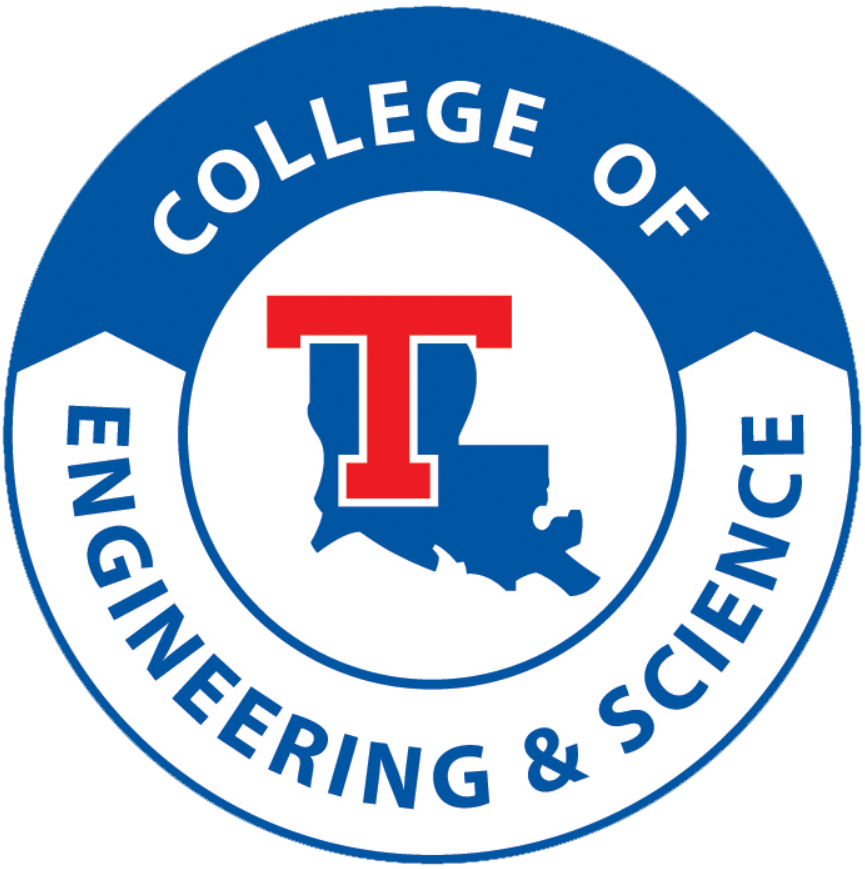
2013

Louisianan Tech College of
Engineering and Science

Champy Gahagan



STEP BY STEP GUIDE TO IMPLEMENTING THE CORTEX-M0 USING A NEXYS2 FPGA BOARD

STEP BY STEP GUIDE TO IMPLEMENTING THE CORTEX-M0 USING A NEXYS2 FPGA BOARD

CONTENTS

INTRODUCTION

This guide describes the process of implementing an ARM Cortex-M0 DesignStart processor in an FPGA board. The DesignStart model of the Cortex-M0 has slightly limited features and functionality when compared to the standard M0 core. However, the DesignStart version is available free of charge to the general public. We will be using the Digilent Nexys2 development board, which is built around a Spartan 3E FPGA.

A small program will be built in order to verify that our system is working properly. This will require a bit more front end work as we will have to add peripherals to our M0 core to be able to execute the code. These peripherals include the following: a reset synchronizer, memory preloaded with the program, a system clock, and a pattern detector on the data bus. Once this is done, our system will be simulated in software to confirm functionality. Finally, our system will be synthesized onto the FPGA. The program will provide LED feedback that will verify whether or not the system is operational.

REQUIRED HARDWARE/SOFTWARE

The following hardware and software will be needed for this project. All of the software packages can be obtained for no cost. The same does not hold true for the Nexys2 board, however.

1. Nexys2 development board
   http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,789&Prod=NEXYS2
2. Xilinx ISE WebPACK design software
   http://www.xilinx.com/products/design-tools/ise-design-suite/ise-webpack.htm
3. Digilent Adept software
   http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,828&Prod=ADEPT2
4. Digilent plug-in for Xilinx tools
   http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,768&Prod=DIGILENT-PLUGIN
5. Cortex-M0 DesignStart
   http://www.arm.com/products/designstart/index.php?tab=processor
6. ARM/Keil MDK evaluation version
   https://www.keil.com/arm/demo/eval/arm.htm
7. BIN to COE conversion utility
   http://sourceforge.net/projects/bin2coe/
8. VHDL module files
   http://web.fi.uba.ar/~pmartos/publicaciones/Deliverables.zip

PROJECT OVERVIEW

The project will divided into the following four parts:

1. Software Development

   We will build a small program using the ARM/Keil MDK that will verify the complete system is handling memory fetches properly. The memory values fetched will hold predefined constants. During hardware simulation (Part 3), we will see these values on the processor's data bus. During hardware verification (Part 4), we will see an LED turn on and off when these values are present on the data bus.

2. System Development

   Using the Xilinx ISE software, we will build a system that is capable of executing the code developed in Part 3. This will involve implementing the Cortex-M0 core as well as several peripheral devices.

