Lab 3

ARITHMETIC CIRCUITS

Presented by: Alaa Eddin Alchalabi

Due July 5th. 2020

Objectives

- Create and simulate a full adder.
- Use a full adder as a component in an 8-bit adder/subtractor.
- Create a hierarchical design, including components for full adders, subtractors and parallel multipliers using the QUARTUS II graphic editor.
- Design an overflow detector for use in a two's complement adder/subtractor.

Background

Arithmetic Circuits

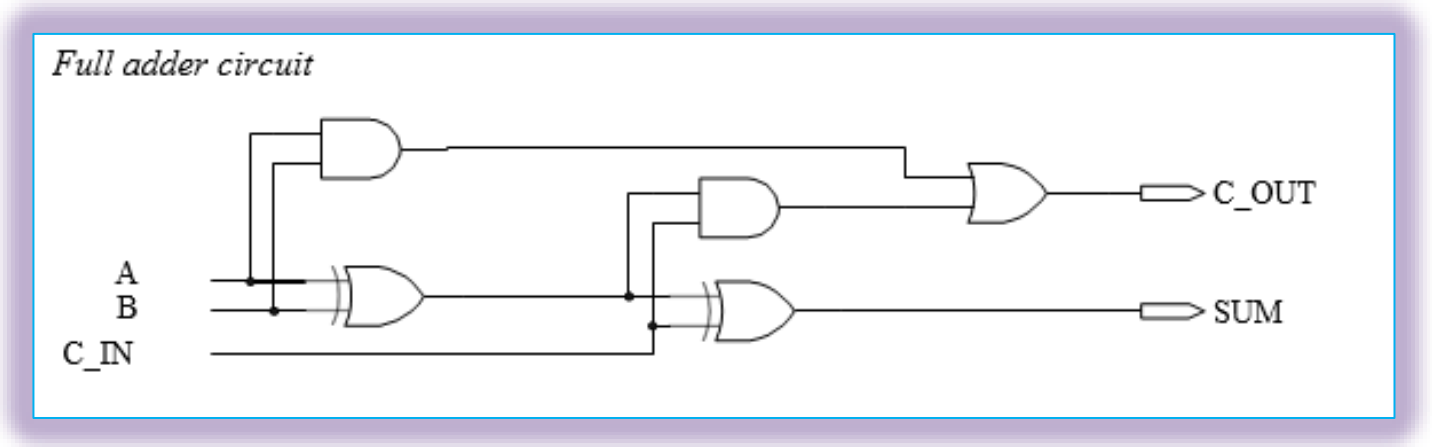
- Circuits for performing binary arithmetic are based on half adders, which add two bits and produce a sum and carry, and full adders, which also account for a carry added from the lesser significant bit.
- Full adders can be grouped together to make a parallel binary adder, with *n* full adders allowing two *n*-bit numbers to be added, generating an *n*-bit sum and a carry output.
- A parallel adder can be converted to a two's complement adder/subtractor by including XOR functions on the inputs of one set of operand bits, say input B, allowing the operations A + B or A - B to be performed.
- Parallel Multipler can be constructed by cascading multiple adders to perform the required product operations.

A control input, SUB (for SUBtract):

SUB	C_IN0	Math Equation	Description
0	0	A+B	Addition
0	1	A+B+1	Addition plus 1
1	0	$(A+\overline{B}=A-B-1)$	One's Complement Subtraction
1	1	$(A+\overline{B}+1=A-B)$	Subtraction

- If HIGH it causes the XORs to invert the B bits, producing the ones complement of B.
- If HIGH and the carry is HIGH then the result is (A + (B' + 1) = $A - B$)
- If LOW, it transfers B to the parallel adder without modification.
- If LOW, and the carry input is HIGH then then the result is B+1
- Sign-bit overflow occurs when a two's complement sum of difference exceeds the permissible range of numbers for a given bit size.
- This can be detected by an SOP circuit that compares the operands and result sign bits of a parallel adder if the 2's complement is used to represent the.

Part I: One-Bit Full adder



- Create a new project in Quartus
- Add a block diagram sheet to the project and draw the logic diagram of the1 bit Full adder circuit Compile the project
- Create and run a simulation
- Take the experimental truth table of the full adder to verify its operation.

