This is to certify that the

thesis entitled

TESTABILITY DESIGN OF THE DKS CHIP

presented by

TEJ PAL SINGH

has been accepted towards fulfillment
of the requirements for

Master's degree in Electrical Engineering

Richard Simkat
Major professor

Date 8-12-88

TESTABILITY DESIGN OF THE DKS CHIP

By

Tej Pal Singh

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Electrical Engineering

1988

ABSTRACT

TESTABILITY DESIGN OF THE DKS CHIP

By

Tej Pal Singh

A VLSI chip has been recently proposed to calculate the Direct Kinematic Solution (DKS) for real time robotic control. It was not considered feasible to implement such a complex chip without any testability features integrated into it during the design stage. Standard Design-for-Testability (DFT) structured approaches require a lot of area overhead (upto 20%). Thus a modification of the existing DFT techniques called Bus Scan Testing (BST) has been developed that makes use of the internal bus available on the chip to access all the storage components in the circuit. The design modifications required to implement BST on the DKS chip are described. The test vectors required to test the DKS chip are generated and lastly, the test application process is simulated. Performance evaluation results indicate that BST required lesser area overhead, shorter test vector length, and lesser test application time than LSSD.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

The number of components on a single IC has been steadily increasing over the last two decades. Due to this, more complex circuits in terms of both transistor count and circuit function are being built and the problem of testing such circuits has become critical. Currently, a significant effort is being devoted to the aspect of Design-For-Testability (DFT) in which the circuitry for testing the chip is incorporated onto the chip during the design process.

In the past, rigorous testing was only carried out on military and aerospace projects [1]. But now, the requirements and usage of integrated circuits in everyday life have become so great that the reliability of such circuits has become essential. The progress in DFT and IC testing techniques has only partially offset the increasing complexity of the systems. One reason for this state of affairs is that the equipment to test a particular circuit must be an order of magnitude faster than the circuit itself [2].

Various factors affect the testing cost of a circuit and there is a tradeoff between the test cost and the repair cost. The cost of testing will be less if the fault coverage of the tests is less, which leads to higher repair costs in the system. It has been estimated that if the cost of testing an IC is 3 cents, then the cost of testing the same on a board is 30 cents, on the system is $3, and in the field is $30 [3]. Thus, it is of utmost importance that the chips be properly tested first and only then inserted into boards and systems.

Test generation for MSI and LSI was initially done by a functional model or by a heuristic approach rather than by logic simulation and fault simulation techniques. The reason was that there was little interaction between the design engineer and the test engineer. The test engineer usually got the device from a product engineer without much documentation. Full logic information was rarely available to the test engineer or the user. This led to a time

consuming, costly, and ineffective test process which was often an overkill for the device [4].

With the advent of VLSI, some efficient test generation algorithms were developed based mostly on stuck-at fault models. The test vectors could then be generated by fault simulation, which further requires good logic simulation. Many CAD systems have been developed to handle logic and fault simulation. However, even with these efficient algorithms and increasing CPU speeds, the time required to generate test vectors for complex VLSI devices is often intolerable. This is especially true for sequential circuits because the output of such circuits depends on the current state of the circuit apart from the primary inputs. Algorithms for for test generation of sequential circuits are still in the research stage. Thus methods to decrease the cost and time required to generate and verify test vectors must be found.

The concept of DFT has evolved from the realization that the only method to significantly reduce the cost of testing is to include circuitry on each chip to facilitate such testing. This has led to the development of some techniques which the designer can incorporate into a design with various degrees of effort. They result in reduction of the complexity of the full testing process from test generation and verification to test application, which makes testing faster and more economical. These DFT techniques are, in some cases, general guidelines to improve testability; others are hard and fast rules. The designer can select among these techniques depending on their cost of implementation, specific test requirements and their effectiveness on specific types of circuits.

Recently, a single chip implementation of a circuit to find the Direct Kinematic Solution (DKS) of a robot arm was developed which has an internal bus architecture [5]. Figure 1.1 shows the block diagram of this DKS chip. The aim of the designer was to do all the calculations on a single custom designed IC. The chip is expected to reduce the computation time for the DKS by three orders-of-magnitude when compared to the time required by a 16-bit microprocessor [5]. However, for reasons enumerated earlier, it was not considered feasible to implement this complex chip without any testability features on it.

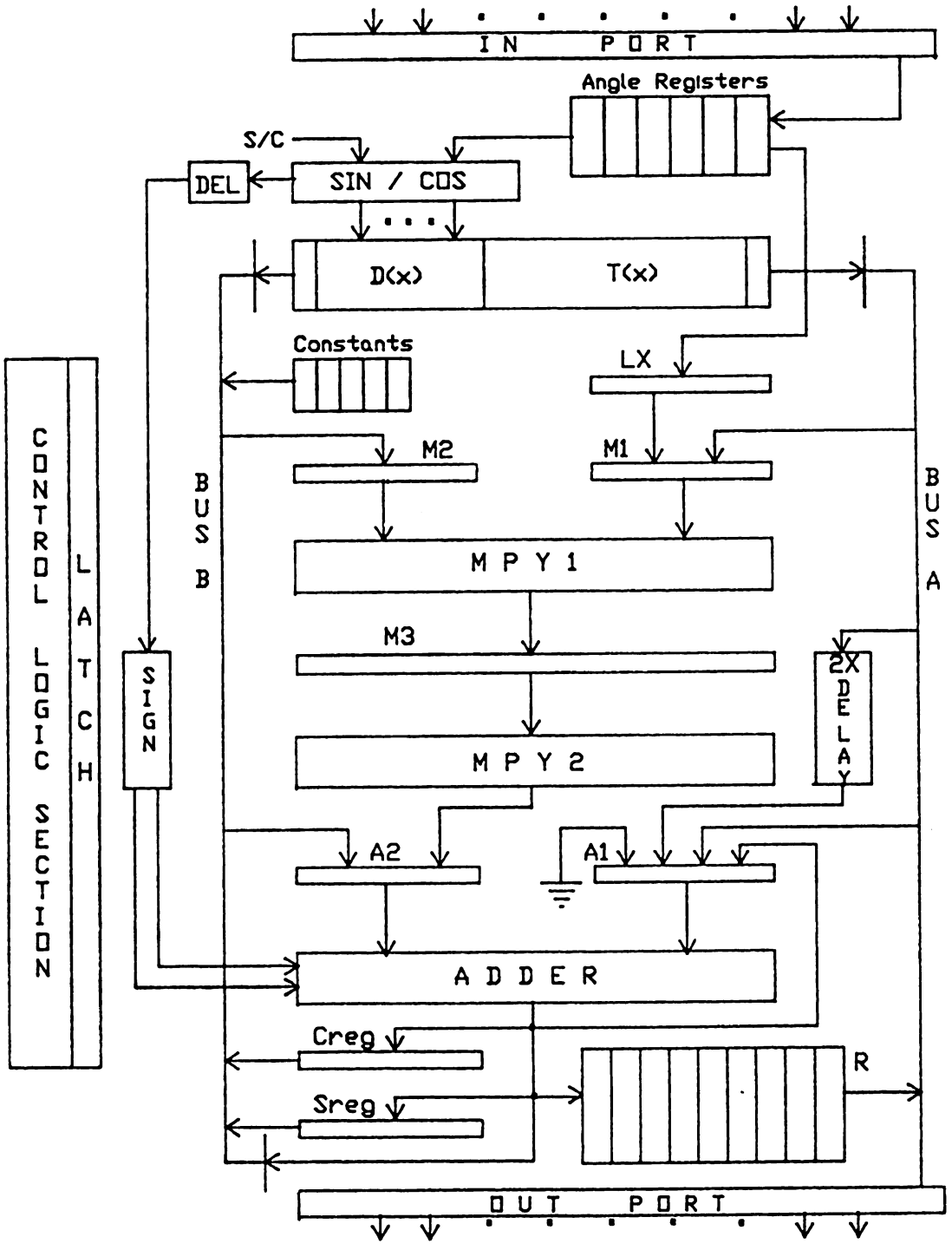

Figure 1.1. The DKS chip block diagram [5].

Systems with bus architectures have been identified as being relatively easy to test. In fact, bus architecture itself is one of the *ad hoc* approaches to DFT suggested in the literature. But, it has always been assumed that the bus is available externally and thus can be easily observed. However, on a chip with an internal bus architecture, the bus is not available at the pins and the problem of testing becomes as complex as any other IC. Such is the case with the DKS chip.

Scan design methodology has emerged as one of the most promising DFT techniques being implemented to test sequential circuits. These techniques conceptually convert a sequential circuit to a combinational one so that well proven combinational testing techniques can be used. Moreover, they provide a structured approach to the problem and thus can be easily adapted to design automation. Various scan techniques could be applied to the DKS chip. Though Level Sensitive Scan Design (LSSD) and Random Access Scan (RAS) techniques seemed to be good choices at first glance, they don't exploit the internal bus available on the chip. It seems obvious that if somehow the bus could be used to access all the components connected to it on the chip, the overhead in area required for extra testability components may be reduced.

This thesis begins with a background section (Chapter II) which is divided into two parts. The first part explains the importance of DFT and the various techniques that have been developed. The second part then gives a brief overview of the DKS chip, its importance, its development, and its structure. In the Chapter III of the thesis, the DKS chip is studied in context of its testing problems. The applicability of the various DFT techniques are then discussed. Finally, the justification for modifying the earlier scan techniques to apply them to the DKS chip is given.

In the Chapter IV of this thesis, a new DFT technique called Bus Scan Testing (BST) is developed that can be applied to circuits with an internal bus architecture. The design modifications required to implement BST are first discussed. This is followed by the description of its operation on a chip. The advantages and disadvantages of BST over other scan

techniques are also presented.

The next three chapters of the thesis are devoted to the specific testing aspects of the DKS chip. First, the modifications required in its design to incorporate BST are presented in Chapter V. Next, the test vectors required to test the chip are generated in Chapter VI assuming a functional fault model for the circuit. Then, the actual test application process is described and simulated in Chapter VII.

Finally, the performance of the DKS chip with BST is evaluated and compared with the estimated figures for other DFT techniques in Chapter VIII. The thesis concludes with a summary and directions for future work in this area.

II. BACKGROUND

The problem of testing can be divided into three parts: *test generation, test verification, and test application.* The test generation problem is to generate a set of test patterns that can adequately exercise the system by providing a sufficient fault coverage. The generation of tests is subject to constraints imposed by circuit access, the tester, and time required to run the tests. The test vectors are generated so as to test specific faults (single stuck-at, bridging, etc.) or to test the functionality of the circuit. Test verification evaluates the test vectors and tries to prove that they are effective (sufficient) towards the end goal of verifying the correct functioning of circuit. This is usually done by making use of logic fault simulators. The last part of the testing process is the actual application of the tests to the circuit. Automatic Test Equipment (ATE) and in-circuit board testers are use to apply the tests, make and record measurements, and in some cases trace the fault to specific areas in the circuit.

2.1. Design For Testability

All the three parts of the testing process are becoming more complex as the ratio of number of devices per chip to the number of pins increases. The costs of testing have gone up considerably. This has given rise to the concept of Design For Testability (DFT) to reduce the cost and effort required to test ICs.

The goal of DFT is to ease the process of test generation and verification by taking some steps during the design process itself. Some extra effort during the design process can save a lot of effort during the test and can also help automate the testing in certain ways. It is appropriate here to note that we are talking here about *explicit testing* which is carried out while the circuit is not in use, as opposed to *implicit/concurrent testing* which refers to on-line testing

to detect errors that occur during system operation.

The first task here is to define what is meant by testability of a circuit, and how is it quantified. Testability can be loosely defined as follows [6]:

> A circuit is *testable* if a set of test patterns can be generated, evaluated and applied in such a way as to satisfy pre-defined levels of performance, defined in terms of fault-detection, fault-location, and test application criteria, within a pre-defined cost budget and timescale.

Controllability and *observability* are the two measures used to quantify testability of a given circuit. Controllability refers to the ease of producing a specific internal signal value by applying signals to the circuit input leads, while observability is the ease with which the state of internal signals can be determined at the circuit output leads. Various methods have been proposed which estimate the testability of a circuit without having to run an Automatic Test Pattern Generation Program (ATPG). ATPG will of course be the direct way to measure testability but it turns out to be very expensive in terms of both cost and time. It may also not give clear indications as to how testability of a particular circuit can be improved.

Several programs have been developed to measure the testability of a circuit. TMEAS (Testability Measure Program) [7], SCOAP (Sandia CY/OY Analysis Program) [8], CAMELOT (Computer Aided Measure for Logic Testability) [9], and VICTOR (VLSI Identifier of CY,TY,OY, and Redundancy) [10], are some of the programs written for this purpose. TMEAS and CAMELOT calculate two values (controllability and observability) for each node of the circuit, while SCOAP calculates a vector of six values for each node. VICTOR can only be used for measuring the testability of combinational circuits.

The main objective of DFT techniques is to increase the controllability and observability of a circuit. The techniques developed for DFT have been classified into two categories: *ad hoc techniques* and *structured techniques*. The *ad hoc* techniques are aimed at the designer and present him with a set of design features which ease the testing of a chip. They are not always applicable to all types of designs. The structured techniques on the other hand can be applied

to almost all types of designs. They provide the designer with a set of rules leading to the design of a testable circuit.

2.1.1. Ad Hoc Techniques

The *ad hoc* techniques present the designer with a list of design features that help increase the testability of the circuit and points to those features that create testing problems for the circuit. Some *ad hoc* techniques identify alternative preferable implementations where ever applicable. Some aspects of the important *ad hoc* techniques are listed below [3,6,11].

1) **Extra Test Points:** The most direct way to enhance the testability of an IC is to add extra test points at critical places in the circuit. Multiplexers and demultiplexers may be used to keep the pin count down and still provide more access to the circuit.

2) **Initialization Circuitry:** Initialization of all stored logic devices to a known state is important before carrying out any test. It is even more helpful if all such devices can be individually initialized to specific values.

3) **Partitioning:** It has been shown that the cost of testing a circuit increases from somewhere in between square to cube of the complexity of the circuit. Thus partitioning the circuit into smaller parts and testing them part by part is much cheaper and easier than testing the full circuit as an entity.

4) **External Test Clock:** It should be possible to disconnect an IC's internal clock, if any, and connect an external test clock in its place. Thus the tester is able to perform single step operations, and reduce the speed of the circuit if required.

5) **Feedback Loops:** If all feedback loops present in the circuit can be broken, then the testing becomes much easier. Tristate control and tester inhibit logic are some of the ways of achieving this objective.

6) **Bus Architecture Systems:** If a bus architecture is used, it then becomes possible to test each component connected to the bus individually by floating all other components. This serves to partition the circuit.

Some other design features that are to be avoided to maximize testability during circuit design are wired logic, high fanout points, deep sequential circuits (like counters), monostables, potentiometers, and asynchronous logic. It has been found that such features on a chip make their testing more complex. Another suggestion that is given with respect to the test program is to avoid its dependency on ROM type devices which are more subject to changes during and after the design process.

2.1.2. Structured Techniques

The techniques to be described under this section reduce the sequential complexity of the system thus increasing the circuits' controllability and observability. They provide access to many points in the IC without assigning one pin to each point; typically four pins are required irrespective of the number of test points. They also transform a sequential circuit into a combinational circuit thus drastically reducing the number of vectors required to test it. The cost of all this, however, is that the test process becomes serialized, i.e., all vectors in to and out of the IC are transmitted in serial. Also, most of these techniques require a large area overhead (up to 20%). However, these disadvantages are often offset by their advantages for most of the complex chips. The various techniques will now be explained.

1) Scan Path: The scan path technique was the first structured approach to DFT and was introduced by members of Nippon Electric Co., Ltd. in 1975 [13]. It makes use of raceless D-type flip-flops which have two latches connected in series where the first latch has two inputs and two clocks as shown in Figure 2.1. All latches in the circuit under test are converted to these raceless D-type flip-flops. There are two modes of operation for the flip-flops: *normal mode* and *test mode*. In the normal mode, the flip-flop works as usual with a D input and C output using the system clock 1. However, during the test mode, the flip-flop takes its input from scan-in and uses test clock 2. The first flip-flop in the circuit is connected to a Scan-In (SI) line. After that, the input of each flip-flop is connected to the output I of the previous flip-flop.

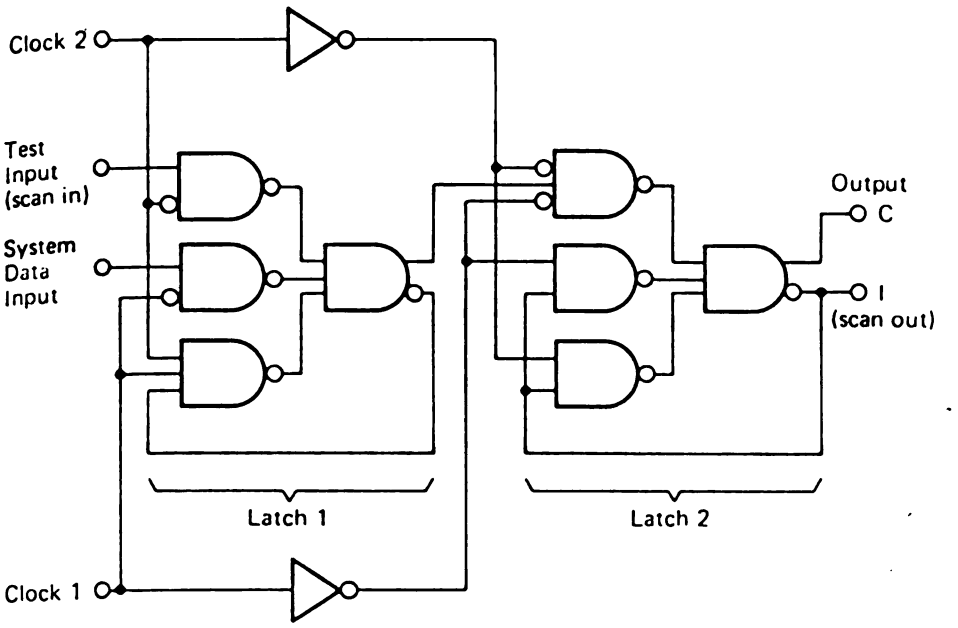

Figure 2.1. Raceless D-type flip-flop with scan path [3].

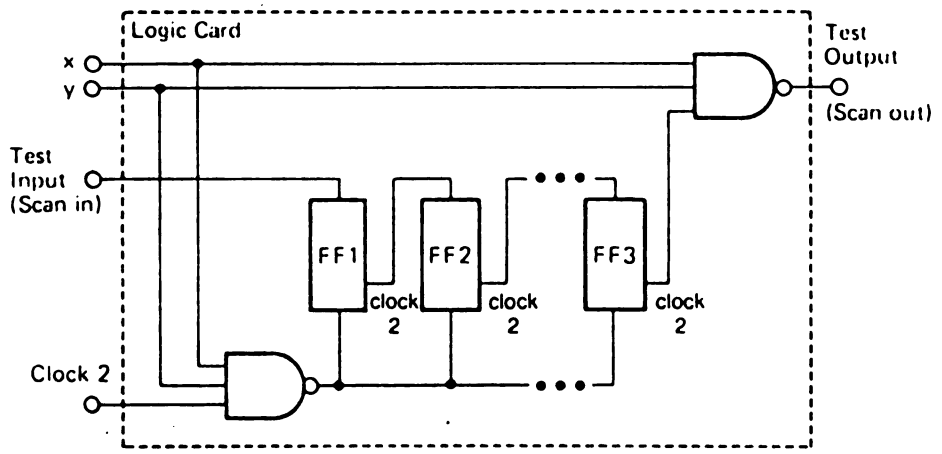

Figure 2.2. Configuration of Scan Path in circuit [3].

The output I of the last flip-flop goes to a pin called Scan-Out (SO). This essentially produces a long shift register spanning all the latches in the circuit (Figure 2.2). The test is carried out in following steps:

Step 1: Shift in and out a test pattern through the shift register. This tests all the flip-flops for correct operation.

Step 2: Shift in a test pattern and set all the primary inputs of the circuit to a test vector value.

Step 3: Apply one system clock to latch the response of the combinational networks in the circuit into the flip-flops.

Step 4: Scan out the contents of the shift register. The next test pattern can be scanned-in from the SI pin concurrently.

During normal operation of the circuit, the SI inputs and SO outputs of the flip-flops are not used and clock 2 is kept at 1 for the entire period. Thus the raceless D-type flip-flop acts just like a normal D-latch. Since only one system clock is used here, the circuit is exposed to a race condition between the two latches in series. The next approach uses two system clocks to rectify this problem.

2) Level Sensitive Scan Design: This technique (LSSD for short), was first presented by T. W. Williams and E. B. Eichelberger of IBM Corp. [14]. Since then many modifications have been suggested [15,16,17,18] and it has gained large popularity among other manufacturers. LSSD is similar to Scan Path, but because it is level sensitive, the constraints on circuit excitation, logic depth, and handling of clocked circuitry are not imposed. It uses a Shift Register Latch (SRL) as shown in Figure 2.3 as the basic design elements instead of raceless D-type flip-flops,. The SRL contains two latches in series operated by separate non-overlapped clocks to avoid races. These SRLs are again threaded, as in the case of Scan Path, to act as a shift register during test mode while they act as normal D-latches during normal system operation (see Figure 2.4). To operate the SRL, scan clock A is set to 1 to enable data from Scan In (SI)

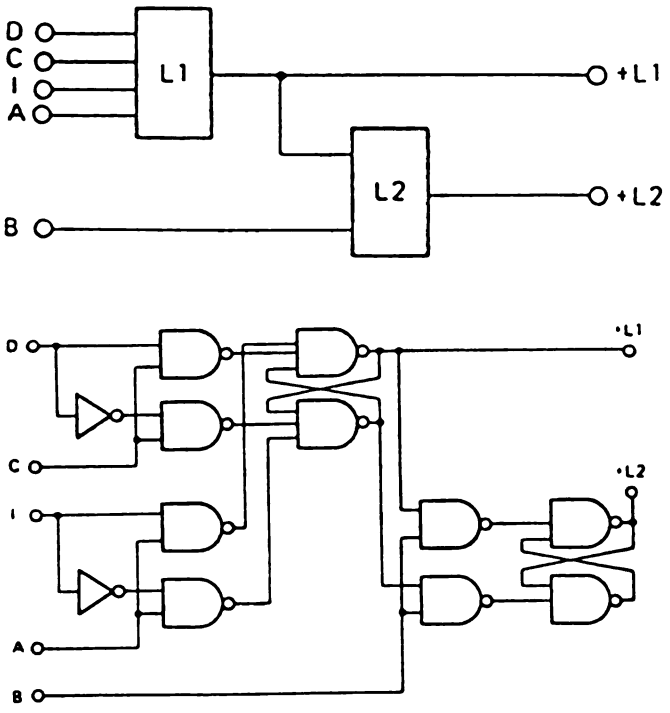

Figure 2.3. Shift Register Latch [3].

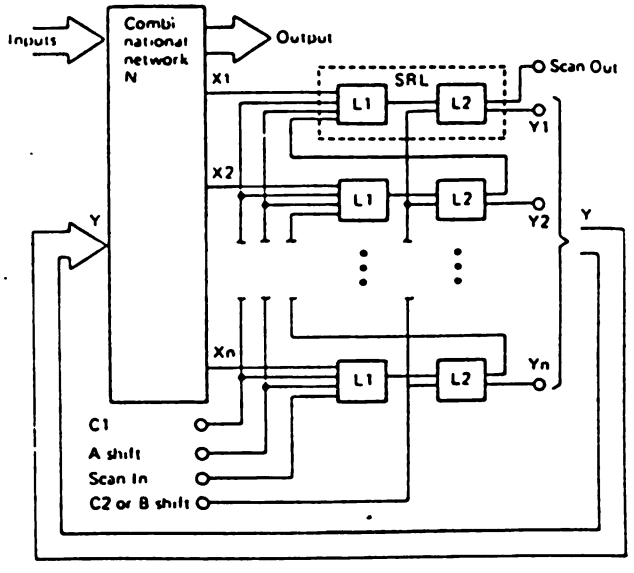

Figure 2.4. Level Sensitive Scan Design [3].

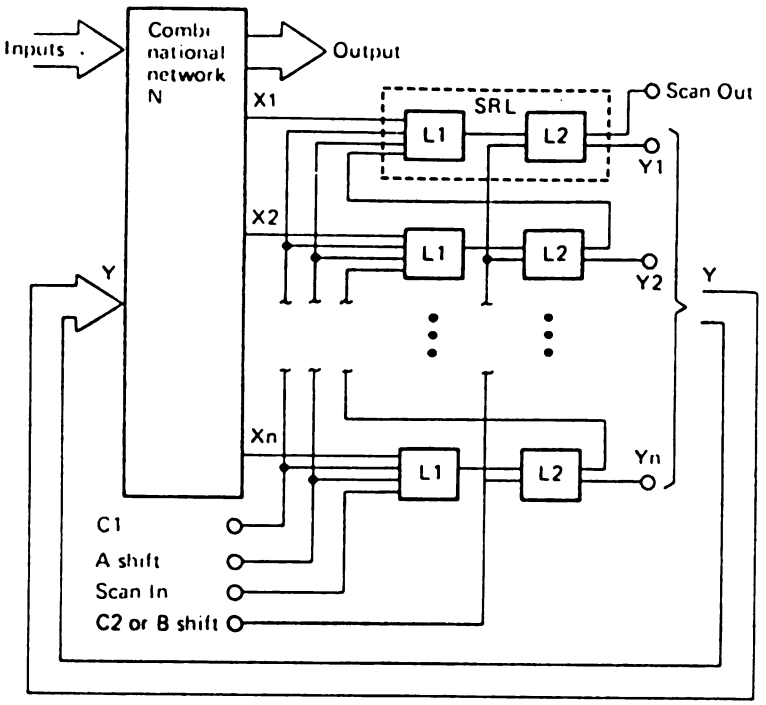

Figure 2.5. LSSD double latch design [3].

