# Chapter 6.  Teaching over the Internet

## 6.1  Introduction

Technological advances in methods of communication and research are beginning to change the traditional scenarios of the classroom. Specifically, the exponential growth of the Internet has promoted opportunities for new ways of teaching and learning. Professors and students are discovering the unique challenges of the Internet and its tools, and they are discovering that it is a valuable source of information that breaks down many of the barriers associated with a traditional classroom.

The Internet is a powerful tool that offers an alternative to traditional teaching methods. It clearly has the potential to become a fundamental medium of teaching and learning. The advantages of the Internet are enormous. It allows educators to remain on the cutting edge of technology. As communication technologies become more familiar, students and educators are trying to incorporate the Internet resources into their educational modules.

The arrival of the World Wide Web signals the beginning of some fundamental changes in teaching, training, and self-directed learning. It is a vehicle for disseminating course materials and for creating active learning experiences within a classroom-based course. With the WWW one can to build on those experiences and incrementally move towards delivering the courses beyond classroom walls. Having access to the World Wide Web is like having the largest library in the world move in next door to you. For many people who are new to the web, the temptation to browse and pursue one link after another is so strong that they spend countless hours glued to the screen. In an educational or training situation, however, we don't have the luxury of spending all of our time that way. As much as we value curiosity and exploration, it's necessary for us to find ways to get the greatest learning benefit from the web in the shortest possible time.

With the WWW, users can find and retrieve information by navigating through linked HTML documents. HTML, an acrostic for Hypertext Markup Language, is the “language” used to create documents for the web. It is a very easy to learn text based (ASCII) tagging language for an author to display text, graphics or other multimedia applications on a page. The intent of the language is to provide a universally accepted method for browser programs such as Netscape or Internet Explorer to display the pages exactly as the author intended. In a hypertext document, selecting the highlighted word, phrase or image causes a new document or image to be retrieved and displayed. WWW documents lead the user to skip from one topic to another, retrieving information from servers scattered around the world.

Teaching does not merely involve presenting a series of chapters in a textual form along with a few hyperlinks, but requires effective student engagement with the training module itself. It is a well known fact that one learns by doing and not by just reading or receiving information. Hence the courseware provided over the Internet must be interactive and encourage student engagement. While the courseware provided over the Internet could help a user gain more knowledge than what he could obtain from a textbook, if the teaching material is not well organized, then the user could easily get frustrated and not learn anything at all. A key to a good tutorial is a good web site design, which will allow the users to explore the material without any difficulty.

## 6.2 Goals of the Computer-Based Educational module

The main goal of the VITAMINS project was to provide low cost modular learning experiences over the Internet on the use of VHDL and its related standards for the acquisition and maintenance personnel of the Air force industry.

The major requirements of the courseware are detailed below :

1. The courseware developed had to be structured in a way such that any changes could be easily incorporated without having to re-do the whole module.
2. The computer-based-educational (CBE) module was mainly developed to cater to the interests of the acquisition and maintenance personnel of the US Air Force. Hence the material provided in the CBE module had to be carefully chosen, based on the interests of those people.
3. Apart from providing textual material, a lot of effort and time had to be spent on making the course as interactive as possible, so that the users would find the information really useful in their day-to-day work. In order for the user to appreciate the material, it was required to encourage student engagement with the CBE module.
4. The courseware developed also had to keep abreast with the current information technology while sustaining the reader’s interest.

5. Another important requirement of the CBE module was to provide hyperlinks for accessing information about RASSP tools, VHDL models and data files.

All the material discussed in chapters 2,3,4 and 5 of this thesis has been carefully integrated into the CBE module. Apart from organizing the material, appropriate hyperlinks were made to all the material in [16] and the MIL-HDBK-62(Department of Defense handbook) for completeness of the module.

## 6.3 Methodology for Teaching

Designing a website is not merely writing a series of chapters and linking them together and presenting them as a course. A good and efficient design strategy had to be developed so that the readers get the fullest benefit possible from the instruction with the help of technology.

Before the actual development of the module, the overall structure of the module was first created. Key topics were identified bearing in mind the requirements of the acquisition and maintenance personnel. A *"Table of Contents"* (TOC) page was created which gave a general idea about the contents of the tutorial. This document was updated and modified as and when new topics were determined and added. Having decided the major topics, each topic was divided into further sub-sections.