3. Hardware Simulation

   Using the ISIM tool, we will simulate the system developed in Part 2. We will also run the code developed in Part 1 on this simulated system to verify that we see the expected constant on the processor's data bus.

4. Hardware Verification

   Finally, the complete system will be synthesized and downloaded to the board. The program we developed in Part 1 will cause an onboard LED to turn on and off when certain constants are seen on the data bus, thereby allowing us to verify the memory fetches.

IMPLEMENTATION

This section will give step by step instructions for implementing the Cortex-M0 processor.

PART 1: SOFTWARE DEVELOPMENT

1.1. Open the ARM/Keil MDK application and create a new project with the ARM Cortex-M0 as the chosen CPU. Name the project "FPGA."

1.2. Add the sources "main.c" and "vectors.c" (source code available in the "Appendix" section). The "Project" window should look like Figure 1 below.

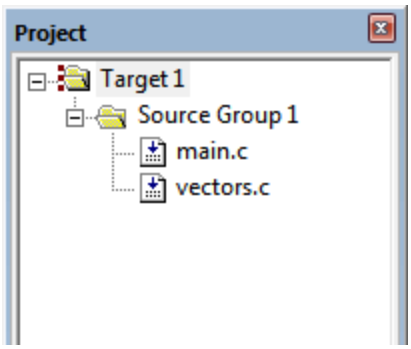


FIGURE 1 - PROJECT WINDOW HIERARCHY

1.3. Open the target options, located under the "Project" menu. Configure the "Target," "Output," "ASM," and "Linker" tabs as shown in Figures 2-5 below. The string for "Misc controls" in the "Linker" tab is the following: " --entry 0x15 --first=vectors.o(__Vectors)"