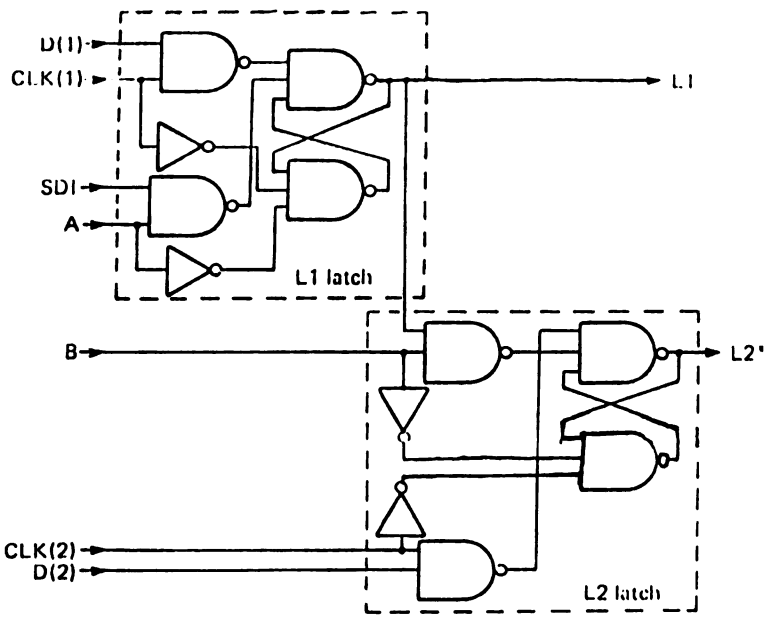

Figure 2.6. LSSD L1/L2* latch design [6].

to be latched into L1. Scan clock A is then returned to 0, latching SI and then B is set to 1 to transfer data from L1 to L2. Data is finally latched in L2 when B returns to zero.

Two configurations using these SRLs are possible. First is a double-latch configuration as shown in Figure 2.5, where the output of the SRL is taken from L2 for both system mode and test mode. Here, both the latches are in the system path thus increasing the propagation delay. A single-latch configuration (Figure 2.3), on the other hand, takes system output from latch L1 and test output from latch L2. Thus only latch L1 remains in the system path. This also implies that now latch L2 is redundant during system operation, leading to a significant hardware overhead cost for improving testability. A modification of the single-latch configuration was recently suggested [17], where L2 is also used as an independent latch during normal system operation. This L1/L2* latch (Figure 2.6), thus results in a very small hardware overhead. The only restriction is that now both the inputs can not be taken from the system simultaneously.

3) Random Access Scan: The objective of Random Access Scan (RAS) is the same as that the earlier two methods i.e. to reduce a sequential circuit to a combinational circuit and to have complete controllability and observability. However, RAS does not uses a scan path for this purpose. It allows each latch in the circuit to be separately accessed to clear/preset it or observe it using an addressing scheme (thus the name random-access) [19]. Polarity-hold type or set/reset type addressable latches can be used (Figures 2.7 and 2.8). They are addressed by a shift register and a decoder. The working of RAS is described next.

The address of the latch to be controlled or observed is entered serially from the scan address line using the scan clock and decoded using the latch select decoder. The latch is then cleared or preset using the CLR or PR line respectively. The procedure is repeated for all the applicable latches and then primary inputs are applied to the IC. After one system clock, all the applicable latches are again addressed one by one and data observed at the SDO pin. It should be noted here that data can be scanned out even during normal system operation. Also, the test is slow because each latch has to be separately addressed and set before each test and then

read after each test.

4) Scan/Set Logic: This scheme is also similar to the first two techniques. Here also there is no scan path present like RAS. The shift registers are present, but they are not in the system path. They are independent of all the latches in the system. In a single clock the full shift register is loaded with values from the system latches. Then the data is shifted out from the shift register using the scan clock (Figure 2.9). The reverse operation is followed to load the system latches, i.e., first data is scanned in to the shift register and then loaded into the system latches by a single system clock [20]. An advantage of scan/set logic is that the data can be clocked in and out of the shift register while the main circuit is in full operation.

5) Built-In Self Tests: This approach of Built-in Self Test (BIST) is a widely used method for testing ICs as well as boards. There are quite a few methods available to employ self testing in a circuit. Built-in Logic Block Observation (BILBO) [21], syndrome testing [22], testing by verifying Walsh coefficients [23], and autonomous testing [24] are some of the techniques available. The advantages of these techniques is that they use an on-chip pseudo random test generation process. However, they only give a go/no-go indication for the chip without any diagnostics.

6) Signature Analysis: [25] Signature analysis lies somewhere between the *ad hoc* and structured techniques. It requires that some design rules be followed during the design stage, but its objective is not the same as for structured techniques (to increase controllability and observability of the circuit). Signature analysis is based on Linear Feedback Shift Registers (LFSR) as illustrated in Figure 2.10. These are shift registers with feedback tapped off from some intermediate points and fed to the input through XOR gates to preserve linearity. The circuit is first initialized and a fixed number of clock pulses applied to it. The resultant value in the shift register is the signature to be compared with a correct signature.

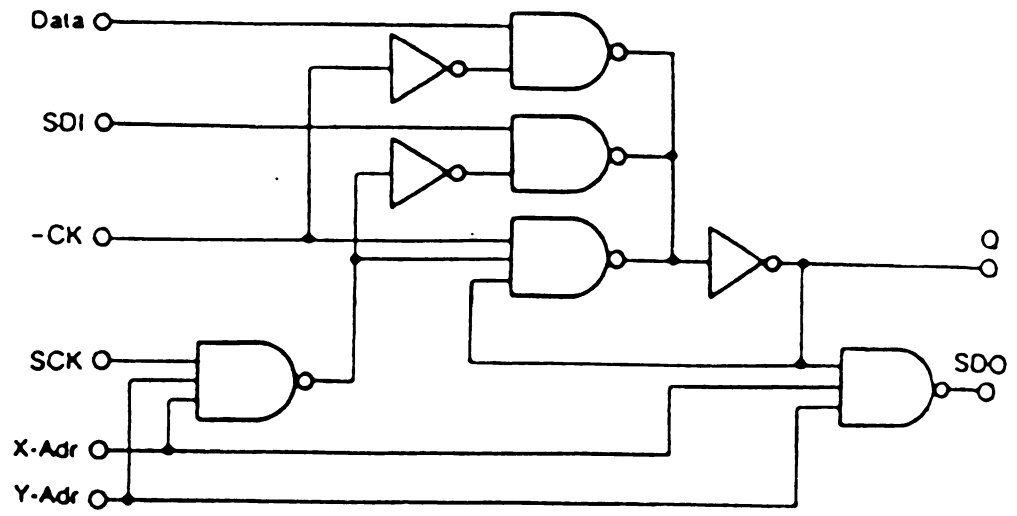

Figure 2.7. Polarity-hold-type addressable latch [3].

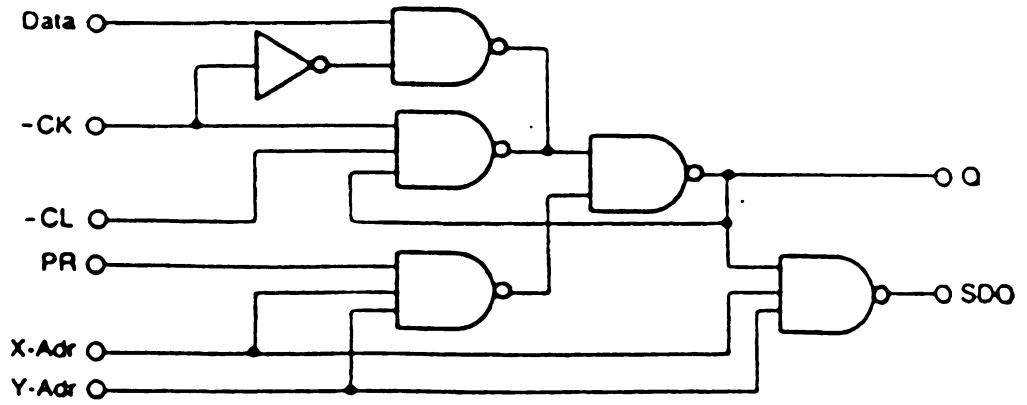

Figure 2.8. Set/Reset type addressable latch [3].

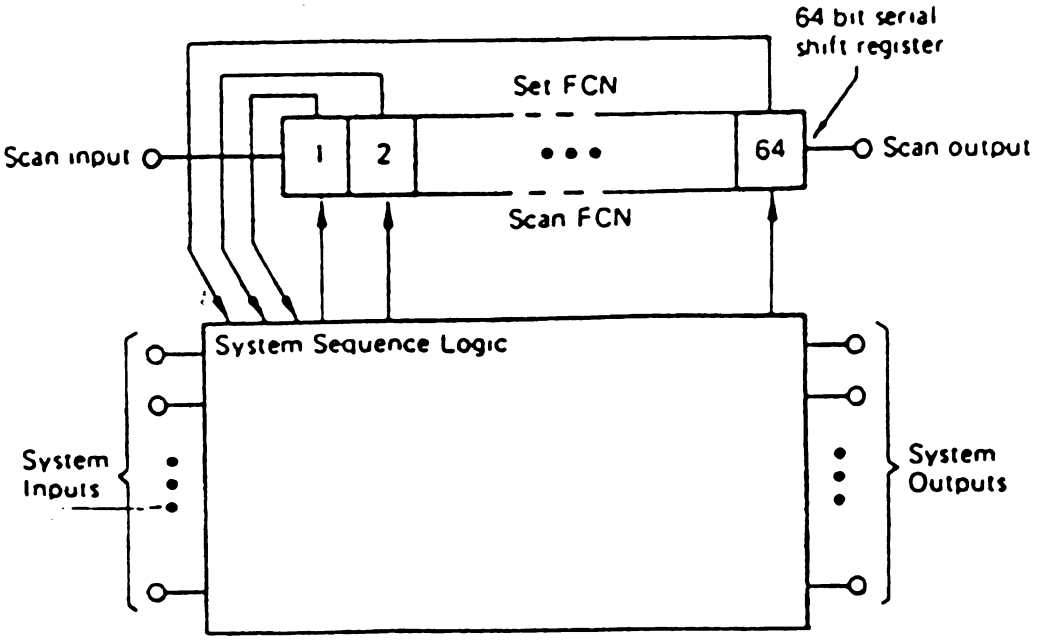

Figure 2.9. Scan/Set logic [3].

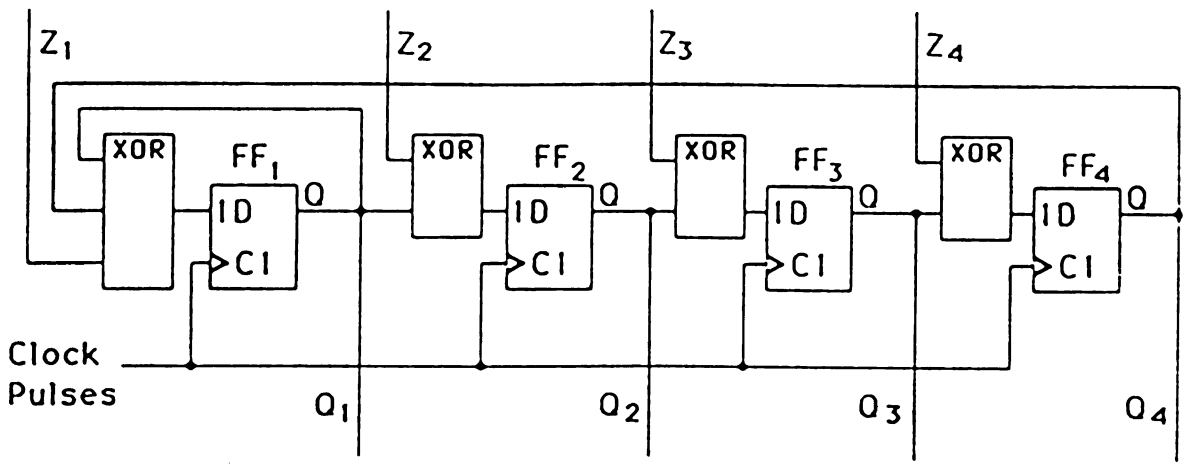

Figure 2.10. LFSR as a parallel signature analyzer [12].

2.1.3. Analysis of Structured Techniques

The various structured techniques described above can drastically reduce the effort required for IC testing by adding some extra effort during the design phase. Moreover, each is easy to automate, because of their strict structured approach. Several companies such as IBM, Fujitsu Ltd., Sperry-Univac, and Nippon Electric Co., who are basically mainframe manufacturers, have realized the importance of DFT and have ongoing research efforts in this area. Despite the advantages of these techniques, there has been a slow growth in their usage because of two main disadvantages.

Firstly, most of the techniques require an extra silicon area overhead and an increased number of pins to implement them on an IC. LSSD and scan path have a 4 to 20% area overhead associated with them and RAS may require up to twice that figure [3]. Thus it is important to somehow modify the existing techniques to decrease the area overhead. All the scan techniques also require at least 4 pins on the IC dedicated to testing.

Secondly, these techniques tend to serialize the test process because the full test vector has to be scanned in and out one bit at a time. The process is repeated for each test pattern applied to the circuit, though the next vector in sequence can be scanned in during the same time as the current pattern is being scanned out. The test application times increase considerably because of this serialization. Built-in tests and signature analysis do better in this aspect, since they use an on-chip pseudo-random test generation process.

2.2. The Direct Kinematic Solution (DKS) Chip

A robotic manipulator with multiple degrees of freedom can be controlled by a computerized system for automatic or remote operation. However, the computer must accurately know the position (usually an angle) of each joint of the robot at all times. For this purpose, one transducer is required for each joint of the robot. These transducers feed back the position information to the computer at regular intervals. The computer can thus calculate the position, velocity, and acceleration of the arm of the robot at any given time. It is conventional to

express all the positions in terms of a reference world coordinate system. The Direct Kinematic Solution (DKS) converts the joint angle vector received from the robot to the reference system's coordinates.

A homogeneous transformation method has been developed to solve the DKS effectively. The method requires successive multiplications of transformation matrices. Any joint i is represented by the angle θ_i in its own coordinate system. A_i, a function of θ_i, maps vector in the link i^{th} coordinate system to link $i{-}1^{th}$ coordinate system. Thus the joint space to cartesian mapping is:

$$T = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 = [n\ s\ a\ p] \qquad [2.1]$$

where n, s, and a give the orientation of the wrist and p gives the arm position.

It has been shown in [5] that the matrices A_2, A_3, and A_4 can be modified to make them more symmetric without any change in the final result. The values of the matrices A_i before (2.2) and after (2.3) the modification are as follows:

$$A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} C_3 & 0 & S_3 & a_3C_3 \\ S_3 & 0 & -C_3 & a_3S_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.2]$$

$$A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & 0 \\ S_2 & C_2 & 0 & a_2S_0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} C_3 & 0 & S_3 & a_2 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4 = \begin{bmatrix} C_4 & 0 & -S_4 & a_3 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [2.3]$$

Also only two out of the three orientation vectors are required to fully specify the orientation of the wrist. Thus s can be dropped making A_6 a 4-by-3 matrix.

In real-time robotic systems, this computation may have to be carried out tens or even hundreds of times a second to track the exact position of the robot arm. This requires a lot of computations and a computer (or a controller as the case may be) may not be able to handle it

especially along with the other computations required for proper operation of the robot. Thus, it is necessary to either decrease the computation time of the DKS or to use a coprocessing chip dedicated to compute the DKS in real-time.

A VLSI ASIC design to calculate the DKS in hardware has been recently developed [5]. It was shown that it would reduce the computation time by three orders-of-magnitude over that required by a 16-bit microprocessor. This was further verified when the design was implemented on a general-purpose signal-processor [26] and the results showed a marked improvement in the time required to calculate the DKS. However, the chip has still not been implemented in ASIC form. The main reason is that it is not considered feasible to implement such a complex chip without any testability features on it.

Figure 1.1 shows the block diagram of the DKS chip as in [5]. It assumes MOS technology and uses a two-phase nonoverlapped clock. It shows that the chip is based on two internal busses *A* and *B*. The input from the robot is latched in a register, and after processing, the results are stored in an output register. The circuit is organized as a finite state machine and the control section, along with a counter, controls the flow of data on the chip. The basic computational structure consists of a two-stage multiplier and an adder in a pipeline, both using two's complement fixed-point arithmetic. They are supported by a number of registers to store intermediate results. The control logic, constants and all table values for the calculation of sines and cosines are stored in a ROM.

The first step is to compute sines and cosines of the input angles. A method for sine-generation proposed by Ruoff was found to be best suited for the chip [27,5]. It is based on a ROM look-up table with linear interpolation using the multiplier-adder pipeline available on the DKS chip. It was shown that an 18-bit word length of 256 entries would be required to get the accuracy necessary for DKS calculation [5].

The algorithm to calculate the full DKS was developed and described in RTL [5]. Symbolic simulation showed that once the six input angles are stored in the angle registers, it takes 73 steps (system clocks) to calculate the full DKS using this circuit [5].

It was estimated that about 4,300 standard cells are required to implement the DKS chip, assuming a 10% estimation error and a 70% cell utilization. The IBM's Master Image approach shows that the chip can be implemented with 1.25 μm NMOS technology with a chip edge of about 5.6 mm. The CMOS chip will require a total of 5,317 cells with the chip edge being about 6.34 mm thus verifying the feasibility of the DKS chip using current technology [5].

III. TESTABILITY CONSIDERATIONS FOR THE DKS CHIP

The DKS chip has been designed for the PUMA robot. But, before it can be manufactured, a prototype will have to be fabricated to verify the design and logic on the chip. This requires better diagnostic capability for the tests to do design verification for the circuit, apart from testing for interconnection faults on the chip. Further, the chip will have to be modified a little to use it for other robots, and again new prototypes will have to be made for each new design. Thus, the testability design should be such that it will work for all of them and is geared towards *prototype testing* rather than *manufacturing testing*, the latter being applicable to those chips which are manufactured in large quantities.

Why is testing the DKS chip so difficult? Well, consider the possibility that the chip is to be tested by the same method as is used for small ICs, which is exhaustive testing. The number of test patterns required to test any sequential circuit is *at least* $2^{(m+n)}$, where m is the number of primary inputs to the circuit and n is the number of latches in the circuit [3]. The DKS chip has a 12-bit primary input and has nineteen 18-bit latches, six 12-bit latches, one 1-bit latch, one 7-bit counter and one 36-bit latch taking the total number of latches to 458. Thus the number of test vectors required to test the DKS chip by exhaustive testing is *at least* 2^{470} ! Even if one pattern is applied to the chip every 1 μsec, it will take 10^{127} years to test it!

Apart from this problem, which is common to every large sequential circuit, the DKS chip also has a ROM which stores the control logic, values for sine and cosine generation, and constants of the transformation matrices of the DKS. Since the contents of the ROM will differ for each robot, the values stored in it must be verified for correct results. Each cell in the ROM should be individually verified as some of the locations are used more than the others during normal robot operation, and a functional test may not be able to detect all the errors. The ROM

is embedded within the chip and thus not accessible from any primary inputs and outputs. Thus any testing procedure for the DKS must be able to test the ROM by exhaustive testing.

In conclusion, it can be summarized that the following two problems have been identified for testing the DKS chip:

- To test the complex sequential circuit effectively without having to resort to exhaustive testing.
- To test the embedded ROM exhaustively.

With these goals in mind, methods to solve these problems can be scrutinized to select the one which is closest to the goals and economical too. As mentioned in the last chapter, DFT techniques are being adopted widely to resolve testing problems. So, first the list of important *ad hoc* techniques can be discussed to see if any of them is applicable to the DKS chip [3,6,11], otherwise one of the structured techniques can be used.

The first *ad hoc* approach is to add extra test points on the chip to increase its controllability and observability directly. This method requires one pin for each node in the circuit that needs to be controlled and observed. Clearly, this is not practical for any chip of the size of the DKS chip.

The second approach is to partition the circuit so that each part of the circuit can be tested independently of the others. Though the DKS chip can be very easily partitioned because of its bus architecture, still it does not in any way increase its controllability and observability. Since this approach does not meet our test goals, it is not suitable for the DKS chip.

The next *ad hoc* approach is bus architecture systems which is very close to the specifications of the DKS chip which also has a bus architecture. But again the same problem arises that the DKS chip uses internal busses that are neither controllable nor observable. So this scheme is also not directly suitable for the DKS chip.

Signature analysis is a DFT technique that lies between *ad hoc* and structured techniques. It can be applied to most of the circuits and requires some modifications during the design stage. The drawback of this approach is that it lacks diagnostic ability and is thus only used to

give a go/no-go decision for a chip. This is hardly the problem for the DKS chip where the main objective is to get good diagnostic information to test the prototype. There are some other built-in and self-test techniques available but they also lack on the same grounds and are more useful for implicit testing which is carried out when the chip is on-line in a system.

One of the structured approaches to DFT may be applied for testing the DKS chip as they are general in nature and applicable to all types of circuits. The main objective of the structured DFT approaches is to increase the controllability and observability of all the storage components in the circuit. If all the storage components in the circuit can be set directly from outside the chip, the circuit is transformed into a combinational circuit for testing purposes. Thus efficient and automatic test generation and verification algorithms that have been developed for combinational circuits can be applied. The same mechanism is also used to provide access to the embedded ROMs in the circuit, thus solving both the problems for testing the DKS chip.

The structured techniques, however do have some drawbacks. One is that they serialize the test application process which further results in long test vectors and long test application time. They also require up to 20% silicon overhead on the chip as well as 4 to 6 extra pins on the chip dedicated to testing [3]. In some of the techniques, the system performance also decreases when simple latches are converted to more complex ones to make them easily testable.

The DKS chip has to be first studied to decide which of the structured techniques is most applicable to it. Referring Figure 1.1 and [5], it is observed that most of the latches in the DKS chip are connected to either of the busses A or B. Only the outputs of M1, M2, A1, and A2 latches are not connected to the bus, and M3, delay latch, and the angle registers are not connected to the bus at all. The angle registers are 12-bit wide and they can be set directly from the input port pins. All the other latches, registers and both the busses are 18 bits wide, and the output port is 16 bits wide. There are 11 control signal fields spanning 36 bits, and therefore a 36-bit control signal latch is required at the output of the control logic section.

The latches in the circuit can thus be classified into the following five categories with most of them falling into the second one:

1) The 12-bit angle registers.

2) The 18-bit latches that are connected to the bus.

3) The 18-bit latches that are not connected to the bus.

4) The 36-bit control signal latch and the 7-bit counter.

5) The 1-bit delay latch.

If any of the scan techniques is applied to the DKS chip, all of these latches behave in a similar manner as far as required design modifications are concerned. For example, if LSSD or scan path technique are used, all of these latches can be converted to polarity hold Shift Register Latches (SRL) or raceless D-type flip-flops respectively, and then interconnected to form a single long shift register. These SRLs operate as normal latches during normal system operation, but during the test mode they act as a shift register so that the data can be scanned into the register from one pin on the IC and scanned out through another pin. Thus using only these two pins for I/O and two pins for clock signals, the full problem of controllability and observability is resolved. A quick calculation shows that if either of LSSD or scan path technique is applied to the DKS chip, the resultant test vector length will be 470 bits (12-bit primary input and 458 latches), and the area overhead as will be shown in Chapter VIII would be approximately 17%.

The scan/set testing, and RAS testing techniques approach the problem in a slightly different manner. Scan/set testing also employs a shift register to scan data in and out of the chip, but the register is not in the system path. This register can set and load data from up to 64 points in the circuit. The DKS chip has 458 latches, so this scheme is not feasible if all of the latches have to be made controllable and observable. RAS on the other hand employs an addressing scheme to select each latch uniquely and can both set it and observe it just as random access memory works. This approach

requires more area overhead and more extra pins than the other scan techniques and thus is rejected.

Thus, LSSD and scan path techniques are the only ones which closely meet the specific testability requirements of the DKS chip. LSSD is always preferable to scan path because it provides a level-sensitive hazard free design and some improvements to basic LSSD are also available to closely match the requirements of the chip under test. But LSSD requires a 20% area overhead on the chip. The next logical step is to look for ways to somehow decrease the area overhead by taking advantage of the particular architecture of the DKS chip.

Earlier in the *ad hoc* techniques, it was noticed that partitioning and bus architecture system techniques came very close to solve the DKS chip's testing problem. The only reason that they could not be applied to the DKS chip was that they did not provide access to storage components in the circuit which is precisely the objective of the structured techniques. It seems quite obvious that if the busses can be somehow utilized to get access to the latches that are already connected to them (category 2), the area overhead may be reduced by a significant amount. It was this idea that prompted the development the Bus Scan Testing (BST) technique, which is applicable to all internal bus architecture systems. It takes the concepts of circuit partitioning and bus architecture systems, and modifies LSSD to provide access to the bus and other parts of the circuit. The following chapter describes BST and how it can be applied to any internal bus architecture system.

IV. BUS SCAN TESTING

Bus Scan Testing (BST) has been developed for those integrated circuits which use one or more internal busses to transfer data within the chip. Usually bus architecture systems are considered very easy to test as compared to other circuits, as explained earlier in the *ad hoc* techniques. The reason is that such systems are usually divided into modules which use the bus to transfer data among them. Thus, partitioning is inherent in these systems and partitioning a circuit always reduces the complexity of the testing process. Each module of the circuit can be tested independently of the others by putting all the other modules in high impedance state. However, in the internal bus architecture ICs like the DKS chip, the bus is itself not accessible and it is very difficult (if not impossible) to put circuit modules on the chip to high impedance states directly from the outside of the chip. BST uses a modification of the earlier scan techniques to make the bus accessible at the pins of the ICs and to control data flow on the bus inside the chip.

The philosophy behind BST is the same as that of the structured scan techniques. That is, if all the latches on a chip can be controlled and observed from the pins of the IC then the sequential circuit is transformed into a combinational one for testing purposes. Doing so enables the test engineer to use efficient algorithms and design automation tools that have been developed only for combinational circuits for generating and verifying test vectors for the circuit.

