School of Electrical Engineering and Computer Science | École de science informatique et de génie électrique

University of Ottawa | Université d’Ottawa

CEG2136 | Fall 2020



**CEG2136**

**Computer Architecture**

Course Professor:Dr. Fadi Malek

**Lab 3 Report**

**Arithmetic Logic Unit**

Presented to: (T.A) Yazan Otoum

Prepared by Group # 3 Section B01

Team Members:

|  |  |
| --- | --- |
| Family Name, Name: | Student ID: |
| Das, Pion | 300124924 |
| Lin, Tom | 300076426 |
| Tang, Matthew | 300138539 |

**Date:** November 11, 2020

**Table of Contents**

[**Objectives**](#_6lqk3crbinmx) **7**

[**Equipment and components**](#_toqithlu7sv0) **7**

[**Algorithmic Solution**](#_ah4psc8myg4u) **7**

[**Design Part**](#_egh9lpkze8re) **9**

[**Discussion**](#_u2y82xxw8kx7) **17**

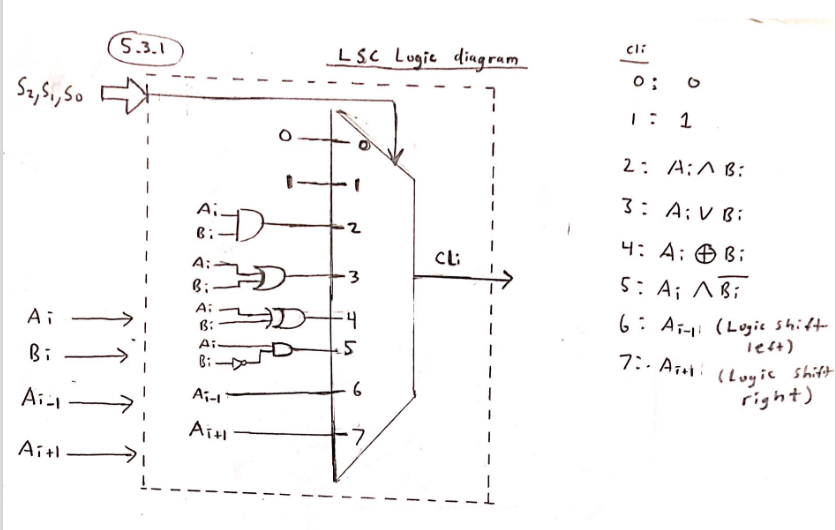
[**Conclusion**](#_gtcefd64lgyz) **17**

[**Group Member Contributions**](#_5ksblhtp0p0p) **17**

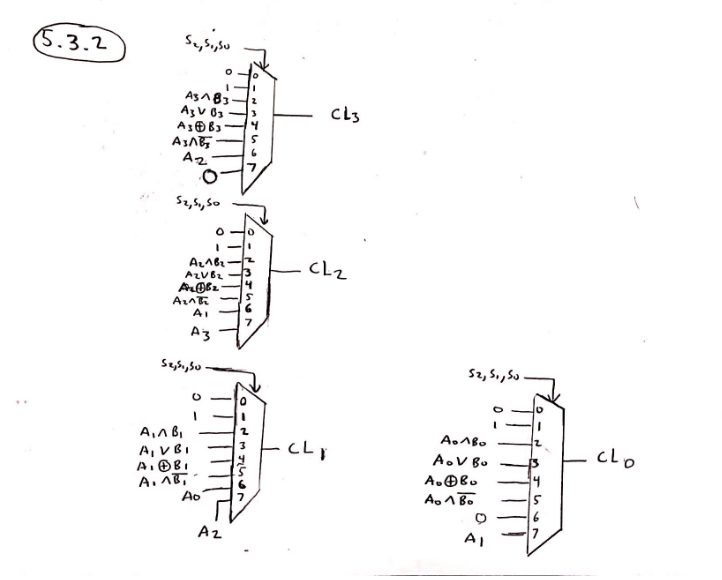
[**References**](#_7d3jfmd9hjbx) **18**

## **Prelab**

**5.3.1**

****

**5.3.2**

****

**5.4.1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S2** | **S1** | **S0** | **op1** | **op2** | **Cy\_in** | **CA output** |
| 0 | 0 | 0 | A | B | 0 | CA ← A + B |
| 0 | 0 | 1 | A | B | 1 | CA ← A + B + 1 |
| 0 | 1 | 0 | A | 0 | 0 | CA ← A |
| 0 | 1 | 1 | A | 0 | 1 | CA ← A + 1 |
