Chapter 1

INTRODUCTION

"Quality is what the customer demands and can never be compromised"

The customer has the right to demand fulfilment of his specifications in the product and the manufacturer has to conform to those specifications. Flawless design of the product has the topmost priority, as it ensures the functionality specified by the customer. Once the design is complete and well-tested, it goes for manufacturing at a mass-production level. Since the prototype design has passed the quality benchmark, it is the responsibility of the manufacturer to ensure that each and every device produced thereafter is of the same standard.

When it comes to production on a large scale, testing of the manufactured product becomes crucial. One method of testing is statistical testing, in which random samples from the manufactured products, which represent the entire sample space, are tested. Some companies insist on one hundred percent testing of the products.

Now when a company opts for one hundred percent testing, it is very crucial to save time on the testing procedure. Faster testing enables quicker delivery to the customer, keeping him satisfied about the quality of the product as well. Siemens AG has a worldwide reputation for maintaining high quality standards, and always delivering products to the specifications.

Siemens Nasik Works: The Factory

Siemens Nasik Works was established in the year 1980, as an extension of the global Siemens family. It was originally set up as a unit specialising in the manufacture of Automation and Drives Equipment. With time, products related to the railway industry were added to the impressive list of products. Now the factory specialises in Inverters for Air-Conditioned Coaches, Railway Accident Warning Systems, Motor Drives and Railway Signalling Relays. Indian Railways is the major customer, with some shipments also being exported to Iran.

As a part of the global quality policy, Siemens Nasik Works undertakes one hundred percent testing of the Railway Signalling Relays. With this in mind it was envisioned to develop a testing equipment to meet the following specifications.

The project specification involves the design and implementation of automatic, intelligent and stand-alone test equipment for Siemens Railway Signalling Relays. The tester is expected to give a detailed error report of the relay being tested. The design should be operator-friendly, robust and be able to sustain round-the-clock use. The design is to find a place on the Siemens Nasik Works shop floor.

Chapter 2

THE SIEMENS RAILWAY SIGNALLING RELAY

Relays are electrically controlled switches. A coil pulls in an armature when sufficient current flows through it. The applications of relays include remote switching and high-voltage (or high-current) switching. Since it is important to keep electronic circuits electrically isolated from the high power line, relays are useful to switch high power while keeping control signals electrically isolated. Another major application is in the form of control logic. Relays are used in logic circuits for railway signalling systems to ensure very high reliability and long life. These signalling systems can operate under extreme weather conditions and yet have a life expectancy of twenty field years.

A typical relay may have maximum of eight contacts and two excitation coils. The relay coils operate on either 24 volts or 60 volts DC. The coil position may be different for different relays. The relays may have different contact configurations - number and positions of Normally Open (NO) and Normally Closed (NC) contacts change from variant to variant. A mini-group of relays consists of two such relays. There is one more special type of a mini-group, in which the two individual relays in the group are interlocked. The other relay is coupled to the first with the help of a shaft. When the upper relay is in unenergised condition, the lower relay is in picked up state. When the upper relay is energised, the lower relay goes into unenergised state due to the mechanical coupling.

There are about 35 such different configurations of mini-groups being manufactured. A mini-group forms the basic building block of a complex network that actually functions as an element in the signalling system, called the Big Group. It contains many other components along with the mini-groups, such as transformers and delay elements to implement the complex signalling logic. The mini-groups in Railway Signalling Systems are equivalent to NAND and NOR gates in digital logic.

The contacts of the relay are specially treated to give quiet and reliable operation while switching. The design of the relay is done carefully so as to achieve a long life of one million switching operations. There is a stringent specification for the force required to engage a contact. All these mechanical aspects of the relays are tested before it goes to the wiring field.

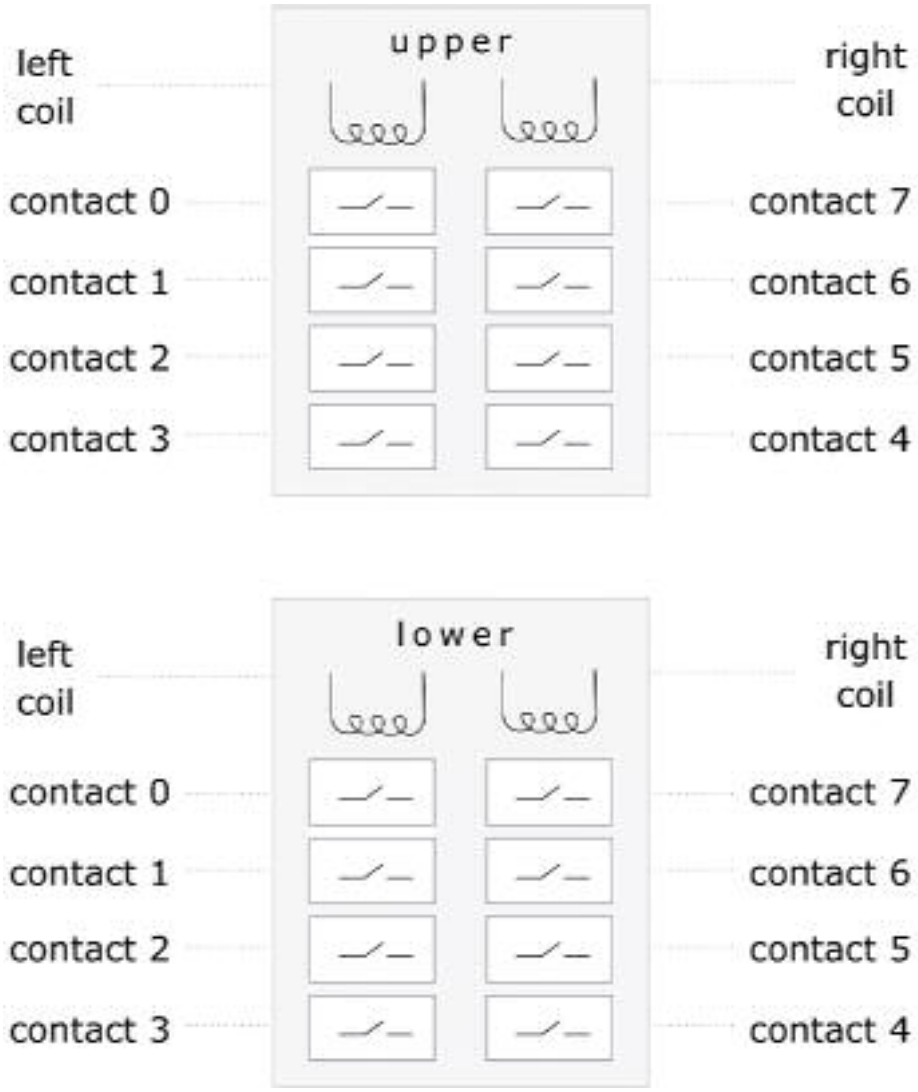

Figure 2.1

All the contact and coil terminals are made available on the front panel for connection. The contact and coil leads are to be soldered manually to the front panel. Different coloured wires are assigned for each contact to simplify the soldering procedure. A contact has both its leads wired to the front panel with same coloured wire. After the wiring is done, the mini-group relay and the panel are grouped together and fitted on a metal chassis. The mini-group is now ready to be used in a more complicated logic network.

Chapter 3

RELAY FAULTS

Relays are susceptible to manufacturing and soldering defects. It is prudent to detect these faults before delivery, considering the critical nature of the intended application of the product. Hence, it is necessary to understand the exact nature of faults that may occur before moving on to the testing principles.

Relay faults can be classified into two broad categories:

1) Wiring Faults

   These are introduced due to errors during the manual soldering of the wires on the front panel. These faults can be further classified into three classes:

   - Input Interchange or Output Interchange

   During soldering it may so happen that the wire to be connected to the input of one contact on the front panel is interchanged with the input of some other contact. This results in the external world signals being routed to the interchanged contacts. The same holds true for the outputs of two contacts.

   - Input-Output Interchange

   Since the colour of wires that bring the contact leads to the front panel is the same, it may happen that the input and output of a given contact are interchanged. As an individual mini-group, this is not a fault as far as the functioning is concerned, but when the same mini-group is used in the big group, it will lead to incorrect behaviour.

   - Input Short or Output Short

   While soldering the wires onto the front panel, it may happen that two or more input wires get shorted. In this case, the signal intended for one particular contact may route through

multiple lines. Also, short connections at inputs may lead to short circuit across supply and ground.

2) Function Faults

These are due to failure of the mechanical assembly of the relay. In this case, the relay does not pick up at all, or a particular contact is faulty. These faults can be further classified into three classes:

- Stuck Contact

A contact may behave as if it is stuck at one position and may fail to get picked up. This is because of the problems during mechanical assembly and the entire relay has to be replaced in such a case.

- Oppositely Behaving Contact

An oppositely behaving contact gets picked up without any problem, but it functions in exactly the opposite fashion. If it is a Normally Open contact, it behaves as a Normally Closed contact and gets open circuited on energising.

- Coil Fault

This is one of the most important faults. The coil gets open circuited, so that even after sending the energising current, the relay fails to pick up. Thus all the NO contacts remain "Open" and all NC contacts remain "Closed" even after energising.

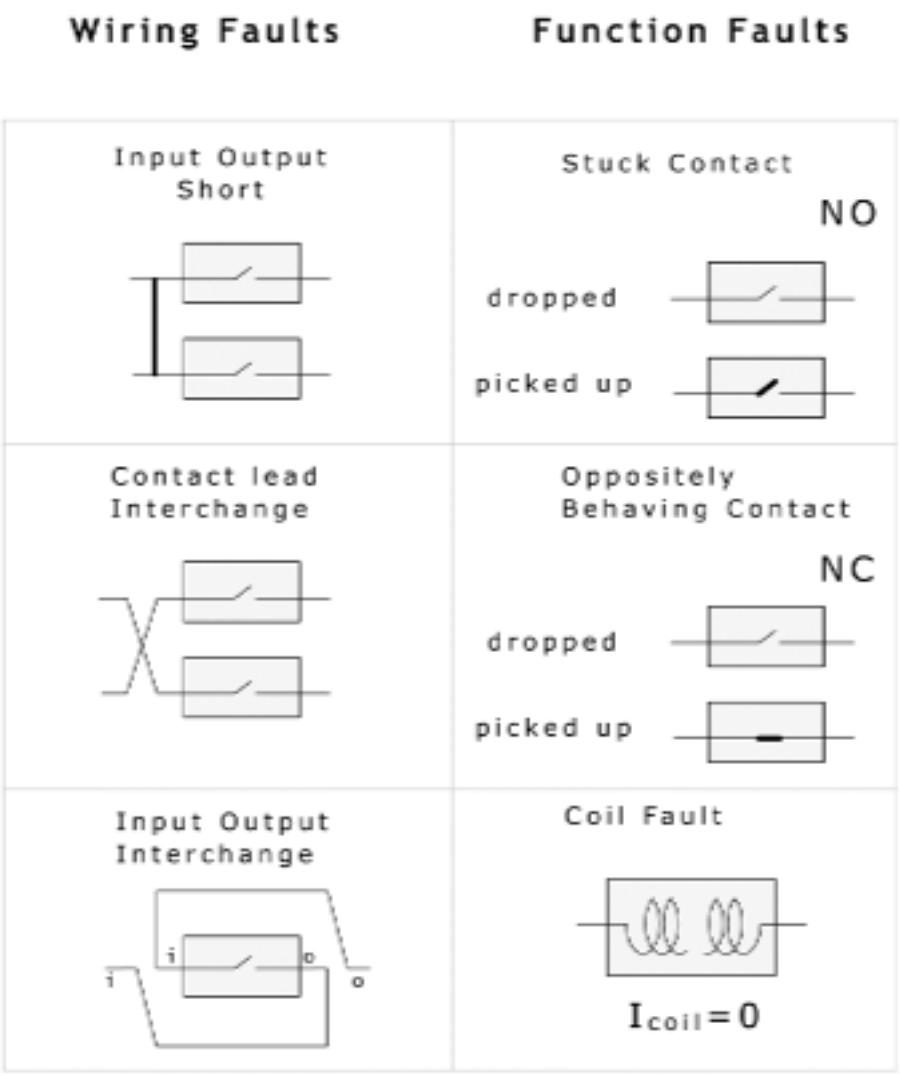

Figure 3.1

The testing system is expected to detect all these faults and locate the positions where they occur. For this, a testing strategy[1] has been evolved.

[1] Details regarding the Test Strategy will be seen in Chapter 5 : The Test Strategy

Chapter 4

THE EXISTING TEST SETUP

The existing test setup for railway signalling relays, at Siemens is a completely electromechanical system. The basic principle of working of the system is comparison between the functionality of the Unit-Under-Test (UUT) and a similar, flawless relay mounted on the jig. The UUT is mounted on the jig and a stimulus voltage gets applied to one end of the contact on the front panel. The operator has to touch each and every contact with a probe. The circuit should get completed only on touching the appropriate contact with the probe. At all other contacts, the circuit should remain open. The system steps to the next stimulus only after successful completion of circuit at the previous contact. The same procedure is followed after energising the relay. If a fault is found at a contact, the system suspends the test sequence, the relay has to be taken out, repaired, and tested again.

The major drawback of the existing system is that there is too much of human intervention involved in the test procedure. Every time, the probe is to be touched to the actual contact for testing that particular contact. This makes the test procedure very slow.

The test system suspends further testing once a fault is found. So, if another mutually exclusive fault is present, the relay will have to be repaired again, which could have been avoided had there been a provision for reporting all the errors at one time only.

The existing test system does not report the type of fault for a particular faulty contact. The operator has to manually find the type of fault that has occurred.

After having understood the exact nature of the faults, and also the drawbacks and limitations of the existing test setup, the project problem definition becomes clearer. Hence we restate the same from a new perspective:

The project specification involves the design and implementation of automatic, intelligent and stand-alone test equipment for Siemens Railway Signalling Relays. The tester is expected to give a detailed error report of the relay being tested. The design should be operator-friendly, robust and be able to sustain round-the-clock use. The design is to find a place on the Siemens, Nasik Works shop floor.

Chapter 5

THE TEST STRATEGY

The basic principle of testing the relay is to check continuity of the contact, based on the energising condition and the contact configuration of the UUT. The simplest solution is to check if the circuit gets completed or not, via the relay contact. A stimulus is applied at one end of the contact and the response is checked at the other end. From this principle, the exact test strategy has been developed.

A logic zero forces all other logic values to zero, when connected as a "Wired-AND" configuration. Hence, both the input and output ports of the UUT have been pulled up to Vcc.

The stimulus applied is in a rippling zero format with a zero being applied to at the most one contact terminal at a given time. This allows us to detect a variety of faults by evaluating the UUT responses at the output as well as the input. A special kind of test called Nail test uses two separate ports for input and output. The ports and their utility in identifying faults has been tabulated below:

Port	**Faults detected**
Relay Input	Input Short, Input Interchange
Relay Output	Output Short, Functional Faults, Output Interchange
Nail Output	Input-Output Interchange

Table 5.1

Various faults explained previously can be detected using this strategy, as follows:

- Input Interchange or Output Interchange

Here two or more input (or output) connections are interchanged while soldering. Therefore, when a zero is applied to one contact via the front panel connection, it actually gets applied to some other contact, which is detected in the input feedback. This appears in the form of a misplaced zero in the input feedback. The contact position where this misplaced zero appears is faulty.

- Input-Output Interchange

Here a special Nail Contact port is employed. The Nail Contacts are clamped laterally to the near end of the actual contacts. The stimulus is applied to the inputs and the same stimulus should appear on these nails, verifying that the input side on the front panel is connected to the inputs of the relay for sure. If the response obtained on the Nail Contact port is not equal to the stimulus applied to the input of the relay, it means that there is input output interchange.

- Input Short or Output Short

Here the zero applied to the relay input port appears at more than one position in the input feedback, indicating that the misplaced zero position is the contact to which the contact under zero stimulus is shorted. Similarly the output port is also stimulated using a shifting zero and corresponding output feedback is checked.

- Stuck Contact

To detect this type of function fault, the shifting zero stimulus is applied to UUT Input port and the corresponding UUT Output port is observed. The UUT is then energised by applying the appropriate supply to appropriate coil. Again the shifting zero stimulus is applied to UUT Input port and response is checked at the UUT Output port. A fault will be detected only for that contact, which is stuck.

- Oppositely Behaving Contact

The test explained above is repeated. In this case, the contact - say an NO contact – but which is behaving in exactly opposite manner will be closed in unenergised condition and will be open on energising the relay. Thus this particular contact is bound to give faults for both, energised and cold conditions.

- Coil Fault

In this fault, the all contacts in the UUT fail to pick up on energising. Therefore, there will be no error for any of the contacts in the cold condition test. All the contacts will give error when the relay is energised, as the relay does not get picked up at all.

This is the strategy that has been implemented in the digital hardware, which forms the topic of the next section.

Chapter 6

IMPLEMENTATION IN HARDWARE

This chapter describes the design choices made regarding the hardware implementation of the testing system. This should serve as the reader's first encounter with the hardware contents of the system.

Considering the number of I/O ports required for testing, there is need for a large number of I/O pins in the system, about 112 of them. A microcontroller or microprocessor based system can not provide such a large number of port I/O pins without port expansion peripherals. The use of peripherals also makes the design of PC board complex. Also the functionality of the I/O pins can not be changed dynamically. There are some specific requirements regarding I/O pins in the system - they should be internally -and externally pulled up. This is not possible with conventional port expansion ICs such as 8255. The use of internal tri-state control avoids the use of external open collector buffers.

In a microprocessor, or a microcontroller, most of the logic functionality is not utilised, e.g. in our system there is no requirement of interrupts or arithmetic logic unit. That means the device is not utilised for the purpose it was designed. Also the logic elements available in a microcontroller are limited, such as limited timers. The digital design for this system needs a lot of timers, with different time outs. The solution to this could have been the use a timer IC externally, which again increases the IC count in the system.

