CHAPTER V

SIMULATION AND VERIFICATION OF VHDL DESIGN OF 32-BIT FPAU IN SIMULINK

The HDL Code Generation step in MATLAB generates HDL code from the fixed-point Matlab code. One can generate either VHDL or Verilog code that implements the Matlab design. In addition to generating synthesizable HDL code, HDL Coder™ also generates various reports, including a traceability report that helps to navigate between the Matlab code and the generated HDL code. It also shows resource utilization report at the algorithm level, approximately what hardware resources are needed to implement the design, in terms of adders, multipliers, and RAMs [Simulink HDL Coder User’s Guide (2006-2010)]. In the recent, Abdullah and Hadi (2010), Mishra, Save and Rane (2011) and Valenzuela and Abdullah (2011) have used the graphical Matlab/Simulink environment for FPGA emulation, ASIC design, verification and chip testing. In the work carried out on VHDL code generation with Matlab/Simulink, there is hardly any reference of work in the literature for linking the VHDL design system on to the Matlab/simulink. Once the link between Modelsim and Matlab/simulink is created and the VHDL design is up-loading on to the simulink the same then can be optimized to get the optimal results in respect of power, area and speed.

This Chapter presents a very novel approach of linking the Modelsim and Simulink of Matlab for up-loading the VHDL design that is used for cross verifying the results of all arithmetic operations of addition/subtraction, division and multiplication by simulation in Modelsim wave window of Matlab after creating the simulink model and subsystem. It covers step 5 and 6 of the work flow depicted in Figure 4.2 of Chapter 4 and elaborated in the contributions in its succeeding section. It describes all the steps for complete process of linking Matlab and Modelsim. ModelSim is an easy-to-use yet versatile VHDL/ (System) Verilog/SystemC simulator by Mentor Graphics. It supports behavioural, Register Transfer Level (RTL) and gate-level modeling. Simulink model and subsystem is
generated and launched to simulate and verify VHDL coded design of 32-bit FPAU described in chapter III and implemented and verified on the FPGA platform/device in Chapter IV. The simulation results of all the floating arithmetic operations in the Modelsim window are presented at the end of the chapter.

5.1 Introduction

HDL Coder generates synthesizable HDL code for FPGA and ASIC implementations by first modeling the design using a combination of Matlab, Simulink and state flow charts. Then it optimizes the models to meet area-speed design objectives. After that HDL code is generated using the integrated HDL Workflow Advisor for MATLAB and Simulink. The generated code can be verified using HDL verifier. [Simulink HDL Coder User’s Guide (2006-2010)]

The HDL Workflow Advisor in HDL Coder automatically converts MATLAB code from floating-point to fixed-point and generates synthesizable VHDL and Verilog code. This capability lets you model your algorithm at a high level using abstract MATLAB constructs and System objects while providing options for generating HDL code that is optimized for hardware implementation. HDL Coder provides a library of ready-to-use logic elements, such as counters and timers, which are written in MATLAB.

The HDL Workflow Advisor generates VHDL and Verilog code from Simulink and State flow. With Simulink, one can model the algorithm using a library of more than 200 blocks, including State flow charts. This library provides complex functions for modeling signal processing and communications systems and generating HDL code.

5.2 Prior Work done in Matlab related to VHDL

Some of the works carried out by the scholars in the field of Matlab related to VHDL are described below:

Abdullah and Hadi (2010) present the design procedure and implementation results of a proposed software defined radio (SDR) using Altera Cyclone II family board. The implementation uses Matlab/SimulinkTM, Embedded MatlabTM blocks, and Cyclone II
development and educational board. The design is first implemented in Matlab/Simulink environment. It is then converted to VHDL level using Simulink HDL Coder. The design is synthesized and fitted with Quartus II 9.0 Web Editor software, and download to Altera Cyclone II board. The paper also presents an efficient design flow of the procedure followed to obtain VHDL netlists that can be downloaded to FPGA boards.

Mishra, Save and Rane (2011) applied embedding knowledge into model based design method, an efficient and cost-effective way to develop embedded systems. The approach is based on a single design description in the graphical Matlab/Simulink environment that is used for FPGA emulation, ASIC design, verification and chip testing.

Watson (2011) at Lockheed Martin Space Systems stated "With Simulink and HDL Verifier, simulation and verification are performed in one environment. As a result, one can test the design from end to end, improving quality and ensuring design accuracy and validity."

Welpot (2011) concentrates in rapid prototyping of duo-binary transmitter/receiver using the newest FPGA tools, including Simulink and Altera tools. FPGAs were chosen for this work because design changes can easily be performed on the field in very little time. On the other hand, it is also easier to make design changes, updates and troubleshooting to Simulink models than it is for source codes and as such, Simulink was used to implement the model for simulation and subsequent HDL code generation using the HDL coder of Matlab.

