CHAPTER-IV

IMPLEMENTATION AND ANALYSIS OF FPGA-BASED
DESIGN OF 32-BIT FPAU

The design of 32 bit FPAU using VHDL presented in chapter III needs to be further implemented and tested on FPGA platform and also to be analyzed in respect of its resource utilization, timing summary and power consumed. As described in the last chapter about the development stages of FPGA systems, the HDL design of the digital system first required to be synthesized, simulate and then to configure the target FPGA device. For this purpose vendors of FPGAs have developed many software tools.

This chapter first summarizes the flow of design methodology of FPGA and then presents the complete work flow of further research work. It describes the brief about the software tools developed by Xilinx and are used for synthesis and analysis of our digital system of 32 bit FPAU which has been presented in Chapter III. It also presents the reports of synthesized design through some of the snap shorts. It then describes the power estimator tool used to estimate the power consumed by our designed FPGA-based digital system. All the reports of the designed system in respect of resource utilization, timing summary and estimated power consumed are demonstrated at the end of this chapter.

4.1 Introduction

In the last chapter it has been described that the first step in FPGA design methodology is to capture the algorithm to be implemented on FPGA using hardware description languages (HDLs) or schematic depending on the complexity of the design. The flow of design methodology is shown Figure 4.1 and it shows that after specifying the design using HDLs or Schematic, the designer needs to validate the logical correctness of the design. This is performed using functional or behavioral simulation. Simulation describes the behavior of the circuit in terms of input signals, output signals and knowledge of delays. Designers usually go through this step right after they finish the coding and logic
synthesis. Synthesis is the reverse process of inferring the hardware from description i.e. logic synthesis converts HDL or schematic-based design into a netlist of actual gates/blocks specified in FPGA devices. Simulation and synthesis are two complementary processes. After logic synthesis, technology mapping is done. In this step, the tool transforms a netlist of technology independent logic gates into one consisting of logic cells and input/output blocks (IOBs) in the target FPGA architectures [Manohararajah, Brown and Vranesic (2006) and Mishchenko, Chatterjee and Brayton (2007)].

Figure 4.1 Flow of steps for FPGA Design Methodology

Placement which follows technology mapping selects the optimal position for each block in a circuit. The basic goal of an FPGA placement is to locate functional blocks such that
the interconnect required to route the signals between them is minimized. A good placement is extremely important for FPGA designs. It directly affects the routability and the performance of a design on FPGA [Mak and Hao (2005)]. A poor placement will lead to lower maximum operating speed and increased power consumption. FPGA placement algorithms can be broadly classified as routability-driven and timing-driven [Marquardt, Betz and Rose (2000)]. The main objective of routability-driven algorithms is to create a placement that minimizes the total interconnect required. In addition to optimizing for routability, timing algorithms use timing analysis to identify critical paths and/or connections and optimize the delay of those connections. Routing is the last step in the design methodology prior to generating the bitstream to program the FPGA [Alexander and Robins (1996)]. FPGA routing is a tedious process because it has to use only the prefabricated routing resources such as wire segments, programmable switches and multiplexers [Nam et al (2004)]. Hence it is a challenging task to achieve 100% routability in FPGAs. After placement and routing, timing simulation is performed to validate the logical correctness of the design taking into account the delays of the FPGA device. Power consumed by a design is further estimated by doing the power analysis such as XPowe and PowerPlay tools used in XilinxISE and Altera Quartus II tools respectively. The final step in the FPGA design methodology is bitstream generation. It takes the mapped, placed and routed design as input and generates the necessary bitstream to program the logic and interconnects to implement the intended logic design and layout on the target device.

4.2 Work-Flow of Further Research Work

The complete work-flow of the further carried out research work is depicted in Figure 4.2 below:

The step-wise contribution made towards the research work is put up here below:

1. Designing/Writing VHDL code of 32-bit Floating Point Arithmetic Unit
2. Implementing the Code on target platform after its synthesis using Xilinx ISE 8.1.
4. Estimating Power using Xilinx XPowe Estimator XPE 11.1 Tool
5. Creating Simulink model for verification of VHDL code in Modelsim
6. Simulation of 32-bit Floating Point Arithmetic Unit in Modelsim wave window of Matlab.