A PC based system with the I/O port expansion cards would be quite uneconomical for the purpose. Also, the operator must have moderate knowledge about the PC.

Considering all these points, it was decided to go for a Programmable Logic Device. The most important advantage is the flexibility of design. The behaviour of each and every block in the PLD can be dictated by writing the code in Hardware Description Language. The use of such a device gives flexibility in terms of I/O pins and each and every I/O pin can be programmed to work as desired. Internal pull-ups, tri-state control, output drive etc. can be specified.

The device that has been chosen is a Spartan II family Field Programmable Gate Array[2].

---

[2] Architecture-specific details of the device will be described in Chapter 18 : The Spartan II FPGA

Board-Level Block Diagram Description

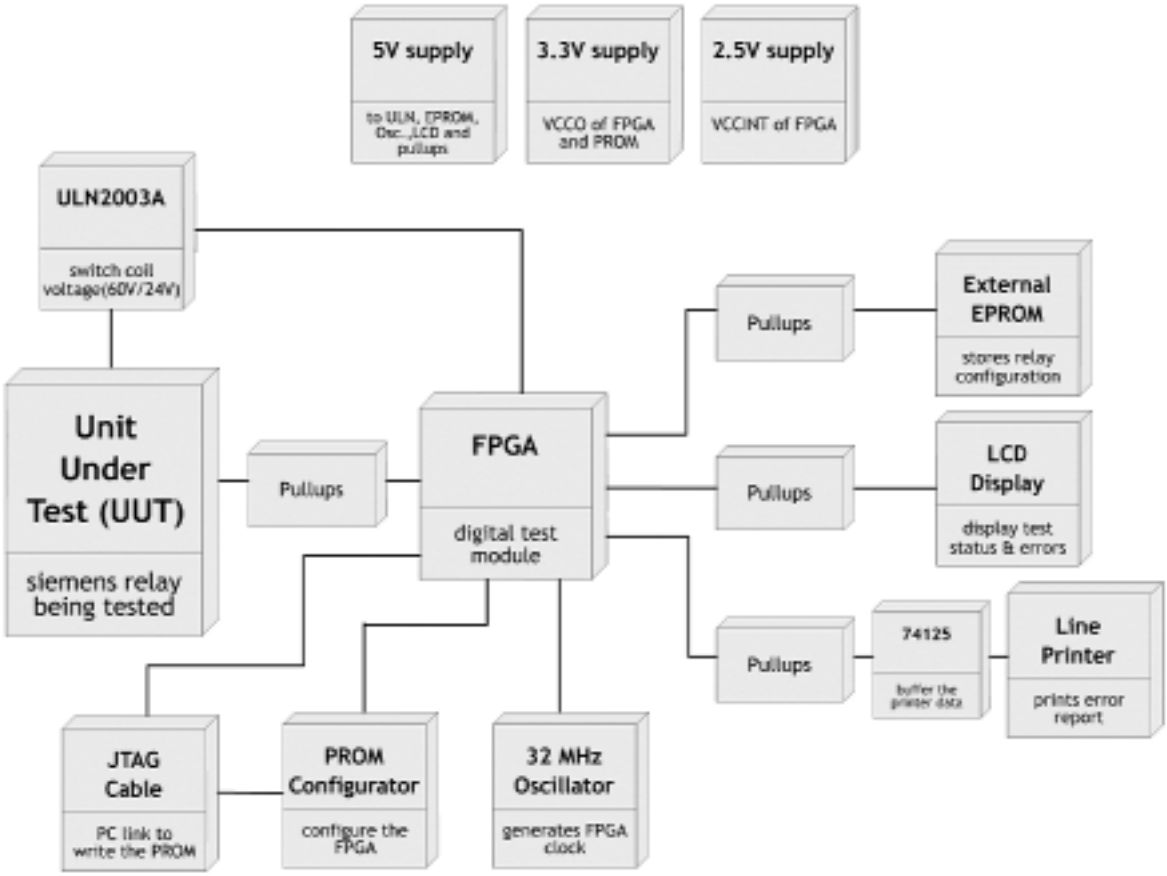

Figure 6.1

- Power supply

The FPGA works on V_{CCINT} of 2.5 volts logic internally and its I/O blocks have V_{CCO} of 3.3 volts. However, other digital ICs such as buffers, relay drivers and the EPROM work on a 5 volts TTL logic supply. All these supplies are derived from a 9 volt input and three LM317 based voltage regulators.

- The FPGA

The FPGA is the heart of the system. The digital system for the testing process is implemented inside the FPGA chip. The FPGA acts as the controller of all the peripheral ICs. It has to float the stimulus patterns on the ports for the testing process. The UUT coils are energised by the FPGA with the help of the small, 5 volt, coil

voltage routing relays, which are driven by a driver – ULN2003A. All the required I/O ports are configured on the FPGA. All these I/O ports are pulled up to 5 volts externally. Even though the FPGA uses 2.5 volts supply for internal operation, and its I/Os require 3.3 volts supply, all the I/Os are 5 volt tolerant and may safely be pulled up to 5 volts. The FPGA is given a clock at 32 MHz, and it is cleaned by the on-chip Delay Locked Loop.

- PROM for Configuration

The PROM contains the configuration data for the FPGA. Every time on power up, the FPGA has to be configured, that is its behaviour has to be loaded into it. The PROM and FPGA exchange handshake signals and initiate the configuration process. The configuration then begins, in which the configuration data is sent to the FPGA bit by bit. Once the FPGA is configured, it starts functioning as per the behaviour specified.

- Library EPROM

There are about 35 different relay variants of mini-groups[3] to be tested using the testing system. The configuration of the UUT must be given to the system by some means. For that, a relay library is generated and burnt into EPROM. The digital system reads the UUT data from the EPROM. The EPROM also contains the ASCII characters to be printed in the error report. The characters to be sent to the LCD display are also stored in this EPROM. Each relay has its own library EPROM and it has to be loaded into the system to test that particular relay. This provides flexibility towards the introduction of new relays into production. Once the library EPROM is modified to suit the newer variant, the testing system can test the latter.

- The Line Printer

The error report can be printed in the form of a ticket containing the test results. A standard parallel line printer with the Centronics interface is used. The printer controller is embedded into the FPGA itself. The data bus and the control signals are buffered by using buffer ICs 74125.

[3] Details on the various types of relays has been described in Chapter 2 : The Siemens Railway Signalling Relay

- The Liquid Crystal Display

A standard 16 x 2 LCD Module is incorporated to display the test results. The LCD module interface is implemented in the FPGA itself. The LCD module has obvious advantages of power saving and programmability.

- The Analog Power Switching Module

The mini-group relays have coil voltages of 24 volts or 60 volts with coil current of about 200mA. There must be some provision for isolation between these higher power loops and the digital logic loops. The analog power switching is done with the help of 5-volt relays, which are available in DIL package. These small relay coils are driven by ULN2003A, which also has in-built back-emf suppression diodes. The ULN provides the energising current for the 5-volt relays, which in turn isolate the digital system from the analog power. The ULN2003A and 5-volt relay system route the 24 volts or 60 volts supply to the UUT coils. The routing of the coil power is controlled by the digital logic inside the FPGA.

After all the components required for the functioning testing system were finalised, the challenge was the design of the digital system that will take care of interfacing of all these components and devices and make them work as desired. The digital system should sequence the test procedure efficiently. There should be room for modification to the design. The digital design is the most challenging aspect of the project, which is described in the section that follows.

Chapter 7

DIGITAL SYSTEM OVERVIEW

Now that the procedure and strategy to be used for testing has been made clear, the reader may find it comfortable delving into the details of how this strategy is to be implemented in programmable logic. The FPGA has several features to offer, some of which have been extensively used in the digital design.

Another advantage that FPGAs offer besides the programmable logic edge, is that the entire design may be partitioned suitably, and then combined together in a hierarchical fashion[4]. This is especially useful in a team project, where separate modules may be independently developed and designed while keeping the other team members informed about only their interfaces. Then these modules are hooked up to each other via these interfaces. Since the functioning of the individual modules is assured, the entire system has to work as specified, provided the modules are correctly interfaced. Thus, the initial problem is broken up into several small problems solved in isolation. This gives much tighter control over the design than with a "global" approach. This "divide-and-conquer" approach has been extensively practised in this design, and will also be followed in the description of the system in this report.

The aim of this chapter is to provide an introduction to the various design units in the system, so that detailed chapters on each unit may follow after that. Before going to the description of the units, one may first find out some properties of the system that are universally applicable.

- The entire design works in synchronism with a global 1 MHz clock. All the state machines, registers, counters, delay elements and other clocked units use the same negative edge of this global clock to synchronise their operations. The external oscillator provides a 32 MHz clock, which is then cleverly divided internally using a Delay Locked Loop (DLL). Since very high speed is not desired, there is no low-end constraint on the clock frequency. 1 MHz has been chosen as it was a convenient value for downstream delay generation.

[4] Details on the partitioned design strategy will be described in Chapter 20 : The FPGA Design Flow

- All design units in the system, with some exceptions, have their own lower level hierarchy, just like the topmost level. Each such hierarchical block consists of a control unit, which is basically a synchronous Finite State Machine (FSM), and is responsible for triggering and sequencing operations within its unit by asserting various control signals for the other logic units within its influence.

- The global reset signal is responsible for determinate reset of all the control units, and bringing the entire system to a known startup state. Those control units then reset the logic units under their control to initialize them. All these reset signals are active high.

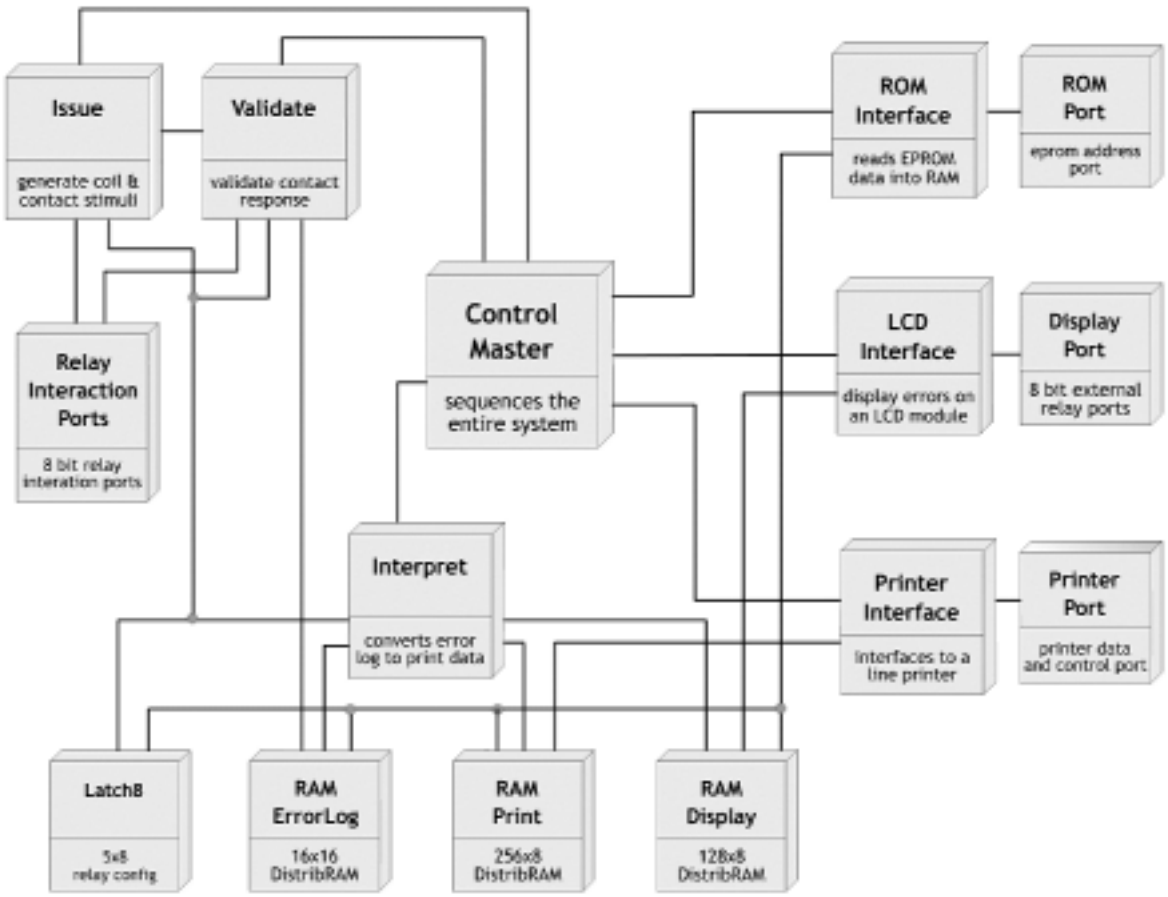

Figure 7.1

The system hierarchy at the top level will now be described.

- ControlMaster

This is the global controller for the system. It is a synchronous FSM, which is responsible for issuing start signals to all the control units according to the test sequence. It basically works in the polling mode; once it issues a start signal to some unit, it waits for that unit to respond with an end signal, indicating the completion of the task, after which the next unit in sequence may be triggered.

- RAM Units

There are three distinct RAM units that are used- RAMPrint, RAMErrorLog and RAMDisplay. The first unit is used to hold the characters to be printed on the test result report. The second unit holds the intermediate error log for each contact of the UUT. This log is a complete description of what went on at that contact for each of the tests that were applied to it. The third unit is similar to the RAMPrint unit, only it contains the characters to be displayed on the LCD Module. The RAMPrint and RAMDisplay units are updated at the end of the testing phase to insert the required characters at appropriate locations.

- RAMRefresh Unit

This unit is the first one to be triggered and is responsible for initializing the contents of all three RAM units to predetermined values[5].

- ROMInterface Unit

This unit forms the external interface to the Library EPROM, from where it loads the configuration bytes, along with characters to be printed and displayed. These characters find their way into their respective RAM units.

[5] The nature of these pre-determined values will become clear in Chapters 9 and 10

- Issue Unit and Validate Unit

The Issue unit and Validate unit perform their jobs in tandem. The Issue unit applies various test patterns to the UUT at various locations, in various conditions, such as coil energised or unenergised. The issuance of these patterns and their sequence is in accordance with the test strategy described in previous chapters. After each test has been issued, the Validate unit reads the output patterns, compares them with expected patterns, and generates an error log for each contact based on the comparison results. This error log is maintained in the RAMErrorLog unit.

- RelayInteractionPorts

This forms the interface between the FPGA and the UUT. This unit is essentially an aggregate of several 8-bit ports that interact directly with the UUT ports. They are of bidirectional type, capable of applying test pattern stimuli as well as receiving response patterns. Since all the ports were already pulled up to 5-volts externally, the FPGA pins were tri-stated internally to obtain logic '1' as outputs.

- Interpret Unit

The Interpret unit reads the error log, and takes an intelligent decision on the type of error that has occurred at each contact. In the event of multiple errors logged by the Validate unit, the Interpret unit decides which errors are actually present, and which have been falsely recorded as a result of the real errors.

- LCDInterface

The LCDInterface unit does the job of communicating the characters to be displayed. These include the welcome message, the messages during testing and the results of the test. The unit first initialises the display module to the desired state, by issuing some command words. Then it starts sending the display characters as and when required. The communication takes place on an 8-bit data bus with the help of three control signals.

- Printer Interface Unit

The Printer Interface Unit establishes an interface with the line printer for printing the test result. It is basically an implementation in hardware of the Centronics Protocol. It takes the characters from the RAMPrint unit, and dumps them onto the printer port. All the printer initialisation, and status checking is done in accordance with the protocol. The communication and handshake take place via an 8-bit data bus using five control lines.

Now that the various design units and their significance has been introduced, a detailed look at all these units may be taken in the next few chapters.

Chapter 8

THE MASTER CONTROLLER

The Master Controller is the part of the system responsible for proper sequencing of operations in the FPGA. It achieves this co-ordination by issuing appropriate signals to the interacting machines in the system at relevant points in time. For smooth functioning of the system, the user must apply some control inputs to the machine. The system operation is managed completely autonomously once the user assigns these required control inputs.

The Entity Structure

The machine requires the user to assert the signals Reset, StartMaster and StartPrinter. Initially the user must makes Reset high for a short period of time to bring the system into the starting state. This also ensures that all the lower level state machines are reset automatically. Thus the entire system is in a predetermined starting state after which determinate system operation is guaranteed. After this the StartMaster signal is to be asserted high by the user at a suitable time (after Reset has been asserted) to commence the testing process. The subsequent machines are then triggered one after the other in a sequential fashion where once the operation of a machine gets over it allows the ControlMaster to enable the next machine for action by asserting its end signal. The StartPrinter signal is required only to enable or disable the printer for operation.

Functioning

- The system wakes up in the Startup state once it is reset. It then waits for the StartMaster signal to go active and until then it keeps looping back to the same state.

- Once the StartMaster signal has been asserted the first phase "Setup" of the machine starts which ensures that the internal RAMs used are initialised to predetermined values. Hence it issues a StartRefresh signal to enable the RAMRefresh unit.

- The RAMRefresh unit signals the end of its job by raising the EndRefresh signal. This then enables the next unit ROMInterface to begin its function of loading the configuration data

pertinent to the UUT being tested from an external EPROM. For this to happen the StartLoad signal is asserted high.

- The ROMInterface issues an EndLoad signal to indicate that the configuration data has been safely loaded into the system's internal memories. This then leads the machine into the next phase of operation "Test" which involves the actual interaction with the UUT. The Issue and Validate units are called one after the other for a maximum possible 176 times with each call representing one stimulus being applied to the UUT.

