6-17-1996

Boundary scan system design

Craig Loomis

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
BOUNDARY SCAN SYSTEM DESIGN

by

Craig R. Loomis

A Thesis Submitted
in
Partial Fulfillment
of the
Requirements for the Degree of
MASTER OF SCIENCE in Computer Engineering

Approved By:

______________________________
George A. Brown, Professor and Thesis Advisor

______________________________
Dr. Pratapa V. Reddy, Associate Professor

______________________________
Dr. Robert E. Pearson, Associate Professor

______________________________
Dr. Roy Czernikowski, Department Head

DEPARTMENT OF COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK
JUNE 17, 1996
Title of Thesis: Boundary Scan System Design

I, Craig R. Loomis, hereby grant permission to the Wallace Memorial Library of Rochester Institute of Technology to reproduce this thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Signature ___________________________ Date ____________________

2 May 1996
ABSTRACT

Given the strong competition in digital design on the national and international levels, boundary scan devices are rapidly becoming a necessary as opposed to a convenient feature on integrated circuits. This thesis serves a dual purpose. First, it demonstrates how boundary scan devices can be used to increase the testability of a circuit and it presents several factors used to quantify the cost associated with the addition of boundary scan compatibility to digital designs. Cost tradeoffs are often the most intimidating hurdle for engineers to cross when deciding if boundary scan compatibility is worth the effort. Second, it demonstrates the use of the Tektronix LV500 (logic verifier) as a general testing tool, using boundary scan designs as examples. These examples provide an understanding of the function of boundary scan cells and the JTAG/1149.1 standard. The LV500, which is used by students in the Department of Computer Engineering and Microelectronic Engineering at RIT, is an indispensable tool for making critical timing measurements. It also allows a user to evaluate and step through simple as well as more complicated designs. It is my hope that this thesis and the tutorial provided will facilitate the use of the LV500 in future testing work performed in RIT's center for Microelectronic and Computer Engineering clean room facilities. Upon following the example circuits described, one should become familiar with boundary scan terminology as well as the methodology used in designing such a system.
# Table of Contents

List of Figures ........................................................................................................... vii  
List of Tables ........................................................................................................... ix  
Glossary .................................................................................................................... x  
1.0 Introduction and History .................................................................................... 1  
  1.1 Testability History .............................................................................................. 1  
    1.1.1 Important Characteristics of Testability ....................................................... 2  
    1.1.2 Current Testability Issues ............................................................................ 4  
  1.2 How Testability Has Been Implemented ............................................................ 8  
    1.2.1 Fault Types .................................................................................................. 9  
    1.2.2 Two Requirements of Fault Detection ....................................................... 10  
    1.2.3 A Classical Example .................................................................................. 11  
  1.3 Limits of Classic Example .................................................................................. 12  
    1.3.1 Increasing Density of ICs .......................................................................... 12  
    1.3.2 Increasing Density of PCBs ...................................................................... 13  
    1.3.3 Physical Interface with Bed-of-Nails ......................................................... 14  
  1.4 Boundary Scan Alternative ............................................................................... 15  
    1.4.1 The Basics .................................................................................................. 16  
    1.4.2 Overcoming Density of ICs ....................................................................... 19  
    1.4.3 Overcoming Density of PCBs and Physical Interface ............................... 19  
  1.5 Problem Statement ............................................................................................. 20  
  1.6 System Constraints ............................................................................................ 21  
  1.7 Terminology ....................................................................................................... 22  
    1.7.1 JTAG Specifications ................................................................................ 22  
    1.7.2 Tektronix LV500 ..................................................................................... 27  
2.0 Functional Specifications ...................................................................................... 28  
  2.1 Overview of Tutorial ........................................................................................... 28  
  2.2 LV500 Setup and Configuration ....................................................................... 29  
    2.2.1 Menus ....................................................................................................... 29  
    2.2.2 Testhead Installation ................................................................................ 33  
  2.3 40 pin ZIF Testhead ........................................................................................... 34  
3.0 Tutorial for Test Boards ....................................................................................... 36  
  3.1 Single Test Chip ................................................................................................ 36  
    3.1.1 BSCAN_J Test Setup ............................................................................... 36  
    3.1.1.1 DUT Wiring ......................................................................................... 39  
    3.1.1.2 LV500 Setup ....................................................................................... 42  
    3.1.1.3 Pattern Test Vectors ............................................................................ 50  
    3.1.2 Single Stepping Test .................................................................................. 54  
    3.1.3 Schmoo Specification ................................................................................ 65  
    3.1.3.1 Input Voltage Limits .......................................................................... 66  
    3.1.3.2 Propagation Delay ............................................................................... 69  
  3.2 Single Chip with Combinational Logic ............................................................... 73  
    3.2.1 Coffee Break ............................................................................................. 73
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2 Design of State Machine</td>
<td>73</td>
</tr>
<tr>
<td>3.2.3 BSCAN_II Test Setup</td>
<td>84</td>
</tr>
<tr>
<td>3.2.3.1 DUT Wiring</td>
<td>85</td>
</tr>
<tr>
<td>3.2.3.2 LV500 Setup</td>
<td>88</td>
</tr>
<tr>
<td>3.2.3.3 Pattern Test Vectors</td>
<td>89</td>
</tr>
<tr>
<td>3.2.4 Running Test</td>
<td>90</td>
</tr>
<tr>
<td>3.2.5 Schmoo Specification Tests</td>
<td>99</td>
</tr>
<tr>
<td>3.2.5.1 TCK Pulse Parameters</td>
<td>99</td>
</tr>
<tr>
<td>3.3 Boundary Scan Circuit Design</td>
<td>105</td>
</tr>
<tr>
<td>3.3.1 Logic Design at the IC level</td>
<td>105</td>
</tr>
<tr>
<td>3.3.2 Boundary Scan Architecture Features</td>
<td>106</td>
</tr>
<tr>
<td>3.3.3 The Design Overview</td>
<td>107</td>
</tr>
<tr>
<td>3.3.4 The Design Details</td>
<td>119</td>
</tr>
<tr>
<td>3.3.4.1 Design of the TAP</td>
<td>119</td>
</tr>
<tr>
<td>3.3.4.2 Design of the Instruction Register</td>
<td>125</td>
</tr>
<tr>
<td>3.3.4.3 Design of the Bypass Register</td>
<td>129</td>
</tr>
<tr>
<td>3.3.4.4 Design of the Boundary Scan Register</td>
<td>130</td>
</tr>
<tr>
<td>3.3.4.5 Design of the Custom Scan Register</td>
<td>133</td>
</tr>
<tr>
<td>3.3.5 Summary of Boundary Scan Circuit Design</td>
<td>135</td>
</tr>
<tr>
<td>4.0 Summary And Conclusion</td>
<td>138</td>
</tr>
<tr>
<td>4.1 Design Details</td>
<td>138</td>
</tr>
<tr>
<td>4.2 Problems Encountered and Remaining Limitations</td>
<td>141</td>
</tr>
<tr>
<td>4.3 Suggestions for Future Enhancements</td>
<td>141</td>
</tr>
<tr>
<td>5.0 References</td>
<td>142</td>
</tr>
<tr>
<td>APPENDIX A Listing of Timing Diagrams</td>
<td>145</td>
</tr>
<tr>
<td>APPENDIX B Code Generation</td>
<td>154</td>
</tr>
<tr>
<td>B.1 Unix Applications</td>
<td>154</td>
</tr>
<tr>
<td>B.1.1 Peg Input Commands</td>
<td>154</td>
</tr>
<tr>
<td>B.1.2 Peg Eqn Output (Eqntott Input File)</td>
<td>155</td>
</tr>
<tr>
<td>B.1.3 Peg Summary Output File</td>
<td>156</td>
</tr>
<tr>
<td>B.1.4 Peg Truth-Table Output File</td>
<td>156</td>
</tr>
<tr>
<td>B.1.5 Eqntott Output File (Espresso Input File)</td>
<td>157</td>
</tr>
<tr>
<td>B.1.6 Espresso Output File</td>
<td>158</td>
</tr>
<tr>
<td>B.2 Mentor Graphics Simulation Files</td>
<td>159</td>
</tr>
<tr>
<td>B.2.1 QuickSim Input Files</td>
<td>159</td>
</tr>
<tr>
<td>B.2.2 QuickFault Input Files</td>
<td>165</td>
</tr>
<tr>
<td>B.2.3 QuickFault Output File</td>
<td>171</td>
</tr>
<tr>
<td>B.3 LV500 Input File</td>
<td>174</td>
</tr>
<tr>
<td>B.3.1 BSCAN I.NEW Pattern File</td>
<td>175</td>
</tr>
<tr>
<td>B.3.2 BSCAN II.NEW Pattern File</td>
<td>178</td>
</tr>
<tr>
<td>APPENDIX C Discussion of Printing Operation</td>
<td>180</td>
</tr>
<tr>
<td>APPENDIX D Contacts</td>
<td>181</td>
</tr>
<tr>
<td>APPENDIX E SN74BCT8373 Specifications</td>
<td>182</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Average Number of Faults During Product Life Cycle</td>
</tr>
<tr>
<td>2</td>
<td>Relative Cost of Repairing a Product During Life Cycle</td>
</tr>
<tr>
<td>3</td>
<td>Total Time Required to Generate Test Patterns</td>
</tr>
<tr>
<td>4</td>
<td>Circuit Example of Logical Faults</td>
</tr>
<tr>
<td>5</td>
<td>Recent Trends in Gate-to-Pin Ratio</td>
</tr>
<tr>
<td>6</td>
<td>Boundary Scan Cell</td>
</tr>
<tr>
<td>7</td>
<td>Boundary Scan Path on PCB</td>
</tr>
<tr>
<td>8</td>
<td>Boundary Scan Architecture</td>
</tr>
<tr>
<td>9</td>
<td>TAP State Diagram</td>
</tr>
<tr>
<td>10</td>
<td>LV500 Keyboard</td>
</tr>
<tr>
<td>11</td>
<td>Power Up Menu</td>
</tr>
<tr>
<td>12</td>
<td>Main Selection Menu</td>
</tr>
<tr>
<td>13</td>
<td>LV500 System Configuration</td>
</tr>
<tr>
<td>14</td>
<td>40 pin ZIF Wiring Diagram</td>
</tr>
<tr>
<td>15</td>
<td>Save/Restore Menu</td>
</tr>
<tr>
<td>16</td>
<td>Testhead Wiring Diagram</td>
</tr>
<tr>
<td>17</td>
<td>DUT Wiring Menu</td>
</tr>
<tr>
<td>18</td>
<td>Channel D Menu</td>
</tr>
<tr>
<td>19</td>
<td>Channel Q Menu</td>
</tr>
<tr>
<td>20</td>
<td>Detail of Clock Phase Signals</td>
</tr>
<tr>
<td>21</td>
<td>Template Menu bscan_stndrd</td>
</tr>
<tr>
<td>22</td>
<td>Template Menu play_templates</td>
</tr>
<tr>
<td>23</td>
<td>Pattern Menu</td>
</tr>
<tr>
<td>24</td>
<td>POWER_UP Macro Menu</td>
</tr>
<tr>
<td>25</td>
<td>RESET Macro Menu</td>
</tr>
<tr>
<td>26</td>
<td>Config Menu</td>
</tr>
<tr>
<td>27</td>
<td>BYPASS Scan Register</td>
</tr>
<tr>
<td>28</td>
<td>Instruction Register</td>
</tr>
<tr>
<td>29</td>
<td>Placement of Single Step Commands</td>
</tr>
<tr>
<td>30</td>
<td>SAMPLE Scan Register</td>
</tr>
<tr>
<td>31</td>
<td>Setting Bit D8 High to Flag Error</td>
</tr>
<tr>
<td>32</td>
<td>Replacing mask_TDI with mask_LATCH template</td>
</tr>
<tr>
<td>33</td>
<td>Define Schmoo Menu</td>
</tr>
<tr>
<td>34</td>
<td>Input Voltage Limits</td>
</tr>
<tr>
<td>35</td>
<td>Determining propagation time from D to Q</td>
</tr>
<tr>
<td>36</td>
<td>Propagation Delay Time</td>
</tr>
<tr>
<td>37</td>
<td>Coffee Break</td>
</tr>
<tr>
<td>38</td>
<td>Gate Level Connections</td>
</tr>
<tr>
<td>39</td>
<td>Percentage of Total Faults</td>
</tr>
<tr>
<td>40</td>
<td>Faults per Cycle</td>
</tr>
<tr>
<td>41</td>
<td>Control and Test Latches</td>
</tr>
<tr>
<td>42</td>
<td>Detail Switch Schematic</td>
</tr>
</tbody>
</table>
Figure 43  Coffee Break--Wiring Diagram ....................................................... 83
Figure 44  Save/Restore Menu ................................................................. 84
Figure 45  Breadboard Wiring Diagram ..................................................... 87
Figure 46  Template Menu for bscan_stndrd ............................................ 88
Figure 47  Pattern Menu ................................................................. 89
Figure 48  Pattern Menu for Manual Mode Stepping ................................. 91
Figure 49  Location of Device State Bits ................................................ 94
Figure 50  '8373 Latching Output in TEST Mode ....................................... 94
Figure 51  Next State Bit Stream Shifting into the '8373 ........................... 95
Figure 52  Conclusion of BSCAN II Pattern ........................................... 98
Figure 53  Boundary Scan Cell ............................................................. 99
Figure 54  Definition of Setup and Hold Times for the Circuit .................... 101
Figure 55  Sweeping the Phase Width and Phase Delay of 0B ..................... 101
Figure 56  TCK Pulse Limits .............................................................. 103
Figure 57  Higher Resolution TCK Pulse Limits ....................................... 104
Figure 58  State Machine Logic for the Second Example ......................... 107
Figure 59  Gate Level Connections ....................................................... 109
Figure 60  Percentage of Total Faults .................................................... 111
Figure 61  Percentage of Total Faults With Feedback Multiplexer .............. 111
Figure 62  Gate Level Connections for Mode Select Multiplexers ............... 112
Figure 63  Fault Coverage of Q0 Observation Point .................................. 113
Figure 64  Fault Coverage of Q1 Observation Point .................................. 114
Figure 65  Fault Coverage of Q2 Observation Point .................................. 115
Figure 66  Fault Coverage of Q0 and Q1 Observation Point ....................... 116
Figure 67  High Level Schematic for Circuit ......................................... 118
Figure 68  TAP Output Signals ............................................................ 122
Figure 69  TAP Schematic ................................................................. 124
Figure 70  TAP Simulation ................................................................. 125
Figure 71  IR Register Level Schematic ................................................ 127
Figure 72  IR Selection of SAMPLE DR ................................................. 127
Figure 73  IR Control Logic ............................................................... 128
Figure 74  Bypass Register Control Logic .............................................. 129
Figure 75  BSC Register Level Schematic .............................................. 132
Figure 76  BSR Scan Path ................................................................. 132
Figure 77  Custom Register Level Schematic ......................................... 134
Figure 78  Custom Register Control Logic ............................................. 134
Figure 79  Schematic for Logic of Boundary Scan Component .................... 137
Figure 80  BSCAN_I e.g. 1 ................................................................. 145
Figure 81  BSCAN_I e.g. 2 Timing Diagram ........................................... 146
Figure 82  BSCAN_I e.g. 2 Timing Diagram (cont.) ................................ 147
Figure 83  BSCAN_II Timing Diagram .................................................. 148
Figure 84  BSCAN_II Timing Diagram (cont.) ........................................ 149
Figure 85  BSCAN_II Timing Diagram ................................................... 150
Table of Tables

Table 1  Logical Mapping of DUT ................................................................. 40
Table 2  Description of Phase Times for Figure 21 ........................................... 46
Table 3  Description of Phase Times for Figure 22 ........................................... 47
Table 4  State Assignment Table ................................................................... 74
Table 5  State Transition Table ....................................................................... 76
Table 6  DUT Wiring Table .............................................................................. 86
Table 7  TAP Output Signals ......................................................................... 121
Table 8  Mapping of IR to its Address Instruction ........................................... 126
Table 9  Number of Logic Cells Required for Implementation ........................ 139
Table 10 BSCAN_I Timing Table e.g. 1 ............................................................. 151
Table 11 BSCAN_I Timing Table e.g. 2 ............................................................. 152
Table 12 BSCAN_II Timing Table .................................................................. 153
Glossary

ASIC  Application Specific Integrated Circuit. A type of IC device that performs specific pre-programmed functions such as controlling electronic instrument panels and the sensing of fluid levels.

BIST  Build In Self Test. Logic included in a design that performs a self-contained self-test of the device.

BSR  Boundary Scan Register. A set of storage cells that use multiplexers to control the flow of data into and out of an I/O pin. The I/O can be directed from the normal path to a serial shift register of similar BSRs.

DFT  Design For Testability. A design methodology that channels early design efforts to ensure that a final design product is testable.

DIP  Dual In-line Package. A rectangular plastic housing for IC and other components that is used on PCBs.

DUT  Device Under Test. Circuit that is being tested for faults.

IC  Integrated Circuit, also called a "chip." An integrated array of active and passive elements with a single semiconductor substrate.

LFSR  Linear Feedback Shift Register. A shift register with some of its outputs exclusively ORED together to form a feedback mechanism which creates a Pseudo-Random Pattern Generator.

LSSD  Level Sensitive Scan Design. A design methodology that ensures race-free operation. It is based upon memory elements that are independent of signal delays.
MCM Multi-Chip Module. A circuit formed by placing ICs on top of each other to increase circuit density and decrease signal trace length.

PCB Printed Circuit Board. The “platform” for electronic circuit construction that contains all the connections between the components printed on the board.

PRPG Pseudo-Random Pattern Generator. A LFSR configured to generate data to drive a logic block. It is often used in conjunction with a PSA during testing.

PSA Parallel Signature Analyzer. A LFSR used to compress the outputs of a logic block into a test register such that a unique value is obtained after a test is run.

SLSI Super Large Scale IC. An IC with 50,000 to 100,000 transistors.

SMT Surface Mount Technology. A method of manufacturing PCBs in which chips are fixed directly onto the surface of the board instead of being soldered into pre drilled holes.

ULSI Ultra Large Scale IC. An IC with over 100,000 transistors.

VLSI Very Large Scale IC. An IC with 5,000 to 50,000 transistors. It is also used with reference to an IC design with any large number of transistors.
1.0 Introduction and History

Boundary scan devices can be used to internally test the integrity of an IC (Integrated Circuit)\(^1\) or MCM (Multi-Chip Module)\(^2\) that is embedded in a system's design. They can also be used to test the external connections between components\(^3\) that are on the same PCB (Printed Circuit Board) even if some of the components are not boundary scannable themselves. The key behind their utility is twofold. First, is the novel idea of using an IC built with serially connected I/O (Input and Output) pins that allow test access. Second, is the use of an IEEE standard to ensure compatibility among different vendors. The IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture document has been under development since the mid 1980's with its formal acceptance in 1990.\(^4\) This thesis provides a brief history of testability and recent trends in technology that have resulted in the new boundary scan standard. It also discusses how students in the Computer Engineering and Microelectronic Engineering Departments at RIT can benefit by designing circuits and ICs that incorporate the boundary scan standard.

1.1 Testability History

It has been said that testability has been around ever since the cave man who invented the club, knocked himself out while trying to prove it worked.\(^5\) This first section gives a definition of testability followed by recent history showing why boundary scan devices are an exciting alternative to several current testability issues.
1.1.1 Important Characteristics of Testability

A circuit's testability is becoming a critical rather than convenient feature for non-trivial digital systems designed today. Before discussing current trends in testability, it is first important to understand what testability is. One definition states, “Testability is a characteristic of a design which allows the status (operable or inoperable) of a system or any of its subsystems to be confidently determined in a timely fashion.” The definition is good as it contains two of the primary ideas of testability; namely the time it takes to perform a test and the information that the test provides. Examining these two ideas will provide insight to the core of many testability issues. Let’s start with the time it takes to perform a test. This time is dependent upon the size of the design and the tools available at the facility in which the tests are performed. In other words, it depends upon how easy it is to exercise a design to obtain a system of measured responses which ascertain whether it is behaving properly. Thus, by increasing the access to test features, testability is increased. This is dependent upon the level and complexity of the system being tested. The system level will vary from testing of VLSI (Very Large Scale Integrated) circuit wafers to testing a PCB (Printed Circuit Board). The system complexity will vary in numbers of components or gates and in the degree of data abstraction. For example, tests being performed may vary from measuring a single voltage with a probing station to measuring a matrix of digital voltages with a logic verifier.

The second aspect of testability, the information that the tests provide, is driven by both the level of the design and the diagnostic detail required for its evaluation. As the design level varies from IC design to PCB design, the amount of information available changes proportionally. One obvious reason for this is that the number of nodes providing useful
information also increases with the design level. Another reason is that the systems complexity increases from monitoring voltage levels to monitoring digital abstractions representing system states and control data. There is much responsibility given to the designer to make use of that information as best they see fit. At times, it is sufficient to acknowledge whether a test has passed or not; while in other instances a system may have to provide detailed diagnostic feedback on its actions in order to isolate possible future faults. For example, in order to detect and correct early design errors the cause of the error must first be found through testing. Hence, testing is not only performed to detect faults but also to determine the cause of a known fault or misbehavior in the system. Again, depending upon the level of design, this may target anything from the detection of a misaligned mask that causes a short between two polysilicon lines on a VLSI wafer, to the clock signal that has an intermittent error in a PCB. The important thing to remember is that the value of a circuit's testability lies not only in how easy it is to detect potential misbehavior, but also how easy it is to determine the cause of it.
1.1.2 **Current Testability Issues**

Now that the characteristics that make up testability have been discussed, let's focus on business concerns and the potential impact that a design's testability has on its life cycle. Three of the main issues in a design are the *reliability*, *cost* and *time to market*. The reliability of a design relates to how it functions both after manufacturing and when it is being used by a customer. The design should operate as specified for the longest time possible. For some designs, (e.g., medical equipment) reliability, relating to the safety and well being of its users, is the most critical issue. In other designs, the desire to minimize failures and maximize product reputation drives the increases of product reliability. Cost is another issue from a customer's standpoint. It should stay within initial budget constraints for the life of the product, not just while being manufactured. The final issue is the time to market for the product. Having a product available when a customer needs or desires it, is often referred to as the market window. The first company to ship products during this time will maintain a high market share. Again, this helps to determine the overall cost of the product. These three issues are very much related with each being intimately linked to testability.

Reliability may show the strongest link to testability since it increases proportional to increased testing. Simply stated, the more aspects of a system tested, the fewer potential faults reach the customer. One link between design reliability and increased testability relates to component failure, as shown in Figure 1. The average reliability of a product is bounded by two stages of increased component failure. The second stage, shown on the right half of the Figure 1, represents "old age." This can be the result of metal migration, normal component wear or exposure to time in a harsh environment. The stage on the left of the graph is infant mortality. This is the time during which an unexpected burnout
can occur. For this reason it is especially important to run a design through its test stages during a “burn in” period prior to customer release.

Figure 1  Average Number of Faults During Product Life Cycle. *For a typical product, the average number of faults is bounded by infant morality and old age.*

As for the relationship between overall product cost and testability, the cost typically decreases with increased product testability. In particular, the cost of fixing a faulty product is much lower when the fault is caught early in the product life cycle. Figure 2 provides a graph describing the “Rule of 10’s” where the cost of detecting and repairing a fault increases ten fold at each progressing stage of the product’s life cycle. As an example, assume that a misaligned mask has caused a trace to create a short circuit between two output signals. In discovering this circuit flaw before packaging the wafer, the cost would encompass the processing and replacement of the misaligned wafers. If the short went undiscovered until customers exercising the system noticed it, then the
cost may be 1,000 fold to correct it. This drastic cost increase includes packaging, shipping and installation of each system that the fault resided in, plus the cost of on-site modification and trouble shooting by field engineers.

Figure 2 Relative Cost of Repairing a Product During Life Cycle.

At each progressive stage, the relative cost of repairing a product increases by approximately 10x's.
The remaining issue, time, also relates to cost because of the time incurred by engineering and marketing departments during product development. However, more specific to testability is the time spent in developing the test hardware and software. Assuredly there will be some finite time, but this time can be minimized by considering and implementing testability during the initial design stages. In recent history, testability has typically been implemented as an afterthought. This is represented by the solid line of Figure 3, which corresponds to a step function at each stage of product development. Each stage of the design requires the generation of new test vectors. For example, if final product testing is not thought of during the initial design stages, simple macros are written to test during the first software prototype. This is followed by the generation of new test vectors at the first hardware prototype, with yet another set generated for field service engineers to test the final product at a customer's location. In all, four separate sets of test vectors may be generated. The desire of designing for testability (DFT) is to implement the generation of test vectors and special test structures at the earliest stages of the design. The dashed line of Figure 3 represents the optimal time required to generate test vectors. This is where most of the test vectors are designed at the beginning stages and used throughout the product's life; thus decreasing the total time devoted to the generation of test patterns. In summary, increased testability should increase the reliability while decreasing product life cycle costs by catching faults at earlier design stages. The total time devoted to testing should also decrease by incorporating testability features into the early product design stages rather than ad hoc or after thought approaches.
1.2 How Testability Has Been Implemented

The definition of testability and its impacts on market needs has been presented. How then is testability implemented? The next section examines the implementation of testability on digital systems.
1.2.1 Fault Types

When a test performed on a digital design fails, the source is always of a physical nature. Often these physical faults are mapped into logical faults for classification and to reduce their complexity. Two typical examples of how common physical faults map into the logical domain are shown in Figure 4. The structural fault of a short circuit between two lines of trace may drive a node to a given voltage level. For instance, if the short occurs between VCC and a node, VCC drives the signal high and it is labeled a logical “stuck-at-one” (SA1) fault. If, on the other hand, GND drives the node, then it is labeled a logical “stuck-at-zero” (SA0) fault. If the fault is due to an open circuit rather than a short circuit, power voltages may still drive the resultant nodes, or they may become a function of the connected signals. In either case, the result of using logical mappings to classify faults greatly reduces the complexity of the remaining fault equations. Test equations are difficult to determine for all but trivial designs because IC packaging hides most nodes of interest. This leads to the question of how the faulty internal nodes of an IC are indirectly monitored through external pins.

