Chapter 2. Background

2.1 Test benches

2.1.1 Introduction

The most common way to verify the functionality of a hardware model is by simulation. The VHDL approach to check the functional correctness of a model is to use a model commonly referred to as a ***Test Bench***. A VHDL test bench is an environment to simulate and verify the operation of the Model Under Test (***MUT***) which represents a design under test. By combining the test bench with a top-down design methodology, designers can boost productivity and gain increased confidence in their circuits.

A test bench is at the top level in the model hierarchy. It is the entity that is simulated when testing the model. Testbenches thus provide the user with a capability to test the MUT thoroughly through simulation [5]. Figure 2.1 shows the block diagram of a basic test bench.

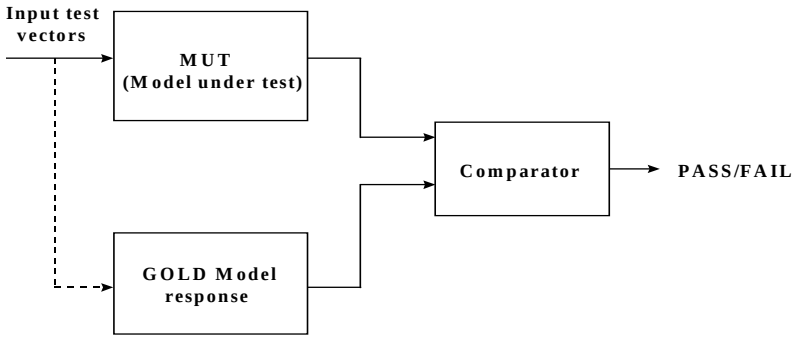

Figure 2.1 Basic Test bench block diagram

A set of input test vectors are applied to the model under test (MUT) and its output response is compared with the output response of the GOLD model in a comparator which generates a PASS / FAIL indication.

2.1.2 Types of Test benches

- **Off-line Test benches :**
  In an off-line test bench, the test bench module (TBM) reads the file of data and at the appropriate time sends the test vectors to the MUT [5]. The outputs of the MUT are stored in an external file. There can be some reformatting done in this process. For example, the input test vectors in the file could be of type integer, but the TBM could send the data to the MUT in the form of std_logic_vectors. In this type of test bench, the comparator reads the output response of the MUT and the GOLD model from external files and alerts the user if the two outputs match or don't match.

- **On-line Test benches :**
  In these test benches, the test vectors are applied simultaneously to both the MUT as well as to the GOLD model and the output responses are compared on-line, rather than being read from a file. The comparator then generates a PASS/FAIL indication.

- **Adaptive Test benches :**
  Adaptive test benches are the most complex type of test bench. These test benches compute the stimuli during the simulation by taking into account the actions of the MUT, i.e. they adapt their input stimuli to match the state of the MUT [12].

2.1.3 Test Vector Generation

The stimulus values or test vectors used to test the MUT are generally synthetically generated. While such vectors are useful for small-scale testing, they do not provide high test coverage. There may be errors in the model that are not revealed by a small number of vectors. More confidence about the model could be gained by providing significantly larger sets of test vectors. One approach is to create a file of test vectors. A test bench can read such files to simulate the model under test. The data-file approach is superior because the VHDL test bench will be considerably smaller, reducing compile time and simulator memory requirements. The advantage of this approach is that the stimulus can be changed without forcing recompilation of the VHDL test bench.

In this thesis, we have followed the approach detailed above. The input test vectors have been stored in text files. The test vectors could be data from an existing real world image like the M1A1 tank image shown in Figure 2.2 or could be synthetically generated by some existing commercial tools. The TBM reads the test data from a file and sends it to the MUT and again the outputs of the MUT are stored in separate text files by the TBM.

Figure 2.2 A real world image

2.2. Testing Strategies

Testing is a critical part of the design process for a digital system. It is a procedure, in which a set of test patterns are applied to the system and the resulting output response is observed to determine whether the system behaved correctly. If the system behaved incorrectly, then the second step would be diagnosis, which involves the location of the faulty component which caused the system to generate an incorrect response.

The purpose of testing at higher levels of abstraction involves functional verification and defining correct responses to the test vectors. During the top-down development of a design, a high level behavioral model can be used to generate the correct responses to a set of test vectors. When a less abstract model is subsequently developed, the responses of the low level model can be verified by comparisons with the results produced by the high level behavioral model. When testing for functionality, we need to develop test vectors that will test all the features of the MUT. Different test sets should be applied to the system to ensure that all the features of the MUT are tested effectively.

As we go down in hierarchy, especially at the Register Transfer and Gate level, there are other aspects to be considered apart from functionality. We need to test for performance problems like setup and holdtime violations, excessive response time of outputs to inputs, low clock rates and insufficient throughput.