Next all the material that was identified as a requirement for the tutorial was converted into a form suitable to be presented over the WWW. Since operation of the web relies mainly on HTML, all the material documented on a Word processor was converted to HTML by using a commercial tool such as Internet Assistant from Microsoft. New pages were then created as required with the help of a UNIX text editor.

Individual HTML pages were then generated based on the TOC file. The next important step was to effectively link the various HTML files together. Even though the WWW allows writers to present readers with a myriad of options, most readers prefer to be led through the material. Since reading is inherently a linear activity, most readers generally prefer to be taken from one page to the other, without them having to search for the succeeding section or chapter in a module. They often like to be able to flip through pages going backwards or forwards as though in a book. This was achieved by providing links to previous and next pages. In some cases the reader also likes to review the table of contents once in a while, hence the TOC file was linked to all the individual files, so that when browsing through a particular section, the reader not only has the option of going to the previous and next sections, but also back to the TOC. The main idea behind this was to allow the reader to take advantage of the non-linear nature of the WWW with ease, while at the same time progressing in an orderly way through the website.

Another important feature of WEB site design is the organization of the text material. It is really hard to read material from a computer screen, since the computer screen is brighter than a printed page. Also it is often difficult to read the horizontal lines of text while scrolling vertically. Hence the web pages were kept as short as possible in order to minimize scrolling and reduce strain on the user's eyes. A simple background color (like white) was chosen which was in good contrast with the lettering. Also margins to the left and right of each page were provided. This provides a separation between the text and the edge of the computer monitor making it easier to read.

An important feature of any Web based tutorial is continuity through the entire course material while sustaining the readers interest. This was achieved by arranging the various sections of the tutorial in a manner so as to provide a logical flow through the entire course.

Next we had to determine how exactly each topic could be explained. Since the main responsibility of the acquisition people is to write contracts, monitor them and make sure that they receive the right deliverables, they only need to know about the concepts of modeling and testing at a high level, and not detailed VHDL coding and syntax. Hence while explaining the various topics, care had to be taken to make them as visually accessible as possible. All the models developed for the project were explained in detail with the aid of block diagrams or waveforms. While explaining any of the testing procedures, care was taken to explain in a step-by-step form providing illustrations wherever possible, which makes it really easy for the user to understand the topic. Instead of embedding the VHDL code along with the explanation of any model or a testing procedure, the source code was documented in a separate file and a hyperlink was provided to the file containing the model description which provided an option to the reader to view the source code. However, the reader could quickly learn the important concepts without learning VHDL syntax.

In order to explain important block diagrams, we made use of imagemaps. This a feature of HTML which allows the user to click on any part of  a block diagram which is hyperlinked to another HTML page. For example, in order to explain Figure 3.1 of this thesis which is the test setup diagram for testing the MUT (Sobel Edge detector), we converted the *Test setup figure* into an imagemap, so that when the reader is viewing the figure, he or she can get information on the various components of the Test setup diagram by clicking on any of the components, each of which were linked to a HTML file providing complete information about the component.

Another important feature of HTML used extensively in the website was frames. Using frames in a website allows the screens to be divided into several different pages which act as separate entities. This feature was especially useful when explanations for a model or its source code was given. Instead of embedding the source code along with the explanation in a single HTML file, we have placed them in separate frames placed side-by-side so that the reader does not have to scroll back and forth to read the code and the explanation. In this way the reader can go through the source code simultaneously with the explanation. Since the source code is in a different frame, it does not disappear when the text is scrolled. Also links were made from sections of the code to the explanation so that the reader can go through both of them simultaneously for a better understanding. Frames were also extensively used when test results were presented for testing procedures like memory testing or diagnostic testing.

## 6.4 Implementation

A web site was developed for the VITAMINS project based on the methodology discussed in Section 6.3.

As stated in Section 6.3, before designing the website, an overall course structure was developed and the key topics were identified. Some of the major topics which were determined necessary for the course were ***Introduction***, ***Background***, ***Abstraction Levels***, ***Testing Techniques,*** and ***Related Links***. The *Introduction* chapter provided a brief overview of the VITAMINS project and its goals while the *Background* chapter provided literature information on topics like the Sobel Edge detection algorithm, WAVES and the RASSP design process. *Abstraction Levels* which was the third section was the most significant chapter of the CBE module. This chapter described the development of the case study model for the VITAMINS project (Sobel edge detector) from the behavioral level down to the gate level. Apart from explaining the modeling procedure and methodologies, it also included test bench development methodologies and techniques with examples at the various levels.

