Part III

IC Design and Simulation Tutorials

C

Appendix C: AMD Vivado Tutorial

In Chapter 2, we introduce the industry IC design flow. Although this book primarily focuses on the RTL design and simulation aspects of the IC design flow, it is important to establish a connection between RTL design and real hardware to write effective Verilog code. To achieve this, we recommend using the synthesis tool AMD Vivado to check the equivalence between the design block diagram and the RTL analysis results. Additionally, AMD Vivado can estimate the resource cost of FPGA designs.

As mentioned in Chapter 1, the elaborated design schematic provided by AMD Vivado differs from the ASIC gate-level netlist. However, it can help users establish a connection between Verilog code and hardware components, and a student free-license version of the software is available. In this tutorial, we present seven steps for using AMD Vivado, from creating a project to the final netlist programming.

1. Step 1: Create a new project.
2. Step 2: Add the design source code and constraint file.
3. Step 3: Run simulation.
4. Step 4: Run RTL analysis.
5. Step 5: Run synthesis.
6. Step 6: Run implementation.
7. Step 7: Program the netlist.

We use the ddot design project, which was introduced in Chapter **??**, as an example for this tutorial.

Step 1: Create a new project.

To create a new project, follow these steps:

1. Click File-Project-New (shown Fig. C.1). And then click Next to display the Project Name window.
2. Name the project (default "project_1") in the "Project name" box and navigate to the **syn** folder in the "Project location" box (as shown Fig. C.2). Click Next to go to the Project Type window.

3. Select RTL project and click Next to proceed to the Add Sources Files window (as shown Fig. C.3).
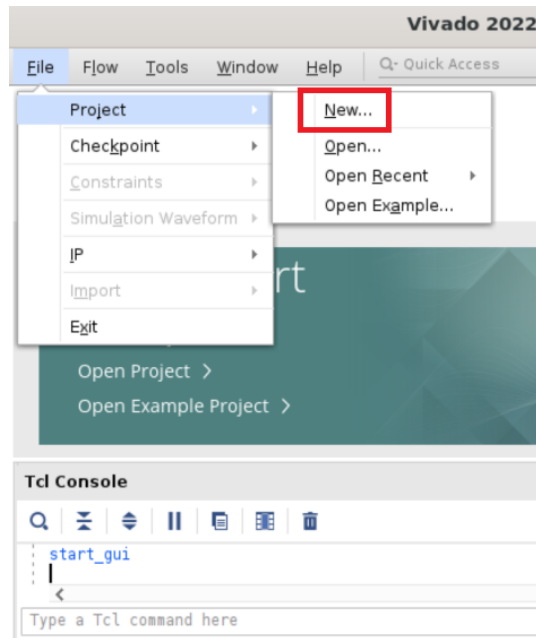
FIGURE C.1
Step 1.1: Create a new project
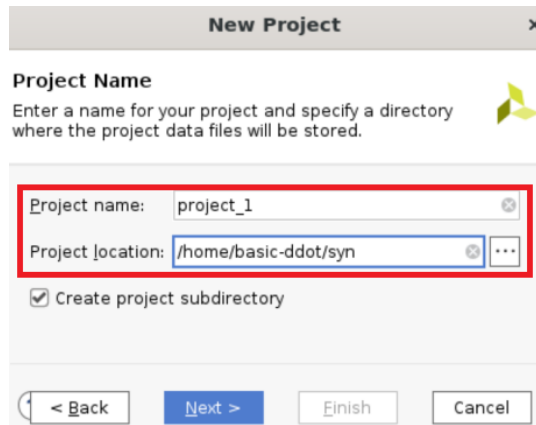
FIGURE C.2
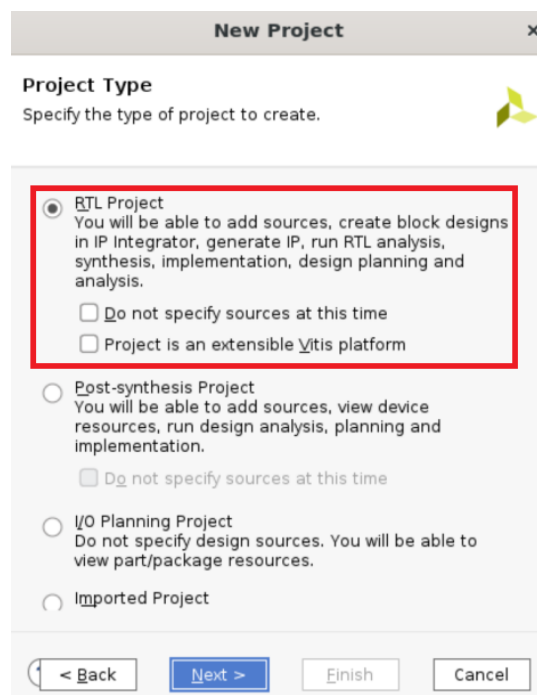Step 1.2: Direct to the **syn** folder

