Multiplier Core Schematic Lab

Multiplier Core Schematic Lab

Introduction

The purpose of this lab is to demonstrate the use of cores in design. Cores are customized, parameterized macros that are used to develop higher complexity designs, with maximum performance, minimum CLB usage, and most importantly, much lower design time.

Objective

After completing this lab, students will be able to:

- Use Xilinx CORE Generator to create a custom module.
- Create a custom macro based on a Core.
- Include this as part of a schematic design.

Procedure

1) Start Foundation Express by <u>clicking</u> on **Start → Programs → Xilinx Foundation Series → Xilinx Foundation Project Manager**.

2) Once inside Foundation Project Manager, <u>click</u> on **File → Open Project…**

3) <u>Browse</u> to **C:\F15_labs\cores**, and <u>select</u> the project, "**schver**". <**Open**>

4) <u>Click</u> on the **Schematic Editor** button in the Flow window. A nearly completed schematic design appears. We will return to this soon.

Creating a new core with Xilinx CORE Generator

5) Start CORE Generator: **Start → Programs → Xilinx CORE Generator → CORE Generator**

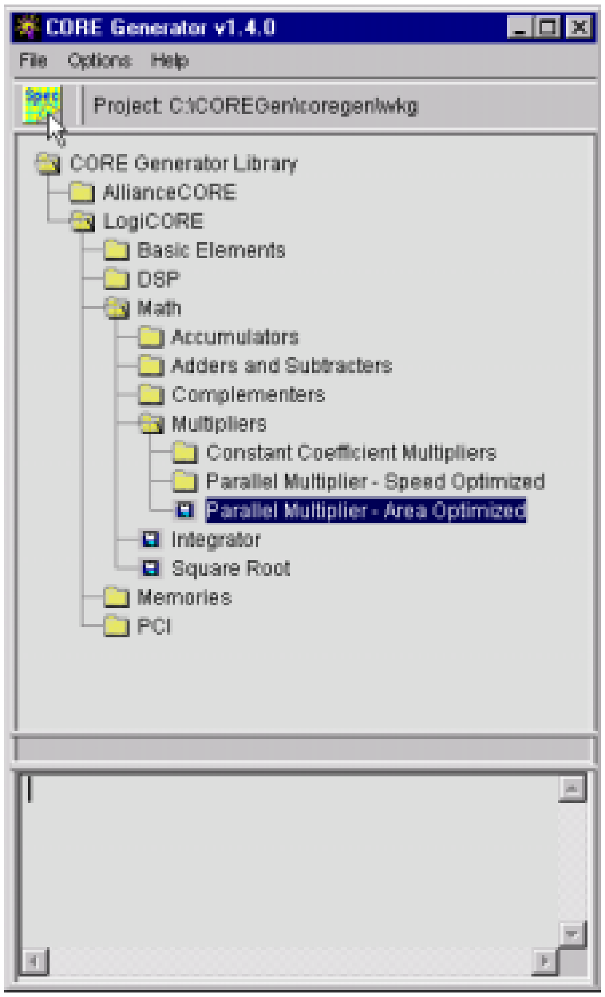
6) In the **CORE Generator Options** window, put checkmarks in the boxes to <u>select</u> ☑ "**EDIF Implementation Netlist**" and ☑ "**Foundation Schematic Symbol**". Both an XNF Netlist and a schematic symbol are required to create a new library macro. <u>Click</u> <**OK**>.

7) Go to the **CORE Generator v1.5.0** window. Note that there is also an empty, CORE Generator window. This is the Java executable window and should not be interrupted or mistaken for the Core Generator v1.5.0 window you should go to now.

8) Set the directory location for the CORE Generator: **Options → System Options…** In the Project Path edit box, type "**`C:\F15_labs\cores\schver`**". This is where the core will be created. Click <**OK**>.

