Physics 623

FPGA II: Construction of a ROM

Dec. 14, 2006

1 Objective

A 32 word 12 bit read-only memory (ROM) is constructed in the Spartan II FPGA and is used to drive three seven-segment displays with a programmed pattern. The ROM is constructed using the Xilinx CORE generator and the ROM words are written into the ROM with the CORE memory editor. The ROM addresses are set by a five bit binary counter which is driven with the test board external 50 MHz clock oscillator. The clock frequency must be stepped down to make the display visible using, for example, the MSB of a 25 bit binary counter. We will program the ROM with the first 32 prime numbers, i.e. 2 through 131 and read it out with the seven-segment displays.

2 Procedure Outline

The top level source will again be a schematic diagram. Since 32 prime numbers takes you into the hundreds, we have to use all three seven-segment displays and three seven-segment decoders. We will also again use a count-down for the external 50 MHz clock. The 32x12 ROM requires five address lines so we will need a five bit counter to sequentially address the ROM. While the ROM can be built directly with VHDL coding, we will use a Xilinx tool called the CORE generator which is used to build preconfigured logic functions. Since **ISE 6.1** does not support the CORE generator we must use a newer version of the Xilinx software, **ISE 8.2**.

2.1 Xilinx Cores

The CORE generator can be used to produce devices ranging in complexity from simple arithmetic operators and delay elements to complex building-blocks such as digital signal decoders, processing filters, multiplexers, transformers, FIFOs, and memories.

3 Core Implementation

The Core to be built is treated in the Project Navigator like another type of source. The Cores are built using the CORE generator which can be run either inside or outside the Project navigator. To start the CORE generator outside of the Navigator select *Start* → *Programs* → *Xilinx ISEAccessories* → *CORE Generator*. To start the CORE generator from within Xilinx ISE select *NewSource* and *IP*(*Coregen&ArchitecturalWizard*). The next window will present a choice of Core types. Choose *Memories* & *StorageElements*, then *RAMS* & *ROMs*, and then *Distributed Memory v*7.1. After executing *Finish* a window will open up which is a tool to construct the memory. Choose *ROM*, specify the required Depth and Width, go to *Next*, and then execute *Generate*. A .xco file will be generated, but remember that the ROM at this point is empty. Now you can run the CORE Generator from within ISE by executing *ManageCores* from the Processes window. Load your Core .cgp Project File and under *Tools* select the *MemoryEditor*. You again specify the depth and width but you will now also be able to edit the memory contents. Select **2** for the Address Radix (since you will address the ROM in binary) and select **16** for the Data Radix since you will write the memory contents in Hex. Each memory entry will contain three Hex characters corresponding to 12 bits. For example, entering the prime 5 will be entered as **005** and so on. When the memory contents have been entered, enter a memory Block name and execute *Generate* under the File menu item. Select *Making a .coefile*. Exit the Memory Editor after saving all the files.

4 Creating the Bitmap

Now Synthesize, Translate, Map, Place and Route as before. In this version of the ISE software the constraints file cannot be edited until **Translate** has executed. Under **Translate** you will find an item **Assign Package Pins Post-Translate**. Clicking on this will open the Constraints (PACE) editor. After editing, **Implement Design** will have to be rerun. After all the green checkmarks are in place, create the .bit file with **Generate Programming File**.

5 Operation

To enable all the segments of the daughter board seven-segment displays, the file **dwnldpa2.svf** must be downloaded into the CPLD on the circuit board. This file is located in the **xstools\xsa** directory. Run the circuit and establish that everything is working correctly.