The main purpose of the VITAMINS project was to impart information on VHDL modeling and testing methodologies and concepts so as to help the help the Air Force acquisition personnel better their skills at negotiating and supervising development of VHDL models and increase the capability of the maintenance personnel to use VHDL for development of effective test benches and testing strategies. To achieve this purpose, the Sobel Edge detector was used as an example throughout to illustrate all the necessary concepts, since the model was simple enough for any student, yet complex enough to explain all the important concepts of modeling and testing. First the model was developed as an executable specification and this was then decomposed to the RTL level and then synthesized to obtain the gate level circuit. Details regarding the different abstraction levels of the Sobel Edge detector can be found in [16]. Since the major focus of the project was to teach the Airforce personnel how to develop a system and test it from the highest level down to the lowest level in hierarchy, all the material in the Abstraction levels section has been arranged hierarchically, so as to guide the reader through the complete cycle of model development and testing at different abstraction levels.

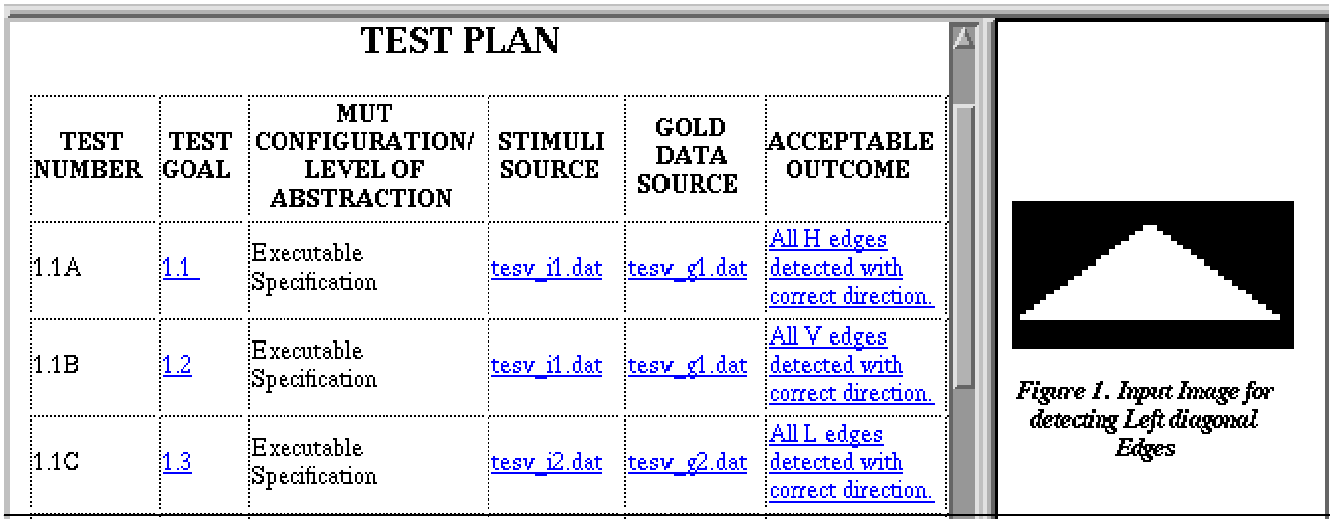
The material on the development of test benches, test plan, configurations and its use, and WAVES test benches at the various levels of abstraction forms a part of this thesis. Apart from this, the section on Testing Strategies also forms a part of the present research work. The subsequent pages demonstrate the implementation procedure for a few key topics on the website.

### 6.4.1 Implementation of Test plan

As stated in section 3.2.3, the functionality of a model can be effectively verified by a test plan which is a form of a table listing all the different test goals, stimuli source, gold data source, acceptable outcome and the coverage value for each test.

The main challenge in teaching this section was to effectively present Table 3.1 on the website. While developing the Test plan, care had to be taken to make the topic as interactive and interesting as possible so as to hold the reader's interest, while fostering understanding of the topic. Efficient usage of frames and HTML code has been made throughout for an effective implementation of this topic.

Figures 6.1 and 6.2 show portions of the test plan presented on the VITAMINS website.

