CHAPTER-VI

GENERATION OF OPTIMIZED VHDL CODE/DESIGN OF 32-BIT FPAU USING SIMULINK MODEL AND COMPARISON OF RESULTS

The very novel approach and methodology with all the design steps for linking Matlab and Modelsim followed by generation and launching of simulink model and subsystem have been described in the last chapter. The digital system of 32-bit FPAU using VHDL designed and duly tested on FPGA platform has also been simulated and verified in Modelsim using that simulink model. The approach is further extended to create test bench for optimization of the designed FPAU system using this generated simulink model. The optimized VHDL design of FPAU so obtained is synthesized using same ISE tool of Xilinx with the same FPGA platform/device as target for implementation which has been used for initial designed VHDL code of FPAU.

This chapter covers steps from 7 to 11 of the work flow depicted in Figure 4.2 of Chapter 4 and elaborated in the contributions in its succeeding section. It devotes in describing all the necessary steps required for further creating of test bench for optimization of the designed 32-bit FPAU system using the generated simulink model presented in Chapter V followed by the optimization process. The next section presents the reports of resources utilized and timing summary obtained after synthesization of optimized VHDL design of 32-bit FPAU using same ISE tool of Xilinx with the same FPGA platform/device as target for implementation which has been used for initial designed VHDL code of FPAU. It then presents the power estimation reports of optimized VHDL design of 32-bit FPAU using same version/release of XPower as has been used for initial designed VHDL code of FPAU. The results of initial VHDL design of 32-bit FPAU and the optimized VHDL code obtained after using the proposed approach in simulink in respect of resources of FPGA utilized, timing summary and estimated power consumed are compared at the end of this chapter.
6.1 Introduction

The study by Nanda, Yang and Markovic (2008) reveals that one can optimize HDL code to achieve speed-area objectives by employing distributed pipelining, streaming, and resource sharing in Matlab/Simulink. In Matlab, one can use advanced loop optimizations, such as loop streaming and loop unrolling, for a Matlab design containing for-loops or matrix operations. As described in Simulink HDL Coder User’s Guide (2006-2010), HDL Coder™ provides the following optimizations to explore the design space trade-offs between area and speed. One can use these options to explore various architectures and trade-offs without rewriting the algorithm manually.

Speed Optimizations

- **Pipelining**: To improve the design's clock frequency, HDL Coder enables to insert pipeline registers in various locations within the design. For example, one can insert registers at the design inputs and outputs, and also at the output of a given Matlab variable in the algorithm.

- **Distributed Pipelining**: HDL Coder also provides an optimization based on retiming to automatically move pipeline registers one has inserted to maximize clock frequency, by minimizing the delay through combinational paths in the design.

Area Optimizations

- **RAM mapping**: HDL Coder™ maps matrices to wires or registers in hardware. If persistent matrix variables are mapped to registers, they can take up a large amount of FPGA area. HDL Coder™ automatically maps persistent matrices to block RAM to improve area efficiency. The challenge in mapping Matlab matrices to block RAM is that block RAM in hardware typically has a limited set of read and write ports. HDL Coder™ solves this problem by automatically partitioning and scheduling the matrix reads and writes to honor the block RAM's port constraints, while still honoring the other control- and data-dependencies in the design.
• **Resource sharing**: This optimization identifies functionally equivalent multiplier operations in Matlab code and shares them. One can control the amount of multiplier sharing in the design.

• **Loop streaming**: A Matlab for-loop creates a FOR_GENERATE loop in VHDL. The body of the loop is replicated as many times in hardware as the number of loop iterations. This results in an inefficient use of area. The loop streaming optimization creates a single hardware instance of the loop body that is time-multiplexed across loop iterations.

• **Constant multiplier optimization**: This design level optimization converts constant multipliers into shift and add operations using Canonical Signed Digit (CSD) techniques.

The model of the VHDL design of 32-bit FPAU generated in simulink by using the proposed novel methodology in Chapter 5 has been further used to create the test bench for optimization using that generated simulink model. The following section describes the steps for this process along with snap shots of some steps.