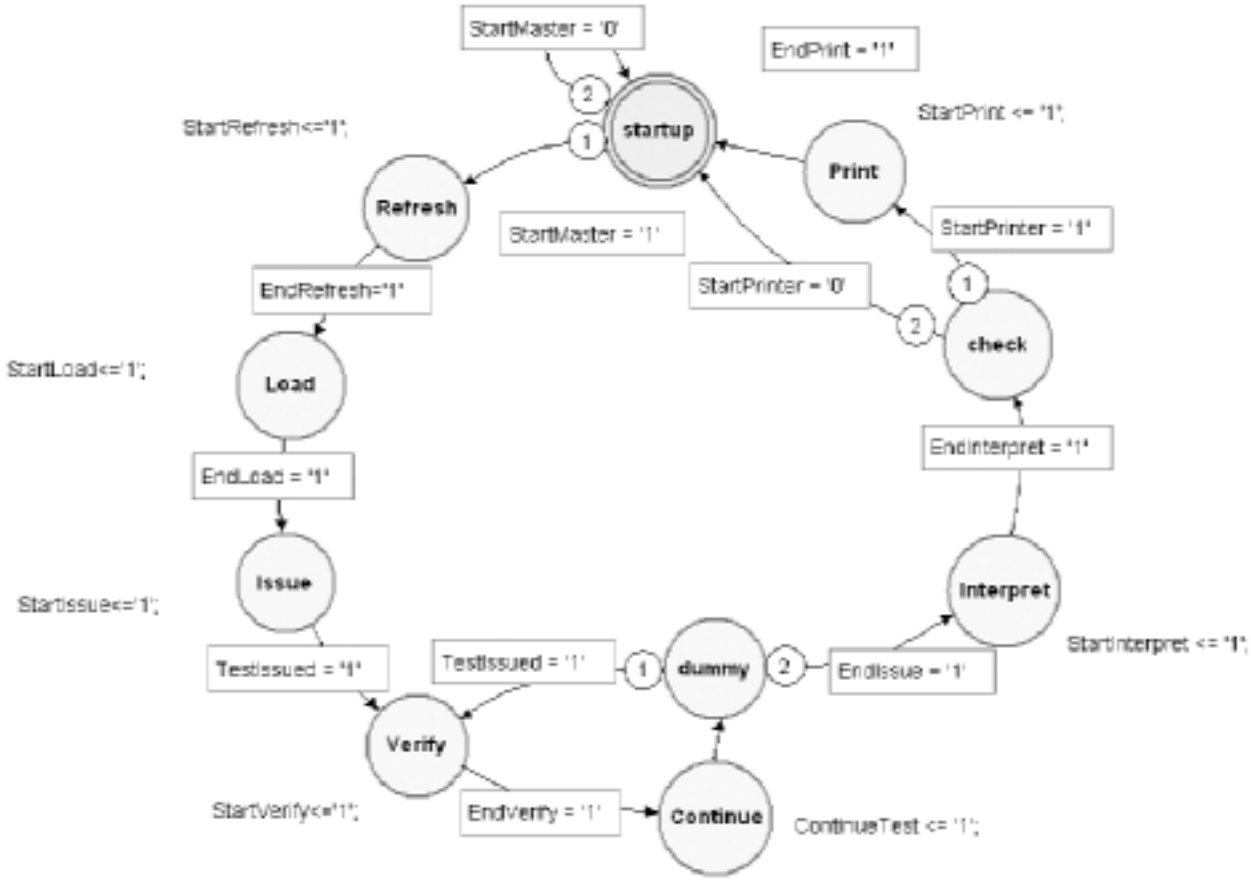

Figure 8.1

- A peculiar mechanism in place for ensuring the smooth application and validation of errors needs some explanation. Once a test pattern has been applied, the Issue unit signals that to the ControlMaster by a signal TestIssued and suspends its own operation. The

ControlMaster then activates the Validate unit by issuing StartVerify which then validates the response of the UUT to the applied stimulus. Once it has finished its job it responds by issuing the VerifyDone signal. This then allows the ControlMaster to issue ContinueTest to the Issue unit to resume operation from its suspended state. This cycle is repeated for a maximum of 176 times as mentioned earlier.

- After the interaction with the relay is done, the Issue unit asserts EndIssue. This leads to the next and most intelligent phase "Error Detection" where the results of the validate unit are correlated to give the actual error present in the Interpret unit. Operation starts when the StartInterpret signal is asserted. Once the error detection is over, the EndInterpret signal is asserted.

- If the StartPrinter signal was asserted and the test returned no errors the Print unit is enabled through the StartPrint signal. The Print unit emulates the Centronics Parallel Port Interface to print the error report on a line printer. Under normal circumstances when the printing of the error report is complete the EndPrint is issued. Any error sensed in the printing operation leads to suspension of the print job and the EndPrint signal is issued prematurely

- All this time the Display unit is active and displaying the progress of the testing process on an LCD module in parallel with the operation of the other units. Its operation is triggered based on the same signals that trigger the operation of ControlMaster.

Chapter 9

THE RAMs

The three RAM units defined in the system help achieve the design objective in a simpler and more organised manner. In fact the presence of the RAMs is not only advantageous but absolutely essential to the operation of the system and forms the backbone of all operations performed. The VHDL file hierarchy declarations of these RAMs are RAMErrorLog, RAMPrint and RAMDisplay. Address busses for all the RAMs are shared between units by using tri-state logic. Similarly the Write Enables for all RAMs are again shared using tri-state logic.

RAMErrorLog

It is a 16 location x 16-bit wide RAM which stores the results of the Validate unit. It is accessed per test and updated accordingly. The 16-bit word is encoded in a format[6] suitable for subsequent processing by the Interpret unit. One 16-bit word is assigned per contact and hence 16 such locations are required. This unit is accessed for write operations by the RAMRefresh and Validate units and for read operation by the Interpret unit. The ErrorLogRAM_WE signal is shared by both RAMRefresh and Validate units and hence tristate logic is used.

RAMPrint

It is a 256 location x 8-bit wide RAM which stores the data that is sent to the printer by the PrintInterface unit. Out of the 256 bytes used the locations with address between 8 and 190 are used for actual printing while the rest are unused. The template used for printing is accessed from the EPROM by the ROMInterface and is updated with the results of the testing process. This unit is accessed for write operations by the ROMInterface and Interpret unit and for read operations by the Print unit. Again using tristate logic the PrintRAM_WE is shared between ROMInterface and Interpret.

[6] The 16-bit error log format will be elaborated in Chapter 14 : The Validate Unit

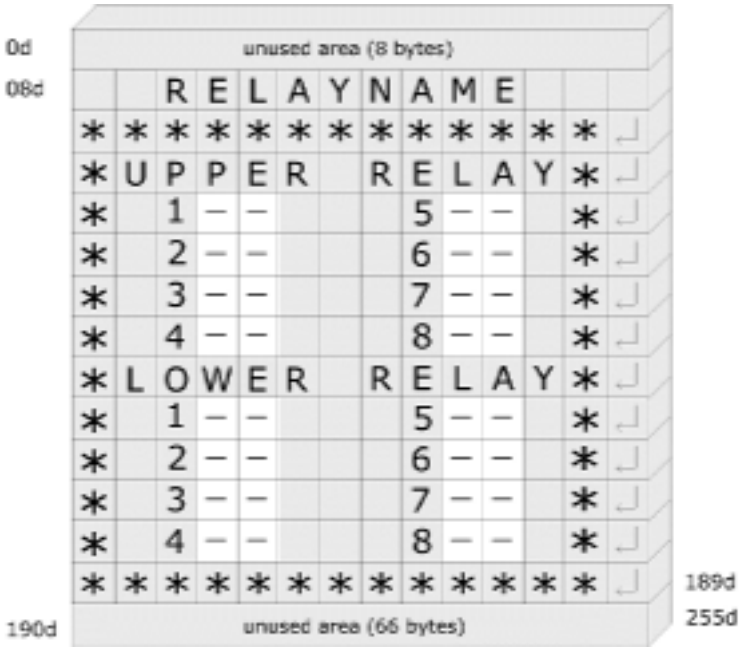

Figure 9.1

RAMDisplay

It is a 128 location x 8-bit wide RAM which stores data which is to be displayed on a 16 x 2 LCD Module. The memory is loaded with message data from an external EPROM and updated with results of the testing process. The data from this RAM is displayed on the LCD Module at appropriate times in the testing cycle. The unit is used for write operations by RAMRefresh and Interpret and read operations by LCDInterface. The DisplayRAM_WE signal is shared between RAMRefresh and Interpret using tri-state logic.

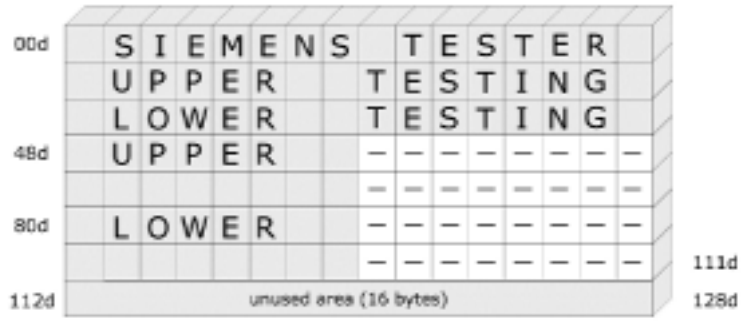

Figure 9.2

Chapter 10

THE RAM REFRESH UNIT

The unit RAMRefresh does the job of initialising the contents of the three internal memories to the predetermined values. This needs to be done prior to the start of every test operation and hence the name "refresh". The memories under the influence of RAMRefresh unit are the 16-word RAMErrorLog and the 128-byte RAMDisplay. RAMErrorLog is cleared to all zeroes while the RAMDisplay is initialised to ASCII "blank space" and "dash" at appropriate locations.

The Entity Structure

The RAMRefresh unit comprises of two sub-units called ControlRAMRefresh and CounterTristate. ControlRAMRefresh is the controller which sequences the memory accesses and writes. It also manages the CounterTristate unit, which does the job of the memory address and data generator, by asserting controls like reset and increment. Thus the RAMRefresh unit has an external interface to the memories being refreshed and the ControlMaster.

Functioning

- The ControlRAMRefresh wakes up in the Startup state once the global reset signal is asserted by the user. It then waits for the StartRefresh signal to be asserted by the ControlMaster. Once the StartRefresh signal has been asserted the CounterTristate is reset through the application of the signal ResetCount.

- The count is then applied to the RAMErrorLog and RAMDisplay address busses. The data relevant to the addresses issued is also supplied by the CounterTristate unit.

- The count is then incremented and checked for overflow. If the count is within the bound of 128 (due to the bound of the RAMDisplay) the loop is continued and if overflow is detected then the machine exits the loop, issues an EndRefresh signal and goes back into its idle state "Startup".

Chapter 11

THE ROM INTERFACE

As explained earlier, there must be some means by which the testing system gets to know which relay it has to test. The relay configuration data is burnt onto an EPROM. Along with the configuration details, ASCII character data such as the name of the relay, error report format are also stored in the EPROM. The last 64 bytes in the EPROM are reserved for ASCII characters required to be displayed on the LCD module. All this data is accessed by the ROMInterface, to transfer it to the internal registers, RAMPrint and RAMDisplay.

The Entity Structure

- ControlROM state machine controls the sequencing of reading data from the EPROM and writing it to corresponding locations in either configuration registers or Print RAM or Display RAM.
- RelayConfigUp, RelayConfigDn, CoilConfig, ContactConfigUp, and ContactConfigDn are the five registers into which the relay configuration data is saved for further sequencing of the testing.
- CounterROM is an 8-bit counter managed by the ControlROM state machine so as to access successive data bytes from the EPROM and RAMPrint.
- CounterDispWR is 7-bit counter used as a pointer into RAMDisplay.

Functioning

- The ROM Interface State Machine starts reading the EPROM once it gets StartLoad signal from the Master Controller. Till then the machine waits in the Wait State, in which all the signals are inactive and wherever required tri-stated to avoid multiple drivers.
- The first 8-byte block is reserved for the configuration data of the relay. The first five bytes read from the EPROM are the relay configuration bytes. These are to be loaded into registers described above. The parallel load signals load1, load2, load3, load4, and load5 load these five bytes into these five registers. The ControlROM machine asserts the RD_L

signal that enables the EPROM data bus, which is directly connected to the RAM write buses. After loading of data into the registers and the RAMs is over, RD_L is inactive, making the data bus tri-stated from the EPROM end.

- The next block of 182 characters contains the ASCII character data, which is useful for printing 13 rows with 14 characters each. The CounterROM address counter is incremented after each byte is read from the EPROM and written to RAMPrint by asserting the PrintRAM_WE signal which enables RAMPrint for write operation. After the 182 characters are read from the EPROM, the CounterROM generates AddressOverflow signal to indicate the end of RAMPrint data.

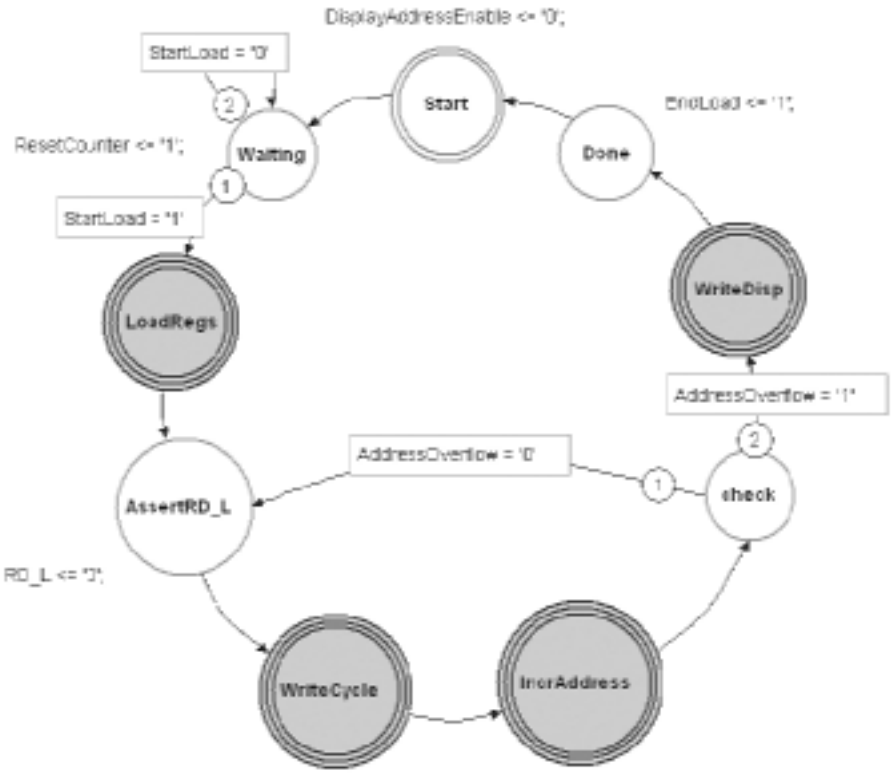

Figure 11.1

- The next 64 bytes in the EPROM contain the ASCII characters for the LCD module. CounterDispWR, which is a 7-bit counter, is used to generate RAMDisplay address. This counter generates DisplayAddrOverflow signal twice in its operation. The ControlROM generates increment address signal to read successive characters from the EPROM and write to RAMDisplay. As shown in the memory map of RAMDisplay, the characters are to

be written into RAMDisplay at irregular addresses. After the first DisplayAddrOverflow is encountered at address "0110110", decimal 54, the ControlROM state machine asserts Preset signal that loads the starting address "1010000", decimal 80, of the next block. Then the address is further incremented to read the remaining characters, until the next DisplayAddrOverflow is asserted by the CounterDispWR, at the count of "1010110", decimal 86. This indicates that all the display characters have been written to RAMDisplay.

- This also indicates that the functioning of the EPROM Interface block is over. The ROMInterface returns the control to the ControlMaster by asserting the EndLoad signal.

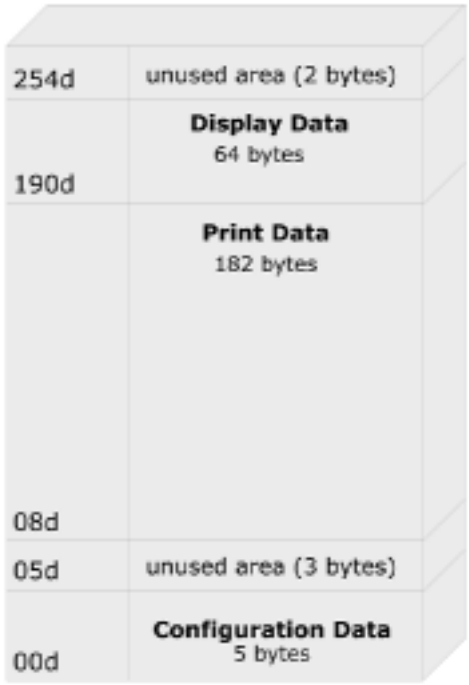

Figure 11.2

C h a p t e r 1 2

THE ISSUE UNIT

The Issue Unit forms the Stimulus generator and router of the test system which is rippling zero or a one-hot encoded 8-bit test pattern. The stimulus is not only generated in the Issue unit but also routed to appropriate test points in the UUT. The job of creating all the required test environments rests with the Issue unit. The Issue unit is triggered by the ControlMaster and after the test vector is applied the TestIssued signal is sent back to the ControlMaster.