![Figure 4](image)

Figure 4 Circuit Example of Logical Faults. Physical stuck-at-faults can be modeled as logical faults.
1.2.2 Two Requirements of Fault Detection

Two important concepts of testability are the properties of controllability and observability. In order to isolate an internal node in a design, the node must have both of these properties at the same time. Specifically, there must exist a set of test vectors that when applied to the primary inputs (e.g., IC pins, PCB test pins) will isolate all the possible faults (controllable) for that node. The logic states of the internal node must be able to propagate to a visible output (observable). If either of these properties do not exist for a given fault, then the fault is not detectable. For example, to test node one (referring to Figure 4) for a SA0 fault, apply the vector \( A=1, B=1 \). If the output is 0, and single stuck-at-faults are assumed, then node one is SA0. Similarly, to test node one for a SA1 fault the vector \( A=0, B=1 \) can be applied. If the output is 1, then node one is SA1. Because node one is controllable from inputs A and B and observable at the output node for the SA0 and SA1 faults, it is fully testable. One the other hand, node two can not be tested for a SA1 fault. In order to do so, the vector \( A=0, B=0 \) is desired, but the output for this is the same as the output for \( A=0, B=1 \) (i.e., the stuck-at-fault case). Although the fault is controllable, it is not observable, and hence not testable. It is important to note that while high numbers for controllability and observability data sets are desirable, it is the intersection of the two sets that provides insight as to a design's testability. This intersection is dependent upon system design, and as the previous circuit (Figure 4) shows, it is not hard to design a circuit that is difficult to test.
1.2.3 A Classical Example

A classic example of the implementation of testability over the past decade uses a bed-of-nails approach. The “bed” consists of a matrix of probes that applies and receives voltage signals from a DUT (Device Under Test). By abutting the bed-of-nails against a PCB and applying a set of parallel test vectors, the board’s response is recorded and compared to the desired output. The probes of the bed contact either IC component pins or test-points embedded into the board layers as designed to increase its testability. This approach has been fueled by through hole technology components with typical pin spacing on 100 mil centers\(^\text{11}\) and has performed well in detecting faults. One significant feature is the ability to diagnose multiple faults with one pass of the over the test head.\(^\text{12}\) It is also able to detect shorts before initially powering up the board, preventing the potentially catastrophic damage due to shorts between power and ground. Another benefit is the ability to test and detect analog related faults of hybrid PCBs. All of these benefits are greatly enhanced through the use software tailored to generating test vectors, storing the DUT’s responses to test head passes, and analyzing the results for possible faults.
1.3 Limits of Classic Example

The bed-of-nails approach has provided a great service to test-engineers in the past. With the increasing complexity of ICs and increasing PCB density, bed-of-nails testing on current PCBs is fast approaching obsolescence. An alternative approach using the I/O boundary of ICs to shift in test vectors through a standard test bus (IEEE 1149.1) is suggested.

1.3.1 Increasing Density of ICs

Among the limiting factors of the bed-of-nails methodology is the ever increasing density of integrated circuits. Currently, sub one micron technologies are producing circuits with densities soaring above the classic VLSI definition of 5,000 to 50,000 transistors and requiring new definitions of Super Large Scale Integration (SLSI) 50,000-100,000 transistors and Ultra Large Scale Integration (ULSI) for greater than 100,000 transistors. Point in fact, one recent design of a microprocessor contains over 3 million transistors! The difficulty with this increased IC complexity is the sheer number of internal nodes that require testability access coupled with the growing gate to pin ratio (Figure 5). Because the growth in the number of gates has not been paralleled with a growth in IC pins, internal circuit nodes have been even more difficult to observe and control from the outside. If the issue of increasing complexity of VLSI circuits does not limit bed-of-nails access from external nodes, it will certainly make testing more costly to develop alternative methods of increasing testability.
The recent trend in gate-to-pin-count ratio will continue in the future. This is one challenge faced when testing digital circuits.

1.3.2 Increasing Density of PCBs

Along with the growth in IC complexity, PCBs have also been subject to increased complexity. The recent surge in PCB density has fueled the emergence of Surface Mount Technology (SMT) components, MCM and multi-layered boards. Unfortunately for most bed-of-nails testing stations, each technology has introduced a new set of challenges. First, most bed-of-nails testers were built with probes suited for 100 mil DIP (Dual In-line Package) centers, not 25 and 50 mil SMT centers.\textsuperscript{5,11} The use of SMT has decreased the degree to which a probe can isolate a signal. Adding to this difficulty is the use of components on both sides of the PCB. The testing of such a board requires
stacking two bed-of-nails testers back to back. Although this is being done, it is the manifestation of an engineer's nightmare. Use of MCMs that limit trace lines between components by “piggy-backing” IC chips also limit the board's testability of these components when performed through external nodes alone. Multi-layered boards have added the ability to create a network of signals as never before possible. This is at the expense of adding many layers of signals hidden from the tester's probes. The technology driven increase in PCB densities has increased the cost and time associated with the testing of such devices yet even higher.

1.3.3 Physical Interface with Bed-of-Nails

Another issue, one that relates to the smaller IC pin spacing with SMT components, is the IC pin to bed-of-nails interface. Because of the decrease in pin size, there is an increased probability of misregistration of the bed-of-nails interface. The misregistration would invalidate the test results, while creating the potential of damaging the PCB exists by driving output pins to an undesired state. There is also the chance of shorting two pins together with probes from the bed-of-nails that would again invalidate test results. In summary, the increasing density of ICs and PCBs have caused the cost of testability with a bed-of-nails testing approach to rise.
1.4 Boundary Scan Alternative

While trying to maintain the cost of associated with increased testability, the philosophy of designing for test became prominent in the late 1980's. The main focus of DFT has been to incorporate testability in the earliest of design stages, thus minimizing its overall cost. In doing so, a balance is sought between the cost of implementing testability and the cost savings that it brings. This fostered the creation of several programs such as SCOAP (Sandia Controllability/Observability Analysis Program)\textsuperscript{14}, COMET (Controllability and Observability Measure for Test)\textsuperscript{9} and PREDICT (Probabilistic Estimation of Digital Circuit Testability)\textsuperscript{15} to attempt to quantify testability in terms of the DUT's observability and controllability characteristics. The problem generally associated with these programs is that it is difficult to decide what design modifications are necessary to maintain testability while minimizing its cost. Several attempts at increasing testability have been made. Most have been ad hoc, depending upon the designer. Some examples include selective insertion of test points, partitioning larger circuits into smaller blocks and using redundant logic. Although the methods work to a degree, there are tradeoffs intrinsic to each choice and the major difficulty of incompatibility among vendors. In 1985, the JTAG (Joint Test Action Group) gathered together to discuss the limits of current testing strategies and develop a common platform to build from. Their work resulted in a standard based upon the use of boundary scan circuits.
1.4.1 The Basics

The essential boundary scan cell is shown in Figure 6. Each cell contains a clocked register controlled through an input multiplexer and observed through an output multiplexer. Because of this clocking, the boundary scan cell is often called a boundary scan register (BSR). Input data enters from either the normal input signal or the Test Data Input (TDI) signal and then output data exits from either the normal output signal or the Test Data Out (TDO) signal, as controlled by their respective multiplexers. The placement of scan cells along the “boundary” rather than at each controlling or controlled node is what distinguishes boundary scannable systems from full scan systems. Although the JTAG committee chose a novel idea, it was not new. What was new, was the suggestion by the committee to use boundary scan cells as the cornerstone of a testability bus standard.\textsuperscript{16,17}
By linking the functional input and output pins of an IC together, a serial shift register is formed. Shifting of serial data into and out of the DUT occurs through a pair of I/O pins. This is analogous to applying and monitoring voltages through probes physical contacting the IC's pins. Extending this principle from a single IC to several on a PCB results in increased testability through a test bus. This bus connects boundary scannable ICs in a serial data path as shown in Figure 7. The added testability is not without penalty. Extra IC pins and PCB signals are now dedicated to test, and additional silicon is required as well. The extra silicon area or "real estate" is proportional to the total number of IC pins plus the extra desired testing functions. This thesis will present several factors used to quantify the cost associated with adding boundary scan compatibility to digital designs. Also presented will be several example designs using testability. The Texas Instruments (TI) scannable Octal D-type Latch (SN74BCT8373)\textsuperscript{18}
is used in some of the examples as it is one of the first ICs manufactured to the IEEE 1149.1 boundary scan specifications.\textsuperscript{4} The specification for the latch is in Appendix E. Some boundary scannable circuits offer powerful test features that are also explored. Moreover since the design of test functionality is on silicon, it can be transparent to the normal circuit operation.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{boundary_scan_path.png}
\caption{Boundary Scan Path on PCB. This extends the boundary scan path from a single IC to a PCB through a test bus.}
\end{figure}
1.4.2 Overcoming Density of ICs

The boundary scan architecture's address to the issue of increasing circuit density occurs on two fronts, silicon design and flexibility. The use of silicon allows testing features to grow as the technology grows. The JTAG/IEEE 1149.1 specifications also leave room for that growth. For example, the current standard specifies that the boundary must be scannable, but to increase testability several functions can be added to the scan path. The use of test methodologies such as BIST (Built In Self Test), PRPG (Pseudo-Random Pattern Generator), and PSA (Parallel Signature Analyzer) can all be used to enhance testability. These methodologies can contribute to the testing of an IC by either generating random input vectors or compressing output vectors into one unique “signature.” The design of these test structures is concurrent and transparent to the normal operation of the IC or PCB being designed. One excellent example of such a test structure is the use of BIST\(^\text{19}\) that initiates from a the JTAG/1149.1 test bus and takes place within the boundary scan architecture. A BIST is an extension of the DFT philosophy where testing functionality is built into the design itself. The test typically consists of a feedback system that generates test vectors and a similar system that collects the results from the vectors. Although a BIST runs independently of boundary scan tests, the JTAG/1149.1 specifications can provide a link into the initializing and running of it.

1.4.3 Overcoming Density of PCBs and Physical Interface

Because boundary scan is developed within the IC, it is independent of the shrinking pin effects that have plagued bed-of-nails approaches. The use of SMT and MCM components does not degrade the testing ability nor does it increase the cost of testing such components. By designing the test features as part of the component, the term
"silicon nail"\textsuperscript{12} is applied to boundary scan devices that can escape the use of a physical probe. Similarly, placing components on either side of the PCB or using multi-layered boards makes no negative impact on testability of boundary scan devices since the interconnects of boundary scan devices are testable, and the boundary scan path can link together ICs on the PCB.\textsuperscript{20} Although power must now be applied to a DUT, the physical interface between the DUT and the tester is limited to the test pins. The limit minimizes the physical interface problems and usually makes them negligible.

### 1.5 Problem Statement

This thesis serves a dual purpose. First, it facilitates the exploration of boundary scan devices with tools that are available at the facilities in the center for Microelectronic and Computer Engineering at RIT. Secondly it allows one to gain familiarity with Tektronix's powerful testing tool, the LV500, currently located in the Microelectronic Engineering test facility.

This thesis accomplishes both of these goals by incorporating the LV500 as a general test tool in the study of example boundary scan digital designs. Three example circuits are used to provide a better understanding of the function of boundary scan cells and the JTAG/1149.1 standard. The first example circuit, consisting of a single TI `8373, explores the BYPASS and SAMPLE operations using the LV500. This example also shows that boundary scan operations run transparent to normal circuit operations. The second example, which has a state machine designed with a TI `8373, shows the value of boundary scan in debugging the combinational logic. The final example steps through the design of a boundary scannable state machine. This example, and the discussion that follows it, provide methods of determining the cost of additional boundary scan
testability. The cost tradeoffs are often the most intimidating hurdle for engineers to cross when deciding if boundary scan compatibility is worth the effort. In choosing to explore both boundary scan and the LV500, my hope is to be straightforward with the most current issues and methodologies.

1.6 System Constraints

The LV500 is used primarily in industry for ASIC (Application Specific IC) verification. It has the capability of generating and analyzing test vectors for up to 256 channels at speeds up to 50 MHz. It can also monitor and plot the results of two dimensional parameter modifications. It is important to note that the choice of the LV500 was based upon its availability in RIT's test lab and its raw capability to complete the testing that was required. The use of the LV500 in this thesis is limited to the generation of test vectors for simple digital logic systems and the generation of boundary scan controlling logic for testing that digital system. Not intended solely as a boundary scan tester, the LV500 sends out test vectors that are boundary scan compatible. As its power goes far beyond what this thesis has used it for, it is an excellent tool to become familiar with and a valuable asset to the repertoire of testing tools in RIT's laboratory. The tutorial circuits of this lab provide are also a stepping stone in the exploring the full potential of the LV500.

The choice of using TI components was made because of their boundary scan capability and their strong presence in this market niche. The components, used in the tutorial circuits provided with this thesis, are also a stepping stone in exploring boundary scannable systems. The enclosed schematics are intended for the test-board maintenance when required.
1.7 Terminology

There may be many new terms and ideas presented to the reader of this thesis. The following sections provide additional detail to the JTAG/IEEE 1149.1 specifications and the Tektronix's LV500 digital tester. A glossary of terms used is found at the front of the thesis text proceeding the first chapter.

1.7.1 JTAG Specification

In 1985 Phillips initiated the European Test Action Group (ETAG) to promote a testing standard for ICs. The group expanded into the Joint Test Action Group (JTAG) when non-European companies such as Texas Instruments and Motorola participated. At the same time the IEEE was forming its own proposal for a test bus standard. The two organizations collaborated and formed the IEEE P1149.1 proposal that was formally accepted in February of 1990. Officially called the IEEE 1149.1 standard, it is often referred to as the JTAG standard because of the Joint Test Action Group's involvement. The goal of the specifications is to allow diversity while maintaining compatibility with other devices. The IEEE 1149.1 standard specifies the physical structure of the test bus and how it can be interconnected to a chip. It also describes the protocol associated with the test bus. The four main test signals associated with the bus are:
- TDI (Test Data In)—Allows the input of data and testing instructions for the IC.
- TDO (Test Data Out)—Allows the data and/or instructions to be shifted out of the IC.
- TMS (Test Mode Select)—Controls the current state of the TAP (Test Access Port). The TAP is the state machine behind the operation of the testing functions.
- TCK (Test Clock)—Clocks the input to the boundary scan device in on the rising edge of TCK and resultant data out on the falling edge.

These signals are used on the boundary scan architecture\(^2\) of Figure 8. The TCK and TMS signals are used directly to drive the TAP, which in turn controls the Instruction Register (IR), Data Register (DR) and various multiplexers for the I/O test data. The IR data may either be shifted out through TDO or used as an address for a DR command. The figure shows a test input (TDI) entering either the IR or the DR and shifting through the BSR and device logic (LOGIC). The data then travels out of the device through the normal output pins or through TDO. Note that all the testing circuitry is in parallel with the normal device circuitry, which allows the two functions to perform without interference.
Figure 8    Boundary Scan Architecture. *This is a high level diagram of the boundary scan logic used in designing a boundary scan device.*
Figure 9 contains the TAP state diagram that controls the mode of the boundary scan device. Operation includes the loading of an instruction into the IR on the right hand side of the state machine and the shifting of data into the DR on the left hand side.
The IEEE 1149.1 specifications list the functions that standard boundary scan devices must be capable of performing, as well as functions defined as extended. The boundary scannable TI '8373 chip contains some of these extended or "suggested" JTAG specifications. Specifically, the '8373 implements a PRPG that provides test vectors for application to the DUT and a PSA to compress the resultant test data into a signature. Faulted components may be identified by comparing their signature to the expected signature for a functionally correct DUT.

The '8373 is an example that shows how scan circuitry can be added to a circuit design to increase its testability. The additional circuitry is fully transparent and its testing capability requires only a four bit wide bus. Furthermore, the boundary scan path need not stop at one IC or one PCB; it may be extended to several boards by simply continuing the scan path. Thus, the four bit test bus facilitates self diagnostics as well as all levels of test, from in-house IC testing to trouble shooting an entire system.
1.7.2 Tektronix LV500

The Tektronix Logic Verifier (LV) is used to test complicated circuit designs. The current system at RIT can handle up to 64 bi-directional signals, driven by up to four independent clocks. The LV500 is primarily used for testing digital circuits and has been modified at RIT by the addition of a 40-pin Zero Input Force (ZIF) testhead. Although at higher frequencies the testhead can be a disadvantage by adding capacitance and resistance to the connection between the tester and DUT, the system is adequate for the operating speed of most normal student projects. With this testhead, any IC directly interfaces to the LV500, as the following tutorial demonstrates. Another feature of the LV500 is its compatibility with the Mentor Graphics EDA (Electronic Design Analysis) tools used in RIT's VLSI Design Laboratory. The application program TekWAVES can be used to translate Mentor Graphics' test vector (event driven) format into the LV500's state driven format*.

Thus, one may design and simulate an IC using Mentor Graphics' tools, fabricate that IC with the aid of RIT's Microelectronics facilities, and then test the circuit using the same test simulation vectors that were used in the design on the HP/Apollo workstations.

* TekWAVES samples the event state changes from the Mentor state file and creates an output file that matches the state format and resolution of the LV500.
2.0 Functional Specifications

The LV500's menus are discussed as they relate to the exploration of boundary scan devices. The first two examples of Chapter 3 utilize the concepts presented in this section.

2.1 Overview of Tutorial

The organization of the tutorial sections of Chapter 3 is similar to a user's manual. The format for each of the examples includes:

1. Overview of Test Purpose—A Textual description of what the test will show with supporting logic diagrams.

2. Test Setup—The steps used to recall stored test parameters on the LV500.

3. DUT Wiring Hardware and Menu—A stepwise installation guide to wiring the testhead circuitry.

4. LV500 Setup Menu—The logical mapping of test vector file signals to DUT pins.

5. Pattern test Vectors—A listing of the test vectors to apply to the DUT.

6. Invoking the test—describes how to start and proceed through a test.
2.2  LV500 Setup and Configuration

This section describes menu information that is displayed on the LV500 and is pertinent to the testing examples. The installation of the testhead is also discussed.

2.2.1 Menus

The LV500 is driven by interactive menus. Physically, these menus are controlled by the keyboard shown in Figure 10. It is divided into the following main groups of keys:

1. A QWERTY (standard style) keyboard.

2. The program-controlled function keys (top row: F1 F8) whose functions change according to the current menu.

3. A dedicated function keys (also on the top row: <Notes>, <, >, <Print>, and <Select Menu>).

4. The display control keys (including the joystick), which provide cursor control.
Figure 10  LV500 Keyboard.  LV500 Keyboard layout and description of standard functions.
When the device is initially turned on, the **Power Up** menu of Figure 11 is displayed, providing information about the system configuration. Be patient, it takes approximately three minutes to fully power up the system. Appendix C discusses how to acquire screen dumps and text files from the LV500. Several other menus will be encountered throughout this tutorial; to see them press the <Select Menu> key. They are categorized (as shown in Figure 12) under four headings:

1. **LV Setup**—Allows the specification of supply voltages, channel assignments, et cetera prior to the running of a test.

2. **Display**—Allows the editing of test patterns, control of the display, and the viewing of test data & results.

3. **Utilities**—Allows the access of system-level tools.

4. **Applications**—Provides access to application software packages.

There are three ways to move about the menus of the LV500. One method is to use the joystick or cursor control keys to position the highlighted cursor on top of the desired menu. Press <Return>, and the highlighted menu will open. The second method of transversing the menus requires the pressing of a function key (F4 through F8). As Figure 12 shows, there are four function keys corresponding to the four menu categories. This will open the menu that is highlighted for a particular category. The third possibility is to hit the <Select Menu> key while in the main menu. This allows for the name of a specific menu to be entered. In a sub menu, the <Select Menu> key can be used to return back to the main menu.

---

* **Bold** font indicates either a menu name or a specified field within a menu.
LU 500  92/164-1  Utilities  Power Up  Idle

TEKTRONIX: LU 500

Configuration: Mainframe

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Controller</td>
<td>PASS</td>
</tr>
<tr>
<td>1</td>
<td>Memory 8 Mbyte</td>
<td>PASS</td>
</tr>
<tr>
<td>2</td>
<td>92LAN Network Controller</td>
<td>PASS</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9216APM 64 Channels/Pattern Generation</td>
<td>PASS</td>
</tr>
<tr>
<td>5</td>
<td>9216AE 64 Channels/Error Storage</td>
<td>PASS</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mainframe F/W  Floppy Disk 1.2 Mbyte  PASS
Hard Disk 43 Mbyte  PASS

System Software Release 2  Version 1.41  Date: 27 June 1991
Previous Shutdown: Normal  Time: 13:43

Press Select Menu or 'Shift Select Menu' to move between menus.
Press 'Notes' twice for a description of the LU 500 NOTES system.

Figure 11  Power Up Menu.  LV500 menu displayed upon initial system power up.

LU 500  92/164-1  Utilities  Power Up  Idle

LU 500 MENU SELECTION

<table>
<thead>
<tr>
<th>Module</th>
<th>LU Setup</th>
<th>Display</th>
<th>Utilities</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>92/164-1</td>
<td></td>
<td></td>
<td>F4</td>
<td>F5</td>
</tr>
<tr>
<td>Sys Config</td>
<td></td>
<td>Pattern</td>
<td>Power Up</td>
<td>LV Toolkit</td>
</tr>
<tr>
<td>Ctrl Setup</td>
<td>Macro</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sys Monitor</td>
<td>Schmoo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Config</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LV Setup</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Template</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Det Schmoo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Press F4, F5, F6 or F7 to move to the selected menu.

Press Select Menu or 'Shift Select Menu' to move between menus.
Press 'Notes' twice for a description of the LU 500 NOTES system.

Figure 12  Main Selection Menu.  LV500 menu showing the four main functional categories.
Open the Orientation menu at this time. It appears under the Utilities category and contains useful information about the LV500. Also, the <Notes> key which provides on-line documentation for the current cursor field. When pressed twice in succession, the notes key describes itself. The rest of this tutorial will provide enough information about the LV500 to step through the boundary scan examples. For a more in-depth look at the LV500's functionality refer to the appropriate Tektronix LV500 manual.23,24,25

2.2.2 Testhead Installation

If the 40 pin ZIF testhead is not the currently installed, then it will need to be done prior to LV500 power up. If you are not familiar with the installation of the testhead, it is wise to ask for the assistance of someone who is. There are gold elastomeric connectors on the underside of the testhead that are easily damaged. These gold "contacts" provide the means to pass the signals from the LV500 testhead to its controller. Avoid touching them because dirt and oil from your fingers may degrade the electrical contacts, and avoid poking at them since this may break the contacts. If it becomes necessary to clean the contacts, do so with a cotton swab and isopropyl alcohol, wiping gently with the "grain" of the gold wires. Figure 13 displays the 40 pin ZIF testhead installed and the LV500's components appropriately connected. Once the testhead is secured, power up the LV500T terminal and then the LV500 controller. This order of power up is preferred because it allows the diagnostic messages to be displayed on the terminal.
**Figure 13**  LV500 System Configuration. *Basic hardware setup for communication between LV500 and tutorial circuits.*

### 2.3 40 pin ZIF Testhead

In this tutorial the 40 pin ZIF testhead will be used as the interface between any DUT and the LV500. Figure 14 shows the proper pin-out for the LV500 to testhead mapping. Since the LV500 has 64 signals and the testhead has only 40 horizontal channels, there are 24 channels on the LV500 that are not connected to the testhead.
Figure 14  40 pin ZIF Wiring Diagram. Pin-out of the 40 pin ZIF testhead showing LV500 port connections.
3.0 Tutorial for Test Boards

This chapter contains the tutorial for exploring boundary scannable devices on the LV500. The use of three examples allows a ramping of the concepts and a concrete foundation for exploring other circuits on the LV500. The first example includes only a single '8373 connected to the LV500. It demonstrates the BYPASS command in test mode of operation, the BYPASS command in normal mode of operation and the SAMPLE command. The second example includes a '8773 with additional combinational logic. This example shows how boundary scannable ICs assist in debugging digital designs. The third and final example utilizes software tools available in RIT's VLSI Laboratory to shows how a circuit's testability is increased through boundary scan design methods.

3.1 Single Test Chip

The single chip test example is used to introduce the world of boundary scan devices. A single '8373 is plugged into the DUT testhead and exercised using the LV500.

3.1.1 BSCAN_1 Test Setup

Now step through the LV500's menus to verify the proper operation of a single SN74BCT8373 IC and provide a basis from which to further examine the '8373's functionality. In order to run a completed test, three components are needed. The following test components can be developed in any order, but they must all be present at the time the device is tested:
- DUT Wiring—A properly wired circuit on the test head.
- LV500 setup—A complete mapping of signals from the test vector file to the
DUT and a description of the timing relationships for each test cycle.
- Pattern test vectors—A complete list of inputs driving the DUT and the
expected outputs.

Once the LV500 is running and the Power Up menu of Figure 11 is successfully
displayed, the BSCAN_I system configuration software must be loaded. This will set
the LV500's current working environment. If still in the Power Up menu, press the <Select
Menu> key to enter the main menu of Figure 12. Now, open the Save/Restore menu
under the Utilities category. A menu similar to the Save/Restore menu of Figure 15
should be displayed.

**Figure 15** Save/Restore Menu. LV500 Menu displayed when saving
or restoring test data.
Set the first field (Operation) of the Save/Restore menu to “Restore Setup” by either using the <Open/Close> key and the cursor arrows, or by typing the command in. Next, enter “BSCAN_I” in the File field as the name of the file to restore. This file contains the system set up for the test to be performed. It was written interactively on the LV500T by myself and saved on this same screen. Upon completing the above steps, press F8:EXECUTE OPERATION to execute the restore. It only takes a few seconds to load the file, after which <Select Main> can be pressed.

![Diagram of LV500 Port Connections, 40 Pin ZIF Wiring, Second Tutorial 8 pin Socket](image)

**Figure 16** Testhead Wiring Diagram. *Pin-to-40 pin ZIF wiring diagram for first tutorial circuit.*
3.1.1.1  DUT Wiring

This is the first of the three sections describing the test components for the DUT. In this section the installation and wiring of the '8373 IC will be performed. Because only one IC is being used, it will be installed directly into the 40 pin ZIF testhead. If the IC is not currently in the testhead then plug it in, conforming to Table 1 below. The IC should be installed such that its first pin contacts port 0.0 of the testhead (and the 12th pin contacts port 1.6). As Figure 16 shows, the IC is inserted directly abutting the end of the 40 pin ZIF closest to you. Because the '8373 IC has VCC and GND pin mappings that differ from the MOSIS 40 pin standard, some of the signals must be routed to alternative channels.