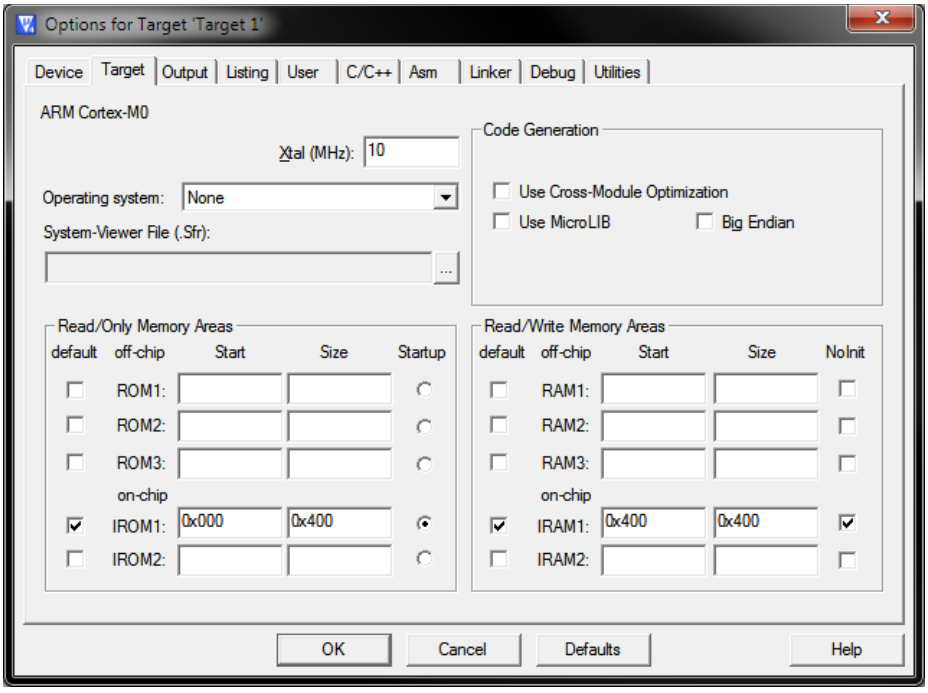


FIGURE 2 - TARGET TAB CONFIGURATION SETTINGS

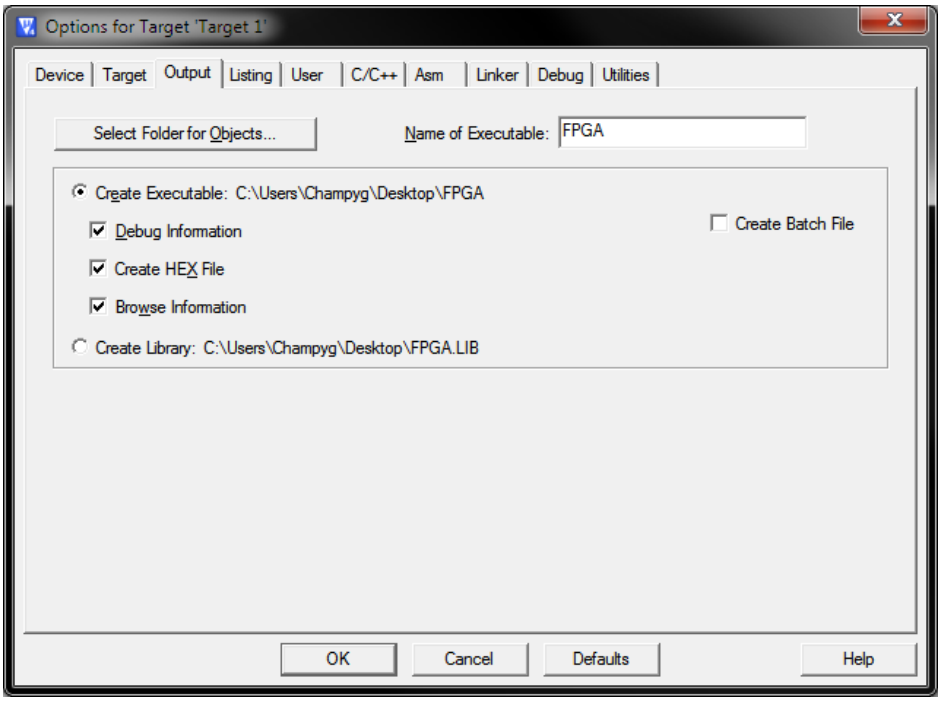


FIGURE 3 - OUTPUT TAB CONFIGURATION SETTINGS

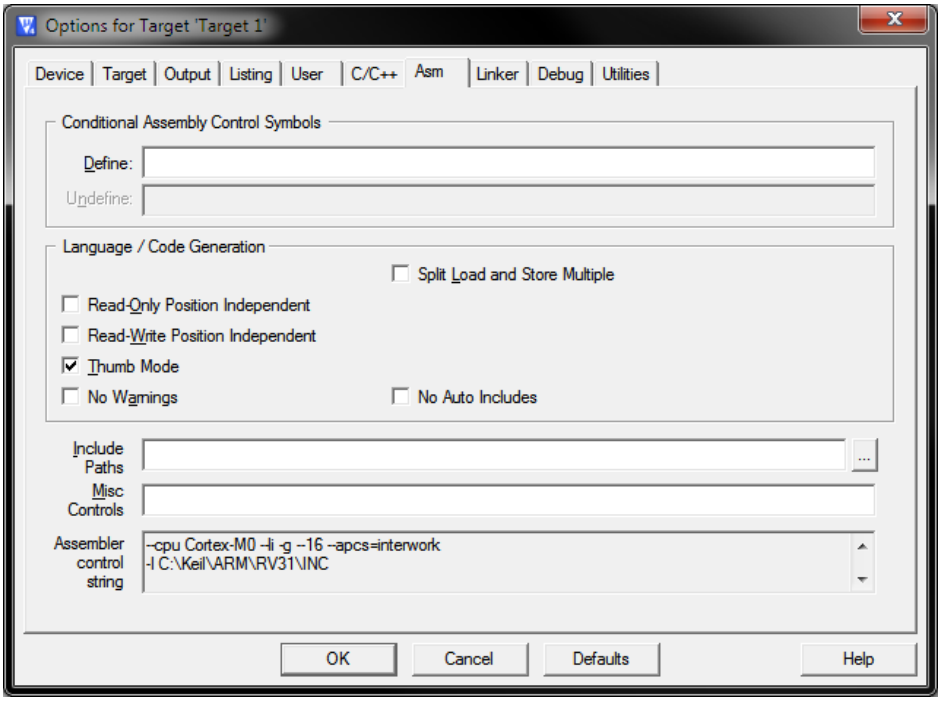


FIGURE 4 - ARM TAB CONFIGURATION SETTINGS

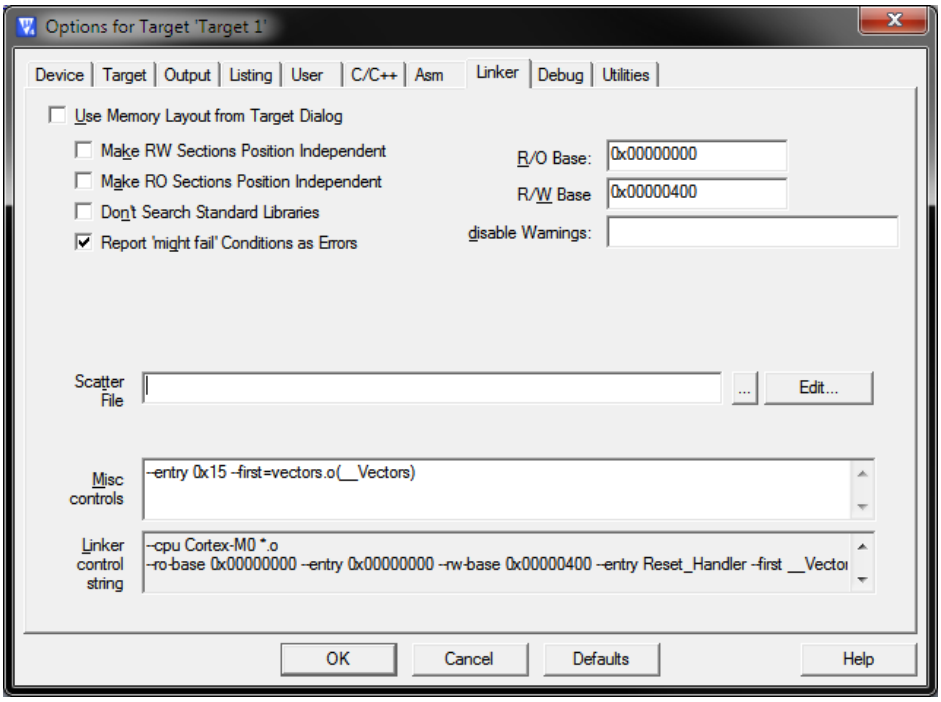


FIGURE 5 - LINKER CONFIGURATION SETTINGS

1.4. Build the executable image by clicking "Build Target" under the "Project" menu. If everything is set up correctly, it should report no errors or warnings and that a ".axf" file was created.