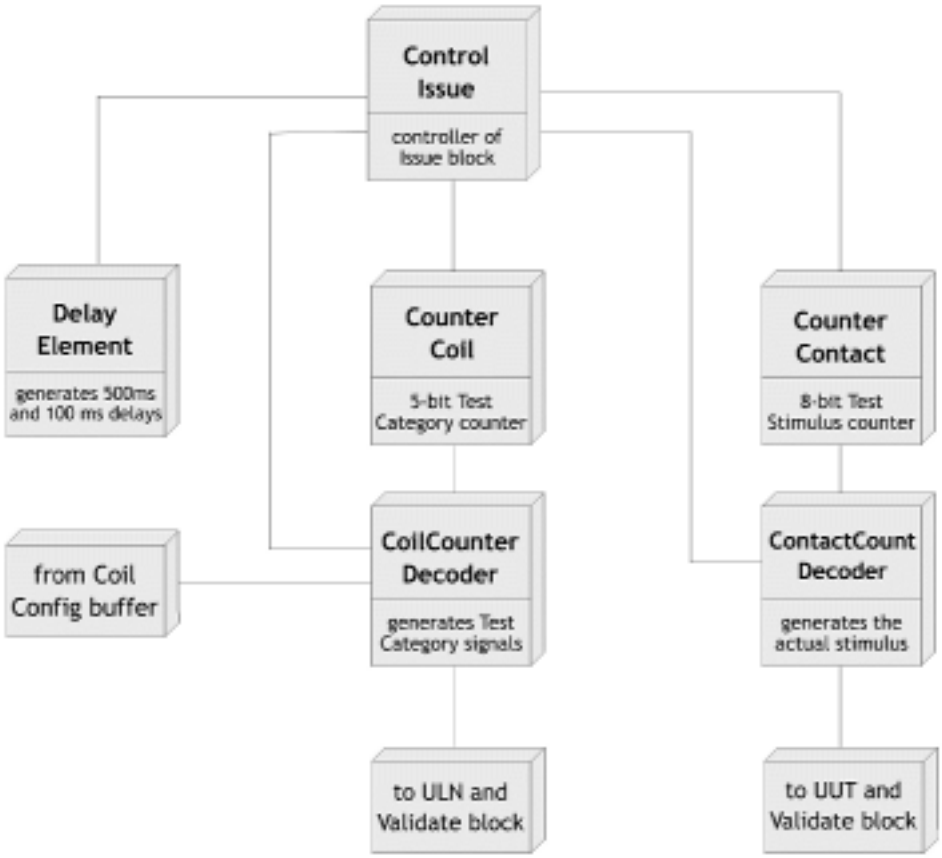

Figure 12.1

The Entity Structure

- ControlIssue

This state machine sequences the operation of the Issue unit.

- CounterCoil and CoilCountDecoder

These units are used to generate all the Test Environments required and to decode the count into the first three Test Environment variables defined below.

- CounterContact and ContactCountDecoder

These units are used to generate a 3-bit, 8-state count and to decode the count to generate the one-hot encoded test stimuli.

- DelayElement

This is an array of flip-flops required to generate 500ms and 100ms delays from the 1 MHz system clock. The 500ms delay is used after applying voltages to the coil circuit. This is to allow the energised coil of the UUT to pick up after current flows through it. This delay is also meant to account for the time required for current build-up in the coil of the UUT from the time of issuing the energising signal. The 100ms delay is a safety delay used to allow the contacts of the UUT to transfer the stimuli.

Defining the Test Environment

The Test Environment can be defined completely and in a simplified manner by the parameters [Test Object], [Test Condition], [Test Mode] and [Test Stimulus].

- [Test Object] Two standard mini-group relays together form one UUT. Separate ports exist for interacting with both these relays although the logic for identifying errors for both remains the same and is distinct. Thus a test object can be defined as the relay being currently tested. The signal UpDnSelect stands for this parameter.

- [Test Condition] Four standard test conditions of Cold Test, Coil 1 Energised Test, Coil 2 Energised Test and Nail Test exist for testing a UUT. They have been classified for ease of error detection and processing. Thus a test condition is defined by the information extracted from the results of the test condition begin applied and its relevance to the interpretation of errors. These conditions are realised by switching the voltages to the appropriate coils and/or stimuli to the appropriate ports. More specifically for Cold Test, none of the coils are energised. For Coil 1 and Coil 2 Tests, voltages are routed to either Coil 1 or Coil 2 respectively. For the Nail test stimulus is switched to the Nail Stimulus Port. TestStatus, a 2-bit array indicates this parameter. NailTest signal is an additional signal that goes high when TestStatus indicates nail test condition.

- [Test Mode] Under first three of the four test conditions stimuli are applied to both the RelayIn and RelayOut ports of the UUT independently. Hence additional logic is required to determine the mode of application of the stimuli. Thus we define mode of application of stimulus as the direction in which the stimulus gets applied to the UUT. InOutSelect is used to identify this parameter.

- [Test Stimulus] For each of the tests all contacts are tested using the one-hot encoded 8-bit test patterns. Thus stimulus is defined by the one-hot encoding of 8-bit test vectors. The 8-bit array Stimulus specifies this parameter.

- An auxiliary parameter of Test Category is defined by Test Object, Test Condition and Test Mode.

Creating the Test Issue Environment

CounterCoil and CounterContact in tandem help create all the 176 possible Test Environments[7]. CounterCoil is a 5-bit counter to generate the required 22 Test Categories. CounterContact is a 3-bit counter to generate 8 test patterns per test category. The ControlIssue unit sequences the CounterContact through all its 8 states per state change of the CounterCoil. The output of the CounterCoil is decoded in the CoilCountDecoder to generate the CoilStimulus signal which is applied to external switches for routing appropriate voltages to the relevant coils. In addition to

[7] Details of the test strategy have been explained in Chapter 5 : The Test Strategy

this, the signals UpDnSelect, InOutSelect and NailTest signals are required to route the Contact stimulus to the proper FPGA port.

Validating the Test Issue Environment

Each of the UUT being tested is unique in its description and does not require all 176 tests to be performed. In fact a few tests are inconsequential and can actually return false errors if performed. Validity of the environment is determined by the CoilConfig array which is loaded from the external EPROM by the ROMInterface. CoilConfig is an 8-bit signal that helps decide if a particular test is required to be carried out. Its bit description is given below:

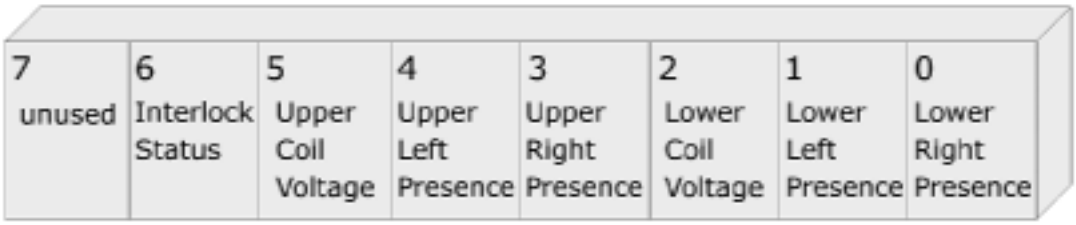

Figure 12.2

- Interlock Status

A special kind of relay called an Interlocked Relay[8] also needs to be tested and requires a minor modification. It has both the upper and lower mini-group relays connected by a shaft such that when either of the coils of the upper or lower relay is energised, the contacts in the entire UUT change their states. When this bit is '1' it indicates that the current UUT is an Interlocked Relay and needs special testing considerations. The Test Issue Environment already has a structure ready for testing this kind of a relay. When the upper relay is being tested, coils of the lower too are activated and the test rerun. Similarly, when the lower relay is being tested the coils of the upper too play a part in the testing process. So, the Test Issue Environment activation sequence has a provision for the Interlocked Relay included already. It is only when the relay is not interlocked do we need to skip these extra Test Issue Environments.

[8] Information of the interlocked relay has been mentioned in Chapter 2 : The Siemens Railway Signalling Relay

- Coil Voltage

A UUT coil can be subjected to 60V or 24V as specified. When this bit is a '1' it indicates that the Coil Voltage is 60V or otherwise it is 24V.

- Coil Presence

Since a UUT can have a maximum of two coils associated with one mini-group relay[9], these bits indicate presence of a coil in the specified locations. When the coils are absent no testing needs to be conducted for those Test Issue Environments and the tests can be skipped.

Thus, the CoilConfig byte can be used effectively to validate the application of a Test Environment. This also highlights the fact that not all 176 tests need to be performed on a UUT and the specifications of that UUT play an important role in the testing process.

Functioning

- The ControlIssue waits for the ControlMaster to send it a StartIssue signal. In the local Startup state, the ControlIssue resets the CounterCoil, ContactStimulus and CoilStimulus. On receiving the StartIssue signal, the Test Category is first indicated by the CoilCountDecoder using the TestValid signal. If this signal is a '1', normal testing can proceed otherwise, the testing is avoided altogether. The CoilStimulus is then applied. The EnableEnergise signal helps avoid the unnecessary 500ms delay that will otherwise be introduced for Test Conditions of Cold Test and Nail Test that do not require energisation of the coil. When this signal is high the DelayElement is called for a 500ms delay.

- Once the DelayElement indicates an overflow, the application of Contact stimulus begins. Initially the CounterContact and the DelayElement are reset. The ContactStimulus is applied and the DelayElement is called for a 100ms delay. After this delay is over, the TestIssued signal is sent to the ControlMaster and operation of the machine is suspended. This allows the ControlMaster to call the Validate unit and verify the UUT responses. When the job of the Validate is done, it informs the ControlMaster which then sends a ContinueTest signal to bring the Issue unit out of its suspended state. Overflow of the

[9] The mini-group finds a mention in Chapter 2 : The Siemens Railway Signalling Relay

CounterContact is sensed and the loop repeated until the overflow is true. In the feedback, the CounterContact is incremented by one to generate the next stimulus.

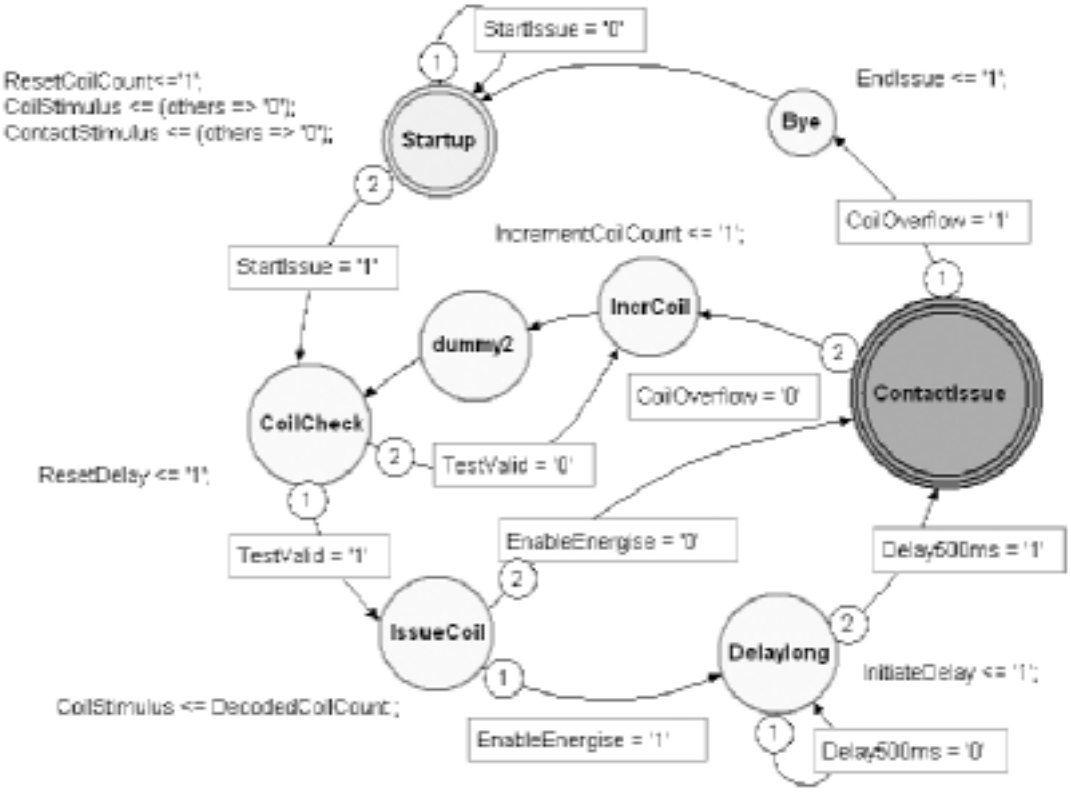

Figure 12.3

- Once an overflow is detected, the CounterCoil is incremented by one and the validity of the incremented count checked in the main loop itself. Thus the looping is continued until all the valid Test Environments have been applied to the UUT and their responses validated. Finally, the EndIssue signal is asserted which signals the completion of interaction with the UUT.

Chapter 13

THE RELAY INTERACTION PORTS

The Relay Interaction Ports form the design's gateway to the UUT. The design interacts, investigates, excites and observes the behaviour of the UUT through these ports. These ports help in applying stimulus to the UUT and also switch voltages to the coil.

The Entity Structure

The RelayInteractionPorts unit comprises of a series of cascaded 8-bit wide demultiplexers which route the ContactStimulus to the appropriate FPGA port and 8-bit wide multiplexers, which route the UUT responses to the Validate unit. The UUT requires one input port, one output port, one nail stimulus port and one nail output port per mini-group relay. Hence a total of eight FPGA ports each of width 8-bit are assigned for relay interaction. The ports have an INOUT behaviour with tri-stated outputs for allowing external control over the logic standard. The TTL 5V logic standard was chosen since the FPGA pins are TTL tolerant[10]. The INOUT behaviour of the ports helps in issuing test vectors to the UUT and receiving the UUT responses from the same port and thus save the I/O resources of the FPGA. It requires the use of UpDnSelect, InOutSelect and NailTest signals for multiplexing. A separate ULNConnect port exists for applying the CoilStimulus signal to the coil voltage switching relays which is also tristated and pulled up externally.

Functioning

The UpDnSelect signal acts as the control signal to map the stimulus and responses to the Upper or Lower FPGA ports. Then the InOutSelect signal helps map the stimulus to the UUT Input or Output ports. It also maps the response to Feedback and RegularOut signals which go to the Validate unit. The NailTest signal maps the stimulus and response to the NailStimulus and NailOut ports respectively.

[10] Information about the FPGA's I/ O capabilities can be found in Chapter 18 : The Spartan II FPGA

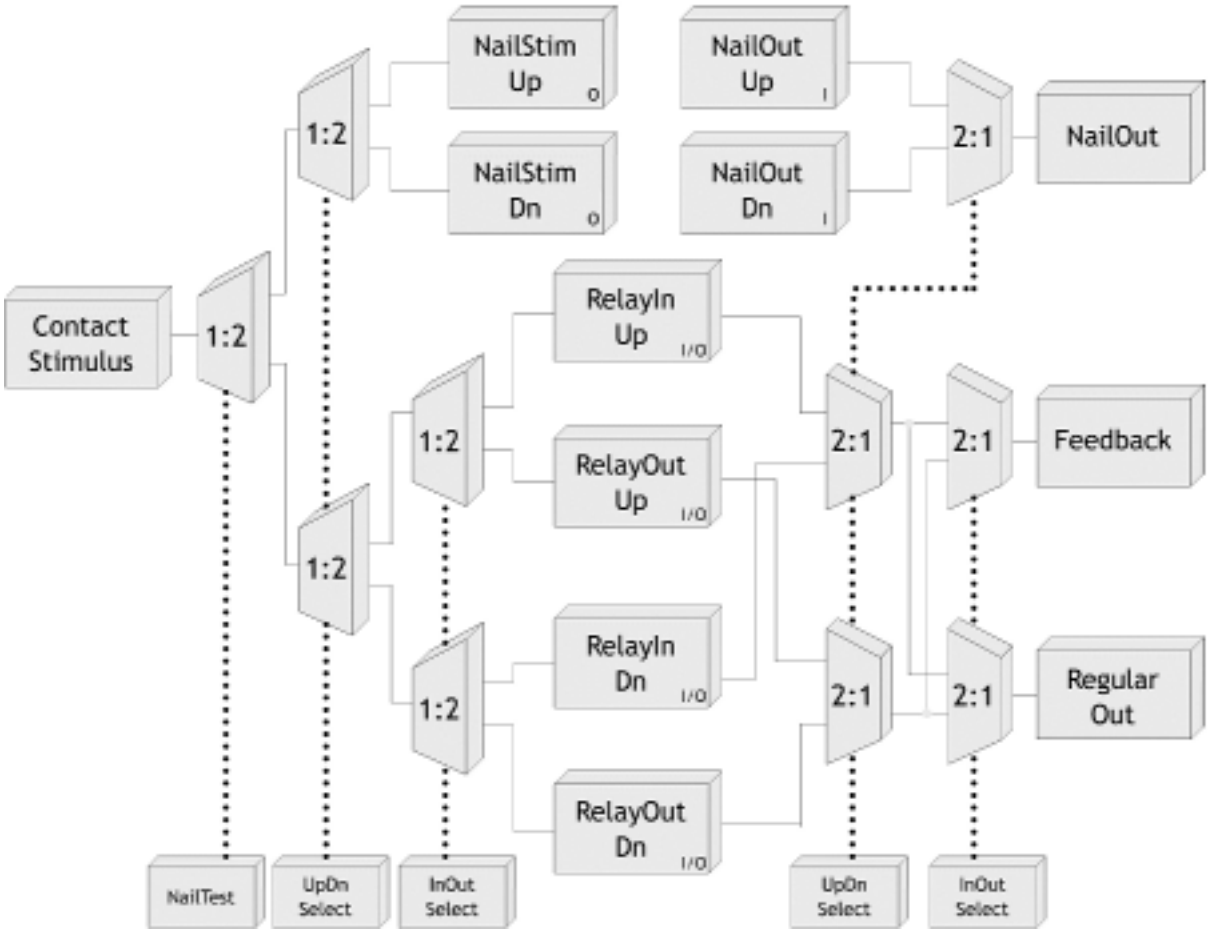
NailStim
Up
O
NailStim
Dn
O
NailOut
Up
I
NailOut
Dn
I
1:2
2:1
NailOut
Contact
Stimulus
1:2
RelayIn
Up
I/O
RelayOut
Up
I/O
RelayIn
Dn
I/O
RelayOut
Dn
I/O
1:2
1:2
1:2
2:1
2:1
2:1
2:1
Feedback
Regular
Out
NailTest
UpDn
Select
InOut
Select
UpDn
Select
InOut
Select

Figure 13.1

Chapter 14

THE VALIDATE UNIT

The Issue Unit and the Validate Unit work in tandem to complete the entire test procedure for the UUT. The Issue Unit asserts a set of signals belonging to a particular Test Environment, and after each of the 176 such assertions[11], the Validate Unit performs its function of reading the results of the test that was just issued, and comparing them with a known pattern of results that are expected of a fault-free relay. The locations of any inconsistencies between the expected and actual patterns help the system in correctly identifying not only the presence of faults, but also the type of fault.