| 1 | 0 | 0 | A | B’ | 0 | CA ← A + B’ |
| 1 | 0 | 1 | A | B’ | 1 | CA ← A + B’ + 1 |
| 1 | 1 | 0 | A’ | 0 | 0 | CA ← A’ |
| 1 | 1 | 1 | A’ | 0 | 1 | CA ← A’ + 1 |

Output = (A, A’) + (B, 00, B’, 00) + S0

Output = (A or A’) + (B or 00 or B’ or 00) + S0

**5.4.2**

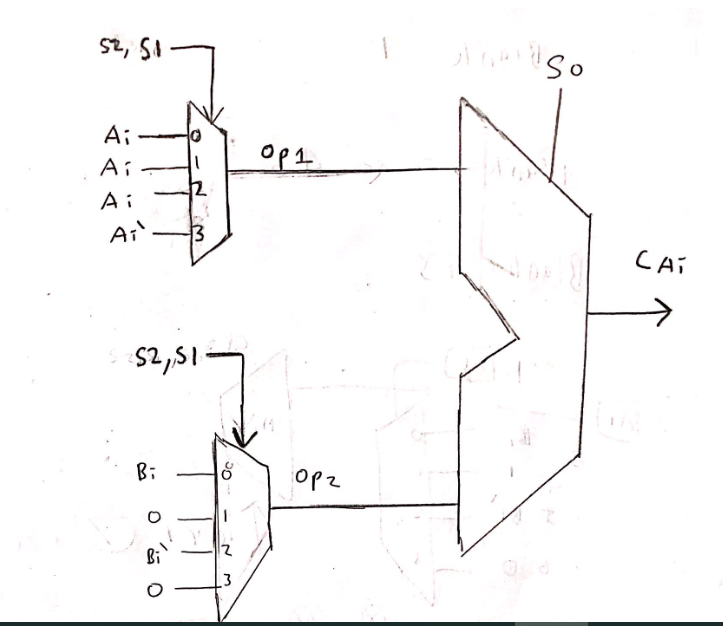
|  |  |  |
| --- | --- | --- |
| **S2** | **S1** | **Ai** |
| 0 | 0 | A0 (A) |
| 0 | 1 | A1 (A) |
| 1 | 0 | A2 (A) |
| 1 | 1 | A3 (A’) |

|  |  |  |
| --- | --- | --- |
| **S2** | **S1** | **Bi** |
| 0 | 0 | B0 (B) |
| 0 | 1 | B1 (0) |
| 1 | 0 | B2 (B’) |
| 1 | 1 | B3 (0) |

**5.4.3**

Cy\_in = S0

**5.4.4**

****

**5.5**

(Cy = 1) only if (Arithmetic and carry out is 1)

Cy = S3’ **∧**  CarryOut

S = 1 only if C3 = 1

S = C3

Z = 1 only if the result is 0000

Z = (C3 + C2 + C1 + C0)’

V = 1 only if there is overflow

Overflow = 1 only if (Carry in of MSB **⊕** Carry out of MSB) = 1

V = Cin MSB **⊕** Cout MSB

#### 

#### **Objectives**

The objective of this lab is to design, build and test an Arithmetic Logic Unit (ALU). The ALU will be able to perform any of the 16 micro operations that are seen in table 1. The control signals S3, S2, S1, and S0 will be used to switch between the micro operations. The inputs will be either A, B, 1 or 0. To determine the status, we have an ALU status register which has 4 status bits Cy, S, Z and V.