A good method to ensure that the system has been tested effectively, is to develop a test plan which will clearly define the following :

- ***Test goals*** : Identify the different functional responses of the MUT that need to be verified at each abstraction level.
- ***Test patterns*** :Define the set of input stimuli associated with each test goal.
- ***Expected output response*** : Represent the pre-computed error-free outputs which are compared with the actual response of the MUT for a set of input stimuli associated with a test goal.

- ***Coverage measure*** : This could be statement coverage at the behavioral level and stuck-at-zero/ stuck-at-1 coverage at the gate level of abstraction.

Developing a test plan like the one detailed above provides confidence that the system has been tested effectively.

Another important aspect of the testing process apart from design verification is fault diagnosis. A fault will be defined informally as "*any condition that causes a device to function incorrectly*"[19]. In digital systems, typical maintenance goals deal with the rapid detection, location and repair of any system faults. Fault detection testing is the process of determining whether or not a fault is present in a given device. Fault diagnosis is the process of determining which component is faulty. The testing process involves the application of test patterns to the MUT and comparing the response of the MUT with a pre-computed expected response. Any discrepancy constitutes an error, the cause of which is said to be a *physical fault* [20]. Such faults for digital circuits can be classified as *logic* or *parametric* [20]. A logic fault is one which causes the logic function of the circuit to be changed to some other function. A typical logical fault is one where a circuit signal is fixed at a constant value say a logical one or zero. Such signals are said to be *stuck-at-one* or *stuck-at-zero* respectively. Many faults such as shorts and opens can be modeled as logical faults. Parametric faults alter the magnitude of the circuit parameters causing a change in some factors such as speed, current or voltage levels.

Testing and diagnosis must be performed throughout the life of the system, since faults may be introduced during assembly, storage and service. During each of these periods, the nature of the faults introduced and consequently the type of testing which must be performed is different. Generation of test patterns for these tests is an important problem and has been under investigation for a long time[21,22,23,24]. An efficient test sequence is seen as one in which the maximum number of faults is detected in a test vector set of given length. The goal is to provide a test sequence which can be applied quickly and which will at the same time assure correct MUT functionality with a high degree of confidence.

Test generation generally concentrates solely on detecting faults in a system. However, the maintenance specialists also needs to locate the fault so that the faulty component can be replaced. Fault location is used to debug circuits and to fix errors. Some problems include the following :

1. A test for one fault can simultaneously detect other faults in the circuit.
2. Two faulty circuits may also have identical responses for a particular test pattern.

Therefore test patterns have to be derived which not only identifies all the faults, but can also help in locating the fault from an analysis of the response.

2.3 Overview of the Sobel Edge detection system

Image processing is a technique commonly used in military image processing systems[13].

Figure 2.3 illustrates an image processing system that receives image data from a sensor and selects targets for weapons fire control. The raw image is first enhanced and then segmented to locate objects or regions of interest. Once an object or region has been located, it is examined for identifying features that can lead to final classification of targets [15]. Before an image can be segmented, the objects in that image must be detected so as to shape the boundary characteristics. Many algorithms have been developed for this purpose in the last two decades, the most common ones being Roberts, Sobel and the Prewitt algorithms[18]. Since the Sobel algorithm is the most popular algorithm used for edge detection purposes it will be used in the current thesis to illustrate testing concepts.

Figure 2.3 An Image Processing system

The Sobel algorithm convolves the incoming image data with four 3x3 masks(i.e.; Sobel operators) weighted to measure the differences in intensity along the horizontal, vertical and left and right diagonals.

Let us consider an arrangement of pixels about the pixel [i,j] shown in Figure 2.4. As observed in the figure each window is made up of nine elements, the current pixel [i,j] and its eight nearest neighbors which are represented by $a_0, a_1, ..., a_7$.

a_0	a_1	a_2
a_7	$[i, j]$	a_3
a_6	a_5	a_4

Figure 2.4 Typical arrangement of a window of pixels for an image.

Four weighted measurements along the horizontal, vertical, left and right diagonal directions can be computed as follows:

$$E_H = (a_2 + ca_1 + a_0) - (a_4 + ca_5 + a_6) \quad \textbf{-- (2.1)}$$

$$E_V = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \quad \textbf{-- (2.2)}$$

$$E_{DL} = (a_1 + ca_2 + a_3) - (a_7 + ca_6 + a_5) \quad \textbf{-- (2.3)}$$

$$E_{DR} = (a_1 + ca_0 + a_7) - (a_3 + ca_4 + a_5) \quad \textbf{-- (2.4)}$$

We will use c=2 as a typical value.

E_H, E_V, E_{DL} and E_{DR} can be implemented using the following convolution masks.