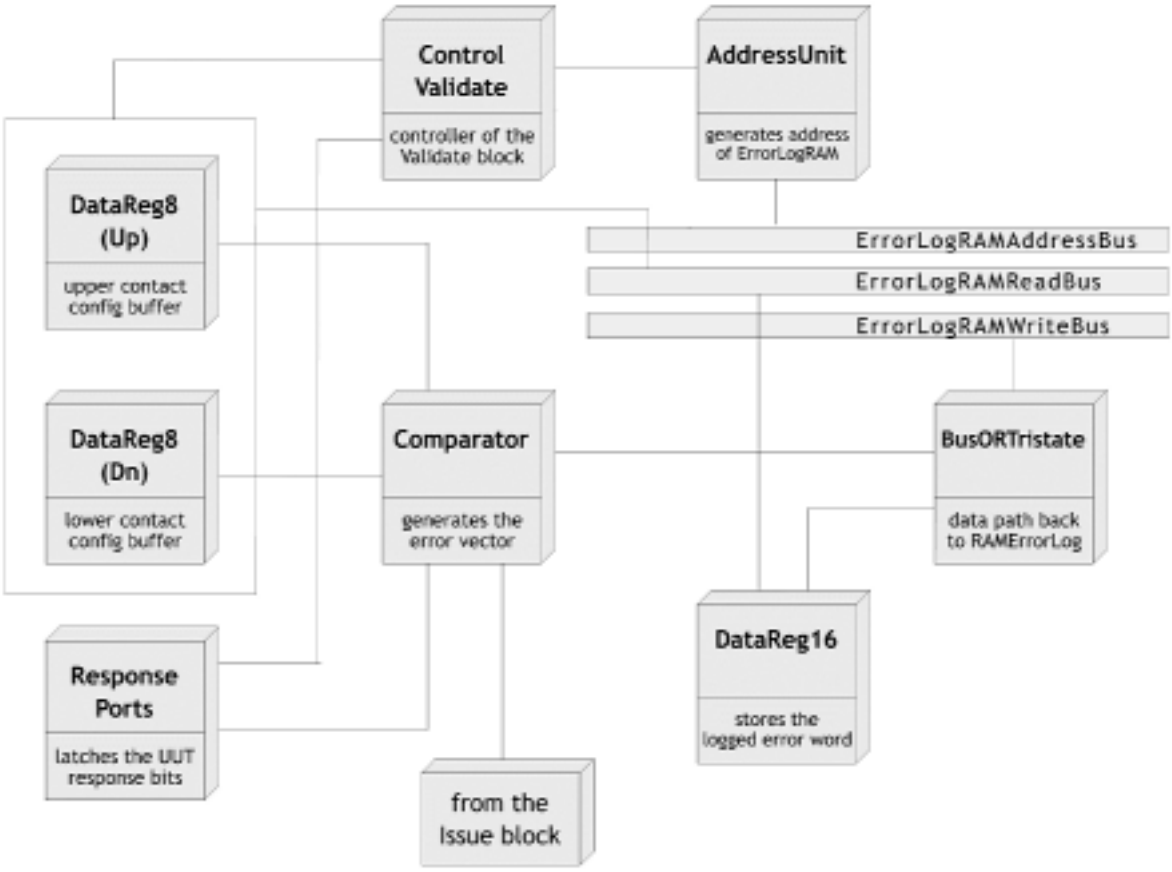

Figure 14.1

[11] The Test Environment has been explained in Chapter 12 : The Issue Unit

The Entity Structure

As in the case of each of the logical units in the system architecture, the Validate Unit has a hierarchical structure, and consists of several logical elements, which together perform the validation function.

- ControlValidate

  This is the unit that issues all the required control signals to every other unit in the Validate Unit. These include control signals to the UUT response ports, signals to the various data registers, memory related signals and others. This section is basically a synchronous FSM, which enters various states based on input status signals, and applies control signals.

- ResponsePorts

  The ResponsePorts unit is basically an aggregate of four 8-bit parallel-in-parallel-out data registers, which are used to hold the bit patterns to be compared. These include the following:

  - RegularOut: The UUT output ports pattern (this may come from either the upper or lower UUT, and will have already been switched appropriately by the Relay Interaction Ports Unit)

  - Feedback: The UUT stimulus feedback port pattern (this may come from either the upper or lower UUT, and may be a result of either the Input or Output Test Mode. In all cases, the appropriate data will have been switched by the Relay Interaction Ports Unit)

  - The UUT Nail output port pattern (which may also come from either the upper or lower UUT, and has been appropriately switched by the Relay Interaction Ports Unit)

  - The applied stimulus bit pattern, which will have been applied at the correct UUT port by the Relay Interaction Port Unit, according to the current Test Environment

- DataReg8

There are three 8-bit data registers which hold the Upper and Lower Relay Contact Configuration bytes. These are used by the Comparator Unit during validation of test results.

- Comparator

The Comparator Unit receives the bit patterns from the Response Ports Unit, and uses the Relay Contact Configuration bytes to perform a validation operation (which will be described shortly). The result of this validation is a 16-bit error code (whose structure will also be explained shortly).

- DataReg16

The DataReg16 register holds the data from the RAMErrorLog for the current contact being tested. The latest error code generated by the Comparator Unit is used to update this existing data (with a logical OR operation) and is stored back to RAM at the same address.

Functioning

- The ControlValidate Unit waits for the Start signal from the ControlMaster, and while in this disabled state, it holds all data registers and ports in the reset condition. Once Start has been issued, it loads the AddressUnit with the appropriate RAMErrorLog address. This address is synthesised from the ContactCount, and the UpDnSelect signal routed from the Issue Unit. After this, the contents of the RAMErrorLog at that address (corresponding to the current contact of the current Test Object according to the Test Environment) are loaded into the 16-bit register.

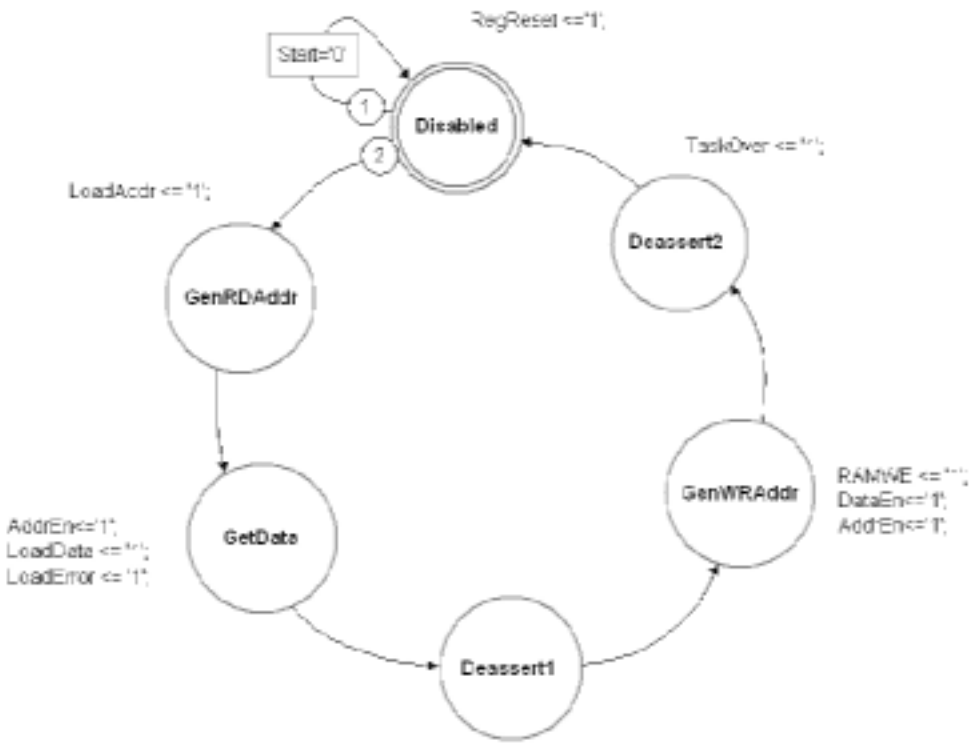

Figure 14.2

As another example, when the stimulus is being applied to the UUT outputs, the InOutSelect signal will be high, and the Comparator will generate the error code in accordance with that. It may also be appreciated, that under either Coil Energised situation, indicated by the TestStatus signal, the Comparator will appropriately invert the relevant Contact Configuration byte before doing anything else.

- At the same time as this is happening, the UUT is expected to have generated the response patterns at its ports, and they have been switched and routed by the RelayInteractionPorts. So they are loaded into the ResponsePorts registers. The contact configuration bytes are also loaded at this time.

- The Comparator now has all the data it requires to perform the validation function. Depending on the Test Environment being currently used (indicated by the TestStatus (2-bit), the NailStatus, the UpDnSelect, and the InOutSelect signals), it will compare relevant port patterns. E.g.: When the cold test is being performed on one of the upper relay contacts, the UUT relay output pattern is compared (XOR-ed) with the OR result of the

upper contact configuration and the applied stimulus pattern. An all-zero result indicates no error at this particular contact during cold condition. The appropriate bit in the error code for this contact is then reset to reflect this.

- The error code generated has a structure defined in the diagram:

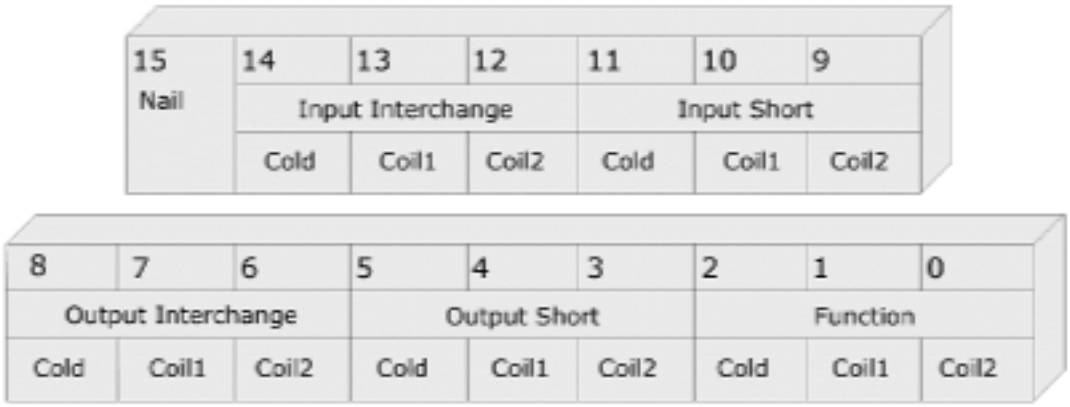

Figure 14.3

- In this table, a zero means no error occurred in that particular Test Environment, while a one indicates an error. E.g. when the cold test was issued for a contact 6, and the stimulus feedback returned two zeroes in the pattern, then either it is an indication of a short at the input of this contact, and the other contact where the extra zero was observed. Accordingly, bit 11 in the error code for contact 6 will be a zero.
- This error code will then be written back to the RAMErrorLog to the address from where it was fetched. Then an indication will be sent to the ControlMaster that the specified task is over. This will cause the ControlMaster to ask the Issue Unit to continue with further tests. After each of those tests, the Issue Unit will stop, the Validate Unit will perform its function of comparison and updating of RAMErrorLog, and this will continue till all the tests have been done. This is the tandem operation which was referred to in the introductory lines of this chapter.
- At this stage, when all possible tests are over, the error codes in RAMErrorLog will be ready, and will finally reflect the results of all those tests. There will be 16 such error codes, one for each contact. These error codes will then be used by the Interpret Unit to perform its function of actually identify the type of error at each contact.

Chapter 15

THE ERROR INTERPRET UNIT

After all the necessary tests have been done and the error log generated for each and every contact, the error log has to be interpreted to generate the error report. The Error Interpreter reads RAMErrorLog and fills in the error report locations in the RAMPrint and RAMDisplay for each and every valid contact of the UUT. While generating the error report, the type of error has to be given priority while decoding the error data.

The reason for prioritising the type of errors is that one type of error may lead to another error. As an example, say there is a wiring error. Due to this wiring error there is a change of reference for further stimuli that will be applied to the UUT. Hence, out-of-place zeroes will obviously be obtained at the output ports. Now this will be interpreted as function error, which is in most likelihood not present at all. To avoid this false interpretation, the error log bits must be read in order of priority.

The Entity Structure

- ControlInterpret, which is the state machine that sequences the interpret operation and generates the required control signals.
- ErrorDecoder is the Truth Table that decodes the Result words into the corresponding ASCII characters used to generate the Print and display data.
- RAMAddressGen generates 4 bit address with inbuilt tristate control, to access the 16-bit Result words from RAMErrorLog.
- CharMux multiplexes two ASCII characters onto one data write bus of RAMPrint.
- Map is another truth table that maps the contact number to the RAMPrint and RAMDisplay addresses where the ASCII characters corresponding to the error report for that contact are to be written. The address bus will be used by other blocks for their functioning and has to be tri-stated.
- TristateBuffAddress block tristates the RAMDisplay address bus and the RAMPrint address bus. TristateBuffData does the same thing for data bus.

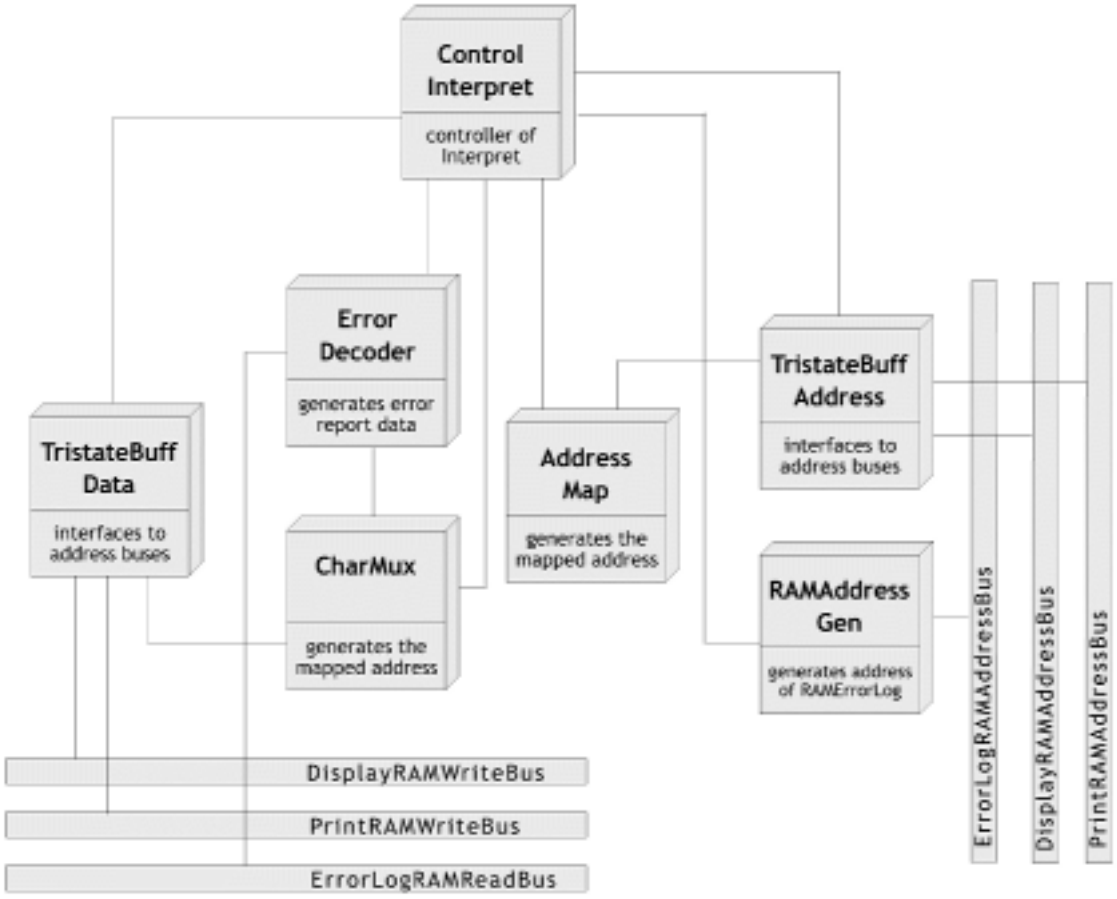

Figure 15.1

As has been explained earlier, the error log is saved in terms of 16-bit word for each contact. Every bit in this 16-bit word[12] represents the result for a test. To implement the priority, the MSB corresponds to the test with highest priority. Then the next significant bit corresponds to the test result with next lower priority and so on. The result of test under Cold, Coil 1 Energised and Coil 2 Energised condition is saved to RAMErrorLog.