6.2 Process of Generation of Optimized VHDL Code in Simulink

The step-wise process flow followed to generate the VHDL design/code of FPAU from simulink, its synthesization and estimation of total power consumption has been depicted in Figure 6.1 below:

After the simulink sub-system is created to launch HDL Simulator as shown in Figure 5.7 of last chapter, the following steps are practically used on the system for creation of test bench for optimization:

- **Test Bench and Code Generation**
  - Right click the ‘Subsystem Unit’. Select “PLC Code generation”
  - Select “Treat as atomic unit” for code generation. Apply and Press “OK”
Figure 6.1 Process Flow to generate the VHDL design/code of FPAU from simulink, its synthesization and estimation of total power consumption

- Click on ‘Simulation’ from main menu
- Select Configuration Parameters. A window shown in Figure 6.2 pops up
- In the Configuration Parameter Window, select “HDL Code Generation” pane
  - Target: Generate HDL for “Root level program(subsystem)” ‘subsystems created’: Language: VHDL
  - Folder: hdlsrc (folder in which VHDL code generated by MAT lab is stored) and “Apply”
- In ‘HDL Code Generation’ select ‘Test Bench’ pane as shown in Figure 6.3.
On Test bench generation output select cosimulation model for use with “Mentor Graphic’s Modelsim” and “Apply” “Generate Test Bench” on same window (This enables test bench pane)

- In “diagnostic” pane select
  - Sample time – single task rate transition as “Error” and multi task rate generation as “Error” and “Apply”
  - Automatic solver parameters selection as “None” and “Apply”

- On Solver pane
  - Solver option: Type as “Variable_step” and solver as “discrete” and “Apply”

(This creates Subsystem for optimization)

Now

- Right click ‘subsystem’
- Select ‘HDL code Generation’
- Opt ‘HDL workflow advisor’
- Go to “Tools” on main menu
  - Select ‘HDL Code Generation’ and Select ‘Generate Test Bench’

(This creates a test bench linking ‘Modelsim’ and ‘Simulink’ after optimization)

The System and sub-system created for Test Bench are shown in Figure 6.4&6.5.

![Figure 6.4 Test bench model generated in Simulink for Optimization](image-url)
Figure 6.5 The Sub-system created for the above model

- Double click ‘blue box’ to launch Modelsim “vsimulink work.subsystem”

“Optimized Code is also generated in VHDL.”

The Optimization pane details are shown in Figure 6.6 and 6.7.

Figure 6.6: Optimization pane for parameter setting

Figure 6.7: Optimization pane for signal parameter setting
The default values for the parameters set for optimization which appeared on the screen during the process are used.

6.3 Synthesis of Optimized VHDL Design of 32-bit FPAU and its Analysis

The optimized VHDL design of 32-bit FPAU generated with the above described process is synthesized using same tool i.e. Xilinx ISE 8.1 i with Virtex 4 (XC4V LX15 device) as target platform as has been used for initial VHDL design for same digital system of 32-bit FPAU. The Resource Utilization Summary and the Timing Summary for the optimized VHDL design of 32-bit FPAU captured are shown in Figure 6.8 and 6.9.

From the report of Resource Utilization Summary, the resources to be used for the target FPGA are analyzed and from Timing Summary, the reports in respect of delay (set up and hold time) are analyzed for the optimized VHDL design of FPAU.

Figure 6.8: Snap shot showing Device Utilization Summary of optimized VHDL design of 32 bit FPAU
Figure 6.9: Snap shot showing Timing Summary of optimized VHDL design of 32 bit FPAU

The analysis of the two snapshots of Resource Utilization Summary and the Timing Summary indicate the following resources utilized and set up and hold timings for the optimized VHDL design of 32-bit FPAU as tabulated below in Table 6.1 and 6.2 respectively:

Table 6.1 Resources Utilized for optimized VHDL design of 32 bit FPAU

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Utilized Resource</th>
<th>Total Resource</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>422 out of 6144</td>
<td>6144</td>
<td>6%</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>332 out of 12288</td>
<td>12288</td>
<td>2%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>781 out of 12288</td>
<td>12288</td>
<td>6%</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>102 out of 240</td>
<td>240</td>
<td>42%</td>
</tr>
<tr>
<td>Number of GCLKS</td>
<td>1 out of 32</td>
<td>32</td>
<td>3%</td>
</tr>
</tbody>
</table>