FIGURE C.3
Step 1.3: Select RTL project

Step 2: Add the design source code and the constraint file.

In the Add Sources Files window, as shown in Fig. C.4, you need to find and select all the design files in the **../dut** folder. Since the project is located in the **syn** folder, you need to return to the parent folder and then enter the **dut** folder to find the design sources. Once you have selected all the necessary design files, click OK to add them to the project.
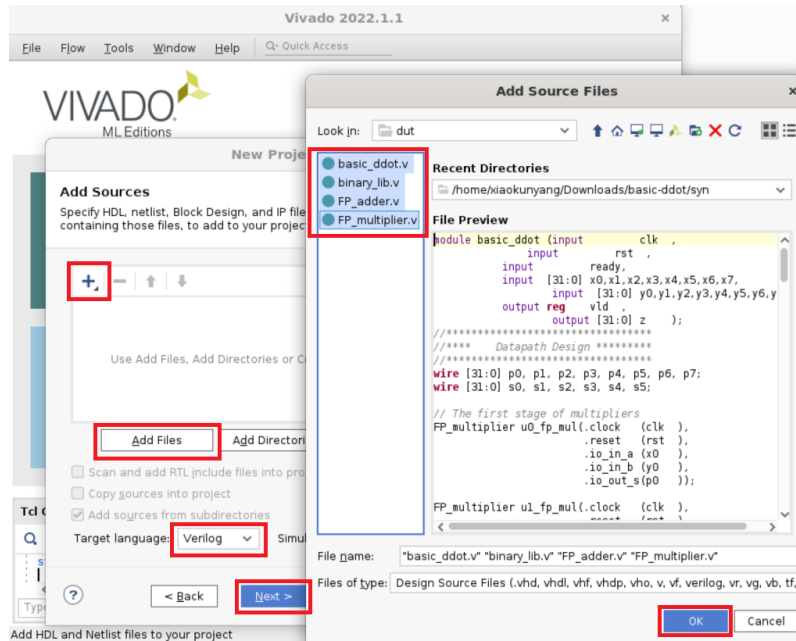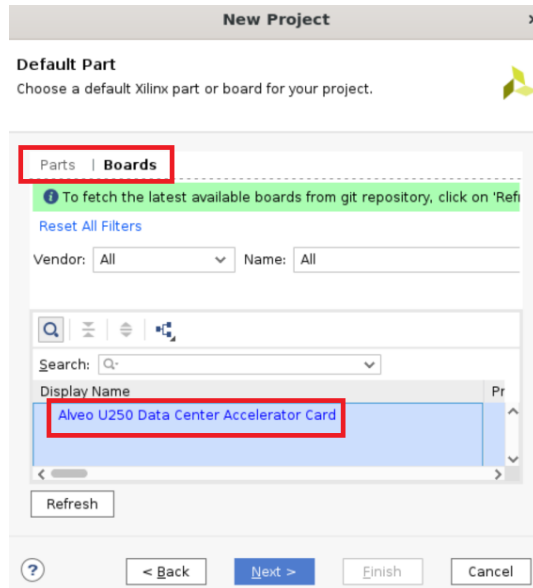

FIGURE C.4
Step 2.1: Add all the design files.