Part 2: Parallel Adder

Table 1				
Grid	Binary Inputs	Carry	Binary Sum	**Hex Equivalent**
1	01111111 + 00000001			
2	11111111 + 00000001			
3	11000000 + 01000000			
4	**11000000 + 10000000**			

In the above table, giving the expected sum in both binary and hexadecimal. Show the carry output separately. Below are examples of addition with separate carry output.

e.g.: 10111111 + 10000001 = 01000000 (Carry output = 1)

Hexadecimal equivalent: $(BF)_{16} + (81)_{16} = (40)_{16}$ (Carry output = 1)

e.g.: 00111111 + 0000001 = 01000000 (Carry output = 0)

Hexadecimal equivalent: $(3F)_{16} + (01)_{16} = (40)_{16}$ (Carry output = 0)

Part 2: Parallel Adder cont.

1) Use the circuit of part I to create a full adder block:
   - Go to the file menu and select create/update and Create Symbol file for current file. The circuit can now be used as a symbol for future use.
   - To use new symbol:→ Go to insert symbol, a new library called ‘Project’ should be available Under ‘Project’ you will find your symbol

2) Add a block diagram sheet to the project and draw the logic diagram of an 8-bit parallel adder using the full adder block. *(Hint) you can refer to the recording of lecture 4.2*

3) Compile the project

4) Simulate your circuit with the inputs of Table 1 and compare your results.

5) Include the final diagram, table, simulation figure in your report.

Part 3: Two's Complement Adder/Subtractor

Table 2						
Grid	Binary Inputs	Overflow	Carry	Binary Sum	Hex Equivalent	**Two's Compl.**
1	01111111 + 00000001					
2	11111111 + 00000001					
3	00000000 - 00000001					
4	00000000 - 01111111					
6	11000000 + 01000000					
6	**11000000 + 10000000**					

Follow the examples shown to add the signed binary numbers in Table 2, giving the sum in both binary and hexadecimal. Show the carry and overflow outputs separately from the sum. o Carry output maintains the same functionality as before, and does not need to be modified for signed arithmetic.

e.g.: 01111111 + 01000001 = 11000000 (Overflow = 1, Carry output = 0);

Hexadecimal equivalent: (7F)16 + (41)16 = (C0)16

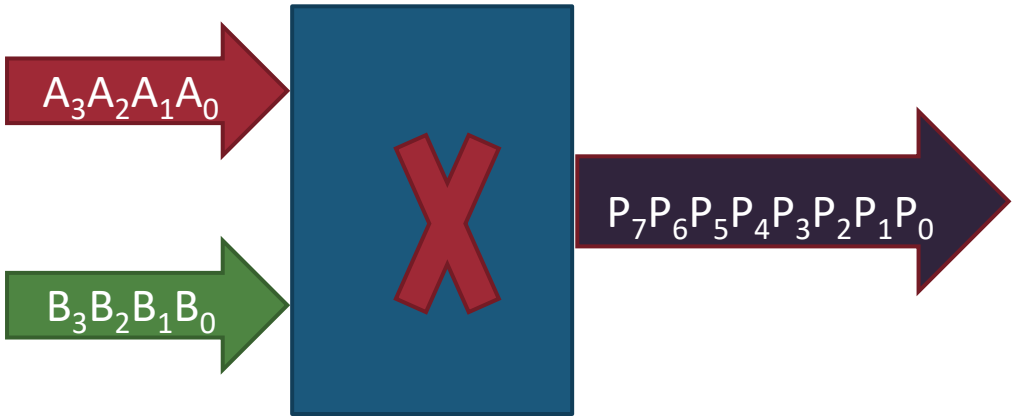
e.g.: 00000000 - 00000011 = 11111101 (Overflow = 0, Carry output = 0); Hexadecimal equivalent: (00)16 – (03)16 = (FD)16

2. (two's complement: (FD)16 = 11111101 = (-3)10)

Part Three: Cont.

1. Modify the 8-bit adder you created to make an 8-bit two's complement adder/subtractor.
2. Include an overflow detector for when the output of the adder/ subtractor overflows beyond the permissible range of the values for an 8-bit (2's complement) signed number.
3. Compile the project
4. Simulate your circuit with the inputs of Table 2→ Test the operation of the circuit by applying the A and B inputs from Table 2.
5. Include the final diagram, table, simulation figure in your report.

Part four: 4bit Multiplier



Grid	AxB	Binary output	Hex Equivalent
1	1010*1000		
2	0000*1111		
3	0110*1110		
4	1111*1111		

Multiplication is the most commonly used in every step of the world. It is nothing but the addition by the multiplicand adds multiplier no. of times (4 times here). With the guidance of Table 3 and lecture 4.3→ Design and simulate a 4 bits binary multiplier using the skills you have mastered in this lab.