**First Uber Cool Design**

Register file, ALU, and Fetch unit

By: Brandon Chin (A11845998), Leandro Lubrico (A12077909), and Kevin Chen (A10623152)

**Q1. Introduction**

**Q2. ISA Summary**

Instruction Formats

|  |  |
| --- | --- |
| **Name** | **Format** |
| A | [ OP ] [ FUNC / SRC REG 1 ] [ SRC REG 2 / IMMEDIATE ]  [ 3 ] [ 3 ] [ 3 ]  Ex: load $r0, $r1  000 000 001 |
| B | [ OPCODE ] [ FUNC ] [ OFFSET ]  [ 3 ] [ 1 ] [ 5 ]  Ex: bno if1  111 0 00101 |

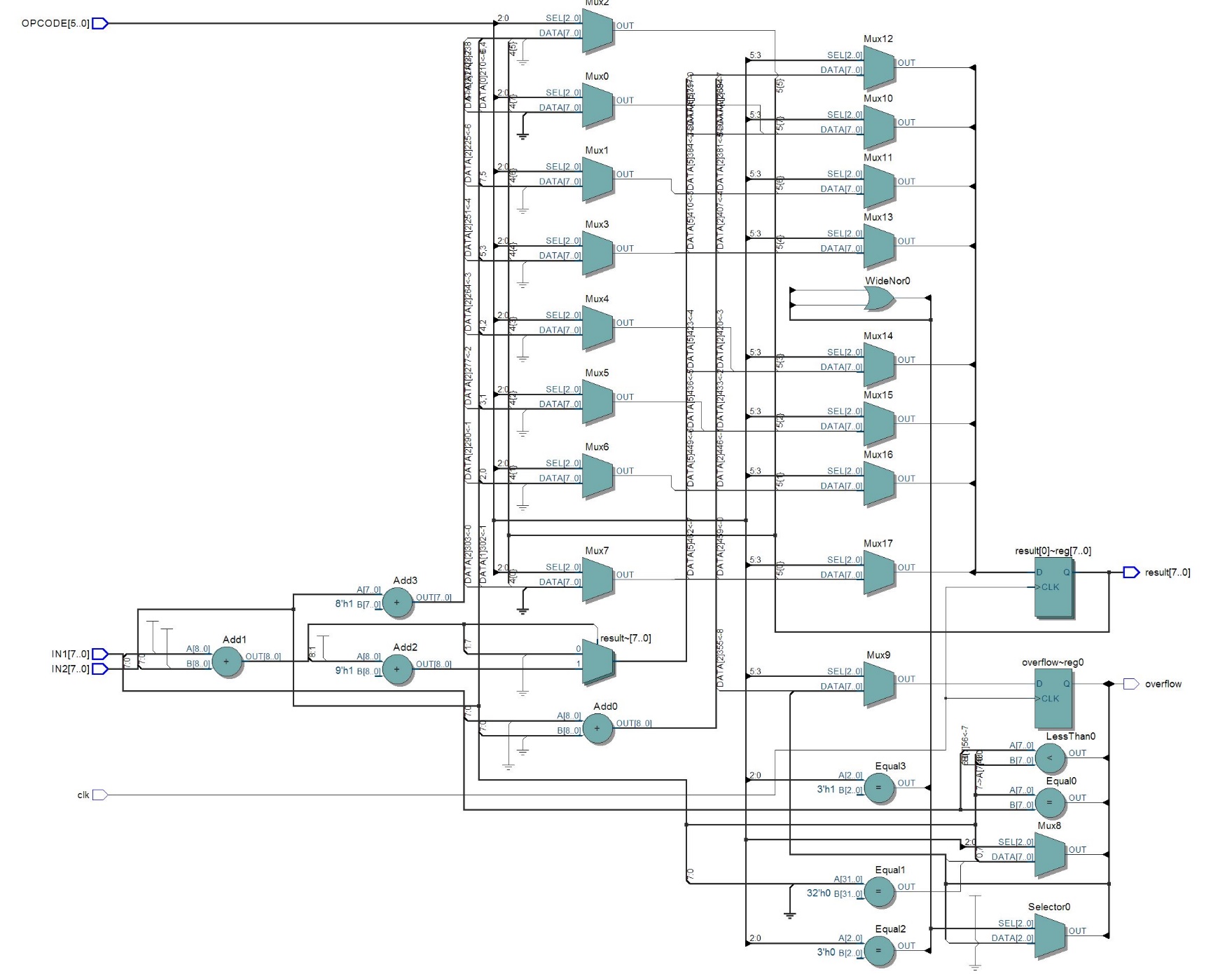
Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **OP / FUNC** | **Format** | **Operation** |
| Load (load) | 000 | A | R[$rs1] = M[ R[$rs2] ] |
| Store (store) | 001 | A | M[ R[$rs2] ] = R[$rs1] |
| Add (add) | 010 | A | R[$rs1] = R[$rs1] + R[$rs2] |
| Match (match) | 011 | A | $ov = ( R[$rs1] == R[$rs2] ) |
| Less than (lt) | 100 | A | if( R[$rs1] < R[$rs2] ) $ov = 1, else $ov = 0 |
| Distance (dist) | 101 | A | R[$rs1] = abs( R[$rs1] - R[$rs2] ) |
| Logical Shift Left (lsl) | 110 / 000 | A | R[$rs] = R[$rs] << 1 |
| Logical Shift Right (lsr) | 110 / 001 | A | R[$rs] = R[$rs] >> 1 |
| Increment (incr) | 110 / 010 | A | R[$rs] = R[$rs] + 1 |
| And with 1 (and) | 110 / 011 | A | $ov = LSB( R[$rs] ) & 1 |
| Equals 0 (eqz) | 110 / 100 | A | $ov = R[$rs] == 0 |
| Assign 0 (zero) | 110 / 101 | A | R[$rs] = 0 |
| TBD | 110 / 110 | A |  |
| Halt (halt) | 110 / 111 | A |  |
| Branch on No Overflow (bno) | 111 / 0 | B | if($ov == 0) PC = PC + SignedOffset; |
| Branch on Overflow (bof) | 111 / 1 | B | if($ov == 1) PC = PC + SignedOffset; |