The '8373 is an octal d-type latch when operated in normal mode of operation. The IC is also fully IEEE 1149.1 compatible when operated in its test mode. When in test mode, the normal operation of the octal latch is inhibited and the test circuitry is enabled to observe and control the device's I/O boundary. Several instructions can be loaded and executed by using the four standard test lines (TDI, TDO, TCK, TMS). In addition to the standard features for IEEE 1149.1 compatibility, the octal latch can also perform additional tests such as PRNG and PSA. The following commands (Appendix E contains detailed description) are executed in this tutorial:

- **EXTEST**—Boundary scan register is selected in the scan path. Data previously loaded into the BSR is applied to device inputs and through device outputs. Device is in test mode of operation.

- **BYPASS**—One bit bypass register is selected in the scan path. A logic 0 is loaded into the bypass register. Device operates in normal mode.
- SAMPLE—Boundary scan register is selected in scan path. A snapshot is taken of the device inputs and outputs. The device operates in normal mode.

<table>
<thead>
<tr>
<th>Logical Path Name</th>
<th>Logical Signal Name</th>
<th>Sector/Channel</th>
<th>DUT Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1Q</td>
<td>0.2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2Q</td>
<td>0.4</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3Q</td>
<td>0.6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4Q</td>
<td>0.a</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
<td>1.2</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>5Q</td>
<td>0.c</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>6Q</td>
<td>0.c</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>7Q</td>
<td>1.0</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>8Q</td>
<td>2.0</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>TDO</td>
<td>1.4</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>TMS</td>
<td>1.6</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>TCK</td>
<td>1.7</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>TDI</td>
<td>2.3</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>8D</td>
<td>1.3</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>7D</td>
<td>1.1</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>6D</td>
<td>0.f</td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td>VCC</td>
<td>1.5</td>
<td>18</td>
</tr>
<tr>
<td>18</td>
<td>5D</td>
<td>0.d</td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>4D</td>
<td>0.9</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>3D</td>
<td>0.7</td>
<td>21</td>
</tr>
<tr>
<td>21</td>
<td>2D</td>
<td>0.5</td>
<td>22</td>
</tr>
<tr>
<td>22</td>
<td>1D</td>
<td>0.3</td>
<td>23</td>
</tr>
<tr>
<td>23</td>
<td>*OC</td>
<td>0.1</td>
<td>24</td>
</tr>
</tbody>
</table>

**Table 1**  Logical Mapping of DUT. *Table listing the mapping of the Logical Signal name with the LV500 channel and DUT pins.*
Note that the DUT Pin to Logical Signal Name mapping is determined from the '8373 specification sheet of Appendix E p 1. The mapping of Table 1 was chosen to minimize the number of hand wired signals.

<table>
<thead>
<tr>
<th>Logical Path Name</th>
<th>LV500</th>
<th>DUT Wiring</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
<td>0.2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>CC</td>
<td>0.4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>CC</td>
<td>0.4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>CC</td>
<td>0.4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>CC</td>
<td>1.2</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>CC</td>
<td>0.8</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>CC</td>
<td>0.4</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>CC</td>
<td>1.0</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>CC</td>
<td>2.0</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>CC</td>
<td>1.4</td>
<td>11</td>
</tr>
</tbody>
</table>

**Figure 17** DUT Wiring Menu. LV500 DUT wiring menu displayed to verify pin-to-LV500 mapping.

Before test vectors can be applied to the DUT, the LV500 must first know how the DUT's signals are mapped to the LV500's channels. This is where the DUT Wiring menu of Figure 17 is required. Invoke this menu now. The DUT Wiring menu shows how the physical channels of the LV500 are mapped to the logical data columns of the
test vectors. If you are unsure of how table is set up, look at the specifications for the '8373 to get the pin-out of the IC and refer back to Table 1. Since this IC is plugged directly onto the testhead, the pin to port number mapping can be confirmed by inspection. These numbers are then entered in the **DUT Wiring** menu to complete the mapping. This procedure is only altered for the four pins that are not directly mapped. The three main columns of the **DUT Wiring** menu are:

1. **Logical Signal Name**—The logical name used to refer to the signal.
2. **Sector/Channel**—The physical location to which the signal is mapped. It is determined from the position on the IC on 40 pin ZIF testhead.
3. **DUT Pin**—The DUT pin mapping which affects no other menus. It shows how the device pin numbers are mapped to the LV500 channels.

This menu is the only place where the pins are mapped. Take the time to double check the connections. If you wish, to move the IC to a different location on the testhead, then be sure to make the appropriate changes to the **Sector/Channel** field of each signal.

### 3.1.1.2 LV500 Setup

Once the DUT wiring is complete, it is time to configure the LV500 itself. The LV500 setup consists of the mapping of each signal for the test vector file to an appropriate pin on the DUT and then describing the timing relationships between the signals for each test cycle. At this time invoke the **Channel** menu. Figures 18 & 19 display this menu for the groups D and Q respectively. The **Channel** menu allows signals to be grouped together as a bus. Therefore it allows for easier data entry, viewing of test results, and specification of a group's timing relationships. Although the signal names may not be
changed here, this menu also allows one to verify that the logical signal and channel number are correct since the channel numbers are displayed.

As for the specific fields of the Channel menu, the Logical Signal field is used to specify the name of the signal in the current group. The next field, labeled Timing, specifies whether the signal is timed for the same clock as the group or whether it is independently clocked. Finally, the Column Mask field allows a signal to mask (disable) the comparison between expected and acquired data for the signal by entering an “ON” in this field. The comparison specifics, discussed fully in the following section, are useful in bypassing a particular signal while Pass/Fail results are still desired on the other signals. Note that the VCC and GND signals are left out of the channel menu because these pins are ignored (high impedance state mode) as far as the LV500 is concerned. This is fine since the signals should be hardwired from the DUT to the appropriate pins of the ZIF Figure 18 also shows an open Channel field, on the right hand side, with the names of the defined channels. Take a moment to review the channels that are set up.
**Figure 18** Channel D Menu. *Display of the LV500 menu grouping the D input of the '8373.*

<table>
<thead>
<tr>
<th>LV500</th>
<th>B7</th>
<th>D1</th>
<th>Setup</th>
<th>Channel</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>1</td>
<td>1</td>
<td>Menu</td>
<td>D1</td>
<td>Off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Digital Input</th>
<th>Channel</th>
<th>Timing</th>
<th>Column Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0.2</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>10</td>
<td>0.6</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>19</td>
<td>0.9</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>58</td>
<td>0.1</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>70</td>
<td>1.0</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>68</td>
<td>2.0</td>
<td>Group</td>
<td>Off</td>
</tr>
</tbody>
</table>

**Figure 19** Channel Q Menu. *Display of the LV500 menu grouping the Q output of the '8373.*

<table>
<thead>
<tr>
<th>LV500</th>
<th>B7</th>
<th>D1</th>
<th>Setup</th>
<th>Channel</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>1</td>
<td>1</td>
<td>Menu</td>
<td>Q1</td>
<td>Off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Digital Input</th>
<th>Channel</th>
<th>Timing</th>
<th>Column Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0.2</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>20</td>
<td>0.4</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>10</td>
<td>0.6</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>19</td>
<td>0.9</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>58</td>
<td>0.1</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>70</td>
<td>1.0</td>
<td>Group</td>
<td>Off</td>
</tr>
<tr>
<td>68</td>
<td>2.0</td>
<td>Group</td>
<td>Off</td>
</tr>
</tbody>
</table>
Now close the **Channel** menu and invoke the **Template** menu. This menu is used to define the data format and timing relationships for all channel groups for one test cycle. Figure 20 shows a detail drawing of the **Delay** and **Width** lengths for the phase B TCK signal of Figure 21. Figure 21 shows the “bscan_stndrd” template used in this tutorial and Figure 22 shows “play_templat” which contains some other useful arrangements of the signal clocks. Towards the top of the menu lies the **Resolution** field, which indicates the smallest increment that can be used to specify other timing requirements. It can be set at one of three time bases. This tutorial uses an internal high speed time base providing a 500 ps leading edge placement and a cycle length from 20 ns to 496 ns. Another option allows a 20 ns leading edge placement and cycle lengths from 200 ns to 4960 ns. Finally, a third possible timing option allows an external clock to drive the LV500. The desired period of the clock is entered in the **Length** field.

![Diagram](image)

**Figure 20**  Detail of Clock Phase Signals. *A graphical description of the fields that make up each clock signal.*
**Figure 21** Template Menu "bscan_stndrd." *The standard boundary scan template.*

<table>
<thead>
<tr>
<th>Phase</th>
<th>Delay (ns)</th>
<th>Width (ns)</th>
<th>Leading Edge Time (ns)</th>
<th>Trailing Edge Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OA</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>150</td>
</tr>
<tr>
<td>OB</td>
<td>40</td>
<td>70</td>
<td>40</td>
<td>110</td>
</tr>
<tr>
<td>OC</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>OD</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

**Table 2** Description of Phase Times for Figure 21. *Table of delay and width times for each clock phase and how they relate to the beginning and ending trigger times.*
Figure 22  Template Menu “play_templat.” An example LV500 template menu showing various timing options followed by a detail timing diagram.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Delay (ns)</th>
<th>Width (ns)</th>
<th>Leading Edge Time (ns)</th>
<th>Trailing Edge Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OA</td>
<td>10</td>
<td>100</td>
<td>10</td>
<td>110</td>
</tr>
<tr>
<td>OB</td>
<td>100</td>
<td>40</td>
<td>100</td>
<td>140</td>
</tr>
<tr>
<td>OC</td>
<td>20</td>
<td>50</td>
<td>20</td>
<td>70</td>
</tr>
<tr>
<td>OD</td>
<td>0</td>
<td>190</td>
<td>0</td>
<td>190</td>
</tr>
</tbody>
</table>

Table 3  Description of Phase Times for Figure 22. Table of delay and width times for each clock phase and how they relate to the beginning and ending trigger times.
Because RIT's LV500 currently has 64 I/O channels, there are four separate clocking phases that can be utilized. These are shown in the Clock Phase field. Each phase can have a unique Delay time and a Width time, but all phases are bound by a common clock cycle length. The Delay time specifies the time prior to leading edge actions. The Width time specifies the time between leading edge and trailing edge actions. Hence, leading edge actions take place after time Delay, and trailing edge actions take place at a time of Delay plus Width. Tables 2 & 3 list the leading and trailing edge times for each phase of Figures 21 & 22 respectively.

Each group or signal can be used as either a controlling node or as an observing node as defined by the Pin Function and Pin Format fields. The options are:

1. MASK—Mask out channels into a high-impedance state (no format options).
2. FORCE—Force the test vectors to the value provided in the Pattern menu.
   a. DNRZ L—Force a delayed, non-return to zero, signal on the clock pulse's leading edge.
   b. DNRZ T—Force a delayed, non-return to zero, forced on the clock pulse's trailing edge.
   c. R1—Force a return to logic 1 prior to, and after, the clock pulse.
   d. R0—Force a return to logic 0 low prior to, and after, the clock pulse.
   e. R INH—Force a return to the high impedance state (inhibit) prior to, and after, the clock pulse.
3. COMPARE—Compare the actual data from the DUT against the expected data as indicated in the Pattern menu.
   a. EDGE L—Compare data on the leading edge of the clock.
b. EDGE T—Compare data on the trailing edge of the clock.

c. WINDOW—Compare a window of data for glitches.

Take a moment to look through Figures 21 & 22 to see how each of these Pin Functions and Formats work. For a boundary scan IC the TMS signal has to be stable over the leading edge of TCK and the output of TDO is valid after the trailing edge of TCK has failed. This is graphically displayed in Figure 20. While referring to this figure, try to answer these questions:

- When do TMS and TDI go active?
  The TMS and TDI signals go active almost immediately since the resolution is set to 500 ps.

- When are the rising and falling edges of TCK?
  The rising and falling edges of TCK are at 40 ns and 110 ns respectively from the delay time and width of the template.

- When is the output TDO considered valid?
  The boundary scan specification state at the falling edge of TCK. From the template this has been chosen to be at 150 ns.

This concludes the discussion of the Template menu. Now that the setup of the LV500 is complete, bring up the main menu once more to allow access to the Pattern and Macro menus listed under the Display category.
3.1.1.3 Pattern Test Vectors

At this time the Pattern menu of Figure 23 should be invoked. Now that the first two components of a complete test are created (DUT wiring and LV setup), it is time to create the test vectors. Test vectors can be created in one of two ways: they can be either translated from another computer system (e.g., HP/Apollo running Mentor Graphics software) or typed in. For this example, the vectors were interactively typed in.

```
<table>
<thead>
<tr>
<th>Line</th>
<th>Display</th>
<th>Pattern</th>
<th>Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>null</td>
<td>TDI TDO</td>
<td></td>
</tr>
</tbody>
</table>

This program is used to explore the TI Boundary
3 | EPH1 register then the E-type latch, and finally
4 | T入学 register will be explored.
5 | T入学 held low for 40ns prior to initial testing. This
6 | allows time for the TIE to reach on initial state.
7 | For II: EPH1 register on.
8 | For III: I-type latch under normal operation
9 | For III: SAMPLE register on.
10 | POWER_UP
11 | RESET

Test_logic=Reset On

Figure 23 Pattern Menu. LV500 menu that allows macro patterns to be written for later use.
```

Note that included in this Pattern menu are calls to the POWER_UP and RESET macros, referring to the macro header comments for their purpose. The two macros are shown in Figures 24 and 25 respectively. Macros allow frequently called modules to be written once and are also capable of accepting parameters to use in pattern generating.
These two macros are the only ones that are used in the BSCAN_I example and can be examined now through the Macro menu under the Display category. Return to the Pattern menu when finished.

**Figure 24**  POWER_UP Macro Menu. *LV500 macro used to all the system to power up into a valid state.*

**Figure 25**  RESET Macro Menu. *LV500 macro used to reset the boundary scan features.*
Move the cursor to a line in the **Pattern** menu beginning with the template name “bscan_stndrd” and note that the signal values and logical names are listed near the top. In this example, all of the groups are represented in binary format. Depending upon whether the signals perform a **Force** (yellow font) or a **Compare** (white font), the vectors displayed will respectively force a voltage out to the DUT or compare input from the DUT with a given reference voltage. The I/O signals and their order are decided in the **Template** menu of Figure 21. From the comparison the current logic level of the DUT’s output (either a logic 1 or a logic 0 for a high or low voltage level respectively) is determined. If no single step commands are in the program, the actual testing of the DUT should take about three seconds for this example. Try running the test by pressing F1:START. If it does not run properly then the message “CAUTION: run: Failed” will be displayed in the lower left hand corner, written in yellow font. To trouble-shoot the design if it did not pass, start with the wiring to find out what went wrong. If the test finished without an error, the message “run: Passed” in white font appears. How did it work? What voltage values were sent to the DUT? The answer to the latter question can be found on the **Config** menu which is listed under the LV500 Setup category. Take a moment to exit this menu and invoke the **Config** menu of Figure 26.

In the **Config** menu, device power supply voltage and current limits are set as well as the drive voltages and compare threshold values. There are also two families that can be defined so that different logic families may be tested on one DUT. This tutorial will use the V1 family exclusively. The values are of standard TTL/CMOS voltage levels (0v to 5v) and all of the fields are with reference to ground. Now for some configuration specifics, the **Force High Voltage** and **Force Low Voltage** fields are set at 2.40v and 0.50v and represent the voltages that are set at the LV500 for logic families “1” and “0” respectively. The threshold or **Compare Voltage** field (set at 1.40v) is used to determine
the logic value of the DUT's output. Voltages that are greater than this compare voltage are considered "high" and values less than or equal to this voltage are considered "low". At the bottom of this menu appears the final field, the **Sector Logic Section** which specifies which logic families the sectors of the testhead receive. A sector is a group of sixteen signals, hence with 64 channel LV500 only four separate sectors are possible. You may now invoke the **Pattern** menu so that the test vectors may be single stepped.

---

**Figure 26**  **Config Menu.** *The LV500 configuration menu used to alter default forcing and comparing voltage levels.*
3.1.2 Single Stepping Test

In this section of the tutorial, edit the Pattern so that the program may be stepped through. Look at the function currently displayed under the F2 key. This key sets the current Pattern menu mode to either EDIT MODE or SEARCH MODE. If the F2 key specifies EDIT MODE, then press F2 once to enter the edit mode. Otherwise, the editing mode is currently selected and you may proceed to modify the program. On Figure 21 it should now be apparent what voltage levels are represented by the logic values. How do these voltage levels control the boundary scan IC? Insert the command “BEGIN_SS” to invoke the single step operation of the LV500. The control of the program will remain in single step mode until the command “END_SS” is entered. Before stepping through the TAP of this example, it is wise to have a copy of the state machine present (Figure 9 would be sufficient). Appendix A contains timing diagrams (Figures 79, 80 and 81) and timing tables (Tables 10 and 11) for this tutorial. They will be extremely helpful in stepping through the examples.

The pattern for BSCAN_1 is divided into the following tests that can be individually explored by stepping through each of them:

1. Inspection of the BYPASS Register with the D-type latch not operating.
2. Inspection of the BYPASS Register with the D-type latch in normal mode operation.
3. Exploration of the SAMPLE Boundary Operation, which captures the data on all of the I/O pins.

The program will now be set up to step through tests 1 and 2. Move the cursor to the first line of code and press F8:ADD LINE. This will duplicate the current line. Now move the cursor over to the Select field and type in the command “BEGIN_SS”. Press
<Return> to change the previous command line of the **Pattern** menu to “BEGIN_SS”, indicating the point where the single stepping starts. In the same manner, add a line at the end of the patterns generated for the first two examples. This location is immediately prior to the comment “Part III: SAMPLE register scan third test”. The command line will be changed into the “END_SS” command, which indicates the end of the single stepping. At this point the program should be run and stepped through by pressing the F1:START key. Progression through the program is provided by the F4:SINGLE STEP key. Each time it is pressed, a single step is taken. Pressing F1:STOP, halts the program. Note that each step of the program issues the pattern only once and is in accordance to the timing specifications of the **Template** menu of Figure 21.

For the first test of the single ‘8373 IC example, the BYPASS instruction will be loaded into the IR and bypass the bit stream “101” from TDI to TDO. Figure 27 shows the BYPASS scan register that performs this operation. Before stepping through this program, realize that the outputs produced are valid at the falling edge of TCK. It is important to explain this subtlety. If the outputs are sampled at the falling edge, they will appear to be generated during the same clock pulse the inputs were made available. This is why the BYPASS operation shows the *outputs ready* during the *same pulse* that the *inputs are set*. However: if the outputs are sampled on the rising edge of TCK, the outputs will appear to be delayed by one TCK pulse. Remember this when creating new templates to explore the ‘8373 or other boundary scannable ICs to avoid potential timing confusion. Now step through the first test, while ignoring the I/O signals for the D-type latch.
Figure 27  BYPASS Scan Register. *High level representation of a bypass scan register.*

TCK 0  Step through the POWER_UP macro to ensure the IC has time to properly initialize itself. Each vector line of the macro is also considered a step, causing several iterations to take place.

TCK 1  The RESET macro steps the TAP into the Test-Logic-Reset state, regardless of any previous states. This is done by holding TMS high for five clock pulses.

TCK 2-5  The TAP is then stepped through Rut-test/Idle and the Capture-IR state. It is during the Capture-IR state that the instruction "10000001" is preloaded into the IR. This preload operation occurs every time the Capture-IR state is entered.

TCK 6-13  The TAP then steps into the Shift-IR state where it remains for a total of eight TCK cycles. This allows the new instruction (eight bits wide) to be shifted into the state machine, and the old (preloaded "10000001") instruction to be shifted out. The new instruction, "11111111", represents the BYPASS operation and is shifted in from LSB to MSB as Figure 28 shows. Incidentally, the shifting of data into the IR does not occur on the rising edge of TCK that brings the TAP to the Shift-IR state. Data begins shifting into the IR on the rising edge of TCK after the Shift-IR state is entered. The last bit (MSB) is shifted in on the first rising edge of TCK.
once the Shift-IR state is exited. This order of shifting delays the shifting of the last instruction bit (MSB) until the Exit1-IR state (TCK 14) has started.

TCK 14  TMS is driven high to force the transition to this state. The last bit of the instruction shifts in as the TAP advances from the Shift-IR state (TCK 13) to the Exit1-IR state (TCK 14).

Figure 28  Instruction Register. *High level representation of instruction register cells in a standard JTAG/IEEE 1149.1 IC.*

TCK 15  Once the BYPASS instruction is shifted in, the TAP enters the Update-IR state and the IR's new instruction is instantiated. The instruction, previously stored in the shadow latch, is now made available to the rest of the system.

TCK 16  Select-DR-Scan state.

TCK 17  The TAP's state machine is then stepped to the Capture-DR state. Here the BYPASS register is loaded with a logic 0, this is shown as the first bit shifts out of TDO during the bypass operation.

TCK 18-24  The Shift-DR state is next. Input from TDI becomes valid and the first bit of the BYPASS register (the "0" that was previously loaded) is shifted out through TDO. The bit pattern "101" is then be shifted through the BYPASS register over the next three clock cycles with the last bit being
shifted out during the Exit1-DR state. Note that the LV500 is set up to sample the output of TDO the same cycle that TDI is driven. This causes the output to appear valid immediately after it is set.

TCK 25 The state machine is stepped to the Test-Logic-Reset state once more, and this program is finished.

At this point you may wish to alter the bit pattern sent to the BYPASS register to see how it affects the output bit pattern. These bits are present in the Shift-DR state. When changing the bits, first alter only the input bits and observe how the invalid outputs are flagged by the LV500. When finished, set the bit pattern back to the original so that the test completes without failure. Now move to the second example dealing with the D-type latch under normal operation.

A desirable feature of boundary scan devices is the capability to execute existing testing operations without interfering with normal circuit operation. In the second test of the single '8373 IC example, the transparent operation of boundary scan testing is explored using the BYPASS instruction. Both normal and testing operations are performed in this example. The normal and test circuit values are shown by the far right hand fields and the four left hand fields of the Pattern menu respectively. The same vector program loaded for the previous BYPASS instruction test is used. Refer to the last four fields of each test vector on the Pattern menu. They specify a desired I/O data sequence for the C, *OC, and D inputs and Q outputs respectively. Follow the data and note that when C is active and *OC is low, the data on the latch passes data through from D to Q. It is possible to alter some of the latching signals to see how they affect the pattern. Note that while the latch is running under normal operation the BYPASS instruction test is also running transparent to the latch! There is no need to worry about interference from the
boundary scan operations when they are being run under controlled operation. Before moving on to the next example, remove any step commands that were inserted. Make sure to remove the single step commands as well. This is done by placing the cursor over the single step command and pressing F7:DELETE LINE. Finally, run the program once more to ensure that any faults that were introduced have been corrected. If the original configuration is lost, it can be reloaded from the Save/Restore menu.

In this third test of the single '8373 IC example, the single step commands must again be entered to allow a stepwise inspection of the program. The state machine operates as in the first test program, but this time the SAMPLE instruction will be loaded and a sample of the Boundary Scan Register (BSR) will be performed. In this test program, the commands “BEGIN_SS” and “END_SS” must again be entered into the Pattern menu. Place the stepping commands on both sides of the code that performs the sample. This will minimize unnecessary stepping. The sample is performed during the Capture-DR state that precedes the loading of the SAMPLE instruction into the IR. Figure 29 shows a valid placement of the single step commands. The specific capture command occurs on the rising edge of TCK once the Capture-DR state is reached, not the rising edge that takes the TAP into the Capture-DR state. The following is a literal description of the state machine operations for this test program.
The RESET macro is used to reset the TAP. Upon completion the TAP will be in the Test-Logic-Reset state.

The TAP is then stepped to the Capture-IR state which preloads “10000001” into the IR.

The TAP is then taken into the Shift-IR state, where it remains for a total of eight TCK cycles. This allows the preloaded instruction to shift out through TDO and the new instruction “00000010” to shift in through TDI.

This command specifies the SAMPLE instruction which is shifted in from LSB to MSB. Remember the delay of one TCK pulse that occurs in shifting the data into the IR.

Figure 29  Placement of Single Step Commands. Proper placement of the “END_SS” instruction single step commands.
TCK 14  TMS is driven high to force the transition to this state. The last bit of the instruction is shifted in as the TAP advances from the Shift-IR state (TCK 13) to the Exit1-IR state (TCK 14).

TCK 15  Once the new instruction is loaded into the IR, the TAP's state machine is stepped past the Update-IR state, which latches the IR from the shadow latches and into the Select-DR-Scan state.

TCK 16  Select-DR-Scan state.

TCK 17  The next state is Capture-DR, where the actual data capturing of the BSR is performed. Figure 30 shows the order in which the "SAMPLE" of the BSR is taken (i.e., the order in which the bits are captured). As in the shifting operations, the physical capture of all the registers is performed on the next rising TCK edge once this state is entered. In other words, the capture is performed on the rising TCK edge taking the TAP into the Shift-DR state.

TCK 18-35  Once the Shift-DR state is entered, the TAP stays there for eighteen clock cycles. This allows the contents of the BSR to be shifted out through TDO. The MSB is shifted out as the TAP moves from the Shift-DR to the Exit1-DR state.

TCK 36  Exit1-DR state is entered.

TCK 37  Update DR state.

TCK 38  Select-DR-Scan state.

TCK 39  Select-IR-Scan.

TCK 40  The state machine is stepped to the Test-Logic-Reset state once more, and the program is finished.
In exploring the SAMPLE scan register functions, two methods may be used in setting up the register values. The first method consists of altering the bits present on the device inputs via the LV500's pattern. At present, all of the LV500's signals for the latch are set low. That is why all of the SAMPLE register bits are also low. By changing any of these bits to a logic 1, the value will be latched in the BSR and shifted out through TDO at a later time. As Figure 31 shows, only one D signal to the latch needs to be altered into a “high” signal. This is enough to flag an error on the output of the LV500. The error occurred because a logic 1 was shifted out of TDO for bit D8 instead of a logic 0 as was expected.