#### **Equipment and components**

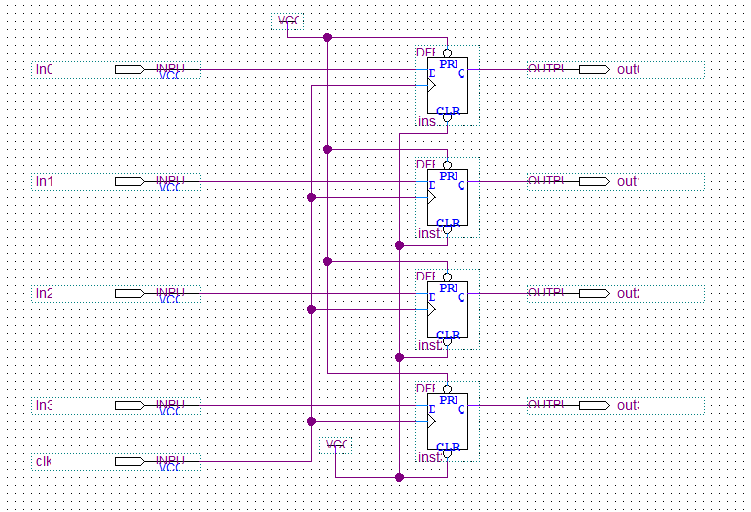
* Quartus II 13.0 Service-Pack 1
* Altera DE2-115 board with USB-Blaster cable and Power supply 12 VDC, 2A

#### **Algorithmic Solution**

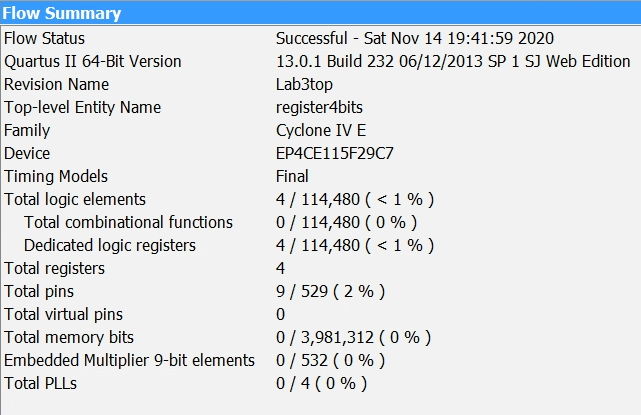
S3 will control switching between the Arithmetic circuit (AC) when S3 = 0 and Logic and shift circuit (LSC) when S3 = 1. S2, S1, and S0 will determine the output operation that is done. Firstly we created the LSC from the prelab by implementing the logic to pass into the 8 by 1 mux e.g. A **∧** B, 1, 0 etc. . The multiplexer will have 3 selection lines which are S2, S1, and S0. We then created the Arithmetic circuit by first finding the pattern in the truth table. We determined that the output would be equal to (A or A’) + ( B or 0 or B’) + (1 or 0). From there, we created a 4 by 1 multiplexer with selection lines S2 and S1. S1 and S2 are used to determine if A, A’, B, B’, or 0 is added. S0 determines whether 1 or 0 is added. Finally, a status register is needed to determine the status of the ALU. Cy is the carry and is 1 when the ALU is computing an arithmetic operation. This can be created into logic Cy = S3’ **∧**  CarryOut. The sign is 1 if C3 is 1. Z is 1 if the ALU contains 0000. This can be represented with a NOR gate of all the inputs. The final part is overflow which can be checked by doing an XOR of the carry in of the MSB and the carry out of the MSB.

