

Digital System Design Applications

Experiment IV ARITHMETIC CIRCUITS 2022

Preliminary

Students should know about arithmetic circuits.

Objectives

- To learn how to design arithmetic circuits.
- Usage of testbench code for making simulation

Requirements

Students are expected to be able to

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

Experiment Report Checklist

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

1. Half Adder

- Half Adder Verilog code
- Test code for Half Adder
- RTL and Technology schematic for Half Adder. How many LUTs does your design use? Add delays of the circuit.

2. Full Adder

- Full Adder Verilog code
- Test code for Full Adder
- RTL and Technology schematic for Full Adder. How many LUTs does your design use? Add delays of the circuit.

3. Ripple Carry Adder

- Ripple Carry Adder Verilog code
- Test code for Ripple Carry Adder
- RTL and Technology schematic for Ripple Carry Adder. How many LUTs does your design use? Add delays of the circuit.

4. Ripple Carry Adder with "generate-for"

- Parametric Ripple Carry Adder Verilog code
- Testbench codes, sims and results for 4 bit and 6 bit versions
- RTL and Technology schematic for parametric Ripple Carry Adder. How many LUTs does your design use? What structural differences observed in schematics, compared to standard 4-bit RCA?

5. Carry Lookahead Adder

- Generate function (g) and propagate (p) function for all bits(from LSB to MSB) of your 4-bit wide adder
- c1, c2, c3, c4 carry outputs by using c0, p and g expressions
- Show that for all bits, the value of a bit is obtained by an EXOR operation of x, y and carry input of that bit
- Carry Lookahead Adder Verilog code
- Test code for Carry Lookahead Adder
- RTL and Technology schematic for Carry Lookahead Adder. How many LUTs does your design use? Add delays of the circuit.

6. Adder-Subtractor Circuit

- Adder-Subtractor Circuit with Overflow detection Verilog code
- Test code for Adder-Subtractor for all cases
- RTL and Technology schematic for Adder-Subtractor. How many LUTs does your design use? Add delays of the circuit.
- Your comments on the results

- **Projects and reports are to be done INDIVIDUALLY. High amounts of points will be deducted from similar works.**
- **Reports must be written in a proper manner. Divide your text to sections and sub-sections if needed, label your figures and connect your sections with proper explanations of your works. Reports filled with imprecisely placed tables and figures, with no verbal explanations in workflow, will not fare well.**
- **Check homeworks section in Ninova for submission dates. There will be two different submissions for project archive folder and a PDF report file.**

Creating Arithmetic Circuits Library

Create a new Vivado project with the settings below:

- Download and extract this Zip file from the link below
<https://reference.digilentinc.com/reference/software/vivado/board-files>
- Copy the files that in this directory "vivado-board-master>new>boards-files"
- Paste the files to C:Xilinx>Vivado>20xx.x>data>boards>board-files
- Restart Vivado
- You are now ready to use a Vivado project for the Digilent Nexys 4 DDR and Nexys A7 Boards.
- You can select an FPGA board on which you want to study.

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

Half Adder

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. Write a test code to show that your circuit is working correctly.
4. It will make your future works easy to use for structure when writing test code.
5. Synthesize your design and make implementation step.
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.

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

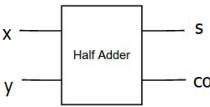


Figure 1: Half Adder Truth Table and block representation

FULL Adder

1. Write down full adder circuit to a new module called FA into your arithmetic_circuits.v file. This module is going to have 1-bit inputs x, y, ci (carry in) and 1-bit outputs cout(carry out) and s(sum).
2. Design your FA module as it contains 2 HA circuits and 1 OR gate. Draw it by hand or with a computer program. Then write the verilog code
3. Ensure that your design provides the truth table shown in Figure 2.
4. Write a test code to show that your circuit is working correctly.

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

5. Synthesize your design and make implementation step.
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

1. Write down your ripple carry adder circuit to a new module called RCA into your arithmetic_circuits.v file.
2. 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.
3. Design your RCA module so that it contains 4 FA circuits as shown in Figure 3.
4. Ensure that your design works correctly.

5. Write a test code to show that your circuit is working correctly.
6. Synthesize your design and make implementation step.
7. 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.
8. Generate BIT file and program your FPGA. Show that your circuit is working correctly.
9. Add all off details to your report.