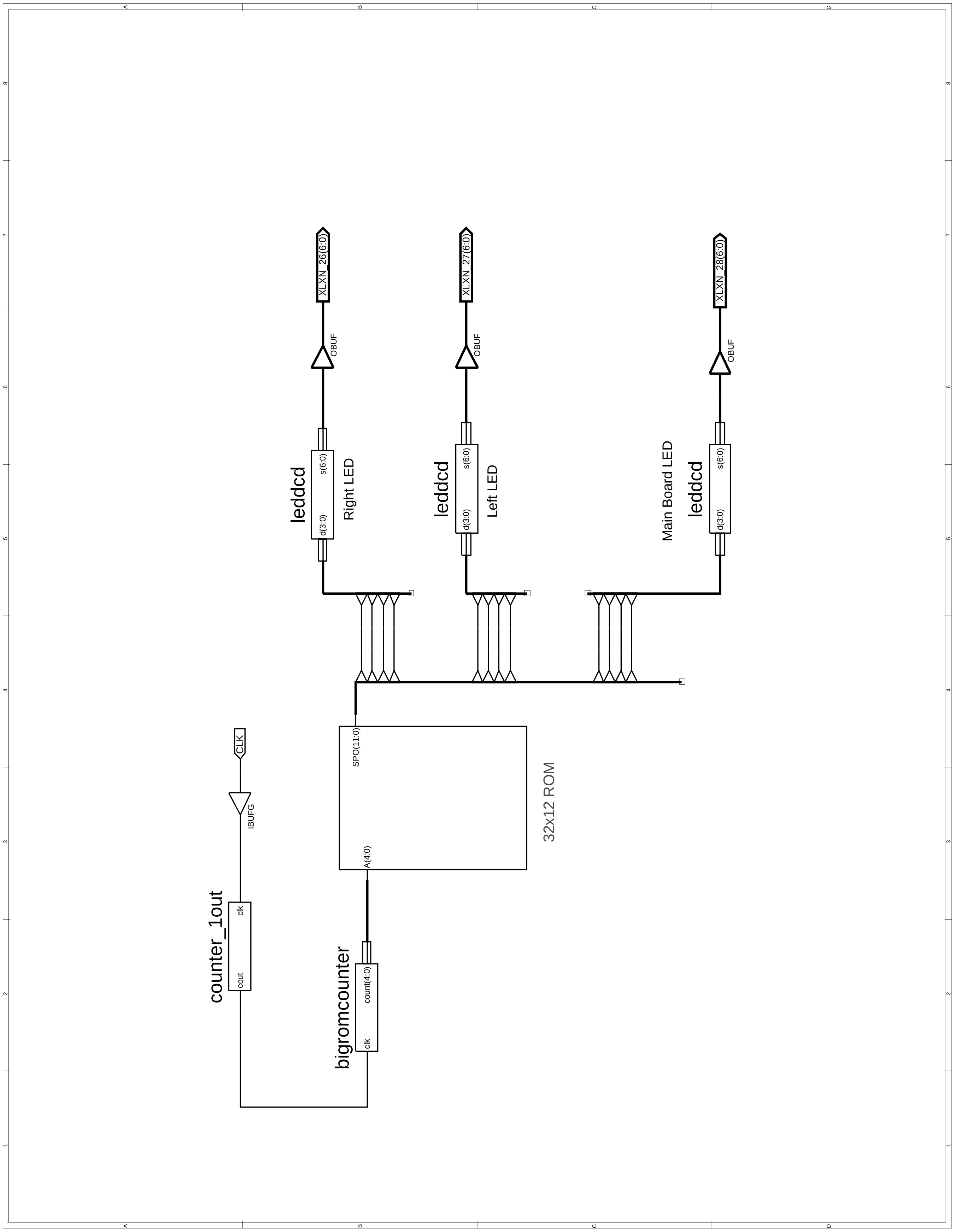
6 Questions

1. The ROM will be made up out of the Spartan II Block RAM. Our chip has eight Block RAM cells each of which is a fully synchronous dual-port 4096-bit RAM giving 32 Kbits of total Block RAM. The rest of the circuit has registers that are made up out of the D Flip-Flops in the CLB. Estimate the number of CLBs required to implement the circuit and compare your estimate to the information in the Project Status Report.

2. Determine the maximum clock frequency for the circuit (It better be above 50 MHz). The required information is in the Post-Place Static Timing report.

3. Now we will look at the performance of the ROM by filtering out everything but the inputs and outputs of the ROM block. Run the Timing Analyzer and once the window is open select **Analyze** and **Against User Specific Paths by Defining Endpoints**. Enter the nets corresponding to the Sources and Destinations of the ROM and run the Timing Analyzer.
   What is the worst case access time of the ROM?
   What is the total worst case delay including nets?

7 Circuit Diagram



XLXN_26(6:0)
XLXN_27(6:0)
XLXN_28(6:0)
OBUF
leddcd
Right LED
Left LED
Main Board LED
s(6:0)
d(3:0)
SPO(11:0)
A(4:0)
32x12 ROM
CLK
IBUFG
counter_1out
clk
cout
bigromcounter
count(4:0)