RAM Lab - CALC

RAM Lab - CALC

Introduction

The XC4000 version of the CALC design can implement a stack in the form of Look Up Table (LUT) -based RAM. This reduces the CLBs from more than eight to only two, since many more bits can be stored per CLB in RAM rather than flip-flops.

Objective

- Show how to include RAM in a schematic.
- Demonstrate the logic resource efficiency obtained using LUT-based RAMs vs. RAMs in flips-flops.
- Demonstrate the Calculator design.

Procedure

1) Examine the design using RAM in the CALC project in the **`C:\F15_labs\memory`** directory by invoking the Foundation Project Manager. **Start → Programs → Xilinx Foundation Series → Xilinx Foundation Project Manager**.

2) Open the **`CALC`** project: Select **File → Open Project...** . Browse to the **`C:\F14_labs\memory`** directory, click the **`CALC`** project, and select <**Open**>.

3) Select the **Schematic Editor**.

4) The RAM is used under the **`STACK_4K`** macro. Select the **`H`** toolbar icon and then double-click the **`STACK_4K`** symbol. You can go back up a level by double-clicking in empty space.

Note that we used the standard **`RAM16X4`** macro but we tied two of the address lines high to make it effectively 4X4. The depth was limited to keep the register-based version at 16 flip-flops, but with RAM there would be no penalty extending the depth to the full 16 words available per lookup table.

5) Examine the results from the implementation process.

6) Go back to **Foundation Project Manager** (<**Alt**> + <**Tab**>) and select **Implement M1**.

7) This design has already been implemented so it not necessary for you to do so. However, you should look at the implementation reports. Select **Utilities → Report Browser...**

8) Double-click the **Place & Route Report**. How many CLBs are being used? _____

9) Go back to the **Foundation Schematic Editor**.

10) Select **File → Open → Browse>>** **`C:\F15_labs\flow\calc\calc.sch`** <**OK**> to view a different, non-RAM based design which otherwise is identical to the design we are working with. Remember now you have two schematic windows open and they both have the same name. One is in your project directory and the new one from another design directory. Be careful to remember which schematic window is which.

11) Select the **H** toolbar icon and then double-click the **STACK_1** symbol. You can go back up a level by double-clicking in empty space. Note that this design is done using standard logic library elements, not RAM.

12) Return to the **M1 Design Manager** and select **File → Open Project…** Find **`C:\F15_labs\flow\calc\xproj\calc.xpj`** and select <**Open**>.

13) Browse the reports : Open the **Place & Route Report**. How many CLBs are being used?____

14) The CLB reduction probably seems higher than you may have expected for the removal of 16 flip-flops. What would explain this large reduction in CLB count? _____

15) Download the design to verify the same functionality. Make sure you test the stack function.

Note: Synchronous RAM is being used in this design. The only timing requirements that need to be met are that the data, address, and write enable need to be set up before the clock edge. In the original design, the **WE** was delayed until two clocks after the cycle **EXC** goes high. This allows **EXC** to enable the address counter, the address counter to change on the next clock, and then the **WE** to be activated on the next clock. With ***synchronous*** RAM, the **WE** can be set up at the same time as the address counter so that both will be available at the next clock edge to write into the RAM.

Answers

How many CLBs are being used in the RAM based design?
How many CLBs are being used in the non-RAM based design?

The RAM-based stack is 2 CLBs (both 16x2, although only four of the addresses are being used). The register-based stack design takes 19 more CLBs, so it must require 21 CLBs for the stack. This includes 8-16 CLBs for the 16 flip-flops, 2-4 CLBs to generate the four clock enables, and 4 CLBs to mux the four outputs. The flip-flops and enables have a CLB range because they could be placed one per CLB or two per CLB.