In Prelab 5.4.1, there are 4 major bdf files. This is the logiccircuit4bit.bdf for the LSC, the arithcircuit4bits.bdf for the Arithmetic circuit, state4bits for the status of the ALU and finally Lab3top.bdf to put everything together. To implement the 4bit LSC, we needed to implement the 1bit LSC. Then we could duplicate the 1bit LSC 4 times as the logic was the same for each bit. The 4bit LSC would produce C3, C2, C1 and C0. The logiccircuit1bit.bdf is a representation of pre lab 5.3.1. The artihcircuit4bits.bdf is done by using prelab 5.4.1 for the inputs and figure 16 in the lab manual for the overall design. The state4bits.bdf represents 5.5 of the prelab and was created with the equations we implemented. Finally, adding all the components together we created the ALU.

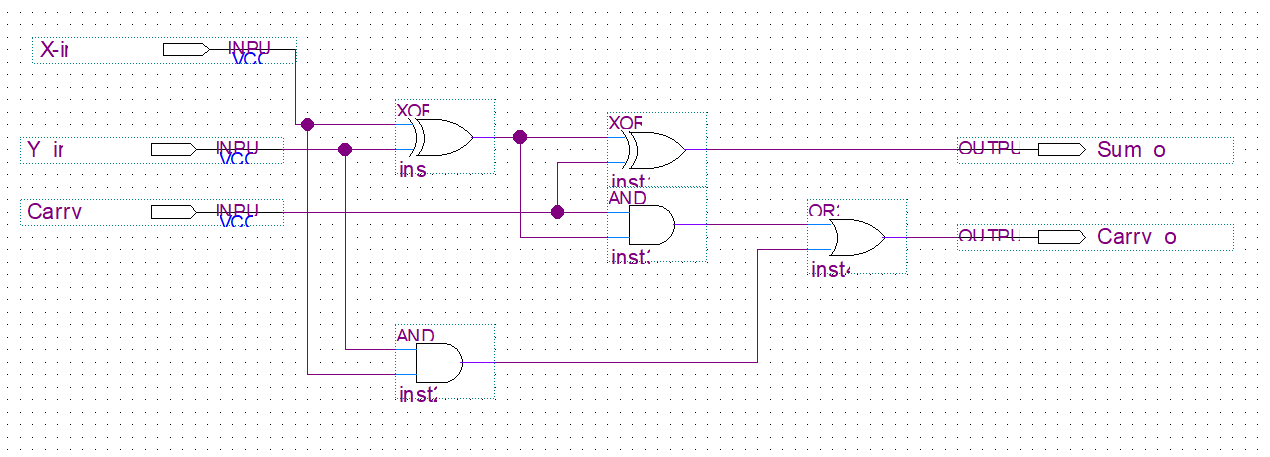
#### **Design Part**



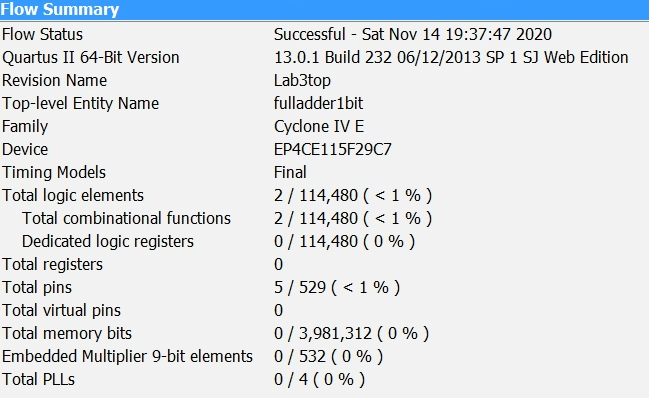
**Figure 1: 4 bit register (Given in the lab report) This is for the shifting**