First of all Nail port error is checked for. If there is an error here, it is to be reported as Input-Output Interchange only. After this error has been found, there is no point in analysing next LS bits because due to I/O interchange, the stimuli intended for input port is actually going to the relay output port.

[12] The error log word has been described in Chapter 14 : The Validate Unit

The next result bits that are checked are Input Short, Output Short, Input Interchange, and then Output Interchange test result bits which form a part of the Wiring tests. These errors are bound to reflect into the function test. Only if all these wiring tests have succeeded, we can go for analysis of Function test results.

The CoilConfig byte also needs to be taken into consideration while decoding the test results. If a particular coil is not present at all, that particular test will always result in an error, which should not be interpreted as an error for a contact. This case is considered by using CoilConfig as a parameter in the ErrorDecoder. Also, a UUT may have less than 8 contacts in a relay. In that case, the tests on these absent contacts will generate errors. The Map unit takes care of this condition. It does not write the error report in case a contact is not present at all.

Functioning

- The ControlInterpret machine waits in the 'waiting' state until ControlMaster issues StartInterpret signal. In this state the machine ensures that all counters are reset.

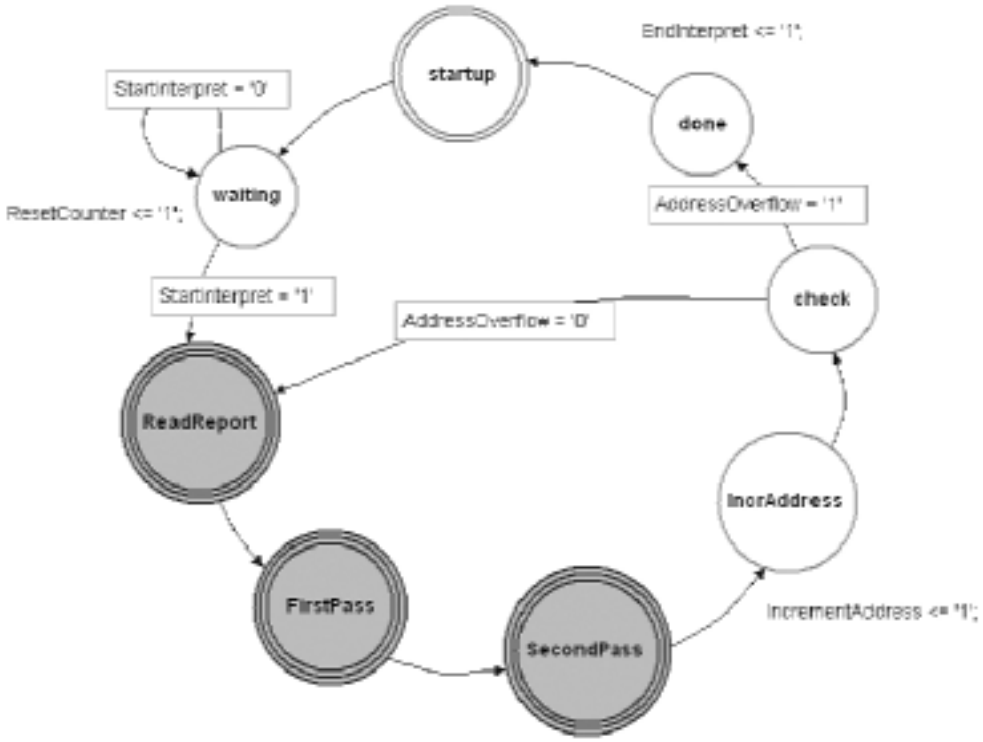

Figure 15.2

- The machine sends the RAMErrorLog address on the address bus and reads the result data for the first contact. This is sent to the ErrorDecoder, a combinatorial circuit that decodes the result as per the priority and gives two ASCII characters as the error report for that contact.
- The current contact address is sent to the Map unit, which is also a combinatorial circuit, which gives the destination addresses in RAMPrint and RAMDisplay. Using Char1 and Char2, which are the ASCII error report characters, and the addresses generated by the Map unit as pointers into RAMPrint and RAMDisplay, the characters are written into the respective RAMs. The same procedure is repeated for all the sixteen contacts for a UUT.
- In the case all the contacts are not present in a UUT, the Map unit, which takes ContactPresence byte into consideration, generates a RAMPrint address, which is not used by the Printer Controller, and hence any error for that contact will not be printed at all. The EPROM print data contains "- -" for the contact that is not present on the UUT at all.
- The RAMAddressGen asserts AddressOverflow signal when all the sixteen result words have been accessed. This indicates the end of the functioning of the ControlInterpret state machine. All the signals are deactivated and tri-stated wherever required. The machine asserts EndInterpret returning the control to the ControlMaster.

Chapter 16

THE PRINTER INTERFACE UNIT

The testing system gives the results in two formats: a visual indication on the Liquid Crystal Display and a printed report on a Line Printer. The operator has a choice of enabling the printed report option. The setting can also be done in such a way that a report is printed only if the UUT has a fault.

The Entity Structure

ControlPrint state machine implements the standard Centronics interface and sends read control signals to the Print RAM. The address pointers for the ASCII characters in the RAMPrint are generated by the AddGenPrinter, which is an 8-bit counter. The AddGenPrinter generates overflow after the required 182 characters have been read and further printed by the line printer.

Functioning

- The ControlPrint state machine waits in the 'Waiting' state until the Master Controller asks it to start functioning by asserting 'StartPrint' signal.

- The ControlPrint then sends SLCT_IN_L signal to the printer to select it. The printer, if connected and powered ON, responds by asserting SLCT.

- After receiving SLCT, the machine resets the printer and clears the Printer Buffer Memory by asserting INIT_L. The printer is now logically connected to the FPGA and is ready to accept the ASCII characters to be printed.

- The machine uses AddGenPrinter output as the pointer into RAMPrint where all the print data has already been generated. The PrintRAMReadBus is directly connected to the printer data bus.

- As per the Centronics standard interface, the data is to be floated on the data bus and the STROBE_L signal is to be asserted after a minimum time of 0.5 μs for it to be recognised correctly by the printer.

- The BUSY signal from the Printer is asserted after it detects the assertion of STROBE_L. The printer interface should not send the next STROBE_L as long as the printer is giving active BUSY signal.

- The machine then increments the RAMPrint address to access next ASCII character. The same procedure is then repeated till all 182 characters have been successfully sent to the printer. This is indicated by AddressOverflow signal generated by the AddGenPrinter. This indicates the end of functioning of the ControlPrint machine. The machine asserts PrintDone signal and returns the control to the ControlMaster.

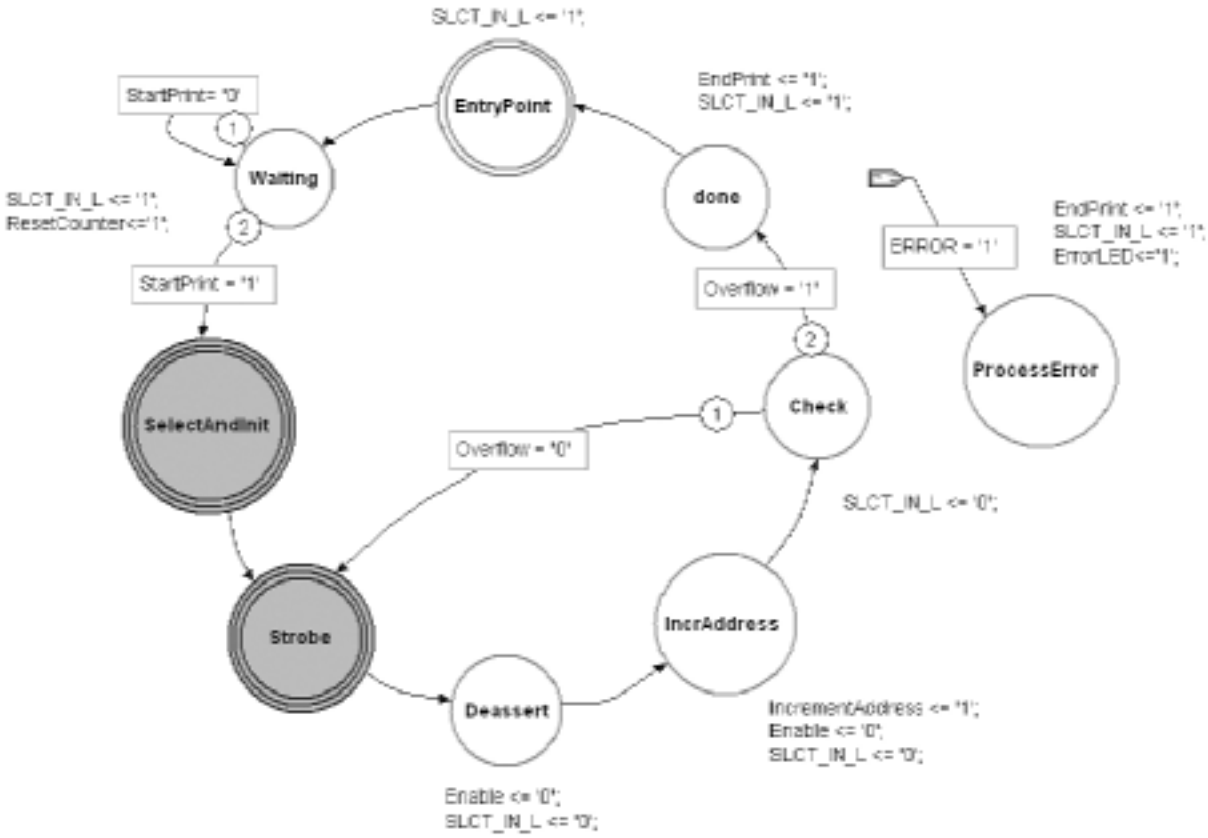

Figure 16.1

- If an error occurs during printing, the printer asserts ERROR signal. If such an error is encountered, the machine suspends printing and goes to "ErrorCondition" state. This error condition is indicated to the operator by lighting ErrorLED. The machine also asserts the PrintDone signal and control returns to ControlMaster.

Chapter 17

LCD INTERFACE

For a visual indication of the results of testing the UUT, an industry-standard Liquid Crystal Display Module has been used. The display is of 16 characters x 2 lines, with an inbuilt ASCII character ROM. The LCD Interface performs the functions of initialising the module to the required modes by sending the appropriate command words, and then sending the display data to it.

There is a particular sequence in which the LCD module is to be initialized. There is a wide variety of modes in which it can be programmed, and thus there are many command words that are used. The DisplayInterface requires an 8-bit data port and three control lines (E, RW, RS) for communication[13] of these command words as well as the display data.

The Entity Structure

- ControlLCDInterface

  This unit is a synchronous FSM, which issues all the control signals required for the operation of various units in the LCD Interface.

- BusyChecker

  This is another synchronous FSM, whose job is to read the LCD Module status byte whenever ordered by the ControlLCDInterface, and check if bit 7 is high, i.e. the LCD Module is busy. Whenever this is so, the BusyChecker will keep performing this read operation till the LCD Module returns a non-busy status. The BusyChecker will then indicate to the ControlLCDInterface that it may go ahead and send any data or commands to the LCD Module.

[13] The communication protocol details were adapted from the LCD module datasheet

- Data Registers

There are two data registers of 8-bits. One of them is used to hold the data to be sent to the LCD Module from the ControlLCDInterface, while the other holds the data that has been read by the BusyChecker.

- Delay Elements

There are two delay elements in the system. One of them is used by the BusyChecker, to generate a 2ms delay, while the other is used by the ControlLCDInterface, to generate delays of 20ms and 2s. They have been kept separate, in order to have independent access for the two FSMs. The number of flip-flops required to generate these delays has been calculated on the basis of the 1 MHz clock frequency.

- ControlMux

This unit is required for proper arbitration of the control signals coming from the ControlLCDInterface and the BusyChecker. Since they are both driving the same FPGA pins, this arbitration becomes necessary. The ControlMux switches between the control signals coming from these two state machines, and applies only the appropriate signals to the FPGA pins.

Functioning

- On power-up, the LCD Module comes on in the default mode of 8 characters per line, 8 bits per character, 5 X 8 font size. We have to change this to our requirement of 16 characters per line, 8 bits per character, 5 X 8 font size. The ControlLCDInterface waits for a StartDisplay signal from the ControlMaster, and then starts the initialization procedure. This consists of sending a few command words to the LCD Module, using a predefined protocol. The ControlLCDInterface synthesises the appropriate command word, and communicates it to the Module using the required protocol on the three control lines. This command word is sent more than once to ensure the correct mode. After each

command word is sent, the BusyChecker does its job of polling the busy bit, and allowing the ControlLCDInterface to go ahead only when the LCD Module is not busy.

- Once the mode has been set, the display is cleared, i.e. the Module display RAM address is reset to the home position. The next command word sets the entry mode to left entry, without shifting and with auto-increment of the display RAM address.

- After the display has been enabled by the next command word, the initialization is over, and the actual characters are sent to the Module. This is done by sending the pre-defined addresses of each character in the character ROM on the Module. For convenience, the addresses have been made identical to the character ASCII codes.

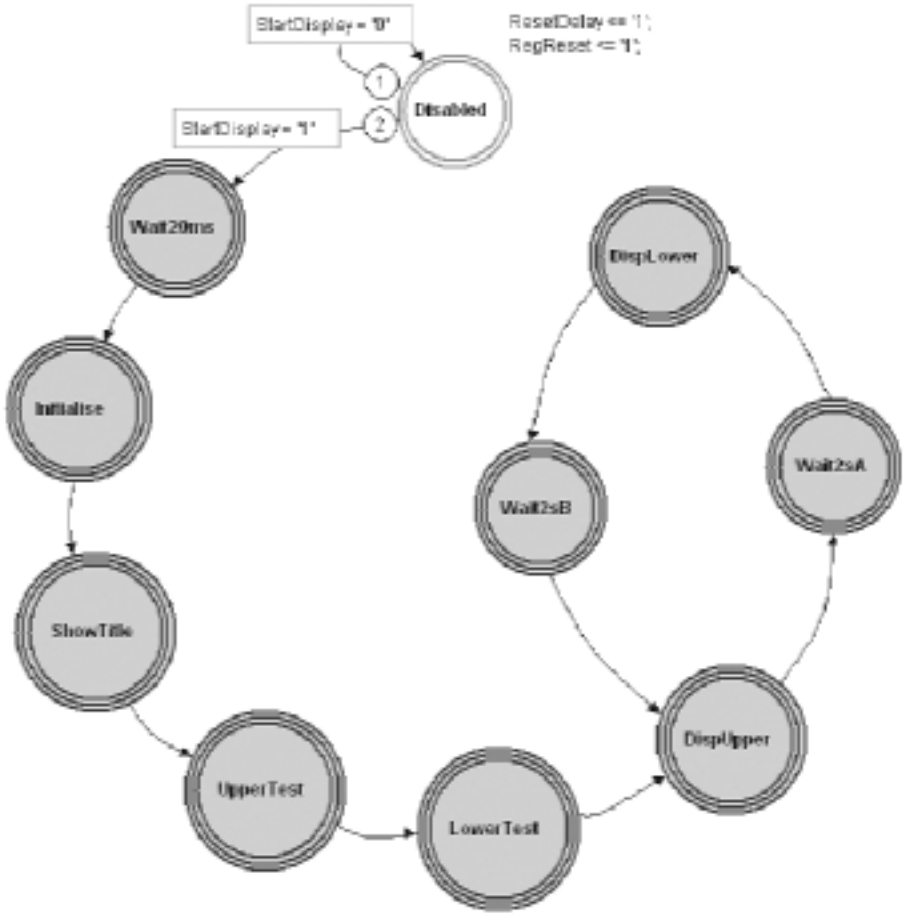

Figure 17.1

- As the initial messages to be displayed have already been stored by the ROMInterface into the RAMDisplay, the ControlLCDInterface has to simply read them sequentially from those locations and dump them onto the LCD data bus. It may also be remembered, that the RAMRefresh Unit has already inserted blank spaces in all the locations where nothing is to be printed.

- The message[14] that is displayed in the beginning is "SIEMENS TESTER". Each character is read from the RAMDisplay, sent to the LCD Module on the data bus accompanied by the assertion of necessary control signals, and waiting for the BusyChecker to give the "go-ahead" for the next character transmission. The entire process takes about 1.5 seconds before the message is flashed.

- When testing actually starts, the messages displayed are "UPPER TESTING", and "LOWER TESTING" as the case may be. Each time, the above procedure is followed.

- After testing is completed, and the Interpret Unit has completed its job of inserting the character codes corresponding to the UUT errors in the RAMDisplay and RAMPrint, the ControlLCDInterface, sensing the EndInterpret signal will go into its final phase of actually displaying the test results. Again, it will only have to get the characters from RAMDisplay and dump them onto the LCD data bus. The RAMDisplay address map has been designed in such a way, that these accesses are sequential for the ControlLCDInterface. Hence, it is the job of the ROMInterface and Interpret Unit to insert the required characters in the correct places. A point worth noting is that when the UUT has fewer than eight contacts, the Validate Unit will find "errors" at the empty contact locations. However, the Interpret unit will already know of such locations, and will not insert error character codes at the corresponding RAMDisplay locations.

- The display for the upper relay result will stay for 2 seconds, followed by the lower relay results. This will continue forever, until the user starts a new test, and the initial messages start again.