7. Creating Test Bench Model in Simulink to generate the optimized VHDL code for design of 32-bit FPAU.
8. Implementing the Optimized Code on same target platform after its synthesis

Work related to step at 1 i.e. designing of 32-bit FPAU has been presented in the last chapter and work related to steps from 2 to 4 presented in this chapter.
4.3 Synthesis of VHDL Design of 32-bit FPAU

Synthesis is the process of generating circuit/gate level implementations from VHDL model with the inputs as VHDL model, design constraints, mapping libraries etc. It converts the design into a netlist of actual gates/blocks specified in FPGA devices. Behavioral synthesis refers to a collection of tasks like operation scheduling, resource binding, control generation etc. that taken together produce the final implementation and results are often described in ‘structural RTL’ description, whereas RTL synthesis is primarily a process of inference and optimization.

Vendors of FPGAs have developed many software tools. For the synthesis of our 32-bit FPAU, Xilinx ISE 8.1i release and version has been used with Virtex 4 (XC4V LX15 device) as target platform. Xilinx ISE (Integrated Software Environment) is a software tool that has been developed by Xilinx for synthesis and analysis of HDL designs, enabling the developer to compile, synthesize their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. The following section describes the salient features and enhancements of this version/release over the previous versions of Xilinx ISE software tools.

4.3.1 About Xilinx ISE 8.1i

For the synthesis of our VHDL design of 32-bit FPAU, Xilinx ISE 8.1i has been used that supports all devices in Virtex 4 LX, 4FX and 4SX along with Spartan 3E family. This release has the following new features in-built [Libraries Guide of Xilinx ISE 8.1i]:

- **Power Optimization**

  Power optimization is available through the Place and Route and can reduce dynamic power in Virtex™-II Pro designs by up to 15 percent.

- **Power Estimation**

  Following are the power estimation enhancements:
- **Web Power**
  o Worst-case process power estimation based on voltage variation, temperature variation, and process variation.
  o Detailed input information — One can select header cells to view detailed information on each input cell. The information provides guidance to improve power estimation accuracy.

- **XPower**
  XPower tracks the changing effect of temperature on quiescent power. When one sets exact temperature conditions, XPower reflects quiescent power at the set temperature

- **ISE Project Navigator**

  Following are the ISE Project Navigator enhancements:

  - **HDL mixed language support enhancements**
    o Full support for projects containing mixed VHDL, and Verilog, and for synthesis and simulation tools that support mixed language
    o Ability to set target language for generated HDL files on a per-process basis

  - **Improved simulation environment support**
    o Simulation-centric project view
    o Support for simulation-only HDL sources in Simulation view
    o Hierarchical test bench support
    o Improved support for multiple hierarchical netlist files generated through Post- Simulation Netlist processes.

  - **ISE Project Navigator Sources tab enhancements**
    o Ability to set top module prevents accidental implementation of lower-level modules
      ▪ Right-click source and select Set as Top Module or select Process > Set as Top Module
      ▪ Icon indicates top module
• Process > Implement Top Module or corresponding toolbar button runs the implementation process on the top module
  o Instance-based sources tab accurately represents hierarchies through multiple instances of the same module
• Multi-Pass Place & Route (MPPR) process available through standard Place and Route process properties
• Ability to view Process Properties while a process is running

• Design Entry

Following are the design entry enhancements:

• Architecture Wizard

  The clocking wizard, available from the Architecture Wizard, now supports new jitter algorithms for Virtex-II Pro and Virtex-4 devices.
• Schematic and Symbol Editors
  o New Library Manager enables to create and manage custom symbol libraries
  o Design Rule Check (DRC) messages, Errors, and Warnings are posted to the Transcript Window
  o New, dedicated color scheme for printing
• ISE Text Editor
  o Zoom In and Out
  o Ability to convert tabs to spaces, and vice-versa, convert end-of-line characters to other EOL types.
  o Optional display of end-of-line characters, white space symbology, vertical indentation guide lines, vertical "long line" marker etc.
  o When interacting with ISE Simulator, the "current line" marker is updated when text lines are added or removed by the user
• RTL and Technology Viewers
The hierarchy browser has been replaced by new View by Category and View by Name tabs, found in the Transcript window, allowing easier traversal of the design hierarchy.

