ISTANBUL TECHNICAL UNIVERSITY

Digital System Design Applications

Experiment 6
Convolution Circuits

Objectives

- To learn how to design convolution circuits.
- To learn how to use divice primitives.

Requirements

Students are expected to be able to

- define hardwares with Verilog
- create project on ISE
- synthesize, simulate, implement designs, generate bitstreams and configure FPGA

References:

1. Nexys2 Reference Manual
2. Spartan-3E Libraries Guide for HDL Designs
3. Constraints Guide
4. Brown&Vrasenic, "Fundamentals of Digital Logic with Verilog Design", McGraw-Hill, p.229-289

Creating Convolution Circuits Library

Create a new ISE project with the settings below:

- Family: **Spartan 3E**
- Device: **XC3S500E**
- Package: **FG320**

- Speed: **-4**
- Synthesis Tool: **XST (VHDL\Verilog)**
- Simulator: **ISim (VHDL\Verilog)**
- Preferred Language: **Verilog**

Add a new Verilog module named **convolution_circuits** to your project. Clear all lines in your module except timescale.

Behavioral Multiplier (40 Minutes)

1. Open **convolution_circuits.v** file.
2. Write down your multiplier circuit to a new module called **MULTB**.This module is going to have 8-bit inputs **A**, **B**; 1-bit inputs **clk**, **reset**, **start**; 1-bit output **done**, 16-bit output **result**.
3. You will design a behavioral 8-bit multiplier by using only an always block.
4. In every positive edge of clock signal; if reset signal is high, then all outputs must be zero. If reset is low and the start input is high, then multiplication of inputs must be assigned to the result and done signal must be raised.
5. If reset and start signals are both low, then the output signals must be zero again. Use two if statements in one always block.
6. Write a test code to ensure that your circuit is working correctly.
7. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.
8. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? How can you increase the circuit performance? Apply "Design Goals and Strategies" to improve performance of the circuit. You must apply strategies for both area and performance separately.
9. Add all off details and results to your report.

Structral Multiplier (60 Minutes)

1. Open **convolution_circuits.v** file.
2. Write down your multiplier circuit to a new module called **MULTS**. This module is going to have 8-bit inputs **A**, **B**; 1-bit inputs **clk**, **reset**, **start**; 1-bit output **done**, 16-bit output **result**.
3. You will use a very simple binary multiplication algorithm for this design as shown in Figure 1. So, you should define 15-bit registers reg1, reg2, reg3, reg4, reg4, reg5, reg6, reg7, reg8 to save partial products.
4. When designing the always block you have to check reset value and start value. For high(1) value of the reset you have to assign zero(0) to all registers. If reset is low(0) and start is equal to one(1) you can save the partial products to registers.

5. If the multiplier bit is zero(0), then the register will be all zero(0). If the multiplier bit is one(1), then the register will be the **left-shifted** multiplicand value!

6. To obtain the result you have to add the partial products. Use 15-bit Ripple Carry Adders that you designed during Experiment4. Define 15-bit wires sum1, sum2, sum3, sum4, sum5, sum6, sum6, sum7 to connect the adders.

7. In always block, you will also designate the result value. The sum7 value will be the the LSB 15 bits of the result. The MSB of the result is the carry out of the last RCA. So you should define a 1-bit wire to connect carry out to the result reg.

8. The done signal is assigned to one(1) where the result is assigned. Otherwise it is zero(0). If your circuit can not produce the output in one clock cycle, you must wait until the result outcome!

9. You can design it in which way you desire. For example, define a done_shift register which has an initial value zero(0).In always block, assign MSB of done_shift to the done signal. Concatanate the other bits of done_shift with the start signal and assign it to done_shift again. Thus, the done signal will be one(1) after a few clock cycle. (Depends on the wide of done_shift register)

10. Write a test code to ensure that your circuit is working correctly.

11. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.

12. Use Carry Lookahead Adder instead of Ripple Carry Adder and repeat your tests.

13. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.

14. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits especially for carry output of last FA. How can you increase the circuit performance? Apply process strategies to improve performance of the circuit.

