0
RAM0, 3, Q0, 3	-	-	16	0

D. Output Enable/Disable Times.

Output disable tests performed with $C_L = 5pF$ and measured to 0.5V change of output voltage level.

Input	Output	Enable	Disable
$\bar{OE}$	Y	36	29

- Notes: 1. A dash indicates a propagation delay path or set-up time constraint does not exist.
2. Certain signals must be stable during the entire clock LOW time to avoid erroneous operation. This is indicated by the phrase "do not change".
3. Source addresses must be stable prior to the clock H → L transition to allow time to access the source data before the latches close. The A address may then be changed. The B address could be changed if it is not a destination; i.e. if data is not being written back into the RAM. Normally A and B are not changed during the clock LOW time.
4. The set-up time prior to the clock L → H transition is to allow time for data to be accessed, passed through the ALU, and returned to the RAM. It includes all the time from stable A and B addresses to the clock L → H transition, regardless of when the clock H → L transition occurs.

Am2910 SWITCHING CHARACTERISTICS

II. GUARANTEED CHARACTERISTICS OVER COMMERCIAL OPERATING RANGE

Am2910PC, DC ($T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$, $V_{CC} = 4.75\text{V}$ to 5.25V , $C_L = 50\text{pF}$)

A. Set-up and Hold Times

Input	t_s	t_h
$D_1 \rightarrow R$	24	6
$D_1 \rightarrow PC$	58	4
$Q_1 \rightarrow Q_2$	104	0
$\overline{CC}$	80	0
$\overline{CCEN}$	80	0
CI	46	5
$\overline{RLO}$	38	6

B. Combinational Delays

Input	τ	PL VECT, MAP	Full
$D_0 \rightarrow D_{11}$	20	-	-
$Q_1 \rightarrow Q_2$	70	51	-
$\overline{CC}$	43	-	-
$\overline{CCEN}$	45	-	-
CP (Note 2)	100	-	60
$I = 8, 9, 15$	125	-	60
CP All other I	55	-	60
$\overline{OE}$ (Note 3)	35/30	-	-

C. Clock Requirements (Note 1)

Minimum Clock LOW Time	50	ns
Minimum Clock HIGH Time	36	ns
Minimum Clock Period	138	ns
$I = 8, 9, 15$ (Note 2)	163	ns
Minimum Clock Period, $I = 14$	93	ns

(From AMD 2910 manual)

Am2921

SWITCHING CHARACTERISTICS
OVER OPERATING RANGE

SWITCHING CHARACTERISTICS OVER OPERATING RANGE		Am2921PC, DC		Am2921DM, FM		Units	Test Conditions
		$T_A = 0^{\circ}\text{C to } +70^{\circ}\text{C}$ $V_{CC} = 5.0\text{V} \pm 5\%$		$T_A = -55^{\circ}\text{C to } +125^{\circ}\text{C}$ $V_{CC} = 5.0\text{V} \pm 10\%$			
Parameters	Description	Min.	Max.	Min.	Max.		
t_{PLH}	A, B, C to Y_1		36		42	ns	$C_L = 50\text{pF}$ $R_L = 2.0\text{k}\Omega$
t_{PHL}			29		37		
t_{PLH}	$\overline{E}_1, \overline{E}_2$ to Y_1		34		39	ns	
t_{PHL}			38		45		
t_{PLH}	E_3, E_4 to Y_1		38		45	ns	
t_{PHL}			43		52		
t_{PLH}	POL to Y_1		29		34	ns	
t_{PHL}			39		49		
t_{ZH}	$\overline{OE}_1, \overline{OE}_2$ to Y_1		38		45	ns	
t_{ZL}			23		25		
t_{HZ}	$\overline{OE}_1, \overline{OE}_2$ to Y_1		29		33	ns	$C_L = 50\text{pF}$ $R_L = 2.0\text{k}\Omega$
t_{LZ}			33		36		

(From AMD 2921 manual)

Am2922

SWITCHING CHARACTERISTICS
OVER OPERATING RANGE

SWITCHING CHARACTERISTICS OVER OPERATING RANGE		Am2922PC, DC		Am2922DM, FM		Units	Test Conditions
Parameters	Description	$T_A = 0^{\circ}\text{C to } +70^{\circ}\text{C}$ $V_{CC} = 5.0\text{V} \pm 5\%$		$T_A = -55^{\circ}\text{C to } +125^{\circ}\text{C}$ $V_{CC} = 5.0\text{V} \pm 10\%$			
		Min.	Max.	Min.	Max.		
t_{PLH}	Clock to Y, POL-L		40		47	ns	$C_L = 50\text{pF}$ $R_L = 2.0\text{k}\Omega$
t_{PHL}			34		38		
t_{PLH}	Clock to Y, POL-H		29		33	ns	
t_{PHL}			35		41		
t_{PLH}	D_N to Y		19		21	ns	
t_{PHL}			22		24		
t_{PLH}	CLR to Y		39		45	ns	
t_{PHL}			39		45		
t_{PLH}	$\overline{ME}$ to Y		22		26	ns	
t_{PHL}			19		20		
t_{ZL}	$\overline{OE}$ to Y		19		24	ns	
t_{ZH}			22		29		
t_{LZ}	$\overline{OE}$ to Y		24		30	ns	$C_L = 50\text{pF}$ $R_L = 2.0\text{k}\Omega$
t_{HZ}			24		30		
t_s	A, B, C POL	11		12		ns	$C_L = 50\text{pF}$ $R_L = 2.0\text{k}\Omega$
	CE	18		20			
t_s	CLR Recovery	6		7		ns	
t_{pw}	Clock	11		12		ns	
	Clear (LOW)	11		12			
t_H	A, B, C, POL, CE	3		3		ns	