Valters (2011) in his paper describes a Matlab/Simulink GUI based tool for automated FPGA implementation of complex Jacobi-like Elementary Generalized Unitary rotation (EGU-rotation). The present work is targeted on multiplier-adder based rotation algorithm. The developed tool supports a large number of EGU-rotation matrix (EGURM) faces. The Symbolic Math Toolbox is used for operations with formulas. An intensive text processing has been used to get elementary expressions suitable for HDL coding. The tool uses Simulink HDL coder to generate implementable VHDL code.
Shi (2013) states that Model-based design (MBD) tools such as HDL Coder from MathWorks are becoming increasingly popular among system designers aiming for fast-prototyping and design reuse. In his paper he discusses the performance issues related to the synchronous designs generated by HDL Coder for multi-rate systems, and proposes an alternative GALS-based design flow that can achieve high performance efficiency and high design modularity at the same time.

From the above stated study, it reveals that in some of the works the graphical Matlab/Simulink environment has been used for HDL code generation, FPGA emulation, ASIC design, verification and chip testing but there is no mention of linking the VHDL design system on to the Matlab/simulink. In the next sections the steps for the novel approach for linking MATLAB and Modelsim followed by generation and launching of Simulink model and subsystem to simulate and verify VHDL coded design of 32-bit FPAU are described.

5.3 Linking of Matlab and Modelsim

For linking Matlab with Modelsim, HDL simulator, the compatible versions of Matlab / Simulink and HDL Simulator ‘Modelsim’ are required to be loaded on the same system. The work flow chart for basic design steps to create Simulink model for verification of VHDL code in Modelsim HDL Simulator is shown in Figure 5.1. The step by step process for linking Matlab with Modelsim is described below:

- Look for the possible Compatible Versions/ Release of Matlab and Modelsim
  - Matlab 2008 B or earlier is compatible with Modelsim 6.2, 6.3 and 6.4
  - Matlab 2009 A is compatible with Modelsim 6.2, 6.3 and 6.4
  - Matlab 2009 B or earlier is compatible with Modelsim 6.2, 6.3, 6.4 and 6.5
  - Matlab 2013 with Modelsim 10.2A, 10.0C

**Version of Matlab 7.12 2011A with Modelsim version 10.2A (PE Student Edition) are used in this work**

- Create VHDL file in a writable folder outside Matlab in C Drive
Invoke Socket Server
(For same machine having Matlab and Modelsim; as shared memory is preferable)

```matlab
>> hdldaemon
```
(This control command enables the Matlab to interact with HDL Simulator)
System responds as: HDLdaemon shared memory server is running with 0 connections

Figure 5.1: Design Steps to create Simulink Model for verification of VHDL code in Modelsim
Launch Modelsim from Matlab/Simulink

On Matlab command prompt

>> Change directory (cd) c:\ writable folder (My Play Area) in Matlab

>> cosimwizard

(An action window shown in Figure 5.2 pops up)

- Select: HDL cosimulation with “SIMULINK”
- Select: HDL Simulator with “MODELSIM” and Press “NEXT”
- Add VHDL script files “ADD”
  
  (go to the directory ‘My Play Area’ created in ‘C’ drive and select ‘work’ folder. Then select fp_alu.vhd) and press “Next “
- HDL files will be compiled on clicking “Next “
- Specify name of HDL module for cosimulation
  
  o Name of HDL module to cosimulate with “fp_alu” and press “Next”
- Specify all input and output ports
  
  (all input/output will be automatically loaded) and press “Next”
- Set all sample times and data types to ‘inherit’ and press “Next”
- Set clock and reset parameters if required and press “Next”
- Clock waveform appears. Change if required and press “Next”
- On clicking “Finish” on the Cosimulation wizard a new window in Figure 5.3 pops up

![Simulink Model window to link with HDL Simulator](image)

Figure 5.3 Simulink Model window to link with HDL Simulator

- Creates and opens a new Simulink model having HDL cosimulation block configured to the given specifications
- Compiles HDL code and Launches HDL Simulator
- Set timescale to “ms” and “Finish”

(This establishes link between Modelsim and Simulink)

- Simulink Model Window
  - Add three ‘constant’ blocks from ‘Library Browser’. Two constant blocks shall be for input 1 and input 2 and third block shall be the select line. Also add scope to view the output.
  - Add three ‘Data Type Conversion’ blocks for the three inputs
    Parameters setting for two inputs blocks, select line block and data type conversion blocks shall be as below:
    - For input 1 and input 2 constant block
      Double Clicking the input block a window shown below in Figure 5.4 pops up
      “Main” Constant Value=2.5 (say)
“Signal Attributes” Output data type: “Inherent from Constant Value” and Press “OK”

- For input 1 and input 2 “Data Type Conversion” setting
  Output data type “single” and Press “OK”