After adding the design files, click Next to go to the Add Constraints window. If you don't need to demonstrate your FPGA design, you can skip adding the constraint file and move on to the Default Part window.

In the Default Part window, as shown in Fig. C.5, you need to select the specific FPGA part. You can select the FPGA part using the Parts or Boards panel. For example, if you're using the Alveo U250 Data Center Accelerator Card, click Boards and select the specific board in the Display Name list. Then, click Next to go to the New Project Summary window. Check the summary and click Finish if no updates are needed.

In summary, adding the design files and selecting the appropriate FPGA part are important steps in creating a new project in Vivado.

FIGURE C.5
Step 2.3: Select the FPGA part.

Step 3: Run simulation.

In what follows, we will build the "project_1" as shown in Fig. C.6. The Flow Navigator panel shows the steps involved in the project creation, including Simulation, RTL Analysis, Synthesis, Implementation, and finally, Programming and Debugging. The Sources panel shows that the design hierarchy begins with the top design module, "basic_ddot". Note that in the simulation sources, the top module is also "basic_ddot" since the testbench has not been added yet. The Project Summary panel provides information about the project name, location, and part. The Design Runs panel indicates that the "synth_1" and "impl_1" projects have not yet been started.

Running simulation in Vivado is not necessary as the simulation was already carried out using Siemens Modelsim. However, to illustrate the differences between different simulators, we will add the testbench located in the **../tb** folder. Click the plus button and then select Add simulation sources. Since the testbench is for simulation purposes only, it must be added as simulation sources, not design sources.

Fig. C.7 illustrates how to add the testbench. The "tb_basic_ddot" file is located in the **../tb** folder. Click OK and then Finish to add the testbench to the simulation sources.

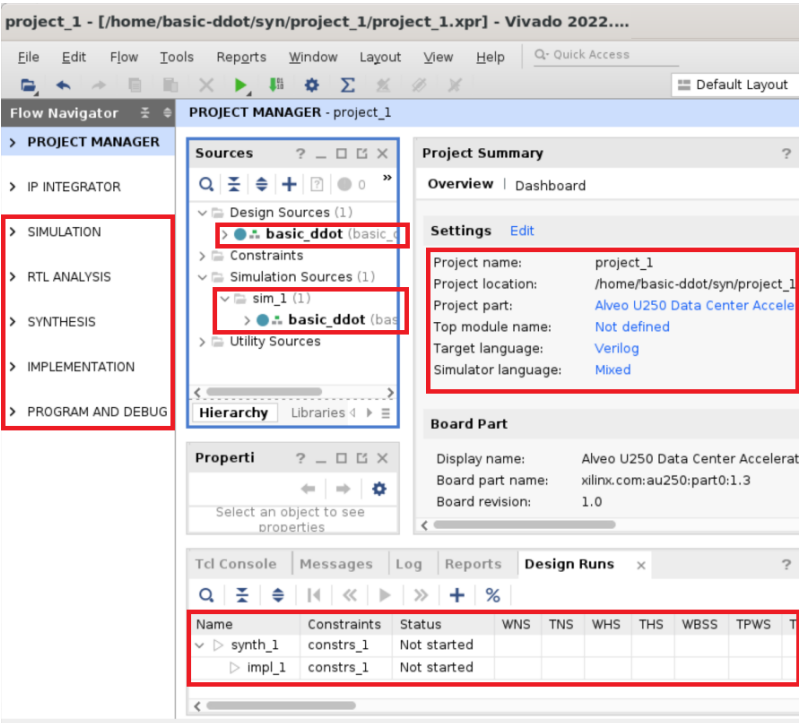Fig. C.8 shows the updated simulation hierarchy after adding the test-