$H =$

1	2	1
0	0	0
-1	-2	-1

$V =$

-1	0	1
-2	0	2
-1	0	1

$DL =$

0	1	2
-1	0	1
-2	-1	0

$DR =$

2	1	0
1	0	-1
0	-1	-2

These four measurements E_H, E_V, E_{DL} and E_{DR} are then combined to compute the magnitude and phase gradients. The magnitude gradient can be estimated as [13] :

$$Magnitude = Max\left[\left|E_H\right|, \left|E_V\right|, \left|E_{DR}\right|, \left|E_{DL}\right|\right] + \frac{1}{8}\left[\left|E_{\perp}\right|\right] \quad \textbf{-- (2.5)}$$

where E_H, E_V, E_{DL} and E_{DR} are the four measurements along the four directions as already explained and $E_{\perp}$ is the measure in the direction perpendicular to the maximum. Magnitude is compared to a particular threshold to determine the edge detected pixels. A pixel is declared to an edge detected pixel if and only if the magnitude is greater than or equal to the threshold. If the pixel is an edge detected pixel, it is represented as 255 else it is represented as 0. Phase is represented by a 3 bit std_logic_vector which represents the phase angle quantized to 45 degrees resolution. Figure 2.5 shows the representation for phase values. A number from 0 to 3 represents the four basic directions shown in Figure 2.5 (000-Horizontal, 001-Left Diagonal, 010-Vertical, 011-Right diagonal).

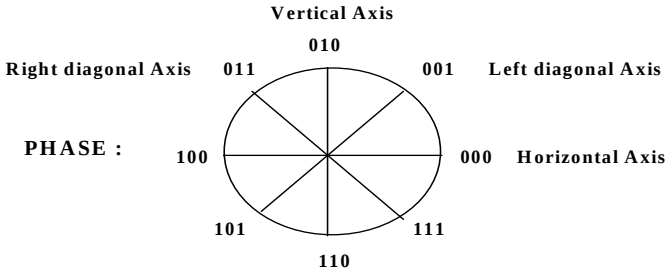

Figure 2.5 Representation for Quantized Phase.

The phase value is determined by choosing the number from 0 to 3 that corresponds to the perpendicular of the maximum magnitude. If the maximum magnitude is negative, then we add 4 to this value to compute the phase. For example, if E_H = (-50) is the maximum magnitude, then the phase is "110"(E_V = "010" is perpendicular to E_H , hence phase = "010" + "100").

The computation of the magnitude and phase values is now explained with a numerical example for a better understanding. Let us consider a 3x3 image as shown below :

1 5 2
3 6 9
8 7 4

The four Sobel coefficients are first applied to this image to compute the measurements along the four directions are :

E_H = (-13) E_V = (9) E_{DR} = (-14) E_{DL} = (-8)

Then $Max\left[\,|-13|\,,\,|9|\,,\,|-14|\,,\,|-8|\,\right] = 14.$

The maximum value corresponds to the Right Diagonal direction. Since E_{DR} =(-14), the phase is "101" (E_{DL} = "001" is perpendicular to E_{DR} , hence phase = "001" + "100").
Since the Left Diagonal is perpendicular to the Right diagonal, $E_{\perp} = E_{DL} = -8$.
Therefore,

$$Magnitude = Max\left[\,|-13|\,,\,|9|\,,\,|-14|\,,\,|-8|\,\right] + \frac{1}{8}\left[\,|-8|\,\right]$$
$$= 15$$

The basic flowchart for the Sobel Algorithm is shown in Figure 2.6. The algorithm consists of three sections. First we obtain the data from an external image and store it in a frame buffer. We then process the data which includes the computation of the four Sobel window outputs and then the magnitude and phase value for each pixel. The magnitude and phase results are then stored in external frame buffers. The implementation of this algorithm for the Sobel Edge Detector model developed for the VITAMINS project can be found in [16]. In this thesis, we will consider the models to be given and will develop methods to test these models.

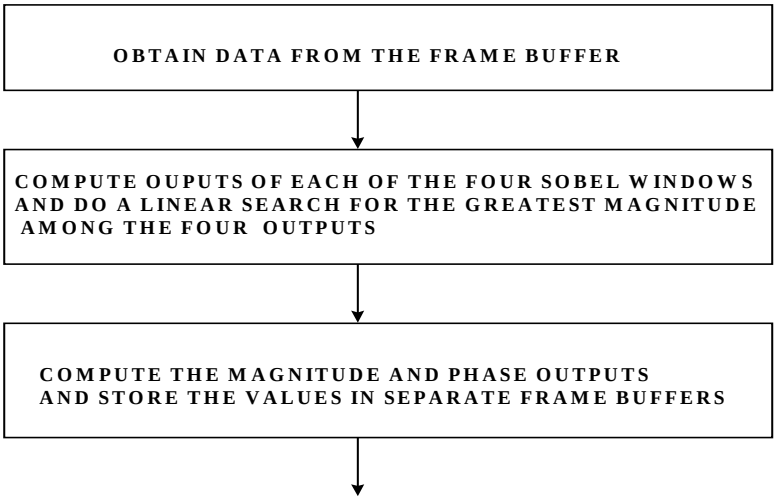

Figure 2.6 Flowchart of the Sobel Algorithm

2.4 WAVES

WAVES [9] (Waveform and Vector Exchange Specification) is the industry standard representation for digital stimulus and response for both the design and test communities. When a model is to be tested, the test usually consists of a number of input stimuli and a number of expected responses. WAVES provides a standardized environment for specifying these test waveforms.