[14] The memory map for the displayed messages can be found in Chapter 9 : The RAMs

Chapter 18

THE SPARTAN II FPGA[15]

The Spartan® II 2.5V Field Programmable Gate Array family provides abundant logic resources, a rich feature set at a low price per system gate. The Spartan II XC2S50 family features are shown in the table below-

Device	Logic Cells	System Gates	CLB Array	Total CLBs	Maximum available user I/O	Total Distributed RAM	Total Block RAM
XC2S50	1728	50K	16x24	384	140(PQFP)	24576 bits	32K

Table 18.1

The basic Spartan architecture consists of an array of Configurable Logic Blocks (CLBs) in a sea of interconnects. I/O Blocks (IOBs), on the periphery, map onto the CLBs through these interconnects. The FPGA also features four Delay Locked Loops (DLLs) which provide zero propagation delay, low clock skew and advanced clock domain control. The Distributed and Block RAM features allow on-the-fly implementation of on-chip RAMs. We now discuss a few synthesis oriented features of the Spartan II FPGA that were employed in the design.

Spartan II IOB

A Spartan II IOB consists of inputs and outputs that support a wide variety of signalling standards (Versatile I/O). Every IOB has registers which can be configured as edge triggered flip flops or level sensitive latches. Independent polarity control and optional pull-ups and pull-downs offer an advantage of reduction of external discrete components which becomes evident when the PCB design is considered. The I/O pads are protected from ElectroStatic Discharge (ESD) and over voltage transients. The Spartan II IOBs also support the IEEE 1149.1 Boundary Scan[16] standard. The Versatile I/O feature supports 16 different signalling standards like LVTTL, LVCMOS, PCI, GTL and HSTL. The system design uses LVTTL which is an EIA/JESDSA standard for 3.3V

[15] Architectural details of the Spartan II were studied from the device datasheet

[16] Boundary Scan will be encountered again in Chapter 19 : FPGA Configuration

applications. The standard uses a LVTTL input buffer, push-pull output buffer and a 3.3V V_{CCO} source. During synthesis LVTTL is inferred by default when no specific standard is mentioned but can be specifically inferred by using the IBUF library symbol. Other standards are the internationally accepted bus standards used for high speed signalling on the Printed Circuit Board tracks.

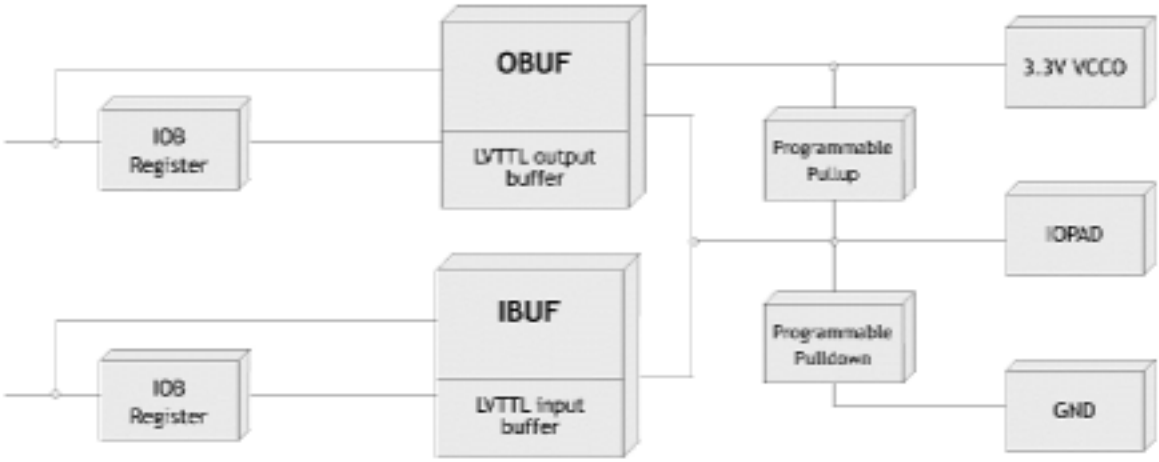

Figure 18.1

Block RAM

The Block RAM feature helps implement on chip true dual port memories without the expense of any flip flop for storage. Eight dedicated Block RAMs are available of total size 32K for implementation. The Block RAMs have dedicated routing for interfacing with other Block RAMs and the CLBs as well. They can be instantiated or inferred in VHDL behaviourally and the synthesis tool will take care of the implementation.

Distributed RAM

Each CLB in the Spartan II array features LUTs which are implemented using SRAMs. This memory can be used to implement RAMs which use the CLB resources and still does not use any flip-flops for storage. Unlike Block RAMs which have dedicated resources, these RAMs have to compete with regular interconnect paths for routing. A maximum of 24576 bits of distributed RAM can be used.

Clock Distribution

FPGA clocks are distributed throughout the FPGA using dedicated high speed, low skew primary clock distribution networks. Four system clocks are provided which are driven by four global clock buffers near the four clock pins. These four buffers can drive the primary clock routing/distribution network inside the FPGA. The secondary routing network can be used to distribute any internal signals and are not limited to clocks. The IBUFG primitive receives the external clock signal from a clock pad. Its output is then routed to the CLKDLL. The output of the CLKDLL then drives a BUFG which connects to the primary clock routing network. All these symbols are available in the macro BUFGDLL which can be instantiated as one module for ease of implementation.

Delay Locked Loops (DLLs)

A DLL is a low clock skew, zero propagation delay clock circuit which is used to manage and distribute system clock inside the FPGA. The DLL acts as a clock multiplier, divider, phase shifter or a mirror to maintain, improve and manage clock integrity on-chip and off-chip. A DLL delays the clock waveform by one clock cycle and ensures that the clock reaches every internal clock net in phase without skew.

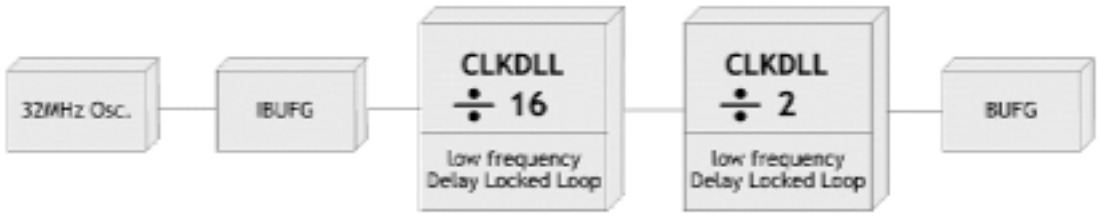

Figure 18.2

Two variants of the DLL are available for LF (25 MHz to 100 MHz) and HF (60 MHz to 200 MHz) applications. They can be instantiated using the CLKDLL (LF DLL) library symbol since 32 MHz has been chosen as the system clock frequency. Internally, the CLKDV is used as output and by setting the CLKDV_DIVIDE property to 16, allows a divide by 16 of the input frequency. Since the target system requirement is of 1 MHz, we make use of another DLL for an additional divide by 2 and route the output of this cascaded DLL to the internal clock distribution network.

Tri-state Bussing

Every Spartan II CLB has two tri-state connections mapping onto four special tri-state lines. This helps in designing Address and Data buses to RAMs from multiple drivers.

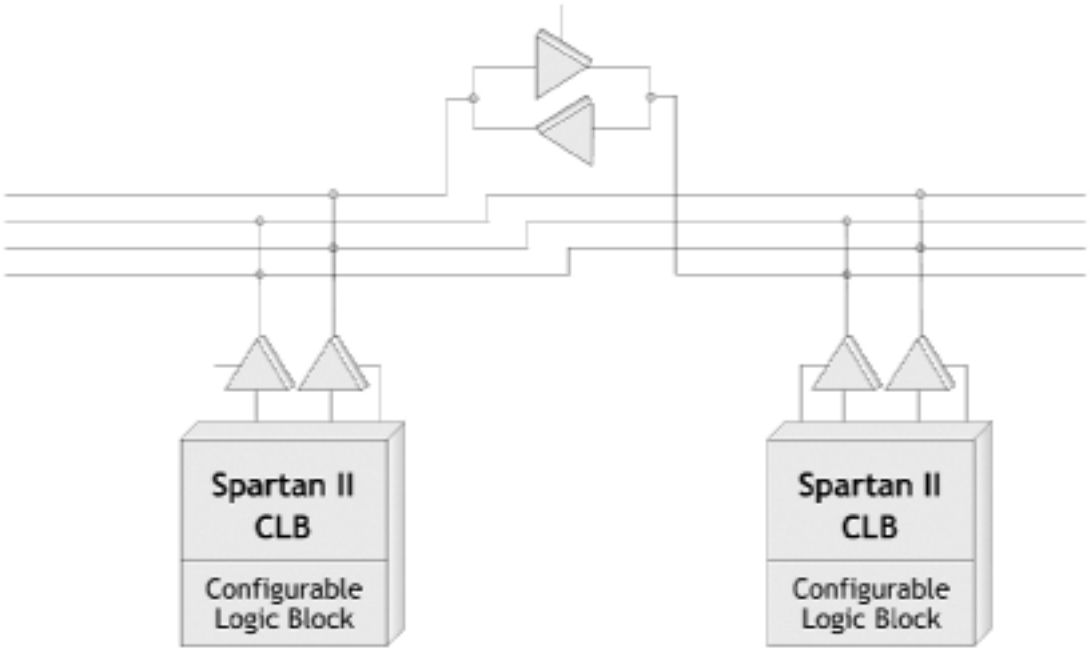

Figure 18.3

Versa Ring

The special feature of Versa ring, which is a series of special routing interconnections made available at the IOBs, enable pin-locking and pin-swapping. This enables the digital design to adapt to the existing PCB directly.

Chapter 19

FPGA CONFIGURATION[17]

The FPGA is an infinitely reprogrammable device. The on-chip logic blocks can be programmed using bit patterns that instruct each logic block to behave in a certain way. Configuration is the process by which this programming bit pattern is loaded onto the FPGA to specify its functioning.

The programming bit pattern is stored in a configuration file, generated automatically by the development software. This is then loaded frame by frame onto the FPGA. The selected device, the Spartan II XC2S50 can be programmed in four different ways: Slave Serial, Master Serial, Slave Parallel and Boundary Scan Mode. In the present design, only the Master Serial Mode and Boundary Scan Mode have been employed, so the others will not be discussed.

Master Serial Mode Description

In this mode, the FPGA is configured from another device which is on the same board. This is usually a non-volatile memory device (an EPROM of some sort), which stores the configuration bit file. This EPROM is then sequentially read by a configuration controller, and the FPGA is programmed. The FPGA senses a power-on condition, and starts clearing its configuration memory. At this time it asserts the active low Init signal to the configuration controller, which then makes Init go high when it is ready to transmit. The FPGA now sends a configuration clock (CCLK), and the configuration data is transferred from the EPROM to the FPGA synchronously with CCLK. The end of configuration is signalled by a pin called DONE going high.

However, Spartan II devices can also be programmed from dedicated PROM devices, which have the configuration controller as well as the non-volatile memory on-chip. They save the user from having to waste a general-purpose microcontroller on such a trivial operation. The PROM comes in a small 20-pin PLCC package, and can be directly interfaced with the FPGA through just four dedicated pins. The entire configuration process takes a few hundred milliseconds from the time when power is applied.

[17] Configuration details were obtained from the Configuration PROM datasheet

One such PROM is the XC17S00 series. But the drawback of this series is that these PROMs are one time programmable (OTP), making them unsuitable for prototype environments. As an alternative the XC18V00 series of In-System-Programmable (ISP PROMs is available. Specifically, this design uses the XC18V01 (1 M-bit) device (whose memory capacity exceeds the 559,200 bits required for configuring the XC2S50). This also has a similar interface to the XC2S50. This PROM can be programmed using the Boundary Scan Mode, which will be described next.

Boundary Scan Mode

Configuration using the Boundary Scan Mode takes place through the dedicated IEEE 1149.1 Joint Test Access Group (JTAG) Test Access Port (TAP). This port is available on the XC2S50 FPGA as well as the XC18V01 PROM. This port consists of the following signals:

- TCK: The clock signal which synchronizes all the JTAG operations, transmitted by the JTAG master device, usually the Personal Computer from which the configuration file is being loaded.
- TDI: The line on which configuration bits are transmitted to the target device.
- TDO: The line on which the device data may be read back.
- TMS: The line on which various JTAG instructions are issued.

During the entire configuration process, the device being programmed is completely isolated from the remaining board, by putting it into high impedance mode. Several devices that need configuration may be connected in a boundary scan chain, so that configuration data "passes through" each device. In situations where only some of these devices need to be configured, they may be bypassed by making appropriate settings in software. Several other options are possible, such as verifying the sent data, identifying devices on the chain, by reading their ID codes etc.

The Download Cable : Parallel Cable III

The configuration download from the PC to the target board via JTAG takes place using a download cable. There are four different cables supported by the development software. The cable that is used in this design is the called the Parallel Cable III. This cable connects to the PC parallel

port, and to the JTAG port on the board. Between these two ends, a circuit consisting of tri-state buffers and diodes is incorporated. This is meant to protect the PC from any dangerous signals coming from the board. The download cable used in this design is not a single cable as such. It consists of a conventional 25-pin DSUB connector cable, which connects the parallel port to the intermediate circuit. The output of this circuit is the collection of four JTAG signals, which connect to the target board through a flat ribbon cable. The protection circuit derives its power from the target board supply.

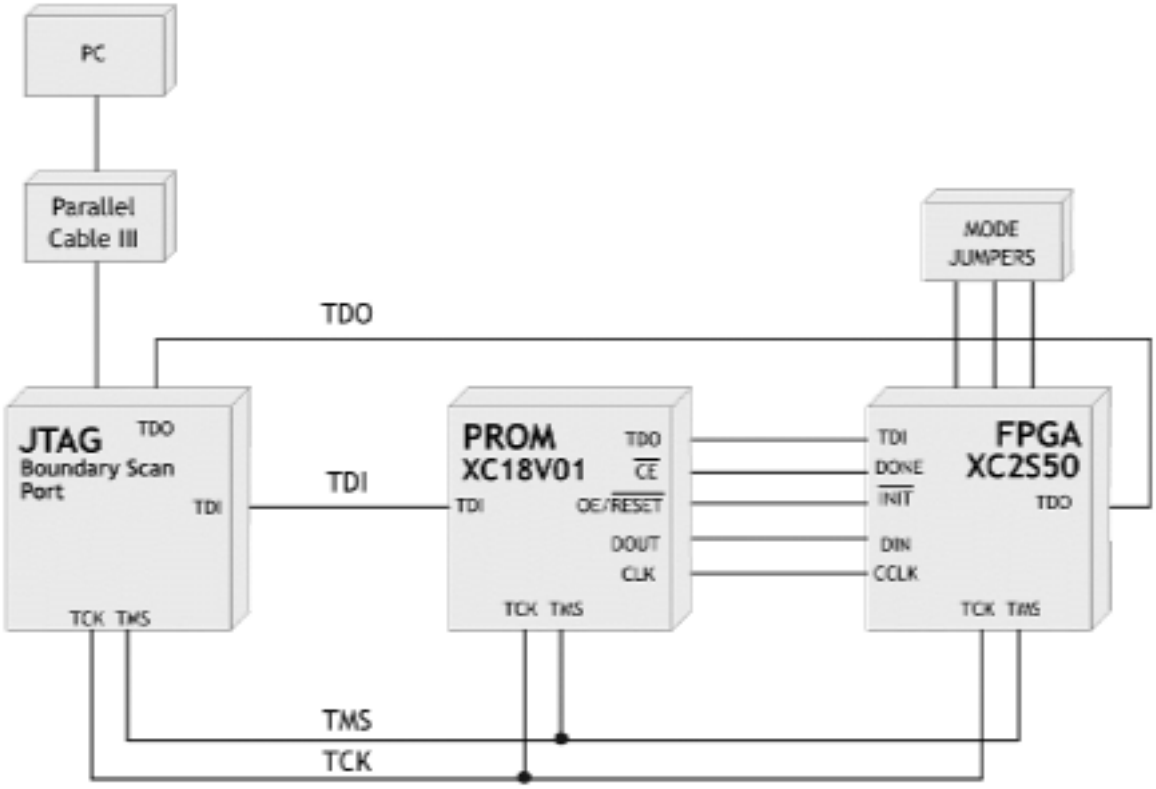

Figure 19.1

Configuration Sequence

The FPGA has three Mode Select pins which decide the mode of configuration to be followed. There are two environments for configuration which will now be described:

- Prototype Environment

Here, the FPGA mode pins are set to the Boundary Scan Mode. The ISP PROM and FPGA are connected in a boundary scan chain, (this is done by connecting their TCK and TMS inputs together. The JTAG TDI pin goes to the PROM TDI pin, while the FPGA TDI pin comes from the PROM TDO pin. The FPGA TDO pin connects to the JTAG TDO pin) and are programmed in system.

The development software generates the configuration bit file, and converts it into a PROM readable format. This PROM file may be used to program the PROM via JTAG, or the bit file may be directly used to configure the FPGA via JTAG. This can be done by bypassing one of the two during the boundary scan chain operations. This is useful when the digital design is under development, and repeated program cycles due to design changes may be required.

- Post-development Environment

Once the design is final, the configuration data may be loaded onto the PROM for the last time, and the FPGA mode set to Master Serial. Now the PC and the JTAG port are no longer needed, and on each power-up, the FPGA will configure itself from the PROM on board. The system may now be called truly stand-alone.

Chapter 20

THE FPGA DESIGN FLOW

The FPGA based digital designs involve the efficient partitioning of design for faster design cycle and less rework due to logical errors. The need for a good design entry tool is essential due to the team design philosophy implemented during the design process. The design entry tool should allow quick integration of the designs and at the same time provide a powerful yet easy medium to code the designs in a Hardware Description Language[18]. A good verification tool helps speed up the design by minimising iterations lost due to inefficient simulations. The verification tool must be able to handle large designs effortlessly and provide several inspection features for examining the simulated waveforms. Synthesis begin the ultimate objective, inference of hardware from the HDLs must be with the utmost of precision and accuracy. The synthesis tool must provide a good degree of control over the inferred hardware and allow verification of post synthesis functionality. It must provide an effortless bridge to the Place and Route tool, proprietary of the FPGA vendor which in this case is Xilinx.