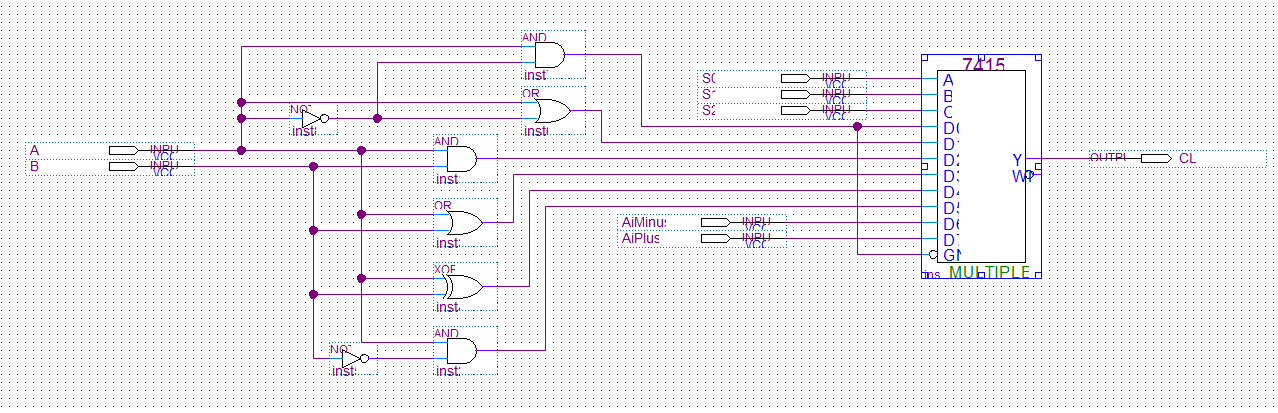
**Figure 2: register4bits.bdf flow summary**



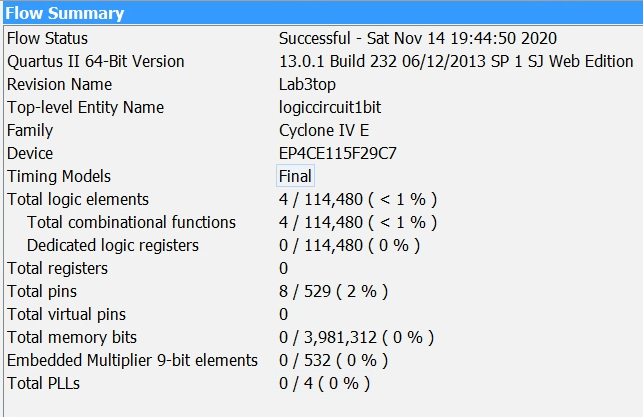
**Figure 3: Full adder 1 bit (fulladder1bit.bdf given in lab manual)**



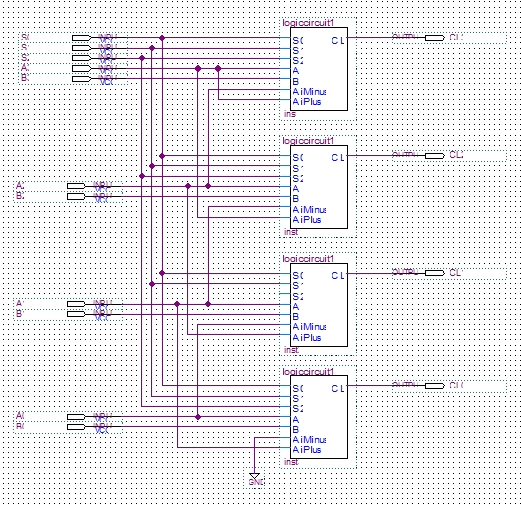
**Figure 4: Fulladder1bit.bdf Flow summary**



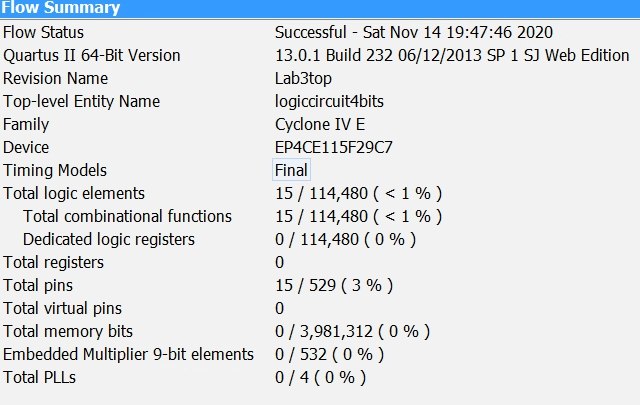
**Figure 5: Logic circuit 1 bit (This is for 5.3.1 of the prelab)**