Consider a hypothetical circuit with an internal bus architecture to understand how the BST technique can be applied and how it works. Figure 4.1 shows the block diagram of such a system. The circuit block has various combinational and sequential parts, communicating through the internal bus. The control section manages the flow of data through the bus, and in

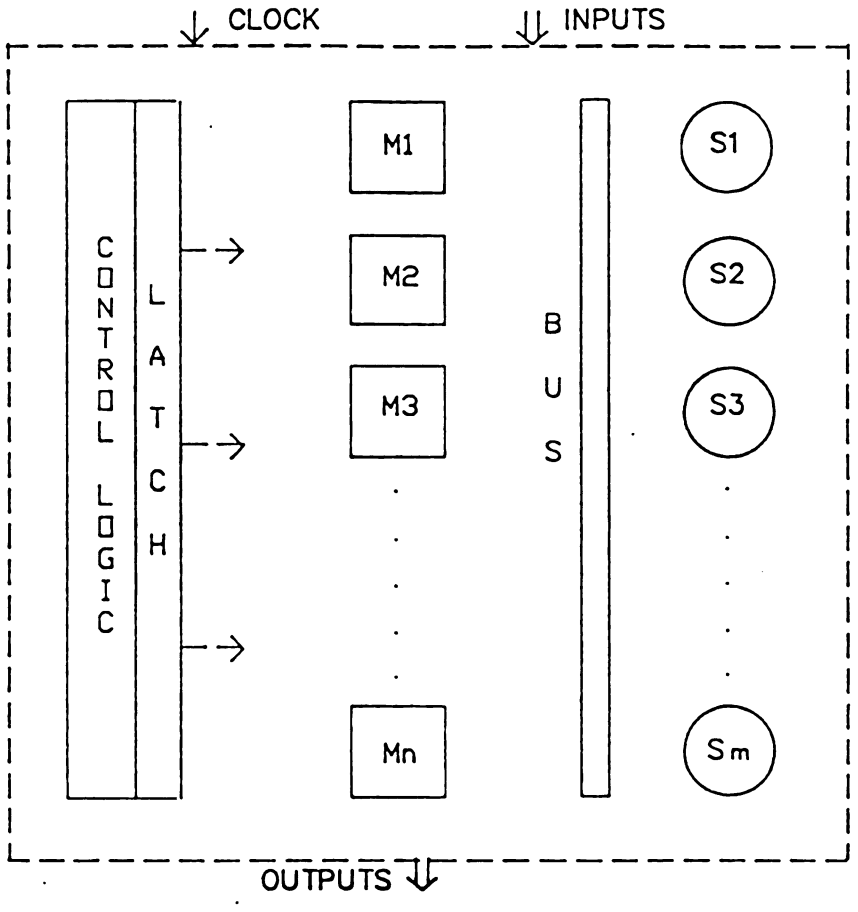

Figure 4.1. Generalized internal bus architecture system.

and out of the chip. The input and output blocks may consist of just buffers/drivers, or they may be circuits in themselves. To make the circuit as general as possible, it will be assumed here that they are in fact sequential circuits.

The following assumptions will be made for the circuit to explain the design modifications and operation of the BST technique:

- The circuit uses synchronous logic with an external two phase non-overlapped clock.
- The bus is precharged before any data transfers take place on it.
- All data transfers on the bus are deterministic.
- The control logic section provides signals for all data transfers on the bus and its output latch is operated by the system clock.

The first step to implement BST is to logically partition the circuit under test. For bus architecture circuits, this only amounts to clearly identifying the various circuit modules connected to the bus. Though no restrictions are placed on the structure of the modules, some of the modules will be easier to test than the others. The modules which consist only of a combinational network with at the most one latch at its input, or the ones which consist of only storage components, will require less test time than those where a lot of combinational as well as sequential networks are intermixed. Thus the partitions should be kept as simple and homogeneous as possible. The reason will become clear when the operation of the BST technique is explained. It is also assumed here that each module in general may have a combinational network as well as some storage components (latches) in it. Since the process to test combinational circuits is independent of the process to test the storage components, so if one of them does not exist in the module under test, then the respective part of the test can be skipped.

First, the design modifications required to implement BST on the chip will be discussed. Then the procedure for testing such a chip will be explained in detail and lastly, its performance will be evaluated.

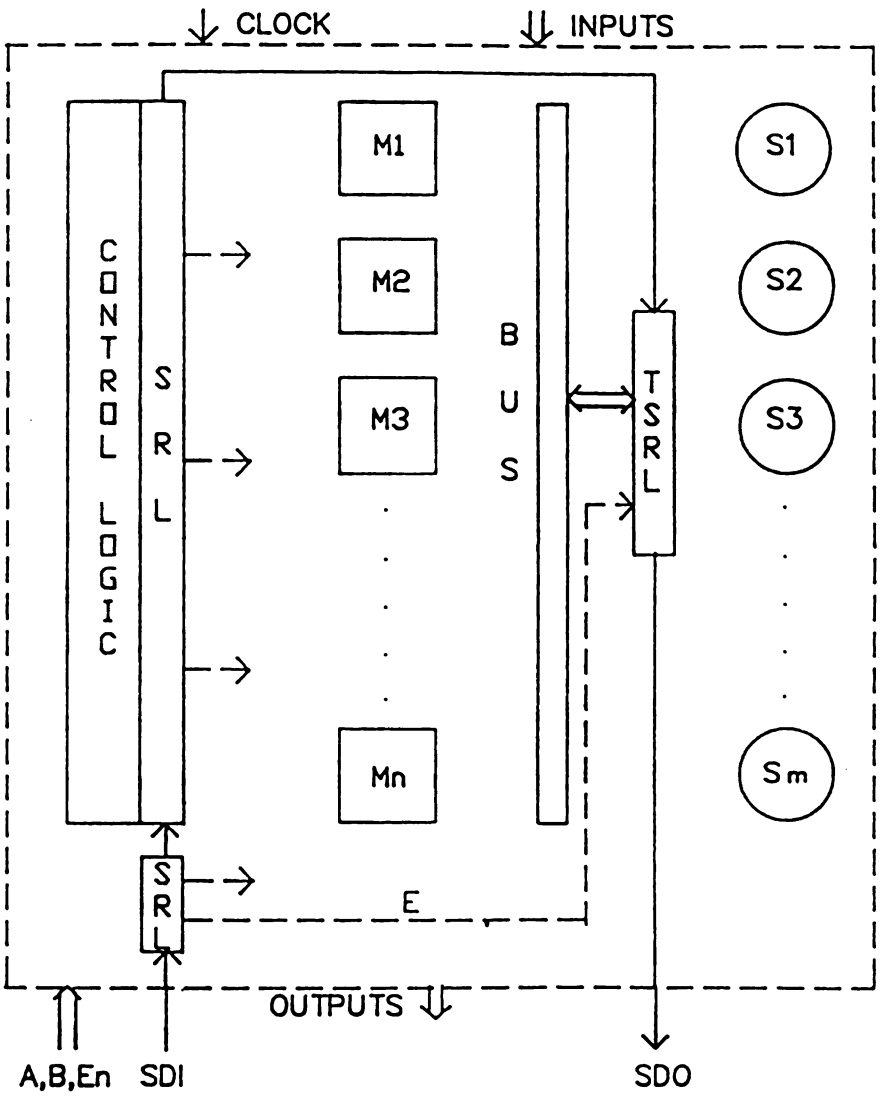

Figure 4.2. Implementation of BST.

4.1. Design Modifications

Figure 4.2 shows the proposed scheme applied to the type of circuit earlier shown in Figure 4.1. The part of the circuit which is based on the bus architecture will be considered first. Testability considerations for the rest of the circuit will be dealt with later in this section. The objective of BST is to exploit the bus to access those latches that are already connected to it. Thus individual circuitry to access each latch can be avoided, thereby decreasing the area overhead. At this point it is observed that the problem can be split into two parts. One is to provide access to the bus (and thus all latches connected to it) and other to provide means to control data flow on the bus from outside of the chip. BST requires two modifications to resolve these problems and each of them will be explained in detail now.

To get full access to the bus, a logic structure is needed that can perform the following three functions :

- Shift (scan) in and out test vectors.
- Set the bus to the test vector value.
- Load data from the bus.

It would seem that a shift register with parallel load can satisfy these requirements. However, this presents two problems. First, the outputs of each shift register cell are the same points at which it is to be loaded in parallel. Second, its output will have to be tristate since it is to be connected directly to the data bus.

So, the basic shift register should be modified to satisfy the above mentioned requirements. We chose to modify the Shift Register Latches (SRL) used in LSSD for this purpose because they are level-sensitive and thus less prone to logic hazards and race conditions. A Two-way SRL (TSRL) is proposed to provide two-way (bi-directional) access to the bus. Various logic structures were considered and three of them seemed to be quite promising. Figures 4.3 shows the structure of all three TSRL candidates.

The first structure, Figure 4.3a, can shift data from SI to SO when the two-phase non-overlapped clock (A & B) is applied to it. Its other output is through a tristate buffer so that it

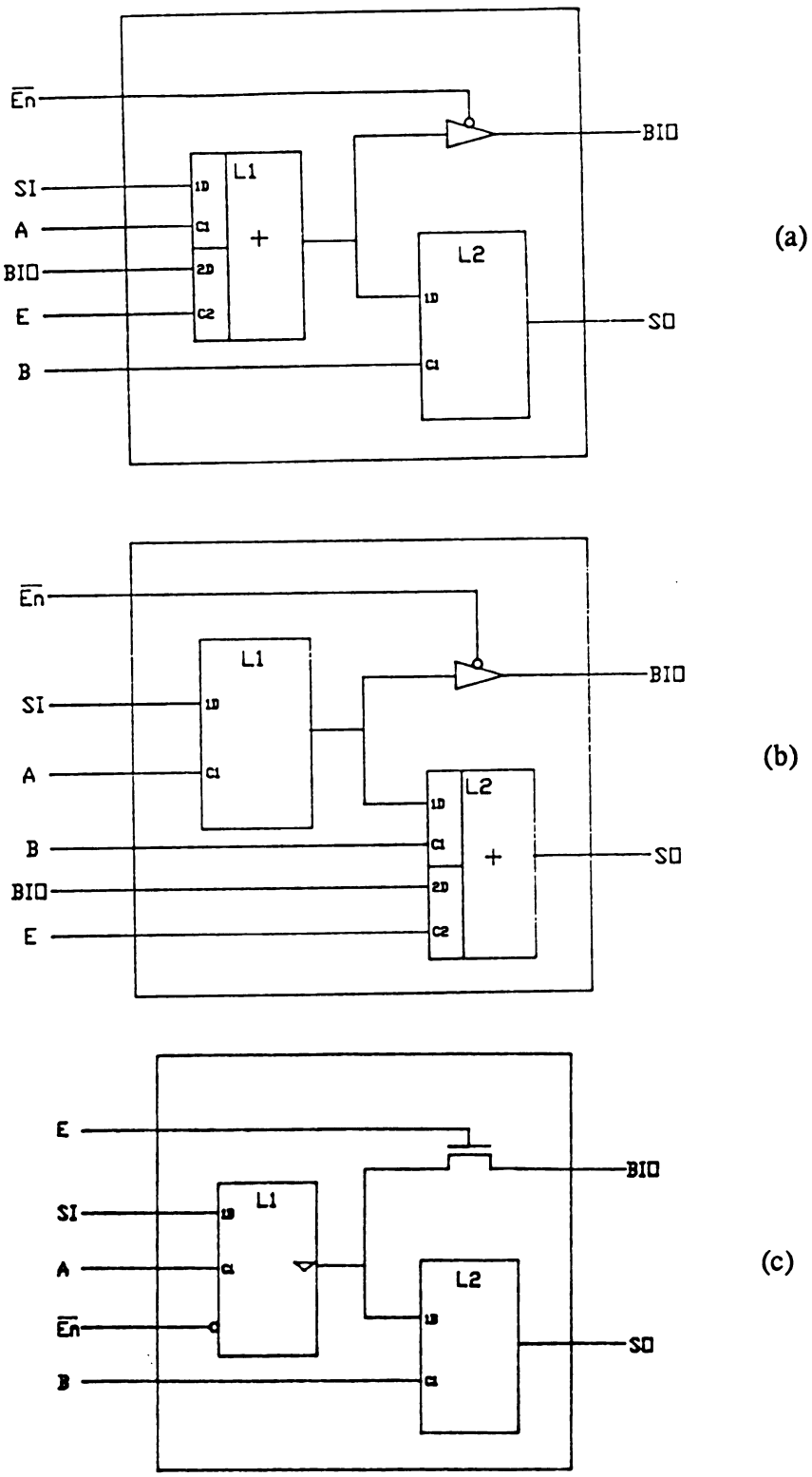

Figure 4.3. Three designs for TSRL.

can be safely connected to the bus. This buffer is controlled by the $\overline{En}$ signal which is provided at one of the pins of the IC. To load the bus, the bus is also connected to the input of the first latch and is controlled by a signal E that is provided internally by the control logic of the IC. This design suffers from two drawbacks. One is that it is susceptible to a race condition because both the input and output of L1 are connected to the bus. The second drawback is that although A and B is a set of two phase non-overlapped clocks, E and B is not, and it would have to be somehow insured that B does not go high when E is high to avoid any hazard condition.

The second design, Figure 4.3b, resembles the first one except for the fact that bus input along with the signal E are now moved to the second latch. Thus it does not suffer from the race condition but still the second drawback is there. Here, A and E should not go high at the same time.

From the above two designs it was realized that we would have to use only one bidirectional line from the TSRL to the bus. So the third design, Figure 4.3c, was developed. It satisfies all the requirements and does not suffer from any hazards. It is now explained here in detail.

The TSRL of Figure 4.3c consists of two D flip-flops, L1 and L2, connected in series. The first latch L1 is a clocked D flip-flop with a tristate output. The input to L1 is Scan-In (SI), the clock is test clock A, and the tristate output is controlled by the enable signal $\overline{En}$. The output of L1 is connected to the bus line through a transmission gate (a pass transistor in NMOS), as well as to the input of L2. The transmission gate is controlled by the complementary signals E and $\overline{E}$. This gate can thus transfer data from output of L1 to the bus and from the bus to the input of L2. L2 is a clocked D flip-flop with clock B and output Scan-Out (SO), which is not a tristate output. If A and B are two-phase non-overlapped clocks then this structure does not suffer from any of the hazard conditions.

The TSRL works in two modes: *normal mode* and *test mode*. In the normal mode of operation, the signal E is kept low so that the output BIO is in a high impedance state and thus does

not interfere with the bus signals. The circuit can function normally as if the TSRLs were not present on the chip. In the test mode, however, the TSRL has been designed to perform the previously mentioned three functions. Firstly, it can act as a cell of a shift register with input SI and output SO. The output SO of each TSRL is connected to the input SI of the next TSRL to make a shift register. In this *SHIFT* operation, E and $\overline{En}$ are both held low to disable the transmission gate and enable the tristate output of L1. A and B are two phase non-overlapped test clocks to avoid any hazard conditions. The number of shifts is equal to the number of periods for which this test clock is applied.

The other two operations of the TSRL are used to *SET* and *LOAD* the bus respectively. The *SET* operation is used to set the bus line connected to BIO from the contents of the L1. In this case $\overline{En}$ is held low to enable the output of L1, and E is held high to enable the transmission gate. Lastly, the *LOAD* operation latches the data on the bus line into L2. For this operation, $\overline{En}$ and E are both held high to disable the output of L1 and enable the transmission gate. Then a single pulse of the B clock is applied to latch the bus data into L2.

Once these TSRL cells are connected to each line of each internal bus in the circuit and interconnected to form a shift register, full access to the bus is available from the external pins as shown in Figure 4.4. A test vector can be shifted into them from the Scan Data In (SDI) pin and the bus can be set to that vector through the transmission gates. To remove a test vector, first the data from the bus can be loaded into the TSRL cells and then can be shifted out through the Scan Data Out (SDO) pin.

The second modification to implement BST is required to gain control over the data flow on the bus from outside the chip. To achieve this objective, the output latch of the control logic section is converted into polarity hold Shift Register Latch (SRL) of the type used in LSSD (Figure 2.3). These SRL cells are also arranged into a shift register and connected in the same scan path as that of the TSRL cells as shown in Figures 4.5 and 4.6. The SRL also has the same two modes of operation, but works in a little different way. In the normal mode, it just acts like a normal latch with input D and output +L1 using the system clock C. In the test

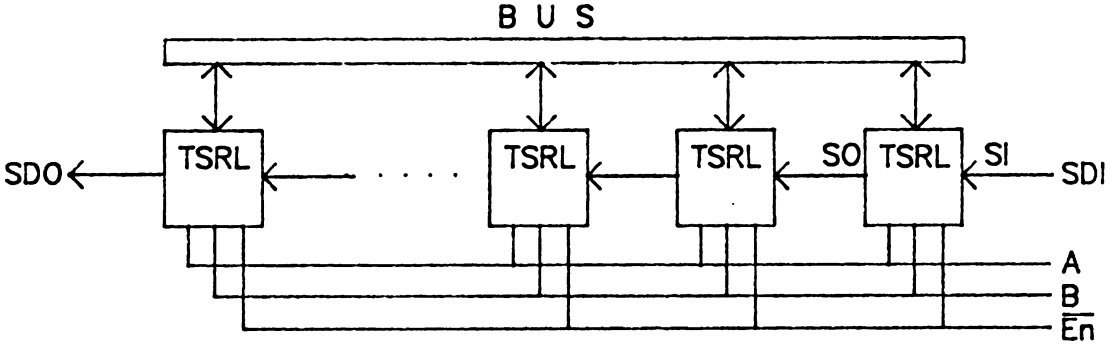

Figure 4.4. TSRL cells interconnected to form a shift register.

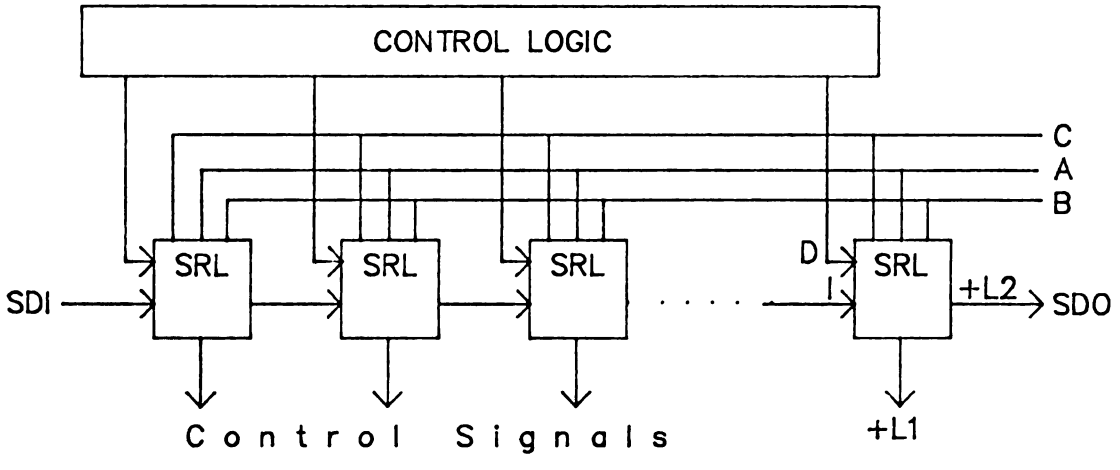

Figure 4.5. SRL cells at the output of control logic section.

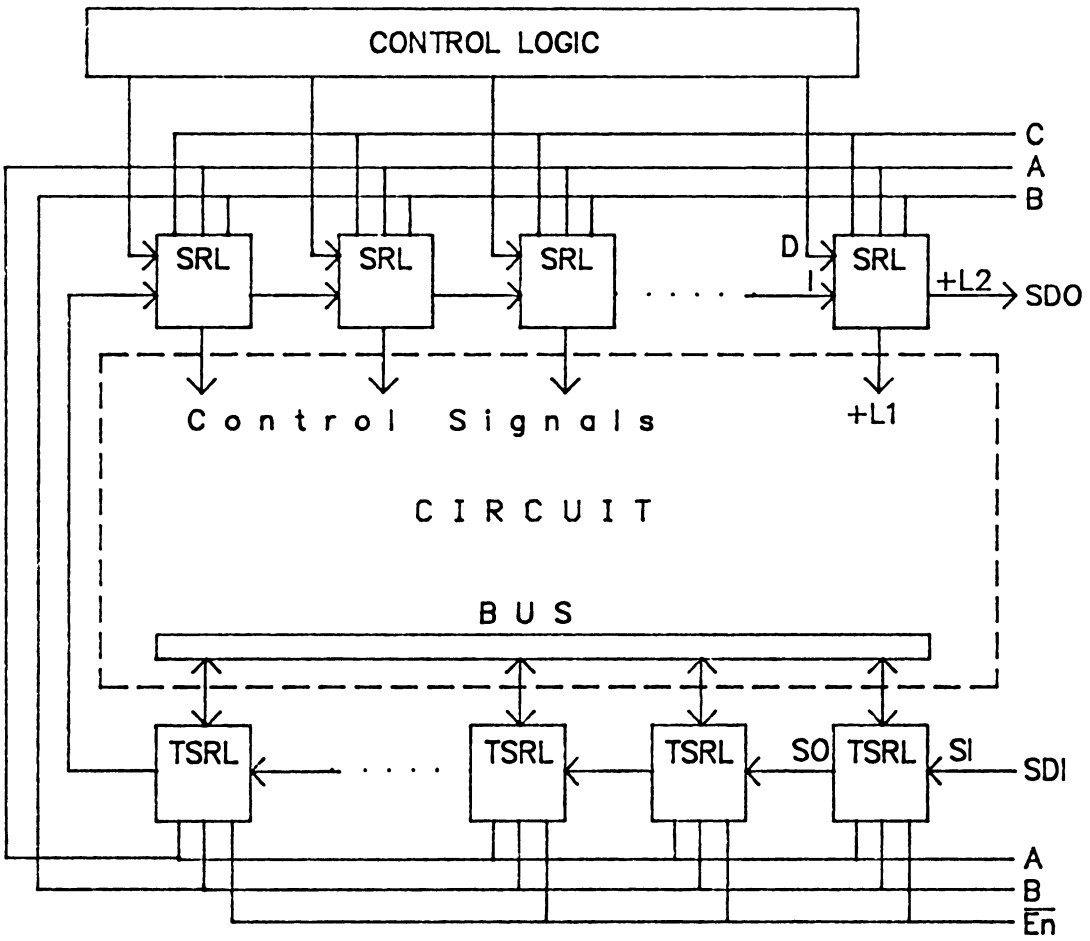

Figure 4.6. SRL and TSRL cells connected in to a single shift register.

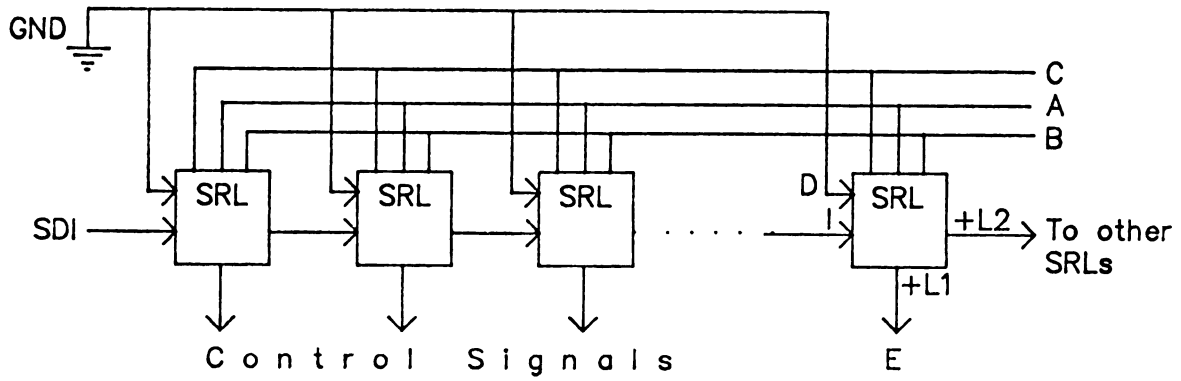

Figure 4.7. Extra SRL cells for providing additional control signals.

mode, however, it takes its input from I, uses test clocks A and B and its output +L2 is connected to the input I of the next latch in sequence.

This arrangement of an SRL at the control output makes it possible to set the control latch directly during the test mode, bypassing the control logic. When the next system clock is applied, then data transfers on the bus take place according to the current control signals in the control latch which have been set in the test mode. It should be made sure that all system operations are clocked, otherwise inadvertent operations are possible when test vectors are scanned in and out of the SRL cells.

The input of the first SRL cell in the scan path is connected to the external pin SDI and the output of the last one is connected to the input of the first TSRL cell. Then, the output of last TSRL cell is connected to the external pin SDO as shown in Figure 4.6. Test clocks A and B and the enable signal $\overline{En}$ are also provided through external pins. The other enable signal E for the transmission gates of the TSRL cells is provided internally in the following manner. It is the output +L1 of a SRL cell, whose normal input D is connected to ground (Figure 4.7), and the input I and output +L2 are connected to other SRL cells. Thus in normal circuit operation, E is always low and in the test mode it can be set to any value just like any other SRL. $\overline{E}$ is just the complement of E and thus can be taken directly from the same cell's output. This SRL cell can be visualized as an extension of the control latch and the TSRL as just another register connected to the bus. All data transfers through the bus are thus controlled by the SRL which can be set from outside.

The SRL and TSRL cells form the basic structure of the BST technique through which the bus is made fully controllable and observable. However, there may be some storage components in the circuit which are not connected (partially or fully) to the bus. The design modifications required for them and rest of the circuit which is not based on the bus architecture are similar and are described next. Again, the objective to test the rest of the circuit is to convert it to a combinational one and this is only possible if all the storage components are made fully controllable and observable. There are two methods available in BST to attack this

problem and both are described now along with their pros and cons.

One method is to connect all the inputs and outputs of such storage components to the bus. The points will have to be connected using tristate buffers, pass transistors, or transmission gates (depending on the technology), so as not to interfere with the normal operation of the bus. Control signals would have to be added to transfer data between these latches and the bus. Note that these control signals need not appear in the control logic section as they are not required for normal circuit operation. Only the output latch of the control logic needs to be extended to include these extra control signals in the same way as it was added to provide the E and $\bar{E}$ signals of the TSRL. One SRL cell will have to be added for each group of latches connected to the bus, and the input D of each cell is connected to ground permanently so that the transmission gates are off during normal circuit operation. Thus, the latches that were not connected to the bus earlier can now transfer data to and from the bus during the test mode.

The second method of providing access to the registers is to convert them to SRL cells and connect them in the scan path exactly as is done in LSSD. Thus they can be set and scanned using the LSSD approach as such. Therefore, LSSD and BST can co-exist on the same chip, BST for testing the bus architecture part of the circuit and LSSD for rest of the circuit.

The second method of using LSSD requires more area in most cases than the first approach of connecting the latches to the bus. It also lengthens the size of the test vectors. But in some cases, using LSSD may reduce the total test time, if one part of the circuit is quite independent of that part of the circuit which is connected to the bus. This would allow some tests to be done in parallel which is otherwise not possible with BST. LSSD will also have to be used if the width of the latch under question is not the same as the bus-width.

4.2. Operation

The testing process is generally divided into three parts: *test generation, test verification, and test application.* Tests can be generated automatically on computers for all the combinational networks in the circuit using one of the available algorithms (like the D-algorithm, PODEM-X, adaptive random test generation, or compiled code Boolean simulation). Test verification is a little more complicated, but mostly a single stuck-at fault model is assumed sufficient to find the fault coverage [9,36] for the test vectors generated earlier (fault simulation). A fault coverage of 95 to 98% is considered sufficient to test any circuit economically [9]. Lastly, these test vectors are applied to the circuit. This is a $3+n$ step process for the BST technique (where n is the number of partitions of the circuit), and is now explained in detail.

1. Test the SRL and TSRL cells: Scanning in a set of vectors and then scanning them out tests all the SRL and TSRL cells in the scan path. Flush test and shift test are usually considered sufficient to test shift registers [9]. In flush test, first a single one in a set of zeroes is passed through the register followed by a zero in a set of ones. In shift test, the sequence 00110011... is shifted through the register. The *SHIFT* operation of the TSRL is used for this purpose. These tests don't test the transmission gates in the TSRL cells, which are tested along with the bus in the next step.