PART 2: SYSTEM DEVELOPMENT

2.1. Open ISE Design Suite and create a new project called "CM0_DSSystem." Click "Next" once. Then select the Spartan-3E Starter Board for "Evaluation Development Board" and VHDL for "Preferred Language."

2.2. First, add the top module. Do this by clicking "Add Source" under the "Project" menu. Select the "CM0_DSSystem.vhd" file from the "Deliverables" download. Now that the top module is complete, the individual modules marked with question marks can be added.

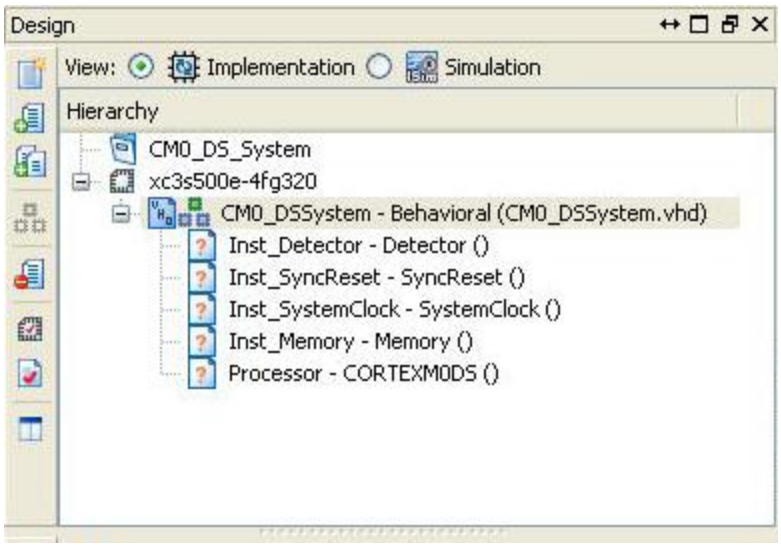


FIGURE 6 - SYSTEM DESIGN HIERARCHY

2.3. Add the two processor files included in the download from the ARM site, "cortexm0ds_logic.v" and "CORTEXMODS.v," in the same manner as above.

2.4. Next, create the 10 MHz clock from the board's 50 MHz external oscillator. Under the "Project" menu, select "New Source." Select "IP" for source type and name the file "SystemClock." In the next screen, click the "View by Name" tab and select "Single DCM_SP."

2.5. Click "OK" on the window that pops up. For the next screen, configure as shown below in Figure 7.

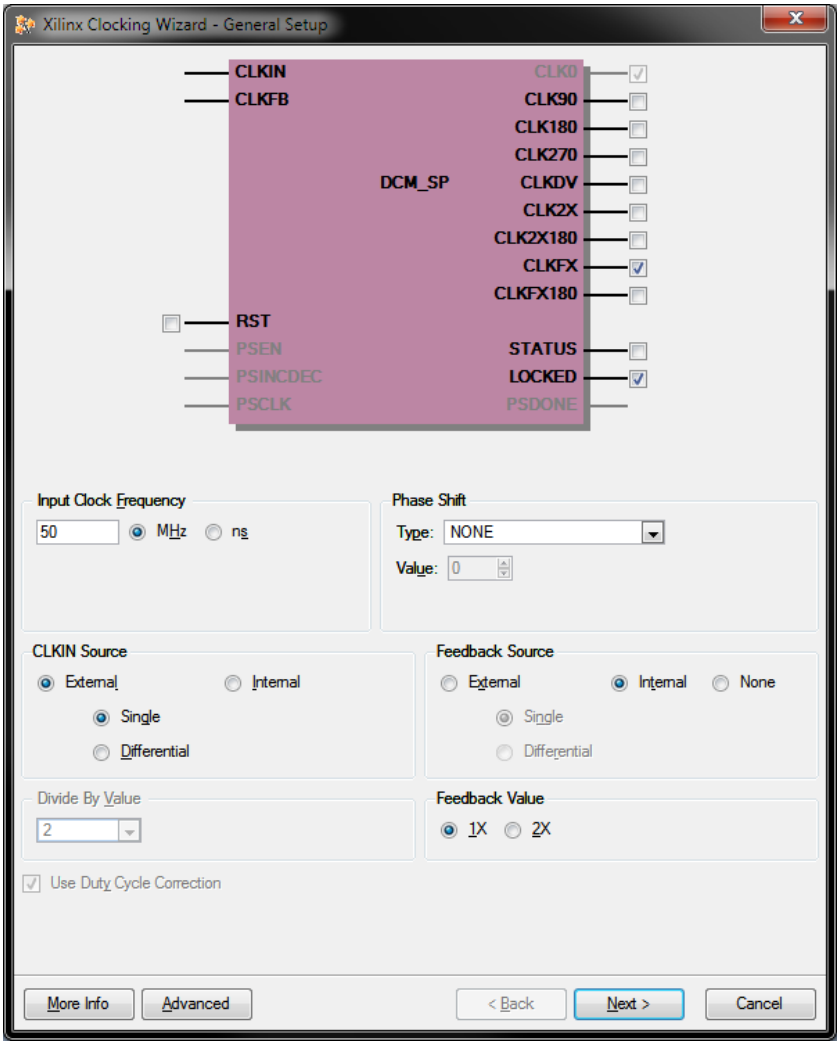


FIGURE 7 - 10 MHZ CLOCK SETUP

2.6. Click "Next." In the next screen, enter 10 MHz as the output frequency and click "Calculate." Figure 8 below shows the configuration for the clock.