(From AMD 2922 manual)

54/7432, 54S/74S32, 54LS/74LS32

DC AND AC CHARACTERISTICS:

SYMBOL	PARAMETER	54/74		54/74S		54/74LS		UNITS	CONDITIONS	
		Min	Max	Min	Max	Min	Max			
ICCH	Power Supply Current	22		32		6.2		mA	V _{IN} = Open	V _{CC} = Max
ICCL		38		68		9.6			V _{IN} = Gnd	
tPLH	Propagation Delay	15	2.0	7.0		15		ns		
tPHL		22	2.0	7.0		15				

*DC limits apply over operating temperature range. AC limits apply at T_A = +25°C and V_{CC} = +5.0 V

54/74151, 54S/74S151, 54LS/74LS151

AC CHARACTERISTICS: V_{CC} = +5.0 V, T_A = +25°C

SYMBOL	PARAMETER	54/74		54/74S		54/74LS		UNITS
		$C_L = 15 \text{ pF}$ $R_L = 400 \Omega$		$C_L = 15 \text{ pF}$ $R_L = 280 \Omega$		$C_L = 15 \text{ pF}$		
		Min	Max	Min	Max	Min	Max	
t_{PLH} t_{PHL}	Propagation Delay S_n to Z	26 30		15 13.5		23 34		ns
t_{PLH} t_{PHL}	Propagation Delay S_n to Z	38 38		18 18		48 30		ns
t_{PLH} t_{PHL}	Propagation Delay E to Z	21 23		13 12		24 30		ns
t_{PLH} t_{PHL}	Propagation Delay E to Z	33 33		16.5 18		42 32		ns
t_{PLH} t_{PHL}	Propagation Delay I_n to Z	14 14		7.0 7.0		21 20		ns
t_{PLH} t_{PHL}	Propagation Delay I_n to Z	20 27		12 12		32 26		ns

(From FAIRCHILD TTL data book)

54/74174, 54S/74S/174, 54LS/74LS174

AC CHARACTERISTICS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

SYMBOL	PARAMETER	54/74		54/74S		54/74LS		UNITS
		$C_L = 15\text{ pF}$ $R_L = 400\ \Omega$		$C_L = 15\text{ pF}$ $R_L = 280\ \Omega$		$C_L = 15\text{ pF}$		
		Min	Max	Min	Max	Min	Max	
f_{max}	Maximum Clock Frequency	25		75		30		MHz
t_{PLH}	Propagation Delay		30		12		25	ns
t_{PHL}	CP to Q_n		35		17		22	
t_{PHL}	Propagation Delay		35		22		35	ns
	MR to Q_n							

AC OPERATING REQUIREMENTS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

SYMBOL	PARAMETER	54/74		54/74S		54/74LS		UNITS
		Min	Max	Min	Max	Min	Max	
$t_s\text{ (H)}$	Setup Time HIGH or LOW	20		5.0		10		ns
$t_s\text{ (L)}$	D_n to CP	20		5.0		10		
$t_h\text{ (H)}$	Hold Time HIGH or LOW	5.0		3.0		5.0		ns
$t_h\text{ (L)}$	D_n to CP	5.0		3.0		5.0		
$t_w\text{ (H)}$	CP Pulse Width HIGH	20		7.0		18		ns
$t_w\text{ (L)}$	MR Pulse Width LOW	20		7.0		18		ns
t_{rec}	Recovery Time MR to CP	25		5.0		12		ns

(From FAIRCHILD TTL data book)

54/74175, 54S/74S 175, 54LS/74LS175

AC CHARACTERISTICS: $V_{CC} = +5.0 \text{ V}$, $T_A = +25^\circ \text{C}$

SYMBOL	PARAMETER	54/74		54/74S		54/74LS		UNITS
		$C_L = 15 \text{ pF}$ $R_L = 400 \Omega$		$C_L = 15 \text{ pF}$ $R_L = 280 \Omega$		$C_L = 15 \text{ pF}$		
		Min	Max	Min	Max	Min	Max	
f_{max}	Maximum Clock Frequency	25		75		30		MHz
t_{PLH}	Propagation Delay CP to Q_n	30		12		25		ns
t_{PHL}		35		17		25		
t_{PHL}	Propagation Delay $\overline{\text{MR}}$ to Q_n	35		22		33		ns
t_{PLH}	Propagation Delay $\overline{\text{MR}}$ to $\overline{Q_n}$	25		15		24		ns

