MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EXPERIMENT 4. Parallel Adders, Subtractors, and Complementors

I. Introduction

I.a. Objectives

In this experiment, parallel adders, subtractors and complementors will be designed and investigated. In the first and second parts of the experiment you will implement your circuits using ICs and connecting them on the breadboard. In the rest of the experiment, you will use Quartus II 14.1 software and FPGA to implement the circuits. In this experiment, you need to download your designs to the FPGA and check the results by physical means, i.e., using LEDs and oscilloscope. Another objective of this experiment is to expose you the hierarchical design method for logic circuits.

I.b. Background

Digital computers perform a number of arithmetic operations for information processing. These tasks are performed using various arithmetic logic circuits. The most commonly used basic arithmetic circuits are adders, subtractors and complementors. A short description of these circuits is given below.

Adders*:*
Adders are divided into two groups: half adders and full adders. Full adders are used to add three bits where one of them is carry from the preceding adder. They have two outputs: sum and carry to the next stage. In half adders, only two inputs are considered as operands; hence carry inputs are ignored. The truth table of a full adder is given in Table 4.1. Two of the input variables, denoted by X_k and Y_k, represent two significant bits to be added. The third input, C_{k-1} represents the carry from the previous lower significant position.

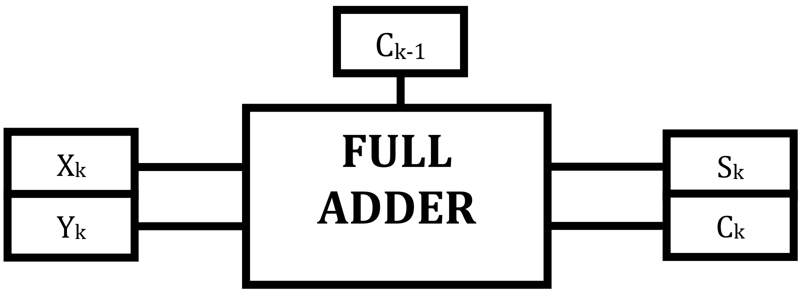


Figure 4.1 Block diagram of a full adder

Table 4.1 Truth table of a full adder (X_k+Y_k)

X_k	Y_k	C_{k-1}	S_k	C_k
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

Subtractors:
Subtractors are similar to adders. There are full subtractors with three inputs one of which is the 'borrow' from the preceding subtractor. The two outputs are difference and borrow to the succeeding unit. Half subtractors do not have a borrow input. Figure 4.2 shows the block diagram of a full subtractor and Table 4.2 gives its truth table.

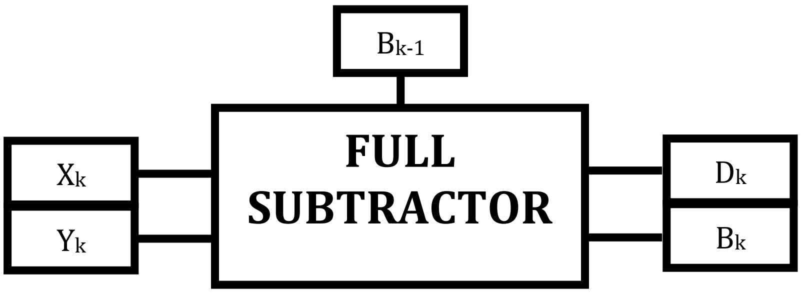


Figure 4.2 Block diagram of a full subtractor

Table 4.2 Truth table of a full subtractor (X_k-Y_k)

X_k	Y_k	B_{k-1}	B_k	D_k
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Complementors:
Complementor units are of two types: 1's complement units and 2's complement units. The truth tables for 2-bit 1's complement and 2's complement units are given in Table 4.3.

Table 4.3 Truth table of complementors

Input Number	1's Complement Unit	2's Complement Unit
00	11	00
01	10	11
10	01	10
11	00	01

Note: 2's complement of a number is its 1's complement plus binary one.

The arithmetic units of computers usually employ the principles of either 2's complement arithmetic or 1's complement arithmetic. Such units accept two operands (positive or negative) and perform addition. If subtraction is desired subtrahend must first be complemented, then added. 2's complement arithmetic is the most widely used register arithmetic. Some examples are given below:

i. Sign-magnitude arithmetic:

```
    +7      +00111        +7      +00111
  + +5    + +00101      - +5    - +00101
   +12      +01100        +2      +00010
```

ii. 1’s complement arithmetic:

```
   +7            0111
+  -5        +   1010  (1’s complement of 0101)
  ----       -----------
   +2          1← 0001
             +     → 1
             -----------
                 0010
```

iii. 2’s complement arithmetic:

```
   +7            0111
+  -5        +   1011  (2’s complement of 0101)
  ----       -----------
   +2          1← 0010  (Ignore the carry bit.)
```

In signed magnitude arithmetic, the most significant bit of an n-bit word is the sign bit. The magnitude can be determined from the remaining n-1 bits. If the result of a 2’s complement arithmetic operation exceeds the range of the available bit length (overflow for a large positive number or for a small negative number), then a wrong (invalid) result will be obtained. The presence of an erroneous result can be detected by the help of a combinational circuit.

II. Preliminary Work

1. Read sections 1.5 and 4.4 from the textbook “Digital Design” by M. Mano (3rd Ed., 2002, Prentice Hall).

2. Design a half adder and a half subtractor using minimum number of gates using two-input NAND (7400) and two-input XOR (7486). State the truth tables of both of your designs. Draw and explain your designs in detail.

3. Design a full adder and a full subtractor using minimum number of two-input NAND (7400) and two-input XOR (7486) gates. Show how you can use half adders and half subtractors to build a full adder and full subtractor. Show and explain your designs in detail.

4. Design a 4-bit binary adder by using full adders. You don’t have to draw each full adder in gate level. Instead, you can put full adder blocks with appropriate input and output connections. At the end, your design should include 9 inputs representing two 4-bit numbers and a carry in, and 5 outputs representing one 4-bit number (which is the sum) and a carry out.

5. Design a 4-bit 2’s complement unit with a control signal E such that

When E=0 output is the 2’s complement of the input;
When E=1 output is the same as the input.

You can use two-input NAND (7400), 4-bit binary adder (7483) and two-input XOR (7486) gates.

6. Design an arithmetic unit that accepts 4-bit wide parallel numbers (X, Y) and an enable signal (E). Assume that input numbers are represented in 2’s complement arithmetic. Enable signal E has a function such that

When E=1 unit performs addition (X+Y);
When E=0 unit performs subtraction (X-Y).

You can use two-input NAND (7400), 4-bit binary adder (7483) and two-input XOR (7486) gates.

7. Consider the circuit you designed in Part 6. Prepare a table showing the state of the outputs when input numbers and control signal are given as shown in Table 4.4.

Table 4.4 Input signals that should be applied to the circuit designed in part 6

E=1 X=0110 Y=0011	E=1 X=1010 Y=0111
E=1 X=0111 Y=0001	E=1 X=-5 Y=8
E=0 X=0110 Y=0001	E=0 X=1001 Y=0110
E=0 X=0011 Y=0100	E=0 X=-1 Y=-14

8. In this experiment, you are expected to learn the fundamentals of hierarchical logic design and use bus structures as inputs and outputs. This question's motivation is to give you an insight about the experimental work.

Consider the 4-bit Fibonacci Number Checker that you have designed in the Preliminary Work of Experiment 3.

- **Implement** it on Quartus II software.
- Using the experimental work part of the Experiment 4 (**III.b.2 Creating the symbol**) **create a symbol** of your design (namely, Fibonacci Number Checker block, FNC block).
- Using that symbol, **design** and **implement** a 4-bit Not-A-Fibonacci Number Checker (which gives 1 as the output when the input is not a Fibonacci number and vice versa).
  Note: Your design can be composed of a NOT gate and an FNC block.
- **Create** a 4-bit bus structure with the help of the experimental work of this document (**III.c.2 Creating a Bus**). This bus will be used as the input of your Not-An-FNC circuit.
- **Simulate** your design with different inputs. The details of including bus structured inputs to the simulation window are given in the same experimental procedure.
- **Add** your results to your preliminary work. Schematic designs of both FNC and Not-An-FNC, and the simulation results should be attached.

III. Experimental Work

III.a. Testing Your Designs on the Breadboard

1. Construct and test the half adder and half subtractor that you designed in Part 2 of the preliminary work using ICs connected onto a breadboard.

2. Construct and test the full adder and full subtractor that you designed in Part 3 of the preliminary work using ICs connected onto a breadboard.

III.b. Designing 4-Bit Binary Adder Using Hierarchical Design Method

3. For the rest of the experiment, you will use Quartus II software and Altera FPGAs. As you remember, you were required to make the designs of Part 4 and Part 5 of the preliminary work using 7483 IC, namely the 4-bit binary adder. Now, you will design a 4-bit binary adder and form a component out of it using the hierarchical design method. Then, you will add it to the symbol directories of your higher level designs that you will perform later in the experiment, and use it. Note that only XOR and NAND gates will be used in the design of your 4-bit binary adder.

   In this part –until you learn how to design hierarchically-, you will present your understanding of proper FPGA and Quartus usage with minimally guided study.

III.b.1 Implementing the full-adder

4. Create a new schematic file and save it as **fulladder** in the folder **fulladder** in the folder **exp4** on the desktop. Don't forget to click "add file to current project" option.