Figure 30 SAMPLE Scan Register. High level representation of scan path showing order scan is taken in.
The second method of presenting data for the BSR register consists of externally applying a voltage to the IC inputs. Prior to doing this, the LV500 sets these signals in the high impedance state to avoid a possible short or damage to the testhead. There are several templates available which describe how the signals are applied to the DUT. One of the templates, namely "mask_LATCH", is already set up to mask the latch's D and Q signals. This template may be used to replace the "mask_TDI" template that currently exists by moving the cursor over the Select field and typing in the desired name (i.e., "mask_LATCH"). Figure 32 displays what the Pattern menu should look like when this procedure is finished. Now the test may be single stepped to the "mask_LATCH" template and a voltage signal applied to the input of the latch. A 5v signal may should be applied to the input of the latch for one step only, and then removed for the remainder of this example. Note that the signal must be tied before the cursor is stepped to the line that captures the BSR because the cursor has already issued the pattern of the current line. Return to the main menu and open the remaining LV Setup menu: Def Schmoo. In the next section the device specifications of the '8373 are examined using the Schmoo menu.
Figure 31  Setting Bit D8 “high” to Flag Error.  *LV500 pattern menu*

displaying error message due to D8 being set high.

Figure 32  Replacing mask_TDI with mask_LATCH template.  *Result of successfully completing first part of tutorial circuit.*
3.1.3 Schmoo Specification Tests

At this time the **Def Schmoo** menu of Figure 33 is introduced. It allows you to automatically repeat test runs while altering one or two system parameters. The test results are displayed in a two dimensional scatter plot diagram representing pass and fail values. Because the resulting graph can resemble any shape, similar to Al Capp's cartoon character, it was appropriately dubbed *Schmoo* plotting. This menu utilizes the full power of the LV500 as a testing tool. Figure 33 displays the open **Variable** field, listing several variables that can be single stepped. By modifying these values, device limitations such as: maximum clock speed, minimum clock width and power limitations are determined.

![Def Schmoo Menu](image)

**Figure 33** Def Schmoo Menu. *LV500 menu used to vary two system parameters simultaneously for testing.*
There are two examples that are used to confirm the '8373 device characteristics and examine the capabilities of Schmoo testing. For each example the Def Schmoo menu of Figure 33 is invoked to set the appropriate device parameters prior to testing. With one variable, the results are displayed as a horizontal graph. With two variables, one represents the abscissa or horizontal graph axis, while the second represents the ordinate or vertical axis. In the following examples, two parameters are modified simultaneously.

3.1.3.1 Input Voltage Limits

This is the first section describing the Schmoo test for the '8373 IC. The input voltage limits of the system specifications will be compared and the results examined. These voltage limits are critical in determining what device families can interface with this circuit. They also assist in trouble shooting faulty components that are running close to these limits. Two voltage limits will be tested, namely the maximum low input voltage and the minimum high input voltage.

The '8373 specifications (Appendix E p14) list these voltages to be:

\[
\begin{align*}
V_{IL} \text{ (Max)} &= 0.8v \\
V_{IH} \text{ (Min)} &= 2.0v
\end{align*}
\]

To test the voltage levels, a range of valid and invalid numbers must be entered. For example:

\[
\begin{align*}
0.8v &\leq V_{IL} \text{ (Max)} \leq 1.3V \\
1.9v &\leq V_{IH} \text{ (Min)} \leq 2.5v
\end{align*}
\]
Using the **Def Schmoo** menu to enter the appropriate ranges, the chosen input for each parameter is:

**Variable X:** Force Low voltage V1  
From: 0.80v  
To: 1.30v  
By: 0.05v

**Variable Y:** Force High Voltage V1  
From: 1.90v  
To: 2.50v  
By: 0.05v

After entering the above values, invoke the **Schmoo** menu to run the test and view the results. The actual tests can be run from any screen, but it is desirable to run Schmoo tests from the **Schmoo** menu so that the results can be viewed in real time. Press F1:START and note how the tests are performed sequentially. Each test represents a particular value pair of the desired voltage ranges. The final results should be similar to Figure 34.

How is the graph interpreted and what does it imply? Given the above specifications it is expected that all the tests with $V_{IH} (\text{Min}) \geq 2.0v$ and $V_{IL} (\text{Max}) \leq 0.8v$ would pass and the remaining cases fail. At a quick glance that appears to be true. However; the circuits response is not linear and the pass-fail line is jagged. As might be expected, the circuit's response is least predictable when both the input high and input low voltages are near their limits. Run the test several times to verify the output pattern. Although the component passed the desired tests, the failures show a possible dependence upon the driving input high and low voltages. Part of this reason is the shared ground line that connects the input and output signals. In CMOS circuits, the greatest current is drawn
when the circuit is switching between states. With a large current draw there may not be enough current available for normal operation when the circuit is near the fringe of its specifications. Since the switching period is 200 ns, the speed of switching is 5 MHz. Another possible test would include running the test under varying operating frequencies. Unfortunately the Schmoo menu does not provide a way to scale the phase parameters based upon the cycle time. Varying frequencies must be entered manually for each template menu. This exercise is left for the user.

![Graph showing input voltage limits for the '8373.](image)

**Figure 34** Input Voltage Limits. *Examine the maximum low input voltage and minimum high input voltage for the '8373.*
3.1.3.2 Propagation Delay

The second example is used to examine the propagation delay for the D latch circuit. The maximum delay must be known in order to determine timing constraints for devices interfaced with the '8373. To measure the propagation delay, the time that the output data is sampled is moved towards the time at which the input is modified. As the propagation time is swept to zero, there will be a time when the data present at the input will not have time to propagate to the output. That time defines the minimum propagation delay. In order to determine the propagation delay for the circuit, the times for D (driving data) and Q (observing data) must be modified.

One way to alter these times is through the phase delay. Since they are modified independently, they must have independent clock phases. Invoke the Template menu to insure the correct phase is adjusted. Select the template Mask_TDI&TDO to verify that D is being updated in Phase A (0A) and modify the clock phase of Q to be Phase C (0C). Also make sure that the comparing of data is done at the leading edge of the clock. This allows the Delay time field to control the time Q is sampled. Refer to Figure 35 for a timing diagram of the appropriate clock phases and propagation delay time (tp).

The specifications of the '8373 (Appendix E p 16) list the maximum time required for data to propagate from input D to output Q as:

\[
Q(t) = D(t - T_{\text{PROP}})
\]

Such That: \( T_{\text{PROP}} = 10.5 \text{ ns} \)

The propagation delay can be verified by sweeping the ranges for D and Q:

\[
0.0 \text{ ns} \leq \text{Time D is set} \leq 4.5 \text{ ns}
\]

\[
12.5 \text{ ns} \leq \text{Time Q sampled} \leq 17.0 \text{ ns}
\]
Figure 35  Determining propagation time from D to Q. By moving the sampling time of Q toward the driving time of D, the propagation delay is verified.

Since the Def Schmoo menu allows each of the four clock phases to be modified, verify from the Template menu that the correct phase is being adjusted. Phase A (0A) is used for setting the D input signals, phase C (0C) is used to trigger the sampling of the Q output signal.

Using the Def Schmoo menu to enter the appropriate ranges:

Variable X:Phase Delay
Template: Mask_TDI&TDO Clock: 0A
From: 0.0 ns To: 4.5 ns
By: 0.5 ns
Variable Y: Phase Delay

Template: Mask_TDI&TDO   Clock: 0C
From: 12.5 ns   To: 17.0 ns
By: 0.5 ns

After entering the above values, invoke the Schmoo menu and press F1:START to begin the test. Note how the tests are performed sequentially, with each test representing a particular value pair of the phase delay. The final results should be similar to Figure 36. Note how the bottom staircase shows that valid results are produced when the time Q is sampled at least 13.0 ns greater than the time D is modified. This is slightly larger than the specified 10.5 ns, but it is very consistent through the range of input delays tested.

When the delay of D's input increases (e.g., 3.0 ns) the valid sampling time of Q's output also increases (e.g., 16.0 ns).
Figure 36 Propagation Delay Time. Examining the time for data to propagate from the D input pin to the Q output pin when latch is open.

At this point in the tutorial, you are free to modify the LV500's Pattern menu to explore the previous examples. When finished with the modifications, begin the next section. There is no need to reset the pattern lines, because the configuration is saved on the hard drive and the next section requires loading new LV500 system configuration vectors.
3.2  Single Chip with Combinational Logic

This section will use a state machine as an example of how the boundary scannable '8373 can aid in circuit testing and fault locating. This section will also provide more time on the LV500 to refine testing skills.

3.2.1  Coffee Break

The previous section dealt with only one IC being tested. This section will allow a small-circuit board to be examined. This board design will contain seven ICs and allow the four bit boundary scan test bus to be utilized while running a simple state machine. The state machine describes the action of a person who is in constant pursuit of a coffee break. The pursuit is hindered by only one factor: the boss. Throughout the day an employee works, makes coffee and takes coffee breaks while watching out for the boss. If the boss looks in, the employee must get back to work as soon as possible (i.e., the next state!).

3.2.2  Design of State Machine

This section discusses the state machine that was used and how it was transferred into hardware. This will be the only state machine review for this tutorial. The State Diagram of Figure 37 and the state assignment table (Table 4) describe the desired operation of the Coffee Break state machine. The basic flow of the state machine is:

1. The employee starts working diligently for his or her boss.
2. The boss peers away and the employee walks to the coffee machine.
3. The employee makes coffee (inserts filter, coffee and turns machine on).
4. The employee waits while the coffee is brewing.

5. The coffee is finished and the employee enjoys a coffee break.

6. The above process repeats.

Note that if at any point in the flow of the process the boss watches the employee, the employee must be “Working” in the next state (i.e., next state will be either “000” or “010”). There are six valid states thus, three state bits are required. There are also three output LEDs which are defined as RED for when the employee is working, YELLOW for when the coffee is done brewing, and GREEN for when the employee is enjoying a coffee break.

<table>
<thead>
<tr>
<th>Present State</th>
<th>State Name</th>
<th>LED</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Working at Desk</td>
<td>R</td>
</tr>
<tr>
<td>001</td>
<td>Go to Coffee Machine</td>
<td>R</td>
</tr>
<tr>
<td>010</td>
<td>Working While Coffee Brewing</td>
<td>R, Y</td>
</tr>
<tr>
<td>011</td>
<td>Prepare Coffee</td>
<td>R</td>
</tr>
<tr>
<td>100</td>
<td>TAKE COFFEE BREAK</td>
<td>G</td>
</tr>
<tr>
<td>101</td>
<td>*** INVALID STATE ***</td>
<td>***</td>
</tr>
<tr>
<td>110</td>
<td>Brew Coffee</td>
<td>Y</td>
</tr>
<tr>
<td>101</td>
<td>*** INVALID STATE ***</td>
<td>***</td>
</tr>
</tbody>
</table>

**Table 4** State Assignment Table*. *State assignments numbers for each state of the Coffee Break example.*
Figure 37  Coffee Break. Algorithmic State Machine, showing the state transition of the Coffee Break circuit.
The next step of the circuit design requires a description of the binary state transitions.

In this design, a Moore state machine is used with the output being a function of the current state only. The state transition table for the circuit is displayed in Table 5.

<table>
<thead>
<tr>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 D1 D2</td>
<td>B</td>
<td>D0+ D1+ D2+</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
<td>0 1 1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
<td>1 1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1</td>
<td>0 1 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

Table 5  State Transition Table. Table of current to next state transitions for Coffee Break circuit.
Karnaugh maps are then used to determine the minimal number of logic gates required to implement the “Coffee Break” state machine. The resulting gate level logic for the next state equations and the output LED logic are shown in Figure 38. The choice of NAND gates is based upon convenience and the desire to utilize a mixed logic notation in the design. The remaining design tasks include implementing a feedback latch for next state increments and providing additional testability through the boundary scan test bus.

Recall that the purpose of adding a boundary scan component is to increase the circuit’s testability. To appreciate gains in testability, it is necessary to quantify the current testability of the circuit. QuickFault, a Mentor Graphics application, is used to generate “testability numbers” to assist with design choices. Quick Fault simulates single stuck-at-faults while examining their effects upon the circuit. If faults can be controlled and observed, then they are considered detected. On the other hand, the faults which cannot be controlled or observed are labeled undetected. QuickFault, a deterministic, concurrent fault simulator, works by comparing single stuck-at-fault results with non-faulted results. If there are no differences between the two results then the fault can not be detected, but if there are differences then the fault is detectable. QuickFault also allows Primary Outputs to be interactively added to the simulation. With these simulated “test points”, the effects of actual test points on fault coverage can be investigated. As in other Mentor Graphics’ applications, probes can be added to the simulation circuit.

Let’s now look at the testability circuit of Figure 38 as it stands. The circuit is simulated in QuickFault using the input sequence of Appendix B.2.2 (coffee.fault). The results of the simulation are represented as a percentage of total faults found and a breakdown of faults found per test cycle, displayed in Figure 39 and Figure 40. Figure 39 shows that for the given input vectors, the current fault coverage is 100%. Figure 40 provides information on individual test vectors. In particular, three of the test cycles do not
provide additional information (cycles number 10, 12 and 14) while one state that does provide information (cycle number 16), is unobtainable. Test number 16 represents state "111" which is not contained in the normal state machine operation (Figure 37). Hence, 100% fault coverage can not be obtained during normal circuit operation.

Driven by a desire for increased fault coverage, additional test logic is included. This additional logic is controlled from the boundary scan testability bus. Since the current design already calls for an edge triggered latch, the ideal choice would consists of a boundary scannable edge triggered latch. Based upon availability, a level sensitive latch is chosen instead. This latch allows data to be shifted in through the test bus and applied to the circuit. With this latch, a state bit pattern (i.e., "111") can be shifted into the state machine's feedback path to increase fault coverage. This will be addressed in section 3.2.4 with the EXTEST instruction.

Figure 41 graphically displays the final hardware configuration of the circuit logic, boundary scan latch and edge triggered flip-flop. Data flows from the edge triggered latch (273) to the boundary scan latch (8373), which is transparent during normal modes of operation. In fact, the clock of the 8373 is hardwired to VCC (Figure 43) In normal mode, this causes data to pass directly through the latch, delayed only by approximately 10 ns of propagation time. The 8373 boundary scan latch is utilized, only during testing of the state machine. During test mode of operation, the boundary scan latch is first loaded with a new state which is then applied to the DUT.
Figure 38 Gate Level Connections. *NAND gate connections used to create the Coffee Break state machine.*
Figure 39  Percentage of Total Faults. *Cumulative percentage of total faults discovered for each cycle test.*

Figure 40  Faults per Cycle. *Number of faults discovered for each test cycle.*
The hardware for the circuit is constructed with two possible modes of operation: manual and automatic. While in automatic mode (jumper set in the lower position of Figure 42), the clocking of the state machine and test IC is performed via signals from the LV500. The manual mode (jumper set in upper position) was originally intended for debugging the state machine but it can be useful when stepping through the state diagram.

While in manual mode, the two Single Pole Double Throw (SPDT) switches can be used to control the CLK signal (WHITE switch) and B (boss is watching) signal (RED switch). The CLK switch returns to its off position when released. Figure 42 displays a detail schematic of the switching circuitry. Mechanical switches have a tendency to bounce when they are triggered. This bouncing results in the switch output oscillating between one rail and "no connection" for a few micro seconds. This is not hazardous for the B switch which only has to be stable at the clocking of the next state, but the bouncing has detrimental effects on the CLK signal which controls the edge triggered latch. The simple debounce circuit of two inverters tied together in series is used to minimize the bouncing effects. The feedback path holds the output constant as long as the input stays below switching values. Fortunately, most good switches do not bounce rail to rail.

This concludes the design of the state machine. The finished wiring schematic is displayed on Figure 43. The parts placement is in the same form as the wiring schematic. This schematic is used to debug the tutorial if necessary, or to answer any wiring questions that you may have. Now it is time to proceed through the LV500 menus and begin exercising this example.
Figure 41  Control and Test Latches. At each clock transition, a new state is loaded into the '273 edge triggered latch.

Figure 42  Detail Switch Schematic. Debouncing switch and jumper connections set for automatic mode.
Figure 43  Coffee Break—Wiring Diagram. Circuit schematic and parts placement for Coffee Break example.
3.2.3 BSCAN\_II Test Setup

As in the first example, the breadboard testhead must be installed on the LV500 prior to the power up of the LV500. The BSCAN\_II tutorial board should now be connected to the breadboard testhead via its eight bit bus. Figure 43 shows the layout of the wire wrapped board. The 8 bit bus is currently wired such that the CLK (bus pin 4) signal wire is colored red. Thus, the lowest right hand corner of the bus socket and the lower right hand corner of the breadboard's bus socket should be connected to the CLK signal's wire. After connecting the bus and powering up the system, bring up the Save/Restore menu of Figure 44. The current working environment will now be loaded with the configuration BSCAN\_II software.

![Image of LV500 Menu](image_url)

**Figure 44** Save/Restore Menu. LV500 Menu displayed when saving or restoring test data for the second boundary scan example.
Set the **Operation** field of the **Save/Restore** menu to "Restore Setup" and type the name "BSCAN_II" in the **File** field. By pressing the F8:EXECUTE OPERATION key, the BSCAN_II configuration will be loaded into the working environment. Now proceed to investigate the three test requirements: DUT Wiring, LV500 system Setup, a list of I/O test vectors for this state machine example.

### 3.2.3.1 DUT Wiring

The LV500 must be mapped to the DUT prior to applying test vectors. In order to achieve this effort, the **DUT Wiring** menu is required. Table 6 provides a cross reference of the values that should be present on the menu while Figure 45 displays the device placement on the breadboard testhead. If the mapping of the Logical Signal Names to the Sector & Channel of the LV500 testhead is not correct, the device will fail the **Pattern** menu tests. The table below lists the signals as they should appear on the **DUT Wiring** menu. If they are in error, then rename them appropriately. The VCC and GND signals must be hand wired, but the other signals can be directly connected. If you have any questions as to the wiring of the DUT, you may use an ohm-meter to perform a continuity check.
Table 6  DUT Wiring Table. Table showing the mapping of the Logical Signal name with the appropriate LV500 channel.

<table>
<thead>
<tr>
<th>Logical</th>
<th>Logical</th>
<th>Sector/Chan</th>
<th>DUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TDI</td>
<td>1.8</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>TMS</td>
<td>1.a</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>1.c</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>*CLR</td>
<td>1.e</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>CLK</td>
<td>2.1</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>VCC</td>
<td>1.5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>TCK</td>
<td>1.b</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>TDI</td>
<td>1.9</td>
<td>8</td>
</tr>
</tbody>
</table>

Once the board is connected to the testhead you should take a moment to become familiar with the layout. There are two manual switches and two jumpers. The jumpers may be used to switch the tutorial from manual to automatic mode as the switch schematic of Figure 42 illustrates. The jumper on the left controls the *CLR signal and the right hand side jumper controls the CLK signal. When both jumpers connect the middle and top wires then the tutorial is functioning in the manual mode. The manual mode ties the *CLR signal high and directly controls the state machine's latch through the CLK SPDT switch. This mode can be used to verify that the hardware is functioning properly or to observe the operation of the state machine. When the bottom and middle wires are connected, the system is functioning in the automatic mode. In this mode, the *CLR signal and CLK signal are taken from the test bus. In automatic mode, the manual
CLK switch has no effect and the state machine may only be run from the LV500's **Pattern** menu. In either mode the B SPDT switch has the same effect. The switches can be differentiated by their position and respective colors with the white CLK on the left and red B switch on the right hand side.

**Figure 45** Breadboard Wiring Diagram. *Pin-to-testhead wiring diagram for the first and second tutorial circuit.*
3.2.3.2 LV500 Setup

Once the DUT wiring is complete, the **Channel** menu may be invoked. Remember that this menu specifies the signal groupings. Since only five controlling and one observing signal is used, there are no separate groups specified at this time. You may use this menu to double check the DUT's wiring. Take a moment to review the channels.

Close the **Channel** menu at this point and open the **Template** menu. This menu is used to define the data format and timing relationships for all channel groups. Figure 46 shows the "bscan_stndrd" **Template** menu used in this tutorial. Two of the testing requirements are now complete (DUT wiring and LV500 setup). You are ready to finish the third set of proper test vectors. Exit the **Template** menu and begin the next section.

---

**Figure 46** Template Menu for "bscan_stndrd." *The standard boundary scan template for the Coffee Break tutorial circuit.*
3.2.3.3 Pattern Test Vectors

Invoke the Pattern menu of Figure 47. As in the BSCAN_I circuit, the test vectors are interactively typed in and saved as part of the configuration. This menu allows you to set up the Force and Compare signals for the BSCAN_II circuit. Run a single test to see if the DUT is properly wired by pressing the Fl:START key. Debug the circuit as necessary. When the test runs without failing, proceed to the next section where the state machine will be explored in detail.

---

**Figure 47** Pattern Menu. *Display of this example's Pattern Menu.*

This allows test vectors to be written and stored for later use.
3.2.4 Running Test

In this section, the state machine will be stepped through while in manual mode. To enter this mode, both jumpers should be positioned in their top locations (i.e., connecting the upper two locations of each jumper). While in manual mode, the DUT must receive power from the testhead, therefore the pattern test must be running. The easiest way to access the DUT in manual mode is by running the LV500 in single step mode. Although no steps will be taken, this will provide power to the DUT. Appendix A contains the timing diagrams (Figures 82, 83 and 84) and timing table (Table 12) for this tutorial circuit. These diagrams will be helpful in working through the examples.

Now use the F8:ADD LINE key to insert a line into the Pattern menu and change the new line into the “BEGIN_SS” command. Figure 48 displays the Pattern menu as it should now appear. It is not necessary to enter an “END_SS” command as only one step will be taken. Once the “BEGIN_SS” command is included in the pattern, enter manual mode by pressing F1:START. Verify the operation of the DUT by stepping through its states. The manual CLK and B switches provide the two required input signals. The top row of red LEDs specifies the current state of the state machine. The diagonal row of colored LEDs on the right hand side displays the current output of Figure 37's state machine. When you are satisfied with the results, press F1:STOP to exit the test. Use F7:DELETE LINE to delete the “BEGIN_SS” line that you previously entered. The DUT can then be changed into automatic mode by moving the jumpers to their lower positions (connecting the lowest two locations of each jumper).
One of the most difficult tests to perform on a device is of sequential logic. The challenge stems from logic's dependence upon previous inputs which may not be known. By using boundary scan devices, one is able to inject a known starting state into the device, allowing the observation of any state sequence. This procedure is similar to including a preset signal, but it is more powerful as any state can be scanned in (not just a single preset state). Using the '8373 IC the state machine will now be verified. This is done by loading in a desired state and then single stepping through its operation.

The **Pattern** menu for this example is similar to the last one explored. If you are comfortable with the LV500's operation, Figure 50 suggests a useful position to start the single stepping. If you wish for a review of the state machine's operation of begin single stepping with the placement shown in Figure 48.

---

**Figure 48**  **Pattern** Menu for Manual Mode Stepping. *LV500 menu used to manually step through the Coffee Break state machine.*
The POWER_UP and RESET macros are stepped through to initialize the IC and place the TAP in the Test-Logic-Reset state.

The TAP is then stepped into the Shift-IR state where the EXTEST instruction (00000000) is loaded into the IR. This instruction, when executed, will set the device in test mode and perform a sample of the device outputs. The instruction register of Figure 28 shows the order of loading the instruction into the IR. Recall that the shifting is delayed by one clock cycle such that the MSB is shifted in at the rising edge of TCK 7 and the LSB is shifted in at the rising edge of TCK 14. Hence, the last bit is shifted in while entering the Exit1-IR state.

TMS is driven high, forcing the transition to this state. The last bit of the instruction is shifted in as the TAP advances from the Shift-IR state (TCK 13) to the Exit1-IR state (TCK 14).

At Update-IR the device enters test mode, this means that the device pins are in high impedance state and the latch is no longer transparent. When the circuit is transparent it is in the normal mode of operation.

Select-DR-Scan state.

Capture-DR state, TDO is driven low.

The next state of interest is Shift-DR where the previous device inputs and outputs are shifted out of the sample register, shown in Figure 30. The device is in the high impedance state of test mode. Thus, all of the Q-output signals are set to logic 0, the floating D inputs are set to logic 1 and the D inputs tied to the circuit logic which drives them to a logic 0. Referring to the coffee-break schematic of Figure 43, note that the
remaining two signals *OC & C are tied to GND & VCC respectively.

This results in the shifting out of the bit pattern: “000000001111100001”.

Let’s examine this step in detail. To shift in the next state, all of the bits are set to logic 0 except for C and the three Q bits representing the current state. C is set to logic 1 so that data may be clocked through the ‘8373 and the three Q bits are set to the next desired state of the machine (010). When these three bits (Q1, Q2, Q3) are applied to the ‘8373 IC, they will become the new state that is then fed through the DUT’s logic. With these three signals, the state machine can be set to a known state. The only other input bits of interest are C and *OC which should be set to logic 1 and logic 0, respectively. Figure 49 displays the state bits set to the valid state (010) “Work while the coffee brews.” Why send the state through the Q outputs as opposed to the D inputs? This is because the ‘273’s Q outputs are tied directly to the ‘8372’s D inputs thus, any data set on the D inputs would be overwritten. Because of this, the desired state is set by providing the state machine with a current state stimulus from the ‘8373’s outputs. This in turn becomes the next state, which will be clocked into the ‘273. Once clocked in, the state machine has been successfully set from the 1149.1 testability bus. Figure 51 helps to visualize the bit stream (including the next state) that is shifted into the ‘8372.
**Figure 49** Location of Device State Bits. *Pattern Menu* displaying the location of the three Coffee Break state machine bits.

**Figure 50** '8373 Latching Output in Test Mode. *The section of code where the output is latched for viewing.*
Figure 51  Next State Bit Stream Shifting into the '8373. The next state is set by shifting stream into the '8373 then propagating it through state machine logic.

TCK 36  Exit1-DR is entered next. This is where the last bit of the sample register is shifted in. Note that the LEDs output is still “000” since the new state hasn’t been latched out of the ‘8373 yet.
TCK 37 Update-DR latches the '8373's pins with the vector that was shifted in during the Shift-DR state. You will see this by looking at the LEDs which now display the current state. Note that while in test mode the '8373 is no longer transparent. Figure 50 shows a useful position for the placement of the "BEGIN_SS" step command, if it has not yet been inserted. At this clock cycle, TDO is in its high impedance state.