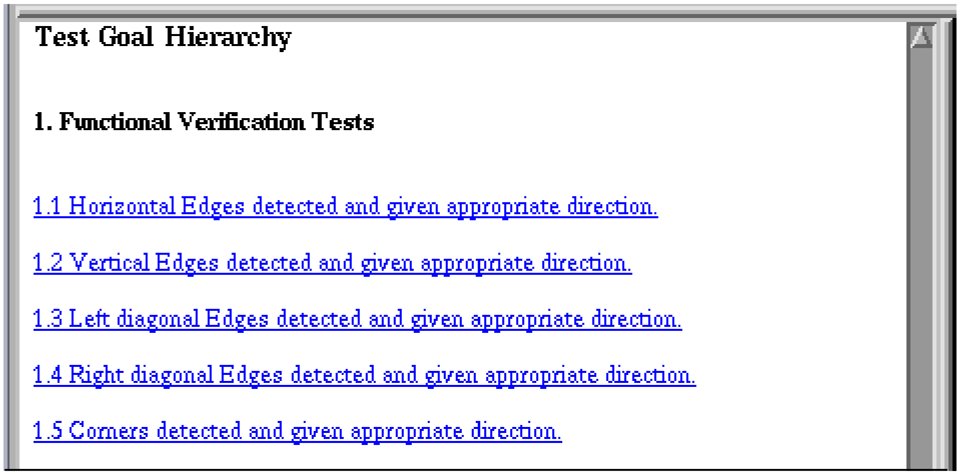


## TEST PLAN

| TEST NUMBER | TEST GOAL | MUT CONFIGURATION/ LEVEL OF ABSTRACTION | STIMULI SOURCE | GOLD DATA SOURCE | ACCEPTABLE OUTCOME |
|---|---|---|---|---|---|
| 1.1A | 1.1 | Executable Specification | tesv_i1.dat | tesv_g1.dat | All H edges detected with correct direction. |
| 1.1B | 1.2 | Executable Specification | tesv_i1.dat | tesv_g1.dat | All V edges detected with correct direction. |
| 1.1C | 1.3 | Executable Specification | tesv_i2.dat | tesv_g2.dat | All L edges detected with correct direction. |

*Figure 1. Input Image for detecting Left diagonal Edges*

**Figure 6.1 Top portion of the Test plan table.**



Test Goal Hierarchy

**1. Functional Verification Tests**

1.1 Horizontal Edges detected and given appropriate direction.

1.2 Vertical Edges detected and given appropriate direction.

1.3 Left diagonal Edges detected and given appropriate direction.

1.4 Right diagonal Edges detected and given appropriate direction.

1.5 Corners detected and given appropriate direction.

**Figure 6.2 Bottom portion of the Test plan table.**

Figure 6.1 represents the top portion of the test plan table. In this figure Table 3.1 has been displayed in the same form as shown in Section 3.3 of the thesis, however the contents of each column have been linked to an image or some explanatory comments for better understanding. For example, each of the columns of the table i.e., ***Stimuli source***, ***Gold data source*** and ***Acceptable outcome*** have been linked to an image pattern. When the user clicks on any of these columns, the corresponding image is displayed in the adjacent frame. This kind of implementation allows the user to clearly understand the test patterns and its output responses being referred to in each of the columns of the Test plan table.