- Placement and routing of RTL and Technology views is now much faster

**XST Synthesis**

Following are the Xilinx Synthesis Technology (XST) enhancements. For more information, see the *XST User Guide*, available from the Software Manuals collection of Xilinx.

- VHDL language support
  - New file write capability
- Verilog language support
  - New file read and write capability
  - Support for $display and $write assertions
  - Support for recursive tasks and functions using "automatic" keyword
  - Support for $signed and $unsigned keywords
- XST Constraints
  - New DSP utilization ratio constraint (DSP_UTILIZATION_RATIO) to specify the number of available DSP blocks for synthesis
  - New pipeline distributed RAMs value (PIPE_DISTRIBUTED) for RAM Style (RAM_STYLE) constraint
- XST Reporting
  - More compact and informative messages for equivalent register detection and removal
  - Structure of LOG file at the Advanced HDL Synthesis step matches the structure of the LOG file at the HDL Synthesis step
  - Improved HDL Advisor reports when a latch is inferred from an incomplete case statement
• Implementation

Following are the implementation enhancements:

- Xplorer (Powered by ISE Fmax Technology)
  
  o Xplorer is a perl script that seeks the best design performance using ISE. Xplorer has been developed to help achieve optimal results for designs by employing smart constraining techniques and a variety of physical optimization strategies. Because no unique set of ISE options or timing constraints work best on all designs, Xplorer works to explore the right set of tools options to either meet design constraints or to find the best performance for the design. For additional information, see http://www.xilinx.com/xplorer.

- Place and Route (Powered by ISE Fmax Technology)
  
  o Improved performance and runtime for PAR tool
  o Automatic performance evaluation capability in which PAR automatically seeks to produce high clock rates with a fast runtime in the absence of timing constraints.

- Constraints
  
  o Clock skew calculations can be switched off when using the FROM-TO constraint. This allows the ability to specify and analyze the data path delays in isolation from the affects of clock path skew. This feature is most useful in analyzing asynchronous cross-clock domain paths.

- Timing Analysis
  
  o The software analyzes the timing of input and output paths as a group. In addition to normal setup and hold checks, the software recommends the appropriate data path and clock phase adjustments to center the clock across the data bus.
  
  o The timing analysis software can enable the analysis of a registered components' set/reset to output and set/reset recovery time separately. This enables to better isolate the asynchronous reset behavior of interest.
• **Reporting**

Following are the reporting enhancements.

- **Design Summary includes report viewer:**
  - Configurable and sortable display of all software messages
  - Capability to identify messages that are new since the last design run
  - Most useful design information shown in customizable tables

- **Message Filtering**
  - Filters can be edited to contain wildcards for more powerful filtering choices.
  - Filtering ability is enabled by default.

- **Map and PAR timing summaries**
  - Now sorted by slack, most negative to most positive.
  - Two new columns, "Absolute Slack" and "Number of Errors"

- **PAR Report**
  - New line that lists the total number of constraints that did not cover any paths.
  - Names of un-routed are listed.
    - Note: The PAR report shows up to 25 un-routed nets. For more than 25 un-routed nets, then use reportgen.
  - PAR reports on all directed routing constraints. Both exact and non-exact DIRT constraints are reported. Any net with a DIRT constraint that is not implemented as constrained will be reported by name in the PAR report as a failure.
  - Reportgen can create the optional Clock Region Utilization report giving information on resource use by clock region.
• Verification

Following are the verification enhancements:

- The Input Output Buffer Information (IBIS) Model writer now sets the direction of user signals in IBIS model. Designers no longer have to manually edit the IBIS file to set individual signal direction.