TCK 38 The Select-DR-Scan latches the next state into the '273 IC. The '273 is the only IC that is controlled by the CLK. The '273 controls the clocking of the state transitions whereas the boundary scan IC aids only in the testing of this circuit. Even though the output of the '8373 was available during the last state (Update-DR), the propagation delay through the circuit requires the latching of the data to wait until this clock cycle. At this time, the '8737 is still in test mode therefore the '273 can not yet release its data. The release of data will wait until the '8373 is again transparent.

TCK 39 Select-IR-Scan state.

TCK 40 Capture-IR state. The TDO signal is forced low.

TCK 41-48 Shift-IR is where the BYPASS (11111111) instruction is loaded into the IR. This instruction allows the '8373 to return to its normal mode of operation. Before the next state is reached, set the B switch to the logic 0 position so that a visible state transition (to 100 "TAKE COFFEE BREAK") may take place.

TCK 49 Exit-IR state.

TCK 50 At Update-IR the BYPASS instruction is issued and the '8373 is again transparent. The next state of the Coffee break is sent through the '8373
and the LEDs show the transition to the next state. The next state will be “100” if the B switch was logic 0 and “010” otherwise.

TCK 51  Select-DR-Scan state.
TCK 52  Select-IR-Scan state.
TCK 53  The state machine is then taken to Test-Logic-Reset where it remains for the duration of this example.
TCK 54-63 At this point no more test data is required. You may finish this menu by altering the B switch according to the state you wish to reach. Each step is taken by sending a clock pulse signal, CLK (viz., pressing the F4:SINGLE STEP key). Figure 52 illustrates how the Pattern menu should appear at this time. This concludes the verification of the state machine. You may wish to alter the input to the state machine and observe the effects on the system.

Before moving on to the next section, let us review the basic boundary scan cell (BSC) of Figure 53. It is this logic block that is responsible for the separation of normal and test modes of operation. It also controls data flow through the boundary scan IC. Specifically, the output MUX controls the device mode of operation through the “MODE_CONTROL” signal. When the device is in test mode, data is channeled through the two latches (ClockDR & UpdateDR). This was performed in the previous example, when the '8373 was loaded with the EXTEST instruction and the “MODE_CONTROL” was set to logic 1.

The next state of interest is the Capture-DR state. At this state the “CLOCK_DR” signal clocks the input of each BSC into the ClockDR latch. The Shift-DR state follows. It changes the “SHIFT_DR” signal to logic 1 so that the input to the input MUX now
comes from TDI (test scan input) and not IN (normal input). This also allows the captured BSR to be serially shifted from TDI to TDO. As the “SHIFT_DR” signal is driven to logic 1, the “CAPTURE_DR” signal is clocked, allowing the data to enter each BSC at TDI and exit through TDO. When the old BSR is shifted out and the new one is shifted in, the Update-DR state sends the “UPDATE_DR” signal. This signal makes the new BSR data available to the output. The only remaining task is to step the TAP state machine back into its normal mode of operation. When this is done, the “MODE_CONTROL” signal again goes to logic 0 thus returning the '8373 into its transparent mode of operation. Feel free to modify the Pattern menu since it can be reloaded at any time. Appendix B.3 contains the LV500 generated code for each of these examples. When you are finished, ensure that the test passes and then invoke the Schmoo menu.

Figure 52 Conclusion of BSCAN-II Pattern. After this section the program is finished, but may be run again.
Figure 53  Boundary Scan Cell. *Logical view of the basic boundary scan building blocks.*

3.2.5  Schmoo Specification Tests

In this last section the Schmoo tests will be executed to verify the setup and propagation timing requirements. This is done with single Schmoo sequence.

3.2.5.1  TCK Pulse Parameters

A valid TCK pulse is critical for the execution of boundary scan features. Three parameters that should be validated include the pulse width, the time that valid data is required before the rising edge of TCK and the time that data remains valid after the falling edge of TCK.
Since there are several timing requirements for '8373, the worst case scenario will be examined for each of the parameters previously described. From the '8373's specifications (Appendix E p 15, 16), the maximum values are chosen:

\[ t_{SU} \text{ (TDI before TCK rising edge)} = 6.0 \text{ ns} \]
\[ t_{W_{min}} \text{ (Minimum TCK pulse width)} = 25 \text{ ns} \]
\[ t_{PZL} \text{ (Q after TCK falling edge)} = 22.9 \text{ ns} \]

To test the time limits, a range of valid and invalid numbers must be entered:

\begin{align*}
0.0 \text{ ns} & \leq t_{SU} \text{ (TDI before TCK rising edge)} \leq 100.0 \text{ ns} \\
0.0 \text{ ns} & \leq t_{W_{min}} \text{ (Minimum TCK pulse width)} \leq 200.0 \text{ ns} \\
0.0 \text{ ns} & \leq t_{PZL} \text{ (Q after TCK falling edge)} \leq 200.0 \text{ ns}
\end{align*}

Figure 54 displays a timing diagram used to measure these three values. To verify the numbers sweep the Phase Delay and Phase Width of TCK through Phase B (0B).

Figure 55 shows how the sweeping of Phase Delay and Phase Width times effects TCK’s starting and ending pulse times. Using the Def Schmoo menu to enter the appropriate ranges, the chosen input for each parameter:

**Variable X: Phase Width**

- **Template:** bscan_std
- **Clock:** 0B
- **From:** 60.0 ns
- **To:** 200.0 ns
- **By:** 10.0 ns

**Variable Y: Phase Delay**

- **Template:** bscan_std
- **Clock:** 0B
- **From:** 9.0 ns
- **To:** 92.0 ns
- **By:** 10.0 ns
**Figure 54** Definition of $t_{\text{Wmin}}$, $t_{\text{SU}}$ and $t_{\text{PZL}}$ for the Circuit. By sweeping the phase and delay of $0B$, these values are verified.

**Figure 55** Sweeping the **Phase Width** and **Phase Delay** of $0B$. *Note the range of pass, fail and untestable values.*
After entering the above values, invoke the Schmoo menu to execute the test and display the results in real time. Press F1:START and note how the tests are performed sequentially, with each one representing a particular value pair of the desired voltage ranges. The final results should be similar to Figure 56 and can be divided into three distinct section: pass, fail and untestable. The upper right diagonal section (area with hash marks) of the Figure 56 is untestable because the Phase Delay plus Phase Width time is greater than 200 ns. Recall from Figure 55 that this is where the falling edge of TCK lies beyond the 200 ns cycle time constraint.

The remaining two sections (pass and fail) are used to verify the three timing specifications. The first specification (\( t_{SU} = 6.0 \text{ ns} \)) is derived from the bottom two rows of data (Phase Delay = 2.0 ns through 12.0 ns). This shows that a setup time of 2.0 ns is too small but that a setup time of 12.0 ns is acceptable. By increasing the resolution of the test (Figure 57) the setup time can be more precisely determined. As Figure 57 shows, the setup time required varies between 4.0 ns and 8.0 ns with an average of 6.0 ns. The second specification (\( t_{Wmin} = 25 \text{ ns} \)) is represented by the first column of data (Phase Width of 20 ns). As Figure 56 shows, 20 ns is still a valid Phase Width and the circuit works even better than specified.

Finally, the diagonal section of failures is based upon the third (\( t_{PZL} = 22.9 \text{ ns} \)) propagation delay examined. One important constraint to note in this analysis is the 150 ns sample time of output data. Hence, results that propagate through the '8373 (i.e., data that experiences \( t_{PZL} \)) must be valid by this 150.0 ns sampling time. Note, if the 22.9 ns propagation time is subtracted from the 150.0 ns sampling time, the result is 122.1 ns (valid signal length). In both Figures 56 and 57 the data results are valid when the Phase Width and Phase Delay sum to a time less than 122.0 ns. When the time is greater than
122.0 ns, failures occur because the high impedance signals do not have adequate time to be driven low.

**Figure 56**  TCK Pulse Limits. *Examine the TCK rising edge, pulse width and falling edge timing constraints for the '8373.*
Figure 57  Higher Resolution TCK Pulse Limits. Using an increased resolution to further examine the TCK rising edge, pulse width and falling edge timing constraints for the '8373.
3.3 Boundary Scan Circuit Design

This third example shows how a boundary scan compatible component is built. Rather than including a boundary scan component, as the previous design, this design consists of a single boundary scan component which encapsulates both the coffee break state machine and the 1149.1 standard. This component may be in the form of an ASIC, MCM or even a MSI. The design is at the logic level only and will contain both normal and test features.

3.3.1 Logical Design at the IC Level

Before starting the design, it is prudent to specify why a typical design may be performed at this level. The reason is usually broken into two requirements—availability and cost. Due to the young age of the 1149.1 standard, there is a limited variety of components from which to choose. If there is not a suitable component for the design, you will have to create one. Even if such a component exists, it may be cost prohibitive to include it in the design. Regardless of the reason, there will be times when a design engineer is required to fill in the gaps for 1149.1 compliance.

Due to the fact that the design now includes both testability and normal circuit features, the initial work load will increase. Although there is this increase, the overall work load may be the same or even decrease due to the benefits of designing a boundary scan component. One benefit is that a design can be tailored with appropriate test features best suited to the DUT. Minimizing hardware duplication may also be achieved by combining common blocks. For example, the duality (test and normal mode of operation) of the feedback registers is accomplished with input and output multiplexers used to direct the data path. This duality is explored in the design of the Custom Scan
Register (CSR). The normal test logic of this design is identical to the previous example, so the rest of this section will examine the design of the testability features.

### 3.3.2 Boundary Scan Architecture Features

The final design is to be IEEE 1149.1 compatible, but which features of the Test Access Port and Boundary Scan Architecture Standard should be included in the design? The 1149.1 document answers this question by detailing both required and suggested design specifications. This IC design includes all of the 1149.1 features required for compliance in addition to optional hardware which satisfies the need for increased circuit testability. The required hardware includes the TAP, IR and two data registers—the BSR and the Scan Bypass Register. Optional hardware is also accessed through the DR and is included when it can substantially increase a design's testability. Common optional features include a BIST, PRNG, PSA, device identification number and silicon test points. Silicon test points are the design's only optional feature used in this design.

The TAP controls the selection of the IR and DR scan paths. This is done during the Select-RI-Scan and Select-DR-Scan operations of the TAP state machine (Figure 9). There are several possible DR paths, only one of which is chosen at a time. It is the IR's address which selects a specific DR scan path. Once selected, the required and optional registers control the flow of serial data from TDI through TDO. The IR's addressable instructions, including DR selection, are presented in more detail in a later section and on Table 8. Each scan path originates at TDI, ends at TDO, and is clocked by TCK. The specifics of this design, including all of the standard 1149.1 features and optional features will be examined next.
3.3.3 The Design Overview

Because the circuit in this example is essentially the encapsulation of the previous circuit (Figure 41), the controlling logic of the state machine is the same and the coffee break feedback latches are still required. The primary difference between the two designs is the relocation of the feedback latches as shown in Figure 58. This results in various problems, including the inability to directly control and observe the latches from the primary (external) inputs. With this constraint, the circuit's current testability must again be quantified in order to identify changes in testability. Using QuickFault, the design's testability is examined before and after circuit modifications.

![State Machine Logic for the Second Example](image)

**Figure 58** State Machine Logic for the Second Example. *This design is essentially the encapsulation of the previous circuit.*

Figure 59 shows the gate level design of the normal circuit. Also shown is the initial testability evaluation of undetected single stuck-at-faults and fault blockages. Fault blockages are shown where bottle necks occur in the fault path. The stuck-at-faults are represented by solid boxes above the pin for SA1 faults and below the pin for SA0. Figure 60 shows the percentage of total faults. Notice that the loss of the latch inputs has
lowered the overall circuit testability. In the previous example, the latches were accessible and any logic state could be observed and controlled through the boundary scannable latch ('8373). With the loss of input pin visibility, only 85.9% of the circuit can now be tested (i.e., controlled and observed). In the previous example it was discovered that several states were required for complete fault coverage, but one particular state (111) was not available through the normal operation of the state machine. As in the last example, this circuit's implementation will have to be able to directly control the input state in order to achieve full fault coverage. One implementation would allow test point access directly to each of the state machine latches. In most designs this is undesirable, as it takes up pin real estate. Because this design includes the boundary scan test bus, it can be utilize to access each of the three state machine latches without taking up additional pin real estate.
Figure 59  Gate Level Connections. NAND gate connections used to implement the Coffee Break state machine.
The final design, similar to Figure 62, includes a multiplexer that directs the inputs to the state machine latches. During the normal mode of operation, input is directed from the combinational logic's feedback path. In test mode of operation, the input to the latches is directed from an internal boundary scannable register.

Figure 62 also displays the undetected single stuck-at-faults and fault blockages. The new cumulative testability of the circuit is shown in Figure 61. Note that it is still less than 100%. The reason for this is that, although is possible to set (i.e., control) the latches to any state, the output is still limited by the three LEDs. To further increase the testability, observability points must also be added to the design. Fortunately, QuickFault provides a means to interactively add Primary Output points to the design simulation. This feature is used to select additional silicon test points for the design.

From Figure 62, it is apparent that the abundance of stuck-at-faults occurs at the output of the combinational logic. Using a Primary Output placed at the end of the Q0 next state logic, the fault coverage is increased to 92.1%. Figure 63 shows the faults remaining after inserting the new observability points. Similarly, Figures 64 and 65 show the faults remaining after placing a single primary output following the Q1 and Q2 next state logic. These two points provide a fault coverage of 92.1% and 88.5% respectively. Notice that the placement of the Q1 primary output point also covers the faults of Q2, hence, only the first two new observation points are required for full test coverage. Using both of these points (Q0 and Q1) results in a testability of 97.8% as shown in Figure 66. Because three of the faults are due to grounding unused input pins, they appear as single stuck-at-faults. These three faults are limited by the simulation and should not be considered since they will not impact the logic of the design or its final implementation. Thus, the testability is actually 100%. To simplify the design of the
testability logic, all three test points (Q0, Q1 and Q2) are included. This allows the controlling and observing test points to share the same three bit register length.

![Graph](image)

**Figure 60** Percentage of Total Faults. *Cumulative percentage of total faults is 85.9% for the new circuit.*

![Graph](image)

**Figure 61** Percentage of Total Faults With Feedback Multiplexer. *Cumulative percentage remains at a low 87.8% with feedback multiplexer installed.*
Figure 62  Gate Level Connections for Mode Select Multiplexers.

*Multiplexer used to select between normal and test modes of operation.*
Figure 63  Fault Coverage of Q0 Observation Point. Fault coverage increases to 92.1% with additional output at Q0.
Figure 64  Fault Coverage of Q1 Observation Point. *Fault coverage increases to 92.1% with additional output at Q1.*
Figure 65  Fault Coverage of Q2 Observation Point. Fault coverage increases to 88.5% with additional output at Q2.
Figure 66  Fault Coverage of Q0 and Q1 Observation Point. *Full fault coverage with additional outputs at Q0 and Q1*
Test points are desired but how are they implemented using the testability bus? The 1149.1 standard specifies only a test bus protocol and, in itself, will not provide the desired access. Fortunately, the 1149.1 standard's flexibility allows optional DRs into the scan path. One such structure is the BIST, which is too comprehensive for this circuit. In this design, a serial DR will be used to access the "silicon" test points. The register will allow controlling data (three bits) and new data (three bits) to be shifted in and then applied to the internal latches. The register will then capture the results of the next state, which can be shifted out.

Combining the normal and test circuit results in the composite logic diagram shown in Figure 67. For the normal circuit logic, note how the latches are encapsulated in the design and only the two inputs (CLK and B) and three outputs (RED, YELLOW and GREEN) are accessible to external test probes. In order to satisfy the IEEE 1149.1 testability bus requirements, this design also includes the TAP, IR, BSR and Bypass register. To control this new logic, the additional testability input (TMS, TCK and TDI) and output (TDO) lines are also included. This is the logical combination of the coffee break state machine (Figure 37) and the testability architecture (Figure 8). The details of each designed block are explored in the next section.
Figure 67 High Level Schematic for Circuit. *Composite of test and normal mode circuits.*
3.3.4 The Design Details

This section explores each component of a boundary scan system design. The result of these components is a design which performs the normal and test functions described above, all accessible throughout the four bit 1149.1 test bus.

3.3.4.1 Design of the TAP

The TAP is a synchronous finite state machine that responds to TMS and TCK to step through its sixteen states as shown in Figure 9. The TAP allows the implementation of the test access protocol specified in the 1149.1 standard. Its main purpose is the selection of either the instruction or data register for shifting of data from TDI to TDO. A typical scenario would include:

1. Select the IR into the scan path.
2. Shift the desired DR address into the IR.
3. Select the DR into the scan path.
4. Shift the serial information in from TDI.
5. Observe the output generated at TDO.

Due to the complexity of the TAP's sixteen states, state minimization tools (peg, eqntott and espresso) are used to generate the logic for implementation. These software tools are available through RIT's ultb mainframe and they have proven to be very effective in minimizing the logic.

From Figure 9 alone, one can design the logic required to control the TAP's state machine. What controlling signals does the TAP generate to control the other registers? In order to select and shift data through either the IR or DR there are three required signals: “CLOCK,” “SHIFT” and “UPDATE.” The signals are used respectively to
clock data into the registers, shift data through the register's serial scan path and finally clock the data to the outside world (i.e., update the register). Table 7 lists the signals required for this operation. Note the breakdown of separate signals for both the IR and the DR. Also, note the unique "RESET," "ENABLE" and "SELECT" signals, used to reset, enable and select an appropriate DR or IR. Figure 68 shows how these controlling signals are presented to the IR and DR. From Table 7 and the TAP state machine the high level language description of the sequence of commands is generated. This is shown in Appendix B.1.1 (Peg Input Commands). This language description is in a form suitable for the PLA Equation Generator (peg) program available on ultb. The output of peg is a Moore model finite state machine representing the TAP. This output is in the eqn format, shown in Appendix B.1.2, and is suitable for input into the eqn to truth table (eqntott) program. For reference, Appendix B.1.3 and B.1.4 contain the summary and truth table output of the eqntott program.
<table>
<thead>
<tr>
<th>STATE</th>
<th>Clock</th>
<th>Update</th>
<th>Shift</th>
<th>Clock</th>
<th>Update</th>
<th>Shift</th>
<th>Reset</th>
<th>Select</th>
<th>Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IR</td>
<td>IR</td>
<td>IR</td>
<td>DR</td>
<td>DR</td>
<td>DR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test-Logic-Reset</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Run-Test/Idle</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Scan-DR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Scan-IR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Capture-DR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Shift-DR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Exit1-DR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Pause-DR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Exit2-DR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Update-DR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Capture-IR</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Shift-IR</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Exit1-IR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Pause-IR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Exit2-IR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Update-IR</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 7** TAP Output Signals. *Mapping of signals produced by the TAP and the state in which they are produced.*
Figure 68  TAP Output Signals. *Representation of controlling output signals sent to the IR and DR and flow of data from TDI to TDO.*

The *eqntott* program generates a truth table of minterms based upon the *peg* boolean input equations. The output of *eqntott*, shown in Appendix B.1.5 (Eqntott Output File), is used as input to the minimization program *espresso*. *Espresso* produces (no pun intended!) a minimal equivalent representation of the coffee break state machine using heuristic boolean minimization. The final output of *espresso* is in the standard sum-of-products format and the minimization process is complete. This output is listed in Appendix B.1.6 (Espresso Output File). As the output shows, the number of minterms has been reduced from 34 to 25. The final state machine has 13 outputs (4 for state transitions latches) and 5 inputs. Their order is the same as specified in Appendix B.1.2. Figure 69 shows the final logic of the minterms, designed with combinational logic and
four feedback latches. The input files required to run and the output files generated by each of these programs is found in Appendix B.1.

There are two input signals (TCK and TMS) used to clock the state machine and select the next state. There are also nine controlling output signals. Three of these signals are used to control the IR (CLOCK_IR, UPDATE_IR and SHIFT_IR) and three are used to control the DR (CLOCK_DR, UPDATE_DR and SHIFT_DR). The three remaining (RESET, SELECT and ENABLE) signals are common to the IR and DR. RESET, resets both the DR and IR to an initial state. SELECT controls the output MUX. If SELECT is a logic 1 then the output of this MUX is the IR; if it is logic 0 then the output of this MUX is the DR. The final signal, ENABLE, enables the high-impedance state of the TDO signal. As the logic shows, although TDI and TDO are not passed through the TAP, they are directly affected by the TAP's operation. The four remaining output signals represent the next state of the TAP. These are feedback into the clocked latch. Figure 70 shows the QuickSim simulation of the TAP operation (using code of Appendix B.2.1). Specifically, the figure shows how the TAP is stepped through its state machine and how it drives the control lines, depending upon the current state.

This concludes the design of the TAP subsystem. In summary, the TAP controls the selection of a IR or DR serial scan path. Once a scan path is selected, the TAP controls the shifting of data from TDI through the scan path and out through TDO. The system registers will be investigated next, beginning with the IR.
Figure 69 TAP Schematic. Logic of TAP presented as the sum of product terms.
3.3.4.2 Design of the Instruction Register

The instruction register consists of a series of shift register bits arranged between TDI and TDO. The TAP is used to select either instructions or data to be shifted into the scan path, but it is the IR that addresses a unique DR scan path. In general the length of the IR is chosen to be eight bits for convenience, but it can be of any size. The only limitation of the IR's size is that it must contain enough bits to uniquely identify each testability instruction included in the design. Table 8 shows the mapping of IR address bits to the desired instruction. Since this design has five different instructions, it has an IR length of three bits.
<table>
<thead>
<tr>
<th>IR Address</th>
<th>Instruction</th>
<th>Data Register Selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>X X 1</td>
<td>BYPASS</td>
<td>bypass register</td>
</tr>
<tr>
<td>0 0 0</td>
<td>SAMPLE</td>
<td>boundary scan register</td>
</tr>
<tr>
<td>0 1 0</td>
<td>EXTEST</td>
<td>boundary scan register</td>
</tr>
<tr>
<td>1 0 0</td>
<td>INTEST</td>
<td>boundary scan register</td>
</tr>
<tr>
<td>1 1 0</td>
<td>CUSTOM</td>
<td>custom scan register</td>
</tr>
</tbody>
</table>

**Table 8**  Mapping of IR to Address Instruction. *Each address of the IR maps to an instruction that is to be run in the data register.*

During normal mode of operation, the IR is selected by the TAP and a default instruction is shifted in during the Capture-IR state. The desired instruction (three bits) is then shifted in through TDI when in the Shift-IR state of the TAP state machine. The new instruction address is then applied to the DRs from a set of shadow latches. These are rewritten during the Update-IR state. Once a valid instruction is selected, the state machine can either step into the Run-Test/Idle state if returning to the normal operation of the system, or the state machine can step to a specific DR. From the previous TAP design, it is known that at least three controlling (CLOCK_IR, SHIFT_IR and UPDATE_IR) signals are used to control the IR. There are two additional signals required for a complete serial scan path, namely TDI and TDO. With these signals the addressing of various DRs is performed. Figure 71 shows a register level view of the IR, and Figure 72 provides an example of the addressing of the SAMPLE instruction.
Figure 71  IR Register Level Schematic. High level view of the Instruction Register.

Figure 72  IR Selection of SAMPLE DR. The logic required to address a unique DR from the IR.
The logic required to realize the IR, shown in Figure 73, consists of a scan path of three bits that proceeds from TDI to TDO, the three IR address lines and the IR controlling logic. During the initial Capture-IR state, the IR is preloaded with a default instruction. The device identification is typically chosen for the default instruction. Since it is not available, the BYPASS instruction is chosen instead. The default sequence, "X01," preloads the BYPASS instruction address in the IR. The CLOCK_IR signal is used to clock data into each of the register bits, and the SHIFT_IR data is used to serially shift the clocked data through the scan path. After loading in the new instruction, the UPDATE_IR signal, updates the selected address shadow latch. The remaining control signal, RESET, resets all of the registers to the default instruction, "111." Note that the RESET instruction bit sequence also corresponds to the BYPASS instruction. The IR design is now complete. The three DRs, starting with the default bypass register, will be considered next.

Figure 73 IR Control Logic. Low level schematic of the Instruction Register logic.
3.3.4.3 Design of the Bypass Register

The scan bypass register is placed into the DR scan path when the IR loads the BYPASS instruction. The bypass register consists of a single register between TDI and TDO. This function is selected when one desires to shortcut or “bypass” the system scan path. Recall that the system scan path consists of a serial shift register through each of the functional inputs and outputs. For ICs with large pin counts, the bypass register presents a substantial savings. As in the IR, the bypass register is preloaded with a default value for the first shift. The 1149.1 standard specifies this default as a logic 0. As with the IR, the default input is loaded into the register during the Capture-DR state. Following the first bit (0), the input to TDI determines the output at TDO. As the register diagram of Figure 74 displays, the bypass register requires the controlling inputs of TDI, SHIFT_DR, CLOCK_DR, UPDATE_DR and the serial scan path signals of TDI and TDO.

![Figure 74 Bypass Register Control Logic](image_url)

The bypass register is only one of three DRs that are included in this design. The next DR examined shifts data through the periphery of the circuit.
3.3.4.4 Design of the Boundary Scan Register

The BSR consists a scan path of five bits, one for each of the functional inputs and outputs of the device. Hence, the two input (CLK and B) and three output (RED, YELLOW and GREEN) signals make up the BSR. As with the other DRs, the BSR is selected by the IR. Specifically, the SAMPLE, EXTEST and INTEST instructions select the BSR into the DR scan path. As exercised through the LV500 examples, the BSR can store and shift normal or test data through the functional inputs and outputs. The choice of either test or normal data depends upon the current instruction. For example, the SAMPLE instruction captures and stores normal data applied at the device's functional input and outputs. EXTEST applies test data only to the device's function outputs and INTEST applies test data only to the functional inputs of the device. The instructions accomplish this with the aid of the four controlling signals (CLOCK_DR, SHIFT_DR, UPDATE_DR, MODE_SELECT) as shown in the boundary scan cell of Figure 75.