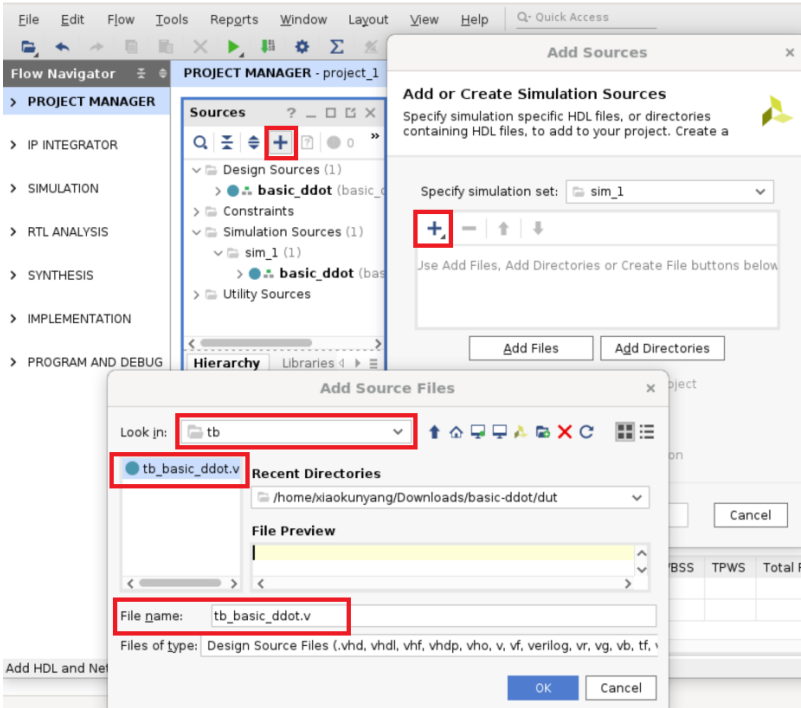**FIGURE C.6**
Step 3.1: Design hierarchy.

FIGURE C.7
Step 3.2: Add testbench.

bench. The unique name of the design, "u_basic_ddto", is under the test-
bench, "tb_basic_ddot". Click Run Simulation to begin the simulation.
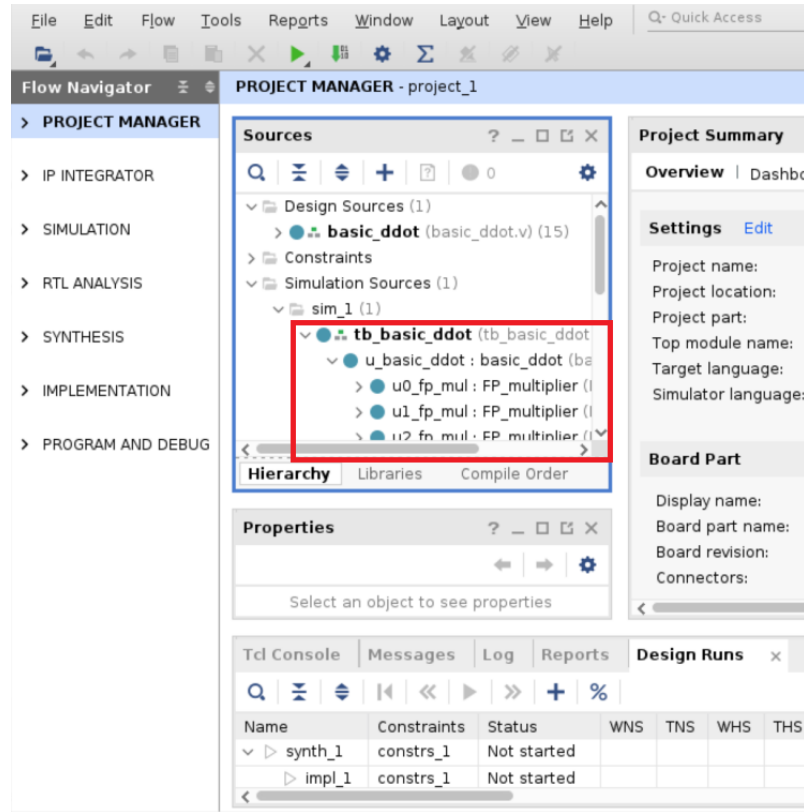

FIGURE C.8
Step 3.3: Simulation hierarchy.

