ALU Design and Tests

The basic design of our ALU follows this finite state machine:

Diagram

Description automatically generated

Here’s a detailed description of each state:

1) **SEL**

|  |  |
| --- | --- |
|  | If the switch marked in green is on it changes state to INPUTAA (manual mode) on pressing the button marked in yellow, else it changes the state to TEST (tester mode). |

MANUAL MODE

2) **INPUTAA**

|  |  |
| --- | --- |
| A picture containing text, electronics  Description automatically generated | Use the dip - switches marked in green to enter A into the FPGA.  State changed to INPUTBB on pressing the button marked in yellow |

3) **INPUTBB**

|  |  |
| --- | --- |
| **A picture containing text, electronics  Description automatically generated** | Use the dip - switches marked in green to enter B into the FPGA.  State changed to ALUFN on pressing the button marked in yellow |

4) **ALUFN**

|  |  |
| --- | --- |
| **A picture containing text, electronics  Description automatically generated** | Use the dip - switches marked in green to enter ALUFN into the FPGA.  State changed to RESULT (if valid ALUFN entered), changed to ILLOP (if invalid ALUFN entered) on pressing the button marked in yellow. |

5) **RESULT**

|  |  |
| --- | --- |
|  | Red box - ALU 16 bit output  Yellow box - z  Blue box - v  Orange box - n  Green box - ALUFN of operation that was      performed  State changed to SEL on pressing yellow button (reset) |

8) **ILLOP**

|  |  |
| --- | --- |
|  | On pressing button in yellow state changes to ALUFN - you can now enter the correct ALUFN value |

---------------------------------------------------------------------------------------------------------------------------

TESTER MODE

2) **DISPA**

|  |  |
| --- | --- |
|  | Yellow Box - Address of test case running  Green box - Input A corresponding to the test case currently running  State changes to DISPB when slowed down clock is at its rising edge |

3) **DISPB**

|  |  |
| --- | --- |
|  | Yellow Box - Address of test case running  Green box - Input B corresponding to the test case currently running  State changes to DISPALUFN when slowed down clock is at its rising edge |

4) **DISPALUFN**

|  |  |
| --- | --- |
| **Graphical user interface  Description automatically generated** | Yellow Box - Address of test case running  Green box - ALUFN signal corresponding to the test case currently running  State changes to CHECK when slowed down clock is at its rising edge |

5) **CHECK**

|  |  |
| --- | --- |
|  | Green box - computed output  Yellow box - Address of test case running  Red box - z value (computed)  Orange box - v value (computed) Blue box - n value (computed)  Purple box - P/F  P if computed output matches answer stored in ROM, F if it doesn’t  State changes to NEXT when slowed down clock is at its rising edge |

6) **NEXT**

* If all test cases have been traversed and all have passed state changes to PASS state
* If the previous test case has failed state changes to FAIL state
* If the previous test case has passed and there are test cases left to check state changes to DISPA state and the address is incremented by 1 to correspond to the next test case

7) **PASS**

|  |  |
| --- | --- |
| **Graphical user interface  Description automatically generated with medium confidence** | If all test cases passed we reach this state, i.e our tester has completed its checking process  The green box has the address 100101 which is the last address of the test case we have indicating that it has traversed each test case from 000000 to 100101 and all have returned the value pass.  On pressing the button marked in yellow, state changes to SEL (reset) |

8) **FAIL**

|  |  |
| --- | --- |
| **Graphical user interface  Description automatically generated** | If a particular test case fails we reach this state, i.e our tester has exited since one of the cases returned the value falls  The green box shows us the computed output of the test case corresponding to the address shown in the blue box  On pressing the button marked in yellow, state changes to SEL (reset) |

Error/Fail Case:

On switching on the switch circled in yellow in the picture above the MSB computed answer from our ALU is inverted, thus giving us an error.

We have this feature to test the functionality of our tester and see if it returns FAIL in such a case or not.

Following is the table stating the corresponding functionality the ALU carries out in relation to the ALUFN signal provided. The underlined functions are extra functions implemented by us:

|  |  |
| --- | --- |
| **Function** | **ALUFN Signal** |
| ADD | 000000 |
| SUBTRACT | 000001 |
| INCREMENT BY 1 | 001000 |
| INCREMENT BY 3 | 001100 |
| MULTIPLY | 000010 |
| AND | 011000 |
| OR | 011110 |
| XOR | 010110 |
| A | 011010 |
| SHIFT LEFT (SHL) | 100000 |
| SHIFT RIGHT (SHR) | 100001 |
| SIGNED SHIFT RIGHT (SRA) | 100011 |
| CMP EQUAL (CMPEQ) | 110011 |
| CMP LESSTHAN (CMPLT) | 110101 |
| CMP LESSTHANEQUAL (CMPLE) | 110111 |
| ILLEGAL ALUFN/OPERATION (ILLOP) | Any other combination |