Figure 75 also shows that this register encompasses each of the input and output signals (the vertical paths), and the primary serial scan data path (the horizontal path).

The logical design of the BSR is the same as that presented in Figure 53. MODE_SELECT controls the passing of normal and test data to and from the functional inputs and outputs, while the three clocking signals control the shifting of data through the scan path. For example, when executing the SAMPLE command, a snapshot of the functional inputs and outputs is taken by clocking the CLOCK_DR signal during the Capture-DR state. The data is then shifted out of the sampled registers during the Shift-DR state by setting the SHIFT_DR signal and clocking the CLOCK_DR signal. Figure 76 shows the serial BSR which proceeds from TDI, through the functional inputs and
outputs and then out through TDO. Since the scan path is five bits wide, 5 shift cycles of SHIFT_DR are required to view all of the sampled data. The TAP’s Update-DR state leaves the circuit in the normal mode of operation.

The EXTEST and INTEST commands are executed similar to the SAMPLE command with two primary differences. The first difference is that test data is shifted into the scan path, rather than normal data being shifted out of the scan path. Once shifted into the scan path, the test data is applied through the functional outputs (EXTEST) or through the functional inputs (INTEST) during the Update-DR state. The second difference is that the data is applied by the Update-DR state rather than the Clock-DR state. The Capture-DR state still captures the current status of the device which can be viewed during the Shift-DR state. However, it is the incoming data, shifted in during the Shift-DR state, that is of most importance. It is then during the Update-DR state that the UPDATE_DR signal forces the test data to drive device inputs(INTEST) or outputs (EXTEST). Recall that the EXTEST command was used in the previous LV500 example to load in the desired state bits of the '8373 and apply them to the device under test.

The BSR and bypass register is designed to comply with the 1149.1 standards. The remaining register, CSR, increases the testability of the DUT which also complies with the 1149.1 standard.
Figure 75  BSC Register Level Schematic. The BSR consists of a serial scan serial path through each BSC.

Figure 76  BSR Scan Path. Serial scan path linking each BSC into the BSR.
3.3.4.5 Design of the Custom Scan Register

The CSR is the final DR designed. When selected through the IR, this DR allows the internal state bits to be set. This capability is similar to the EXTEST and INTEST operations, but consists of setting internal rather than external functional pins. To control the shifting and setting of the CSR, the three DR controlling signals CLOCK_DR, SHIFT_DR and UPDATE_DR are again required. These signals clock and shift the data through the registers and then perform the final update of the coffee break state machine bits. Figure 77 shows the register level design of the CSR.

The coffee break state machine requires one bit for each latch; hence, the CSR has a length of three bits. The input to the CSR represent the new, or desired state. The output from the CSR represents the next or future state. Both the inputs and the outputs are clocked during the Shift-DR state, starting with the MSB and ending with the LSB. This allows the observability and controllability of the state machine. Similar to the BSR, the new state takes effect during the Update-DR state. It is triggered by the UPDATE_DR signal, which updates the shadow latch with the new state of the system. The logic required to implement this register is shown in Figure 78. Note that the MODE_SELECT signal is not shown. It directs the input to the coffee break state machine from either the normal feedback path, or the CSR. Because this design is at a low enough level, we can use this feature to minimize redundant logic. The coffee break state machine feedback latches will now have dual modes (normal and test) as controlled by the MODE>Select signal and an additional MUX. When selected, the CSR will be able to control and observe the state machine latches and when not selected the coffee break state machine will remain isolated from the CSR.
The design of each IR and DR has been explored. The next section summarizes the operation of the full design.

![CUST REG Diagram]

**Figure 77**  Custom Scan Register Level Schematic. Details of the input and output signals required for the CSR.

![CSR Logic Diagram]

**Figure 78**  Custom Scan Register Control Logic. Low level schematic of the CSR logic.
3.3.5 Summary of Boundary Scan Circuit Design

The focus of this example has been the creation of the logic required to implement a circuit that is 1149.1 compatible. The final circuit, presented in Figure 79, includes the interfaces between each component and the testability bus. It also includes four components not previously discussed. Two of the new components are used to control the flow of the DR logic (DR Decoder and DR MUX) and the remaining two are used to control TDO's output (DR/IR select MUX and tri-state output enable buffers). The DR Decode logic allows the BSR or CSR DRs to modify data only when they are appropriately selected as the current instruction. Because the Bypass register is not invasive to the circuit, it is not included as part of the decode logic. The DR Decoder decodes the IR's address into a mode select signal for the respective DR. If this logic were not included, the BSR or CSR could corrupt the normal circuit's operation by introducing test data at improper times. Note that the MODE_SELECT signal that is shown as part of the CSR is physically placed in the feedback path of the coffee break state machine. This signals directs the input to the coffee break state machine latches from either the CSR (when the CSR is selected by the IR's address) or the normal feedback path (when the CSR is not selected).

The IR's address controls the DR output to TDO through the DR MUX. This three input multiplexer selects the appropriate DR (BSR, CSR or Bypass register) to implement the current instruction. Through the use of the IR Decode logic and the DR MUX, each of the instructions is applied to the appropriate data register, and then allowed to propagate out through TDO.
The remaining two controlling logic components are common to both the IR and the DR. First, the select MUX selects the serial scan path (IR or DR) as based upon the current state of the TAP's state machine. This 2:1 MUX is at a logic 0 when the DR scan path is selected and a logic 1 when the IR scan path is selected. The remaining component, the output enable buffer, provides the high-impedance state mode for the output signals. These buffers are controlled by the ENABLE signal from the TAP which forces the output into a the high-impedance mode when the TDO signal is inactive.

Bus signals are shown linking the common control and address lines. A serial scan path that begins with TDI and ends with TDO links each of the IR and DRs (BSR, CSR and Bypass) in parallel. The coffee break state machine still requires its two input and three output signals. These are gated through the BSR for input and output signals and through the output enable buffer for output signals only. The TAP is shown driven by TMS and TCK; linking all of the IR's and DR's scan paths together; and controlling the normal and test modes of circuit operation.
**Figure 79** Schematic for Logic of Boundary Scan Component. *High level schematic showing the logic of the boundary scan component.*
4.0 Summary And Conclusion

This thesis presented several examples of how the IEEE 1149.1 testability bus is used to increase a circuit's testability. First, it explored a single boundary scannable IC. This was done while gaining familiarity with the LV500. Next, it showed how to increase a sequential circuit's stuck-at-fault coverage to 100% by including a boundary scan IC in the feedback path. Finally, it detailed a high level design of the logic required to implement a boundary scan compatible component. The final design of this component included all of the 1149.1 standard's required features and one optional feature. The optional feature was used to increase the single stuck-at-fault coverage of the sequential circuit to 100%. As the next section shows, this increased testability was not without cost.

4.1 Design Tradeoffs

As in any design, creating a boundary scannable component requires the consideration of many tradeoffs. Three of the main concerns are: test component area, power consumption and time delays incurred with the additional logic.

Table 9 shows the initial coffee break state machine contains only eighteen logic cells as compared to 107 logic cells required for boundary scan access. This presents a testability overhead of an alarming 84.6% when comparing logic cells or of 89.1% when comparing transistors. Much of the silicon overhead (i.e., the TAP, IR, Bypass register and controlling MUXs) is constant, regardless of the design size or logic. The remaining overhead either varies with the pin count, as the BSR and Output Enable buffer, or is controlled by the designer, as with the optional CSR. Thus, this overhead can be minimized by the designer. Taking this into consideration, if we increase the design size to that of a typical 1,000 transistor circuit, the overhead drops to 52.3%. Further
designing of typical VLSI (10,000 with 20 pins) or ULSI (100,000 with 64 pins and a 1,000 transistor BIST) circuit results in an overhead of 17.7% and 5.8% respectively.

<table>
<thead>
<tr>
<th>System Component</th>
<th>Testability Overhead of this Boundary Scan Design</th>
<th>Testability Overhead of Typical Designs (in Number of Transistors)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Number for Non-Test Design</td>
<td>LSI Design (1,000)</td>
</tr>
<tr>
<td>Total Number for Non-Test Design</td>
<td>18</td>
<td>135</td>
</tr>
<tr>
<td>TAP</td>
<td>37</td>
<td>326</td>
</tr>
<tr>
<td>Instruction Register</td>
<td>8</td>
<td>169</td>
</tr>
<tr>
<td>Boundary Scan Register</td>
<td>5*4=20</td>
<td>5*66=330 (66 trans./pin)</td>
</tr>
<tr>
<td>Custom Scan Register</td>
<td>10</td>
<td>180</td>
</tr>
<tr>
<td>Bypass Register</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>DR Decode</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>DR Select MUX</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>Output Select MUX</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Output Enable Buffer</td>
<td>3*1=3</td>
<td>3*9=27 (9 trans./pin)</td>
</tr>
<tr>
<td>Total Number for Test Design</td>
<td>89</td>
<td>1098</td>
</tr>
<tr>
<td>Grand Total</td>
<td>107</td>
<td>1233</td>
</tr>
<tr>
<td>% Test Overhead</td>
<td>83.2</td>
<td>89.1</td>
</tr>
</tbody>
</table>

**Table 9** Number of Logic Cells and Estimated Transistors Required for Implementation. *The transistor data more accurately represents silicon area on an IC. this was extracted from the authors previous design and layout of TI's '8373 IC using CMOS technology*.

---

1 One BSR is associated with each pin (input and output).
Assuming that one half of the total pins are output pins.
Given the increased number of transistors in a boundary scan design, there will also be an increased power consumption. Fortunately, the testability functions are inactive during normal circuit operation. When the circuit is performing test functions, the use of CMOS or BiMOS technologies can maintain a low power consumption. The remaining issue, increased time of the circuit's operation, can be separated into the time to access normal features and testability features. When accessing normal features, there are time delays due to the extra multiplexer gating the output of each BSR. This delay can also be minimized by the choice of an appropriate technology. Boundary scan devices intrinsically keep the cycle time short by clocking data in on the rising edge and out on the falling edge of TCK. Even so, the time to access testability features is affected by the length of the serial scan path that links TDI and TDO of the test bus. This time can be minimized by using the Bypass operation to "short-cut" components not being tested.

The reliability of the component has been increased through addition of the 1149.1 test bus. At a cost of four test pins and the testability design overhead, 100% single stuck-at-fault coverage was obtained. The designer has to weight the increased device testability with the cost required to achieved it and the timing of the market window. The boundary scan development time will be most substantial in the initial stages of the design, but with experience and support of the various 1149.1 development packages, this time should decrease. Since no design is without fault, the time spent during the initial design usually rewards the designer with a decrease in overall testability time and, as a result, more reliable products entering the market.
4.2 Problems Encountered and Remaining Limitations

Most of the limitations of the 1149.1 testability bus are due to its brief history. The simple boundary scan instructions (SAMPLE, BYPASS and EXTEST) that were executed on the LV500 were typed in manually, and required 162 lines of code for the first example alone. The availability of software to automatically generate boundary scan test vectors has been limited. A few vendors (e.g., Teradyne L200 family) are beginning to provide automatic test generation tools take advantage of this market need. These tools and the proposed test bus language (Boundary-Scan Description Language–BSDL) will become the cornerstones of the 1149.1 standard and facilitates its growth in industry.

The LV500 is not on a scheduled backup routine, therefore crashes are inevitable. When writing detailed plans for a circuit, either use TekWAVES22 to port them from an existing QuickSim file or keep frequent backups of your test files. Backup simulation files can be recorded onto a floppy disk or at the LV500 or transferred to an Internet node through the ftp utility.

4.3 Suggestions for Future Enhancements

The examples in chapter 3 are suitable for a pull-out tutorial on the utilization of the LV500 and the basics of the 1149.1 testability bus. They also provide a good reference for someone who may be building a boundary scannable component. One potential advancement of this thesis is in the area of IC design. With the resources available at RIT, a boundary scan compatible IC could be fabricated and then digitally tested using the LV500. The design of the TAP, IR, BSR, Bypass register, Output enable buffer and DR decoders changes little for varying designs. These could be constrained and verified once then placed in a standard cell library for future use.
5.0 References


APPENDIX A  LISTING OF TIMING DIAGRAMS

Figure 80  BSCAN_1 e.g. 1 Timing Diagram. Timing diagram for the first Boundary scan example.
Figure 81  BSCAN_1 e.g. 2 Timing Diagram. Timing diagram for the second boundary scan example.
Figure 82  BSCAN_I e.g. 2 Timing Diagram (cont.).  Continuation of the timing diagram for the second boundary scan example.
Figure 83  BSCAN_{II} Timing Diagram. *Timing diagram for the third boundary scan example.*
Figure 84  BSCAN_II Timing Diagram (cont.). Continuation of the timing diagram for the third boundary scan example.
Figure 85  BSCAN_II Timing Diagram (cont.). *Continuation of the timing diagram for the third boundary scan example.*
<table>
<thead>
<tr>
<th><strong>TCK CYCLE(S)</strong></th>
<th><strong>TAP STATE AFTER TCK</strong></th>
<th><strong>DESCRIPTION OR COMMENT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test-Logic-Reset</td>
<td>TAP cycles on reset state.</td>
</tr>
<tr>
<td>2</td>
<td>Run-test/Idle</td>
<td>TAP begins advancing towards desired state.</td>
</tr>
<tr>
<td>3</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Capture-IR</td>
<td>IR is preloaded with “10000001.” TDO is active on falling edge of TCK.</td>
</tr>
<tr>
<td>6</td>
<td>Shift-IR</td>
<td>IR is ready to shift in the instruction. TDI is active prior to next state.</td>
</tr>
<tr>
<td>7-13</td>
<td>Shift-IR</td>
<td>Serially load BYPASS (11111111) instruction into the IR.</td>
</tr>
<tr>
<td>14</td>
<td>Exit1-IR</td>
<td>The last bit of the instruction is shifted into IR upon entering this state.</td>
</tr>
<tr>
<td>15</td>
<td>Update-IR</td>
<td>IR is updated with new instruction (BYPASS). TDO goes into high impedance state on TCK’s falling edge.</td>
</tr>
<tr>
<td>16</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Capture-DR</td>
<td>Bypass register preloads 0. TDO becomes active.</td>
</tr>
<tr>
<td>18</td>
<td>Shift-DR</td>
<td>Bypass register is in scan path so that data will shift directly from TDI to TDO.</td>
</tr>
<tr>
<td>19-20</td>
<td>Shift-DR</td>
<td>“101” is shifted through the bypass reg. Note that the last bit (1) remains in the bypass register and is not shifted to the next element.</td>
</tr>
<tr>
<td>21</td>
<td>Exit1-DR</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Update-DR</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Test-Logic-Reset</td>
<td>The test operation is completed.</td>
</tr>
</tbody>
</table>

**Table 10**  BSCAN_1 Timing Table e.g. 1. *Timing table for the first LV500 boundary scan example.*
<table>
<thead>
<tr>
<th>TCK CYCLE(S)</th>
<th>TAP STATE AFTER TCK</th>
<th>DESCRIPTION OR COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test-Logic-Reset</td>
<td>TAP cycles on reset state.</td>
</tr>
<tr>
<td>2</td>
<td>Run-test/Idle</td>
<td>TAP begins advancing towards the desired state.</td>
</tr>
<tr>
<td>3</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Capture-IR</td>
<td>IR is preloaded with “10000001.” TDO is active on falling edge of TCK.</td>
</tr>
<tr>
<td>6</td>
<td>Shift-IR</td>
<td>IR is ready to shift in the next instruction (SAMPLE). TDI is active prior to next state.</td>
</tr>
<tr>
<td>7-13</td>
<td>Shift-IR</td>
<td>Serially load the SAMPLE (00000010) instruction into the IR.</td>
</tr>
<tr>
<td>14</td>
<td>Exit1-IR</td>
<td>The last bit of the instruction is shifted into IR upon entering this state.</td>
</tr>
<tr>
<td>15</td>
<td>Update-IR</td>
<td>IR is updated with new instruction (SAMPLE). TDO goes into high impedance state on TCK's falling edge.</td>
</tr>
<tr>
<td>16</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Capture-DR</td>
<td>TDO becomes active. Data will be captured on the next rising edge of TCK.</td>
</tr>
<tr>
<td>18</td>
<td>Shift-DR</td>
<td>Boundary Scan Register is in the scan path. Data is shifted out through TDO as new data is shifted in through TDI.</td>
</tr>
<tr>
<td>19-35</td>
<td>Shift-DR</td>
<td>The TDI input is ignored. The contents of the sample register should be all logic 0's since no data was applied to the latch.</td>
</tr>
<tr>
<td>36</td>
<td>Exit1-DR</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Update-DR</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Test-Logic-Reset</td>
<td>The test operation is completed.</td>
</tr>
</tbody>
</table>

**Table 11** BSCAN_I Timing Table e.g. 2. *Timing table for the second LV500 boundary scan example.*
<table>
<thead>
<tr>
<th>TCK CYCLE(S)</th>
<th>TAP STATE AFTER TCK</th>
<th>DESCRIPTION OR COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test-Logic-Reset</td>
<td>TAP cycles on reset state.</td>
</tr>
<tr>
<td>2</td>
<td>Run-test/Idle</td>
<td>TAP begins advancing towards the desired state.</td>
</tr>
<tr>
<td>3</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Capture-IR</td>
<td>IR is preloaded with &quot;10000001.&quot; TDO is active on falling edge of TCK.</td>
</tr>
<tr>
<td>6</td>
<td>Shift-IR</td>
<td>IR is ready to shift in the next instruction (EXTEST). TDI is active prior to next state.</td>
</tr>
<tr>
<td>7-13</td>
<td>Shift-IR</td>
<td>Serially load the EXTEST (00000000) instruction</td>
</tr>
<tr>
<td>14</td>
<td>Exit1-IR</td>
<td>The last bit of the instruction is shifted into IR upon.</td>
</tr>
<tr>
<td>15</td>
<td>Update-IR</td>
<td>IR is updated with new instruction (EXTEST). TDO goes into high impedance state on TCK's falling edge and the device is now in the test mode of operation.</td>
</tr>
<tr>
<td>16</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Capture-DR</td>
<td>TDO becomes active. Data will be captured on the next rising edge of TCK.</td>
</tr>
<tr>
<td>18</td>
<td>Shift-DR</td>
<td>Boundary Scan Register is in the scan path. The Current state shifts out through TDO and the new state shifts in through TDI.</td>
</tr>
<tr>
<td>19-35</td>
<td>Shift-DR</td>
<td>New state bits: Q3 @ TCK 24, Q2 @ TCK 25, Q1 @ TCK 26.</td>
</tr>
<tr>
<td>36</td>
<td>Exit1-DR</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Update-DR</td>
<td>The new state is available and clocked into the '8373. TDO goes into high impedance.</td>
</tr>
<tr>
<td>38</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Capture-IR</td>
<td>The IR is preloaded. TDO is active.</td>
</tr>
<tr>
<td>41</td>
<td>Shift-IR</td>
<td>IR is ready to shift in the next instruction (BYPASS).</td>
</tr>
<tr>
<td>42-48</td>
<td>Shift-IR</td>
<td>Serially load BYPASS (11111111) instruction into</td>
</tr>
<tr>
<td>49</td>
<td>Exit1-IR</td>
<td>The last bit of the instruction is shifted into IR.</td>
</tr>
<tr>
<td>50</td>
<td>Update-IR</td>
<td>The device enters normal mode of operation</td>
</tr>
<tr>
<td>51</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>Test-Logic-Reset</td>
<td>The test operation is completed.</td>
</tr>
</tbody>
</table>

**Table 12**  BSCAN_II Timing Table. *Timing table for the third LV500 boundary scan example.*
APPENDIX B  Code Generation

B.1  Unix Applications

B.1.1  Peg Input Commands

```
--Author: Ted H. Li
--File: tasppeg
--Description: Peg program for the finite state machine of the
--TAP coronary scan cell design.

INPUTS: TAP;
STATEMENTS: UPDATE_TAP, SHIFT_TAP, SELECT, ENABLE, CLOCK_TAP, UPDATE_IR, SHIFT_DR;

Desired:   ASSERT SELECT;
IS THE THEN SELECT ELSE Run Test Table;
Else:
  ASSERT SELECT, ENABLE;
  IF THE THEN SELECT ELSE Run Test Table;

SELECT:
  ASSERT SELECT;
  IF THE THEN SELECT ELSE Display;

Display:
  ASSERT SELECT, ENABLE,
  CLOCK DR, COUNTER;
  IF THE THEN Display ELSE Shift;

Shift:
  ASSERT SELECT, ENABLE, SHIFT DR, CLOCK DR;
  IF THE THEN Display ELSE Shift;

State:
  ASSERT SELECT, ENABLE;
  IF THE THEN Display ELSE Fail;

Fail:
  ASSERT SELECT, ENABLE;
  IF THE THEN Display ELSE Fail;

Fail:
  ASSERT SELECT, ENABLE;
  IF THE THEN Display ELSE Fail;

Display:
  ASSERT SELECT, ENABLE, UPDATE IR, UPDATE DR;

Update IR:
  ASSERT SELECT, ENABLE, SELECT, COUNTER, SHIFT DR;
  IF THE THEN Display ELSE Shift;

Update DR:
  ASSERT SELECT, ENABLE, SELECT, COUNTER, Shift DR;
  IF THE THEN Display ELSE Shift;

Reset:
  ASSERT SELECT, ENABLE, SELECT, SHIFT DR;
  IF THE THEN Display ELSE Shift;

Exit:
  ASSERT SELECT, ENABLE, SELECT, SHIFT DR;
  IF THE THEN Display ELSE Shift;

Start:
  ASSERT SELECT, ENABLE, SELECT, UPDATE IR, SHIFT DR;
  IF THE THEN Display ELSE Shift;
```

154
B.1.2 Peg Eqn Output (Eqntott Input File)
B.1.3 Peg Summary Output File

B.1.4 Peg Truth-Table Output File
B.1.5   Eqntott Output File (Espresso Input File)
B.1.6 Espresso Output File
B.2 Mentor Graphics Simulation Files

B.2.1 QuickSim Input Files

* File: tap.sim
* Author: Craig Loomis
* Description: QuickSim file for operating the TAP

Simulation file exercises TAP through predetermined sequence of states. The output is compared to desired output to see if TAP is functionally sound.

```
# BEGIN LOGIC SIMULATION SERVER V3.1_4.5 Friday, April 19, 1991 11:00:14 am (EST)
# RAZOR Hex
Assign: nil_Razor_raz Hex
Assign: nil_Inverter_razo Hex
Initial: Line 1
while Trace Time 100
    Initialize VAR
    PRAPE List:
End Trace 1:

TRACE INPUTS
    list 100
    list 100

TRACE OUTPUTS
    type 3;
    line 3;
    trace 3;
    trace 3;

    type 2;
    line 2;
    trace 1;
    trace 1;

    type 1;
    line 1;
    trace 1;
    trace 1;

    type 0;
    line 0;
    trace 0;
    trace 0;

list = list 1;
list = list 1;

list = list 1;
list = list 1;
```

159
FORCE PLACES

* 
CLOCK Period 130
FORCE TLF 0 0 -Repeat
FORCE TLF 1.50 -Repeat

FORCE INPUTS

FORCE TIF 1 0
FORCE TLA 0 0
FORCE TLA 1 100
FORCE TMS 0 800
FORCE TMS 1 500
FORCE TMS 0 900
FORCE TMS 1 100
FORCE TIF 0 130
FORCE TIF 1 180
FORCE TIF 0 180
FORCE TIF 1 180
FORCE TIF 0 180
FORCE TIF 1 180
FORCE TIF 0 180
FORCE TIF 1 180
FORCE TIF 0 180
FORCE TIF 1 180
FORCE TIF 0 180
FORCE TIF 1 180
File: coffee.sim

Author: Craig Loomis

Description: QuickSim simulation file for basic coffee break circuit. The circuit contains no latches at this point and driven by state input bits (D0, D1, D2)

Friday, April 12, 2001 11:02:14 AM (CST)

------------------------------

PLACE Inputs

TRADE 0
TRADE D1
TRADE D2
TRADE E

PLACE Outputs

TRADE L
TRADE M
TRADE N
TRADE Outputs
TRADE P

PORT DATA

C 0m 0m 0m
C 0m 1m 0m
C 1m 1m 0m

PORT LINES

F.A. L
F.B. L
F.C. L
F.A. L
F.B. L
F.C. L
F.A. L
F.B. L
F.C. L

PORT D

F.A. D
F.B. D
F.C. D
F.A. D
F.B. D
F.C. D
F.A. D
F.B. D
F.C. D

PORT E

F.A. E
F.B. E
F.C. E
F.A. E
F.B. E
F.C. E
F.A. E
F.B. E
F.C. E

PORT F

F.A. F
F.B. F
F.C. F
F.A. F
F.B. F
F.C. F
F.A. F
F.B. F
F.C. F

PORT G

F.A. G
F.B. G
F.C. G
F.A. G
F.B. G
F.C. G
F.A. G
F.B. G
F.C. G

PORT H

F.A. H
F.B. H
F.C. H
F.A. H
F.B. H
F.C. H
F.A. H
F.B. H
F.C. H

PORT I

F.A. I
F.B. I
F.C. I
F.A. I
F.B. I
F.C. I
F.A. I
F.B. I
F.C. I

PORT J

F.A. J
F.B. J
F.C. J
F.A. J
F.B. J
F.C. J
F.A. J
F.B. J
F.C. J

PORT K

F.A. K
F.B. K
F.C. K
F.A. K
F.B. K
F.C. K
F.A. K
F.B. K
F.C. K

PORT L

F.A. L
F.B. L
F.C. L
F.A. L
F.B. L
F.C. L
F.A. L
F.B. L
F.C. L

PORT M

F.A. M
F.B. M
F.C. M
F.A. M
F.B. M
F.C. M
F.A. M
F.B. M
F.C. M

PORT N

F.A. N
F.B. N
F.C. N
F.A. N
F.B. N
F.C. N
F.A. N
F.B. N
F.C. N

PORT P

F.A. P
F.B. P
F.C. P
F.A. P
F.B. P
F.C. P
F.A. P
F.B. P
F.C. P

PORT Q

F.A. Q
F.B. Q
F.C. Q
F.A. Q
F.B. Q
F.C. Q
F.A. Q
F.B. Q
F.C. Q

PORT R

F.A. R
F.B. R
F.C. R
F.A. R
F.B. R
F.C. R
F.A. R
F.B. R
F.C. R

PORT S

F.A. S
F.B. S
F.C. S
F.A. S
F.B. S
F.C. S
F.A. S
F.B. S
F.C. S

PORT T

F.A. T
F.B. T
F.C. T
F.A. T
F.B. T
F.C. T
F.A. T
F.B. T
F.C. T

PORT U

F.A. U
F.B. U
F.C. U
F.A. U
F.B. U
F.C. U
F.A. U
F.B. U
F.C. U

PORT V

F.A. V
F.B. V
F.C. V
F.A. V
F.B. V
F.C. V
F.A. V
F.B. V
F.C. V

PORT W

F.A. W
F.B. W
F.C. W
F.A. W
F.B. W
F.C. W
F.A. W
F.B. W
F.C. W

PORT X

F.A. X
F.B. X
F.C. X
F.A. X
F.B. X
F.C. X
F.A. X
F.B. X
F.C. X

PORT Y

F.A. Y
F.B. Y
F.C. Y
F.A. Y
F.B. Y
F.C. Y
F.A. Y
F.B. Y
F.C. Y

PORT Z

F.A. Z
F.B. Z
F.C. Z
F.A. Z
F.B. Z
F.C. Z
F.A. Z
F.B. Z
F.C. Z

END
File: coffeee.sim
Author: Craig Loomis
Description: QuickSim simulation file for basic coffee break circuit with feedback latches in place. The circuit is driven through each state by controlling signal (B).

```
* LOCAL SIMULATION SERVER 7.0.4.5 Friday, April 19, 1991 11:02:14 am EDT
***************************************************************
VDD = 5V
RES1 = 5K fixed_resistor
RES2 = 5K fixed_resistor
SCL = 10K Tpoly 1
STOE = 10K Tpoly 1
INITIALISE DC
REALSList
TERM Time

PARAMETERS

PARAM IN
PARAM CN
PARAM RL

INITIAL LOOP

PARAM t e 0.0 10 0.5
PARAM t e 0.0 10 0.5

FOR t 0 10 0.5

PARAM t e 0.0 10 0.5
PARAM t e 0.0 10 0.5