2. Test the internal bus(ses): The test vector is scanned into the TSRL using the *SHIFT* operation. Then a *SET* and a *LOAD* operation is performed consecutively in the same clock cycle to test the bus. The latched data is then scanned out and compared with the input that was applied. If the vectors are not the same, then there is a fault on the corresponding bus lines, transmission gates, or any one of the modules connected to them. Faults on busses are usually very difficult to locate and fault diagnosis in this case is limited to identifying the bus lines on which faults exist.

3. Test the control logic: The control logic is usually either hardwired, or microcoded in a ROM with a counter at its address lines. To test the control logic, the circuit is first RESET to initialize the system, and then one system clock is applied. This latches the first control

sequence into the control logic's SRL. This sequence can be shifted out through the SDO pin by the *SHIFT* operation. Another system clock will latch the second control sequence into the SRL, which can be then similarly observed. This process continues until all states of the control logic have been verified and the system returns to its original state. If the control logic is microcoded, the counter's latch cells will also have to be converted to SRL cells, included in the scan path, and tested with the other SRL cells in step 1.

4. Test the circuit modules: Since the circuit has already been partitioned into n modules, this step has to be repeated n times to test each of them separately. The test vectors required to test each module are generated ahead of time. The procedure to test any one part of the circuit involves the testing of the latches in that module first, and then the application of test vectors to those latches and primary inputs to test the combinational part of the module. The following steps are required to test each latch in that module:

a) Shift the test vector into the TSRL shift register and the control vector into the SRL shift register of the control latch (*SHIFT* operation).

b) Apply one system clock. The control signals are set such that they transfer data from TSRL to the latch under test during the first phase of the clock and from the latch to the TSRL during the second phase.

c) Shift the TSRL's data out through the SDO pin (*SHIFT* operation). (Actually this step can be combined with step a such that when the data in the TSRL is being shifted out, the next test vector can be shifted in at the same time.)

After all the input and output latches in the module have been tested, the combinational network can then be tested using the following steps :

a) Set all the latches and primary inputs that directly effect the combinational network under test. Each latch/register can be set by a *SHIFT* operation followed by a *SET* operation with appropriate control signals. If the module under test is a fully combinational one, then the input to the network can be applied directly from the TSRLs.

b) Apply one system clock to do a single operation of that module. The output of the module may go to an output latch or directly to the TSRL through the bus. If the output goes to an output latch then control signals to transfer data from that latch to the TSRL are scanned in and a *LOAD* operation performed. It is for this reason that the SRLs are kept in the scan path before the TSRLs, so that if the TSRLs are not required to be set to any particular value it is then possible to scan in just the control vector itself.

c) The result which is now latched in the TSRL is scanned out. Again this step can be merged with step a and the next test vector can be scanned in simultaneously with this step.

The testing of that part of the circuit which is based on the internal bus architecture is now complete. The procedure to test the rest of the circuit depends on which of the two options was used during the testability design of that circuit. If all the latches in that part of the circuit were connected to the bus then that circuit acts just like one more module in the bus architecture and is tested accordingly, as explained in the previous few paragraphs.

On the other hand, if the second option is used then all the latches are converted to SRLs and the LSSD technique can be used. The testing of these parts can thus be carried out using LSSD in parallel with the testing of the other parts of the circuit using BST. When the test vector is scanned in, it includes the values for all the SRLs as well as the TSRLs. Next, when a system clock is applied, the data from the SRL cells is applied to the combinational circuit and latched into its output SRL cells. This data can be shifted out and verified for correctness.

One more thing needs to be pointed out before completing the discussion of BST. If there is more than one internal bus in the circuit, BST can still be applied with the same effectiveness. The other busses may be internal or external and controlled from inside or from outside the chip, as long as data transfers on them are deterministic. Figure 4.8 shows an example of how BST can be implemented on a circuit with two internal busses. It can be easily extended to any number of busses. The same E signal can be used for all of them to transfer data from and to the TSRLs at the same time.

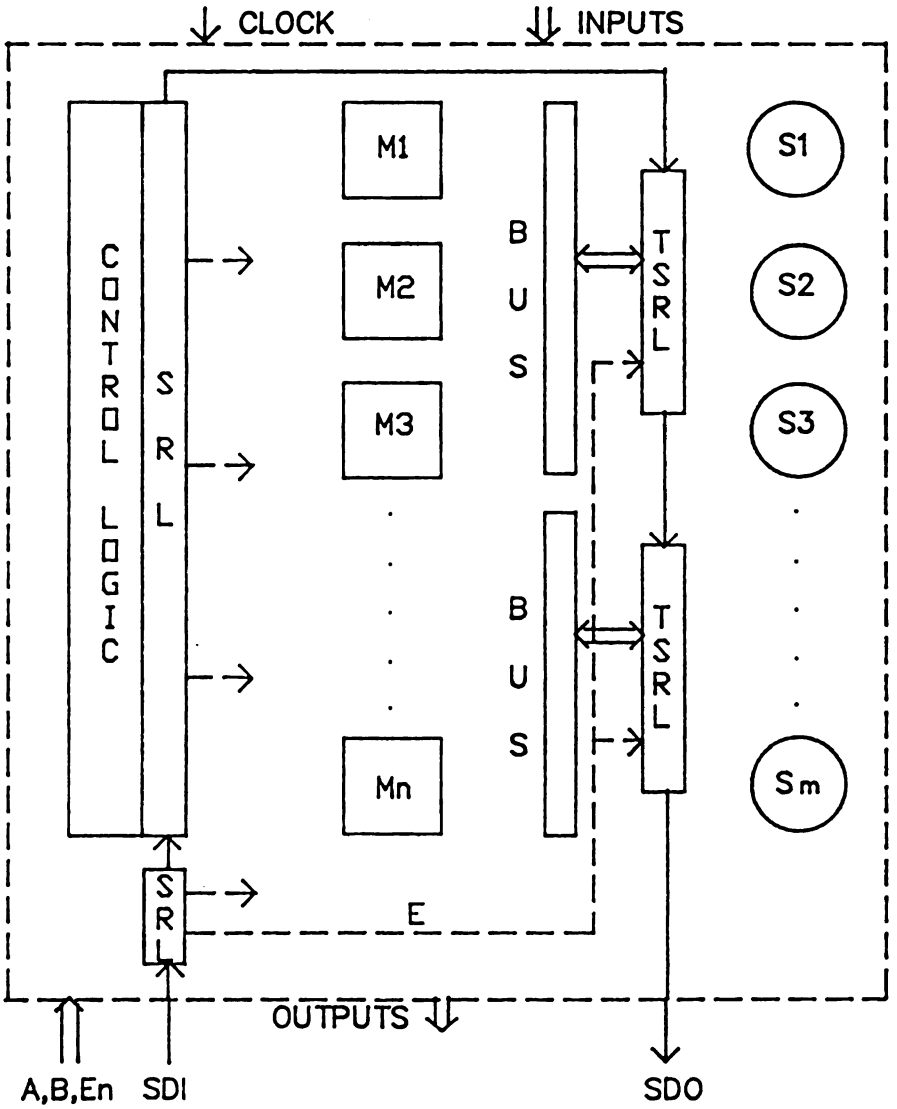

Figure 4.8 BST implemented on circuit with two internal busses.

4.3. Performance

The reduction of silicon overhead from LSSD to BST can be quickly figured out by using a simple example. If a circuit has 10 registers of 16 bits each connected to the bus, and the LSSD approach is used, it would be required to convert all the 160 latches involved to SRL cells. In BST, however, only 16 TSRL cells need to be added to the circuit and the control latch converted to SRL cells to access the the 16-bit bus and thus all of the 10 registers. In fact, the number of TSRL cells will remain constant at 16 irrespective of the number of latches connected to the bus. Only the control signal latches have to be converted to SRLs. Thus, the more the number of latches connected to the bus, the lesser the proportional area overhead. This is unlike all other scan techniques, where the area overhead increases in direct proportion with the number of latches in the circuit.

The second advantage of using BST over other scan techniques is that there is no performance degradation during normal operation of the circuit. To implement LSSD or scan path in a circuit, all latches in the system have to be modified resulting in an increased time delay for each of them. In BST, no such modifications are required thus avoiding any performance degradation for those modules in the circuit that are connected to any of the busses.

Another advantage of using BST is that it results in shorter test vectors than either of LSSD and scan path techniques. This directly follows from the fact that there are less SRL and TSRL cells in BST than the others. Thus at any time, fewer of these cells have to be set in BST leading to shorter test vector lengths.

BST, however, does suffer from one disadvantage when compared to the other techniques. Since many latches are connected to a bus, only one of them can be set from the TSRLs at one time. Further, the structure of BST had been optimized to test one circuit module at a time. Both of these characteristics of BST lead to longer test application times than would be required by LSSD for the same circuit. LSSD can set all the latches in the circuit at one time and can thus test most of the circuit modules in parallel.

In conclusion, it can be adjudjed that the advantages of reduced area overhead, no performance degradation, and shorter test vectors because of partitioning outweigh the only disadvantage of a little longer test application time. Thus BST should be used for testing all internal bus architecture systems. The area overhead in BST can be further reduced by converting one of the system latches into TSRL cells, instead of adding extra TSRL cells at the expense of a more complicated design.

V. DKS CHIP TESTABILITY AND INPUT/OUTPUT DESIGN

The testability considerations for the DKS chip were presented in chapter III and a decision was justified that one of the scan techniques must be used to improve its testability. Then, a new technique *Bus Scan Testing* was presented in Chapter IV. Since the DKS chip has two internal busses, the advantages of using BST over the other scan techniques make it the obvious choice. This chapter explains what modifications are required in the design of the DKS chip for the purpose of incorporating BST. The input/output design is then presented which completes the design of the DKS chip in all respects.

5.1. Design modifications

The basic strength of the bus architecture (and thus BST) lies in their ability to be quickly partitioned. One glance at the DKS chip's block diagram (Figure 1.1) gives clear evidence to that effect. It is observed that most of the blocks are connected to the bus and are either purely sequential or purely combinational in nature. The only sequential circuits present on the chip are the latches and registers and the most complicated combinational networks are the two stage multiplier and the adder. Thus each block in Figure 1.1 can be considered a module and each of them can be tested independently.

Now that the DKS chip is partitioned, each module or a group of modules can be analyzed one by one to take appropriate steps to make each of them easily testable. Since the main objective of the BST approach is to make all the storage elements in the circuit controllable and observable, design modifications are required only for such modules which have one or more storage elements. The latches and registers in the DKS chip have been classified into

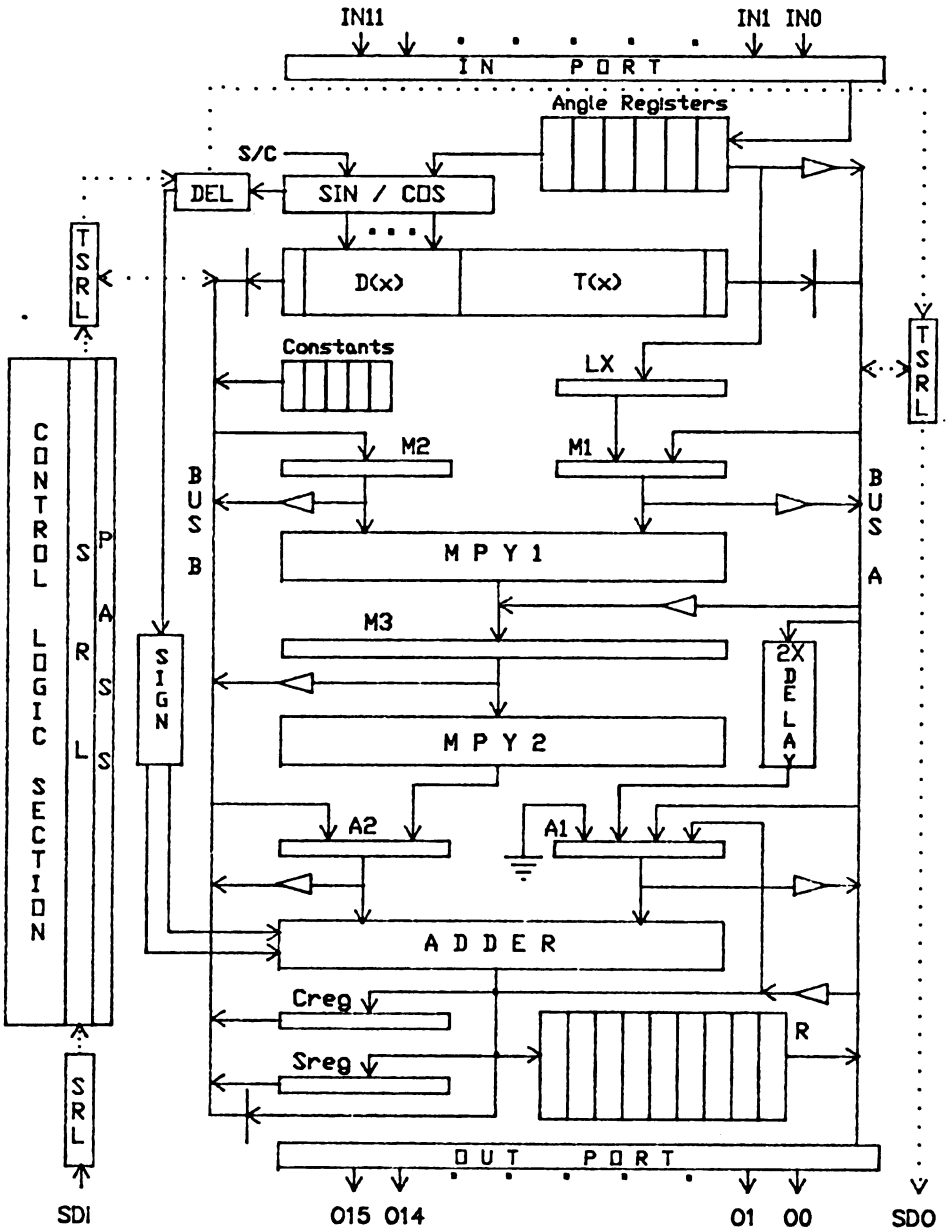

Figure 5.1 BST implemented on the DKS chip.

be discussed now.

The latches of category 2 and 4 are easily dealt with by using the basic BST structure. Since the latches of category 2 are already connected to the bus and control signals which transfer data among them through the busses already exist, they meet exactly the specifications of the BST design. Thus one TSRL cell is connected to each bus line of both the busses, which are then interconnected to form a shift register. Next, all the output latches of the control logic section and the counter (category 4) are converted to SRL cells.

One SRL cell is added to provide the E and $\overline{E}$ signals to the TSRL cells. The input D of this SRL is connected to ground so that during normal circuit operation, E is always low thus keeping the output of TSRLs in a high impedance state. Finally all of these SRL and TSRL cells are interconnected to form a single shift register, the input of which is from the external pin SDI and output to another pin SDO.

Most of the latches of category 2 have either their input or their output connected to the bus, but not both. To make them fully controllable and observable, they should have their inputs as well as outputs connected to the bus. Thus the outputs of M1, M2, A1, and A2 latches and the output of the adder which feeds Creg, Sreg and the register file are connected to the bus through transmission gates as shown in Figure 5.1. One control signal will be required for each set of transmission gates connected to the bus. So five SRL cells are added in the scan path to control the 5 sets of transmission gates with their D-inputs connected to ground to disable the transmission gates during normal circuit operation.

The remaining latches of categories 1, 3, and 5 are those which are not connected to the bus. Since there are two options available for the purpose, each option will be assessed for each category of latch separately. The first option is to connect both their inputs and outputs to the bus and the second one is to use the LSSD approach which converts each latch to an SRL. LSSD will be preferred only if the latches in question are quite independent of the part of the circuit based on the bus structure or if the width of the latch is greater or much less than the bus width.

The angle registers of category 1 are 12-bits wide and their input is directly connected to the input pins of the IC. Thus they are already controllable so only their output needs to be made observable. If LSSD is used 72 (12×6) of these latches will have to be converted to SRL cells and the test vector length will also increase by the same amount. On the other hand, if their 12-bit output is connected to the least significant 12 bits of one of the two 18-bit busses, the overhead will be 12 transmission gates and one SRL cell, and the test vector length will only increase by one bit. The most significant 6 bits of the bus can be ignored when the angle registers are scanned out through the bus. Thus this alternative is much more economical than using LSSD in this case. So 12 transmission gates are added to connect the output of the angle registers to bus A and one SRL cell is added in the scan path to control these transmission gates with its D-input connected to ground.

The only latch that falls into category 3 is M3. Since it is 18-bits wide and is in middle of the part of the circuit based on the bus architecture, there is no reason to use LSSD. So 18 transmission gates are used to connect the input of M3 to bus A and the same number to connect its output to bus B. Two SRL cells are added in the scan path to control the two set of transmission gates with their D-inputs connected to ground.

The last category (5) contains just a single 1-bit delay latch. Since its 1-bit width is far less than the 18-bit bus width, it is best to use LSSD in this case. The delay latch only needs to be converted to an SRL cell and then connected in the scan path with other SRL and TSRL cells. The resultant increase in test vector length is only one bit.

This concludes the design modifications required on the DKS chip to be able to use the BST technique for testing it. The various control signals that have been added to the chip in the process are summarized in Table 5.1. The next section now describes the I/O design of the DKS chip.

Table 5.1. Control signals added to the DKS chip for implementing BST.

No.	Control Signal	Function
1	EXGt	Enable transmission gate of TSRL
2	MO	Connect M1 to bus A and M2 to bus B
3	AO	Connect A1 to bus A and A2 to bus B
4	M3I	Connect bus A to input of M3
5	M3O	Connect output of M3 to bus B
6	Av	Connect angle registers to bus A
7	AT	Connect bus A to register file

Table 5.2. I/O pins required for testing the DKS chip.

No.	I/O pin	Normal Connection
1	A clock	Ground
2	B clock	Ground
3	En	+5V
4	SDI	Ground
5	SDO	Open

5.2. Input/Output Design

When the DKS chip design was presented in [5], the I/O design was not finalized because it was assumed that DFT will have an impact on its design. Now that the design of the DKS chip is complete in all other respects, the I/O design can be finalized. The I/O signals can be broadly classified into two categories: test I/O and operational I/O. The five test I/O pins required to implement BST have already been described in the previous section. These pins are only used during the testing process. Table 5.2 summarizes the various test pins and at what values to tie them during normal system operation of the circuit.

The operational I/O signals are used in both the normal as well as test mode of the circuit. They include all power lines, control signals, as well as data I/O lines. Following is a brief description of all the pins required on the IC.

1. V_{cc} and *Ground*: These two pins provide power to all the chip circuitry. Since TTL compatible CMOS technology is being used for the IC, the V_{cc} pin has to be connected to the positive terminal of a +5V power supply and *Ground* pin to its negative terminal.
2. ϕ_1 and ϕ_2: The DKS chip requires an external two-phase non-overlapped clock which is provided through these two pins.
3. *RESET*: This pin is connected to the counter of the control logic section and to all other latches and registers in the circuit which have a *CLEAR* signal available. It is an active high signal. Externally, it can be connected to the controlling computer and/or to a small power-on reset circuitry.
4. *Chip Select* ($\overline{CS}$): The I/O of DKS chip is designed so that it can serve as a part of a full robot control system. Most computerized systems use data and address busses to transfer data from the CPU to all the other chips in the system. When a chip is selected by the CPU, the address is decoded and then an appropriate chip is selected by the $\overline{CS}$ signal. It is an active low signal, thus it allows data transfers to and from the system bus only if this signal is held low.

5. $\overline{SILE}$ and *DIN*0 – *DIN*11: These 13 pins work together to input data from the system into the angle registers of the chip. Data input *DIN*0 to *DIN*11 is the 12-bit input port of the chip which is connected to the input angle register-5. If $\overline{SILE}$ (Shift and Input Latch Enable) and $\overline{CS}$ are both low, then during phase one of the system clock, data from all angle registers is shifted to the next lower register in sequence and data at the input port is latched into angle register-5. Thus all the six angle registers can be set in six system clocks by applying one angle in each clock cycle in sequence from 0 to 5.

6. $\overline{START}$: Once all the angle registers are set, then calculation of the DKS can begin. The changing of this signal $\overline{START}$ from high to low signals the counter of control logic to start counting until all the 73 states required to calculate the DKS are over. If at the end of 73 states, $\overline{START}$ is still low, the calculation starts all over again otherwise the counter stops and the chip does nothing until some other external signal becomes active.

7. $\overline{SOUT}$, *DVAL*, and *DOUT*0 – *DOUT*15: These pins provide the final values from the register file to the output port. The *DOUT*0 to *DOUT*15 (Data Output) pins provide the 16-bit tristate output that can be connected to the system data bus. When $\overline{SOUT}$ (Shift Out) and $\overline{CS}$ are both low then during phase one of the system clock data from each register in the register file is shifted to the next higher register and the contents of register-8 are provided at the output pins. The *DVAL* (Data Valid) pin is then taken high at the end of phase 1 of the system clock to signal the system that data on the bus is valid data. *DVAL* is kept high until either $\overline{CS}$ goes high or until the beginning of the next clock cycle.

8. *HALT*: This is an emergency signal that stops all operations on the chip by stopping the control logic state counter. The data in the chip is undefined after this signal is given so that the only valid operations after a *HALT* signal are a *RESET* or a $\overline{SILE}$ with $\overline{CS}$ and *DIN*.

Table 5.3 summarizes all the operational I/O signals for the DKS chip. There are a total of 38 pins including the 12-bit input data and the 16-bit output data pins. The number of pins required to implement BST is five taking the grand total to 43. This number is well within the capacity of the inexpensive Dual-in Line Packages (DIPs).

Table 5.3. I/O pins required for the DKS chip.

I/O Pin	Function
Vcc	+5V Power supply
GND	Power supply ground
ϕ_1	phase 1 clock
ϕ_2	phase 2 clock
RESET	Reset all latches
$\overline{CS}$	Chip Select
$\overline{SILE}$	Shift and input latch enable
DIN0 - DIN11	12 bit angle input
$\overline{START}$	Start DKS claculation
DOUT0 - DOUT15	16 bit data output
$\overline{SOUT}$	Shift out data
DVAL	Data valid at DOUT pins
HALT	Emergency stop

VI. TEST GENERATION

Test generation involves finding a set of input vectors for the circuit under test which allow for the detection of faults by observing the primary outputs of a circuit. One such set of vectors is the complete set of all possible input vectors of the circuit under test (Exhaustive testing). Though such a test would test the circuit for all possible faults, it requires a very long time to run, especially for complex sequential circuits, and thus, is not practical.

Success of a testing process is generally measured in terms of the percent fault coverage against a pre-defined set of faults. Many different types of faults can occur on an IC. Stuck-at-zero, stuck-at-one, bridging faults, open circuit faults and CMOS-stuck-open faults are just a few categories. There may also be multiple faults in the circuit. Usually a single stuck-at fault model is assumed and then test vectors generated for the model. It has been shown that this model is able to detect many of the bridging, open circuit, and multiple faults too, and is thus considered sufficient for the testing of most circuits [3].

It is here, during the test generation process, that the importance of the DFT techniques is realized. All the structured DFT techniques are aimed at reducing sequential circuits to combinational ones for testing purposes. Further, the circuit can be partitioned along functional lines and test vectors can be generated for each of the partitions independent of the others. This becomes possible with the DFT techniques because they provide full access to all the partitions in the circuit.

The goal of test generation for the DKS chip is severalfold and in the order of importance is:

1. chip design verification;
2. fault detection using fault simulation;

3. manufacturing testing.

Only the first two are of our interest here because the chip is being initially fabricated as a prototype. If manufacturing testing is to be done eventually, the test vectors developed here can be applied and more added to increase the fault coverage if necessary.

There are two types of test strategies that can be used to test a circuit. One is *structural testing* and the other is *behavioral (or functional) testing*. Structural testing requires a low level (at least gate level) description of the circuit and a fault model. It is used to verify the physical device against the logic design, which is assumed to be correct. Thus a design verification is not possible by this method.

Behavioral testing, on the other hand, tests the functional behavior of the circuit. Various device descriptions above the gate level can be used to test a circuit. It is usually used when a gate level description of the circuit is not available.

The test goals for the DKS chip at this stage closely match the requirements of behavioral testing. One reason is that gate level description of the circuit is not available and second is that design verification is required for it. Thus if test vectors could be generated to test the functionality of the circuit, then they can be used independently of the implementation.

6.1. Test Generation Methods

There are four methods available to generate tests for a circuit: manual test generation, universal tests, pseudo-random test patterns, and algorithmic methods. Manual test generation is only possible for very small circuits or some special components like RAMs. It is usually based on the intuition of the designer with no guarantee for success.

Universal tests provide a general set of vectors to test all or a subset of any circuit without any calculation or simulation. Exhaustive testing lies under this category, which is obviously impractical for large circuits. Some other test sets have also been prepared for PLAs and a few specific types of circuits. However, at this time no universal tests really exist that can handle all circuits.

Pseudo-random testing technique uses random numbers as test vectors for the circuit. Still, a fault model and description of the circuit is required to calculate the fault coverage. Probabilistic methods are also being used nowadays to find the expected fault coverage using random patterns [29,30,31]. It is very difficult to do a design verification using this method.

The most widely used test generation method is the algorithmic approach. This approach is mostly based on path sensitization techniques whose aim is to provide a path from the primary inputs to all points in the circuit and then from there to the primary outputs. D-algorithm [28,32], PODEM [33], and compiled code Boolean simulation [34] are a few of the algorithms developed for this purpose. Most of these algorithms are deterministic and can find a test vector if it exists. However, even such techniques become very time consuming for large circuits and thus are not very attractive for VLSI.

Since the DKS chip has been partitioned into modules and all the modules can be tested independently, the type of test generation method used for each can be selected individually. The next section deals with the selection of the test process as well as actual test vector generation for each type of module in the chip.

6.2. Test Vector Generation

The various circuit modules on the DKS chip can be classified into the following six categories:

- SRL and TSRL cells;
- control logic ROM;
- busses A and B;
- ROM to store table values;
- storage latches (LX, M1, M2, M3, A1, A2, Creg, Sreg, and 2-stage delay);
- shift registers (angle registers and register file);
- combinational circuits (MPY1, MPY2, and Adder).

The test patterns required to test the SRL and TSRL cells are specified in the BST technique. The same two tests (shift test and flush test) can be used here without any further changes. Similarly the procedure to test the control logic ROM and the internal busses has been completely specified in the BST procedure. Thus no test vectors are required to be generated for the first three categories of modules.