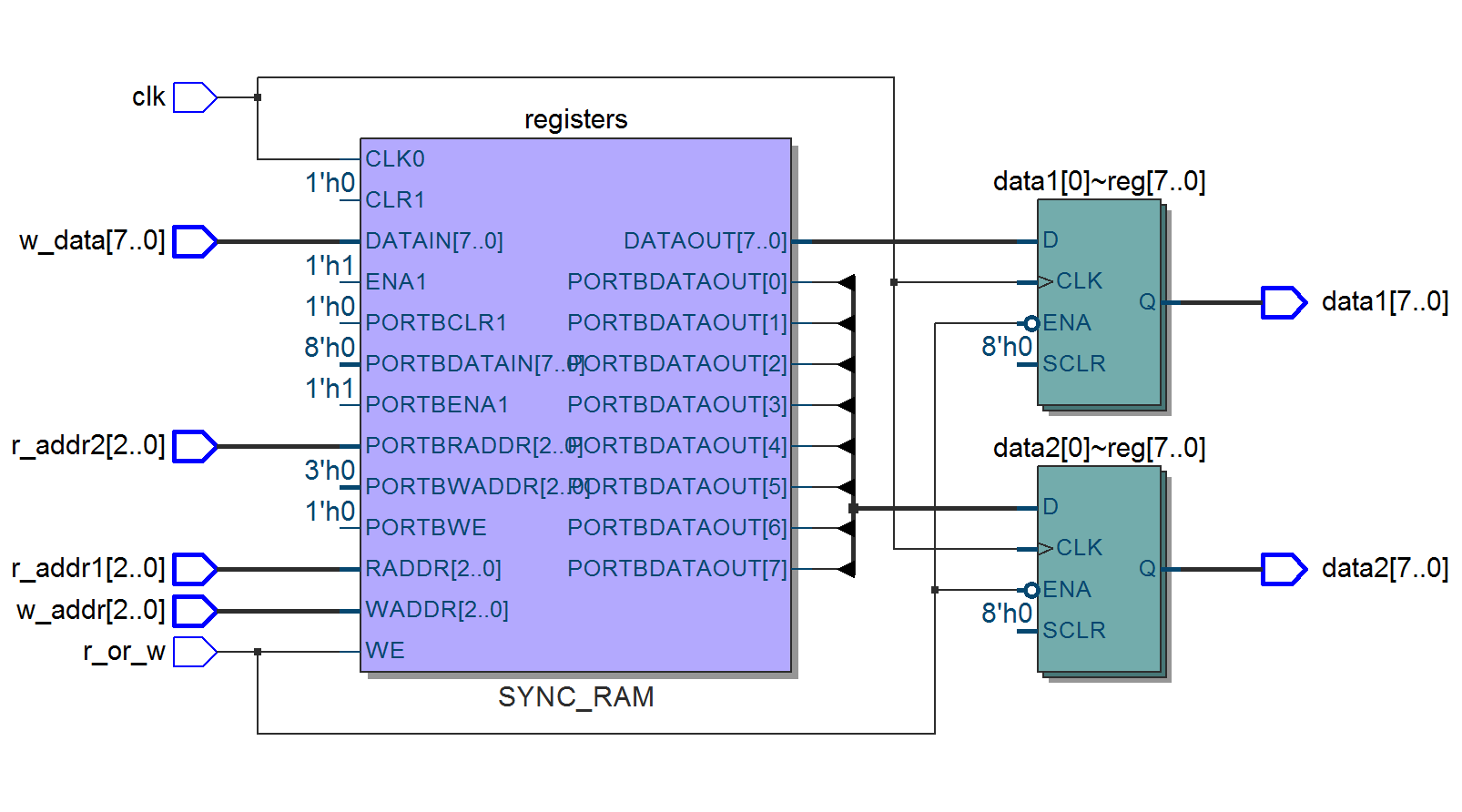
**Q3. ALU Operations demonstration**

**Q4. Schematics and Verilog models**

ALU

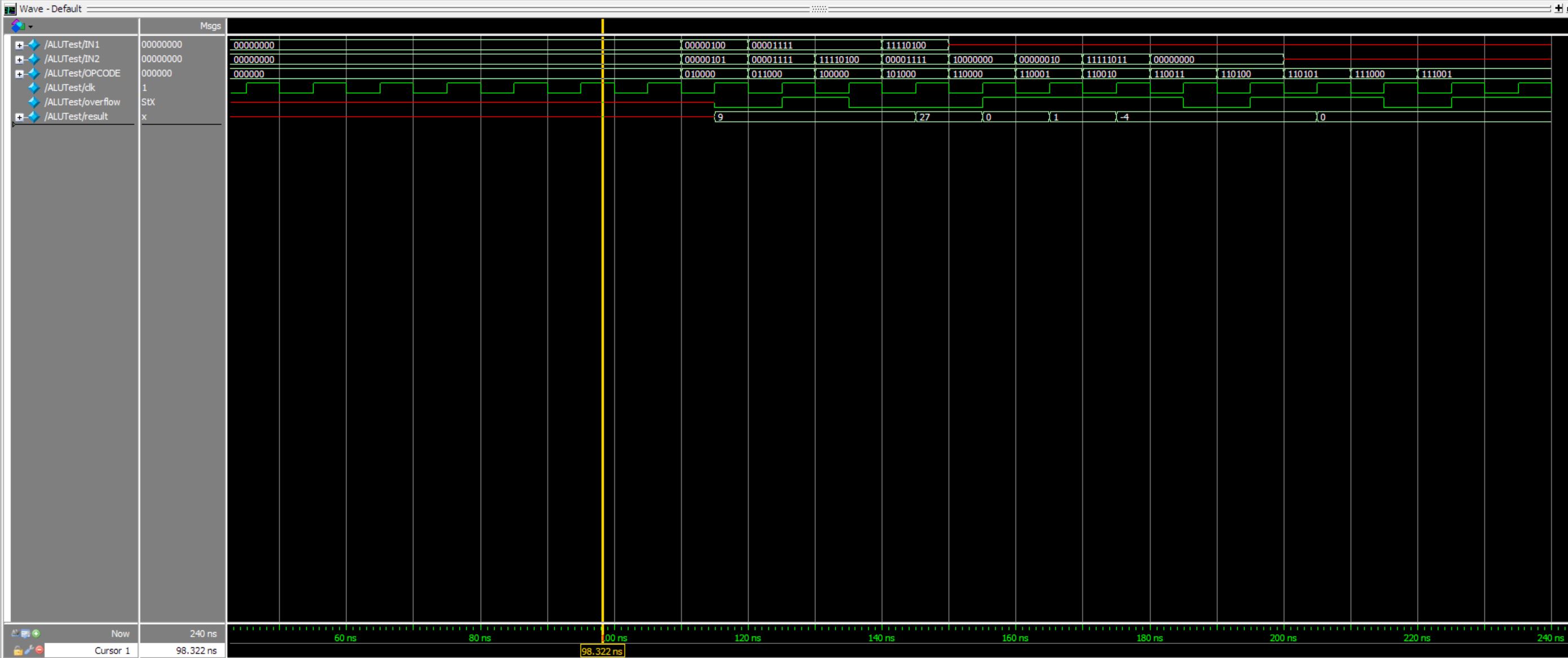


Register File



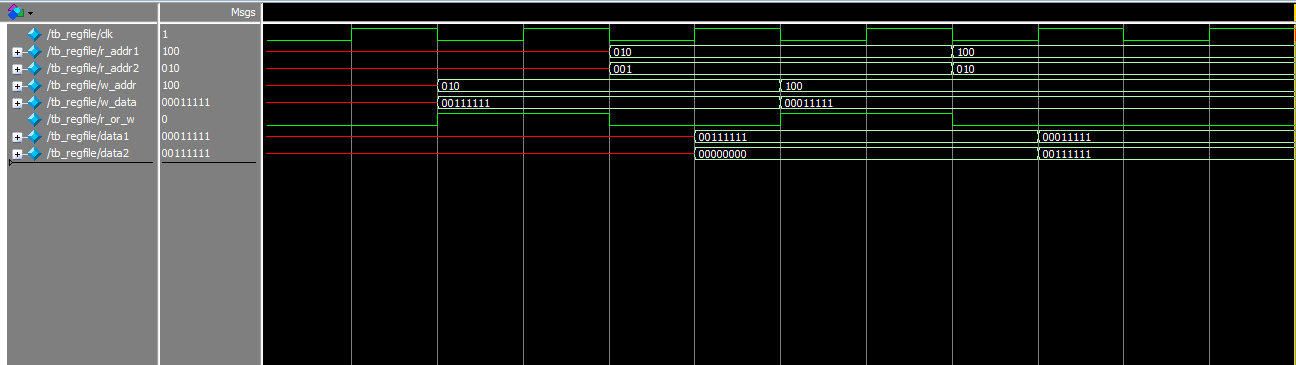
**Q5. Timing diagrams**

ALU



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | Add (010000) | Match (011000) | Less Than (100000) | Dist (101000) | LSL (110000) | LSR (110001) | Increment (110010) | And1 (110011) | Equals 0 (110100) | Zero (110101) | Branch no overflow (111000) | Branch on overflow (111001) |