![Source Block parameters Block to set input ‘Constant’ Value](image)

Figure 5.4 Source Block parameters Block to set input ‘Constant’ Value

- For select line 3rd constant block
  Double Click input block and window shown in Figure 5.5 pops up
  “Main” Constant Value=00/01/10 (Binary)
  “Signal Attributes” Output data type: “uint8” and Press “OK”

![Source Block parameters Block to set ‘Select Line’ value](image)

Figure 5.5 Source Block parameters Block to set ‘Select Line’ value
• For select line “Data Type Conversion” setting
  Output data type “Inherent via back propagation” and Press “OK”
• Add clk in fp_alu/fp_alu_clk “Apply” and Press “OK”
- Save the model (with ext as .mdl) to “My Play Area” folder
- Select ‘Data Type Conversion’ blocks and Modelsim Block to create a subsystem
  □ Go to “Edit” window and Select “create subsystem”

(This creates a subsystem block)

This creates the final model for VHDL code fp_alu (FPAU) shown below in Figure 5.6 below:

Figure 5.6 Simulink model to generate and verify 32-bit FPAU

Input 1 and Input 2 are the two 32 bit floating point inputs to the model and ‘Select’ is set to ‘01’ for Adder, ‘11’ for Divider and ‘10’ for Multiplier. It also has a scope to view the output. A sub-system is created to launch the Modelsim Simulator from Simulink as shown in Figure 5.7.
5.4 Results and Discussions

Double clicking the ‘Launch HDL Simulator’ in the Simulink model loads the test bench for simulation. The Modelsim Simulator opens a display window for monitoring the simulation as the test bench runs. The wave window in Figure 5.8 shows the simulation of two exponential inputs and Select set to ‘01’ for ‘adder’ result as HDL waveform. Figure 5.9 shows the simulation of two decimal inputs for ‘adder’. Figure 5.10 and 5.11 show the simulation of two decimal inputs for ‘divider’. Figure 5.12 and 5.13 show the simulation of two decimal inputs for ‘multiplier’.

Figure 5.8 Simulation results of decimal inputs 1.1 & 1.1 for ‘adder’ in Modelsim wave window
Figure 5.9 Simulation results of decimal inputs 2.5 & 4.75 for ‘adder’ in Modelsim wave window

Figure 5.10 Simulation results of decimal inputs 1.1 & 1.1 for ‘divider’ in Modelsim wave window

Figure 5.11 Simulation results of decimal inputs 2.5 & 4.75 for ‘divider’ in Modelsim wave window
Figure 5.12 Simulation results of decimal inputs 1.1 & 1.1 for ‘multiplier’ in Modelsim wave window

Figure 5.13 Simulation results of decimal inputs for 2.5 & 4.75 ‘multiplier’ in Modelsim wave window
Table 5.1 below shows the input output details of the 32-bit FPAU architecture designed and linked using Simulink and Modelsim.

<table>
<thead>
<tr>
<th>Wave</th>
<th>Select</th>
<th>Input 1</th>
<th>Input 2</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 5.8</td>
<td>01</td>
<td>32’h3F8CCCCCD</td>
<td>32’h3F8CCCCCD</td>
<td>32’h404CCCCCD</td>
</tr>
<tr>
<td>Figure 5.9</td>
<td>01</td>
<td>32’h402000000</td>
<td>32’h409800000</td>
<td>32’h414400000</td>
</tr>
<tr>
<td>Figure 5.10</td>
<td>11</td>
<td>32’h3F8CCCCCD</td>
<td>32’h3F8CCCCCD</td>
<td>32’h3F06BCA1</td>
</tr>
<tr>
<td>Figure 5.11</td>
<td>11</td>
<td>32’h402000000</td>
<td>32’h409800000</td>
<td>32’h3F800000</td>
</tr>
<tr>
<td>Figure 5.12</td>
<td>10</td>
<td>32’h3F8CCCCCD</td>
<td>32’h3F8CCCCCD</td>
<td>32’h3F9AE148</td>
</tr>
<tr>
<td>Figure 5.13</td>
<td>10</td>
<td>32’h402000000</td>
<td>32’h409800000</td>
<td>32’h413E0000</td>
</tr>
</tbody>
</table>

5.5 Conclusions

The novel approach with all the design steps for linking Matlab and Modelsim followed by generation and launching of simulink model and subsystem have been described and implemented. The digital system of 32-bit FPAU using VHDL designed in Chapter 3 and tested on FPGA platform in Chapter IV has been simulated and verified in Modelsim using that simulink model. The results are verified through simulation for the same inputs as has been applied for practical implementations and testing of the digital system design of 32-bit FPAU on the FPGA platform/device described in Table 4.3. The next chapter is devoted to describe the complete process to create test bench for generation of optimized VHDL code of the designed FPAU system using this generated simulink model to carry out and consolidate the further work.