   Note: We will enter the hierarchical design as multiple projects. We'll start with the lowest-level block and then work our way up to the top-level block. The lowest-level block in this work is the full adder. Each different block design will be represented as a new project. Hierarchical designs may also be entered as a single project but lower level block testing is more difficult.

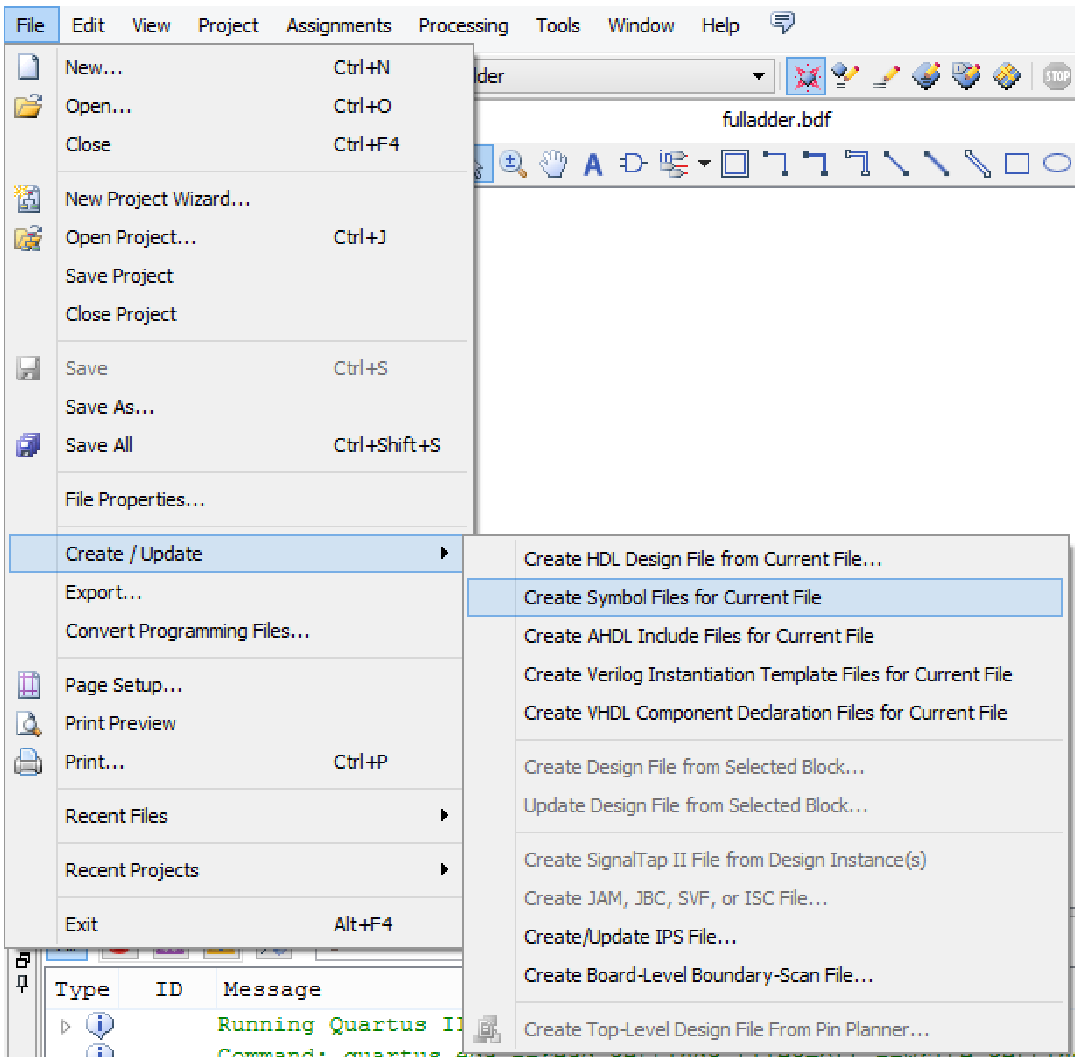
5. Based on the logic circuit design in your Preliminary Work Part 3, create the full adder by adding appropriate components, input/output pins and wiring.

6. Save the schematic with the name **fulladder** and compile the design.

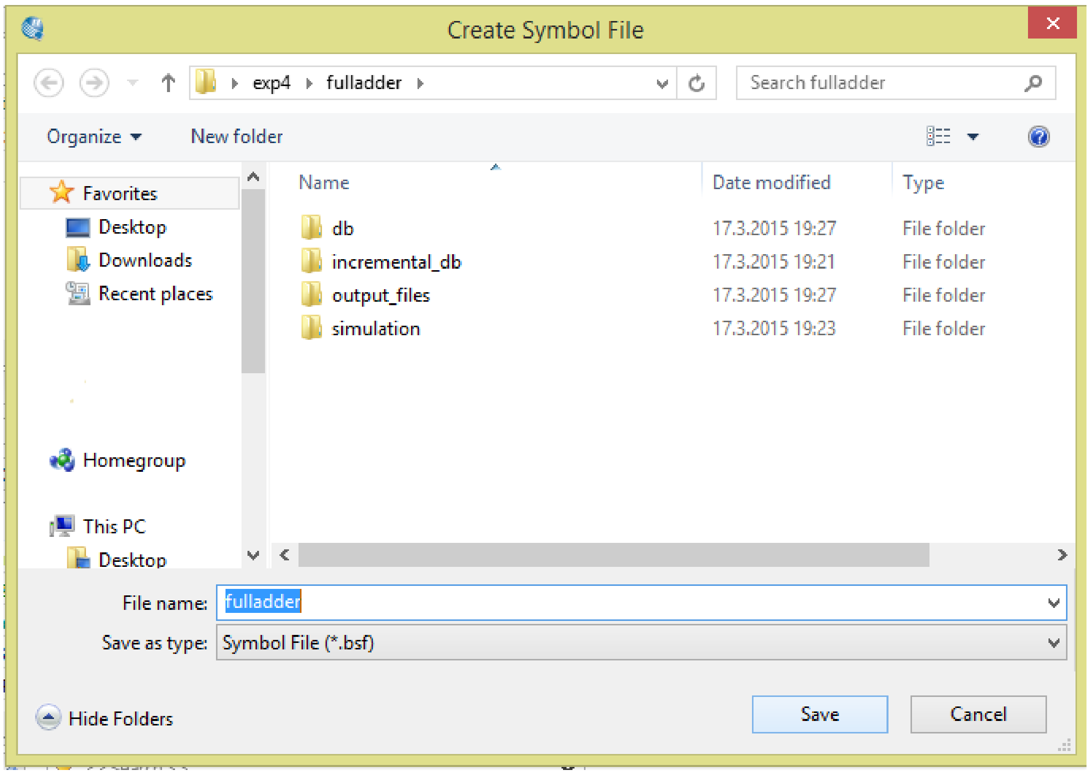
III.b.2 Creating the symbol

7. After determining that the current project functions correctly in the simulation, you will need to create a symbol for this block to be able to use it in a higher-level of the design hierarchy.

8. Open **File > Create/Update > Create Symbol Files for Current File** as it's shown on Figure 4.3.


File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
New... Ctrl+N
Open... Ctrl+O
Close Ctrl+F4
New Project Wizard...
Open Project... Ctrl+J
Save Project
Close Project
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
File Properties...
Create / Update
Export...
Convert Programming Files...
Page Setup...
Print Preview
Print... Ctrl+P
Recent Files
Recent Projects
Exit Alt+F4
Create HDL Design File from Current File...
Create Symbol Files for Current File
Create AHDL Include Files for Current File
Create Verilog Instantiation Template Files for Current File
Create VHDL Component Declaration Files for Current File
Create Design File from Selected Block...
Update Design File from Selected Block...
Create SignalTap II File from Design Instance(s)
Create JAM, JBC, SVF, or ISC File...
Create/Update IPS File...
Create Board-Level Boundary-Scan File...
Create Top-Level Design File From Pin Planner...
fulladder.bdf
Type
ID
Message
Running Quartus II
Figure 4.3 Symbol creation window

9. This symbol should have the same file name as the design file (fulladder) but will have a *.bsf* file extension as it's shown on Figure 4.4. It will be saved in the same project folder. Click **Save**. Click **OK**.