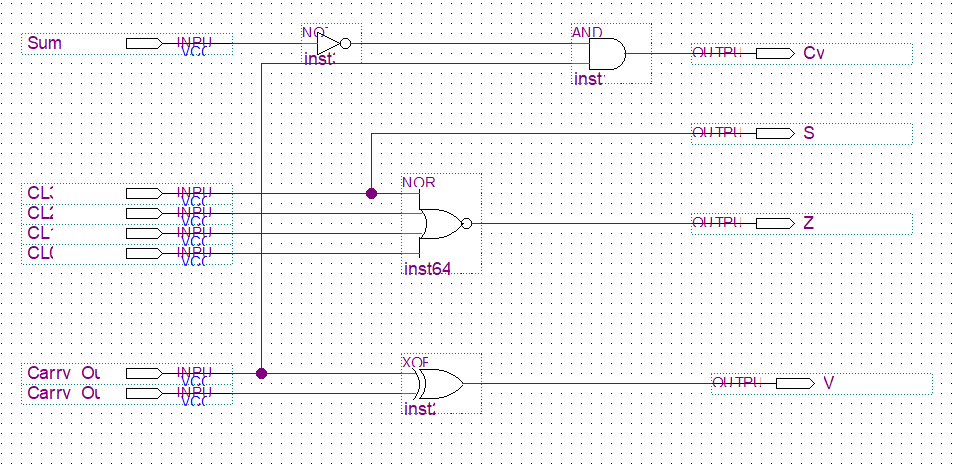
**Figure 6: logiccircuit1bit.bdf Flow Summary**