9) Go back to the **CORE Generator v1.5.0** window. You'll notice two categories of Cores: **AllianceCORE**s and **LogiCORE**s.
AllianceCOREs are products developed out of a cooperative effort between Xilinx and independent third-party core developers. It is designed to produce a broad selection of industry-standard solutions dedicated for use in Xilinx programmable logic.
LogiCOREs are cores developed and available from Xilinx directly. Most of these are free and included with the CORE Generator, or directly off of Xilinx's website.

10) Select the desired Core: Double-click on each subcategory, **LogiCORE → Math → Multipliers,** and click once on **"Parallel Multiplier – Area Optimized"** to highlight it. You can view the datasheet on this core by selecting the **"Spec"** button on the top left.



11) Once you are finished viewing the data specification, double-click on the **Parallel Multiplier – Area Optimized**.

12) Under **Component Name**, type "**`mul6x6`**" in lowercase letters. If you type an invalid name, it will appear in red letters.

13) The smallest multiplier is 6x6, so set the **A** and **B** inputs to **6** bits, and set the sign to **Unsigned**. Click **Generate**, and watch the bottom of the CORE Generator v1.5.0 window. When finished, CORE Generator will read, "CORE Generator has built mul6x6 (Parallel Multiplier - Area Optimized v1.17)".

14) Exit CORE Generator by clicking the **X** in the top right corner of the window. The Java window will also close automatically.

15) Go back to the Schematic Editor. You can use <**Alt**> + <**Tab**>.

16) Click on the Symbols toolbox icon and look in the toolbox for "**mul6x6**". You probably will not find it.

17) Click on the Symbols toolbox icon again until the toolbox window disappears.

18) Select **Hierarchy → Create Macro Symbol from Netlist**. Make sure the directory being looked in is **`C:\F15_labs\cores\schver`**. Select **Files of Type** to be the appropriate netlist format, in this case "**Xilinx [*.XN*, *.BAX]**". Select "**`mul6x6.xnf`**" and then <**Open**>. This creates a Foundation symbol, simulation netlists, and functional descriptions of the new core.

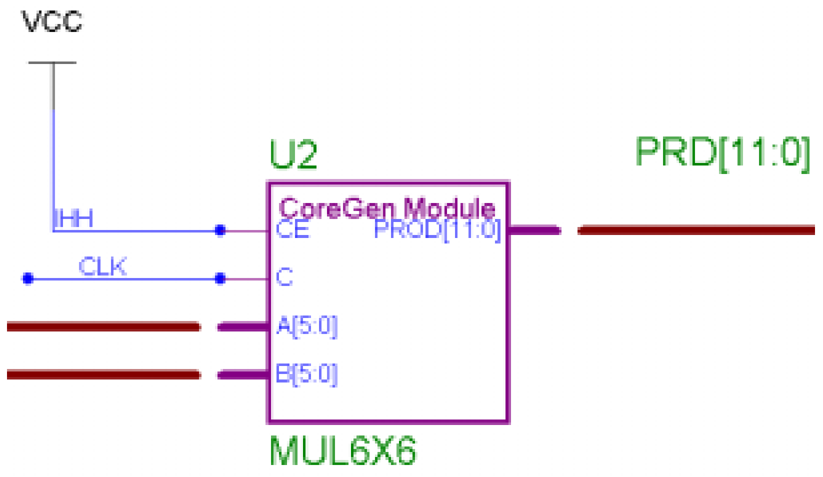
19) Zoom in on your schematic to the area near the center where several busses and nets seem to converge but are not connected.

20) In step **18**, above, the **`mul6x6`** macro should have been added to your project library. However, let's confirm this just to show how the library manager works: Select **File → Project Libraries**. Here you see **Project Libraries**, including Unified libraries (**XC4000EX**) and **simprims**, as well as the libraries created in this project, **schver**.

21) Select the **Lib Manager** button. Library Manager shows all recorded projects and libraries (System and User libraries). Find **`SCHVER`**, located at **`C:\F15_labs\cores\schver\lib`** directory and highlight it. Click the **Objects** tab. Now you should see various custom macros for this design, including the newly added **`mul6x6`** module we have just created.
(If mul6x6 does not appear, you may need to update your library: From Schematic Editor, select **File → Update Libraries**. This refreshes the new **MUL6X6** macro into your library.)