These are the test cases that we have hardcoded in the ROM (37 test cases + default case):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Address** | **A** | **B** | **ALUFN** | **Output** | **z** | **v** | **n** |
| 000000 | 0111111111110000 | 0111111111111111 | 000000 | 1111111111101111 | 0 | 1 | 1 |
| 000001 | 1000011111111111 | 1000011010101010 | 000000 | 0000111010101001 | 0 | 1 | 0 |
| 000010 | 1111111111111111 | 0000000000000001 | 000000 | 0000000000000000 | 1 | 0 | 0 |
| 000011 | 0100111101010111 | 1000001000010011 | 000000 | 1101000101101010 | 0 | 0 | 1 |
| 000100 | 0111111111110000 | 1000000000000001 | 000001 | 1111111111101111 | 0 | 1 | 1 |
| 000101 | 1000011111111111 | 0111100101010110 | 000001 | 0000111010101001 | 0 | 1 | 0 |
| 000110 | 1111111111111111 | 1111111111111111 | 000001 | 0000000000000000 | 1 | 0 | 0 |
| 000111 | 0100111101010111 | 0111110111101101 | 000001 | 1101000101101010 | 0 | 0 | 1 |
| 001000 | 1111111111111111 | 0000000000000001 | 000001 | 1111111111111110 | 0 | 0 | 1 |
| 001001 | 0111111111110000 | 0111111111111111 | 001000 | 0111111111110001 | 0 | 0 | 0 |
| 001010 | 0111111111111111 | 0111111111111111 | 001000 | 1000000000000000 | 0 | 1 | 1 |
| 001011 | 1111111111111111 | 0100111101010111 | 001000 | 0000000000000000 | 1 | 0 | 0 |
| 001100 | 0100111101010111 | 1000001000010011 | 001000 | 0100111101011000 | 0 | 0 | 0 |
| 001101 | 0111111111110000 | 0111111111111111 | 001100 | 0111111111110011 | 0 | 0 | 0 |
| 001110 | 0111111111111111 | 0111111111111111 | 001100 | 1000000000000010 | 0 | 1 | 1 |
| 001111 | 1111111111111111 | 0100111101010111 | 001100 | 0000000000000010 | 0 | 0 | 0 |
| 010000 | 0100111101010111 | 1000001000010011 | 001100 | 0100111101011010 | 0 | 0 | 0 |
| 010001 | 0100111101010111 | 1000001000010011 | 000010 | 0001000101110101 | 0 | 0 | 1 |
| 010010 | 0100111101010111 | 1000001000010011 | 011000 | 0000001000010011 | 0 | 0 | 1 |
| 010011 | 0100111101010111 | 1000001000010011 | 011110 | 1100111101010111 | 0 | 0 | 1 |
| 010100 | 0100111101010111 | 1000001000010011 | 010110 | 1100110101000100 | 0 | 0 | 1 |
| 010101 | 0100111101010111 | 1000001000010011 | 011010 | 0100111101010111 | 0 | 0 | 1 |
| 010110 | 0111111111110000 | 0111111111111111 | 100000 | 0000000000000000 | 0 | 1 | 1 |
| 010111 | 1000011111111111 | 1000011010101010 | 100000 | 1111110000000000 | 0 | 1 | 0 |
| 011000 | 0111111111110000 | 1000000000000001 | 100001 | 0011111111111000 | 0 | 1 | 1 |
| 011001 | 1000011111111111 | 0111100101010110 | 100001 | 0000001000011111 | 0 | 1 | 0 |
| 011010 | 0111111111110000 | 1000000000000001 | 100011 | 0011111111111000 | 0 | 1 | 1 |
| 011011 | 1000011111111111 | 0111100101010110 | 100011 | 1111111000011111 | 0 | 1 | 0 |
| 011100 | 1111111111111111 | 0000000000000001 | 110011 | 0000000000000000 | 0 | 0 | 1 |
| 011101 | 1111111111111111 | 1111111111111111 | 110011 | 0000000000000001 | 1 | 0 | 0 |
| 011110 | 1000011111111111 | 0111100101010110 | 110101 | 0000000000000001 | 0 | 1 | 0 |
| 011111 | 0011111111111000 | 0001111000000000 | 110101 | 0000000000000000 | 0 | 0 | 0 |
| 100000 | 1111111111111111 | 1111111111111111 | 110111 | 0000000000000001 | 1 | 0 | 0 |
| 100001 | 1000011111111111 | 0111100101010110 | 110111 | 0000000000000001 | 0 | 1 | 0 |
| 100010 | 0011111111111000 | 0001111000000000 | 110111 | 0000000000000000 | 0 | 0 | 0 |
| 100011 | 0000000000000000 | 0000000000000000 | 000000 | 0000000000000000 | 1 | 0 | 0 |
| 100100 | 0000000000000000 | 0000000000000000 | 000001 | 0000000000000000 | 1 | 0 | 0 |
| 100101 | default case (same as case 100011) | | | | | | |