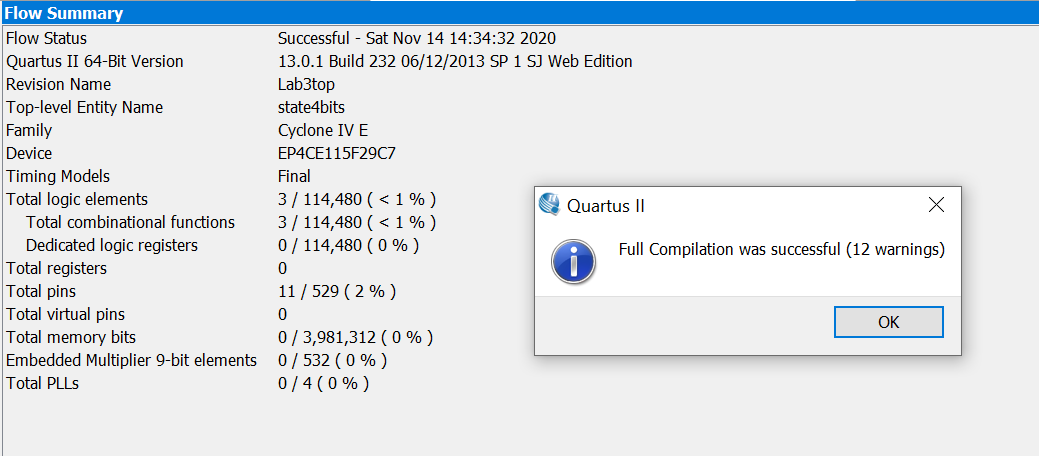
**Figure 7: logiccircuit4bits.bdf**



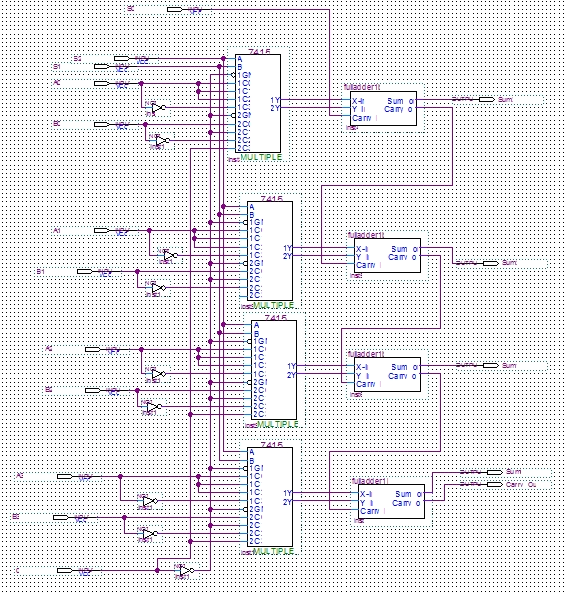
**Figure 8: logiccircuit4bit.bdf Flow summary**



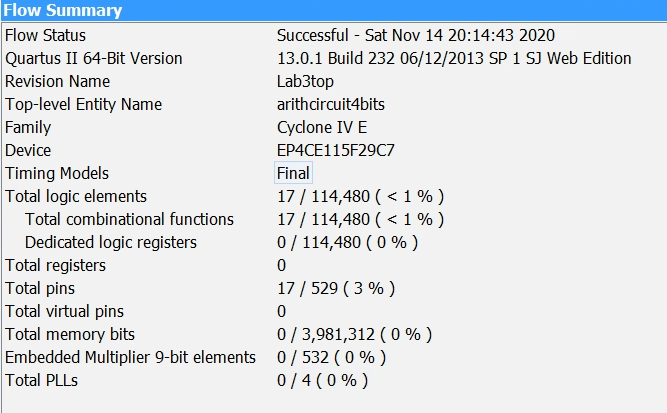
**Figure 9: Prelab 5.5 State indicator diagram (state4bits.bdf)**



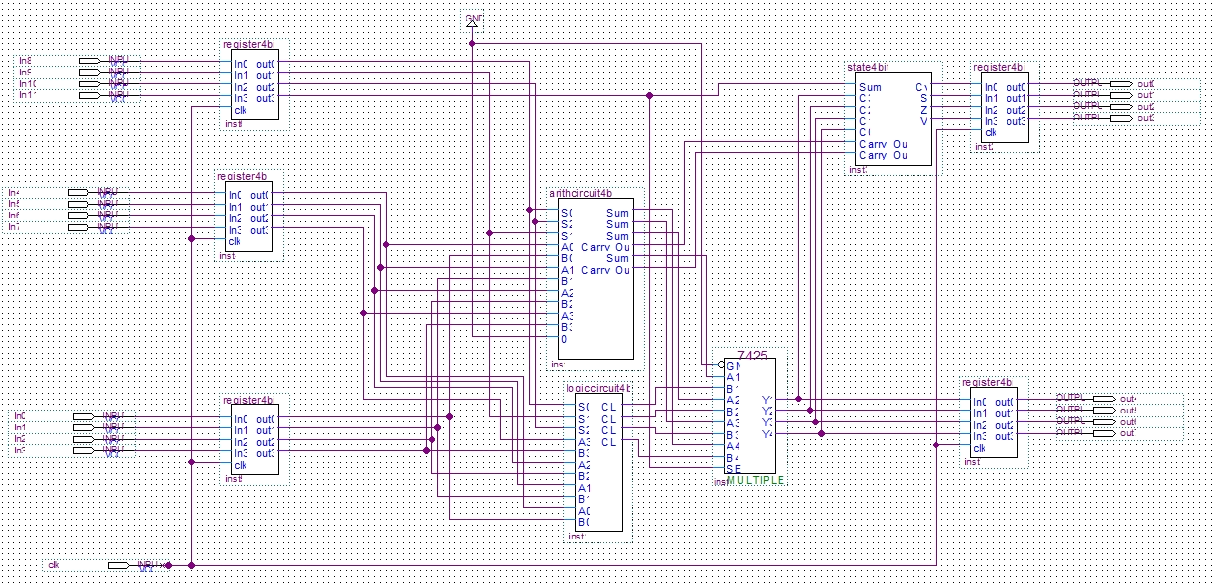
**Figure 10: state4bits.bdf flow summary**