Fig. C.9 shows the simulation result in the Scope panel. The waveform is
very similar to the waveform obtained from ModelSim. In this test case, the
eight inputs, "x0-x7", are all hexadecimal 0x3f800000 (FP 1.0), and the eight
inputs, "y0-y7", are all hexadecimal 0x40000000 (FP 2.0). Therefore, the ddot
result should be hexadecimal 0x41800000 (FP 16.0), which is shown precisely
in the simulation waveform. Note that the "vld" signal is asserted when the
valid output, "z", is pushed out. By comparing the simulation result with our
expected timing diagram designs shown in Fig. **??** in Chapter **??**, we can
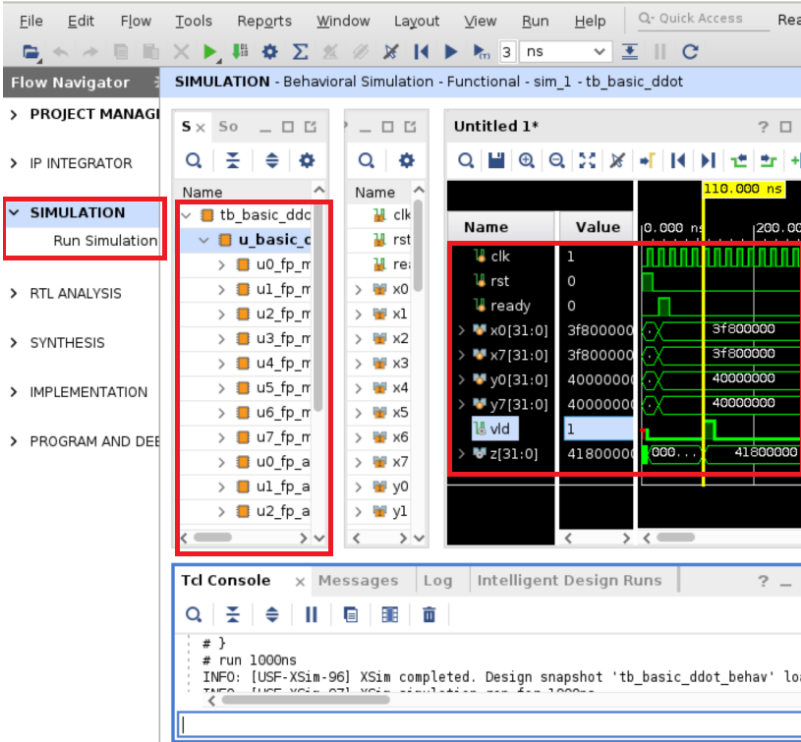ensure that our design features has been correctly verified.

FIGURE C.9
Step 3.4: Run simulation.

Step 4: Run RTL analysis.