Name	Date modified	Type
db	17.3.2015 19:27	File folder
incremental_db	17.3.2015 19:21	File folder
output_files	17.3.2015 19:27	File folder
simulation	17.3.2015 19:23	File folder

File name: fulladder

Save as type: Symbol File (*.bsf)

Figure 4.4 Saving newly created symbol

III.b.3 Starting a New Design Project for the Higher-Level Block

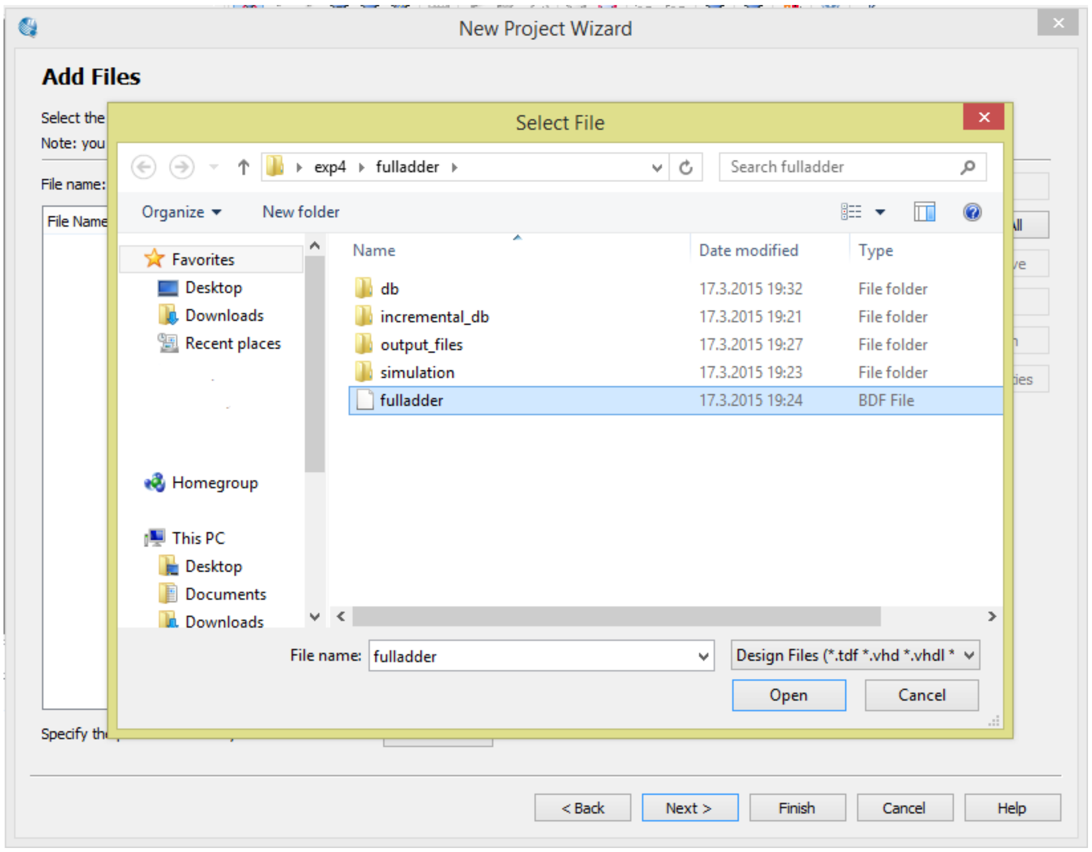
Note: This hierarchical design is being entered as multiple projects. We will now work on a higher-level project. The higher-level block in this work is 4-bit adder. Each different block design will be represented as a new project. Lower-level design blocks will be contained in this project.

10. Create a new directory in the folder exp4 with the name **fourbitadder.** Then **create** a new schematic file and save it as **fourbitadder** in the folder **fourbitadder** in the folder **exp4** on the desktop. Don't forget to click "add file to current project" option.

Note: A new directory should be created for each separate design project. This will be a different directory name then was used for the lower-level project that we just completed.

Note: The project name must be different than the lower-level project. We cannot use duplicate design file names in a project and the lower-level project will be contained in this new project.

11. When the **Add Files** dialog box is opened, click the **Browse** button to locate the working directory for the lower-level project as it's shown on Figure 4.5. We will now identify the files to be included in this top-level project.



Name	Date modified	Type
db	17.3.2015 19:32	File folder
incremental_db	17.3.2015 19:21	File folder
output_files	17.3.2015 19:27	File folder
simulation	17.3.2015 19:23	File folder
fulladder	17.3.2015 19:24	BDF File

File name: fulladder
Design Files (*.tdf *.vhd *.vhdl *
Open
Cancel

Figure 4.5 Adding symbol file to the new project

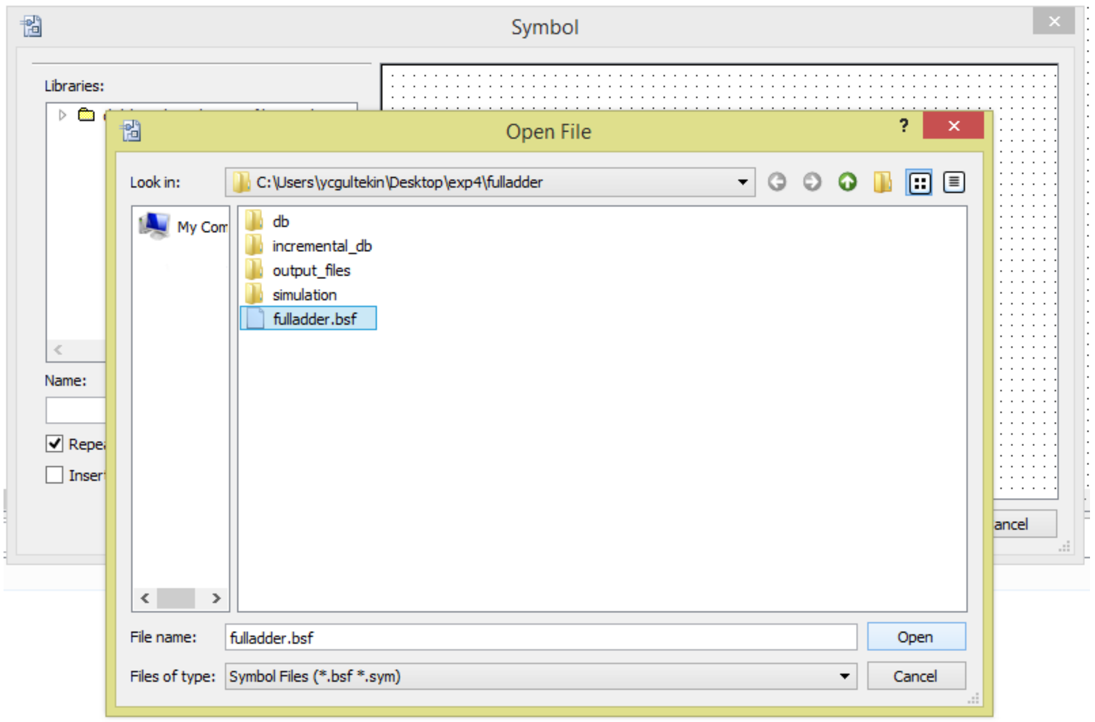
12. Use the Select File dialog box to locate the lower-level project. Remember that it will be in a different folder. Navigate to the lower-level project directory (**...desktop/exp4/fulladder** which probably is already selected). Select the project name (**fulladder**) and click **Open**. As the file name for the design file of the lower-level project appears in the box, click **Add** to move this file name to the list of selected files.

13. Click **Next**, and complete the usual procedure.

III.b.4 Implementing the 4-bit binary adder

14. Based on the logic circuit design in your Preliminary Work Part 4, create the 4-bit binary adder by adding appropriate components, input/output pins and wiring.

Note: To use the lowest-level design (fulladder); click the Symbol Tool button (gate symbol) on the top of the Block Diagram/Schematic File editor window. Click **Browse** button (...) and locate the project directory (**...desktop/exp4/fulladder**) of the lowest-level **fulladder** file as it's shown on Figure 4.6. **fulladder** file selected, click **Open**.


Look in: C:\Users\ycgultekin\Desktop\exp4\fulladder

db
incremental_db
output_files
simulation
fulladder.bsf

File name: fulladder.bsf
Files of type: Symbol Files (*.bsf *.sym)

Figure 4.6 Using previously created symbol

Note: By the way, you can view the schematic for the lowest-level fulladder block by double clicking on any one of the block symbols. The schematic will open in another Block Diagram/Schematic File window.

Full-adder block should be seen like in the Figure 4.7.

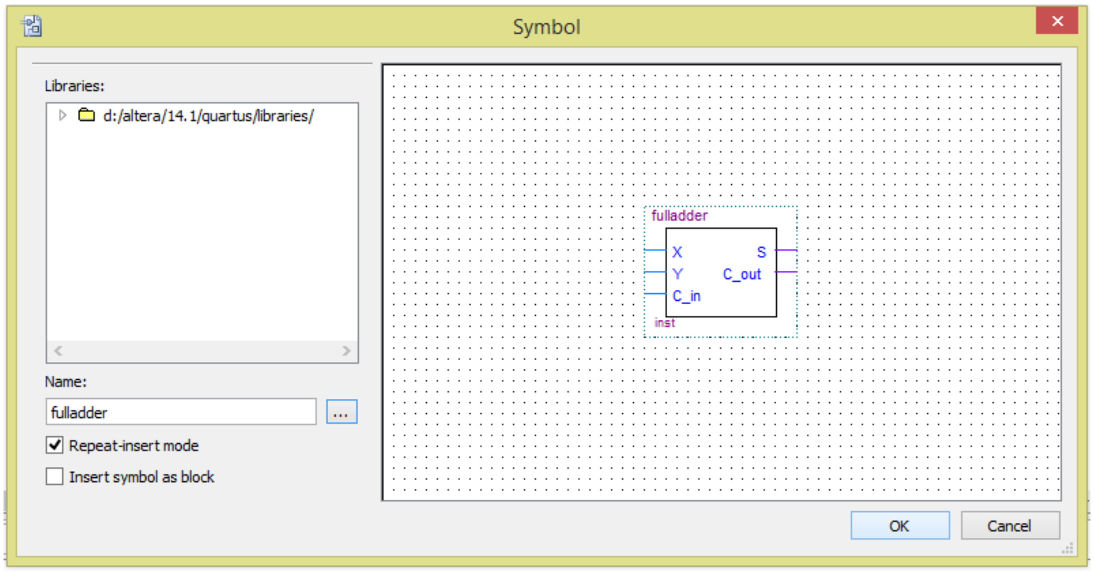


Figure 4.7 Full-adder symbol in your library

15. Save the schematic with the name **fourbitadder** and compile the design.

III.b.5 Creating the symbol

16. By the same method that you use for the full-adder symbol, create symbol for 4-bit binary adder. This symbol should have the same name as the design file (fourbitadder) but will have a *.bsf* file extension. It will be saved in the same project folder. Click **Save**. Click **OK**.