Table 6.2 Set up and Hold Time for optimized VHDL design of 32 bit FPAU

<table>
<thead>
<tr>
<th>Timing Parameter</th>
<th>Time Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Input Time before Clock (Setup Time)</td>
<td>23.523 ns</td>
</tr>
<tr>
<td>Maximum output required time after clock (Hold time)</td>
<td>3.935 ns</td>
</tr>
</tbody>
</table>

6.4 Estimation of Total Power of Optimized VHDL design of 32-bit FPAU

The Xilinx Xpower Estimator 11.1 is used to estimate the power consumed with the target device of Virtex-4 (XC4V LX15) after the synthesization of the optimized VHDL
design of 32 bit FPAU by using others parameters as default values. The summary sheet of the total estimated power as shown in the below snapshot in Figure 6.10 for optimized VHDL design of 32 bit FPAU is 164 mW.

Figure 6.10 Snapshot of the estimated Power for optimized VHDL design of 32 bit FPAU

6.5 Implementation and Testing of Optimized VHDL design of 32-bit FPAU on FPGA Device

The optimized FPGA-based digital system design using VHDL for 32-bit FPAU is tested for all its operations of additions, subtraction, multiplication and division by selecting different select line i.e. 01 for addition/subtraction, 10 for multiplication and 11 for division operation for various combinations of two inputs and results are successfully verified for all combinations on the above mentioned Xilinx FPGA Vertex 4 device (XC4V LX15).

All the results indicated in Table 4.3 in Chapter 4 for different operations of addition/subtraction, multiplication and division carried out by giving ‘01’, ‘10’ and ‘11’ inputs respectively for its select line for various combinations of two 32-bit floating point inputs for the initial design of 32-bit FPAU are verified for the optimized VHDL design for 32-bit FPAU and found O.K.
6.6 Comparison and Analysis of Results of Initial Digital Design and Optimized VHDL Digital Design of 32-bit FPAU

The results of Initial Digital Design and Optimized VHDL Digital Design of 32-bit FPAU are compared and analysed in this section.

6.6.1 Comparison of Results

By keeping Xilinx Synthesis tool ISE 8.1i, FPGA platform/device Vertex 4 device (XC4V LX15) and Power Estimator tool 11.1 of Xilinx same, the results of initial VHDL design of 32-bit FPAU and the optimized VHDL design obtained after using the proposed approach in simulink (Matlab) in respect of resources of FPGA utilized, timing summary and estimated power consumed are compared as obtained in Chapter 4 (Table 4.1 and Table 4.2) and this Chapter (Table 6.1 and Table 6.2). The comparison showing results for both initial and optimized design is tabulated in Table 6.3.

Table 6.3 Comparison of parameters of Initial and Optimized VHDL Digital Design of 32-bit FPAU

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Parameters</th>
<th>Initial VHDL Design of 32-bit FPAU</th>
<th>Optimized VHDL Design of 32-bit FPAU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of Slices</td>
<td>1185 out of 6144 19%</td>
<td>422 out of 6144 6%</td>
</tr>
<tr>
<td>2</td>
<td>Number of Slice Flip Flops</td>
<td>333 out of 12288 2%</td>
<td>332 out of 12288 2%</td>
</tr>
<tr>
<td>3</td>
<td>Number of 4 input LUTs</td>
<td>2175 out of 12288 17%</td>
<td>781 out of 12288 6%</td>
</tr>
<tr>
<td>4</td>
<td>Number of bonded IOBs</td>
<td>99 out of 240 41%</td>
<td>102 out of 240 42%</td>
</tr>
<tr>
<td>5</td>
<td>Number of GCLKs</td>
<td>3 out of 32 9%</td>
<td>1 out of 32 3%</td>
</tr>
<tr>
<td>6</td>
<td>Minimum input arrival time before clock (Setup Time)</td>
<td>48.274ns</td>
<td>23.523ns</td>
</tr>
<tr>
<td>7</td>
<td>Maximum output required time after clock (Hold time)</td>
<td>4.007ns</td>
<td>3.935ns</td>
</tr>
<tr>
<td>8</td>
<td>Estimated Total Power Consumed</td>
<td>167mW</td>
<td>164mW</td>
</tr>
</tbody>
</table>
The Bar Chart showing the comparison of resources of FPGA utilized in respect of number of slices, Number of Slice Flip Flops, Number of 4-input LUTs, Number of bonded IOBs and Number of GCLKs for initial design and optimized design of 32-bit FPAU has been depicted in Figure 6.11. The Bar Chart showing the comparison Minimum input arrival time before clock (Set-up Time) and Maximum output required time after clock(Hold time) for initial design and optimized design of 32-bit FPAU has been depicted in Figure 6.12. The Bar Chart showing the comparison of total estimated power consumed for initial design and optimized design of 32-bit FPAU has been depicted in Figure 6.13.