Showing the RTL analysis result is a useful step in Vivado, as it can help us verify that our Verilog code has been correctly synthesized into hardware. To view the RTL analysis schematic, we can click on Open Elaborated Design, as shown in Fig. C.10.
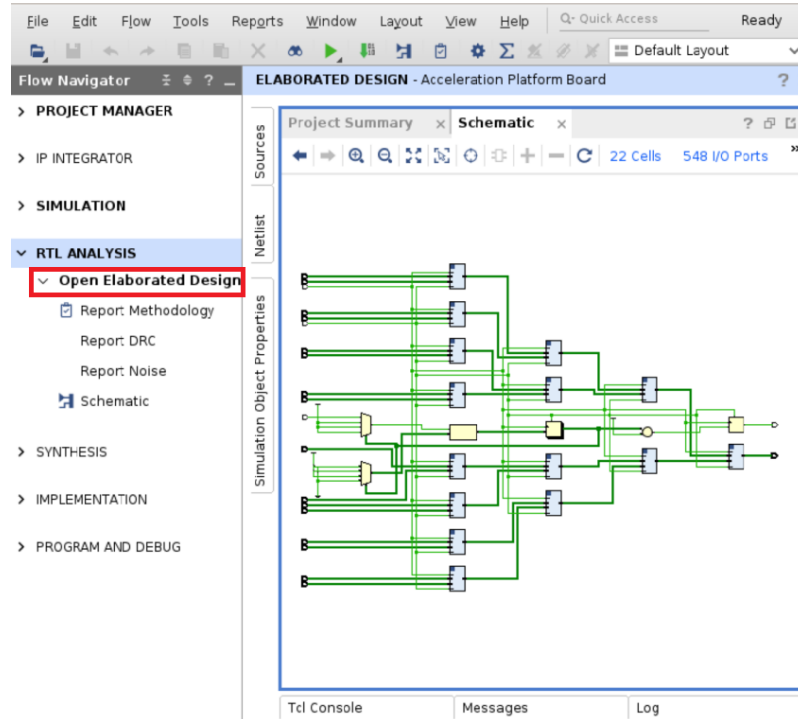

FIGURE C.10
Step 4.1: Click Open Elaborated Design.

The RTL analysis schematic can provide a visual representation of the hardware design structure. For example, in Fig. C.11, we can see that all the FP multipliers are instanced in the red boxes, while the FP adders are instanced in the following three layers of the orange boxes. The middle blue boxes contain the registers used in the timing controller design. By comparing the RTL analysis result with our expected RTL designs shown in Fig. **??** in Chapter **??**, we can ensure that our Verilog code has been correctly synthesized into hardware.
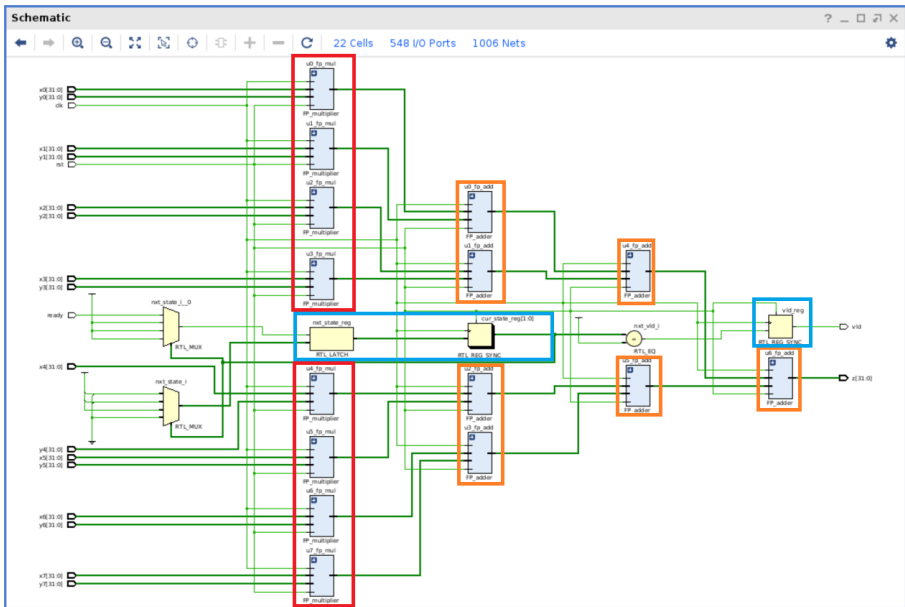
FIGURE C.11
Step 4.2: RTL analysis result.

Step 5: Run synthesis.

Synthesis is the process of converting a high-level RTL design into a gate-level representation. To run synthesis in Vivado, refer to Fig. C.12, and click Run Synthesis. Once synthesis finishes, click Report Utilization to see a more accurate estimate of the resource cost. In the Utilization panel, the design is reported to take 4,329 LUTs, 489 Registers, 16 DSPs, 548 IOs, etc.