The next category of modules is the ROM to store table values. There is no other method to test a ROM other than by exhaustive testing. Moreover, it can be easily used now since the input and output of ROM are both accessible through the TSRL cells. Since the input angles are 12 bits each, 2^{12} (4096) test patterns are required to verify all the ROM contents along with the SIN/COS combinational circuitry at its input. Five test vectors are also required to verify the contents of the ROM used to store the five constants.

Test vectors for all the latches are required next. Since the latches are only used to store 18 bit data and are not intended for any other special operation, they can be simply tested by using the following four vectors for each latch. First is a set of all zeroes, then a set of all ones, then a sequence of 00110011..., and lastly a sequence of 11001100... . These vectors can detect all stuck-at faults as well as most of the shorting and bridging faults in the latches.

The six angle registers and the nine registers in the register file constitute the sixth category. As has been already identified in Chapter IV, shift tests and flush tests have been found sufficient to test all shift registers. Thus the same tests that are used to test SRL and TSRL cells can be used here also.

The last category of modules is the combinational circuits. The 18-bit Baugh-Wooley multiplier and the 18-bit Ripple Carry Adder (RCA) are the only two combinational circuits on the chip. Test vector generation of these circuits is not as easy as has been for the other modules. Thus various test generation methods are discussed in the next section and one of them chosen to test the combinational circuits.

6.3. Test Generation for Combinational Circuits

The first and the easiest method would be to do an exhaustive test of each combinational circuit. Since each of these circuits operates on two 18-bit operands, the number of test vectors required to exhaustively test them will be 2^{36} each. Since it requires approximately 100 clock cycles to apply one test vector to each circuit and assuming a clock speed of 10 MHz (since the DKS chip has been designed for 1 to 14 MHz), the time required to test each of them is about 8 days. Such a long time to test any circuit is totally unacceptable, so this method is discarded.

The next method is to generate test vectors manually. This is also not suitable considering the size of the 18-by-18 multiplier and adder. Test generation by hand usually relies on intuition of the designer who observes the truth table of the circuit and decides what tests to use.

Another method is to use pseudo-random test patterns. Though random patterns have been used to test many circuits with good results, their disadvantage is that they do not provide ample capability for the type of design verification which is important for the DKS chip as a prototype.

The last method is to use one of the algorithmic methods to generate test vectors for each of the modules. Computerized test generation programs are usually used for all large combinational circuits. Since both the multiplier and the adder have very regular structures, the extended D-algorithm as proposed in [35] was chosen to generate test vectors for the functional tests. The extended D-algorithm was chosen because it is a simple extension of the D-algorithm, which is the most widely used algorithm for generating test vectors for combinational circuits.

The basic components used for the adder and the multiplier are inverters, AND gates, XOR gates, Full Adders (FA), and 2-to-1 multiplexers. The D-propagation tables for these components are first made as shown in Tables 6.1 to 6.5. The tables show how a logical value D at one of the component's inputs will be reflected at its output. The aim of D-algorithm is to provide test vectors so that each node in the circuit can reach the value D from its primary

Table 6.1. D-propagation table for an Inverter.

Input	Output
0	1
1	0
D	$\overline{D}$
$\overline{D}$	D

Table 6.2. D-propagation table for an AND Gate.

AND	0	1	D	$\overline{D}$	X
0	0	0	0	0	0
1	0	1	D	$\overline{D}$	X
D	0	D	D	0	X
$\overline{D}$	0	$\overline{D}$	0	$\overline{D}$	X
X	0	X	X	X	X

Table 6.3. D-propagation table for an XOR Gate.

XOR	0	1	D	$\overline{D}$	X
0	0	1	D	$\overline{D}$	X
1	1	0	$\overline{D}$	D	X
D	D	$\overline{D}$	0	1	X
$\overline{D}$	$\overline{D}$	D	1	0	X
X	X	X	X	X	X

Table 6.4. Partial D-propagation table for a Full Adder.

A	B	Cin	Sum	Cout
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
0	0	1	1	0
1	1	0	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1
D	0	0	D	0
0	0	D	D	0
D	0	1	$\overline{D}$	D
D	1	0	$\overline{D}$	D
D	1	1	D	1
D	D	0	0	D
D	D	1	1	D
D	0	D	0	D
D	1	D	1	D
D	D	D	D	D

Table 6.5. D-propagation table for a 2-to-1 multiplexer.

A	B	C	OUT
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1
D	X	0	D
X	D	1	D
0	0	D	0
0	1	D	D
1	0	D	$\overline{D}$
1	1	D	1

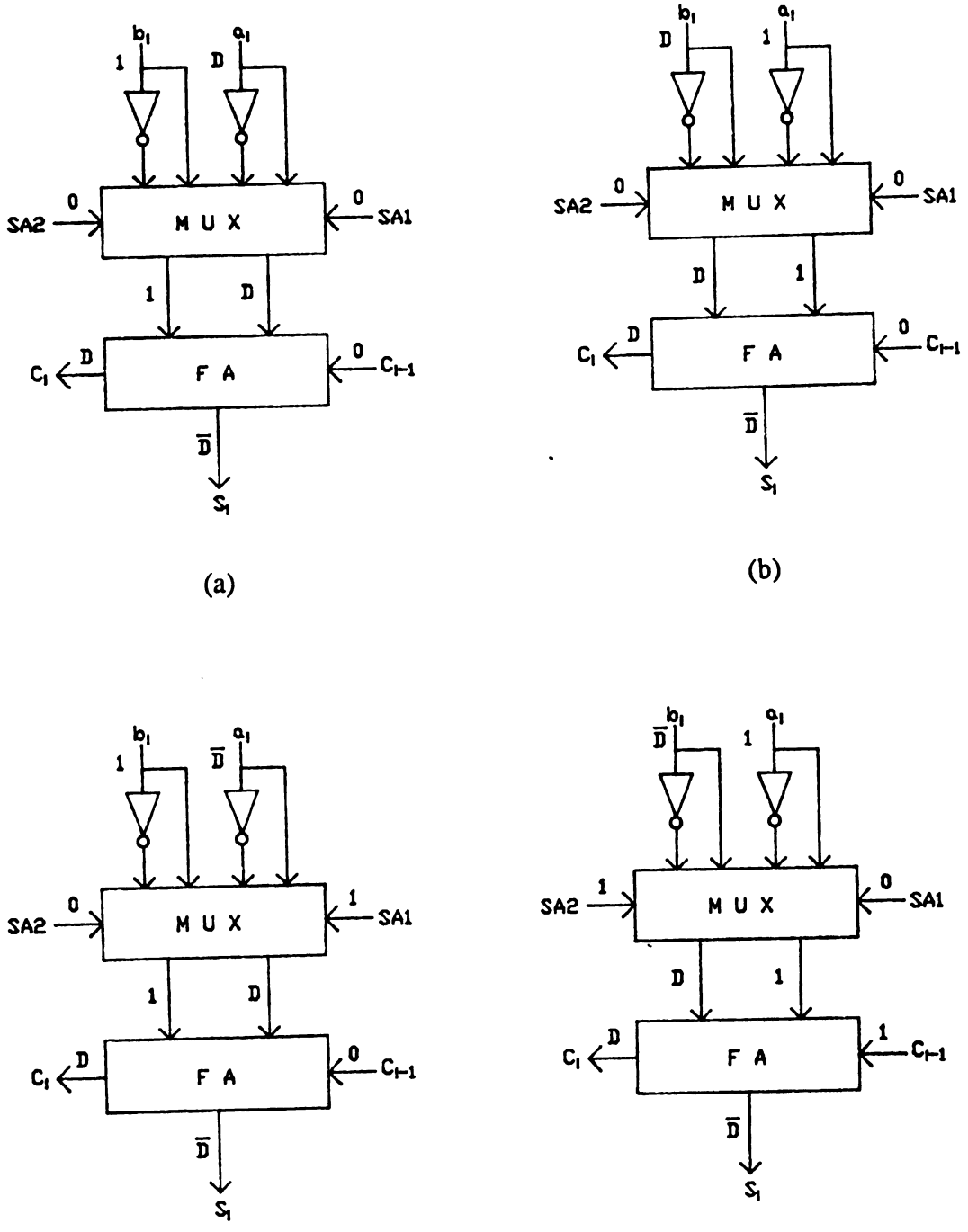

Figure 6.1. Various test vectors applied to a slice of the RCA adder.

inputs and its effect should be directly observable at the circuit's primary outputs.

To generate test patterns for the adder, consider the slices of the RCA adder as shown in Figure 6.1. Each of the slices shows a different input vector being applied to it and the output that it will produce (which is incidentally the same in all the cases). Each figure actually represents two tests since each D has to first be replaced by a 0 and then by a 1. Observe that if all these eight tests are performed, then each node in the circuit goes to a 0 as well as to a 1 at one time or another.

The inputs to each slice can be set independently of the others except for the carry-in which is propagated from the previous slices. A carry-in of 0 for each slice can easily be achieved by forcing all the inputs of all other slices of the adder to zeroes. The only exception is the first slice where the carry-in is connected to SA. For the first three vectors when SA=0, the first slice behaves in a manner similar to the others. However, when SA=1 (fourth vector), the sum is equal to D and carry out is 1.

Thus eight tests are required to test each of the slices in the adder, taking the total number of test vectors for the adder to 144 (18×8). Appendix A shows all the test vectors generated for the Adder. Only four test vectors are shown for each slice. The D can be first replaced by a 0 and then by a 1 to get the actual eight vectors. Again, since the expected result for each set of four test vectors is the same, one result is shown after each set of four vectors.

The last module in the circuit for which tests have to be generated is the multiplier, which is an 18-by-18 Baugh-Wooley array multiplier. It is the most complex module of the DKS chip with regard to size. It is made of 309 full adders, 324 AND gates, and 36 inverters.

The whole array is regularly structured except for the last two rows and the last slanting column. On close observation it is found that if the sum and carry outputs of each of the full adders is set to D and is made observable at the output, all stuck-at faults at the nodes can be detected. From the D-propagation table of the full adder, it is seen that if one of the inputs is 0, another is 1, and yet another is D, then the sum is $\overline{D}$ and the carry-out is D. The sum can be propagated down to the output if all the carry-ins for other FAs are zeroes. That can easily be

achieved with a (0,0,0) input to all other adders. The carry-out can be similarly propagated down in the next higher column. So two tests are required to test the output nodes of each of the FAs, one with D=0 and the other with D=1. Thus the expected result will be all zeroes except for a $\overline{D}$ in that column and a D in the next higher bit column.

However, there are a few complications. One is that when trying to set a particular node to 1 or D, the inputs to some other full adders may also be set to a non-zero value in the process. This will be reflected in the output as a 1 or a D in those particular columns. So the expected result has to be changed in those columns. Another piculiarity is the last FA for S_{35} where one of the inputs is always set to 1. When testing the FA above it which has $\overline{a}_{17}$ and $\overline{b}_{17}$ as its inputs, the last FA receives a D carry-in and a 0 from the carry-out of its predecessor. So its output is equal to $\overline{D}$ instead of a D.

Based on these observations, test vectors were generated for all the FAs. Since there are 309 full adders and the lowermost row of 19 FAs is tested at the same time as the one above it, 580 (290×2) test vectors are required to test the multiplier. Appendix B gives the full list of test vectors along with the expected results for each vector.

This completes the test generation process for the DKS chip. A few things need to be pointed out regarding the matter of test verification, which always follows a test generation process. The purpose of test verification is to prove that the test vectors that have been generated are *sufficient* to test the circuit. The latches and registers in the DKS chip use standard shift and flush tests, so there is no need to verify the vectors. Similarly, the ROMs are being tested exhaustively, so the question of test verification does not arise at all. The only modules that need test verification are the adder and the multiplier.

Fault coverage for a circuit is measured in terms of the percentage of faults covered by the tests with respect to the fault model of the circuit. For both the adder and the multiplier, a functional stuck-at model was assumed to generate test vectors. Since test vectors to test each node in the functional circuit were individually generated by hand, a 100% fault coverage was achieved. However, it has to be emphasised that a general stuck-at fault model was not

assumed, so the fault coverage with respect to such a model will be less than the 100% figure shown here.

The expected results for each test vector were generated with respect to the circuit connections. To confirm that the results actually matched the test vector inputs to the adder and multiplier, a random sample of 30 vectors for each of them was selected. More vectors were selected near the boundaries where probability of error is higher. The input vectors for the adder were then added by hand and the results always matched the expected results generated (see Appendix A). Similarly for the multiplier, the test vectors were multipled by hand and again results matched the expected results shown in Appendix B.

This concludes the test generation and verification process for the DKS chip. The next chapter deals with the process of applying these vectors to the chip once it has been fabricated.

VII. TESTING THE DKS CHIP

Test application is the final phase of a testing process and can begin after the chip has been fabricated and test vectors generated and verified. The tests are applied to the chip under test with the help of Automatic Test Equipment (ATE). The ATE consists of a test bed on which to place the chip under test. It is then programmed to apply the test vectors to the chip and store the results in its memory. Most ATEs can also compare the results with the expected results and generate diagnostics, depending on their capability.

The procedure to test a chip without any structured DFT features is different from the procedure for a chip with DFT. Further, it also depends on which of the DFT techniques has been used. In the earlier chapters, it was shown that BST is the most suitable technique for the DKS chip. The process to test a chip with BST in general was explained in Section 4.2. It involves testing each module on the chip one at a time independently of the others. The procedure to test the DKS chip in particular is explained in the first section of this chapter. The second section then evaluates the performance of the testing process.

7.1. Test Application Process

The DKS chip has been partitioned into modules and each of them is tested independently of the others. Since there are two internal busses on the DKS chip, it is possible to test two modules at a time. So an effort is made to test two modules in parallel whereever possible. To test any two modules in parallel, it should be possible to set both of them independently, but at the same time. Also, it should be possible to scan out their outputs without any resource conflicts.

The procedure to test each module depends on its type: whether it is a ROM, a latch, a register, a delay element, or a combinational circuit. Also, the sequence in which some of the modules have to be tested is crucial because some of the modules are accessed through some other ones. The sequence in which the various modules are tested for the DKS chip is listed next.

1) Test the SRL and TSRL cells.

2) Test the bus and the transmission gates of TSRL cells.

3) Test the control logic ROM.

4) Test the angle registers and constants ROM.

   The input of the angle registers is from the chip's primary inputs and the ROM does not require any input except for the address which is provided from the control logic. The output of angle registers is through bus A TSRLs and of the constants ROM through bus B TSRLs. Thus their are no resource conflicts among these two modules. Though the angle registers require more test vectors than the ROM, some time is still saved by testing both of them at the same time.

   To test the angle registers, first set all of them to the test vector values using the normal DKS chip's input process. Then, scan in control signals to transfer data from one of the registers and one of the constants to the TSRLs of bus A and B respectively. Apply one system clock and scan out the TSRL contents to verify the results. The procedure is repeated for each test vector and then for each angle register and each constant in the ROM.

5) Test the M1 and M2 latches.

   Since M1 uses bus A for both its input and output, and M2 uses bus B, there are no resource conflicts among them. To test these latches, test vectors for each of them along with the control signals required to set them, are scanned in. One system clock is applied and then control signals to transfer data into the TSRLs are scanned in. Again one system clock is applied to transfer the data and the results are scanned out.

6) Test the A1 and A2 latches.

The same procedure that was used to test M1 and M2 can be used here.

7) Test the LX latch and the C-register.

The LX latch uses the angle registers for its input, and M1 latch and bus A for its output. On the other hand, the C-register uses bus B for its input as well as output, so both of these latches can be tested at the same time. Since the angle registers and M1 have already been tested at this point, they can now be used for providing access to the LX latch.

To test them, set one of the angle registers to the LX latch's test vector value and set TSRL for bus B to the test vector value of the C-register. Transfer data from that angle register to LX latch and then to the M1 latch. Now load the TSRLs of bus A from M1 and of bus B from C-register and scan them out. Repeat the procedure for all the test vectors.

8) Test the two-stage delay (LA) and the S-register.

The two stage delay uses bus A for its input, and A1 and bus A again for its output. The S-register, on the other hand, uses bus B for its input as well as output. Since there are no resource conflicts among them, they can be tested in parallel.

First the test vectors for both of them are scanned into their respective TSRLs and then one system clock is applied to set them. Two more system clocks are applied to give two clock delays and set the A1 latch from the two stage delay's output. Lastly, A1 and the C-register are both loaded into the TSRLs, the data in which is then scanned out.

9) Test the M3 latch and the register file.

The M3 latch uses bus A for its input and bus B for its output, while the register file uses bus B for its input and bus A for its output. Each register in the register file is to be tested independently of the others. So the M3 latch can be tested while testing one of the registers and rest of the registers can be tested alone.

To test both of the modules together, both sets of TSRLs are set to test vector values. One system clock is then applied to transfer data to M3 and the register under test. Control signals to transfer data back into the TSRLs from the modules are scanned in and one

system clock is applied to load the TSRLs. The data in the TSRLs is then scanned out.

10) Test Dx and Tx ROM tables.

There is only one input for the two parts of the table ROM and it is through the angle registers. The output of Dx is through bus B and the output of Tx is through bus A. Since the ROM is to be tested exhaustively, the small combinational circuit at its input can also be included in the tests. The test vector is first scanned into one of the angle registers and is applied to the ROM during the next system clock cycle. The data at their output is then transferred to the respective TSRLs and scanned out.

11) Test the MPY1 unit.

The first part of multiplier requires two inputs: one from the M1 latch through bus A and the other from the M2 latch through bus B. Since it utilizes both the busses for its inputs, nothing else can be tested in its parallel. First, both its input latches M1 and M2 are set to test vector values through the TSRLs. One system clock is applied and the result of the operation is stored in the M3 latch, which can be scanned out through the TSRLs of the bus B. The procedure is repeated for all the test vectors.

12) Test the MPY2 unit.

The procedure to test MPY2 is the same as for MPY1 except that only M3 needs to be set to the test vector value. Its output is latched into A2 and is scanned out through the TSRLs of bus B.

13) Test the adder.

The procedure to test the adder is exactly the same as for MPY1, except that A1 and A2 are used for input instead of M1 and M2, and its output is directly latched into the TSRLs of bus B through the pass transistors.

This completes the test application process for the DKS chip. To show that this application process is really feasible and adequate for testing the DKS chip, the process was simulated. Appendix C gives the complete listing of the simulation program. Given the structure of the DKS chip along with the modifications suggested in Chapter V to implement BST, the

objective of the simulation is to show that this application process provides access to all the modules on the chip through the scan path. Following is an explanation of the simulation program.

The chip structure is simulated in two functions: *phase_one* and *phase_two*. Each of the functions simulates all the operations performed during the respective clock cycles. The only parameters that can be passed or accessed from these functions are the control signals and TSRL cell values. Also, no other function in the rest of the program can use or set any of the modules in the circuit except for the control signals and the TSRL cells that are accessible through the scan path and the I/O signals.

The main program first initializes the ROM contents to some specific values and sets all control signals, latches, and registers to an undefined value (-1). Then the main program calls one function each to test each / each-pair of the modules, as numbered in sequence from 2 to 13 earlier in this Chapter. Thus each of the modules is tested independently of the others. The module number 1 is the shift register of SRL and TSRL cells themselves that are assumed to be accessible and thus do not need any simulation.

Each function to test modules, first prints the name of the module it is testing and then reads the number of test vectors from the test vector file. The steps required to actually test the modules are exactly as explained in that particular section in this Chapter and they are all repeated for the number of test vectors for that module. The only operations that these functions can perform are:

- set SRL control signals;
- set TSRL cells (from the test vector file);
- reset the chip (set the counter of control logic to zero);
- set I/O signals;
- call *phase_one* and *phase_two* functions;

- print the contents of TSRL cells.

Restricting the operations that can be performed by the modules to these is in accordance with the objective of the simulation.

The simulation ran successfully showing that it is possible to access all circuit modules of the DKS chip using the SRL and TSRL cells. It also verified the test application process as detailed earlier in this Chapter. The simulation proves the feasibility of using BST on the DKS chip. The performance of the DKS chip with BST is now evaluated in the next section.

7.2. Performance Evaluation

The performance of a DFT technique can be evaluated in terms of the following parameters:

1. Area overhead required on the chip to implement the technique;
2. Number of IC pins dedicated to testing;
3. Test application time;
4. Test vector length.

The performance of the testing process of the DKS chip will be compared among the three possibilities that are present to test it. First is to design the chip without any testability features and test it. The other two are to either implement LSSD or BST during the design process and then test it. A measure of performance for each of the four parameters above will be calculated for the three test possibilities.

The first parameter is the area overhead required to implement the DFT technique. Clearly, there will be no area overhead if none of the DFT techniques is used at all. If LSSD is used then all 458 latches have to be converted to SRLs. This amounts to an area overhead of 17%. On the other hand, if BST is used then 44 latches have to be converted to SRLs, and 7 SRLs, 36 TSRLs, and 138 pass-transistors have to be added. This is equivalent to an area overhead of a little less than 10% on the DKS chip.

The second parameter of performance is the number of additional pins required on the chip that are dedicated to testing. These pins do not provide any function during the normal operation of the circuit and since I/O pins are very precious in VLSI, every effort should be made to keep them to a minimum. Of course, if none of the DFT technique is used, no extra pins will be required. LSSD requires four extra pins (SDI, SDO, and clocks A and B), while BST requires five of them (the above four and $\overline{En}$).

The primary and the most important measure of a testing process is the test application time. The objective of the DFT techniques is to reduce the time required to test the chip. Since the DKS chip has a 12-bit data input and 458 latches, the number of test vectors required to test the chip exhaustively is $2^{12+458} = 2^{470}$. This converts to a testing time of 10^{127} years. If BST is used then the chip is partitioned and each module is available separately for testing. Table 7.1 shows the number of test vectors and the corresponding time required to test each of the modules. It has been assumed that 100 clocks at 1 MHz clock speed are required to apply one test vector, since the length of the scan path is 80 and a few system clocks may have to be applied in the process. The total number of test vectors required are 5,629, so the total time required to test the DKS chip using BST is 0.56 secs.

If LSSD is used, then the length of scan path is 458 and thus 458 μseconds are required to apply each test vector. However, all the modules can be tested in parallel now. So the total time required to test the DKS chip will be the same as for the module which has the most number of test vectors. Table 7.1 shows that the ROM requires 4096 test vectors, so the total time required to test the chip is 4096×458μsecs = 187 secs ≈ 3 mins.

The last parameter for measuring the performance of a DFT technique is the test vector length. If none of the DFT techniques is used, then the only access to the chip circuitry is through the 12-bit data input pins on the chip. Thus the test vector length is only 12 bits. However, if some structured DFT technique is used, then the test vector also includes all cells in its scan path, in addition to the 12-bit primary data input. Larger test vector lengths require ATEs with greater capacities to handle them, so it should be kept as small as possible. Since the

number of latches in the DKS chip is 458, the test vector length is 458 + 12 = 470. On the other hand, if BST is used, two sets of TSRLs are added (2×18 = 36), control logic output latch is converted to SRLs (37), and 7 SRLs are added to control extra signals, taking the total test vector length to (36 + 37 + 7 + 12) or 92.

The various performance measures are summarized in Table 7.2. LSSD requires more time for testing the chip, a larger area overhead, and a longer test vector length as compared to BST. The only disadvantage of using BST is that it requires an extra pin over LSSD. Since the advantages of BST far outweigh its only disadvantage, it is concluded that BST should be implemented on the DKS chip to improve its testability.

Table 7.1. Test application time for various modules in the DKS chip.

Module name	Number of test vectors	Total time (msec)
SRL and TSRL cells	4	0.4
Busses	4	0.4
Control logic	73	7.3
Angle registers	24	2.4
Constants ROM	5	*
M1 and M2	4	0.4
A1 and A2	4	0.4
LX	4	0.4
C-register	4	*
LA	4	0.4
S-register	4	*
Register file	36	3.6
M3	4	*
Dx and Tx ROM	4096	409.6
MPY1	616	61.6
MPY2	616	61.6
Adder	144	14.4
TOTAL	5629	562.9

Table 7.2. Comparison of the estimated overheads for the DKS chip.

DFT technique	Area Overhead (%)	Number of extra pins	Test vector length (bits)	Test Application time (secs)
none	-	-	12	
LSSD	17	4	458	187
BST	10	5	85	0.56

*The time taken to test these modules is not used in the calculation of the total time because they are tested in parallel with other modules.

VIII. CONCLUSION

The objective of this thesis is to incorporate DFT into the design of the DKS chip. It has been recognized that testing VLSI chips is by no means a trivial process. Design for testability techniques are being used nowadays to design the chips in such a way that they are easily testable. This is especially true for semi-custom ASICs that require fast turnaround times and can pay the price of a little area overhead on the chip. For these reasons, it was decided that some DFT technique must be implemented on the DKS chip.

The DKS chip has two internal data busses and it was observed that if they could be somehow used to provide access to the majority of the storage elements on the chip, then the area overhead required to implement a DFT technique can be reduced. This idea developed into a new DFT technique - *Bus Scan Testing* (BST), that can be applied to all internal bus architecture systems. BST integrates the concepts of partitioning, bus architecture systems and scan techniques to provide access to all the storage elements in the circuit. It results in a reduction of area overhead and shortens the test vector length. However, in some cases it may result in longer test application times than the other structured DFT techniques.

Appropriate modifications were made in the design of the DKS chip to implement BST. Test vectors were then generated separately for each of the partitioned modules in the DKS chip. The test application procedure for the DKS chip was then drawn out that included the sequence in which various modules had to be tested and the steps to test each of them. A simulation was then carried out to show that the modifications made on the DKS chip and the test application process can provide access to all circuit modules and is sufficient to test the DKS chip. Lastly, the testability design of the DKS chip was evaluated which confirmed that BST was the best available DFT technique that could be applied to the DKS chip.

The BST technique, as described, provides many advantages over other DFT techniques for internal bus architecture systems. However, there is still further scope of improvement. One is to try to convert one of the circuit latches themselves to TSRLs and then use BST. Though the idea seems to be plausible, further simulations and tests are required to prove that it will not cause problems during normal circuit operation.

One area in which BST lacks diagnostic abilty is bus faults. Though BST detects most of the bus faults on the chip, it does not give any indication as to the location of the faults. If such a system can be developed, then even the control busses can be tested in addition to the data busses being tested now.