III.c. Designing the Complementor

Note: This hierarchical design is being entered as multiple projects. We will now work on the top-level projects. The first top-level block in this work is the Complementor. Each different block design will be represented as a new project. Lower-level design blocks will be contained in these projects.

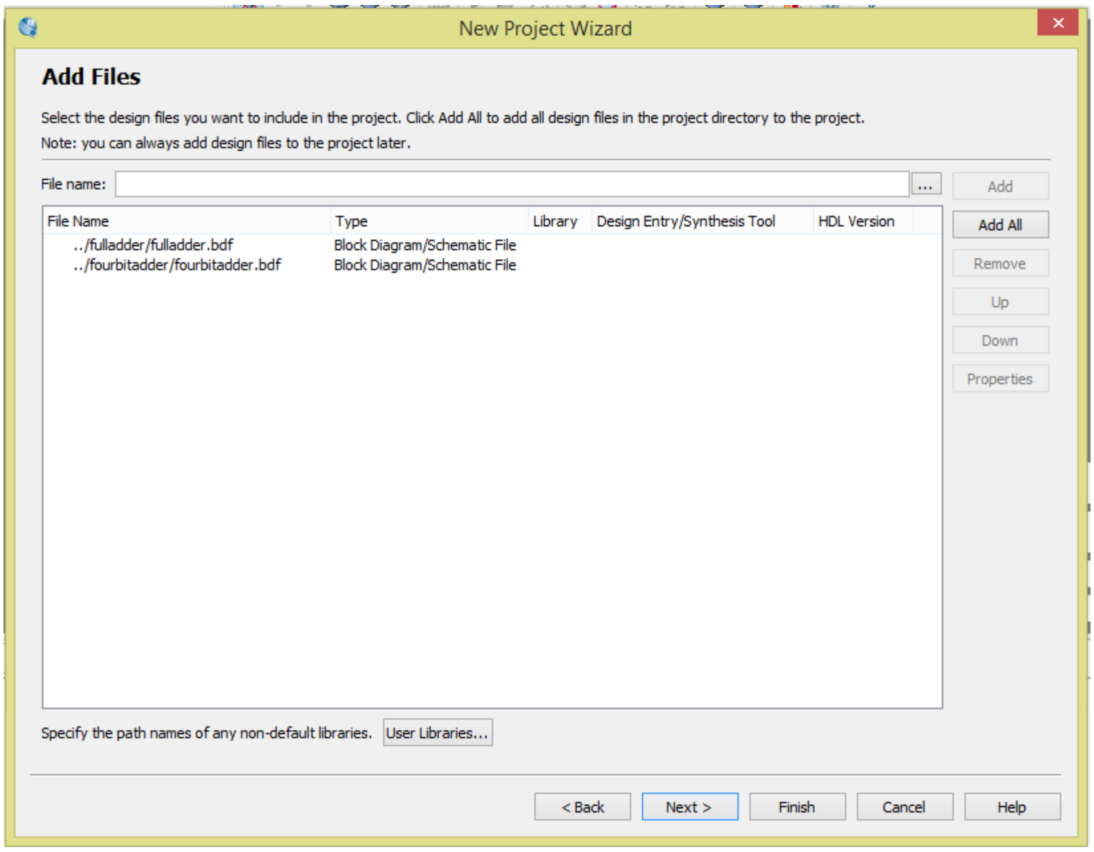
17. Create a new directory in the folder exp4 with the name **fourbitadder.** Then create a new schematic file and save it as **complementor** in the folder **complementor** in the folder **exp4** on the desktop. Don't forget to click "add file to current project" option.

Note: A new directory should be created for each separate design project. This will be a different directory name then was used for the lower-level project that we just completed.

Note: The project name must be different than the lower-level project. We cannot use duplicate design file names in a project and the lower-level project will be contained in this new project.

18. When the **Add Files** dialog box is opened, click the **Browse** button to locate the working directory for the lower-level project. We will now identify the files to be included in this top-level project.

19. Use the Select File dialog box to locate the lower-level project. Remember that it will be in a different folder. Navigate to the lower-level project directory (**...desktop/exp4/fourbitadder** which probably is already selected). Select the project name (**fourbitadder**) and click **Open**. As the file name for the design file of the lower-level project appears in the box, click **Add** to move this file name to the list of selected files. Repeat this procedure for **fulladder** project, and add it too. At the end, **Add Files** dialog box should be seen like in Figure 4.8.


New Project Wizard

Add Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.
Note: you can always add design files to the project later.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version
../fulladder/fulladder.bdf	Block Diagram/Schematic File			
../fourbitadder/fourbitadder.bdf	Block Diagram/Schematic File			

Specify the path names of any non-default libraries. User Libraries...

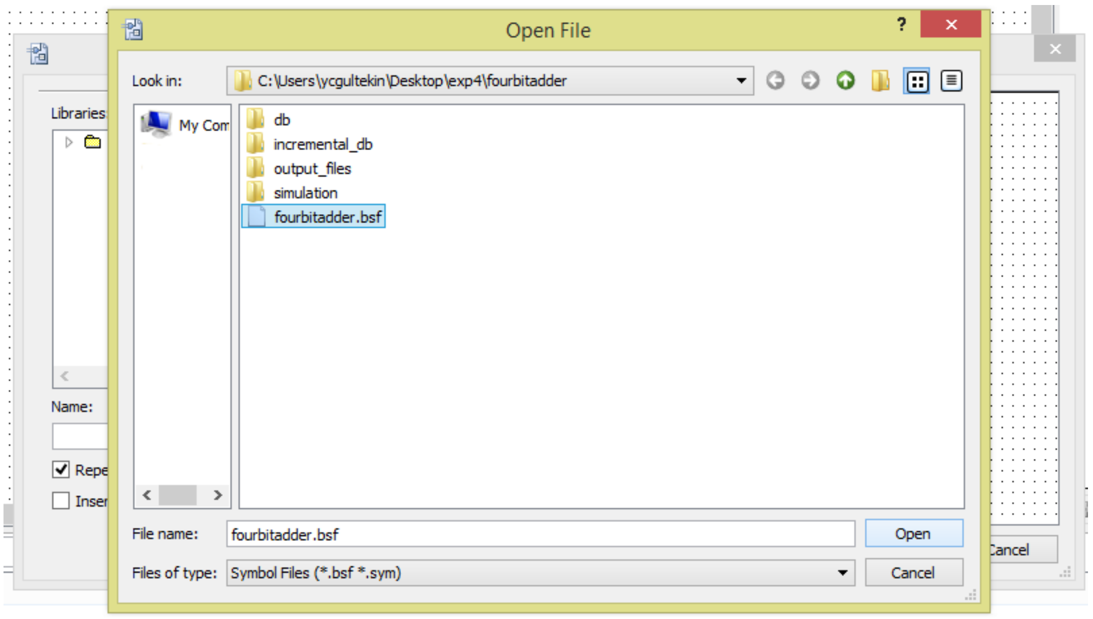
Figure 4.8 Add Files dialog box when you add low-level projects

20. Click **Next**, and complete the usual procedure.

III.c.1 Implementing the Complementor

21. Based on the logic circuit design in your Preliminary Work Part-5, create the 4-bit 2's complementor by adding appropriate components, input/output pins and wiring. Assume that you have a 4-bit binary number $X_3X_2X_1X_0$. This unit must result in $Z_3Z_2Z_1Z_0$ (which is the 2's complement of $X_3X_2X_1X_0$), when the control bit E is 0, and should give the input as output when E is 1.

Note: To use the low-level design (fourbitadder); click the Symbol Tool button (gate symbol) on the top of the Block Diagram/Schematic File editor window. Click **Browse** button (...) and locate the project directory (**...desktop/exp4/fourbitadder**) of the lower-level **fourbitadder** file as it's shown on Figure 4.9. Fourbitadder file selected, click **Open**.



Look in: C:\Users\ycgultekin\Desktop\exp4\fourbitadder

fourbitadder.bsf

File name: fourbitadder.bsf

Files of type: Symbol Files (*.bsf *.sym)

Figure 4.9 Using previously created symbol

Full-adder block should be seen like in the Figure 4.10.