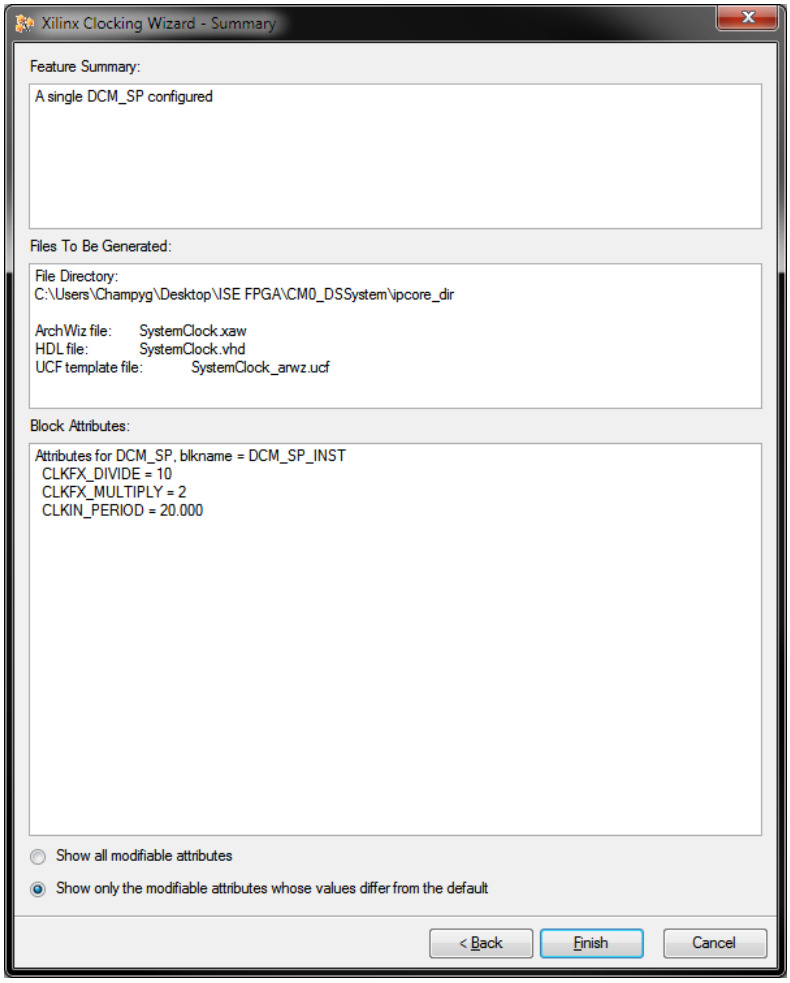


FIGURE 8 - CLOCK FINAL CONFIGURATION

2.7. Next build the data bus detector, which is also includes in the "Deliverables" download. As in step 2.2, select "Add Source," and add the "Detector.vhd." file.

2.8. The reset synchronizer module is implemented as a module and 3 sub-modules. As before, use the "Add Source" button to add these four files: "SyncReset.vhd," "DelayCounter.xco," "Counter2Constant.vhd," and "Constant2Pulse.vhd."

2.9. Next generate the reset synchronizer. Start by using the "New Source" button. The source type is, once again, "IP." Name the file "DelayCounter," and select "Binary Counter" for type. Click "Finish."