- ModelSim Xilinx Edition III is upgraded to version 6.0d.

- Timing Simulation
  o The timing simulation model is improved to include advanced timing analysis features previously only available in static timing analysis tools. These features include relative minimum delays for a more accurate I/O timing, accounting for clock and system jitter, and advanced clock tree skew modeling.
  o The post-place and route netlists include location information. This allows you to better trace simulation failures back to the physical design.
  o The simulation netlist supports the boundary scan component for Virtex-4 devices.

- Simulation Libraries
The Xilinx simulation libraries are enhanced to increase the speed of simulation of both behavioral and structural models. The simulation speed enhancements have focused on the clock subsystem and slice look up tables and registers.

- Waveform Editor
Waveform Editor's Pattern Wizard can now generate patterns for single signals

- ISE Simulator
  o Support for mixed VHDL/Verilog designs
  o User-defined signal ordering is retained between invocations of the simulator
  o Support of conversion function and type conversion for VHDL component instances
  o Generation of the new XAD format, which allows XPower to read "value change dump" data generated by ISE Simulator much faster
The following five recommendations reduce the time needed to achieve timing closure of the design:

1. Use the default settings without timing constraints for initial run.

In the absence of any user timing constraints, the 8.1i implementation tools enter into Performance Evaluation Mode (PEM). By using the default settings, PEM evaluates all clocks in the design during implementation and dynamically constrains them. The result of using PEM is a balance of good performance for each clock in a reasonable run time.

To ensure that design runs without timing constraints, either do not add timing constraints in to the User Constraint File (UCF), or instruct ISE to ignore all timing constraints by deselecting the Use Timing Constraints property as follows.

- In the Sources tab, select a UCF file.
- In the Processes tab, Right-click User Constraints and select Properties.
- In the User Constraints Properties dialog box, set the Use Timing Constraints property to False (checkbox is unchecked).

2. Use the default settings plus Place and Route Effort Level HIGH.

Running PEM with the Place and Route Effort Level property set to High can significantly improve the performance for each clock but the run time will increase substantially.

To set the Place and Route Effort Level property in ISE, in the Sources tab select the top module, and in the Processes tab, expand Implement Design.

In the Right-click Place & Route and select Properties.

In the Place and Route Properties dialog box, set the Place and Route Effort Level property to High.

If the performance achieved by PEM is slightly below the required performance for the design, consider the next recommendation.

3. Apply Timing Constraints.

A design with user-provided timing constraints has an advantage over designs without timing constraints. Providing timing constraints tells the tools to stop when
the timing requirements have been achieved. Timing constraints can also provide information on clock relationships and the relative importance of each clock and/or path through the timing requirement.

Providing timing constraints can be a disadvantage if they are far beyond what can be achieved. Unrealistic constraints, which can lead to poor results and long run times, can be avoided by following recommendation 1 and 2. Over constraining a design is not recommended for improving performance in Xilinx implementation tools.

4. Use Xplorer Best Performance Mode

In Best Performance Mode, Xplorer optimizes the performance for the one-clock domain specified by the user. Xplorer runs multiple iterations with different optimization strategies to seek out the best performance. Xplorer implements the design using timing driven techniques to tighten or relax the timing target, depending on whether the frequency goal is achieved. Using this method will determine the practical performance limit for the specified clock in the design.

5. Use Xplorer Timing Closure Mode.

The Xplorer Timing Closure Mode operates on designs with the timing targets in the UCF. Normally, Timing Closure Mode is used for designs that did not meet the timing targets using standard implementation options. Xplorer uses multiple tested optimization strategies, such as Global Optimization, Timing-Driven Packing and Placement, Register Duplication, and Cost Tables to deliver optimal design performance.

4.3.2 Reports of Synthesis

A 32-bit FPAU designed system is successfully compiled and synthesized using Xilinx ISE 8.1i with Virtex 4 (XC4V LX15 device) as target platform.