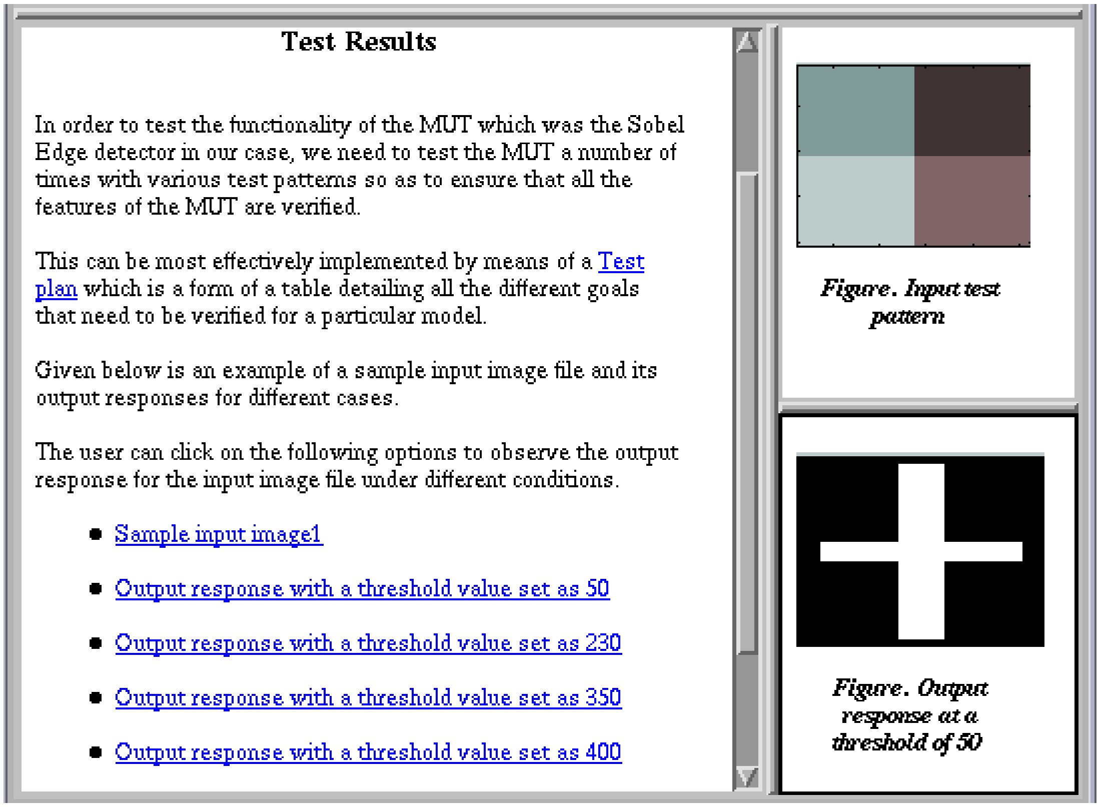
The image patterns for Gold data source and acceptable outcome have been placed along with the *Stimuli source* pattern so that when the user is observing the input pattern, he can simultaneously observe its expected(*GOLD data source*) and output response(*Acceptable outcome*) by just scrolling down the page. This allows the user to understand the functionality of the MUT very clearly by just viewing one file instead of having to look at two other files. Explanatory comments have been provided with each of the image patterns, for clear understanding. In Figure 6.1, the image pattern shown is for the stimuli source of test goal 1.3.

Figure 6.2 shows the lower portion of Table 3.1 in Section 3.3 of the thesis. Each of the test goals listed are linked to their respective test goal number in the upper portion of the table and vice-versa. This allows the user to click on the goal number in the upper portion of the Test plan table which immediately leads him to the actual test goal listed in the lower portion of the table without having to actually scroll down and trace the explanatory text for the particular goal number. Similarly the user can also click on the test goal in the lower portion of the table, which immediately takes him to the corresponding row in the upper portion of the table.

### 6.4.2 Implementation of Test results

Figure 6.3 shows a portion of the Test results section for the executable specifications of the MUT ( Sobel edge detector). The HTML page for this section is divided into three frames. Let us designate the large frame as frame1 and the upper and lower portions of the adjacent frame as frame2 and frame3.

In this section, a sample input image file has been provided along with different options for output response at different threshold values. Several options of output responses have been provided in order to make the user understand how threshold values affect the output response in an edge detector system. When the user clicks on the line “*Sample input image1*” in frame1, its corresponding image appears in frame2. The different output response patterns appear in frame3. Explanatory comments have been provided with each of the output response patterns, clearly explaining the reason for the particular response pattern. This section has also been linked to the Test goals section which lists all the different goals that the MUT needs to be tested for in order to verify the functionality of the MUT completely.



**Test Results**

In order to test the functionality of the MUT which was the Sobel Edge detector in our case, we need to test the MUT a number of times with various test patterns so as to ensure that all the features of the MUT are verified.

This can be most effectively implemented by means of a Test plan which is a form of a table detailing all the different goals that need to be verified for a particular model.

Given below is an example of a sample input image file and its output responses for different cases.

The user can click on the following options to observe the output response for the input image file under different conditions.

- Sample input image1
- Output response with a threshold value set as 50
- Output response with a threshold value set as 230
- Output response with a threshold value set as 350
- Output response with a threshold value set as 400