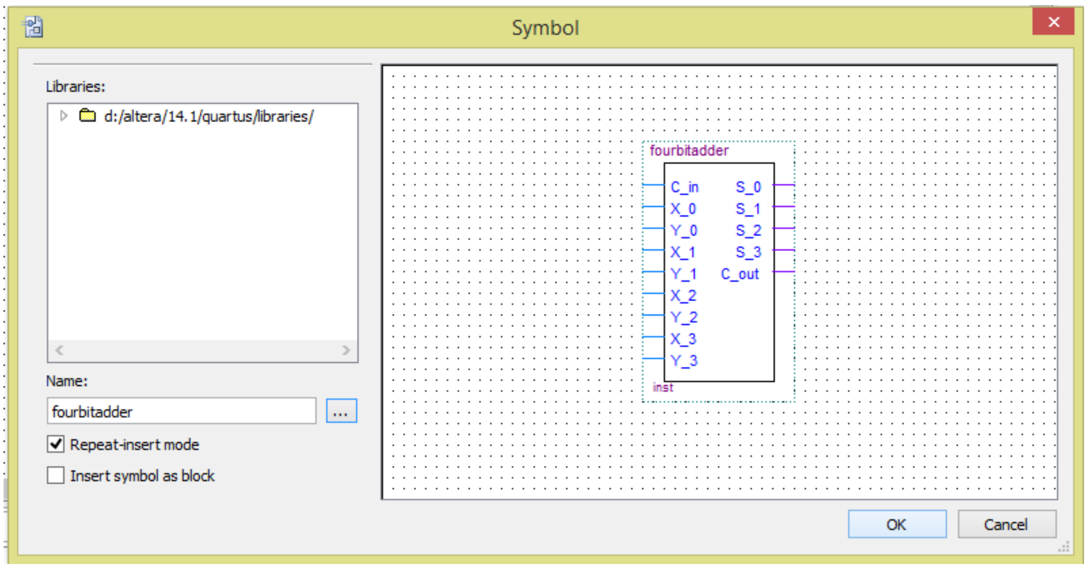


Figure 4.10 4-Bit Binary Adder symbol in our library

22. By using **fourbitadder** symbol, draw the schematic of your complementor design. You may (or will) need a constant "0" (zero, low) or a constant "1" (one, high) to use as an input etc. You may use a 0 (GND) or a 1 (V_{cc}) which can be found in **Other** category in the **primitives** tab of the **Symbol box.**

III.c.2 Creating a Bus

23. You will need to use **busses** in the circuit schematic to represent input/output pins. A bus is basically a wire that represents more than one inputs or outputs. To In Quartus II, a bus is named as, for example, A[3..0], which means the bus is 4-bit, and composed of the signals A_3,A_2,A_1, and A_0 where A_3 is the most significant bit.

24. To create a bus, you should put an input pin from the pins library of the Symbol Wizard box.

25. Name this input as '**X[3..0]**' as it's shown on Figure 4.11. You can easily do this by right-clicking the input object, selecting 'properties', and changing the name.

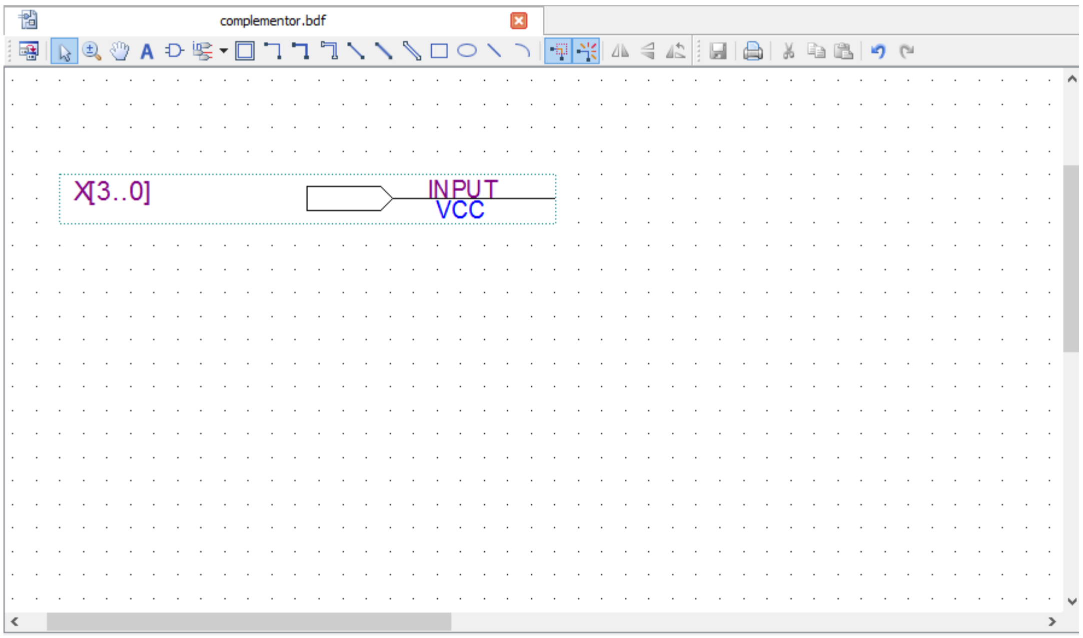


Figure 4.11 Input pin for input bus

26. Add a bus to the input as it's shown in Figure 4.12. Do this by selecting the bus symbol (like the net symbol but with a thicker line, 'Orthogonal Bus Tool') and drawing the bus line as in the same figure. Name this bus as '**X[3..0]**'.

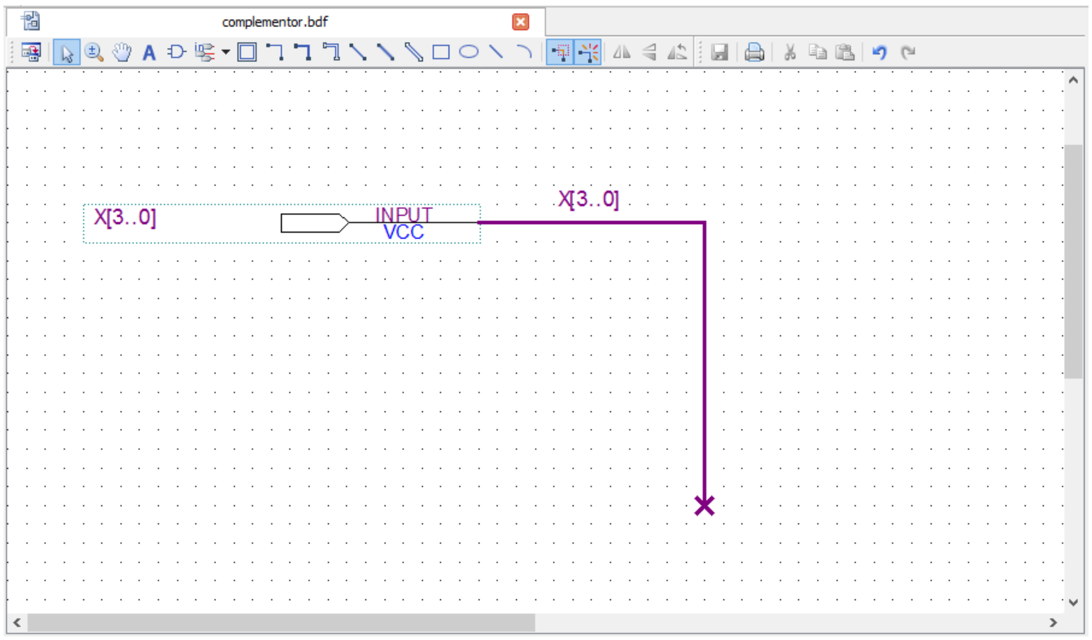


Figure 4.12 4-bit bus with its input pin connected

27. Now connect the inputs of your design to the bus, and name the nets properly as X[1], X[2] etc. as its shown on Figure 4.13.

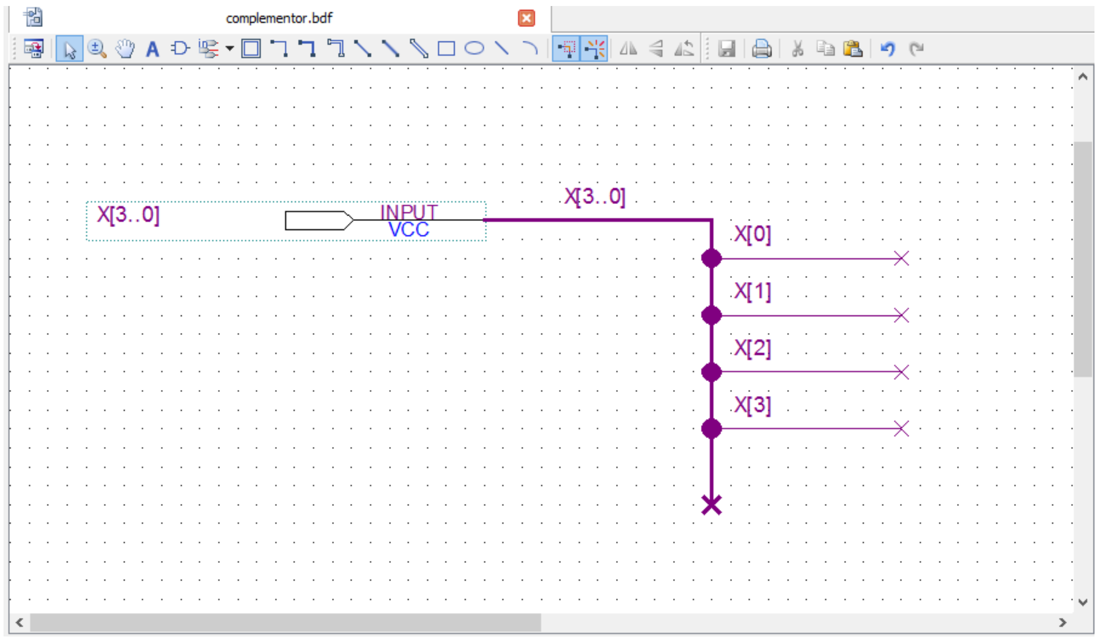


Figure 4.13 4-bit bus with its input pin and connections

28. Apply the same methodology to create output bus.

29. Your bus design should be seen as in Figure 4.14. (Of course there should be your complementor design in between them with wires etc. connected accordingly.)