APPENDICES

APPENDIX A

APPENDIX A

TEST VECTORS FOR ADDER

$g = \overline{d}$

	ba ba ba ba ba ba ba ba ba ba ba ba ba ba ba ba	SA1	SA2
bit#	17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00		
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1d	0	0
0	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1g	1	0
0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 d1	0	0

Result: 00 0000 0000 0000 00dg

0	10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 g1	0	1

Result: 00 0000 0000 0000 001d

1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1d 00	0	0
1	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1g 01	1	0
1	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 d1 00	0	0
1	10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 g1 10	0	1

Result: 00 0000 0000 0000 0dgx

2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1d 00 00	0	0
2	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1g 01 01	1	0
2	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 d1 00 00	0	0
2	10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 g1 10 10	0	1

Result: 00 0000 0000 0000 dg0x

3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00	0	0
3	01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1g 01 01 01	1	0
3	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 d1 00 00 00	0	0
3	10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 g1 10 10 10	0	1

Result: 00 0000 0000 000d g00x

4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00	0	0
4	01 01 01 01 01 01 01 01 01 01 01 01 01 01 1g 01 01 01 01	1	0
4	00 00 00 00 00 00 00 00 00 00 00 00 00 00 d1 00 00 00 00	0	0

4 10 10 10 10 10 10 10 10 10 10 10 10 g1 10 10 10 10 0 1

Result: 00 0000 0000 00dg 000x

5 00 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 0 0
5 01 01 01 01 01 01 01 01 01 01 01 01 1g 01 01 01 01 01 1 0
5 00 00 00 00 00 00 00 00 00 00 00 00 d1 00 00 00 00 00 0 0
5 10 10 10 10 10 10 10 10 10 10 10 10 g1 10 10 10 10 10 0 1

Result: 00 0000 0000 0dg0 000x

6 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 0 0
6 01 01 01 01 01 01 01 01 01 01 01 1g 01 01 01 01 01 01 1 0
6 00 00 00 00 00 00 00 00 00 00 00 d1 00 00 00 00 00 00 0 0
6 10 10 10 10 10 10 10 10 10 10 10 g1 10 10 10 10 10 10 0 1

Result: 00 0000 0000 dg00 000x

7 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 0 0
7 01 01 01 01 01 01 01 01 01 01 01 1g 01 01 01 01 01 01 1 0
7 00 00 00 00 00 00 00 00 00 00 00 d1 00 00 00 00 00 00 0 0
7 10 10 10 10 10 10 10 10 10 10 10 g1 10 10 10 10 10 10 0 1

Result: 00 0000 000d g000 000x

8 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 0 0
8 01 01 01 01 01 01 01 01 01 1g 01 01 01 01 01 01 01 01 1 0
8 00 00 00 00 00 00 00 00 00 d1 00 00 00 00 00 00 00 00 0 0
8 10 10 10 10 10 10 10 10 10 g1 10 10 10 10 10 10 10 10 0 1

Result: 00 0000 00dg 0000 000x

9 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 0 0
9 01 01 01 01 01 01 01 01 1g 01 01 01 01 01 01 01 01 01 1 0
9 00 00 00 00 00 00 00 00 d1 00 00 00 00 00 00 00 00 00 0 0
9 10 10 10 10 10 10 10 10 g1 10 10 10 10 10 10 10 10 10 0 1

Result: 00 0000 0dg0 0000 000x

10 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 0 0
10 01 01 01 01 01 01 01 1g 01 01 01 01 01 01 01 01 01 01 1 0
10 00 00 00 00 00 00 00 d1 00 00 00 00 00 00 00 00 00 00 0 0
10 10 10 10 10 10 10 10 g1 10 10 10 10 10 10 10 10 10 10 0 1

Result: 00 0000 dg00 0000 000x

11 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 0 0

11 01 01 01 01 01 01 1g 01 01 01 01 01 01 01 01 01 01 01 1 0
11 00 00 00 00 00 00 d1 00 00 00 00 00 00 00 00 00 00 00 0 0
11 10 10 10 10 10 10 g1 10 10 10 10 10 10 10 10 10 10 10 0 1

Result: 00 000d g000 0000 000x

12 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 0 0
12 01 01 01 01 01 1g 01 01 01 01 01 01 01 01 01 01 01 01 1 0
12 00 00 00 00 00 d1 00 00 00 00 00 00 00 00 00 00 00 00 0 0
12 10 10 10 10 10 g1 10 10 10 10 10 10 10 10 10 10 10 10 0 1

Result: 00 00dg 0000 0000 000x

13 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
13 01 01 01 01 1g 01 01 01 01 01 01 01 01 01 01 01 01 01 1 0
13 00 00 00 00 d1 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
13 10 10 10 10 g1 10 10 10 10 10 10 10 10 10 10 10 10 10 0 1

Result: 00 0dg0 0000 0000 000x

14 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
14 01 01 01 1g 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1 0
14 00 00 00 d1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
14 10 10 10 g1 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 1

Result: 00 dg00 0000 0000 000x

15 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
15 01 01 1g 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1 0
15 00 00 d1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
15 10 10 g1 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 1

Result: 0d g000 0000 0000 000x

16 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
16 01 1g 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1 0
16 00 d1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
16 10 g1 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 1

Result: dg 0000 0000 0000 000x

17 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
17 1g 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 1 0
17 d1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0 0
17 g1 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 1

Result: g0 0000 0000 0000 000x

APPENDIX B

APPENDIX B

TEST VECTORS FOR MULTIPLIER

$g = \bar{d}$

	ba 17	ba 16	ba 15	ba 14	ba 13	ba 12	ba 11	ba 10	ba 09	ba 08	ba 07	ba 06	ba 05	ba 04	ba 03	ba 02	ba 01	ba 00
a0b1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1d	11
a0b1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	d0	g1
a1b1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	11	10
a1b1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	0g	10
a0b2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1d	00	11
a0b2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	dg	01
a2b1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	01	10	10
a2b1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	d0	g1	00
a1b2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	10	01	10
a1b2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	dd	g0	10
a0b3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	1d	00	00	11
a0b3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	0d	g0	01
a3b1	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	01	00	10	10
a3b1	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	0g	10	00	
a2b2	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	00	11	00	10
a2b2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	dg	01	00
a1b3	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	10	00	01	10
a1b3	00	00	00	00	00	00	00	00	00	00	00	00	00	d0	dg	00	10	
a0b4	00	00	00	00	00	00	00	00	00	00	00	00	1d	00	00	00	11	
a0b4	00	00	00	00	00	00	00	00	00	00	00	00	0d	00	dg	00	01	
a4b1	00	00	00	00	00	00	00	00	00	00	00	00	0d	01	00	00	10	10
a4b1	00	00	00	00	00	00	00	00	00	00	00	00	00	d0	g1	00	00	
a3b2	00	00	00	00	00	00	00	00	00	00	00	00	0d	00	01	10	00	10
a3b2	00	00	00	00	00	00	00	00	00	00	00	00	00	dd	g0	10	00	
a2b3	00	00	00	00	00	00	00	00	00	00	00	00	0d	00	10	01	00	10
a2b3	00	00	00	00	00	00	00	00	00	00	00	00	0d	0d	g0	01	00	
a1b4	00	00	00	00	00	00	00	00	00	00	00	00	0d	10	00	00	01	10
a1b4	00	00	00	00	00	00	00	00	00	00	00	00	d0	0d	g0	00	10	
a0b5	00	00	00	00	00	00	00	00	00	00	00	1d	00	00	00	00	11	
a0b5	00	00	00	00	00	00	00	00	00	00	00	0d	00	0d	g0	00	01	
a5b1	00	00	00	00	00	00	00	00	00	00	00	0d	01	00	00	00	10	10
a5b1	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	0g	10	00	00
a4b2	00	00	00	00	00	00	00	00	00	00	00	0d	00	01	00	10	00	10
a4b2	00	00	00	00	00	00	00	00	00	00	00	00	00	0d	dg	01	00	00
a3b3	00	00	00	00	00	00	00	00	00	00	00	0d	00	00	11	00	00	10

a3b3 00 00 00 00 00 00 00 00 00 00 00 00 00 d0 dg 00 10 00
a2b4 00 00 00 00 00 00 00 00 00 00 00 00 0d 00 10 00 01 00 10
a2b4 00 00 00 00 00 00 00 00 00 00 00 00 0d 00 dg 00 01 00
a1b5 00 00 00 00 00 00 00 00 00 00 00 00 0d 10 00 00 00 01 10
a1b5 00 00 00 00 00 00 00 00 00 00 00 00 00 d0 00 dg 00 00 10
a0b6 00 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 11
a0b6 00 00 00 00 00 00 00 00 00 00 00 00 0d 00 00 dg 00 00 01
a6b1 00 00 00 00 00 00 00 00 00 00 00 0d 01 00 00 00 00 10 10
a6b1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 d0 g1 00 00 00
a5b2 00 00 00 00 00 00 00 00 00 00 00 0d 00 01 00 00 10 00 10
a5b2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 dd g0 10 00 00
a4b3 00 00 00 00 00 00 00 00 00 00 00 0d 00 00 01 10 00 00 10
a4b3 00 00 00 00 00 00 00 00 00 00 00 00 00 0d 0d g0 01 00 00
a3b4 00 00 00 00 00 00 00 00 00 00 00 0d 00 00 10 01 00 00 10
a3b4 00 00 00 00 00 00 00 00 00 00 00 00 00 d0 0d g0 00 10 00
a2b5 00 00 00 00 00 00 00 00 00 00 00 0d 00 10 00 00 01 00 10
a2b5 00 00 00 00 00 00 00 00 00 00 00 00 0d 00 0d g0 00 01 00
a1b6 00 00 00 00 00 00 00 00 00 00 00 0d 10 00 00 00 00 01 10
a1b6 00 00 00 00 00 00 00 00 00 00 00 00 d0 00 0d g0 00 00 10
a0b7 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 11
a0b7 00 00 00 00 00 00 00 00 00 00 00 0d 00 00 0d g0 00 00 01
a7b1 00 00 00 00 00 00 00 00 00 00 0d 01 00 00 00 00 00 10 10
a7b1 00 00 00 00 00 00 00 00 00 00 00 00 00 0d 0g 10 00 00 00
a6b2 00 00 00 00 00 00 00 00 00 00 0d 00 01 00 00 00 10 00 10
a6b2 00 00 00 00 00 00 00 00 00 00 00 00 00 0d dg 01 00 00 00
a5b3 00 00 00 00 00 00 00 00 00 00 0d 00 00 01 00 10 00 00 10
a5b3 00 00 00 00 00 00 00 00 00 00 00 00 00 d0 dg 00 10 00 00
a4b4 00 00 00 00 00 00 00 00 00 00 0d 00 00 00 11 00 00 00 10
a4b4 00 00 00 00 00 00 00 00 00 00 00 00 0d 00 dg 00 01 00 00
a3b5 00 00 00 00 00 00 00 00 00 00 0d 00 00 10 00 01 00 00 10
a3b5 00 00 00 00 00 00 00 00 00 00 00 00 d0 00 dg 00 00 10 00
a2b6 00 00 00 00 00 00 00 00 00 00 0d 00 10 00 00 00 01 00 10
a2b6 00 00 00 00 00 00 00 00 00 00 00 0d 00 00 dg 00 00 01 00
a1b7 00 00 00 00 00 00 00 00 00 00 0d 10 00 00 00 00 00 01 10
a1b7 00 00 00 00 00 00 00 00 00 00 00 d0 00 00 dg 00 00 00 10
a0b8 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 11
a0b8 00 00 00 00 00 00 00 00 00 00 0d 00 00 00 dg 00 00 00 01
a8b1 00 00 00 00 00 00 00 00 00 0d 01 00 00 00 00 00 00 10 10
a8b1 00 00 00 00 00 00 00 00 00 00 00 00 00 d0 g1 00 00 00 00
a7b2 00 00 00 00 00 00 00 00 00 0d 00 01 00 00 00 00 10 00 10
a7b2 00 00 00 00 00 00 00 00 00 00 00 00 00 dd g0 10 00 00 00
a6b3 00 00 00 00 00 00 00 00 00 0d 00 00 01 00 00 10 00 00 10
a6b3 00 00 00 00 00 00 00 00 00 00 00 00 0d 0d g0 01 00 00 00
a5b4 00 00 00 00 00 00 00 00 00 0d 00 00 00 01 10 00 00 00 10
a5b4 00 00 00 00 00 00 00 00 00 00 00 00 d0 0d g0 00 10 00 00
a4b5 00 00 00 00 00 00 00 00 00 0d 00 00 00 10 01 00 00 00 10
a4b5 00 00 00 00 00 00 00 00 00 00 00 0d 00 0d g0 00 01 00 00

a3b6 00 00 00 00 00 00 00 00 0d 00 00 10 00 00 01 00 00 10
a3b6 00 00 00 00 00 00 00 00 00 00 d0 00 0d g0 00 00 10 00
a2b7 00 00 00 00 00 00 00 00 0d 00 10 00 00 00 00 01 00 10
a2b7 00 00 00 00 00 00 00 00 00 0d 00 00 0d g0 00 00 01 00
a1b8 00 00 00 00 00 00 00 00 0d 10 00 00 00 00 00 00 01 10
a1b8 00 00 00 00 00 00 00 00 00 d0 00 00 0d g0 00 00 00 10
a0b9 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 11
a0b9 00 00 00 00 00 00 00 00 0d 00 00 00 0d g0 00 00 00 01
a9b1 00 00 00 00 00 00 00 0d 01 00 00 00 00 00 00 00 10 10
a9b1 00 00 00 00 00 00 00 00 00 00 00 0d 0g 10 00 00 00 00
a8b2 00 00 00 00 00 00 00 0d 00 01 00 00 00 00 00 10 00 10
a8b2 00 00 00 00 00 00 00 00 00 00 00 0d dg 01 00 00 00 00
a7b3 00 00 00 00 00 00 00 0d 00 00 01 00 00 00 10 00 00 10
a7b3 00 00 00 00 00 00 00 00 00 00 00 d0 dg 00 10 00 00 00
a6b4 00 00 00 00 00 00 00 0d 00 00 00 01 00 10 00 00 00 10
a6b4 00 00 00 00 00 00 00 00 00 00 0d 00 dg 00 01 00 00 00
a5b5 00 00 00 00 00 00 00 0d 00 00 00 00 11 00 00 00 00 10
a5b5 00 00 00 00 00 00 00 00 00 00 d0 00 dg 00 00 10 00 00
a4b6 00 00 00 00 00 00 00 0d 00 00 00 10 00 01 00 00 00 10
a4b6 00 00 00 00 00 00 00 00 00 0d 00 00 dg 00 00 01 00 00
a3b7 00 00 00 00 00 00 00 0d 00 00 10 00 00 00 01 00 00 10
a3b7 00 00 00 00 00 00 00 00 00 d0 00 00 dg 00 00 00 10 00
a2b8 00 00 00 00 00 00 00 0d 00 10 00 00 00 00 00 01 00 10
a2b8 00 00 00 00 00 00 00 00 0d 00 00 00 dg 00 00 00 01 00
a1b9 00 00 00 00 00 00 00 0d 10 00 00 00 00 00 00 00 01 10
a1b9 00 00 00 00 00 00 00 00 d0 00 00 00 dg 00 00 00 00 10
a0b10 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 11
a0b10 00 00 00 00 00 00 00 0d 00 00 00 00 dg 00 00 00 00 01
a10b1 00 00 00 00 00 00 0d 01 00 00 00 00 00 00 00 00 10 10
a10b1 00 00 00 00 00 00 00 00 00 00 00 d0 g1 00 00 00 00 00
a9b2 00 00 00 00 00 00 0d 00 01 00 00 00 00 00 00 10 00 10
a9b2 00 00 00 00 00 00 00 00 00 00 00 dd g0 10 00 00 00 00
a8b3 00 00 00 00 00 00 0d 00 00 01 00 00 00 00 10 00 00 10
a8b3 00 00 00 00 00 00 00 00 00 00 0d 0d g0 01 00 00 00 00
a7b4 00 00 00 00 00 00 0d 00 00 00 01 00 00 10 00 00 00 10
a7b4 00 00 00 00 00 00 00 00 00 00 d0 0d g0 00 10 00 00 00
a6b5 00 00 00 00 00 00 0d 00 00 00 00 01 10 00 00 00 00 10
a6b5 00 00 00 00 00 00 00 00 00 00 0d 00 0d g0 00 01 00 00 00
a5b6 00 00 00 00 00 00 0d 00 00 00 00 10 01 00 00 00 00 10
a5b6 00 00 00 00 00 00 00 00 00 00 d0 00 0d g0 00 00 10 00 00
a4b7 00 00 00 00 00 00 0d 00 00 00 10 00 00 01 00 00 00 10
a4b7 00 00 00 00 00 00 00 00 00 0d 00 00 0d g0 00 00 01 00 00
a3b8 00 00 00 00 00 00 0d 00 00 10 00 00 00 00 01 00 00 10
a3b8 00 00 00 00 00 00 00 00 00 d0 00 00 0d g0 00 00 00 10 00
a2b9 00 00 00 00 00 00 0d 00 10 00 00 00 00 00 00 01 00 10
a2b9 00 00 00 00 00 00 00 00 0d 00 00 00 0d g0 00 00 00 01 00
a1b10 00 00 00 00 00 00 0d 10 00 00 00 00 00 00 00 00 01 10

a1b10 00 00 00 00 00 00 00 d0 00 00 00 0d g0 00 00 00 00 10
a0b11 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 11
a0b11 00 00 00 00 00 00 0d 00 00 00 00 0d g0 00 00 00 00 01
a11b1 00 00 00 00 00 0d 01 00 00 00 00 00 00 00 00 00 10 10
a11b1 00 00 00 00 00 00 00 00 00 00 0d 0g 10 00 00 00 00 00
a10b2 00 00 00 00 00 0d 00 01 00 00 00 00 00 00 00 10 00 10
a10b2 00 00 00 00 00 00 00 00 00 00 0d dg 01 00 00 00 00 00
a9b3 00 00 00 00 00 0d 00 00 01 00 00 00 00 00 10 00 00 10
a9b3 00 00 00 00 00 00 00 00 00 00 d0 dg 00 10 00 00 00 00
a8b4 00 00 00 00 00 0d 00 00 00 01 00 00 00 10 00 00 00 10
a8b4 00 00 00 00 00 00 00 00 00 0d 00 dg 00 01 00 00 00 00
a7b5 00 00 00 00 00 0d 00 00 00 00 01 00 10 00 00 00 00 10
a7b5 00 00 00 00 00 00 00 00 00 d0 00 dg 00 00 10 00 00 00
a6b6 00 00 00 00 00 0d 00 00 00 00 00 11 00 00 00 00 00 10
a6b6 00 00 00 00 00 00 00 00 0d 00 00 dg 00 00 01 00 00 00
a5b7 00 00 00 00 00 0d 00 00 00 00 10 00 01 00 00 00 00 10
a5b7 00 00 00 00 00 00 00 00 d0 00 00 dg 00 00 00 10 00 00
a4b8 00 00 00 00 00 0d 00 00 00 10 00 00 00 01 00 00 00 10
a4b8 00 00 00 00 00 00 00 0d 00 00 00 dg 00 00 00 01 00 00
a3b9 00 00 00 00 00 0d 00 00 10 00 00 00 00 00 01 00 00 10
a3b9 00 00 00 00 00 00 00 d0 00 00 00 dg 00 00 00 00 10 00
a2b10 00 00 00 00 00 0d 00 10 00 00 00 00 00 00 00 01 00 10
a2b10 00 00 00 00 00 00 0d 00 00 00 00 dg 00 00 00 00 01 00
a1b11 00 00 00 00 00 0d 10 00 00 00 00 00 00 00 00 00 01 10
a1b11 00 00 00 00 00 00 d0 00 00 00 00 dg 00 00 00 00 00 10
a0b12 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 11
a0b12 00 00 00 00 00 0d 00 00 00 00 00 dg 00 00 00 00 00 01
a12b1 00 00 00 00 0d 01 00 00 00 00 00 00 00 00 00 00 10 10
a12b1 00 00 00 00 00 00 00 00 00 00 d0 g1 00 00 00 00 00 00
a11b2 00 00 00 00 0d 00 01 00 00 00 00 00 00 00 00 10 00 10
a11b2 00 00 00 00 00 00 00 00 00 00 dd g0 10 00 00 00 00 00
a10b3 00 00 00 00 0d 00 00 01 00 00 00 00 00 00 10 00 00 10
a10b3 00 00 00 00 00 00 00 00 00 0d 0d g0 01 00 00 00 00 00
a9b4 00 00 00 00 0d 00 00 00 01 00 00 00 00 10 00 00 00 10
a9b4 00 00 00 00 00 00 00 00 00 d0 0d g0 00 10 00 00 00 00
a8b5 00 00 00 00 0d 00 00 00 00 01 00 00 10 00 00 00 00 10
a8b5 00 00 00 00 00 00 00 00 0d 00 0d g0 00 01 00 00 00 00
a7b6 00 00 00 00 0d 00 00 00 00 00 01 10 00 00 00 00 00 10
a7b6 00 00 00 00 00 00 00 00 d0 00 0d g0 00 00 10 00 00 00
a6b7 00 00 00 00 0d 00 00 00 00 00 10 01 00 00 00 00 00 10
a6b7 00 00 00 00 00 00 00 0d 00 00 0d g0 00 00 01 00 00 00
a5b8 00 00 00 00 0d 00 00 00 00 10 00 00 01 00 00 00 00 10
a5b8 00 00 00 00 00 00 00 d0 00 00 0d g0 00 00 00 10 00 00
a4b9 00 00 00 00 0d 00 00 00 10 00 00 00 00 01 00 00 00 10
a4b9 00 00 00 00 00 00 0d 00 00 00 0d g0 00 00 00 01 00 00
a3b10 00 00 00 00 0d 00 00 10 00 00 00 00 00 00 01 00 00 10
a3b10 00 00 00 00 00 00 d0 00 00 00 0d g0 00 00 00 00 10 00

a2b11 00 00 00 00 0d 00 10 00 00 00 00 00 00 00 00 01 00 10
a2b11 00 00 00 00 00 0d 00 00 00 00 0d g0 00 00 00 00 01 00
a1b12 00 00 00 00 0d 10 00 00 00 00 00 00 00 00 00 00 01 10
a1b12 00 00 00 00 00 d0 00 00 00 00 0d g0 00 00 00 00 00 10
a0b13 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 11
a0b13 00 00 00 00 0d 00 00 00 00 00 0d g0 00 00 00 00 00 01
a13b1 00 00 00 0d 01 00 00 00 00 00 00 00 00 00 00 00 10 10
a13b1 00 00 00 00 00 00 00 00 00 0d 0g 10 00 00 00 00 00 00
a12b2 00 00 00 0d 00 01 00 00 00 00 00 00 00 00 00 10 00 10
a12b2 00 00 00 00 00 00 00 00 00 0d dg 01 00 00 00 00 00 00
a11b3 00 00 00 0d 00 00 01 00 00 00 00 00 00 00 10 00 00 10
a11b3 00 00 00 00 00 00 00 00 00 d0 dg 00 10 00 00 00 00 00
a10b4 00 00 00 0d 00 00 00 01 00 00 00 00 00 10 00 00 00 10
a10b4 00 00 00 00 00 00 00 00 00 0d 00 dg 00 01 00 00 00 00 00
a9b5 00 00 00 0d 00 00 00 00 01 00 00 00 10 00 00 00 00 10
a9b5 00 00 00 00 00 00 00 00 d0 00 dg 00 00 10 00 00 00 00
a8b6 00 00 00 0d 00 00 00 00 00 01 00 10 00 00 00 00 00 10
a8b6 00 00 00 00 00 00 00 0d 00 00 dg 00 00 01 00 00 00 00
a7b7 00 00 00 0d 00 00 00 00 00 00 11 00 00 00 00 00 00 10
a7b7 00 00 00 00 00 00 00 d0 00 00 dg 00 00 00 10 00 00 00
a6b8 00 00 00 0d 00 00 00 00 00 10 00 01 00 00 00 00 00 10
a6b8 00 00 00 00 00 00 0d 00 00 00 dg 00 00 00 01 00 00 00
a5b9 00 00 00 0d 00 00 00 00 10 00 00 00 01 00 00 00 00 10
a5b9 00 00 00 00 00 00 d0 00 00 00 dg 00 00 00 00 10 00 00
a4b10 00 00 00 0d 00 00 00 10 00 00 00 00 00 01 00 00 00 10
a4b10 00 00 00 00 00 0d 00 00 00 00 dg 00 00 00 00 01 00 00
a3b11 00 00 00 0d 00 00 10 00 00 00 00 00 00 00 01 00 00 10
a3b11 00 00 00 00 00 d0 00 00 00 00 dg 00 00 00 00 00 10 00
a2b12 00 00 00 0d 00 10 00 00 00 00 00 00 00 00 00 01 00 10
a2b12 00 00 00 00 0d 00 00 00 00 00 dg 00 00 00 00 00 01 00
a1b13 00 00 00 0d 10 00 00 00 00 00 00 00 00 00 00 01 10
a1b13 00 00 00 00 d0 00 00 00 00 00 dg 00 00 00 00 00 00 10
a0b14 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 11
a0b14 00 00 00 0d 00 00 00 00 00 00 dg 00 00 00 00 00 00 01
a14b1 00 00 0d 01 00 00 00 00 00 00 00 00 00 00 00 00 10 10
a14b1 00 00 00 00 00 00 00 00 00 d0 g1 00 00 00 00 00 00 00
a13b2 00 00 0d 00 01 00 00 00 00 00 00 00 00 00 00 10 00 10
a13b2 00 00 00 00 00 00 00 00 00 dd g0 10 00 00 00 00 00 00
a12b3 00 00 0d 00 00 01 00 00 00 00 00 00 00 00 10 00 00 10
a12b3 00 00 00 00 00 00 00 00 0d 0d g0 01 00 00 00 00 00 00
a11b4 00 00 0d 00 00 00 01 00 00 00 00 00 00 10 00 00 00 10
a11b4 00 00 00 00 00 00 00 00 d0 0d g0 00 10 00 00 00 00 00
a10b5 00 00 0d 00 00 00 00 01 00 00 00 00 10 00 00 00 00 10
a10b5 00 00 00 00 00 00 00 0d 00 0d g0 00 01 00 00 00 00 00
a9b6 00 00 0d 00 00 00 00 00 01 00 00 10 00 00 00 00 00 10
a9b6 00 00 00 00 00 00 00 d0 00 0d g0 00 00 10 00 00 00 00
a8b7 00 00 0d 00 00 00 00 00 00 01 10 00 00 00 00 00 00 10