*Figure. Input test pattern*

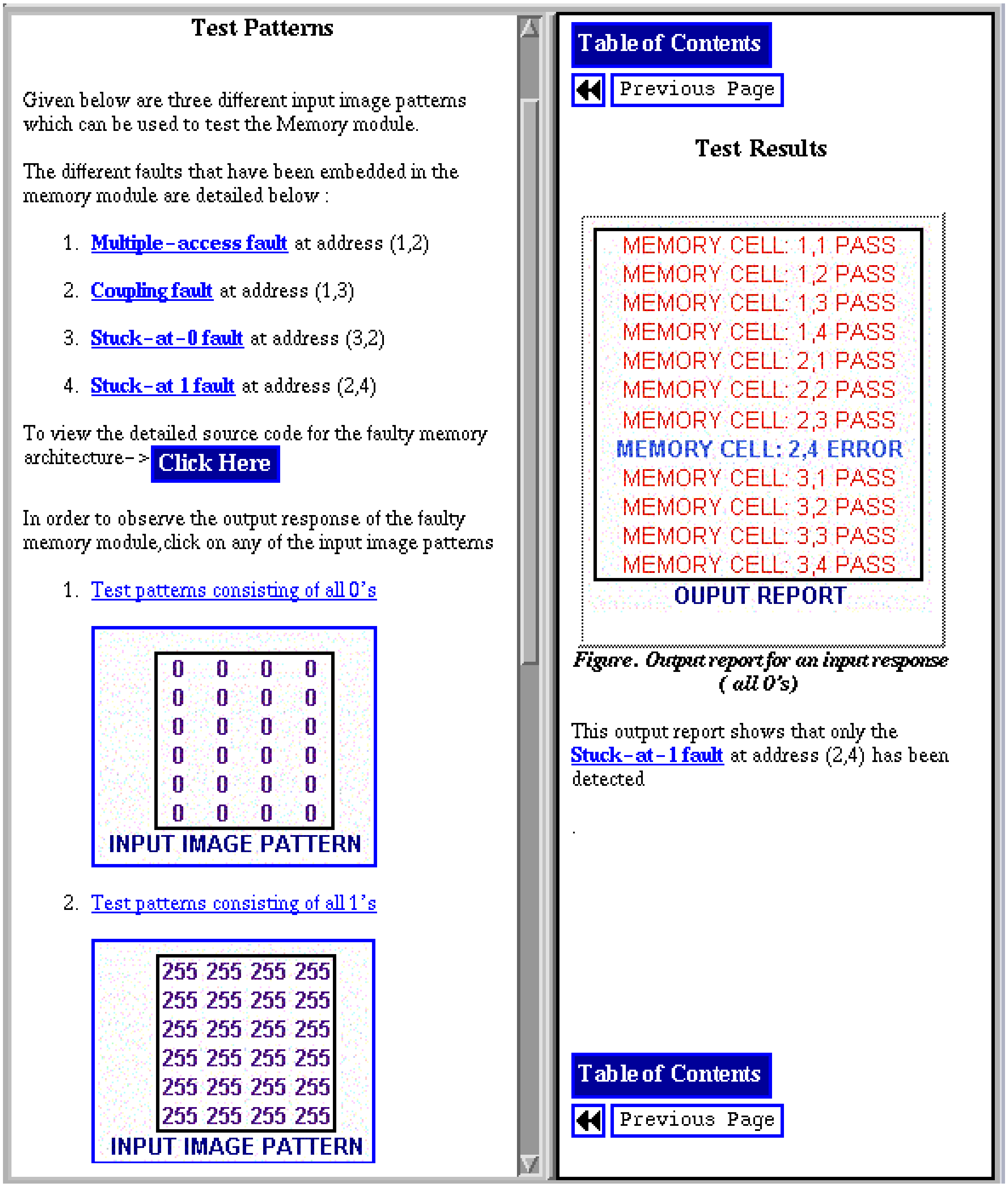
*Figure. Output response at a threshold of 50*

**Figure 6.3 Portion of the Test results section**

### 6.4.3 Implementation of Memory testing

Figure 6.4 shows a portion of the testing procedure and results for Memory testing, explained in Section 4.2 of the thesis.

The HTML page for this topic has been divided into two frames placed adjacent to each other. In an earlier page, the entire testing procedure detailed in Section 4.2 of the thesis was explained in a step-by-step form. The testing procedure was explained in a manner such that the user can easily understand the topic without having to refer to any source code. However a hyperlink to the source code was also provided, which allows the user to view the source code if desired. This kind of an implementation was adopted because the Air Force acquisition personnel are usually interested only in methodologies and techniques of testing using VHDL and not in the syntax of the language itself. At the end of the testing procedure, a hyperlink was provided to the test patterns section, a portion of which is shown in Figure 6.4.



### Test Patterns

Given below are three different input image patterns which can be used to test the Memory module.