The WAVES standard (IEEE Std 1029.1-1991) was developed by the WAVES Analysis and Standardization Group (WASG) in response to a need for a standard representation for exchanging information between the design and test environments. The words "*waveform*" and "*vector*" indicate that WAVES can represent simulator event trace data as well as the highly structured test vectors typical of automatic test equipment (ATE). The word "*exchange*" means that WAVES is meant for the exchange of information between environments but was not intended to replace stimulus/ response formats used within environments [9]. Together VHDL and WAVES provide powerful support for top-down design and test methodologies and concurrent engineering practices.

WAVES is a subset of VHDL (IEEE Standard 1076-87)[10]) and uses only sequential statements. Hence people familiar with languages like C or Ada should be able to understand WAVES[9]. VHDL was chosen as a basis for WAVES because it is the standard representation for modeling and simulating digital circuits and hence allows WAVES to serve as a standard stimulus/response format for VHDL at no cost to VHDL users.

WAVES is standardized at two levels of complexity, level 1 and 2. Level 1 is simple to use, but it meets only 90% of the industry's needs. Level 2 generates a standard that is more expensive to implement, but solves more difficult problems. Level 1 is a subset of level 2, which in turn is a subset of VHDL.

Detailed information on WAVES, concepts on Waveforms, WAVES datasets and operations can be found in the WAVES Language Reference Manual[9]. Test bench models generated using WAVES to test the Sobel Edge detector model are explained in detail in Chapter 5 of this thesis.

2.5 Computer Based Education

The use of computers to provide an integrated environment for teaching has received much attention in recent years. Until recently courseware existed as stand-alone packages, however with the advent of the World Wide Web (WWW) on the Internet and accompanying WWW browsers, such as Netscape, the development of teaching material has taken on a whole new dimension.

The Internet has now been accepted as the most efficient way to provide distance education [8]. This form of communication can be an invaluable tool in the hands of a person seeking information. The Internet allows educators and students to remain on the cutting edge of technology. Using the Internet as a reference resource for teaching purposes has not only the advantage of allowing quick access to vast resources, but also provides a valuable communication tool. In both cases, as a resource or as a communication tool, the Internet is superior to conventional education tools [8].

The World Wide Web is a distributed information delivery system which makes a huge array of resources available through simple sequences of mouse clicks on an enjoyable and easy-to-use graphical interface [7]. Operation of the web mainly relies on HTML (Hypertext Markup Language) as its means of interacting with users. HTML is a structured formatting language, and can be read, searched or edited and in addition to all these capabilities, it also contains pointers within the text to other documents. Apart from text, HTML can also be used to display 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.

Developing a tutorial over the Internet is by no means an easy task. It is not merely a matter of writing a series of chapters, linking them together and presenting the result as a tutorial. It is a well known fact that the web fails as a delivery mechanism when it merely acts as a page turner [17]. It is an old saying : " *We learn by doing, not by reading or listening*". Hence in order to encourage learning, we need to stimulate the reader and engage him in an interactive process.

A course which consists entirely of lecture-style presentation fails as a course because the student has no opportunity to practice. Key to an effective design, is to guide the users effortlessly into the flow of the pages, making navigation an easy, almost thoughtless process. The reason why website design is important is that if your readers are lost, or have to spend time thinking about where to go next, they are not focusing on the learning material. Their level of frustration is increased and they will switch off the computer.

The development of the computer based training module should mainly focus on the following issues :

- The interactive training module developed should keep abreast with the current information technology while sustaining the reader's interest.
- The training modules developed should have high reuse content i.e., the modules should be such that it can be easily updated with emerging technology without having to restart from scratch.

- A strategy on how to actually present the material in an efficient manner should be developed before the actual development of the educational module. Presenting the entire course in a textual form is uninteresting, hence care has to be taken to make the information as visually stimulating as possible.

- Web site design is the most important part of the project. Care has to be taken to ensure the development of an efficient design, where the readers need to learn as little as possible about navigating the course and focus more on the learning material itself.

In order to develop the on-line training module for the VITAMINS project, the World-Wide Web and its associated technologies were required as a delivery mechanism. The design strategy adopted to develop the modules and the key issues involved in the effective development and delivery of the Internet based educational module are explained in detail in Chapter 6 of this thesis.