2.10. Configure the Binary Counter as shown below in Figure 9.

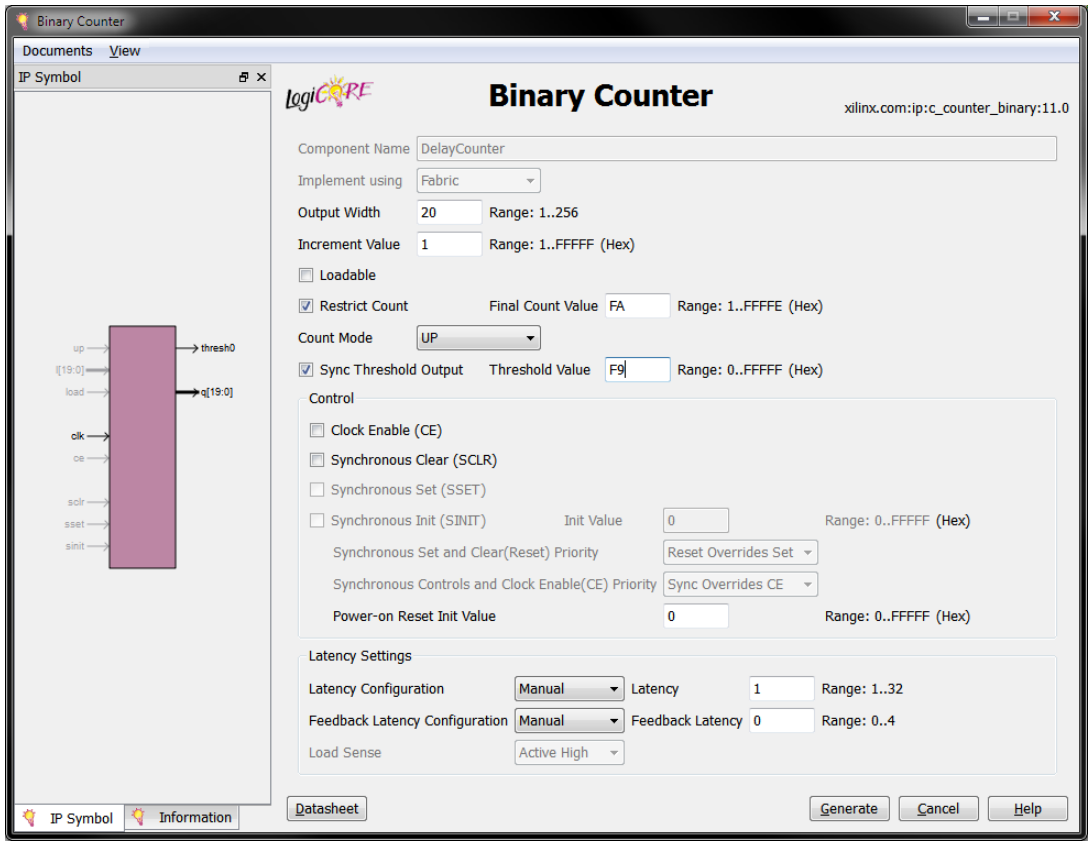


FIGURE 9 - BINARY COUNTER CONFIGURATION

2.11. Generate a ".bin" file from the ".axf" created in Part 1 by using the Fromelf command line utility bundled with ARM/Keil MDK. Use the following syntax to invoke it:

fromelf - -bin - -o FPGA.bin FPGA.axf

2.12. Generate a ".coe" file from the bin you just made by using the Bin2Coe utility. This is also a command line utility, and can be invoke using the following syntax:

bin2coe FPGA.bin FPGA.coe 512

2.13. Add a new source of type "IP" named "Memory." Select "Block Memory Generator" for the IP type. Click "Next" until you arrive on page 3 of the configuration. Configure this page as shown below in Figure 10.