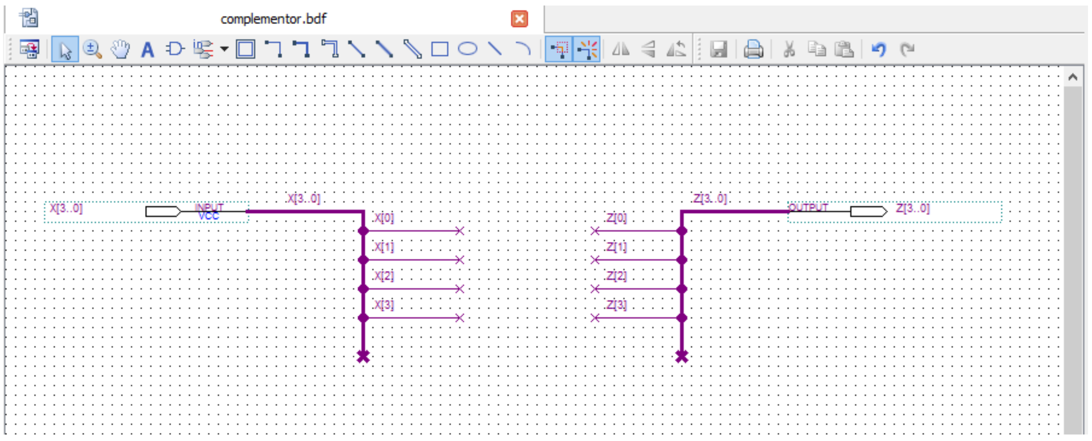


Figure 4.14 A bus implementation which is missing a logic design

30. Save the schematic with the name **complementor** and compile the design.

III.c.3 Functional Simulation of the Complementor

31. Now we're going to verify the behavior of the top-level circuit design. Simulation of this project can be a little bit more complicated since there are a total of 4 inputs applied to the top-level circuit. That means there are 16 possible input combinations! That is probably a little more than we want (or need) to actually test the design with. So, what we should do is develop a simulation strategy or plan that will adequately test the circuit design without taking the time to exhaustively test all possible combinations. Table 4.5 lists some input conditions that might be chosen to test our logic circuit. We will draw the corresponding test vectors in a Vector Waveform File (.vwf).

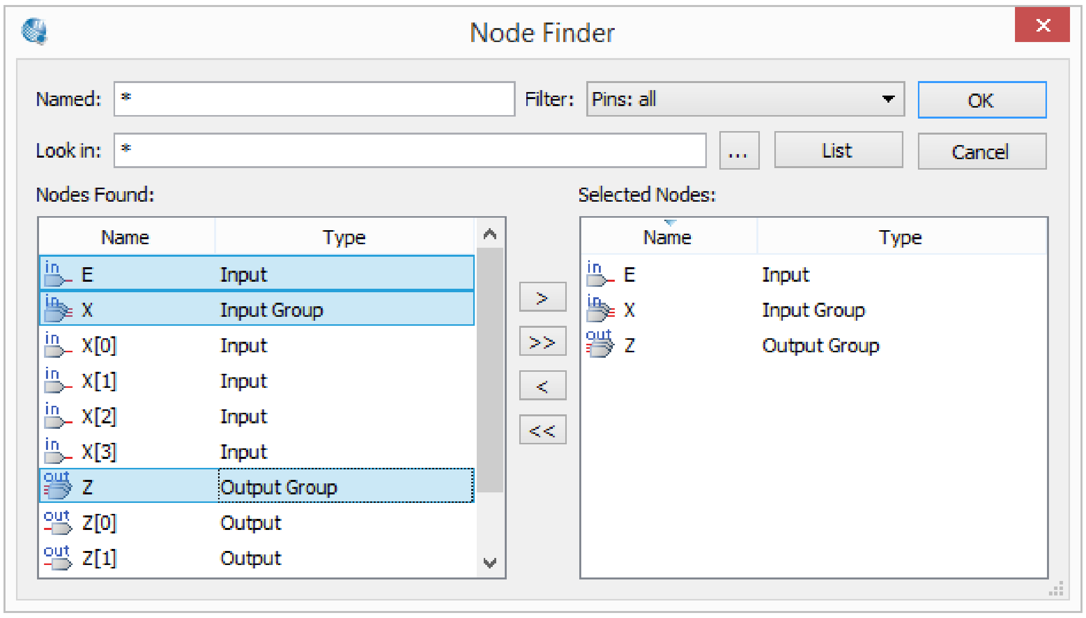
Table 4.5 Selected input test vectors for complementor design

X[3]	X[2]	X[1]	X[0]	X	E
0	0	0	0	**0**	**0**
1	0	1	1	**B**	**1**
0	1	1	0	**6**	**1**
0	0	1	0	**2**	**0**
1	1	1	0	**E**	**1**
1	0	0	1	**9**	**0**
1	1	1	1	**F**	**0**
1	0	0	0	**8**	**0**
0	0	1	1	**3**	**1**
1	1	0	0	**C**	**0**

32. Open the Waveform Editor.

33. Save the file under the name complementor.vwf. Set the desired simulation to run from 0 to 300 ns by selecting **Edit > End Time** and entering 300 ns. By selecting **View > Fit in Window**, display the entire simulation range of 0 to 300 ns in the window. Divide waveform into 10 pieces by selecting **Edit > Grid Size** and entering 30 ns.

34. Next, we want to include the input and output nodes of the circuit to be simulated. Click **Edit > Insert > Insert Node or Bus** to open the dialog box. Click **Node Finder**. Select **Pins:all** and click List. To add your input and output nodes easily, select the Input Group type variable instead of selecting every bit of it separately. Do this for the output bus as it's shown on Figure 4.15. Click **OK** twice.



Nodes Found:

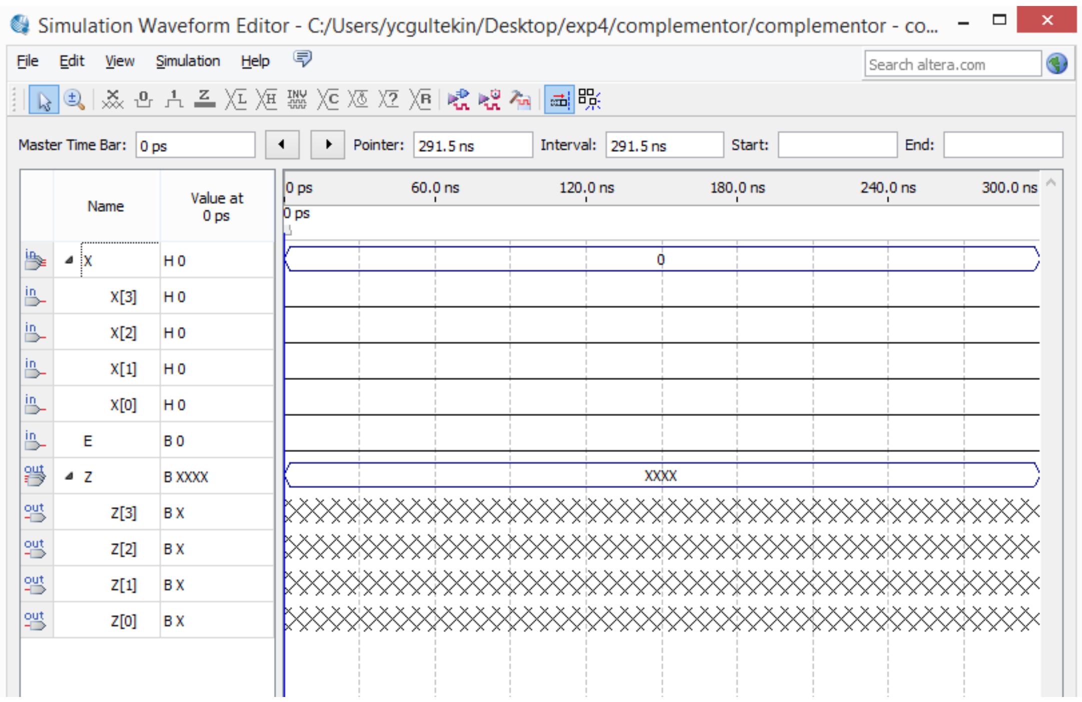
Name	Type
E	Input
X	Input Group
X[0]	Input
X[1]	Input
X[2]	Input
X[3]	Input
Z	Output Group
Z[0]	Output
Z[1]	Output

Selected Nodes:

Name	Type
E	Input
X	Input Group
Z	Output Group

Figure 4.15 Adding busses to the functional simulation setup

35. If you extend your 4-bit input groups by double clicking on their names, the waveform will be seen like in Figure 4.16. Otherwise they will be seen like they have a single value (which is actually the case.)