| IN1 | 4 | 00001111 | 15 | -12 | Don’t care | Don’t care | Don’t care | Don’t care | Don’t care | Don’t care | Don’t care | Don’t care |
| IN2 | 5 | 00001111 | -12 | 15 | 10000000 | 00000010 | -5 | 00000000 | 00000000 | Don’t care | Don’t care | Don’t care |
| Result | 9 | Prev Val | Prev val | 27 | 00000000 | 00000001 | -4 | Prev val | Prev val | 0 | Prev val | Prev val |
| Overflow | 0 | 1 | 0 | Prev val | 1 | Prev val | Prev val | 0 | 1 | Prev val | 0 | 1 |

Register File



**Q6. ALU Operations Supported**

**Q7. Will your ALU be used for non-arithmetic instructions (e.g., address calculation, branches)?  If so, how does that complicate your design?**

**Q8. Name one thing you could have done differently in your ISA design to make your ALU design easier.**

**Q9. Name one thing you could have done differently in your ISA design to make your register file design easier.**

**Q10. What is your most complex instruction, from the standpoint of the ALU?**

**Q11. Now that your ALU is designed, are there any instructions that would be particularly straightforward to add given the hardware that is already there?**

**Q12. Is there anything you could have done in your ISA to make your fetch unit design job easier?**