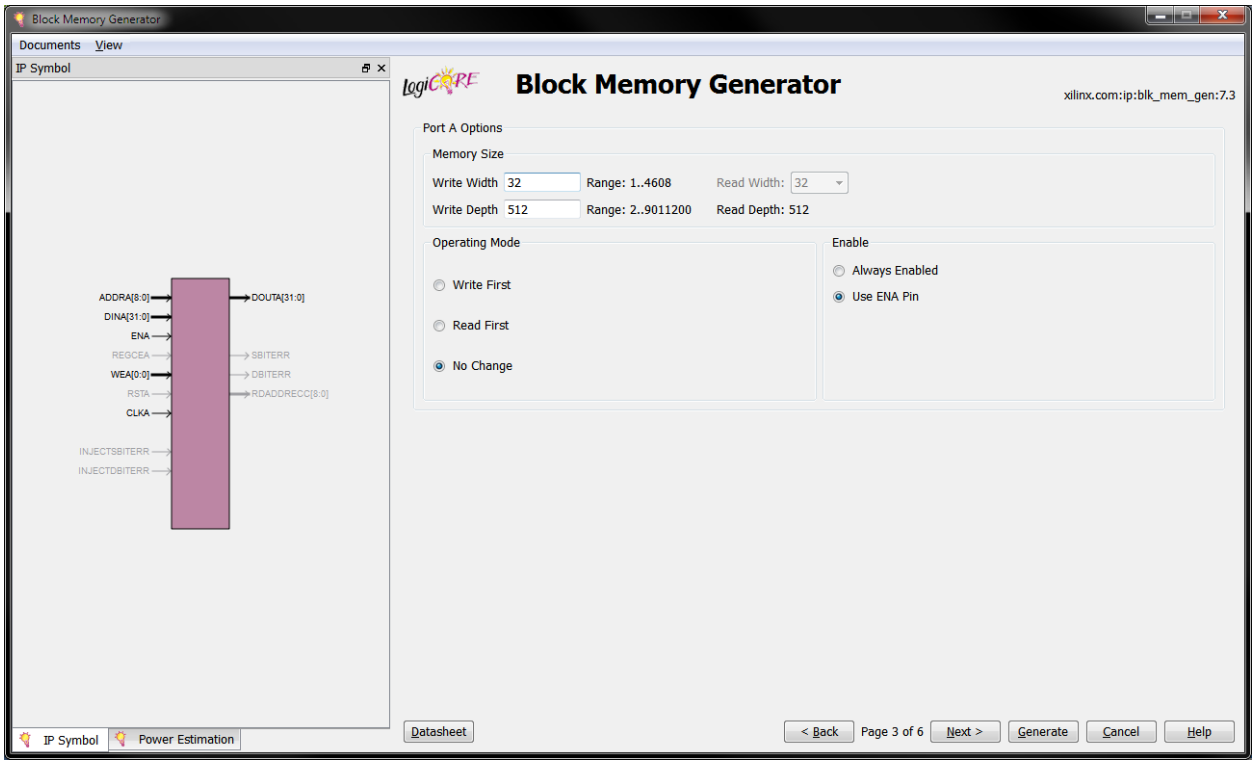


FIGURE 10 - BLOCK MEMORY CONFIGURATION PAGE 3

2.14. On page 4, check the "Load Init File" box and point to the ".coe" file generated in step 2.12. On the next page, check the "Use RSTA Pin (set/reset pin)" box. The final page can be left default settings.

PART 3: HARDWARE SIMULATION

3.1. In the top of the Design View window, select the "Simulation" radio button. This can be difficult to find, so see Figure 11 below for reference.

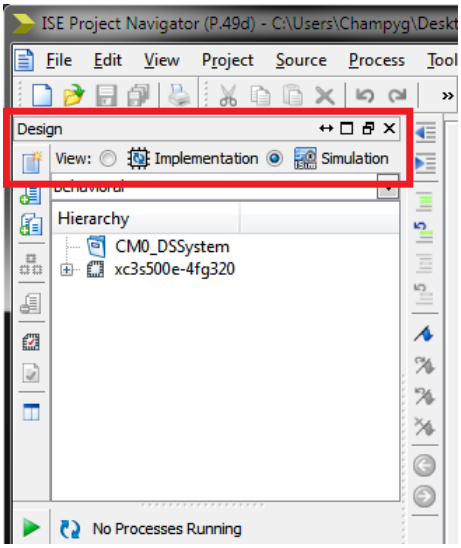


FIGURE 11 - SWITCHING TO SIMULATION VIEW

3.2. Highlight the top module (CM0_DSSystem) and run the "Simulate Behavioral Model" process, which can be found below in Figure 12.

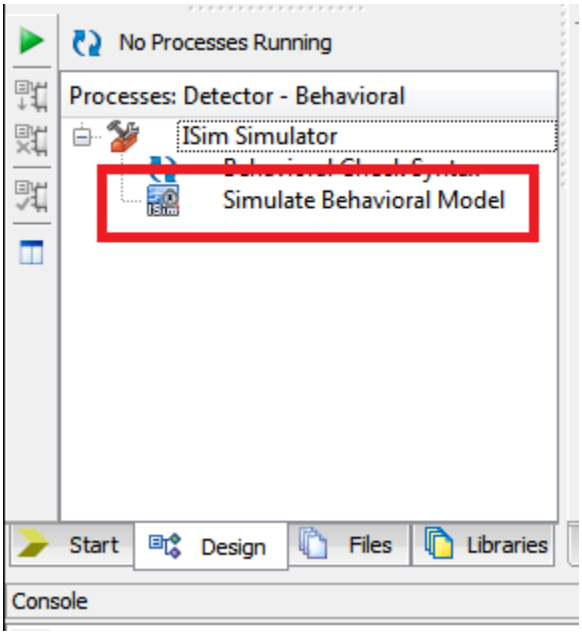


FIGURE 12 - LOCATING THE "SIMULATE BEHAVIORAL MODEL" FEATURE

3.3. Ignore the initial simulation ISIM displays as the clock signal is not yet defined. To do this, right click the "clock_in" signal and select the "force clock" option. Use the parameters shown below in Figure 13.

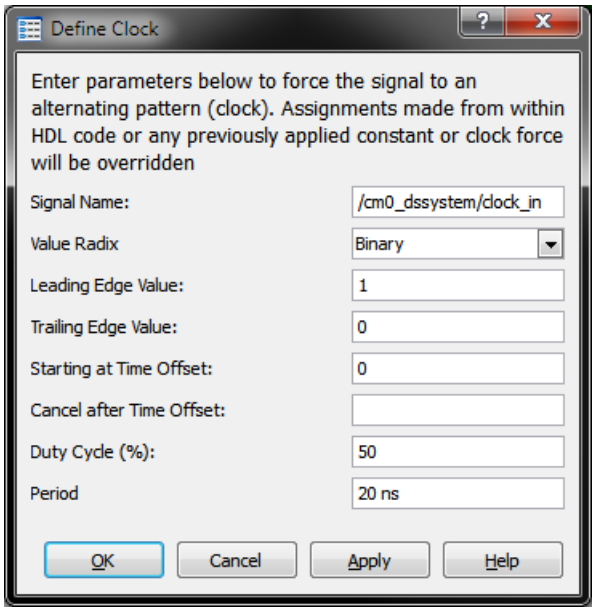


FIGURE 13 - CLOCK_IN PARAMETERS

3.4. Manually key in 300 us in the simulation time box on the far right side of the toolbar. Then click the "Run for the time specified on the toolbar button to the immediate left. See Figure 14 for assistance in locating these items.