AC OPERATING REQUIREMENTS: $V_{CC} = +5.0 \text{ V}$, $T_A = +25^\circ \text{C}$

SYMBOL	PARAMETER	54/74		54/74S		54/74LS		UNITS
		Min	Max	Min	Max	Min	Max	
$t_s (H)$	Setup Time HIGH or LOW	20		5.0		10		ns
$t_s (L)$	D_n to CP	20		5.0		10		
$t_h (H)$	Hold Time HIGH or LOW	5.0		3.0		5.0		ns
$t_h (L)$	D_n to CP	5.0		3.0		5.0		
$t_w (H)$	CP Pulse Width HIGH	20		7.0		15		ns
$t_w (L)$	$\overline{MR}$ Pulse Width LOW	20		7.0		18		ns
t_{rec}	Recovery Time $\overline{MR}$ to CP	25		5.0		12		ns

(From FAIRCHILD TTL data book)

54S/74S253, 54LS/74LS253

AC CHARACTERISTICS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

SYMBOL	PARAMETER	54/74S		54/74LS		UNITS
		$C_L = 15\text{ pF}$ $R_L = 280\ \Omega$		$C_L = 15\text{ pF}$		
		Min	Max	Min	Max	
t_{PLH} t_{PHL}	Propagation Delay S_n to Z_n	18	18	29	24	ns
t_{PLH} t_{PHL}	Propagation Delay I_n to Z_n	9 0	9 0	20	15	ns
t_{PZH} t_{PZL}	Output Enable Time	19 5	21	22	22	ns
t_{PHZ} t_{PLZ}	Output Disable Time	8 5	14	32	22	ns

54LS374

AC CHARACTERISTICS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

74LS147				
SYMBOL	PARAMETER	54/74LS		UNITS
		CL = 45 pF		
		Min	Max	
fmax	Maximum Clock Frequency	35		MHz
tPLH tPHL	Propagation Delay CP to On		28 28	ns
tPZH tPZL	Output Enable Time		28 28	ns
tPHZ tPLZ	Output Disable Time		20 25	ns

AC OPERATING REQUIREMENTS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

SYMBOL	PARAMETER	54/74LS		UNITS
		Min	Max	
t_s (H) t_s (L)	Setup Time HIGH or LOW D_n to CP	20 20		ns
t_h (H) t_h (L)	Hold Time HIGH or LOW D_n to CP	0 0		ns
t_w (H) t_w (L)	CP Pulse Width HIGH or LOW	15 15		ns

(From FAIRCHILD, TTL data book)

54LS/74LS375

AC CHARACTERISTICS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

54/74LS				
SYMBOL	PARAMETER	C _L = 15 pF		UNITS
		Min	Max	
t _{PLH} t _{PHL}	Propagation Delay D _n to Q _n		27 17	ns
t _{PLH} t _{PHL}	Propagation Delay D _n to Q _n		20 15	ns
t _{PLH} t _{PHL}	Propagation Delay E _n to Q _n		27 25	ns
t _{PLH} t _{PHL}	Propagation Delay E _n to Q _n		30 15	ns

AC OPERATING REQUIREMENTS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

SYMBOL	PARAMETER	54/74LS		UNITS
		Min	Max	
t_s (H) t_s (L)	Setup Time HIGH or LOW D_n to E_n	20		ns
t_h (H) t_h (L)	Hold Time HIGH or LOW D_n to E_n	0		ns
t_w (H)	E_n Pulse Width HIGH	20		ns

(From FAIRCHILD TTL data book)

54LS/74LS377

AC CHARACTERISTICS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

SYMBOL	PARAMETER	54/74LS		UNITS
		$C_L = 15\text{ pF}$		
		Min	Max	
f_{max}	Maximum Clock Frequency	30		MHz
t_{PLH} t_{PHL}	Propagation Delay CP to Q_n		25 25	ns

AC OPERATING REQUIREMENTS: $V_{CC} = +5.0\text{ V}$, $T_A = +25^\circ\text{C}$

SYMBOL	PARAMETER	54/74LS		UNITS
		Min	Max	
t_s (H) t_s (L)	Setup Time HIGH or LOW D_n to CP	10 10		ns
t_h (H) t_h (L)	Hold Time HIGH or LOW D_n to CP	5.0 5.0		ns
t_s (H) t_s (L)	Setup Time HIGH or LOW $\bar{E}$ to CP	10 20		ns
t_h (H) t_h (L)	Hold Time HIGH or LOW $\bar{E}$ to CP	5.0 5.0		ns
t_w (H) t_w (L)	CP Pulse Width HIGH or LOW	20 20		ns

(From FAIRCHILD TTL data book)

END

06-09-18-4

FIN