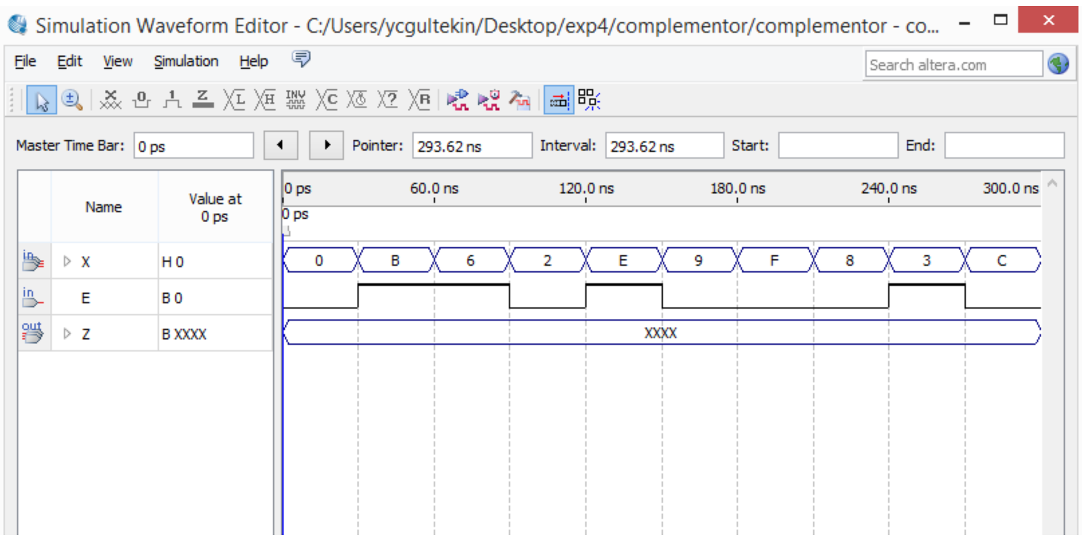

Simulation Waveform Editor - C:/Users/ycgultekin/Desktop/exp4/complementor/complementor - co...
File
Edit
View
Simulation
Help
Search altera.com
0 ps
60.0 ns
120.0 ns
180.0 ns
240.0 ns
300.0 ns
0
XXXX

Master Time Bar: 0 ps Pointer: 291.5 ns Interval: 291.5 ns Start: End:

Name	Value at 0 ps
X	H 0
X[3]	H 0
X[2]	H 0
X[1]	H 0
X[0]	H 0
E	B 0
Z	B XXXX
Z[3]	B X
Z[2]	B X
Z[1]	B X
Z[0]	B X

Figure 4.16 One possible way to visualize the inputs/outputs

36. Arrange the waveform by selecting every 30 ns period, right clicking, selecting **Value>Arbitrary Value>Radix: Hexadecimal** and entering the values according to the Table 4.5. Your waveform should be seen like the Figure 4.17.



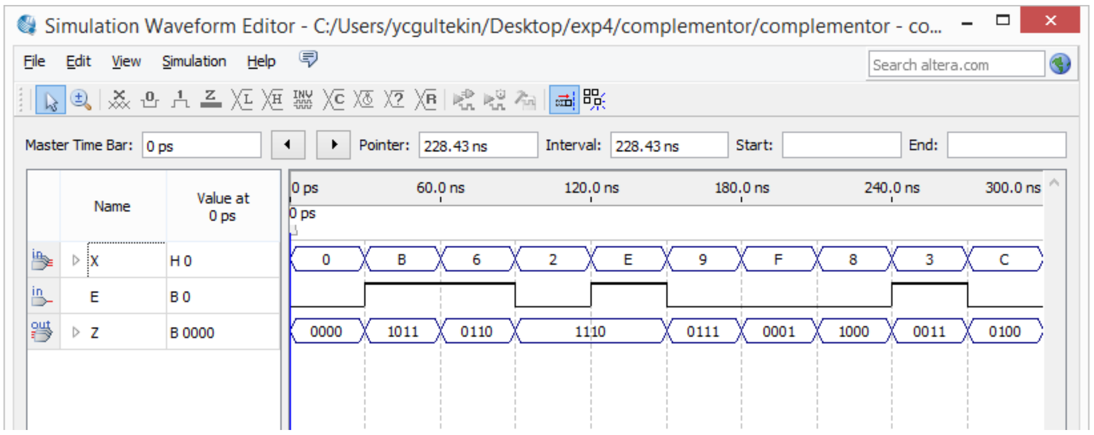
Master Time Bar: 0 ps Pointer: 293.62 ns Interval: 293.62 ns Start: End:

Name	Value at 0 ps
X	H 0
E	B 0
Z	B XXXX

Figure 4.17 Functional simulation window prior to simulation

37. Save the waveform. Run Functional Simulation.

38. You will see the simulation results of your design according to the inputs that you give. Check the output for different input combinations. Being sure the design is working properly, close Waveform Editor Window. Your functional simulation results will hopefully be seen like in Figure 4.18.

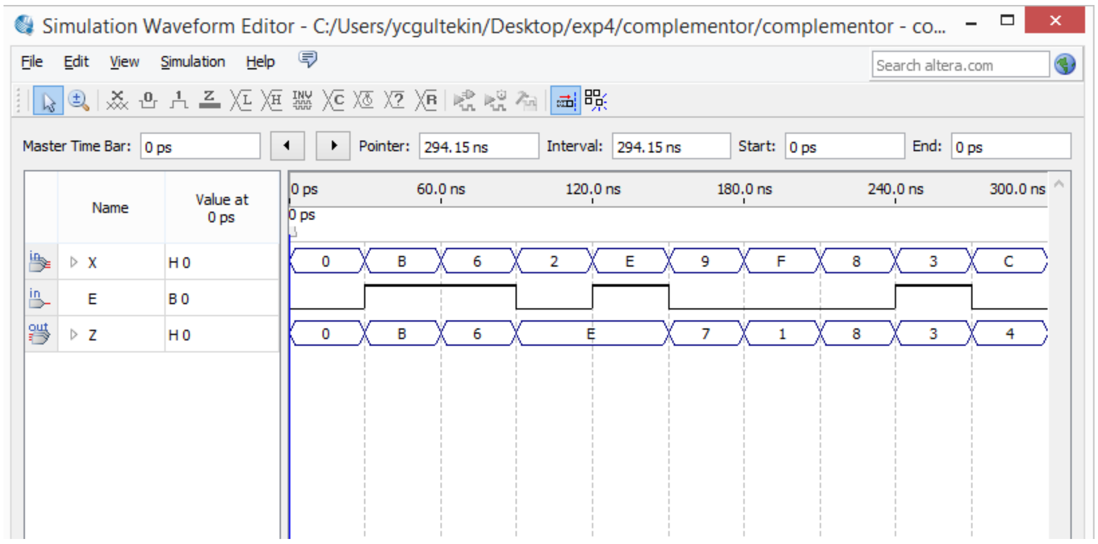

Simulation Waveform Editor - C:/Users/ycgultekin/Desktop/exp4/complementor/complementor - co...
File
Edit
View
Simulation
Help
Search altera.com
0 ps
60.0 ns
120.0 ns
180.0 ns
240.0 ns
300.0 ns

Master Time Bar: 0 ps Pointer: 228.43 ns Interval: 228.43 ns Start: End:

Name	Value at 0 ps										
X	H 0	0	B	6	2	E	9	F	8	3	C
E	B 0										
Z	B 0000	0000	1011	0110	1110		0111	0001	1000	0011	0100

Figure 4.18 One way to see simulation results

To change the representations of the input and output, you can right click on the values under the name tab situated on he left and select **Radix>Binary, Radix>Hexadecimal** or anyone of them as you like. Another representation can be seen in Figure 4.19.


Simulation Waveform Editor - C:/Users/ycgultekin/Desktop/exp4/complementor/complementor - co...
File
Edit
View
Simulation
Help
Search altera.com
0 ps
60.0 ns
120.0 ns
180.0 ns
240.0 ns
300.0 ns

Master Time Bar: 0 ps Pointer: 294.15 ns Interval: 294.15 ns Start: 0 ps End: 0 ps

Name	Value at 0 ps										
X	H 0	0	B	6	2	E	9	F	8	3	C
E	B 0										
Z	H 0	0	B	6	E		7	1	8	3	4

Figure 4.19 Another way to see simulation results

III.c.4 Testing the Complementor Design on FPGA

39. Assign input and output pins to your complementor design by using your experience. For example, you can use SW9, SW8, SW7, and SW6 for data input vector X, SW0 for control input E, and LED0, LED1, LED2, and LED3 for output vector. **Save** your assignment.

40. Recompile your design, and use output files to **program** your FPGA. **Test** the input combinations given in Table 4.5 and more on FPGA.

III.d. Designing the Adder/Subtractor

41. Based on your design in Preliminary Work Part 6, **implement** your adder/subtractor design on a new project. **Don't forget** to create a new project with a unique name in a separate folder, and add files of your low-level projects (full-adder, and 4-bit binary adder) while creating your project.

Note: While implementing your adder/subtractor design, you should use busses to represent input and output vectors. After all, your design should include 3 input pins (2 of them are for input busses, and the other is for control input E), and 1 output pin for the output bus. For the sake of convenience, use X, Y, and E as input names, and Z for output name.