```
```
QuickSim simulation file for basic coffee break circuit with feedback latches in place.

This circuit also has a multiplexer to choose the input path as follows:

- $S = 0$ --> driven externally without feedback
- $S = 1$ --> driven with feedback latches, internally

This circuit also has a multiplexer to choose the input path as follows:

- $S = 0$ --> driven externally without feedback
- $S = 1$ --> driven with feedback latches, internally
FORTE D 0
FORTE D 1 1000
FORTE D 2 4000
FORTE D 3 6000
FORTE D 4 8000
FORTE D 5 10000
FORTE D 6 12000
FORTE D 7 14000

FORTE D 0
FORTE D 1 400
FORTE D 2 600
FORTE D 3 800
FORTE D 4 1000

THEN DRIVE THE GEARSET THROUGH THE FEEDBACK LATCH
B.2.2 QuickFault Input Files

File: coffee.fault
Author: Craig Loomis
Description: QuickFault simulation file for basic coffee break circuit.

LOGIC SIMULATION VERIFICATION_3.3 Friday, April 19, 1991 11:00:14 am

END INPUTS

FOR B 1
FOR E 1
FOR X 1
FOR Y 1
FOR Z 1
FOR Q 1
FOR R 1
FOR T 1
FOR U 1
FOR V 1
FOR W 1
FOR X 1
FOR Y 1
FOR Z 1
FOR Q 1
FOR R 1
FOR T 1
FOR U 1
FOR V 1
FOR W 1

165
FILE: coffee_fault

Author: Craig Loomis

Description: QuickFault simulation file for basic coffee break circuit with feedback path.

LOGIC SIMULATION SERVER V7.0.4.3 Friday, April 10, 2001 11:00:14 am (EDT)

***********

FORAll CLOCKS

FORAll Inputs

FORAll Outputs

END
**File:** coffee2.fault

**Author:** Craig Loomis

**Description:** QuickFault simulation file for basic coffee break circuit with feedback latches in place.

* This circuit also has a multiplexer to choose the input path as follows:
  * $S = 0 \rightarrow$ driven externally without feedback
  * $S = 1 \rightarrow$ driven with feedback latches, internally

---

LOGIC SIMULATION SERVER V7.0.4.9_Feb 17, 1992 11:55:11 am (PDT)

**transcription n**

fault yep pass 400
cycle 100 ps

FOR $E$ CLOCKS

CLOCK Hello LOGO
FOR $E$ CLP 6 - Repeat
FOR $E$ CLP 1 B6 - Repeat

FOR $E$ INPUTS

FOR $E$ TTL 1
FOR $E$ MLA 4 9

FOR ELEMENT THE LATCH AND
- DRIVE THE INPUT DIRECTLY

FOR $E$ TTL

FOR $E$ T
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1
FOR $E$ T 1 1

FOR $E$ OUT

FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1
FOR $E$ OUT 1

THEN DRIVE THE REGISTERS
Through the feedback latch

<table>
<thead>
<tr>
<th>Place No.</th>
<th>1</th>
<th>14.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>16.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>16.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>16.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>16.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>16.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>16.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>0</td>
<td>14.00</td>
</tr>
<tr>
<td>Place B</td>
<td>1</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>0</td>
<td>17.00</td>
</tr>
<tr>
<td>Place L</td>
<td>1</td>
<td>17.00</td>
</tr>
</tbody>
</table>

```
* File: coffee_min_fault
* Author: Chris Loomis
* Description: QuickFault simulation file for basic coffee break circuit.
This file contains the minimal set of vectors to provide complete signal stuck at coverage.

```
LOGIC SIMULATION SERVER V7.0_4-5 Friday, April 19, 1991 11:01:14 am (PST)
******************************************************************************
fault file: basic.cfc
cycle level 0.

FORZE/all

<table>
<thead>
<tr>
<th>FORZE</th>
<th>B</th>
<th>L</th>
<th>I</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORZE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FORZE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FORZE</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>FORZE</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>FORZE</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>FORZE</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>FORZE</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>FORZE</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>FORZE</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>FORZE</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>FORZE</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>FORZE</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>
`---`---`---`---`---`---

April 10, 1001 11:00:14 am (FT? = SAA = 33333!3i!a

169
<table>
<thead>
<tr>
<th>FORCE DL 1</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORCE DO 1</td>
<td>1000</td>
</tr>
<tr>
<td>FORCE B</td>
<td>1100</td>
</tr>
<tr>
<td>FORCE DL 1</td>
<td>1100</td>
</tr>
<tr>
<td>FORCE DL 3</td>
<td>1200</td>
</tr>
<tr>
<td>FORCE DL 3</td>
<td>1200</td>
</tr>
<tr>
<td>FORCE DL 1</td>
<td>1200</td>
</tr>
</tbody>
</table>

END OF
### B.2.3 QuickFault Output File

**ALL_FAULTS**

<table>
<thead>
<tr>
<th>FILE</th>
<th>FAULT INST_ID/PIN_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>1</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>2</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>3</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>4</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>5</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>6</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>7</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>8</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>9</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>10</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>11</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>12</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>13</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>14</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>15</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>16</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>17</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>18</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>19</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>20</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>21</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>22</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>23</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>24</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>25</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>26</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>27</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>28</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>29</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>30</td>
<td>CT /156/10</td>
</tr>
<tr>
<td>31</td>
<td>CT /156/10</td>
</tr>
</tbody>
</table>

171
QuickFault STATUS REPORT
-------------------------

L2: L2 CYCLE COMPLETED 1397
FAILS: 1
INCORRECT DETECTION: 0
FAILS THIS FAB 0
FAILS THIS FAB GROUP 0
TEMPORARY FAILS 140
UNTESTABLE FAILS 0
TOTAL FAILS 160
DETECTED FAILS 150
UNDETECTED FAILS 0
INCORRECT TEMPORARY FAILS 0
INCORRECT TEMPORARY FAILS 0
DETECTED FAILS 150
UNDETECTED FAILS 0
INCORRECT TEMPORARY FAILS 0
INCORRECT TEMPORARY FAILS 0

[NOTE: IT = Detected, UT = Undetected, UT = UnTestable,
IN = Incorrect, IF = F. faults, IT = Possible Detected,
GST = Gradual, EQ = Equivalent to Previous Fault,
REP = Representative Fault, Not in Requested Fault Group]
B.3 LV500 Input Files

Note that the format of the listing file is different from that of the Pattern menu. The ASCII output files list the I/O signals in the default order, which is:

D, TCK, TDI, TDO, TMS, C, *OC, and Q  (for BSCAN_I.NEW)

TCK, TDI, TDO, TMS, CLK and *OC         (for BSCAN_I.II.NEW)

Also the unknowns are labeled with a "0" or "1" and not with an "X". Nonetheless, the file gives a complete listing of all the signals in the pattern menu and can serve as a reference when generating new Pattern menus.
B.3.1 BSCAN I.NEW Pattern File

II Seq. 1095 13:16
LV 500 RV641 LV Pattern Screen

pattern
1: "This program is used to explore the TI Boundary;"
2: "it is usable to explore the D-type latch (SN7472), first the;"
3: "A type latch, then the D-type latch, and finally,";
4: "the SAMPLE register will be explored."
5: ""
6: "We hold low for 400ns prior to initial testing. This;"
7: "allows time for the TAP to reach an initial state."
8: ""
9: "Part II: SAMPLE register scan;"
10: "Part III: D-type latch under normal operation;"
11: "The TAP SAMPLE register scan;"
12: ""
13: "SAMPLE TAP;"
14: ""TAP Register Reset 41;"
15: "End-Test Idle;"
16: ""Test-Test Idle;"
17: "Select TA-Mode;"
18: ""Select-Test Mode;"
19: "End-Test Idle;"
20: ""Select-Test Mode;"
21: "Data Input;"
22: "Data Input;"
23: "Data Input;"
24: "Data Input;"
25: "Data Input;"
26: "Data Input;"
27: "Data Input;"
28: "Data Input;"
29: "Data Input;"
30: "Data Input;"
31: "Data Input;"
32: "Data Input;"
33: "Data Input;"
34: "Data Input;"
35: "Data Input;"
36: "Data Input;"
37: "Data Input;"
38: "Data Input;"
39: "Data Input;"
40: "Data Input;"
41: "Data Input;"
42: "Data Input;"
43: "Data Input;"
44: "Data Input;"
45: "Data Input;"
46: "Data Input;"
47: "Data Input;"
48: "Data Input;"
49: "Data Input;"
50: "Data Input;"
51: "Data Input;"
52: "Data Input;"
53: "Data Input;"
54: "Data Input;"
55: "Data Input;"
56: "Data Input;"
57: "Data Input;"
58: "Data Input;"
59: "Data Input;"
60: "Data Input;"
61: "Data Input;"
62: "Data Input;"
63: "Data Input;"
64: "Data Input;"
65: "Data Input;"
66: "Data Input;"
67: "Data Input;"
68: "Data Input;"
69: "Data Input;"
70: "Data Input;"
71: "Data Input;"
72: "Data Input;"
73: "Data Input;"
74: "Data Input;"
75: "Data Input;"
76: "Data Input;"
77: "Data Input;"
78: "Data Input;"
79: "Data Input;"
80: "Data Input;"
81: "Data Input;"
82: "Data Input;"
83: "Data Input;"
84: "Data Input;"
85: "Data Input;"
86: "Data Input;"
87: "Data Input;"
88: "Data Input;"
89: "Data Input;"
90: "Data Input;"
91: "Data Input;"
92: "Data Input;"
93: "Data Input;"
94: "Data Input;"
95: "Data Input;"
96: "Data Input;"
97: "Data Input;"
98: "Data Input;"
99: "Data Input;"
100: "Data Input;"
Remember that the first bit of the IR follows next.

The truth is that the SAMPLE register is not loaded;
- 1 on rising edge of the next clock cycle. This is:
- . . . . the status out of the first SAMPLE:
- . . . . SAMPLER is rising edge, examined in table.

Et. is address LBA 

vt v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.

v t.
153: "task_T11" 00000000 1 1 1 1 0 0 00000000;
154: - "Update-DR (37)";
155: "task_T1100000 1 1 1 1 0 0 00000000;
156: "Select-DR-Scan (38)";
157: "task_T1110000 1 1 1 1 0 0 00000000;
158: "Select-IR-Scan (39)";
159: "task_T1120000 1 1 1 1 0 0 00000000;
160: "Test-Logic_Reset (40)";
161: "task_T1130000 1 1 1 1 0 0 00000000;
162: "THIS IS THE END "

B.3.2 BSCAN II NEW Pattern File

The program is used to examine a state machine that is controlled by memory scanable in the CPU.
APPENDIX C  Discussion of Printing Operation

This section provides a summary to the LV500 Operator's Manual\textsuperscript{24} which can be referenced for further questions. The menu information of the LV500 can be sent to a printer for output in one of two forms, either as an ASCII file or as a screen dump (i.e., copy of the screen). The former is created by pressing <Shift> and <Print> simultaneously to invoke the \texttt{Overlay} menu. Once in this menu the F6:PRINT ALL key creates a file of all of the current menu while F5:PRINT creates a file for only the visible portion of the current menu. This description looks similar to a C program and can be used in documenting or debugging the LV500 for a particular DUT. Note that this option is available for most but not all of the LV500's menus. This output can be sent to a file under the Print Output directory or sent directly to a serial printer. Files sent to the Print Output directory can be transferred or retrieved using the file transfer protocol (ftp). The source listing of Appendix B.3 were acquired this way. To further explore file transferring refer to the LV500 Operators Manual\textsuperscript{24}. If a printer is desired for the output, it must be connected to the serial port labeled "Auxiliary" on the back of the LV500 was used. The method of creating a screen dump is discussed next.

All LV500 menu figures for this report were generated via the screen copy operation. In order to retrieve a screen copy from the LV500, a parallel printer must be connected to the back of the LV500T (terminal of the LV500). Pressing <Print> will then create a screen dump of the current menu. The printer used for this tutorial was a monochrome ThinkJet. Due to the fact that a threshold value was used in creating the hard copy, the highlighting of the LV500's fields was not reproducible for the menu figures enclosed. If a ThinkJet printer is not used for the screen copies then refer to the LV500 Operators Manual\textsuperscript{24} page C-30 for specific information about setting up the terminal.
APPENDIX D  Contacts

Below is a list of valuable contacts and resources if you should have any questions or wish to expand upon this tutorial.

Craig Loomis, GE Medical Systems
loomisc@med.ge.com
(414) 548-2043
General questions on the thesis content and setup of the LV500.

Texas Instruments, Customer Response Center
(800) 336-5236
(800) 232-3200
For more information on the Texas Instruments' JTAG/IEEE 1149.1 products.

IEEE
(800) 678 IEEE
For additional literature pertaining to the 1149.1 Standard Test Access Port and Boundary-Scan Architecture.

Tektronix LV500
(800) TEK-WIDE
For more information on the LV500, including DUT cards and additional software to complement or to upgrade RIT's LV500.
APPENDIX E  SN74BCT8373 Specifications
Members of the Texas Instruments
SCOPE™ Family of Testability Products

Octal Test Integrated Circuits

Compatible With the IEEE Standard
1149.1-1990 (JTAG) Serial Test Bus

Functionally Equivalent to SN54/74F373
and SN54/74BCT373 In the Normal
Function Mode

Test Operation Synchronous to Test
Access Port (TAP)

Implement Optional Test Reset Signal on
TAP by Recognizing a Double-High Level
Voltage (10 V) on TMS Pin

SCOPE™ Instruction Set
- Conforms to the IEEE 1149.1-1990
  Boundary Scan Instructions
- Provides Data Compression of Inputs
- Provides Pseudo-Random Pattern
  Generation From Outputs
- Samples Input/Output Data and Toggles
  Device Outputs
- Places Device Outputs In
  High-Impedance State

Fabricated Using State-of-the-Art BICMOS
Technology

Package Options include Plastic
Small-Outline Packages, Ceramic Chip
Carriers, and Standard Plastic and Ceramic
300-mil DIPs

description

The SN54BCT8373A and SN74BCT8373A are
scan test devices with octal D-type latches that are
members of the Texas Instruments SCOPE™
testability IC family. This family of devices blends
test circuitry with standard logic functions to
facilitate testing of complex circuit board assem-
bles. Scan access to the test circuitry is
accomplished via the 4-wire test access port
(TAP) interface.

In the normal mode, these devices are functionally equivalent to the SN54/74F373 and SN54/74BCT373 octal
D-type latches. While in the normal mode, the test circuitry can be activated by the TAP to take snapshot samples
of the data appearing at the device pins or to perform a self test on the boundary test cells. Activating the TAP
in normal mode does not affect the functional operation of the SCOPE™ octal latches.
description (continued)

In the test mode, the normal operation of the SCOPE™ octal latch is inhibited and the test circuitry is enabled to observe and control the device's I/O boundary. When enabled, the test circuitry can perform boundary scan test operations as described in IEEE Standard 1149.1-1990. Four dedicated test pins are used to control the operation of the test circuitry: test data in (TDI), test data out (TDO), test mode select (TMS), and test clock (TCK). Additionally, the test circuitry can perform other testing functions such as parallel signature analysis on data inputs and pseudo-random pattern generation from data outputs. All testing and scan operations are synchronized to the TAP interface.

The SN54BCT8373A is characterized for operation over the full military temperature range of −55°C to 125°C. The SN74BCT8373A is characterized for operation from 0°C to 70°C.

FUNCTION TABLE
(normal mode, each latch)

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OE</td>
<td>LE</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>X</td>
</tr>
</tbody>
</table>

logic symbol†

† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

Pin numbers shown are for DW, JT, and NT packages.
## Terminal Functions

<table>
<thead>
<tr>
<th>PIN NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>GND</td>
<td>Ground.</td>
</tr>
<tr>
<td>LE</td>
<td>Latch enable. See function table for normal-mode logic. This control input, at a high level, activates the D inputs. This input has an internal pullup that forces the pin to a high level if left unconnected.</td>
</tr>
<tr>
<td>OE</td>
<td>Output enable. See function table for normal-mode logic. This input has an internal pullup that forces the pin to a high level if left unconnected.</td>
</tr>
<tr>
<td>TCK</td>
<td>Test clock. One of four pins required by IEEE Standard 1149.1-1990. Test operations of the BCT8373A are synchronous to the test clock. Data is captured on the rising edge of TCK and outputs change on the falling edge of TCK. An internal pullup forces TCK to a high level if unconnected.</td>
</tr>
<tr>
<td>TDI</td>
<td>Test data input. One of four pins required by IEEE Standard 1149.1-1990. The test data input is the serial input for shifting data through the instruction register or selected data register. An internal pullup forces TDI to a high level if left unconnected.</td>
</tr>
<tr>
<td>TDO</td>
<td>Test data output. One of four pins required by IEEE Standard 1149.1-1990. The test data output is the serial input for shifting data through the instruction register or selected data register. An internal pullup forces TDO to a high level when it is not active and is not driven from an external source.</td>
</tr>
<tr>
<td>TMS</td>
<td>Test mode select. One of four pins required by IEEE Standard 1149.1-1990. The test mode select input direct the BCT8373A through its test access port (TAP) controller states. An internal pullup forces TMS to a high level if left unconnected. The TMS pin also provides the optional test reset (TRST) signal of IEEE Standard 1149.1-1990. This is implemented by recognizing a third logic level, double-high (V_{HDD}), at TMS.</td>
</tr>
<tr>
<td>VCC</td>
<td>Supply voltage.</td>
</tr>
<tr>
<td>1D–8D</td>
<td>Data inputs. See function table for normal-mode logic. These inputs have internal pullups that force the pins to a high level if left unconnected.</td>
</tr>
<tr>
<td>1Q–8Q</td>
<td>Data outputs. See function table for normal-mode logic. These outputs have internal pullups that force the pins to a high level if left unconnected.</td>
</tr>
</tbody>
</table>
Figure 1. Tap State Diagram

state diagram description

The TAP proceeds through the states in Figure 1 according to IEEE Standard 1149.1-1990. There are six stable states (indicated by a looping arrow in Figure 1) and ten unstable states in the diagram. A stable state is defined as a state the TAP can retain for consecutive TCK cycles. Any state that does not meet this criterion is an unstable state.

There are two main paths though the state diagram: one to manipulate a data register and one to manipulate the instruction register. It is necessary to finish manipulating one register before accessing another.

Test-Logic-Reset

In this state, the test logic is not active and the device operates in its normal function mode. The TAP returns to this state in no more than five TCK cycles if TMS is high. The TMS pin has an internal pullup resistor that forces it high if it is unconnected or if a board defect causes it to be open-circuited. The device powers up in the Test-Logic-Reset state.
state diagram description (continued)

Run-Test/Idle
The TAP must pass through this state before executing any test operations. The test operations controlled by
the boundary control register (see Table 2) are performed while in the Run-Test/Idle state.

Select-DR-Scan, Select-IR-Scan
No specific function is performed in these states, and the TAP exits either of them on the next TCK cycle.

Capture-DR
The selected data register is placed in the scan path. Depending on the current instruction, data may or may
not be loaded or captured by that register on the rising edge of TCK, causing the TAP state to change. On the
falling edge of TCK in Capture-DR, TDO goes from the high-impedance state to the active state. If the TAP has
not passed through the Test-Logic-Reset state since the last scan operation, TDO enables to the level present
when it was last disabled. If the TAP has passed through the Test-Logic-Reset state since the last scan
operation, TDO enables to a low level.

Shift-DR
In this state, data is serially shifted through the selected data register from TDI to TDO on each TCK cycle. The
first shift does not occur until the first TCK cycle after entering this state (i.e., no shifting occurs during the TCK
cycle in which the TAP changes from Capture-DR to Shift-DR or from Exit2-DR to Shift-DR). On the falling edge
of TCK in Shift-DR, TDO goes from the high-impedance to the active state. If the TAP has not passed through
the Test-Logic-Reset state since the last scan operation, TDO enables to the value present in the
least-significant bit of the selected data register.

Exit1-DR, Exit2-DR
These are temporary states used to end the shifting process. It is possible to return to the Shift-DR state from
either Exit1-DR or Exit2-DR without recapturing the data register. The last shift occurs on the TCK cycle in which
the TAP state changes from Shift-DR to Exit1-DR. On the falling edge of TCK in Exit1-DR, TDO goes from the
active state to the high-impedance state.

Pause-DR
The TAP can remain in this state indefinitely. The Pause-DR state provides the capability of suspending and
resuming shift operation without loss of data.

Update-DR
If the current instruction calls for the latches in the selected data register to be updated with current data, the
latches are updated only during this state. TDO goes to the high-impedance state on the falling edge of TCK
in Update-DR.

Capture-IR
The instruction register is preloaded with a 10000001 pattern and placed in the scan path. On the falling edge
of TCK in Capture-IR, TDO goes from the high-impedance state to the active state. If the TAP has not passed
through the Test-Logic-Reset state since the last scan operation, TDO enables to the level present when it was
last disabled. If the TAP has passed through the Test-Logic-Reset state since the last scan operation, TDO
enables to a logic 0.

Shift-IR
While in this state, data is serially shifted through the instruction register from TDI to TDO on each TCK cycle.
The first shift does not occur until the first TCK cycle after entering this state (i.e., no shifting occurs during the
TCK cycle in which the TAP changes from Capture-IR to Shift-IR or from Exit2-IR to Shift-IR). On the falling edge
of TCK in Shift-IR, TDO goes from the high-impedance state to the active state and enables to a high level.
state diagram description (continued)

Exit1-IR, Exit2-IR
These are temporary states used to end the shifting process. It is possible to return to the Shift-IR state from either Exit1-IR or Exit2-IR without recapturing the instruction register. The last shift occurs on the TCK cycle in which the TAP state changes from Shift-IR to Exit1-IR. On the falling edge of TCK in Exit1-IR, TDO goes from the active state to the high-impedance state.

Pause-IR
The TAP can remain in this state indefinitely. The Pause-IR state provides the capability of suspending and resuming shift operations without loss of data.

Update-IR
The latches shadowing the instruction register are updated with the new instruction. TDO goes to the high-impedance state on the falling edge of TCK in Update-IR.

Instruction register description

Serial test information is conveyed by means of a 4-wire test bus. Commands, data, and control signals are all passed along the 4-wire bus. The function of the TAP is to extract the state control information and synchronous control signals for the bus and generate the appropriate on-chip control signals for the test structures in the device. The TAP monitors two signals from the bus, namely TCK and TMS. Figure 1 shows the TAP state diagram. The functional block diagram illustrates the IEEE Standard 1149.1-1990 4-wire test bus and boundary scan architecture and the relationship between the TAP, the test bus, and the boundary scan test elements.

Data is captured on the rising edge of TCK and outputs change after the falling edge of TCK.

As shown in the functional block diagram, the SN54BCT8373A contains an eight-bit instruction register and three data registers: the 18-bit boundary scan register, the two-bit boundary control register, and the one-bit bypass register. Any register can be thought of as a serial shift register with a shadow latch on each bit. Latches may be loaded during the Update-DR and Update-IR TAP states.

The instruction register (IR) is eight bits long and is used to tell the device what instruction is to be executed. Information contained in the instruction includes the test operation to be performed, the state of the functional inputs and outputs (whether or not the device will perform its normal function during the test operation), which of the three data registers is to be selected for inclusion in the scan path during the next data register scan operation, and from where to preload the data register during the Capture-DR state. Table 1 lists the instructions supported by the SN54BCT8373A. Any SCOPE™ instructions not supported default to BYPASS.

The IR is loaded during the Capture-IR state with the value 10000001. As an instruction is shifted in, this value is shifted out via TDO and can be inspected as verification that the IR is in the scan path.

The instruction register order of scan is shown in Figure 2.
data register descriptions

boundary scan register
The boundary scan register (BSR) contains 18 bits (one for each functional input and output on the device). The BSR is used to store test data that is to be applied internally and/or externally to the device and to capture and store data that is applied to the functional inputs and outputs of the device. The origination of the value loaded in the BSR during the Capture-DR state is determined by the current instruction.

The boundary scan register order of scan is shown in Figure 3.