****

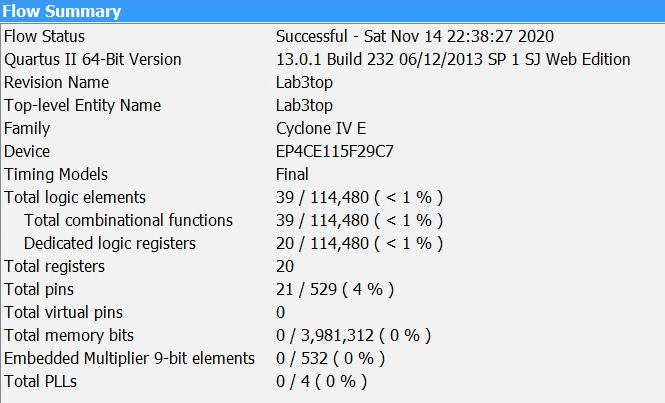
**Figure 11: arithcircuit4bits.bdf**



**Figure 12: arithcircuit4bits.bdf Flow Summary**



**Figure 13: lab3top.bdf**



**Figure 14: lab3top.bdf Flow Summary**

**Group 3**

**3%5 = 3 : A = 1100, B = 0111, A = -410 B = 710**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Clock**  **Cycle** | **A3** | **A2** | **A1** | **A0** | **B3** | **B2** | **B1** | **B0** | **S3** | **S2** | **S1** | **S0** | **C** | **V,Z, S,Cy** | **RTL Micro Operations** |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0011 | 0001 | C ← A + B |
| 2 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0100 | 0001 | C ← A + B + 1 |
| 3 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1100 | 0010 | C ← A |
| 4 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1101 | 0010 | C ← A + 1 |
| 5 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0100 | 1001 | C ← A + B’ |
| 6 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0101 | 1001 | C ← A + B’ + 1 |
| 7 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0011 | 0000 | C ←A’ |
| 8 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0100 | 0000 | C ←A’ + 1 |
| 9 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0000 | 0100 | C ← “0000” |
| 10 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1111 | 0010 | C ←”1111” |
| 11 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0100 | 0000 | C ← A **∧** B |
| 12 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1111 | 0010 | C ← A ∨ B |
| 13 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1011 | 0010 | C ← A ⊕ B |
| 14 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1000 | 0010 | C ← A **∧** B’ |
| 15 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1000 | 0010 | C ← ashl A |
| 16 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0100 | 0000 | C ←ashr A |

#### **Discussion**

This lab consisted of creating an ALU. The ALU was created with 4 primary components which is the register4bit.bdf, logiccircuit4bits.bdf, arithcircuit4bits.bdf, and state4bits.bdf. The logic circuit does logic and shift operations while the arithmetic circuit would do arithmetic calculations. Both the logic circuit and arithmetic circuit consisted of 4 registers. Each register has their own designated CU outputs which are inputs for the data path. This consists of the ALU Control, DIP switches, and data input. Similarly, two registers also provide a total of 8 outputs which are the ALU status and the Data outputs. These outputs ultimately behave as outputs through the control unit.

#### **Conclusion**

Therefore we created an ALU in this lab. An Arithmetic Logic Unit is used for arithmetic and logic operations as well as shifting in the CPU. They use control inputs to determine the operation that needs to be performed.

#### **Group Member Contributions**

**Das Pion** - circuit building, board testing, report writing

**Lin Tom** - prelab, circuit building, board testing, report writing

**Tang Matthew** - circuit building, testing

#### **References**

“DE2-115 User Manual,” Terasic Technologies Inc.,http://www.terasic.com.tw/cgibin/page/archive.pl?Language=English&CategoryNo=165&No=502&PartNo=4 , 2013