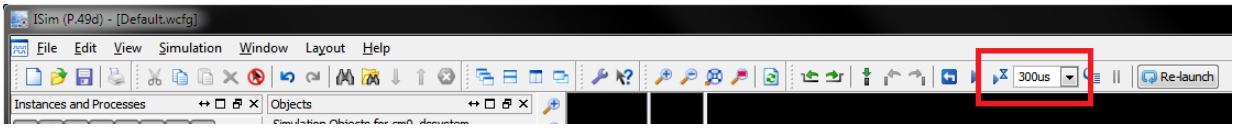


FIGURE 14 - MANUAL SIMULATION TIME TOOLBAR LOCATION

3.5. After this simulation runs, verify that the system is performing as expected by locating the led3 signal. Confirm that this signal switches between a high and low state as shown in Figure 15.

FIGURE 15 - LED3 SIGNAL SWITCHING FROM HIGH TO LOW

PART 4: HARDWARE VERIFICATION

4.1. Add another new source to the project. Select "Implementation Constraints File" for the source type and name it "CM0_DSSystem." This adds an empty ".ucf" file. Copy and paste the Nexys2 UCF code from the "Appendix" section into this file.

4.2. Change the "main.c" program so that it uses the period of 20000000 instead of 200 (200 was used for simulation purposes). Recompile the program, and repeat steps 2.11 through 2.14 in order to use this new period value.

4.3. Under the "Tools" menu, select "iMPACT." This tools will flash the system with built in program onto the FPGA. Once this is done, you should see the LED3 blinking.

CONCLUSIONS

Congratulations! You have successfully implemented an ARM Cortex-M0 DesignStart processor running a pre-loaded program on a Xilinx FPGA. You have also performed functionality tests along the way including software simulation and hardware simulation.

APPENDIX

The source code needed to carry out this project is listen below

MAIN.C

```
// Define where the top of memory is.
#define TOP_OF_RAM 0x800U

// Define heap starts...
#define HEAP_BASE 0x47fU

//--------------------------------------------------------------------------
// Simple "Blinking Led via Memory Access detection" program.
// This program makes a memory access at regular intervals
// In the Nexys2 system there is a pattern detector attached to the
// HWRead bus, so when two specific patterns are detected, a Led toggles its state
// pattern 0xaaaa5555 turns on the led, pattern 0xf0f0f0f0 turns it off.
//--------------------------------------------------------------------------

#define LedOn 0xaaaa5555
#define LedOff 0xf0f0f0f0

int main(void)
{
	unsigned int counter;		// dummy
	unsigned int ii;		 // loop iterator
	unsigned int trap;		// memory access pattern receiver
	unsigned int period;		// time interval for memory access

	//period=20000000;		// period for FPGA implementation; roughly 3 seconds
					for a 10MHz osc in CM0_DS
	period=200;		// period for simulations in ARM/Keil MDK and Xilinx ISIM tool

	while (1)
	{
		counter=0;
		for (ii=0;ii<period;ii++)
		{
			counter++;
		}
		trap=LedOn;		// memory access pattern (turn on)
		for (ii=0;ii<period;ii++)
		{
			counter++;
		}
		trap=LedOff;		// memory access pattern (turn off)
		trap++;			// dummy
	}
}
```

VECTORS.C

```
// Define where the top of memory is.
#define TOP_OF_RAM 0x400U

extern int main(void);                  // Use C-library initialization function.

__attribute__ ((section("__Vectors")))
static void (* const vector_table[])(void) =
{
        (void (*)(void)) TOP_OF_RAM,    // Initial value for stack pointer.
        (void (*)(void)) main,          // Reset handler is C initialization.
        0,                              // No HardFault handler, just cause lockup.
        0,                              // No NMI handler, just cause lockup.
        0//...                          // Additional handlers would be listed here.
};
```

UCF FILE

```
# clock pin for Nexys 2 Board
NET "Clock_In" LOC = "B8"; # Bank=0, Pin name=IP_L13P_0/GCLK8, Type=GCLK, Sch
name=GCLK0

# Leds
NET "Led0" LOC = "J14"; # Bank=1, Pin name=IO_L14N_1/A3/RHCLK7, Type=RHCLK/DUAL,
Sch name=JD10/LD0

NET "Led1" LOC = "J15"; # Bank=1, Pin name=IO_L14P_1/A4/RHCLK6, Type=RHCLK/DUAL,
Sch name=JD9/LD1

NET "Led2" LOC = "K15"; # Bank=1, Pin name=IO_L12P_1/A8/RHCLK2, Type=RHCLK/DUAL,
Sch name=JD8/LD2

NET "Led3" LOC = "K14"; # Bank=1, Pin name=IO_L12N_1/A7/RHCLK3/TRDY1,
Type=RHCLK/DUAL, Sch name=JD7/LD3

NET "Led4" LOC = "E17"; # Bank=1, Pin name=IO, Type=I/O, Sch name=LD4 s3e500 only
NET "Led5" LOC = "P15"; # Bank=1, Pin name=IO, Type=I/O, Sch name=LD5 s3e500 only
NET "Led6" LOC = "F4"; # Bank=3, Pin name=IO, Type=I/O, Sch name=LD6 s3e500 only
NET "Led7" LOC = "R4"; # Bank=3, Pin name=IO/VREF_3, Type=VREF, Sch name=LD7
s3e500 only
```