Design Specification

The digital design is first understood, objectives set, and logically partitioned. A team member then takes up one of the partitioned blocks and is responsible for the complete internal description of the block. The blocks are then to be combined to obtain the required behaviour. Design partitioning was done on a functional basis with each block performing a certain function as defined in the design sequence.

Design Entry

The design entry tool chosen was Mentor Graphics HDL Designer Series ™ V2001.5a fully functional evaluation version. It features a rich set of design entry options of which the design

[18] The HDL used for design entry was Very High Speed Integrated Circuit Hardware Description Language (VHDL)

exploits the facilities like State Diagram, Truth Table, Interface Based Design and VHDL Entity/Architecture based design entries fully.

- The use of the State Diagram Editor allowed a powerful and simple description of controllers of the major design blocks. The machines were clocked synchronously, applied asynchronous reset and encoded in one-hot manner. The generated VHDL file contained two processes for next state assignment logic and the next state decoder logic. Default values were given to the state machine outputs to allow control over the inferred hardware (latched v/s combinatorial outputs). The most powerful part of the state machine view was Debug Detective ™ which allows the verification tool to interact directly with the state diagram and animate the state transitions for greater understanding due to the innovative visualisation provided.

- Truth Table based entry provided a convenient, easy to understand format for entering Boolean equations which needed to be described in a truth table form (similar to k-maps in ordinary Boolean algebra). The truth table is converted automatically into IF-THEN-ELSE statements in the generated VHDL thus saving time of manual coding and yet allowing an easy to view and interpret interface for the same.

- The Interface Based Design (IBD) was another of the HDL Designer's interesting and powerful features that allowed modular designs to be implemented in a quick, fault free manner. The embedded components were to be simply included in the IBD and the signals mapped automatically by name using the "Signal Stubs" feature. This enforced a discipline of using consistent signal names in all levels of hierarchy which enabled quick mapping using "Signal Stubs" and also quick tracing of signals due to same naming convention used.

- The VHDL Entity/Architecture entry allowed specification the entity interface directly and simply entering the architecture of the design unit directly in VHDL. Counters, Buffers, Flip-flops, Registers and VHDL simulation models of the UUT, Printer, LCD Module and EPROM were defined directly in VHDL.

- Another interesting feature was the ability to assign Synthesis constraints relevant to the synthesis tool directly in the interface of the top level entity. Features of synthesis such as single pin-locking, array pin-locking and clock distribution buffers were specified at the

design entry stage itself, greatly simplifying the arduous and time consuming task of specifying constraints during synthesis.

- HDL Designer also allowed direct interface with the simulation and synthesis tools of choice with no limitation of the vendor of those packages, provided the vendor EDA tools were compliant with the generated VHDL of this tool. This did not place the constraint of selecting packages and provided great flexibility in their choice.

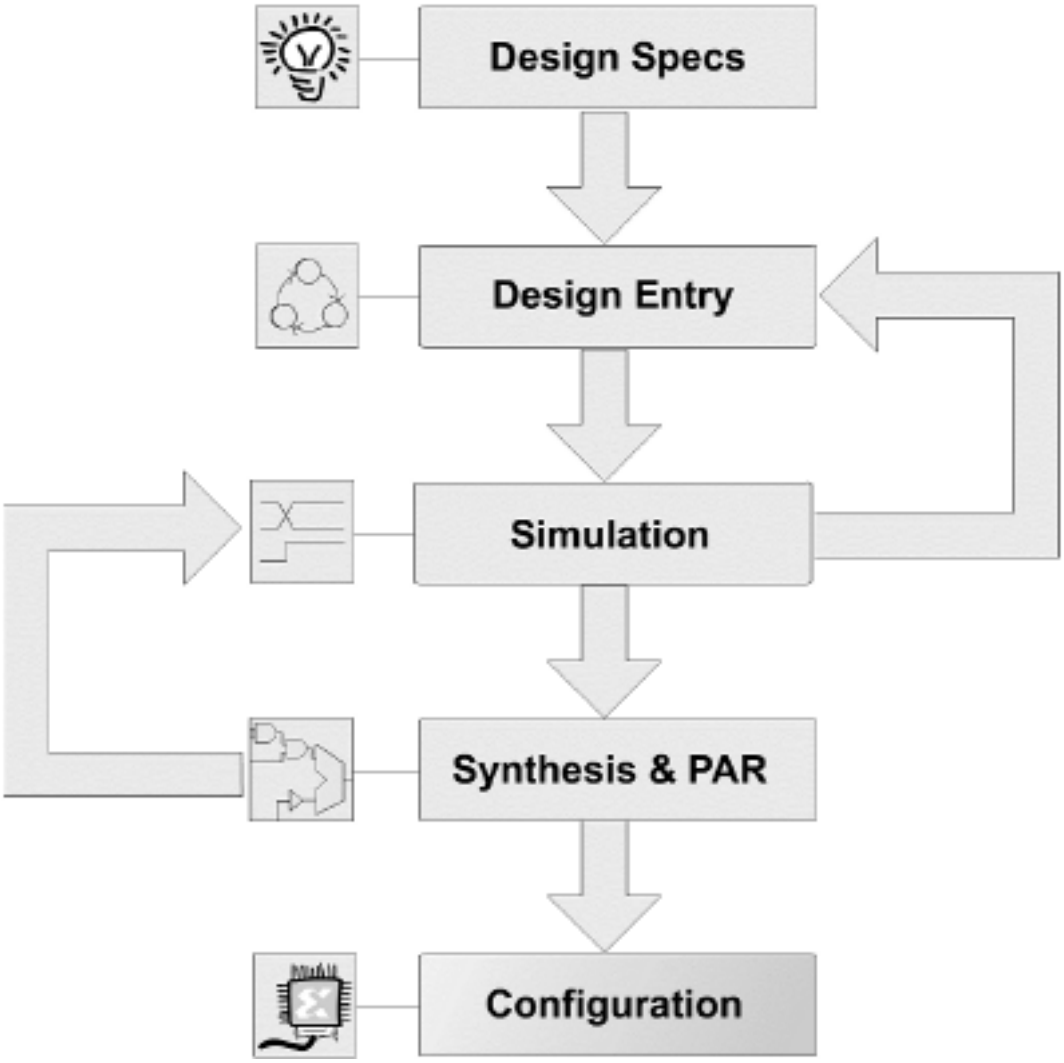

Figure 20.1

Design Verification

The verification tool of choice was the industry standard Model Technology's ModelSim ™ V5.5SE fully functional evaluation version and ModelSim XE, which had an easy to use interface. With support for HDL Designer inbuilt, the utility of ModelSim was evident right from the beginning since designs could be verified instantly at the click of a mouse. Simulation at three stages in the design was possible due to the robust and powerful library management used by the simulator. Post Entry, Post Synthesis and Post Place and Route. Verification helps keep a tab on the system functioning and ensures that at every stage the design goal is met.

- The Post Entry simulation is conducted at several levels of hierarchy in the design. All that was required to do was to invoke the simulator directly from the HDL Designer windows and assign signal values and set other preferences and simply observe the simulated waveforms.

- The Post Synthesis and Post PAR simulations are necessary to check the mapped hardware correctness only. It was necessary to export the netlist in VHDL format and import it into ModelSim, set library mappings to the simprim library (which is a part of the Xilinx Simulation Primitives) and simulate the design using simulation models from the primitives instantiated in the VHDL netlist.

- Special features such as Breakpoints, Radix change, Transition tracing, flexible Simulation time intervals and enforcing an instance based nomenclature for signals helps in quick and effortless simulation. Simulations at the topmost level are simplified by the use of the nomenclature scheme. Use of breakpoints and transition tracing help in quickly traversing across long waveform boundaries.

Design Synthesis

This stage is probably the most critical stage of the entire design. Choice of proper device, specification of proper synthesis constraints, and use of scripting all play an important role in this process. Synthesis was done using the most powerful synthesis tool available in the market Exemplar Leonardo Spectrum™ V2001.d fully functional evaluation version which too had a direct interface with HDL Designer.

- The XC2S50 Spartan II device with PQFP 208 pin and -5 Speed Grade package library was chosen (XC2S50-5PQFP208).

- Pin locking and clock buffer constraints were read from the "custom code" generated from HDL Designer.

- Additional synthesis constraints like "32 MHz" input clock frequency, Optimisation for Area, Preservation of hierarchy, Electronic Data Interchange Format (EDIF) output and integrated Place and Route runs need to be chosen while invoking the synthesis tool from HDL Designer itself.

- Any other constraints, if required, may be either entered in the script files (Tcl/Tk scripts) or in the Advanced Setup Flow Tab in Leonardo Spectrum.

- The entire design flow takes around 75 minutes and use of scripting eliminates manual intervention at any point of time. At the end of the synthesis run, a detailed report indicating cell usage at all levels of hierarchy, total device utilisation, I/O utilisation statistics, and other reports are made available. It gives an indication of the device suitability for the system design. The choice of the XC2S50 device was influenced by the fact that the whole design took up 82% of the device (with 60% LUT utilisation) and all synthesis constraints were met.

- Errors occurring during synthesis served as insightful pointers into the design flaws that were made which were subsequently rectified. This helped our design remain synthesis proof at all stages which meant that the VHDL code was always synthesisable.

Device Programming and Configuration

Xilinx WebPACK™ V4.2 contains all the downstream tools required for migrating the digital design onto the FPGA. These tools are proprietary of Xilinx and can be invoked from the synthesis tool directly (except the programming tools).

- The Integrated Place and Route tool is a part of Xilinx WebPACK V4.2 which generates the bit file required for device programming. It also exports the VHDL netlist in the end once the synthesised hardware has been mapped onto the target device.

- The PAR tool maps the actual hardware and routes the interconnects between the blocks onto the target device as per the delay and area constraints specified.

- Once the mapping and routing is done, a VHDL netlist is generated for simulation in ModelSim. A "BIT" file too is generated which can directly be used for programming the device. But since the design was to make use of programming the device through an external PROM, the Prom File Formatter was required to convert this BIT file into a PROM file.

- Finally the device programming tool iMPACT ™ V4.2 was used to specify the PROM – FPGA boundary scan chain and indicate that programming is to be carried out for the PROM only and not the FPGA. Once programmed, the system could operate as a standalone system and no link to the PC is then required to program the device on every power-up.

C h a p t e r 2 1

PRINTED CIRCUIT BOARD DESIGN

Printed Circuit Board design is the crucial aspect of the implementation of the testing system. Since the system involves many logic elements, those too at different logic levels, extra care has been taken in design of the board. Also the importance is given to the flexibility for minor changes, if required, to be done on the board. The first step in designing the PCB is the schematic drawing. Cadence CAD software, OrCAD ™ fully functional evaluation version was used for the purpose of design of the artwork for the PCB.

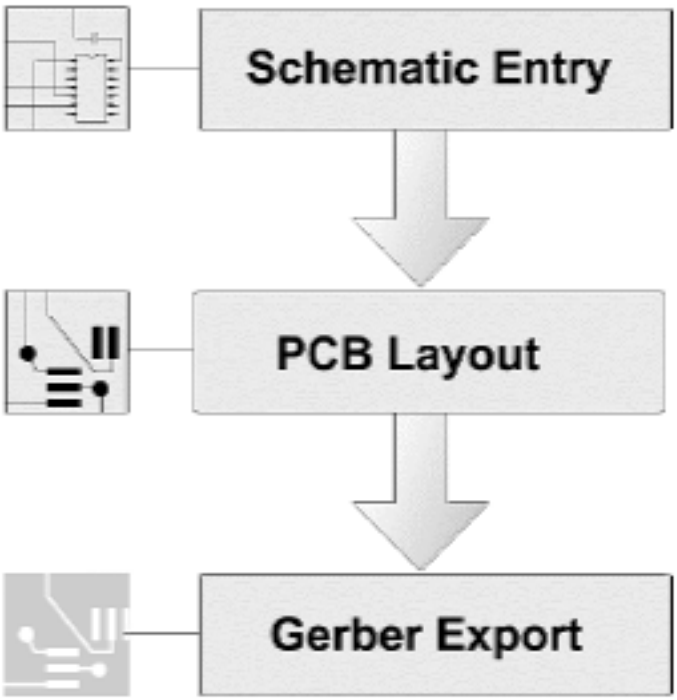

Figure 21.1

Schematic Entry

OrCAD Capture CIS is a powerful design environment, which is easy to learn and use. It is a schematic design entry tool. The tool has an extensive device library. The symbols and pin-outs for the ICs and components can be directly included from the library.

Unfortunately, the XC2S50 was not available in the library. The import tool in the Capture was used to solve this problem. After synthesis, the PAR tool generates the Xilinx M1 .pad file, which can be directly imported in Capture. A new component XC2S50 was thus created, that has the pin-out with the pin names that were used at the time of pin-locking the FPGA, besides the default non-user-I/O pins. This makes further connections in the schematic easier

The configuration PROM component was created by editing one of the existing 20 pin ICs in the Capture library. The relay library EPROM 27C256 was readily available. Other components such as buffers, pull-up resistors, voltage regulators, decoupling capacitors and printer port DB-25 connector were included in the design directly from the library. Eight and twelve pin headers were used to connect the FPGA ports and the display ports to the corresponding devices in the system.

Although there are three logic supplies, 5 volts, 3.3 volts, and 2.5 volts, their ground lines are common. Several decoupling capacitors were used to decouple the noise on the various Vcc lines and to avoid variations in Vcc and ground bounce during fast switching of the logic gates inside the digital system.

The Design Rule Check (DRC) feature helps in debugging incorrect and floating connections. The DR Check provides the exact location and component name for which an error has been found. This helped in drawing correct schematics, which were further used to prepare the artwork layout for the PCB.

PCB Layout

After the entire schematic was completed, the Capture tool was used to create the netlist for the schematic. The netlist contains all the information regarding connections in the schematic design. The netlist generated is then exported to the Layout Plus™ software, which is used to prepare the artwork for the PCB.

After loading the netlist in Layout Plus, the tool asks the user to specify the footprint for every component used in the design. The footprint relates the component to its package that will be used in the actual implementation of the system.

- The FPGA has PQFP208 as its footprint, which was carefully edited and generated to perfection (with a padstack of 12 x 80 and 80 x 12mils, pitch of 0.5mm and package dimensions of 27.9 x 27.9mm).

- The configuration PROM is available in PLCC20 pin package. The footprint for the PLCC20 pin socket, in which the IC will be mounted, was also created using the Footprint Builder tool.
- The analog power switching relays fit in the DIP14 socket and the footprint was accordingly specified. The headers have SIP8 or SIP10 as their footprint.
- The LM317s have TO220 package. The printer port is standard DB-25 pin connector.
- For resistors and capacitors, the footprint is in the form of two drills with specified spacing in-between. The decoupling capacitors for the FPGA are SMD components, with a standard footprint,"0805".

After the footprints for all the components were specified, all these components were visible on the screen. It was a time-consuming job to place all the components such that interconnections between them will be simplified. After many trials and errors, the placement of the components was finalised.

The Layout Plus software gives the user many options for designing a single layer or a multi-layer PCB. The technology templates available in the software were used to specify the defaults for routing parameters, such as one track between two pads, or two tracks between two through holes etc.

Another file loaded was the strategy file. This file decides the strategy used for routing the nets. The option two layers with via holes was chosen.

After placement was done, the Smart Route tool (which is a gridless, shape-based auto-router) was invoked to perform the task of routing.

The following parameters were specified before the auto routing feature was used:

Parameter	Option Used
Layers	TOP, BOTTOM (2 layer PCB)
Track to Track Spacing	7 mils
Via to Pad Spacing	45 mils
Track to Via Spacing	30 mils
Track width	8 mils for Signal Nets, 14 mils for Supply Nets (inter-track spacing of 7 mils)

Table 21.1

After all these constraints were fed to the tool, the AutoRoute Board utility was invoked. It was sometimes obvious that some connections cannot be routed for want of space or bad placement of the components. The board was then completely unrouted and the placement and the spacing were fine-tuned. After many tries all the nets were routed and the Design Rule Check was run to detect any unconnected nets, or placement violations.

- Gerber Export

The design was then post-processed to generate the Gerber files. These are required for manufacturing the PCB. In the Gerber files, the following files were included:

File	Utility
.TOP	Route Map in Top Layer
.BOT	Route Map in Bottom Layer
.SST	Silk Screen Top (component outlines and labels)
.SSB	Silk Screen Bottom
.SMT	Solder Mask in Top Layer
.SMB	Solder Mask in Bottom Layer
DRILL.TXT	Drill Co-ordinate file for CNC machine drilling

Table 21.2

CONCLUDING REMARKS

Before the commencement of the final year, the project team spent one month at Siemens Nasik Works in the form of in-plant training. The project objectives were studied and the test strategy was developed during the period of training, keeping a PC based system in mind. The strategy was successfully tested on a PC with an I/O expansion card.

As it happens with real life projects in the industry, the specifications for the project were changed midway. The initial specification of a PC based testing system was changed and a complete stand-alone system was to be developed. Nevertheless, the test strategy which was already verified did not change. The implementation was to be done in an entirely different fashion.

Keeping the objectives of flexibility, functionality and robustness of the system in mind, the digital design was completed. The design was tested for its functionality by using an industry standard simulator. The synthesis and PAR procedures were subsequently executed successfully.

The PCB design process went on in parallel with the digital design thanks to the pin-locking feature of the FPGA. Unfortunately, due to a critical error in the footprint of the FPGA itself, the fabricated PCB had to be scrapped. The PCB was redesigned and fabricated subsequently. This caused a delay in the completion of the project work.

At the time of submission of the report, all aspects of the digital design have been verified for their correctness. The system components have been procured. The actual implementation of the system on the shop floor is pending.

INDEX

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X