a8b7 00 00 00 00 00 00 0d 00 00 0d g0 00 00 01 00 00 00 00
a7b8 00 00 0d 00 00 00 00 00 00 10 01 00 00 00 00 00 00 10
a7b8 00 00 00 00 00 00 d0 00 00 0d g0 00 00 00 10 00 00 00
a6b9 00 00 0d 00 00 00 00 00 10 00 00 01 00 00 00 00 00 10
a6b9 00 00 00 00 00 0d 00 00 00 0d g0 00 00 00 01 00 00 00
a5b10 00 00 0d 00 00 00 00 10 00 00 00 00 01 00 00 00 00 10
a5b10 00 00 00 00 00 d0 00 00 00 0d g0 00 00 00 00 10 00 00
a4b11 00 00 0d 00 00 00 10 00 00 00 00 00 00 01 00 00 00 10
a4b11 00 00 00 00 0d 00 00 00 00 0d g0 00 00 00 00 01 00 00
a3b12 00 00 0d 00 00 10 00 00 00 00 00 00 00 00 01 00 00 10
a3b12 00 00 00 00 d0 00 00 00 00 0d g0 00 00 00 00 00 10 00
a2b13 00 00 0d 00 10 00 00 00 00 00 00 00 00 00 00 01 00 10
a2b13 00 00 00 0d 00 00 00 00 00 0d g0 00 00 00 00 00 01 00
a1b14 00 00 0d 10 00 00 00 00 00 00 00 00 00 00 00 00 01 10
a1b14 00 00 00 d0 00 00 00 00 00 0d g0 00 00 00 00 00 00 10
a0b15 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 11
a0b15 00 00 0d 00 00 00 00 00 00 0d g0 00 00 00 00 00 00 01
a15b1 00 0d 01 00 00 00 00 00 00 00 00 00 00 00 00 00 10 10
a15b1 00 00 00 00 00 00 00 00 0d 0g 10 00 00 00 00 00 00 00
a14b2 00 0d 00 01 00 00 00 00 00 00 00 00 00 00 00 10 00 10
a14b2 00 00 00 00 00 00 00 00 0d dg 01 00 00 00 00 00 00 00
a13b3 00 0d 00 00 01 00 00 00 00 00 00 00 00 00 10 00 00 10
a13b3 00 00 00 00 00 00 00 00 d0 dg 00 10 00 00 00 00 00 00
a12b4 00 0d 00 00 00 01 00 00 00 00 00 00 00 10 00 00 00 10
a12b4 00 00 00 00 00 00 00 0d 00 dg 00 01 00 00 00 00 00 00
a11b5 00 0d 00 00 00 00 01 00 00 00 00 00 10 00 00 00 00 10
a11b5 00 00 00 00 00 00 00 d0 00 dg 00 00 10 00 00 00 00 00
a10b6 00 0d 00 00 00 00 00 01 00 00 00 10 00 00 00 00 00 10
a10b6 00 00 00 00 00 00 0d 00 00 dg 00 00 01 00 00 00 00 00
a9b7 00 0d 00 00 00 00 00 00 01 00 10 00 00 00 00 00 00 10
a9b7 00 00 00 00 00 00 d0 00 00 dg 00 00 00 10 00 00 00 00
a8b8 00 0d 00 00 00 00 00 00 00 11 00 00 00 00 00 00 00 10
a8b8 00 00 00 00 00 0d 00 00 00 dg 00 00 00 01 00 00 00 00
a7b9 00 0d 00 00 00 00 00 00 10 00 01 00 00 00 00 00 00 10
a7b9 00 00 00 00 00 d0 00 00 00 dg 00 00 00 00 10 00 00 00
a6b10 00 0d 00 00 00 00 00 10 00 00 00 01 00 00 00 00 00 10
a6b10 00 00 00 00 0d 00 00 00 00 dg 00 00 00 00 01 00 00 00
a5b11 00 0d 00 00 00 00 10 00 00 00 00 00 01 00 00 00 00 10
a5b11 00 00 00 00 d0 00 00 00 00 dg 00 00 00 00 00 10 00 00
a4b12 00 0d 00 00 00 10 00 00 00 00 00 00 00 01 00 00 00 10
a4b12 00 00 00 0d 00 00 00 00 00 dg 00 00 00 00 00 01 00 00
a3b13 00 0d 00 00 10 00 00 00 00 00 00 00 00 00 01 00 00 10
a3b13 00 00 00 d0 00 00 00 00 00 dg 00 00 00 00 00 00 10 00
a2b14 00 0d 00 10 00 00 00 00 00 00 00 00 00 00 00 01 00 10
a2b14 00 00 0d 00 00 00 00 00 00 dg 00 00 00 00 00 00 01 00
a1b15 00 0d 10 00 00 00 00 00 00 00 00 00 00 00 00 00 01 10
a1b15 00 00 d0 00 00 00 00 00 00 dg 00 00 00 00 00 00 00 10

```
a0b16  00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 11
a0b16  00 0d 00 00 00 00 00 00 00 dg 00 00 00 00 00 00 00 01
a15b2  00 01 0d 00 00 00 00 00 00 00 00 00 00 00 00 10 10 00
a15b2  00 00 00 00 00 00 00 00 dg gd 00 00 00 00 00 00 00 00
a14b3  00 01 00 0d 00 00 00 00 00 00 00 00 00 00 10 00 10 00
a14b3  00 00 00 00 00 00 00 00 1d g0 d0 00 00 00 00 00 00 00
a13b4  00 01 00 00 0d 00 00 00 00 00 00 00 00 10 00 00 10 00
a13b4  00 00 00 00 00 00 00 01 0d g0 0d 00 00 00 00 00 00 00
a12b5  00 01 00 00 00 0d 00 00 00 00 00 00 10 00 00 00 10 00
a12b5  00 00 00 00 00 00 00 10 0d g0 00 d0 00 00 00 00 00 00
a11b6  00 01 00 00 00 00 0d 00 00 00 00 10 00 00 00 00 10 00
a11b6  00 00 00 00 00 00 01 00 0d g0 00 0d 00 00 00 00 00 00
a10b7  00 01 00 00 00 00 00 0d 00 00 10 00 00 00 00 00 10 00
a10b7  00 00 00 00 00 00 10 00 0d g0 00 00 d0 00 00 00 00 00
a9b8   00 01 00 00 00 00 00 00 0d 10 00 00 00 00 00 00 10 00
a9b8   00 00 00 00 00 01 00 00 0d g0 00 00 0d 00 00 00 00 00
a8b9   00 01 00 00 00 00 00 00 10 0d 00 00 00 00 00 00 10 00
a8b9   00 00 00 00 00 10 00 00 0d g0 00 00 00 d0 00 00 00 00
a7b10  00 01 00 00 00 00 00 10 00 00 0d 00 00 00 00 00 10 00
a7b10  00 00 00 00 01 00 00 00 0d g0 00 00 00 0d 00 00 00 00
a6b11  00 01 00 00 00 00 10 00 00 00 00 0d 00 00 00 00 10 00
a6b11  00 00 00 00 10 00 00 00 0d g0 00 00 00 00 d0 00 00 00
a5b12  00 01 00 00 00 10 00 00 00 00 00 00 0d 00 00 00 10 00
a5b12  00 00 00 01 00 00 00 00 0d g0 00 00 00 00 0d 00 00 00
a4b13  00 01 00 00 10 00 00 00 00 00 00 00 00 0d 00 00 10 00
a4b13  00 00 00 10 00 00 00 00 0d g0 00 00 00 00 00 d0 00 00
a3b14  00 01 00 10 00 00 00 00 00 00 00 00 00 00 0d 00 10 00
a3b14  00 00 01 00 00 00 00 00 0d g0 00 00 00 00 00 0d 00 00
a2b15  00 01 10 00 00 00 00 00 00 00 00 00 00 00 00 0d 10 00
a2b15  00 00 10 00 00 00 00 00 0d g0 00 00 00 00 00 00 d0 00
a1b16  00 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1d 00
a1b16  00 01 00 00 00 00 00 00 0d g0 00 00 00 00 00 00 0d 00
a0b17  10 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 0d
a0b17  00 10 00 00 00 00 00 00 0g d0 00 00 00 00 00 00 00 g0
a15b3  00 01 0d 00 00 00 00 00 00 00 00 00 00 00 10 10 00 00
a15b3  00 00 00 00 00 00 00 0d gg d0 00 00 00 00 00 00 00 00
a14b4  00 01 00 0d 00 00 00 00 00 00 00 00 00 10 00 10 00 00
a14b4  00 00 00 00 00 00 00 01 dg 0d 00 00 00 00 00 00 00 00
a13b5  00 01 00 00 0d 00 00 00 00 00 00 00 10 00 00 10 00 00
a13b5  00 00 00 00 00 00 00 10 dg 00 d0 00 00 00 00 00 00 00
a12b6  00 01 00 00 00 0d 00 00 00 00 00 10 00 00 00 10 00 00
a12b6  00 00 00 00 00 00 01 00 dg 00 0d 00 00 00 00 00 00 00
a11b7  00 01 00 00 00 00 0d 00 00 00 10 00 00 00 00 10 00 00
a11b7  00 00 00 00 00 00 10 00 dg 00 00 d0 00 00 00 00 00 00
a10b8  00 01 00 00 00 00 00 0d 00 10 00 00 00 00 00 10 00 00
a10b8  00 00 00 00 00 01 00 00 dg 00 00 0d 00 00 00 00 00 00
a9b9   00 01 00 00 00 00 00 00 1d 00 00 00 00 00 00 10 00 00
```

a9b9 00 00 00 00 00 10 00 00 dg 00 00 00 d0 00 00 00 00 00
a8b10 00 01 00 00 00 00 00 10 00 0d 00 00 00 00 00 10 00 00
a8b10 00 00 00 00 01 00 00 00 dg 00 00 00 0d 00 00 00 00 00
a7b11 00 01 00 00 00 00 10 00 00 00 0d 00 00 00 00 10 00 00
a7b11 00 00 00 00 10 00 00 00 dg 00 00 00 00 d0 00 00 00 00
a6b12 00 01 00 00 00 10 00 00 00 00 00 0d 00 00 00 10 00 00
a6b12 00 00 00 01 00 00 00 00 dg 00 00 00 00 0d 00 00 00 00
a5b13 00 01 00 00 10 00 00 00 00 00 00 00 0d 00 00 10 00 00
a5b13 00 00 00 10 00 00 00 00 dg 00 00 00 00 00 d0 00 00 00
a4b14 00 01 00 10 00 00 00 00 00 00 00 00 00 0d 00 10 00 00
a4b14 00 00 01 00 00 00 00 00 dg 00 00 00 00 00 0d 00 00 00
a3b15 00 01 10 00 00 00 00 00 00 00 00 00 00 00 0d 10 00 00
a3b15 00 00 10 00 00 00 00 00 dg 00 00 00 00 00 00 d0 00 00
a2b16 00 11 00 00 00 00 00 00 00 00 00 00 00 00 00 1d 00 00
a2b16 00 01 00 00 00 00 00 00 dg 00 00 00 00 00 00 0d 00 00
a1b17 10 01 00 00 00 00 00 00 00 00 00 00 00 00 00 10 0d 00
a1b17 00 10 00 00 00 00 00 00 gd 00 00 00 00 00 00 00 g0 00
a15b4 00 01 0d 00 00 00 00 00 00 00 00 00 00 10 10 00 00 00
a15b4 00 00 00 00 00 00 00 dg gd 00 00 00 00 00 00 00 00 00
a14b5 00 01 00 0d 00 00 00 00 00 00 00 00 10 00 10 00 00 00
a14b5 00 00 00 00 00 00 00 1d g0 d0 00 00 00 00 00 00 00 00
a13b6 00 01 00 00 0d 00 00 00 00 00 00 10 00 00 10 00 00 00
a13b6 00 00 00 00 00 00 01 0d g0 0d 00 00 00 00 00 00 00 00
a12b7 00 01 00 00 00 0d 00 00 00 00 10 00 00 00 10 00 00 00
a12b7 00 00 00 00 00 00 10 0d g0 00 d0 00 00 00 00 00 00 00
a11b8 00 01 00 00 00 00 0d 00 00 10 00 00 00 00 10 00 00 00
a11b8 00 00 00 00 00 01 00 0d g0 00 0d 00 00 00 00 00 00 00
a10b9 00 01 00 00 00 00 00 0d 10 00 00 00 00 00 10 00 00 00
a10b9 00 00 00 00 00 10 00 0d g0 00 00 d0 00 00 00 00 00 00
a9b10 00 01 00 00 00 00 00 10 0d 00 00 00 00 00 10 00 00 00
a9b10 00 00 00 00 01 00 00 0d g0 00 00 0d 00 00 00 00 00 00
a8b11 00 01 00 00 00 00 10 00 00 0d 00 00 00 00 10 00 00 00
a8b11 00 00 00 00 10 00 00 0d g0 00 00 00 d0 00 00 00 00 00
a7b12 00 01 00 00 00 10 00 00 00 00 0d 00 00 00 10 00 00 00
a7b12 00 00 00 01 00 00 00 0d g0 00 00 00 0d 00 00 00 00 00
a6b13 00 01 00 00 10 00 00 00 00 00 00 0d 00 00 10 00 00 00
a6b13 00 00 00 10 00 00 00 0d g0 00 00 00 00 d0 00 00 00 00
a5b14 00 01 00 10 00 00 00 00 00 00 00 00 0d 00 10 00 00 00
a5b14 00 00 01 00 00 00 00 0d g0 00 00 00 00 0d 00 00 00 00
a4b15 00 01 10 00 00 00 00 00 00 00 00 00 00 0d 10 00 00 00
a4b15 00 00 10 00 00 00 00 0d g0 00 00 00 00 00 d0 00 00 00
a3b16 00 11 00 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00
a3b16 00 01 00 00 00 00 00 0d g0 00 00 00 00 00 0d 00 00 00
a2b17 10 01 00 00 00 00 00 00 00 00 00 00 00 00 10 0d 00 00
a2b17 00 10 00 00 00 00 00 0g d0 00 00 00 00 00 00 g0 00 00
a15b5 00 01 0d 00 00 00 00 00 00 00 00 00 10 10 00 00 00 00
a15b5 00 00 00 00 00 00 0d gg d0 00 00 00 00 00 00 00 00 00

a14b6 00 01 00 0d 00 00 00 00 00 00 00 10 00 10 00 00 00 00
a14b6 00 00 00 00 00 00 01 dg 0d 00 00 00 00 00 00 00 00 00
a13b7 00 01 00 00 0d 00 00 00 00 00 10 00 00 10 00 00 00 00
a13b7 00 00 00 00 00 00 10 dg 00 d0 00 00 00 00 00 00 00 00
a12b8 00 01 00 00 00 0d 00 00 00 10 00 00 00 10 00 00 00 00
a12b8 00 00 00 00 00 01 00 dg 00 0d 00 00 00 00 00 00 00 00
a11b9 00 01 00 00 00 00 0d 00 10 00 00 00 00 10 00 00 00 00
a11b9 00 00 00 00 00 10 00 dg 00 00 d0 00 00 00 00 00 00 00
a10b10 00 01 00 00 00 00 00 1d 00 00 00 00 00 10 00 00 00 00
a10b10 00 00 00 00 01 00 00 dg 00 00 0d 00 00 00 00 00 00 00
a9b11 00 01 00 00 00 00 10 00 0d 00 00 00 00 10 00 00 00 00
a9b11 00 00 00 00 10 00 00 dg 00 00 00 d0 00 00 00 00 00 00
a8b12 00 01 00 00 00 10 00 00 00 0d 00 00 00 10 00 00 00 00
a8b12 00 00 00 01 00 00 00 dg 00 00 00 0d 00 00 00 00 00 00
a7b13 00 01 00 00 10 00 00 00 00 00 0d 00 00 10 00 00 00 00
a7b13 00 00 00 10 00 00 00 dg 00 00 00 00 d0 00 00 00 00 00
a6b14 00 01 00 10 00 00 00 00 00 00 00 0d 00 10 00 00 00 00
a6b14 00 00 01 00 00 00 00 dg 00 00 00 00 0d 00 00 00 00 00
a5b15 00 01 10 00 00 00 00 00 00 00 00 00 0d 10 00 00 00 00
a5b15 00 00 10 00 00 00 00 dg 00 00 00 00 00 d0 00 00 00 00
a4b16 00 11 00 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00
a4b16 00 01 00 00 00 00 00 dg 00 00 00 00 00 0d 00 00 00 00
a3b17 10 01 00 00 00 00 00 00 00 00 00 00 00 10 0d 00 00 00
a3b17 00 10 00 00 00 00 00 gd 00 00 00 00 00 00 g0 00 00 00
a15b6 00 01 0d 00 00 00 00 00 00 00 00 10 10 00 00 00 00 00
a15b6 00 00 00 00 00 00 dg gd 00 00 00 00 00 00 00 00 00 00
a14b7 00 01 00 0d 00 00 00 00 00 00 10 00 10 00 00 00 00 00
a14b7 00 00 00 00 00 00 1d g0 d0 00 00 00 00 00 00 00 00 00
a13b8 00 01 00 00 0d 00 00 00 00 10 00 00 10 00 00 00 00 00
a13b8 00 00 00 00 00 01 0d g0 0d 00 00 00 00 00 00 00 00 00
a12b9 00 01 00 00 00 0d 00 00 10 00 00 00 10 00 00 00 00 00
a12b9 00 00 00 00 00 10 0d g0 00 d0 00 00 00 00 00 00 00 00
a11b10 00 01 00 00 00 00 0d 10 00 00 00 00 10 00 00 00 00 00
a11b10 00 00 00 00 01 00 0d g0 00 0d 00 00 00 00 00 00 00 00
a10b11 00 01 00 00 00 00 10 0d 00 00 00 00 10 00 00 00 00 00
a10b11 00 00 00 00 10 00 0d g0 00 00 d0 00 00 00 00 00 00 00
a9b12 00 01 00 00 00 10 00 00 0d 00 00 00 10 00 00 00 00 00
a9b12 00 00 00 01 00 00 0d g0 00 00 0d 00 00 00 00 00 00 00
a8b13 00 01 00 00 10 00 00 00 00 0d 00 00 10 00 00 00 00 00
a8b13 00 00 00 10 00 00 0d g0 00 00 00 d0 00 00 00 00 00 00
a7b14 00 01 00 10 00 00 00 00 00 00 0d 00 10 00 00 00 00 00
a7b14 00 00 01 00 00 00 0d g0 00 00 00 0d 00 00 00 00 00 00
a6b15 00 01 10 00 00 00 00 00 00 00 00 0d 10 00 00 00 00 00
a6b15 00 00 10 00 00 00 0d g0 00 00 00 00 d0 00 00 00 00 00
a5b16 00 11 00 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00
a5b16 00 01 00 00 00 00 0d g0 00 00 00 00 0d 00 00 00 00 00
a4b17 10 01 00 00 00 00 00 00 00 00 00 00 10 0d 00 00 00 00

a4b17 00 10 00 00 00 00 0g d0 00 00 00 00 00 g0 00 00 00 00
a15b7 00 01 0d 00 00 00 00 00 00 00 10 10 00 00 00 00 00 00
a15b7 00 00 00 00 00 0d gg d0 00 00 00 00 00 00 00 00 00 00
a14b8 00 01 00 0d 00 00 00 00 00 10 00 10 00 00 00 00 00 00
a14b8 00 00 00 00 00 01 dg 0d 00 00 00 00 00 00 00 00 00 00
a13b9 00 01 00 00 0d 00 00 00 10 00 00 10 00 00 00 00 00 00
a13b9 00 00 00 00 00 10 dg 00 d0 00 00 00 00 00 00 00 00 00
a12b10 00 01 00 00 00 0d 00 10 00 00 00 10 00 00 00 00 00 00
a12b10 00 00 00 00 01 00 dg 00 0d 00 00 00 00 00 00 00 00 00
a11b11 00 01 00 00 00 00 1d 00 00 00 00 10 00 00 00 00 00 00
a11b11 00 00 00 00 10 00 dg 00 00 d0 00 00 00 00 00 00 00 00
a10b12 00 01 00 00 00 10 00 0d 00 00 00 10 00 00 00 00 00 00
a10b12 00 00 00 01 00 00 dg 00 00 0d 00 00 00 00 00 00 00 00
a9b13 00 01 00 00 10 00 00 00 0d 00 00 10 00 00 00 00 00 00
a9b13 00 00 00 10 00 00 dg 00 00 00 d0 00 00 00 00 00 00 00
a8b14 00 01 00 10 00 00 00 00 00 0d 00 10 00 00 00 00 00 00
a8b14 00 00 01 00 00 00 dg 00 00 00 0d 00 00 00 00 00 00 00
a7b15 00 01 10 00 00 00 00 00 00 00 0d 10 00 00 00 00 00 00
a7b15 00 00 10 00 00 00 dg 00 00 00 00 d0 00 00 00 00 00 00
a6b16 00 11 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00
a6b16 00 01 00 00 00 00 dg 00 00 00 00 0d 00 00 00 00 00 00
a5b17 10 01 00 00 00 00 00 00 00 00 00 10 0d 00 00 00 00 00
a5b17 00 10 00 00 00 00 gd 00 00 00 00 00 g0 00 00 00 00 00
a15b8 00 01 0d 00 00 00 00 00 00 10 10 00 00 00 00 00 00 00
a15b8 00 00 00 00 00 dg gd 00 00 00 00 00 00 00 00 00 00 00
a14b9 00 01 00 0d 00 00 00 00 10 00 10 00 00 00 00 00 00 00
a14b9 00 00 00 00 00 1d g0 d0 00 00 00 00 00 00 00 00 00 00
a13b10 00 01 00 00 0d 00 00 10 00 00 10 00 00 00 00 00 00 00
a13b10 00 00 00 00 01 0d g0 0d 00 00 00 00 00 00 00 00 00 00
a12b11 00 01 00 00 00 0d 10 00 00 00 10 00 00 00 00 00 00 00
a12b11 00 00 00 00 10 0d g0 00 d0 00 00 00 00 00 00 00 00 00
a11b12 00 01 00 00 00 10 0d 00 00 00 10 00 00 00 00 00 00 00
a11b12 00 00 00 01 00 0d g0 00 0d 00 00 00 00 00 00 00 00 00
a10b13 00 01 00 00 10 00 00 0d 00 00 10 00 00 00 00 00 00 00
a10b13 00 00 00 10 00 0d g0 00 00 d0 00 00 00 00 00 00 00 00
a9b14 00 01 00 10 00 00 00 00 0d 00 10 00 00 00 00 00 00 00
a9b14 00 00 01 00 00 0d g0 00 00 0d 00 00 00 00 00 00 00 00
a8b15 00 01 10 00 00 00 00 00 00 0d 10 00 00 00 00 00 00 00
a8b15 00 00 10 00 00 0d g0 00 00 00 d0 00 00 00 00 00 00 00
a7b16 00 11 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00
a7b16 00 01 00 00 00 0d g0 00 00 00 0d 00 00 00 00 00 00 00
a6b17 10 01 00 00 00 00 00 00 00 00 10 0d 00 00 00 00 00 00
a6b17 00 10 00 00 00 0g d0 00 00 00 00 g0 00 00 00 00 00 00
a15b9 00 01 0d 00 00 00 00 00 10 10 00 00 00 00 00 00 00 00
a15b9 00 00 00 00 0d gg d0 00 00 00 00 00 00 00 00 00 00 00
a14b10 00 01 00 0d 00 00 00 10 00 10 00 00 00 00 00 00 00 00
a14b10 00 00 00 00 01 dg 0d 00 00 00 00 00 00 00 00 00 00 00

a13b11 00 01 00 00 0d 00 10 00 00 10 00 00 00 00 00 00 00 00
a13b11 00 00 00 00 10 dg 00 d0 00 00 00 00 00 00 00 00 00 00
a12b12 00 01 00 00 00 1d 00 00 00 10 00 00 00 00 00 00 00 00
a12b12 00 00 00 01 00 dg 00 0d 00 00 00 00 00 00 00 00 00 00
a11b13 00 01 00 00 10 00 0d 00 00 10 00 00 00 00 00 00 00 00
a11b13 00 00 00 10 00 dg 00 00 d0 00 00 00 00 00 00 00 00 00
a10b14 00 01 00 10 00 00 00 0d 00 10 00 00 00 00 00 00 00 00
a10b14 00 00 01 00 00 dg 00 00 0d 00 00 00 00 00 00 00 00 00
a9b15 00 01 10 00 00 00 00 00 0d 10 00 00 00 00 00 00 00 00
a9b15 00 00 10 00 00 dg 00 00 00 d0 00 00 00 00 00 00 00 00
a8b16 00 11 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00
a8b16 00 01 00 00 00 dg 00 00 00 0d 00 00 00 00 00 00 00 00
a7b17 10 01 00 00 00 00 00 00 00 10 0d 00 00 00 00 00 00 00
a7b17 00 10 00 00 00 gd 00 00 00 00 g0 00 00 00 00 00 00 00
a15b10 00 01 0d 00 00 00 00 10 10 00 00 00 00 00 00 00 00 00
a15b10 00 00 00 00 dg gd 00 00 00 00 00 00 00 00 00 00 00 00
a14b11 00 01 00 0d 00 00 10 00 10 00 00 00 00 00 00 00 00 00
a14b11 00 00 00 00 1d g0 d0 00 00 00 00 00 00 00 00 00 00 00
a13b12 00 01 00 00 0d 10 00 00 10 00 00 00 00 00 00 00 00 00
a13b12 00 00 00 01 0d g0 0d 00 00 00 00 00 00 00 00 00 00 00
a12b13 00 01 00 00 10 0d 00 00 10 00 00 00 00 00 00 00 00 00
a12b13 00 00 00 10 0d g0 00 d0 00 00 00 00 00 00 00 00 00 00
a11b14 00 01 00 10 00 00 0d 00 10 00 00 00 00 00 00 00 00 00
a11b14 00 00 01 00 0d g0 00 0d 00 00 00 00 00 00 00 00 00 00
a10b15 00 01 10 00 00 00 00 0d 10 00 00 00 00 00 00 00 00 00
a10b15 00 00 10 00 0d g0 00 00 d0 00 00 00 00 00 00 00 00 00
a9b16 00 11 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00
a9b16 00 01 00 00 0d g0 00 00 0d 00 00 00 00 00 00 00 00 00
a8b17 10 01 00 00 00 00 00 00 10 0d 00 00 00 00 00 00 00 00
a8b17 00 10 00 00 0g d0 00 00 00 g0 00 00 00 00 00 00 00 00
a15b11 00 01 0d 00 00 00 10 10 00 00 00 00 00 00 00 00 00 00
a15b11 00 00 00 0d gg d0 00 00 00 00 00 00 00 00 00 00 00 00
a14b12 00 01 00 0d 00 10 00 10 00 00 00 00 00 00 00 00 00 00
a14b12 00 00 00 01 dg 0d 00 00 00 00 00 00 00 00 00 00 00 00
a13b13 00 01 00 00 1d 00 00 10 00 00 00 00 00 00 00 00 00 00
a13b13 00 00 00 10 dg 00 d0 00 00 00 00 00 00 00 00 00 00 00
a12b14 00 01 00 10 00 0d 00 10 00 00 00 00 00 00 00 00 00 00
a12b14 00 00 01 00 dg 00 0d 00 00 00 00 00 00 00 00 00 00 00
a11b15 00 01 10 00 00 00 0d 10 00 00 00 00 00 00 00 00 00 00
a11b15 00 00 10 00 dg 00 00 d0 00 00 00 00 00 00 00 00 00 00
a10b16 00 11 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00
a10b16 00 01 00 00 dg 00 00 0d 00 00 00 00 00 00 00 00 00 00
a9b17 10 01 00 00 00 00 00 10 0d 00 00 00 00 00 00 00 00 00
a9b17 00 10 00 00 gd 00 00 00 g0 00 00 00 00 00 00 00 00 00
a15b12 00 01 0d 00 00 10 10 00 00 00 00 00 00 00 00 00 00 00
a15b12 00 00 00 dg gd 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b13 00 01 00 0d 10 00 10 00 00 00 00 00 00 00 00 00 00 00