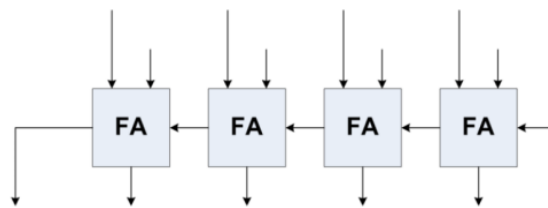


Figure 3: Ripple Carry Adder

Ripple Carry Adder Using "parameter" and "generate-for"

1. Write down your ripple carry adder circuit to a new module called **parametric_RCA**. This module should include a parameter named **SIZE**, to represent number length. By only changing the **SIZE** parameter, a user must be able to easily synthesise an adder in specific length. Thus, define your inputs **x,y**; and your output **s**, with their length depending on **SIZE** parameter. The module also needs to have a one bit "**ci**" input and one bit "**cout**" output.
2. Design your RCA module similarly to the previous step. But this time, use **generate-for** struct and **SIZE** parameter to come up with a rule to connect multiple FAs, using Figure 3. How many FAs used should be solely determined by **SIZE** parameter. "**generate-for**" struct must make the proper connections automatically.
3. Set your size parameter to 4, and test your circuit for five different 4-bit number pairs. Show your simulation results clearly.
4. Synthesize your design and make implementation step.
5. View RTL, Technology schematics, and investigate Design Summary. How many LUTs does your design use? What structural differences you observe at RTL and Technology schematics, compared to your 4-bit RCA module?
6. Now Set your size parameter to 8, and test your circuit for five different 8-bit number pairs. Show your simulation results clearly.
7. Generate BIT file and program your FPGA. Show that your circuit is working correctly. (You can use buttons and 7 segments for extra bits.)
8. Add all off details to your report.

Carry Lookahead Adder

1. Research the Carry Lookahead Adder circuit from Digital Design, 5th Edition By M. Morris Mano And Michael Ciletti.
2. Write down your carry lookahead adder circuit to a new module called CLA. This module is going to have 4-bit inputs x, y, 1-bit carry input c0 and 1-bit output carry out cout, 4-bit sum output s.
3. Write down g (generate function), p (propagate function) functions on a sheet for all bits (from LSB to MSB) of your 4-bit wide adder.
4. Obtain c1, c2, c3, c4 carry outputs by using c0, p and g expressions.
5. Show that the value of sum bits are obtained by EXOR operation of x, y and carry input of that bit.
6. Write down all of these functions by using assign and logic operators to your CLA module.
7. Ensure that your design works correctly.
8. Write a test code to show that your circuit is working correctly
9. Synthesize your design and make implementation step.
10. 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.
11. Generate BIT file and program your FPGA. Show that your circuit is working correctly.
12. Add all off details to your report

Adder-Subtractor Circuit with Overflow detection

1. You will design a 4-bit adder-subtractor circuit to add or subtract the **signed binary** numbers. Research this circuit and **arithmetic overflow** from Digital Design, 5th Edition By M. Morris Mano And Michael Ciletti.
2. The negative signed numbers must be shown in two's complement form.
3. You must design your circuit so that it detects the arithmetic overflow by interpreting the carry outputs of the last two FA!
4. The circuit must be design as given in Figure 4. Which operator should be used in the marked boxes in the figure?
5. Write down a module which is called Add_Sub. This module is going to have 4-bit inputs A and B, 1-bit carry input 4-bit output SUM, 1-bit output Cout and 1-bit output V to show overflow. Obtain the circuit using FA's from your arithmetic_circuits.v file.
6. Write a test code to show that your circuit is working correctly. In your test code:
 - Change the value of A and B inputs from -8 to +7. (4-bit wide!)
 - Change the value of Cin from 0 to 1.

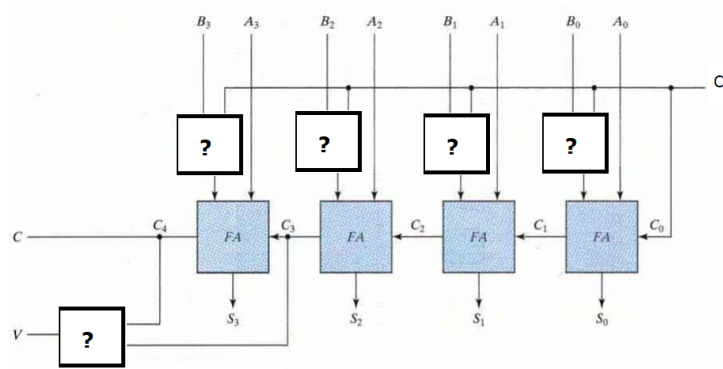


Figure 4: The 4-bit Adder Subtractor Circuit with Overflow Detector

- Show all cases
 - Show the carry outputs of the last two FA on simulation so we can check if V signal is working as expected
 - Observe SUM when overflow occurs ($V=1$) and when it doesn't ($V=0$)
7. Synthesize your design and make implementation step.
 8. 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!.
 9. Generate BIT file and program your FPGA. Show that your circuit is working correctly.
 10. Add all off details to your report.

References:

1. Digital Design, 5th Edition By M. Morris Mano And Michael Ciletti
2. Nexys4DDR Reference Manual
3. Constraints Guide
4. Testbench code