The different faults that have been embedded in the memory module are detailed below :

1. **Multiple-access fault** at address (1,2)
2. **Coupling fault** at address (1,3)
3. **Stuck-at-0 fault** at address (3,2)
4. **Stuck-at 1 fault** at address (2,4)

To view the detailed source code for the faulty memory architecture-> **Click Here**

In order to observe the output response of the faulty memory module, click on any of the input image patterns

1. Test patterns consisting of all 0's

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

INPUT IMAGE PATTERN

2. Test patterns consisting of all 1's

| | | | |
|---|---|---|---|
| 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 |

INPUT IMAGE PATTERN

### Test Results

```
MEMORY CELL: 1,1 PASS
MEMORY CELL: 1,2 PASS
MEMORY CELL: 1,3 PASS
MEMORY CELL: 1,4 PASS
MEMORY CELL: 2,1 PASS
MEMORY CELL: 2,2 PASS
MEMORY CELL: 2,3 PASS
MEMORY CELL: 2,4 ERROR
MEMORY CELL: 3,1 PASS
MEMORY CELL: 3,2 PASS
MEMORY CELL: 3,3 PASS
MEMORY CELL: 3,4 PASS
```

OUPUT REPORT

***Figure. Output report for an input response ( all 0's)***

This output report shows that only the **Stuck-at-1 fault** at address (2,4) has been detected

.

**Figure 6.4 Portion of  Memory testing - Testing procedure and test results**

In Figure 6.4, the user has been provided with three different options of input image patterns each of which generates different output response patterns for the faulty memory architecture chosen to explain memory testing.

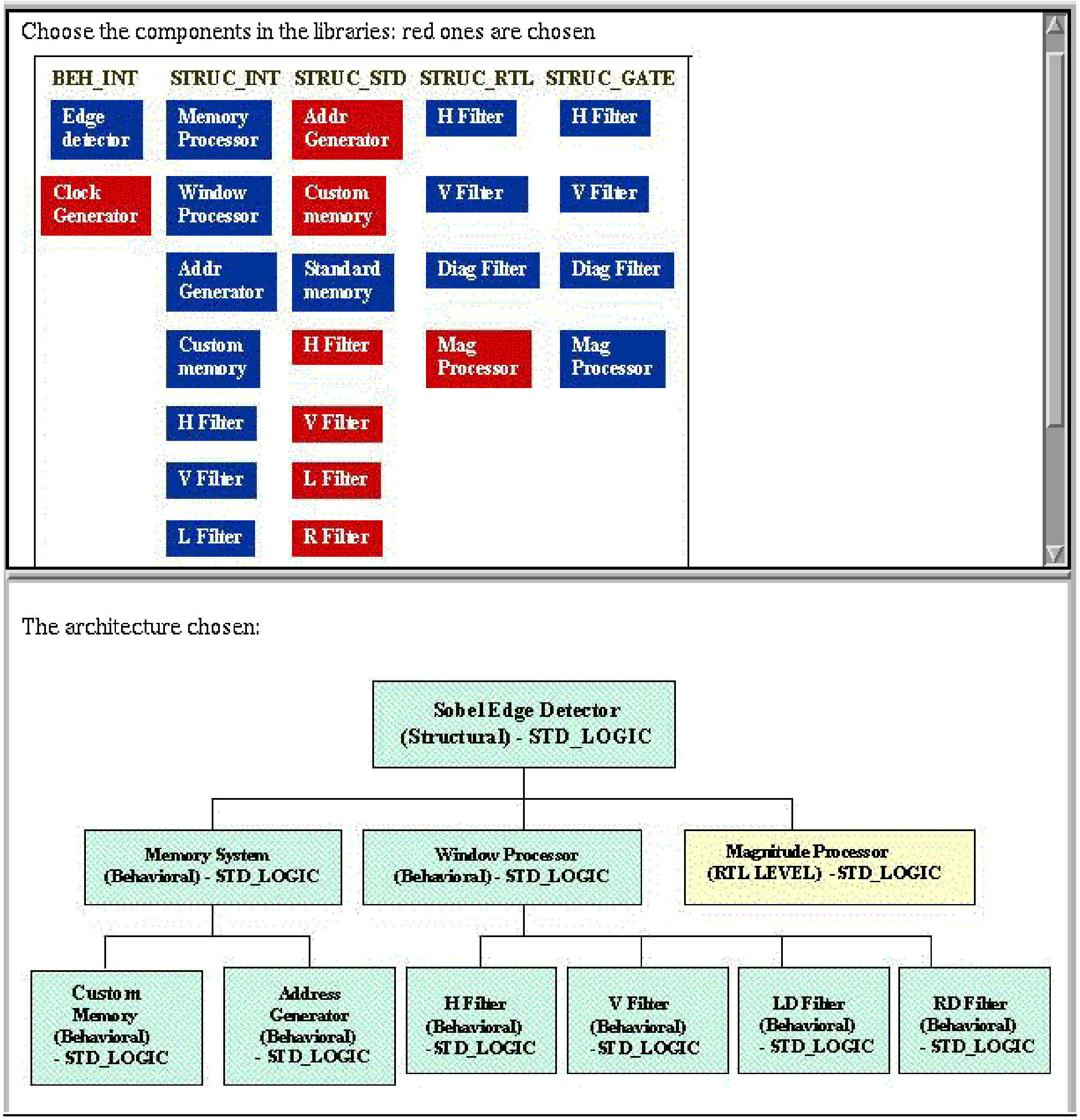
Before listing the various options for the input image patterns, all the different faults which were embedded in the sample faulty memory module have been listed clearly, so that the user has a clear understanding when he observes the output responses for the input test patterns. Hyperlinks have been provided between the faults listed in Figure 6.4 and their definitions which have been listed on a separate page, so that the user can quickly go to the related page and understand the definitions of the various faults. Explanatory comments have been provided with each output response explaining the type of fault which could be detected with a particular input pattern. By observing the different output responses for the various input image patterns, the user can clearly understand as to which pattern is most suitable to effectively detect faults in the memory module.