```
+----------------+-------------------+-------------------+-------------------+
| BIT 17         | BIT 16            | BIT 15            | BIT 14            |
| LE (MSB)       | OE                | 1D                | 2D                |
+----------------+-------------------+-------------------+-------------------+
| BIT 8          | BIT 9             | BIT 10            | BIT 11            |
| 8D              | 7D                | 6D                | 5D                |
+----------------+-------------------+-------------------+-------------------+
| BIT 7          | BIT 6             | BIT 5             | BIT 4             |
| 1Q              | 2Q                | 3Q                | 4Q                |
+----------------+-------------------+-------------------+-------------------+
| BIT 0          | BIT 1             | BIT 2             |
| 8Q (LSB)       | 7Q                | 6Q                |
+----------------+-------------------+-------------------+-------------------+
```

Figure 3. Boundary Scan Register Order of Scan

boundary control register
The boundary control register (BCR) contains two bits and is used to implement additional test operations not included in the SCOPE™ instruction set, including pseudo-random pattern generation (PRPG) and parallel signature analysis (PSA) operations. The BCR retains its current value during the Capture-DR state. The BCR resets to the PSA operation.

The boundary control register order of scan is shown in Figure 4.

```
+----------------+-------------------+-------------------+
| TDI            | BIT 1             | BIT 0             |
| (MSB)          | (LSB)             | TDO               |
+----------------+-------------------+-------------------+
```

Figure 4. Boundary Control Register Order of Scan
data register descriptions (continued)

bypass register
The bypass register is a one-bit scan path that can be selected to shorten the length of the system scan path, thereby reducing the number of bits per test pattern that must be applied to complete a test operation. The bypass register is loaded with a zero during the Capture-DR state.

The bypass register order of scan is shown in Figure 5.

```
Figure 5. Bypass Register Order of Scan
```

Table 1. Instruction Register Opcodes

<table>
<thead>
<tr>
<th>BINARY CODE 6</th>
<th>SCOPE OPCODE</th>
<th>DESCRIPTION</th>
<th>SELECTED DATA REGISTER</th>
<th>MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0000000</td>
<td>EXTEST</td>
<td>Boundary scan</td>
<td>Boundary scan</td>
<td>Test</td>
</tr>
<tr>
<td>X0000001</td>
<td>BYPASS$</td>
<td>Bypass scan</td>
<td>Bypass</td>
<td>Normal</td>
</tr>
<tr>
<td>X0000010</td>
<td>SAMPLE/PRELOAD</td>
<td>Sample boundary</td>
<td>Boundary scan</td>
<td>Normal</td>
</tr>
<tr>
<td>X0000011</td>
<td>INTEST</td>
<td>Boundary scan</td>
<td>Boundary scan</td>
<td>Test</td>
</tr>
<tr>
<td>X0000100</td>
<td>BYPASS$</td>
<td>Bypass scan</td>
<td>Bypass</td>
<td>Normal</td>
</tr>
<tr>
<td>X0000101</td>
<td>BYPASS$</td>
<td>Bypass Scan</td>
<td>Bypass</td>
<td>Normal</td>
</tr>
<tr>
<td>X0001110</td>
<td>TRIBYP</td>
<td>Control boundary to high impedance</td>
<td>Bypass</td>
<td>Modified test</td>
</tr>
<tr>
<td>X0001111</td>
<td>SETBYP</td>
<td>Control boundary to I/O</td>
<td>Bypass</td>
<td>Test</td>
</tr>
<tr>
<td>X0001000</td>
<td>BYPASS$</td>
<td>Bypass scan</td>
<td>Bypass</td>
<td>Normal</td>
</tr>
<tr>
<td>X0001001</td>
<td>RUNT</td>
<td>Boundary run test</td>
<td>Bypass</td>
<td>Test</td>
</tr>
<tr>
<td>X0001010</td>
<td>READBN</td>
<td>Boundary read</td>
<td>Boundary scan</td>
<td>Normal</td>
</tr>
<tr>
<td>X0001101</td>
<td>READBT</td>
<td>Boundary read</td>
<td>Boundary scan</td>
<td>Test</td>
</tr>
<tr>
<td>X0001100</td>
<td>CELLTST</td>
<td>Boundary self test</td>
<td>Boundary scan</td>
<td>Normal</td>
</tr>
<tr>
<td>X0001101</td>
<td>TOPHIP</td>
<td>Boundary toggle outputs</td>
<td>Bypass</td>
<td>Test</td>
</tr>
<tr>
<td>X0001110</td>
<td>SCANCN</td>
<td>Boundary control register scan</td>
<td>Boundary control</td>
<td>Normal</td>
</tr>
<tr>
<td>X0001111</td>
<td>SCANCT</td>
<td>Boundary control register scan</td>
<td>Boundary control</td>
<td>Test</td>
</tr>
<tr>
<td>All others</td>
<td>BYPASS</td>
<td>Bypass scan</td>
<td>Bypass</td>
<td>Normal</td>
</tr>
</tbody>
</table>

† The SCOPE™ instruction set specifies even parity in the 8-bit instruction. This feature is not implemented in the 'BCT8373A.
‡ X = Don't care
§ A SCOPE™ opcode exists but is not supported in the 'BCT8373A.

Instruction register opcode descriptions
The 'BCT8373A runs test instructions based on the value scanned into the instruction register. The test functions are defined as follows:

boundary scan
This instruction simultaneously executes the IEEE Standard 1149.1 EXTEST and INTEST instructions. The boundary scan register is selected in the scan path. Data appearing at the device inputs and outputs is captured. Data previously loaded into the boundary scan register is applied to the device inputs and through the device outputs.
Instruction register opcode descriptions (continued)

bypass scan
This conforms to the IEEE Standard 1149.1-1990 BYPASS instruction. The one-bit bypass register is selected in the scan path. A logic 0 is loaded in the bypass register. The device operates in the normal mode.

sample boundary
This conforms to the IEEE Standard 1149.1-1990 SAMPLE/PRELOAD instruction. Data appearing at the device inputs and outputs is sampled without affecting normal device operation. The boundary scan register is selected in the scan path.

control boundary to high impedance
The device outputs are placed in the high-impedance state. The bypass register is selected in the scan path. Device inputs remain operational, and the internal logic function is performed.

control boundary to I/O
The data in the boundary scan register is applied to the functional inputs and through the device outputs. The bypass register is selected in the scan path.

boundary run test
A test operation is run as decoded by the boundary control register. The desired test must be preloaded in the boundary control register and the TAP placed in the Run-Test/Idle state. The four test operations decoded by the boundary control register are:

1. parallel signature analysis (PSA)
   Data appearing on the functional data inputs is compressed into 16 bits. An initial seed value should be scanned into the boundary scan register prior to performing this operation. Figure 6 shows the algorithm that generates the signature.

2. pseudo-random pattern generation (PRPG)
   A pseudo-random pattern is generated at the functional outputs. An initial seed value should be scanned into the boundary scan register prior to performing this operation. Figure 7 shows the algorithm by which the patterns are generated. Note that a seed value of all zeroes will not produce additional patterns.

3. simultaneous PSA and PRPG
   Both PSA and PRPG operations are performed as shown in Figure 8.

4. sample inputs/toggle outputs
   Data appearing at the functional inputs is sampled on each TCK rising edge, and the functional outputs are toggled on each TCK falling edge.

boundary read
The boundary scan register is selected in the scan path. No load operation is performed prior to shifting. This instruction is useful for inspecting data after a PSA operation.

boundary control register scan
The two-bit boundary control register is placed in the scan path. This register must be loaded prior to executing a boundary run test operation.

boundary self test
The boundary scan register is selected in the scan path. This operation tests the logic in the boundary scan cells by loading the inverse of the current value of the cells. By loading a known value in the boundary scan register, executing CELLTST, and inspecting the resulting data through a scan operation, the integrity of the boundary scan register can be verified.
Instruction register opcode descriptions (continued)

boundary toggle outputs
Functional outputs are toggled on each TCK falling edge. Data appearing on the device's functional inputs is not captured.

tap bits for PSA and PRPG

The BCR opcodes are as shown in Table 2. The use of these tap bits and the algorithms used for 8- and 16-bit PSA and PRPG operations are shown in Figures 6 through 8. The latch-enable (LE) and output-enable (OE) inputs are ignored during these operations (meaning they are not included in the algorithms) but do control the status (enabled or disabled) of the outputs.

Table 2. Boundary Control Register Opcodes

<table>
<thead>
<tr>
<th>BINARY CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT 1 → BIT 0</td>
<td></td>
</tr>
<tr>
<td>MSB → LSB</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>Sample inputs/toggle outputs</td>
</tr>
<tr>
<td>01</td>
<td>PRPG/16-bit mode</td>
</tr>
<tr>
<td>10</td>
<td>PSA/16-bit mode</td>
</tr>
<tr>
<td>11</td>
<td>Simultaneous PRPG and PSA/6-bit mode</td>
</tr>
</tbody>
</table>

A PSA operation on the eight data inputs proceeds as the eight data outputs are held static.
SN54BCT8373A, SN74BCT8373A
SCAN TEST DEVICES
WITH OCTAL D-TYPE LATCHES
SCBS044A–08373, JUNE 1990–REVISED APRIL 1992

A PRPG operation from the eight data outputs proceeds as the eight data inputs are ignored.

Figure 7. 16-Bit PRPG Configuration

Figure 8. 8-Bit PSA and PRPG Configuration

An eight-bit PSA operation proceeds on the eight data inputs while an 8-bit PRPG operation proceeds from the eight data outputs.

Timing description

All test operations of the 'BCT8373A are synchronous to the test clock (TCK). Data on the TDI, TMS, and functional inputs is captured on the rising edge of TCK. Data appears on the TDO and functional output pins after the falling edge of TCK.

The 'BCT8373A advances through the states in Figure 1 by changing the value of TMS and applying a clock pulse to TCK. A simple timing example is shown in Figure 9. In this example, the device begins in the Test-Logic-Reset state and is loaded with an instruction to select the bypass register in the scan path. The binary logic value 101 is shifted through the bypass register from TDI to TDO, and the device is returned to the Test-Logic-Reset state. Table 3 explains the function of the test circuitry during each TCK cycle.
### Table 3. Explanation of Timing Example

<table>
<thead>
<tr>
<th>TCK CYCLE(S)</th>
<th>TAP STATE AFTER TCK</th>
<th>DESCRIPTION/COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Test-Logic-Reset</td>
<td>Recycle on reset state.</td>
</tr>
<tr>
<td>2</td>
<td>Run-Test/Idle</td>
<td>Begin advancing toward desired state.</td>
</tr>
<tr>
<td>3</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Capture-IR</td>
<td>IR loads with 10000001; TDO becomes active after falling edge of TCK.</td>
</tr>
<tr>
<td>6</td>
<td>Shift-IR</td>
<td>Ready to shift in instruction; TDI must be active before next clock.</td>
</tr>
<tr>
<td>7–13</td>
<td>Shift-IR</td>
<td>A BYPASS instruction (11111111) is serially loaded into the IR.</td>
</tr>
<tr>
<td>14</td>
<td>Exit1-IR</td>
<td>Note that TMS goes high prior to TCK No. 14. The last bit of the instruction is shifted in as the TAP advances from Shift-IR to Exit-IR.</td>
</tr>
<tr>
<td>15</td>
<td>Update-IR</td>
<td>The IR is updated with the new instruction. TDO goes inactive (high-impedance) on the falling edge of TCK No. 15.</td>
</tr>
<tr>
<td>16</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Capture-DR</td>
<td>The bypass register loads with a logic 0; TDO becomes active.</td>
</tr>
<tr>
<td>18</td>
<td>Shift-DR</td>
<td>The bypass register is now in the scan path. Data will shift from TDI to TDO.</td>
</tr>
<tr>
<td>19–20</td>
<td>Shift-DR</td>
<td>The binary value 101 is shifted from TDI to TDO through the bypass register. Note that the last value shifted in (a logic 1) remains in the bypass register and is not shifted to the next test element.</td>
</tr>
<tr>
<td>21</td>
<td>Exit1-DR</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Update-DR</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Select-DR-Scan</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Select-IR-Scan</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Test-Logic-Reset</td>
<td>Test operation completed.</td>
</tr>
</tbody>
</table>

#### Figure 9. Timing Example

- **TCK**
- **TMS**
- **TDI**
- **TDO**

3-state (TDO) or don’t care (TDI)
absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

<table>
<thead>
<tr>
<th></th>
<th>SN54BCT8373A</th>
<th>SN74BCT8373A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage range, $V_{CC}$</td>
<td>4.5–5.5 V</td>
<td>4.5–5.5 V</td>
</tr>
<tr>
<td>Input voltage range (except TMS)</td>
<td>2–2 V</td>
<td>2–2 V</td>
</tr>
<tr>
<td>Input voltage range (TMS)</td>
<td>10–24 mA</td>
<td>10–24 mA</td>
</tr>
<tr>
<td>Voltage range applied to any output in the disabled or power-off state</td>
<td>-0.5 V to 5.5 V</td>
<td></td>
</tr>
<tr>
<td>Voltage range applied to any output in the high state</td>
<td>-0.5 V to $V_{CC}$</td>
<td></td>
</tr>
<tr>
<td>Current into any output in the low state:</td>
<td>SN54BCT8373A (TDO) 40 mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SN54BCT8373A (Any Q) 96 mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SN74BCT8373A (TDO) 48 mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SN74BCT8373A (Any Q) 128 mA</td>
<td></td>
</tr>
<tr>
<td>Operating free-air temperature range:</td>
<td>SN54BCT8373A -55°C to 125°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SN74BCT8373A 0°C to 70°C</td>
<td></td>
</tr>
<tr>
<td>Storage temperature range</td>
<td>-65°C to 150°C</td>
<td></td>
</tr>
</tbody>
</table>

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: The input voltage rating may be exceeded if the input clamp-current rating is observed.

recommended operating conditions

<table>
<thead>
<tr>
<th></th>
<th>SN54BCT8373A</th>
<th>SN74BCT8373A</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN</td>
<td>NOM</td>
</tr>
<tr>
<td>$V_{CC}$ Supply voltage</td>
<td>4.5</td>
<td>5</td>
</tr>
<tr>
<td>$V_{IH}$ High-level input voltage</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$V_{IHH}$ Double high-level input voltage</td>
<td>TMS 10</td>
<td>12</td>
</tr>
<tr>
<td>$V_{IL}$ Low-level input voltage</td>
<td>0.8</td>
<td>0.8</td>
</tr>
<tr>
<td>$I_{K}$ Input clamp current</td>
<td>-18</td>
<td>-18</td>
</tr>
<tr>
<td>$I_{OH}$ High-level output current</td>
<td>TDO</td>
<td>-3</td>
</tr>
<tr>
<td></td>
<td>Any Q</td>
<td>-12</td>
</tr>
<tr>
<td>$I_{OL}$ Low-level output current</td>
<td>TDO</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Any Q</td>
<td>48</td>
</tr>
<tr>
<td>$T_A$ Operating free-air temperature</td>
<td>-55</td>
<td>125</td>
</tr>
</tbody>
</table>
electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>TEST CONDITIONS</th>
<th>SN54BCT8373A</th>
<th>SN74BCT8373A</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MIN</td>
<td>TYP†</td>
<td>MAX</td>
</tr>
<tr>
<td>V_{IN}</td>
<td>$V_{CC} = 4.5,\text{V}$, $I_L = -18,\text{mA}$</td>
<td>-1.2</td>
<td>-1.2</td>
<td>V</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>Any Q</td>
<td>$V_{CC} = 4.75,\text{V}$, $I_{OH} = -3,\text{mA}$</td>
<td>2.7</td>
<td>3.4</td>
</tr>
<tr>
<td></td>
<td>$V_{CC} = 4.5,\text{V}$, $I_{OH} = -12,\text{mA}$</td>
<td>2.4</td>
<td>3.4</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td>$V_{CC} = 4.5,\text{V}$, $I_{OH} = -15,\text{mA}$</td>
<td>2</td>
<td>3.2</td>
<td>2</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>Any Q</td>
<td>$V_{CC} = 4.5,\text{V}$, $I_{OL} = 46,\text{mA}$</td>
<td>0.38</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>$V_{CC} = 4.5,\text{V}$, $I_{OL} = 64,\text{mA}$</td>
<td>0.3</td>
<td>0.5</td>
<td>0.35</td>
</tr>
<tr>
<td>$I_I$</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_I = 5.5,\text{V}$</td>
<td>1</td>
<td>1</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{IH}$</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_I = 2.7,\text{V}$</td>
<td>-1</td>
<td>-35</td>
<td>-100</td>
</tr>
<tr>
<td>$I_{IHH}$</td>
<td>TMS</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_I = 10,\text{V}$</td>
<td>70</td>
<td>-200</td>
</tr>
<tr>
<td>$I_{IL}$</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_I = 0.5,\text{V}$</td>
<td>50</td>
<td>50</td>
<td>µA</td>
</tr>
<tr>
<td>$I_{OZH}$</td>
<td>Any Q</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_O = 2.7,\text{V}$</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>TDO</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_O = 2.7,\text{V}$</td>
<td>-1</td>
<td>-35</td>
</tr>
<tr>
<td>$I_{OZH}$</td>
<td>Any Q</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_O = 0.5,\text{V}$</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>TDO</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_O = 0.5,\text{V}$</td>
<td>-70</td>
<td>-200</td>
</tr>
<tr>
<td>$I_{OS}$‡</td>
<td>$V_{CC} = 5.5,\text{V}$, $V_O = 0$</td>
<td>-100</td>
<td>-225</td>
<td>-100</td>
</tr>
<tr>
<td>$I_{CC}$</td>
<td>$V_{CC} = 5.5,\text{V}$, Outputs open</td>
<td>3.5</td>
<td>7.5</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>Outputs high</td>
<td>31</td>
<td>52</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Outputs low</td>
<td>1.5</td>
<td>3.5</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Outputs disabled</td>
<td>10</td>
<td>10</td>
<td>pF</td>
</tr>
<tr>
<td>$C_I$</td>
<td>$V_{CC} = 5,\text{V}$, $V_I = 2.5,\text{V}$ or 0.5,\text{V}</td>
<td>14</td>
<td>14</td>
<td>pF</td>
</tr>
<tr>
<td>$C_O$</td>
<td>$V_{CC} = 5,\text{V}$, $V_O = 2.5,\text{V}$ or 0.5,\text{V}</td>
<td>14</td>
<td>14</td>
<td>pF</td>
</tr>
</tbody>
</table>

† All typical values are at $V_{CC} = 5\,\text{V}$, $T_A = 25^\circ\text{C}$.
‡ Not more than one output should be shorted at a time, and the duration of the short circuit should not exceed one second.
SN54BCT8373A, SN74BCT8373A
SCAN TEST DEVICES
WITH OCTAL D-TYPE LATCHES
SCBS044A–O8373. JUNE 1990–REVISED APRIL 1992

Timing requirements over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>FROM (INPUT)</th>
<th>TO (OUTPUT)</th>
<th>$V_{CC} = 5, V$, $T_F = 25^\circ C$</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{clk}}$</td>
<td>Clock frequency</td>
<td>TCK</td>
<td>SN54BCT8373A MIN MAX SN74BCT8373A MIN MAX</td>
<td>MHz</td>
</tr>
<tr>
<td>$t_{w}$</td>
<td>Pulse duration</td>
<td>TCK high or low</td>
<td>25 25</td>
<td>25 25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LE high</td>
<td>5 5</td>
<td>5 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMS reset high</td>
<td>50* 50*</td>
<td>50 50</td>
</tr>
<tr>
<td>$t_{\text{su}}$</td>
<td>Setup time</td>
<td>Data before LE</td>
<td>3 3</td>
<td>3 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMS before TCK</td>
<td>12 12</td>
<td>12 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TDI before TCK</td>
<td>6 6</td>
<td>6 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any D before TCK</td>
<td>6 6</td>
<td>6 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OE before TCK</td>
<td>6 6</td>
<td>6 6</td>
</tr>
<tr>
<td>$t_{\text{h}}$</td>
<td>Hold time</td>
<td>Data after LE</td>
<td>2 2</td>
<td>2 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMS after TCK</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TDI after TCK</td>
<td>4.5 4.5</td>
<td>4.5 4.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Any D after TCK</td>
<td>4.5 4.5</td>
<td>4.5 4.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OE after TCK</td>
<td>4.5 4.5</td>
<td>4.5 4.5</td>
</tr>
<tr>
<td>$t_{\text{d}}$</td>
<td>Delay time</td>
<td>Power up to TCK</td>
<td>100* 100*</td>
<td>100 100</td>
</tr>
</tbody>
</table>

* On products compliant to MIL-STD-883, Class B, this parameter is not production tested.

Switching characteristics over recommended ranges of supply voltage and operating free-air temperature, $C_L = 50\, \text{pF}$ (normal mode) (unless otherwise noted) (see Figure 10)

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>FROM (INPUT)</th>
<th>TO (OUTPUT)</th>
<th>$V_{CC} = 5, V$, $T_F = 25^\circ C$</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{PLH}}$</td>
<td>D</td>
<td>Q</td>
<td>SN54BCT8373A MIN MAX SN74BCT8373A MIN MAX</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{\text{PHL}}$</td>
<td>LE</td>
<td>Q</td>
<td>3 6.8 9</td>
<td>3 11 3 11</td>
</tr>
<tr>
<td>$t_{\text{PLH}}$</td>
<td>LE</td>
<td>Q</td>
<td>3 6.7 8.5</td>
<td>3 11 3 11</td>
</tr>
<tr>
<td>$t_{\text{PLH}}$</td>
<td>OE</td>
<td>Q</td>
<td>3 6.5 8.5</td>
<td>3 10.5 3 10</td>
</tr>
<tr>
<td>$t_{\text{PHL}}$</td>
<td>OE</td>
<td>Q</td>
<td>3.5 7.5 9.5</td>
<td>3.5 11.5 3.5 11</td>
</tr>
<tr>
<td>$t_{\text{PHZ}}$</td>
<td>OE</td>
<td>Q</td>
<td>3 6.1 8</td>
<td>3 10 3 10</td>
</tr>
<tr>
<td>$t_{\text{PZL}}$</td>
<td>OE</td>
<td>Q</td>
<td>2.5 5.8 7.5</td>
<td>2.5 9.5 2.5 9.5</td>
</tr>
</tbody>
</table>
switching characteristics over recommended ranges of supply voltage and operating free-air temperature, \( C_L = 50 \text{ pF (test mode)} \) (unless otherwise noted) (see Figure 10).

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>FROM (INPUT)</th>
<th>TO (OUTPUT)</th>
<th>( V_{CC} = 5 \text{ V}, \ T_A = 25^\circ \text{C} )</th>
<th>( \text{SN54BCT8373A} )</th>
<th>( \text{SN74BCT8373A} )</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I_{\text{max}} )</td>
<td>TCK</td>
<td>Q</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>MHz</td>
</tr>
<tr>
<td>( I_{\text{PLH}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>6</td>
<td>13</td>
<td>15.5</td>
<td>6</td>
</tr>
<tr>
<td>( I_{\text{PHL}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>6</td>
<td>12</td>
<td>15.5</td>
<td>6</td>
</tr>
<tr>
<td>( I_{\text{PZH}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>6.5</td>
<td>14</td>
<td>17</td>
<td>6.5</td>
</tr>
<tr>
<td>( I_{\text{PHZ}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>7</td>
<td>15</td>
<td>20</td>
<td>7</td>
</tr>
<tr>
<td>( I_{\text{PLZ}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>6</td>
<td>14</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>( I_{\text{PLH}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>6</td>
<td>14</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>( I_{\text{PHL}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>3.5</td>
<td>7.6</td>
<td>10.5</td>
<td>3.5</td>
</tr>
<tr>
<td>( I_{\text{PZH}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>3.5</td>
<td>8</td>
<td>10.5</td>
<td>3.5</td>
</tr>
<tr>
<td>( I_{\text{PHZ}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>3</td>
<td>8</td>
<td>11.5</td>
<td>3</td>
</tr>
<tr>
<td>( I_{\text{PLZ}} )</td>
<td>TCK↓</td>
<td>Q</td>
<td>3</td>
<td>7.5</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>( I_{\text{PLH}} )</td>
<td>TCK↑</td>
<td>Q</td>
<td>7.5</td>
<td>16.5</td>
<td>20</td>
<td>7.5</td>
</tr>
<tr>
<td>( I_{\text{PHL}} )</td>
<td>TCK↑</td>
<td>Q</td>
<td>7.5</td>
<td>17</td>
<td>21</td>
<td>7.5</td>
</tr>
<tr>
<td>( I_{\text{PZH}} )</td>
<td>TCK↑</td>
<td>Q</td>
<td>8</td>
<td>18</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>( I_{\text{PHZ}} )</td>
<td>TCK↑</td>
<td>Q</td>
<td>8</td>
<td>18</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>( I_{\text{PLZ}} )</td>
<td>TCK↑</td>
<td>Q</td>
<td>8</td>
<td>18.5</td>
<td>22</td>
<td>8</td>
</tr>
</tbody>
</table>

Texas Instruments
PARAMETER MEASUREMENT INFORMATION

LOAD CIRCUIT

From Output Under Test

(see Note A)

C_L

R_L = R1 = R2

Test Point

LOAD CIRCUIT

Timing

Input 3 V

1.5 V

0 V

Data

Input 1.5 V

1.5 V

3 V

0 V

VOLTAGE WAVEFORMS
SETUP AND HOLD TIMES

High-Level Pulse

1.5 V

1.5 V

0 V

Low-Level Pulse

1.5 V

1.5 V

0 V

VOLTAGE WAVEFORMS
PULSE DURATIONS

Output Control

1.5 V

1.5 V

0 V

Waveform 1

(see Note C)

Waveform 2

(see Note C)

VOLTAGE WAVEFORMS
ENABLE AND DISABLE TIMES, 3-STATE OUTPUTS

NOTES:

A. C_L includes probe and jig capacitance.

B. Input pulses are supplied by generators having the following characteristics: PRR ≤ 10 MHz, Z_o = 50 Ω, t_r = 3 ns, t_f = 3 ns. For testing pulse duration: t_r = 1 to 3 ns, t_f = 1 to 3 ns. Pulse polarity can be either high-to-low-to-high or low-to-high-to-low.

C. Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.

D. The outputs are measured one at a time with one input transition per measurement.

Figure 10. Load Circuit and Voltage Waveforms