Some of the snapshots of reports of 32-bit FPAU designed system after successful synthesization in respect of primary inputs and outputs for the FPAU, Internal Blocks in FPAU, Internal Structure of FPAU, LUT’s and IO blocks (i) and (ii), internal gate level
design for a particular LUT, truth table for a particular LUT and K-map of a particular LUT block are depicted below in Figures 4.3, 4.4, 4.5, 4.6 (a), 4.6 (b), 4.7, 4.8 and 4.9 respectively.

Figure 4.3: Snap-shot showing primary inputs and outputs for the FPU

Figure 4.4: Snapshot showing Internal Blocks in FPAU
Figure 4.5: Snapshot showing Internal Structure of FPAU

Figure 4.6 (a): Snap shot showing LUT’s and IO blocks (i)
Figure 4.6 (b): Snap shot showing LUT’s and IO blocks (ii)

Figure 4.7: Snap shot showing internal gate level design for a particular LUT
Figure 4.8: Snap Shot showing truth table for a particular LUT

Figure 4.9: Snap shot showing K-map of a particular LUT block
The Resource Utilization Summary and the Timing Summary for the designed system of 32-bit FPAU captured are shown in Figure 4.10 and 4.11.

Figure 4.10: Snap shot showing Device Utilization Summary of initially designed 32 bit FPAU

Figure 4.11: Snap shot showing Timing Summary of initially designed 32 bit FPAU
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 analysed. The analysis of the two snapshots of two above mentioned Resource Utilization Summary and the Timing Summary indicate the following resources utilized and set up and hold timings for the designed 32-bit FPAU as tabulated below in Table 4.1 and 4.2 respectively:

<table>
<thead>
<tr>
<th>Table 4.1 Resources Utilized for initially designed 32-bit FPAU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
</tr>
<tr>
<td>Number of GCLKS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4.2 Set up and Hold Time for initially designed 32-bit FPAU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Input Time before Clock (Setup Time)</td>
</tr>
<tr>
<td>Maximum output required time after clock (Hold time)</td>
</tr>
</tbody>
</table>

4.4 Estimation of Total Power Consumed

The XPower Estimator (XPE) spreadsheet is a power estimation tool typically used in the pre-design and pre-implementation phases of a project. XPE assists with architecture evaluation, device selection, appropriate power supply components, and thermal management components specific for particular application.

XPE considers the design’s resource usage, toggle rates, I/O loading, and many other factors which it combines with the device models to calculate the estimated power distribution. The device models are extracted from measurements, simulation, and/or extrapolation. XPE is a spreadsheet, so all Microsoft Excel functionality is fully retained in the writable or unprotected sections of the spreadsheet. XPE has additional
functionality oriented to ease of use. The drop-down menus and the comment-enabled cells are helpful features to inform and guide the user.

XPE requires a licensed version of Microsoft Excel 2003 or Microsoft Excel 2007 to be installed. Download the latest available spreadsheet for your targeted device. Make sure your Microsoft Excel settings allow macro executions. XPE uses several macros built into the XPE spreadsheet.

XPE uses the design and environmental input and then combines this information with the device data model to compute and present an estimated distribution of the power in the targeted device.

XPE presents multiple views of the power distribution [White Paper of Xilinx (2008)].

- **Power by Voltage Supplies** - For each required voltage source, this information is useful to select and size power supply components such as regulators, etc. Supply power includes both off-chip and on-chip dissipated power.

- **Power by User Logic Resources** - For each type of user logic in the design, XPE reports the expected power. This allows you to experiment with architecture, resources, and implementation trade-off choices in order to remain within the allotted power budget.

- **Thermal Power** - XPE lets you enter device environment settings and reports thermal properties of the device for your application, such as the expected junction temperature. With this information you can evaluate the need for passive or active cooling for your design.

