Name: Matthew Zaldana

CECS 225 Logic Design and Computer Architecture

California State University Long Beach

Assignment #6: LogiSim Design of a N-bit ALU Due: 12 NOV 2020

***(from Hennesey and Patterson, Computer Organization and Design)***

Use LogiSim to design the 32-bit ALU discussed before. The overflow flag OF is the XOR or the CarryIn and CarryOut of the msb. The carry flag CF is the CarryOut of the msb. The zero flag ZF is a Zero detector. The sign flag SF is the msb of result. Submit a printed copy of the LogiSim ALU design, complete this table by entering the values of A, B, and ALUOp shown in the table and record your results as shown on the LogiSim simulation and also indicate the operation and common name (if there is one) that the ALUOp code performs, e.g. operation A&B is AND for ALUOp =0. Get your results from your ALU unit. Some ALUOp inputs do not perform very useful operations, and leave the common name of those blank. The various inputs below investigate the 16 possible operations of the ALU. Notice which pattern of (ZF,CF,OF) does not appear.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | A16 | B16 | ALUOp | Operation  Performed | Common  Name | 32-bit Result (8 digit HEX) | ZF | CF | OF | SF |
| 1 | 0x00000009 | 0x00000005 | 0 | A&B | AND | 0x00000001 | 0 | 0 | 0 | 0 |
| 2 | 0x00000005 | 0x0000000d | 1 | A|B | OR | 0X00000008 | 0 | 1 | 0 | 1 |
| 3 | 0x80000000 | 0x80000000 | 2 | A+B | ADD | 0X00000000 | 1 | 1 | 0 | 0 |
| 4 | 0x0000005c | 0x0000003e | 3 | A<B | LESS | 0XFFFFFE3 | 0 | 1 | 0 | 1 |
| 5 | 0x0000008c | 0x00000042 | 4 | A&!B |  | 0X000000CE | 0 | 0 | 0 | 0 |
| 6 | 0x00000027 | 0x00000055 | 5 | A|!B |  | 0XFFFFFFDD | 0 | 1 | 0 | 0 |
| 7 | 0x0000003f | 0000000x61 | 6 | A+(!B+1) | SUBTRACT | 0XFFFFFFBF | 0 | 0 | 0 | 1 |
| 8 | 0x00000074 | 0x0000009a | 7 | A<!B |  | 0XFFFFFFEF | 0 | 1 | 0 | 1 |
| 9 | 0x00000012 | 0x00000088 | 8 | !A&B |  | 0X0000009A | 0 | 0 | 0 | 0 |
| 10 | 0x000000cd | 0x000000bf | 9 | !A|B |  | 0XFFFFFFF1 | 0 | 0 | 0 | 1 |
| 11 | 0x000000d3 | 0x00000000 | 10 | !A+B |  | 0X000000D3 | 0 | 1 | 0 | 0 |
| 12 | 0x00000078 | 0x00000032 | 11 | !A<B |  | 0XFFFFFF55 | 0 | 1 | 0 | 1 |
| 13 | 0xf7e0005f | 0xabc00031 | 12 | !A&!B |  | 0X00000001 | 0 | 1 | 0 | 1 |
| 14 | 0x00000034 | 0x000000f9 | 13 | !A|!B |  | 0X00000000 | 0 | 1 | 0 | 0 |
| 15 | 0xffffffff | 0x00000001 | 14 | !A+!B |  | 0X00000001 | 0 | 1 | 0 | 1 |
| 16 | 0x00000001 | 0xffffffff | 15 | !A<!B |  | 0X00000001 | 0 | 0 | 0 | 1 |
| 17 | 0x7fffffff | 0x00000003 | 2 | A+B |  | 0X7FFFFFFC | 0 | 1 | 0 | 0 |
| 18 | 0xffffffff | 0x00000002 | 6 | A+(!B+1) |  | 0X00000000 | 0 | 0 | 0 | 1 |
| 19 | 0x80000000 | 0xf0000000 | 2 | A+B |  | 0XE00000002 | 0 | 1 | 0 | 1 |
| 20 | 0xf0000000 | 0x10000000 | 2 |  |  |  |  |  |  |  |