To view the power estimation after implementation, click Report Power and Schematic. The implementation of the ddot design is reported to consume 16.205 W static power and 128.081 W dynamic power, as shown in Fig. C.13.

Step 6: Run implementation.

Synthesis is the process of translating an RTL-specified design into a gate-level representation. Implementation, on the other hand, involves a series of steps necessary to place and route the netlist onto the FPGA device resources.

To check the resource utilization after implementation, refer to Fig. C.14 and click Run Implementation. Once the implementation is complete, click Report Utilization to obtain a more accurate estimation of the resource cost. The Utilization panel will display the number of LUTs, Registers, DSPs, IOs, and other resources used by the implementation. In this example, the imple-
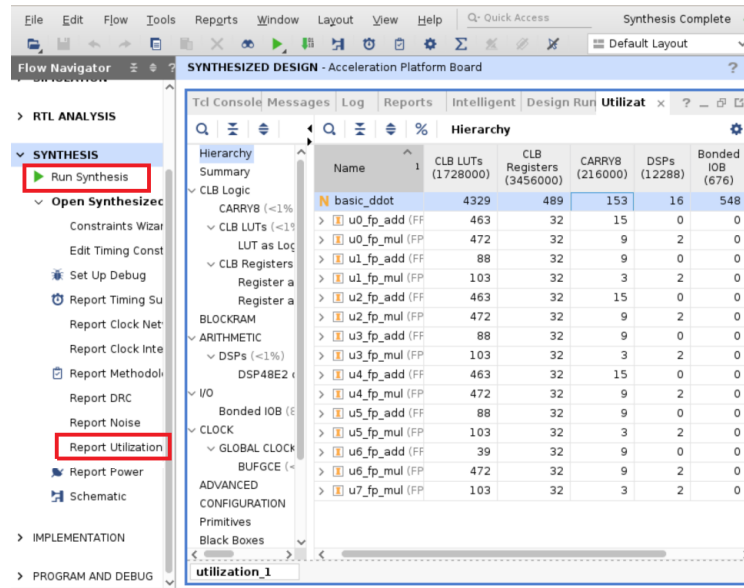
FIGURE C.12
Step 5.1: Synthesis result of resource utilization.

mentation requires 4,315 LUTs, 489 Registers, 16 DSPs, 548 IOs, and so on.

To view the power estimation after implementation, refer to Fig. C.15 and click Report Power and Schematic. The power report shows that the ddot implementation consumes 23.093 W static power and 145.975 W dynamic power.

Step 7: Program the netlist.

