ISTANBUL TECHNICAL UNIVERSITY

Digital System Design Applications

Experiment 4
ARITHMETIC CIRCUITS

Objectives

- To learn how to design adder and multiplier circuits.
- To learn how to use arithmetic circuits.

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 Arithemetic Circuits Library

Create a new ISE project with the settings below:

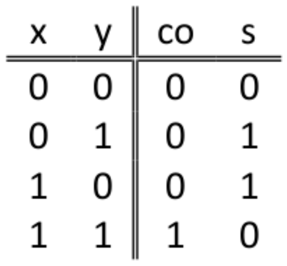
- Family: **Spartan 3E**
- Device: **XC3S500E**
- Package: **FG320**

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

Add a new Verilog module named **arithmetic_circuits** to your project. Clear all lines in your module.

HALF ADDER(20 Minutes)

1. Add a new source to your projects named **arithmetic_circuits.v**
2. Write down a module which is called **HA** into your **arithmetic_circuits.v** file. This module is going to have 1-bit inputs **x** and **y** and 1-bit outputs **cout**(carry out) and **s**(sum). Ensure that your design provides the truth table shown in Figure 1.
3. Use KEEP constraint to keep your wires against optimization.
4. Write a test code to show that your circuit is working correctly.
5. It will make your future works easy to use for structure when writing test code.
6. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.
7. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits.
8. Add all off details to your report.

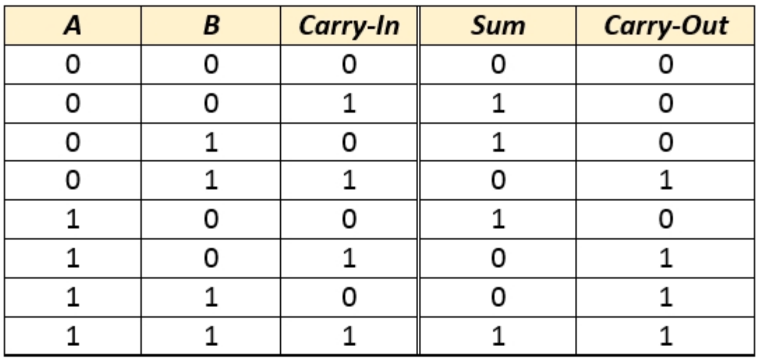


x	y	co	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Figure 1: Half Adder Truth Table

Full Adder (20 Minutes)

1. Write down your full adder circuit to a new module called **FA**. This module is going to have 1-bit inputs **x**, **y**, **ci** (carry in) and 1-bit outputs **cout**(carry out) and **s**(sum). Ensure that your design provides the truth table shown in Figure 2.
2. Design your FA module as it contains 2 HA circuits and 1 OR gate.
3. Use KEEP constraint to keep your wires against optimization.



A	B	Carry-In	Sum	Carry-Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 2: Full Adder Truth Table

4. Write a test code to show that your circuit is working correctly.
5. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.
6. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits.
7. Add all off details to your report.

Ripple Carry Adder (20 Minutes)

1. Write down your ripple carry adder circuit to a new module called **RCA**. This module is going to have 4-bit inputs **x**, **y**, 1-bit carry input **ci** and 1-bit output carry out **cout**, 4-bit sum output **s**.
2. Design your RCA module so that it contains 4 FA circuits as shown in Figure 3.
3. Ensure that your design works correctly.
4. Use KEEP constraint to keep your wires against optimization.
5. Write a test code to show that your circuit is working correctly.
6. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.
7. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits.
8. Add all off details to your report.

Serial Bit Adder(20 Minutes)

1. Write down your serial bit adder circuit to a new module called **SBA**. This module is going to have 4-bit inputs **x**, **y**, 1-bit carry input **ci**, 1-bit permission input **add** and 1-bit output carry out **cout**, 4-bit sum output **s**. A structure is shown in Figure 4.
2. Define 4-bit wide 3 registers which have initial value 0(zero) and named xReg, yReg,sReg.

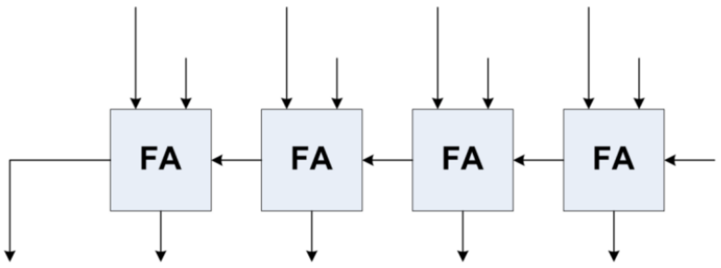


Figure 3: Ripple Carry Adder

3. Define 1-bit wide 1 register which have an initial value 0(zero) and named cReg.
4. Use a FA in your design. Connect least significant bits of xReg, yReg, to x and y input of FA. Connect a wire named wc to carry out of FA and a wire named ws to sum output of FA. Connect output of cReg to co output of your circuit.
5. You must have a clock input named clk.
6. When each positive clock edge in which the add input is 1:
   - Make most significant bit of xReg and yReg 0(zero) and shift them 1-bit left.
   - Save value of ws to most significant bit of sReg and shift it 1-bit rigth.
   - Save value of wc to cReg.
7. When each positive clock edge in which the add input is 0:
   - Save value of x input to xTut and value of y input to yReg.
   - Save value of ci to cReg.
   - Keep value of sReg.
8. Use behavioral coding method with always for this design.
9. Ensure that your design works correctly.
10. Use KEEP constraint to keep your wires against optimization.
11. Write a test code to show that your circuit is working correctly.
12. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.
13. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits especially the last FA's co!.
14. Add all off details to your report.