Table 4.6 Selected input test vectors for adder/subtractor design

X[3]	X[2]	X[1]	X[0]	X	Y[3]	Y[2]	Y[1]	Y[0]	Y	E
0	1	1	0	**6**	0	0	1	1	**3**	**1**
0	1	1	1	**7**	0	0	0	1	**1**	**1**
1	1	0	1	**D**	0	1	0	1	**5**	**0**
0	0	1	1	**3**	0	1	0	0	**4**	**0**
1	1	1	0	**E**	1	1	1	1	**F**	**1**
1	0	0	1	**9**	1	0	0	1	**9**	**1**
1	1	1	1	**F**	1	1	0	0	**C**	**0**
1	0	0	0	**8**	0	0	0	0	**0**	**0**

42. Simulate your design functionally by using input combinations given in Table 4.6. **Don't forget** to **save** and **compile** your design beforehand.

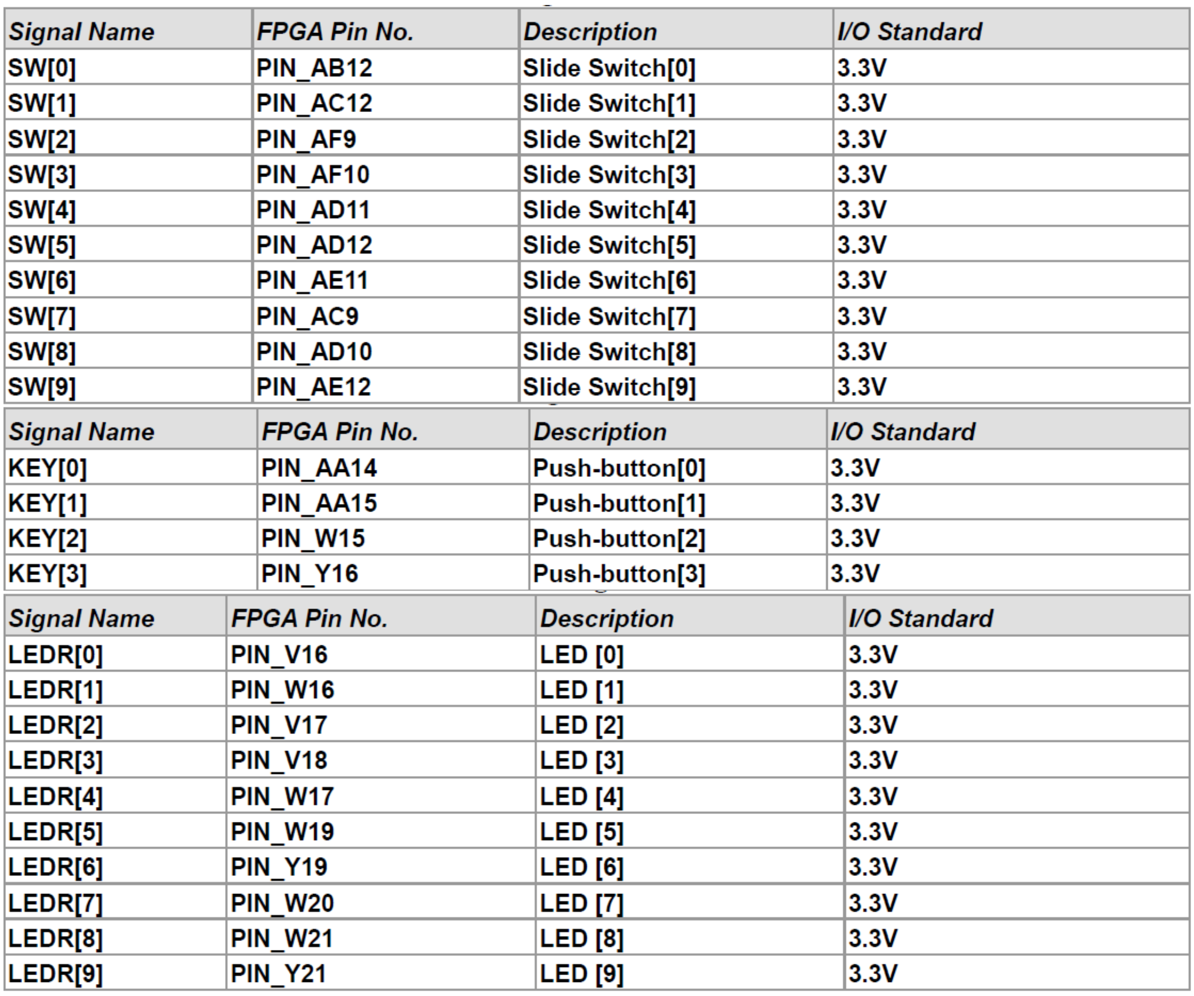
43. Assign pins to your inputs and outputs, and test your design on FPGA. **Don't forget** to **save** and **recompile** your design after assigning pins.

IV. References

[1] Manual for Experiment 4: Parallel Adders, Subtractors, and Complementors. EE-314 Digital Electronics Laboratory, METU. (Used until 2015)

[2] G.L. Moss, "Quartus Tutorial 3–Hierarchical Designs, A step-by-step tutorial using Quartus II v9.x." May 2010.

V. FPGA Pin Assignment Codes



Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
SW[0]	PIN_AB12	Slide Switch[0]	3.3V
SW[1]	PIN_AC12	Slide Switch[1]	3.3V
SW[2]	PIN_AF9	Slide Switch[2]	3.3V
SW[3]	PIN_AF10	Slide Switch[3]	3.3V
SW[4]	PIN_AD11	Slide Switch[4]	3.3V
SW[5]	PIN_AD12	Slide Switch[5]	3.3V
SW[6]	PIN_AE11	Slide Switch[6]	3.3V
SW[7]	PIN_AC9	Slide Switch[7]	3.3V
SW[8]	PIN_AD10	Slide Switch[8]	3.3V
SW[9]	PIN_AE12	Slide Switch[9]	3.3V

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
KEY[0]	PIN_AA14	Push-button[0]	3.3V
KEY[1]	PIN_AA15	Push-button[1]	3.3V
KEY[2]	PIN_W15	Push-button[2]	3.3V
KEY[3]	PIN_Y16	Push-button[3]	3.3V

Signal Name	*FPGA Pin No.*	*Description*	*I/O Standard*
LEDR[0]	PIN_V16	LED [0]	3.3V
LEDR[1]	PIN_W16	LED [1]	3.3V
LEDR[2]	PIN_V17	LED [2]	3.3V
LEDR[3]	PIN_V18	LED [3]	3.3V
LEDR[4]	PIN_W17	LED [4]	3.3V
LEDR[5]	PIN_W19	LED [5]	3.3V
LEDR[6]	PIN_Y19	LED [6]	3.3V
LEDR[7]	PIN_W20	LED [7]	3.3V
LEDR[8]	PIN_W21	LED [8]	3.3V
LEDR[9]	PIN_Y21	LED [9]	3.3V

Figure 4.20 DE1 S0C pin assignment descriptions

VI. Required IC List

7400 IC Four NAND2, i.e., four two input NAND gates
7483 IC 4-bit binary adder
7486 IC Four XOR2, i.e., four two-input XOR gates

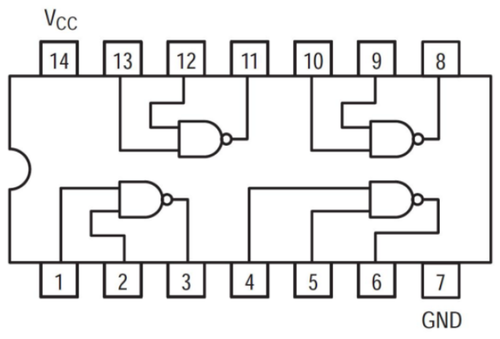


Figure 4.21 Pin Diagram for 7400 IC

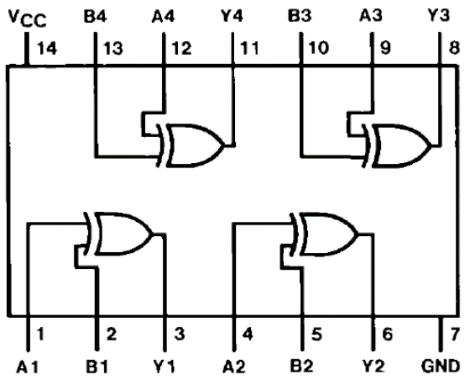


Figure 4.22 Pin Diagram for 7486 IC

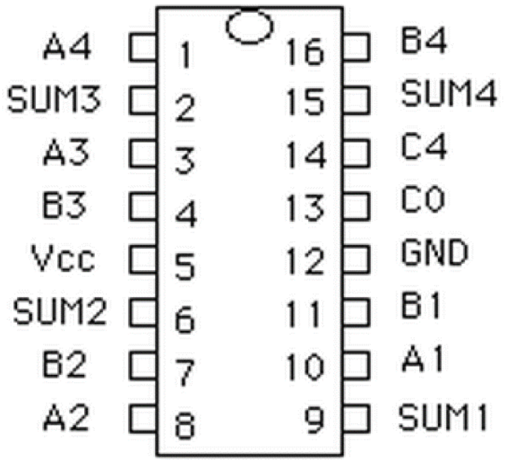


Figure 4.23 Pin Diagram for 7483 IC