The Summary sheet in XPE shows the total power for the device. Other sheets show usage-based power. The total FPGA power is calculated as follows:

**Total FPGA power = Device Static + Design Static + Design Dynamic**

Device Static Power is also referred to as Leakage Power. It represents the transistor leakage power when the device is powered and not configured.
Design Static Power represents the additional power consumption when the device is configured and there is no switching activity. It includes static power in I/O DCI terminations, clock managers, etc. For design static power calculations, XPE starts by assuming a blank bitstream. To "instantiate" design elements for the design static power calculations, you must enter the appropriate resource counts on the sheets with count fields and non-zero clock frequencies for the sheets without count fields. I/O termination must be set to match the board and the design.

Design Dynamic Power represents the additional power consumption from the user logic utilization and switching activity.

The following seven basic steps to an Accurate Power Estimation are recommended in [9]:

- Obtain the latest version of Xilinx Power Estimator for the selected target device
- Complete the Device information on the Summary tab
- Complete the Thermal Information on the Summary tab
- Set worst-case voltages for all supplies
- Enter clock and resource information
- Set the toggle and connectivity parameters.
- Analyze the results

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 designed 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 4.12 for our designed system is 167 mW.
Figure 4.12 Snapshot of the estimated Power for the initial designed system of 32-bit FPAU

4.5 Implementation and Testing of 32-bit FPAU on FPGA Device

As mentioned above Xilinx ISE8.1 has been used as synthesis tool for 32-bit FPAU. After successful compilation and generation of the synthesis report, the file fp_alu of the VHDL design of 32-bit FPAU is loaded on to the targeted FPGA Vertex 4 device (XC4VLX15) for its practical implementation and testing.

The FPGA-based digital system designed for 32-bit FPAU is tested for all its operations of additions, subtraction, multiplication and division by selecting different select line i.e. 00 for addition/subtraction, 01 for division and 10 for multiplication operation for various combinations of two inputs and results are successfully verified for all combinations on the above mentioned Xilinx FPGA Vertex 4 device (XC4VLX15).

Table 4.3 below shows the results of 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.
Table 4.3: Results of designed 32-bit FPAU on XC4V LX15

<table>
<thead>
<tr>
<th>Input A</th>
<th>Input B</th>
<th>Operation</th>
<th>Select Line</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F8CCCCCDH</td>
<td>3F8CCCCCDH</td>
<td>Add/ Subtract</td>
<td>01</td>
<td>404CCCCCDH</td>
</tr>
<tr>
<td>3F8CCCCCDH</td>
<td>3F8CCCCCDH</td>
<td>Multiplication</td>
<td>10</td>
<td>3F9AE148H</td>
</tr>
<tr>
<td>3F8CCCCCDH</td>
<td>3F8CCCCCDH</td>
<td>Division</td>
<td>11</td>
<td>3F06BCA1H</td>
</tr>
<tr>
<td>40200000H</td>
<td>40980000H</td>
<td>Add/ Subtract</td>
<td>01</td>
<td>41440000H</td>
</tr>
<tr>
<td>40200000H</td>
<td>40980000H</td>
<td>Multiplication</td>
<td>10</td>
<td>413E0000H</td>
</tr>
<tr>
<td>40200000H</td>
<td>32'h40980000H</td>
<td>Division</td>
<td>11</td>
<td>3F800000H</td>
</tr>
</tbody>
</table>

4.6 Conclusions

The complete synthesization process using Xilinx ISE 8.1i for the designed digital system of 32-bit FPAU has been presented in this chapter followed by the estimation of the power consumed by the designed system using Xilinx Xpower Estimator 11.1. The designed system of FPAU has also been implemented, tested and verified for all its operations on the real FPGA platform of Xilinx Vertex-4 (XC4V LX15). The summary of the resources of target FPGA utilized, timing summary along with total estimated power for this initial designed system have been recorded for their further comparison with the same reports for the optimized designed system of this 32-bit FPAU that will result after applying the novel approach described in the next chapters. Next chapter devotes to explain and describe the very novel approach of linking the Modelsim and Simulink of Matlab for up-loading the VHDL design and then verifying the all arithmetic operations of addition/subtraction, division and multiplication by simulation in Modelsim wave window of Matlab after creating the simulink model and subsystem.