### 6.4.4 Implementation of Configurations

Figure 6.5 shows a portion of the configurations chapter explained in Section 3.2.2.6 and Section 3.3.3 of the thesis.

This lesson mainly constitutes two HTML pages. In the first page of this topic, the user is provided with several options of configuration declarations for the structural model of the MUT. The second HTML page consists of two frames. Let us designate the upper and lower frames as frame1 and frame2 respectively.

In frame1 all the components of the Sobel edge detector available in the various libraries namely *BEH_INT, STRUC_INT, STRUC_STD, STRUC_RTL and STRUC_GATE* are displayed in blue color. In frame2 we have a decomposition tree of an empty structural model of the Sobel Edge detector. When the user clicks on any of the options provided in the first HTML page, the empty structural model in frame2 is filled up with appropriate components from their respective libraries according to the listed option. Simultaneously in frame1, all the components chosen from the respective libraries are highlighted to red color, which gives the user a clear understanding as to which of the components have been chosen from the various libraries. Each of the options listed have also been hyperlinked to their respective source code which clearly explains the VHDL syntax for configurations for mixed abstraction level simulation, type conversion functions in the port maps and other key issues required in the configurations for mixed-data-type simulations. In this way, the user can easily view the source code and by quickly going to the respective HTML page. In the example shown in Figure 6.5, all the sub-components of the Sobel Edge detector are at the behavioral level ( STD_LOGIC data type ) except the magnitude processor which is at the RTL level of abstraction. Once the various components have been chosen from the respective libraries, the configuration file is simulated and its output response is verified with the expected response.



Choose the components in the libraries: red ones are chosen

| BEH_INT | STRUC_INT | STRUC_STD | STRUC_RTL | STRUC_GATE |
|---|---|---|---|---|
| Edge detector | Memory Processor | [✓ Addr Generator] | H Filter | H Filter |
| [✓ Clock Generator] | Window Processor | [✓ Custom memory] | V Filter | V Filter |
| | Addr Generator | Standard memory | Diag Filter | Diag Filter |
| | Custom memory | [✓ H Filter] | [✓ Mag Processor] | Mag Processor |
| | H Filter | [✓ V Filter] | | |
| | V Filter | [✓ L Filter] | | |
| | L Filter | [✓ R Filter] | | |

The architecture chosen:

Sobel Edge Detector (Structural) - STD_LOGIC
Memory System (Behavioral) - STD_LOGIC
Window Processor (Behavioral) - STD_LOGIC
Magnitude Processor (RTL LEVEL) - STD_LOGIC
Custom Memory (Behavioral) - STD_LOGIC
Address Generator (Behavioral) - STD_LOGIC
H Filter (Behavioral) - STD_LOGIC
V Filter (Behavioral) - STD_LOGIC
LD Filter (Behavioral) - STD_LOGIC
RD Filter (Behavioral) - STD_LOGIC

**Figure 6.5 Portion of the configurations chapter**