Figure 6.11: Bar Chart showing the comparison of resources of FPGA utilized for initial design and optimized design of 32-bit FPAU

Figure 6.12: Bar Chart showing the comparison of Set-up and Hold time for initial design and optimized design of 32-bit FPAU
6.6.2 Analysis of Results

The comparative reports of resource utilization summary, timing summary and total estimated power consumed for the initial VHDL design and optimized VHDL designed obtained after applying the proposed novel approach and methodology for base digital system designed for 32-bit FPAU are elaborated and concluded as follows:

A) Resource Utilization Summary Reports:

- 1185 number of slices utilized out of 6144 total slices used for initial FPGA-based VHDL design of digital system of 32-bit FPAU has reduced to 422 for the optimized VHDL design of same system. There is reduction of 13%.
- 333 number of slice flip flops utilized out of 12288 total slice flip flops used for initial FPGA-based VHDL design of digital system of 32-bit FPAU has reduced to 332 for the optimized VHDL design of same system. There is almost no change.
- 2175 number of 4-input LUTs utilized out of 12288 total 4-input LUTs used for initial FPGA-based VHDL design of digital system of 32-bit FPAU has reduced to 781 for the optimized VHDL design of same system. There is reduction of 11%.
- 99 number of bonded IOBs utilized out of 240 total 4-input LUTs used for initial FPGA-based VHDL design of digital system of 32-bit FPAU has increase to 102
for the optimized VHDL design of same system. There is an increase by almost 1%.

- 3 number of Global Clocks (GCLKs) utilized out of 32 total 4-input LUTs used for initial FPGA-based VHDL design of digital system of 32-bit FPAU has reduced to 1 for the optimized VHDL design of same system. There is reduction of 6%.

B) Timing Summary Report

- Minimum input arrival time before clock i.e. Set-up Time of 48.274ns for initial FPGA-based VHDL design of digital system of 32-bit FPAU has reduced to 23.523 ns. It means the Set-up Time for the clock has decreased and hence improved by 24.751ns.
- Maximum output required time after clock i.e. Hold Time of 4.007 ns for initial FPGA-based VHDL design of digital system of 32-bit FPAU has reduced to 3.935 ns. It means the Hold Time for the clock has decreased and hence improved by 0.027ns.

C) Estimated Total Power Consumption Report

- Estimated Total Power Consumption of 167 mW for initial FPGA-based VHDL design of digital system of 32-bit FPAU has reduced to 164 mW. Total power consumption has also reduced by 3 mW.

6.7 Conclusions

Test bench created for optimization using the generated simulink model in the Chapter V has been used for generation of optimized VHDL code for the base digital system designed for 32-bit FPAU by up-loading the initial VHDL design of that system in simulink. The default values for the parameters set for optimization which appeared on the screen during the process have been used. The optimized VHDL design so obtained has been synthesized using same Xilinx ISE 8.1i tool with Vertex-4 device (XC4VLX15) as target FPGA and the estimated total power consumed has been calculated using
power estimator Xilinx 11.1. The optimized VHDL code/design has also been implemented and tested on FPGA Xilinx Virtex-4; XC4V LX15 and results have been verified for same inputs as were applied for initial VHDL design and tabulated in Table 4.3.

From the analysis of the reports it is clear that almost all the resources of the FPGA have reduced and hence it can very well be concluded that the area utilized has reduced/improved substantially by using this approach/methodology. As there is an improvement in both Set-up and Hold time for the clock, it means the overall delay has also reduced substantially and the total power consumption has also reduced by 3 mW.