a14b13 00 00 00 1d g0 d0 00 00 00 00 00 00 00 00 00 00 00 00
a13b14 00 01 00 10 0d 00 10 00 00 00 00 00 00 00 00 00 00 00
a13b14 00 00 01 0d g0 0d 00 00 00 00 00 00 00 00 00 00 00 00
a12b15 00 01 10 00 00 0d 10 00 00 00 00 00 00 00 00 00 00 00
a12b15 00 00 10 0d g0 00 d0 00 00 00 00 00 00 00 00 00 00 00
a11b1 00 11 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00
a11b16 00 01 00 0d g0 00 0d 00 00 00 00 00 00 00 00 00 00 00
a10b17 10 01 00 00 00 00 10 0d 00 00 00 00 00 00 00 00 00 00
a10b17 00 10 00 0g d0 00 00 g0 00 00 00 00 00 00 00 00 00 00
a15b13 00 01 0d 00 10 10 00 00 00 00 00 00 00 00 00 00 00 00
a15b13 00 00 0d gg d0 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b14 00 01 00 1d 00 10 00 00 00 00 00 00 00 00 00 00 00 00
a14b14 00 00 01 dg 0d 00 00 00 00 00 00 00 00 00 00 00 00 00
a13b15 00 01 10 00 0d 10 00 00 00 00 00 00 00 00 00 00 00 00
a13b15 00 00 10 dg 00 d0 00 00 00 00 00 00 00 00 00 00 00 00
a12b16 00 11 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00
a12b16 00 01 00 dg 00 0d 00 00 00 00 00 00 00 00 00 00 00 00
a11b17 10 01 00 00 00 10 0d 00 00 00 00 00 00 00 00 00 00 00
a11b17 00 10 00 gd 00 00 g0 00 00 00 00 00 00 00 00 00 00 00
a15b14 00 01 0d 10 10 00 00 00 00 00 00 00 00 00 00 00 00 00
a15b14 00 00 dg gd 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b15 00 01 10 0d 10 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b15 00 00 1d g0 d0 00 00 00 00 00 00 00 00 00 00 00 00 00
a13b16 00 11 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00
a13b16 00 01 0d g0 0d 00 00 00 00 00 00 00 00 00 00 00 00 00
a12b17 10 01 00 00 10 0d 00 00 00 00 00 00 00 00 00 00 00 00
a12b17 00 10 0g d0 00 g0 00 00 00 00 00 00 00 00 00 00 00 00
a15b15 00 01 1d 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a15b15 00 0d gg d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b16 00 11 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b16 00 01 dg 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a13b17 10 01 00 10 0d 00 00 00 00 00 00 00 00 00 00 00 00 00
a13b17 00 10 gd 00 g0 00 00 00 00 00 00 00 00 00 00 00 00 00
a15b16 00 11 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a15b16 00 dg gd 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b17 10 01 10 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a14b17 00 1g d0 g0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a15b17 10 11 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a15b17 0d gg d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b17 11 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b17 01 d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b16 01 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b16 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b15 01 0d 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b15 00 01 d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b14 01 0d 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b14 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

a16b13 01 0d 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b13 00 00 01 d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b12 01 0d 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00
a16b12 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b11 01 0d 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00
a16b11 00 00 00 01 d0 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b10 01 0d 00 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00
a16b10 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00 00
a16b9 01 0d 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00 00
a16b9 00 00 00 00 01 d0 00 00 00 00 00 00 00 00 00 00 00 00
a16b8 01 0d 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00
a16b8 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00 00
a16b7 01 0d 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00
a16b7 00 00 00 00 00 01 d0 00 00 00 00 00 00 00 00 00 00 00
a16b6 01 0d 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00
a16b6 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00 00
a16b5 01 0d 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00 00
a16b5 00 00 00 00 00 00 01 d0 00 00 00 00 00 00 00 00 00 00
a16b4 01 0d 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 00
a16b4 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00 00
a16b3 01 0d 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00
a16b3 00 00 00 00 00 00 00 01 d0 00 00 00 00 00 00 00 00 00
a16b2 01 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00
a16b2 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00 00
a16b1 01 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 00
a16b1 00 00 00 00 00 00 00 00 01 d0 00 00 00 00 00 00 00 00
a16b0 01 0d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10
a16b0 00 00 00 00 00 00 00 00 00 1d 00 00 00 00 00 00 00 00
a17b17 d1 11 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
a17b17 dg d0 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

APPENDIX C

APPENDIX C

SIMULATION PROGRAM FOR TESTING OF THE DKS CHIP

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

static int shift_register [31];

static int *EnL1 = shift_register ;
static int *EXGt = shift_register + 1;
static int *MO   = shift_register + 2;
static int *AO   = shift_register + 3;
static int *Av   = shift_register + 4;
static int *M3I  = shift_register + 5;
static int *M3O  = shift_register + 6;
static int *AT   = shift_register + 7;
static int *Ev   = shift_register + 8;
static int *V    = shift_register + 9;
static int *sc   = shift_register + 10;
static int *Et   = shift_register + 11;
static int *EM   = shift_register + 12;
static int *Madr = shift_register + 13;
static int *T    = shift_register + 14;
static int *TRadr= shift_register + 15;
static int *L    = shift_register + 16;
static int *LRadr= shift_register + 17;
static int *M1   = shift_register + 18;
static int *M2   = shift_register + 19;
static int *A1zd = shift_register + 20;
static int *A1bs = shift_register + 21;
static int *A2   = shift_register + 22;
static int *Gate = shift_register + 23;
static int *CTRL = shift_register + 24;
static int *SGN  = shift_register + 25;
static int *Lc   = shift_register + 26;
static int *Tc   = shift_register + 27;
static int *Ls   = shift_register + 28;
static int *Ts   = shift_register + 29;
static int *Etla = shift_register + 30;

static int tsrl_A, tsrl_B, one_delay[2];
```

```
static int Adder, MPY1, MPY2, LX_latch, LA_latch[3], constants [5];
static int A1_latch, A2_latch, M1_latch, M2_latch, M3_latch;
static int c_register, s_register, register_file[9], angle_register[6];
static int counter, TXPORT, DXPORT, Tx, Dx;

static int sile,din;
static int bus_A, bus_B, Sign_A1, Sign_A2;
static int test_steps, test_vector;

FILE *fp;

/* Program to simulate the testing of the DKS chip */

main()
{

   set_undef();          /* Set all signals to undefined value */
   init_rom ();          /* Initialize ROM contents */

   fp = fopen ( "tvfile" , "r" );    /* Open the test-vector file */

/* Test all modules in parallel where possible */

   test_bus ();
   test_con_logic ();
   test_angle_and_cons ();
   test_M1_M2 ();
   test_A1_A2 ();
   test_LX_creg ();
   test_LA_sreg ();
   test_M3_rfile ();
   test_Dx_Tx ();
   test_MPY1 ();
   test_MPY2 ();
   test_Adder ();

}

test_bus ()
{
   int i;

   printf ("\nTest Bus " );
   test_steps = getvec ();          /* Read number of test patterns */

/* Repeat the following procedure for all test patterns */
```

```
   for ( i=1 ; i<=test_steps ; i++ ) {
     reset_signals ();                /*   Set TSRLs and SRLs */
     tsrl_A = getvec ();
     tsrl_B = getvec ();
     *EnL1 = *EXGt = 1;SRLs

     phase_one ();              /* Apply phase one of system clock */

     tsrl_A = tsrl_B = 0;
     *EnL1 = 0;
     phase_one ();

     dump_scan_path ();              /* Scan-out results */
   }

   return (0);
}

test_con_logic ()
{
   int i;
   printf ("\nTest control logic ");

   reset ();                    /* Reset the control logic counter */

/* Repeat the following procedure 73 times to test each state */

   for (i=0 ; i<=7 ; i++) {
      phase_one ();              /* Apply one system clock */
      phase_two ();
      dump_srls ();              /* Scan out SRL contents  */
      printf ("\n");
   }

   return (0);
}

test_angle_and_cons ()
{
  int i;
  printf ("\nTest angle registers and constants");

  for ( i=0 ; i<=5 ; i++ ) {      /* First set all the angle-registers */
     reset_signals ();
     sile = 1;
     din = getvec ();            /* from the test vector file.        */
     phase_one ();
```

```
    phase_two ();
  }

  for ( i=0 ; i<=5 ; i++ ) {
    reset_signals ();          /* Now transfer each angle-register  */
    *V = *Madr = i;            /* and constant one at a time to     */
    *Av = *EM = *EXGt = 1;     /* TSRLs of bus A and B respectively */
    phase_one ();
    phase_two ();
    dump_scan_path ();         /* and scan-out the results.         */
  };

  return (0);
}

test_M1_M2 ()
{
  int i;

  printf ("\nTest M1 and M2 " );
  test_steps = getvec ();

  for ( i=1 ; i<=test_steps ; i++ ) {
    reset_signals ();
    *M1 = 2;
    *M2 = *EnL1 = *EXGt = 1;
    tsrl_A = getvec ();
    tsrl_B = getvec ();

    phase_one ();
    phase_two ();
    reset_signals ();
    *MO = *EXGt = 1;
    phase_one ();
    phase_two ();
    dump_scan_path ();
  }

  return (0);
}

test_A1_A2 ()
{
  int i;

  printf ("\nTest A1 and  A2 " );
  test_steps = getvec ();
```

```
  for ( i=1 ; i<=test_steps ; i++ ) {
    reset_signals ();
    *A1bs = *A2 = 2;
    *EnL1 = *EXGt = 1;
    tsrl_A = getvec ();
    tsrl_B = getvec ();
    phase_one ();
    phase_two ();
    reset_signals ();
    *AO = *EXGt = 1;
    phase_one ();
    phase_two ();
    dump_scan_path ();
  }

  return (0);
}

test_LX_creg ()
{
  printf ("\nTest LX and C-register " );
  reset_signals ();
  sile = 1;
  din = getvec ();
  tsrl_B = getvec ();
  phase_one ();
  phase_two ();
  reset_signals ();
  *Ev = 1;
  *V = 5;
  phase_one ();
  phase_two ();
  reset_signals ();
  *AT = *EnL1 = *EXGt = *M1 = 1;
  phase_one ();
  phase_two ();
  reset_signals ();
  *EXGt = *MO = *Tc = 1;
  phase_one ();
  phase_two ();

  dump_scan_path ();

  return (0);
}

test_LA_sreg ()
```

```
{
  printf ("\nTest LA and S-register " );
  reset_signals ();
  *Etla = *EXGt = *EnL1 = *AT = 1;
  tsrl_A = getvec ();
  tsrl_B = getvec ();
  phase_one ();
  phase_two ();
  phase_one ();
  phase_two ();
  reset_signals ();
  *A1zd = 1;
  phase_one ();
  phase_two ();
  reset_signals ();
  *EXGt = *AO = *Ts = 1;
  phase_one ();
  phase_two ();

  dump_scan_path ();

  return (0);
}

test_M3_rfile ()
{
  int i;

  printf ("\nTest M3 and Register file " );

  for ( i=0 ; i <= 8 ; i++ ) {
    reset_signals ();
    *M3I = *AT = *EXGt = *EnL1 =  1;
    tsrl_A = getvec ();
    tsrl_B = getvec ();
    *LRadr = i;
    phase_one ();
    phase_two ();
    reset_signals ();
    *M3O = *T = *EXGt = 1;
    *TRadr = i;
    phase_one ();
    phase_two ();

    dump_scan_path ();
  }
```

```
  return (0);
}

test_Dx_Tx ()
{
  printf ("\nTest Dx and Tx " );

  reset_signals ();
  sile = 1;
  din = getvec ();
  *V = 5;
  phase_one ();
  phase_two ();
  reset_signals ();
  *Ev = 1;
  *V = 5;
  phase_one ();
  phase_two ();
  reset_signals ();
  *Et = *EXGt = 1;
  phase_one ();
  phase_two ();

  dump_scan_path ();

  return (0);
}

test_MPY1 ()
{
  int i;
  printf ("\nTest MPY1 " );

  test_steps = getvec ();
  for ( i=1 ; i<=test_steps ; i++ ) {
    reset_signals ();
    *M1 = 2;
    *M2 = *EXGt = *EnL1 = 1;
    tsrl_A = getvec ();
    tsrl_B = getvec ();
    phase_one ();
    phase_two ();
    reset_signals ();
    phase_one ();
    phase_two ();
    reset_signals ();
    *M3O = *EXGt = 1;
```

```
    phase_one ();
    phase_two ();

    dump_B_only ();
  }

  return (0);
}

test_MPY2 ()
{
  int i;
  printf ("\nTest MPY2 " );

  test_steps = getvec ();
  for ( i=1 ; i<=test_steps ; i++ ) {
    reset_signals ();
    *M3I = *EXGt = *EnL1 = 1;
    tsrl_A = getvec ();
    phase_one ();
    phase_two ();
    reset_signals ();
    *A2 = 1;
    phase_one ();
    phase_two ();
    reset_signals ();
    *AO = *EXGt = 1;
    phase_one ();
    phase_two ();

    dump_B_only ();
  }

  return (0);
}

test_Adder ()
{
  int i;
  printf ("\nTest Adder " );

  test_steps = getvec ();
  for ( i=1 ; i<=test_steps ; i++ ) {
    reset_signals ();
    *A1bs = *A2 = 2;
    *EXGt = *EnL1 = 1;
    tsrl_A = getvec ();
```

```
    tsrl_B = getvec ();
    phase_one ();
    phase_two ();
    reset_signals ();
    *Gate = *EXGt = 1;
    phase_one ();
    phase_two ();

    dump_B_only ();
  }

  return (0);
}

/* This function simulates all functions performed in the phase-one
   of the system clock of the DKS chip */

phase_one ()
{
   int i;
   int error_code = 0;

   counter++;

   if ( *EnL1 + *MO + *AO + *Av + *T + *Et > 1 )
     error_code = 1;

   if ( *EnL1 + *MO + *AO + *Tc + *Ts + *EM + *M3O + *Gate + *Et > 1 )
     error_code = 2;

   if ( *EnL1==1 && *EXGt==1 ) {
     bus_A = tsrl_A;
     bus_B = tsrl_B;
   }

   if ( *MO==1 ) {
     bus_A = M1_latch;
     bus_B = M2_latch;
   }

   if ( *AO==1 ) {
     bus_A = A1_latch;
     bus_B = A2_latch;
   }

   if ( *Av==1 ) {
     bus_A = angle_register [*V];
```

```
}

if ( *Tc==1 ) {
   bus_B = c_register;
}

if ( *Ts==1 ) {
   bus_B = s_register;
}

if ( *EM==1 ) {
   bus_B = constants [*Madr];
}

if ( *T==1 ) {
   bus_A = register_file [*TRadr];
}

if ( *M3O==1 ) {
   bus_B = M3_latch;
}

if ( *Gate==1 ) {
   bus_B = Adder;
}

if ( *Et==1 ) {
   bus_A = TXPORT;
   bus_B = DXPORT;
}

if ( sile==1 ) {
   for (i=0 ; i<=4 ; i++)
      angle_register [i] = angle_register [i+1];

   angle_register [5] = din;
}

if ( *Ev==1 ) {
   TXPORT = Tx;
   DXPORT = Dx;
   LX_latch = angle_register [*V];
}

if ( *M1 == 1 ) {
   M1_latch = LX_latch;
}
```

```
    if ( *M1 == 2 ) {
      M1_latch = bus_A;
    }

    if ( *M2==1 ) {
      M2_latch = bus_B;
    }

    one_delay [1] = one_delay [0];
    LA_latch [2] = LA_latch [1];
    LA_latch [1] = LA_latch [0];

    if ( *A1zd && *A1bs )
      error_code = 3;

    if ( *A1zd == 1 ) {
      A1_latch = LA_latch[2];
    }

    if ( *A1zd == 2 ) {
      A1_latch = 0;
    }

    if ( *A1bs == 1 ) {
      A1_latch = Adder;
    }

    if ( *A1bs == 2 ) {
      A1_latch = bus_A;
    }

    if ( *A2 == 1 ) {
      A2_latch = MPY2;
    }

    if ( *A2 == 2 ) {
      A2_latch = bus_B;
    }

    if ( *M3I ) {
      M3_latch = bus_A;
    }
    else {
      M3_latch = MPY1;
    }
```

```
    if ( *Etla==1 ) {
      LA_latch[0] = bus_A;
    }

    if ( ( *AT && *Lc ) || ( *AT && *Ls ) )
      error_code = 4;

    if ( *AT==1 ) {
      c_register = bus_B;
      s_register = bus_B;
      register_file [*LRadr] = bus_B;
    }

    if ( *L==1 ) {
      register_file [*LRadr] = Adder;
    }

    if ( *Lc==1 ) {
      c_register = Adder;
    }

    if ( *Ls==1 ) {
      s_register = Adder;
    }

    if ( ( !*EnL1==1 ) && *EXGt==1 ) {
      tsrl_A = bus_A;
      tsrl_B = bus_B;
    }

    if ( error_code != 0 )
      printf ( "Error code = %d \n", error_code );

    return (0);
}

/* This function simulates all functions performed in the phase-two
   of the system clock of the DKS chip */

phase_two ()
{
    int i;

    bus_A = -1;
    bus_B = -1;

    MPY1 = M1_latch * M2_latch / 2;
```

```
   MPY2 = 2 * M3_latch;

   if ( ! *CTRL ) {
     Sign_A1 = one_delay[1];
   }
   else {
     Sign_A1 = 0;
     Sign_A2 = *SGN;
   }

   if ( Sign_A1==1 )
     A1_latch = -A1_latch;
   if ( Sign_A2==1 )
     A2_latch = -A2_latch;

   Adder = A1_latch + A2_latch;

   Tx = Dx = angle_register [*V];

   for ( i=0 ; i<=30 ; i++ )
     shift_register [i] = counter * 10;

   return (0);
}

/* Scan out (print) the contents of both sets of TSRLs */

dump_scan_path ()
{
   printf ("\ntsrl_A = %d", tsrl_A );
   printf ("\ntsrl_B = %d\n", tsrl_B );

   return (0);
}

/* Scan out (print) the contents of TSRLs of bus B only */

dump_B_only ()
{
   printf ("\ntsrl_B = %d\n", tsrl_B );

   return (0);
}

/* Print contents of all the SRLs in the scan path */

dump_srls ()
```

```
{
   int i;

   for ( i=0 ; i<=30 ; i++ )
      printf (( i%10 == 0 ) ? "\n%d   " : "%d   ", shift_register[i] );
   return (0);
}

/* Reset the control logic counter and all control signals */
reset ()
{
   counter = 0;
   reset_signals ();
   return(0);
}

/* Reset all control signals */
reset_signals ()
{
   int i;
   for ( i=0 ; i<=30 ; i++ ) {
      shift_register [i] = 0;
   }
   *EnL1 = 0;
   *EXGt = 0;
   sile = 0;
   din =0;
   return(0);
}

/* Initialize ROM contents at beginning of the program */

init_rom ()
{
   int i;
   for ( i=0 ; i<=4 ; i++ ) {
      constants [i] = i * 25;
   }
   return(0);
}

/* Set all latches and registers to undefined values. */

set_undef ()
{
   int i;
```

```
    counter = Adder = MPY1 = MPY2 = -1;
    LX_latch = LA_latch [0] = LA_latch [1] = LA_latch [2] = -1;
    A1_latch = A2_latch = -1;
    M1_latch = M2_latch = M3_latch = -1;
    c_register = s_register = -1;
    sile = din = -1;
    tsrl_A = tsrl_B = -1;

    for ( i=0 ; i<=8 ; i++ )
        register_file [i] = -1;
    for ( i=0 ; i<=4 ; i++ )
        angle_register [i] = -1;
    for ( i=0 ; i<=30 ; i++ )
        shift_register [i] = -1;

    return(0);
}

getvec ()
{
    int temp;

    fscanf ( fp, "%d\n", &temp );
    return ( temp );
}
```

BIBLIOGRAPHY

BIBLIOGRAPHY

1. P. K. Lala, "Fault Tolerant and Fault Testable Hardware Design", Prentice Hall, 1985.
2. R. G. Bennetts. "Introduction to Digital Board Testing", Crane, Russack, and Co., Inc., 1982.
3. T.W. Williams, and K. P. Parker, "Design for Testability - A Survey", Proc. of the IEEE, Vol. 71, no. 1, Jan 1983, pp. 98-112.
4. E. R. Huatek, "Integrated Circuit Quality and Reliability", Mercer Dekker, 1987.
5. S. S. Leung, and M. A. Shanblatt, "A VLSI Chip Architecture for the Computation of the Direct Kinematics Solution", Dept. of Electrical Engineering, Michigan State University.
6. R. G. Bennetts, "Design of Testable Logic Circuits", Addison-Wesley Publishing Co., 1984.
7. J. E. Stephenson, and J. Grason, "A Testability Measure for Register Transfer Level Digital Circuits", Proc. 6th IEEE Fault Tolerant Computing Symp., 1976, pp. 101-117.
8. L. H. Goldstein, "Controllability/Observability Analysis for Digital Circuits", IEEE Tran. Circuits and Systems, CAS-26, No. 9, 1979, pp. 1685-1693.
9. R. G. Bennetts, C. M. Maunder, and G. D. Robinson, "CAMELOT: a Computer-Aided Measure for Logic Testability", IEE Proc., Vol. 128, Part E, no. 5, pp. 177-189.
10. I. M. Ratiu, et. al. , "VICTOR: A Fast VLSI Testability Analysis Program" Proc. IEEE Test Conf., 1982, pp. 397-401.
11. J. P. Roth, "Computer Logic, Testing, and Verification", Pitman, 1980.
12. E. J. McCluskey, "Logic Design Principles: with emphasis on Testable Semicustom Circuits", Prentice Hall, 1986.
13. S. Funatsu, N. Wakatsui, and T. Arima, "Test Generation Systems in Japan", Proc. 12th Design Automation Systems, June 1975, pp. 114-122.
14. T. W. Williams and E. B. Eichelberger, "A Logic Design Structure for Testability", Proc. 14th Design Automation Conference, 1977, pp. 463-468.
15. G. H. Stange, "A Test Methodology for Large Logic Networks", Proc. IEEE 15th Design Automation Conference, pp. 103-109.
16. T. P. Singh, S. S. Leung, and M. A. Shanblatt, "Bus Scan Testing for Internal Bus Architecture Systems", Sixth IEEE VLSI Test Conf., Mar. 1988.

17. M. G. Mercer, and M. A. Breuer, "Scan Path with Look Ahead Shifting (SPLASH)", 1986 Intl. Test Conference, pp. 696-704.

18. M. R. Mercer, and V. D. Agrawal, "A Novel Clocking Technique for VLSI Circuit Testability", IEEE J. of Solid State Circuits, Vol. SC-19, No. 2, Apr. 1984, pp. 207-211.

19. H. Audo, "Testing VLSI with Random Access Scan", Digest IEEE Compcon, 1980, 80CH1491-OC, pp. 50-52.

20. James H. Stewart, "Future Testing of Large LSI Cards", Dig. Papers 1977 Semiconductor Test Symp., Oct. 1977, pp. 6-17.

21. B. Konemann, J. Mucha, and G. Zwiekoff, "Built-In Logic Block Observation Technique", Proc. IEEE Test Conf., pp. 37-41.

22. J. Savir, "Syndrome Testable Design of Combinational Circuits", IEEE Trans. Computing, Vol. C-29, June 1980, pp. 442-451.

23. A. K. Susskind, "Testing by Verifying Walsh Coefficients" Proc. 11th Ann. Symp. an Fault Tolerant Computing, June 1981, pp. 206-208.

24. E. J. McCluskey and S. Bozorgui, "Design for Autonomous Test", IEEE Trans. Computing, Vol. C-30, Nov. 1981, pp. 866-875.

25. H. J. Nodig, "Signature Analysis - Concepts, Examples, and Guidelines", Hewlett Packard J., May 1977, pp. 15-21.

26. V. Seshadri, "A Real_Time VLSI Architecture for Direct Kinematics", Robotics and Automation Conf., 1987.

27. C. F. Ruoff, "Fast Trig Functions for Robot Control", Robotics Age in the beginning, ed. by C. T. Helmers, Hayden Book, 1983.

28. S. K. Jain, and V. D. Agrawal, "Test Generation for MOS Circuits Using D-algorithm", Proc. 20th Design Automation Conference, June 1983, pp. 65-70.

29. J. Savir, G. Ditlow, and P. H. Bardell, "Random Pattern Testability", Proc. 13th Intl. Symp. on Fault Tolerant Computing, June 1983, pp.80-89.

30. P. Agrawal, and V. D. Agrawal, "Probalistic Analysis of Random Test Generation Method for Irredundant Combinational Networks", IEEE Trans. Computers, Vol. C-24, No. 7, July 1975, pp.691-695.

31. E. B. Eichelberger, and E. Lindbloom, "Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test, IBM J. of Research and Development, Vol. 27, No. 3, May 1983, pp. 265-272.

32. J. P. Roth, "Diagnosis of Automata Failures - A Calculas and a Method", IBM J. of Research and Development, Vol. 10, July 1966, pp.278-291.

33. P. Goel, and B. C. Rosales, "An Automatic Test Generation System for VLSI Logic Structures", Proc. 18th IEEE Design Automation Conference, 1981, pp. 260-268.

34. V. Agrawal and P. Agrawal, "An Automatic Test Generation System for ILLIAC IV Logic Boards", IEEE Trans. Computers, Vol. C-21, No. 9, Sept. 1972, pp. 1015-1017.

35. Y. H. Levendel, and P. R. Menon, "Test Generation Algorithms for Computers HDLs", IEEE Trans. Comp., Vol. C-31, No. 7, July 1982, pp. 577-588.

36. J. P. Mucha, "VLSI Testing - Problems and Solutions", VLSI 85, Elsivier Science Publishers B. V., 1986, pp. 363-370.

37. Stephen Y. H. Su, and Tonysheng Lin, "Functional Testing Techniques for Digital LSI/VLSI Systems", ACM/IEEE 21st DAC Proc., 1984, pp. 517-528.