15. What kind of differences have you been seen when you changed the adder type?

16. Add all off details and answers to your report.

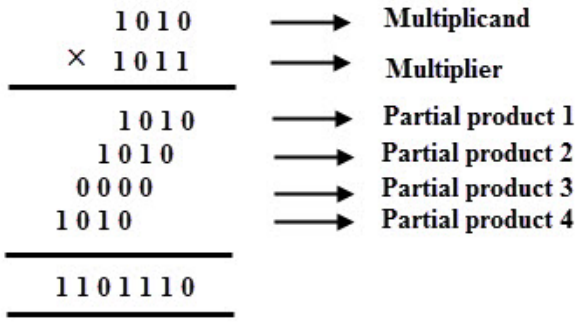

Figure 1: Binary Multiplication

2D-Convolution Without DSP(70 Minutes)

1. Write down a module which is called **CWODSP** into your **convolution_circuits.v** file.
2. This module is going to have 8-bit wide inputs **x11**, **x12**, **x13**, **x21**, **x22**, **x23**, **x31 x32 x33**, **k11**, **k12**, **k13**, **k21**, **k22**, **k23**, **k31**, **k32**, **k33**, 1-bit wide inputs **clk**, **reset**, **start**, 20-bit wide output **result** and 1-bit wide output **done**.
3. You have to learn about 2D-Convolution algorithm. You can make a quick search on the internet or use the source shared with you. To do some practicises you can use Matlab or the free and online version of Octave software.
4. By using the command conv2(image,kernel,'same') in Matlab or Octave you can obtain 2D convolution of the image matrix with kernel matrix. Please note that before the multiplications the kernel matrix is turned both horizontal and vertical.
5. For this design, we will use 3x3 matrices for both image and kernel which are all positive 8-bit integers(you can also use zero(0)).
6. Firstly, you need to make nine multiplications like follows; x11*k11, x12*k12, x13*k13, x21*k21, x22*k22, x23*k23, x31*k31, x32*k32, x33*k33. (These are just examples, you have to determine the correct multiplications according to 2D-Convolution) Use your own multiplier circuit for this design.
7. Then all you have to do is to add the results of these nine multiplication and obtain the main output **result** as follows; result = m1 + m2 + m3 + m4 + m5 + m6 + m7 + m8 + m9. Use your own adder circuits for this design. Use 19-bit RCA and CLA circuits seperatly and investigate the delay and area differences between these two offers.
8. The done signal assignment can be done as told in multiplier design.
9. Write down a test code to ensure that your circuit is working correctly.
10. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.
11. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits especially for carry output of last FA. How can you increase the circuit performance?
12. Apply "Design Goals and Strategies" to improve performance of the circuit. You must apply strategies for both area and performance separately. What would happen if we use another type of adder?
13. Add all off details and answers to your report.

Convolution With DSP(70 Minutes)

1. Write down a module which is called **CWDSP** into your **convolution_circuits.v** file.
2. This module is going to have 8-bit wide inputs **x11**, **x12**, **x13**, **x21**, **x22**, **x23**, **x31 x32 x33**, **y11**, **y12**, **y13**, **y21**, **y22**, **y23**, **y31**, **y32**, **y33**, 1-bit wide inputs **clk**, **reset**, **start**, 20-bit wide output **result** and 1-bit wide output **done**.

3. You will basically design the same 2D-Convolution circuit. But this time you will use Xilinx Language Templates and Device Primitives. That means you will use predefined adder and multiplier circuits instead of your own designs.

4. Click on the Language Templates icon and find a synchronous multiplier for your device. Copy the code and then add a new source called multiplier. Define ports for this module to connect it to multiplier. Paste the primitive multiplier code to new module and instantiate it from the CWDSP module.

5. Because of Spartan3E does not have an Adder Primitive, you have to use Coregen to create such an IP. Add a new source and called adder and choose IP Core Genarator from the Source Wizard.

6. Click the Math Functions from the new wizard and choose Adder-Subtractor IP. Then create your 19-bit adder IP so that it contains clock, reset, cin, cout, clk_enable ports. Instantiate and use it like multiplier.

7. Define needed wires to connect the multipliers and adders each other.

8. The done signal assignment can be done as told in multiplier design.

9. Write down a test code to ensure that your circuit is working correctly.

10. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.

11. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits especially for carry output of last FA. How can you increase the circuit performance?

12. Apply "Design Goals and Strategies" to improve performance of the circuit. You must apply strategies for both area and performance separately. What would happen if we use another type of adder?

13. Add all off details and answers to your report.

Experiment Report

Each group member is going to prepare his/her own report. Reports should include:

- Screenshots, results etc. and your comments about the results.
- What kind of differences have you seen from these two methods? Which one do you prefer? Why? Explain your idea in details and support it from the formal results.
- ISE project directory (rar format).

Reports are to be submitted before next experiment. Submission includes reports and project files.