Adder Circuits For Signed Numbers (40 Minutes)

1. Write down your igned ripple carry adder circuit code to a new module called **SRCA**. Use the same RCA circuit for this design. But this time, you must design your circut so that it avoids the arithmetic overflow by interpreting the carry outputs of the last two FA!

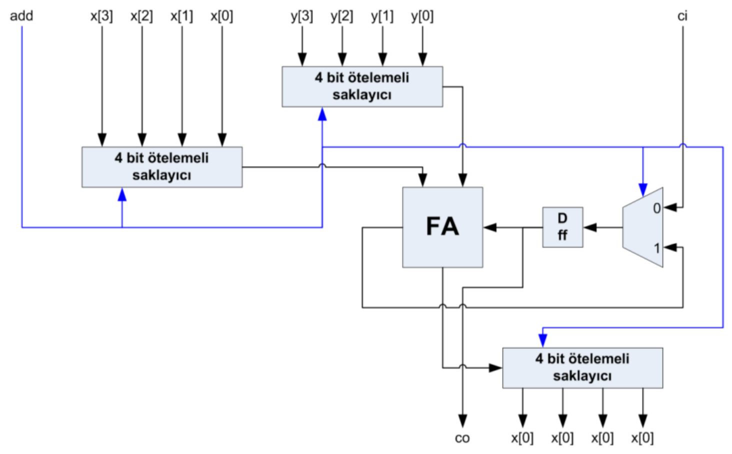

add
x[3]
x[2]
x[1]
x[0]
y[3]
y[2]
y[1]
y[0]
ci
4 bit ötelemeli saklayıcı
4 bit ötelemeli saklayıcı
FA
D ff
0
1
4 bit ötelemeli saklayıcı
co
x[0]
x[0]
x[0]
x[0]

Figure 4: Serial Bit Adder

2. Ensure that your design works correctly.
3. Use KEEP constraint to keep your wires against optimization.
4. Write a test code to show that your circuit is working correctly. In your test code:
   - Make ci input zero(0) for every time.
   - Change value of x and y input from -8 to +7. (4-bit wide!)
   - Concatanete the sum output and the carry output. Then, assign result to 5-bit wide wire which is defined as signed!
5. Synthesize your design and implement it with the processes, translate, map and Place and Route (PAR), respectively.
6. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? Investigate the delays of your circuits especially the last FA's co!.
7. Add all off details to your report.

Block Multiplier (20 Minutes)

1. Write down your signed block multiplier circuit code to a new module called **BM**. Define the basic multiplier that shown bottom of Figure 5 in your MB module.
2. Define a new module called **M**. The M module must have 4-bit wide x and y inputs and 8-bit wide output p.
3. Design your 4x4 block multiplier so that the p would be product of **x** and **y**.
4. Ensure that your design works correctly.
5. Use KEEP constraint to keep your wires against optimization.
6. Write a test code to show 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? Investigate the delays of your circuits.

9. Add all off details to your report.

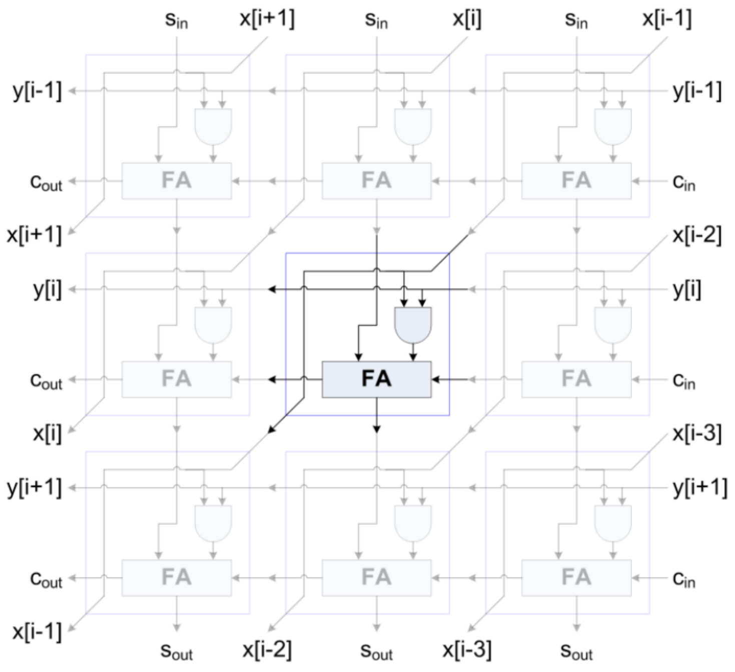


Figure 5: Block Multiplier

Experiment Report

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

- Screenshots and results asked above.
- Your comments on results.
- ISE project directory (rar format).

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