To program the selected FPGA board, click Generate Bitstream and then open Hardware Manager. From Hardware Manager, select Open Target and then Program to load the generated bitstream onto the FPGA board.
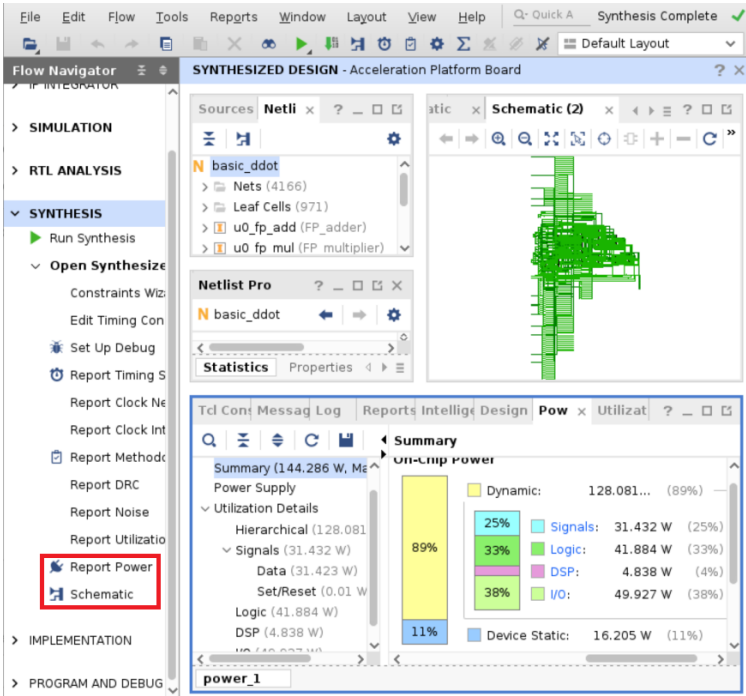
FIGURE C.13
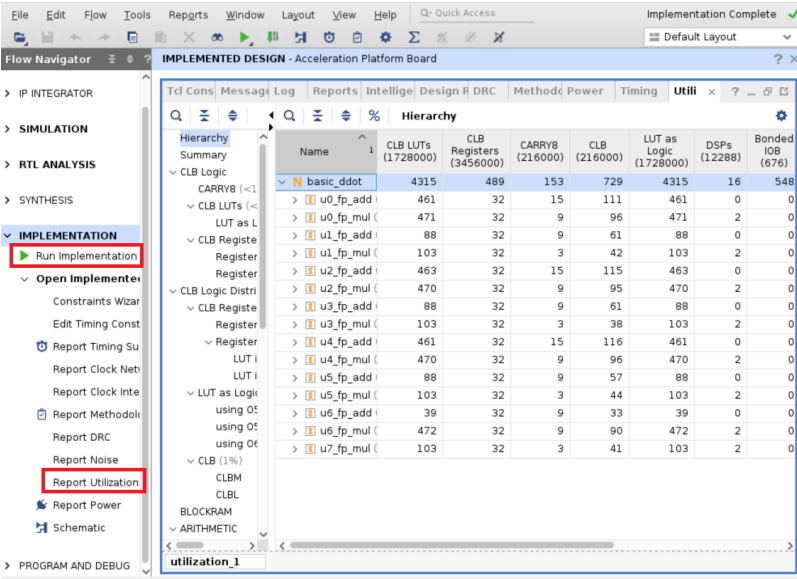Step 5.2: Synthesis result of power report and schematic.
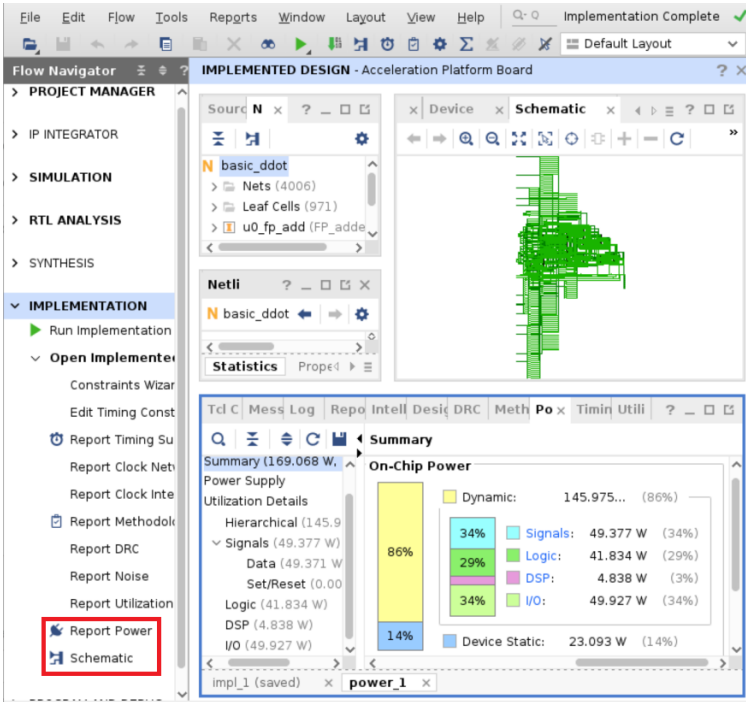
FIGURE C.14
Step 6.1: Implementation result.

FIGURE C.15
Step 6.2: Implementation result.