22) Now select **File → Exit** to exit the Library Manager. Also close the Project Libraries window.

23) Return to the **Schematic Editor** and click on the Symbols toolbox icon again, and now look for the **`MUL6X6`** macro. Place this symbol between the busses so that the busses have an equal gap from **MUL6X6** on either side.



24) Select the **MUL6X6** macro, indicated by a red rectangle around the selected object. Then click the Connect Symbol icon . The nets and busses should connect.

25) Save your new schematic: **File → Save**.

26) Go back to the **Foundation Project Manager**. Click on the **Implement M1** button. Implement your design.

27) Download the design to the XS40 & XSTEND board. At the **DOS prompt**,
   A) type "**`cd \F15_labs\cores\schver`**" <**enter**>
   B) type "**`xsload schver.bit`**" <**enter**>

Verifying your design in-system

28) The schematic version of the multiplier has two 4-bit inputs, A and B, which are multiplied together to form a binary output at LEDs D[8:1]. The inputs are multiplexed to come from either the 8-bit DIP Switch on the XSTEND board or from the computer's parallel port. To verify proper operation, let's try both input sources.

29) ***SWITCH INPUTS***
   A) Press the **SPARE** button on the XSTEND board until the *bottom segment* of the *XS40 board*'s 7-segment display is lit up. Note that this may take a few attempts since the SPARE button does not have any debounce circuitry included in this design.
   B) When low, or '0', each DIP switch pulls the connected pin of the XS Board to ground. So, to provide inputs of B=3 times A=5, or 0011 times 0101, position the switches like this: ⇑⇑⇓⇓⇑⇓⇑⇓, where ⇑ points up towards the High, or '1', position. The resulting pattern on the LEDs should be 15(dec), or 00001111.

30) PARALLEL PORT INPUTS
   A) Alternatively, you can push 8 bit vectors into the XS40 or XS95 PLDs. Push the **SPARE** button until the upper segment on the XS40 7-segment display is lit. Now the system is in parallel input mode.
   B) Open a DOS session: **Start → Run… →** type **`cmd`** <**OK**>.
   C) Type "**`XSPORT 00110101`**" <**enter**>. This is equivalent to step 26-B.
   This capability is useful for other experiments. Another example of the novelty of this feature is that one can include the **`XSPORT`** command in a 'C' shell script as a function call in order to provide several 8 bit inputs or test vectors in rapid sequence.

<u>Questions</u>

1) How many CLBs did the 6x6 multiplier require? ________
   How many CLBs did the entire design require? ________
   What percent of the entire XC4005XL was this? ________

2) Assuming the XC4005XL-3 has about the same timings as an XC4005E-3 device, what is the expected performance of the multiplier? ________
   How fast could you expect this to operate in a –1 speedgrade device? ______
   What was the actual overall performance of the entire design? ________

Answers

1) According to the CORE Generator spec sheet, a 6x6 bit unsigned multiplier requires 37 CLBs.
   According to the Placec and Route report, the entire design requred the following resources:

```
PAR: Xilinx Place And Route
Device utilization summary:

   Number of CLBs                    30 out of 196    14%
      Total Latches:                  0 out of 392     0%
      Total CLB Flops:               47 out of 392    12%
```

2) According to the CORE Generator spec sheet, this multiplier can run at approximately 60MHz with a '–3' speedgrade XC4000E-3 FPGA. According to the spec sheet, this multiplier could run at 96MHz with a '–1' speedgrade XC4000E device.
   The Post Layout Timing Report verifies this performance:

```
Post Layout Timing Report
Timing summary:
---------------

Design statistics:
   Minimum period:  14.537ns (Maximum frequency:  68.790MHz)
   Maximum net delay:   11.095ns
```