**CDA 4205 Project Part I: Data Path Main Components Design**

1. **Objectives**

Using the Logisim Simulator to design the main data path components: a 32x 32 bit register file and a 32 bit arithmetic and logic unit (ALU)

1. **Tasks**
2. Model the designed 32x32-bit register file as one single module in Logisim and test the register file for correct operation by writing to and reading from different register combinations.
3. Design a 32-bit ALU to perform all the arithmetic, logic and shift operations required by your data path, and model the your designed 32-bit ALU in Logisim, then test the correct functionality of all operations implemented by the ALU.
4. **Components Design Specifications**
5. **32 x 32 bit Register File**

The register file consists of 32 x 32-bit registers and has the following interface:

Register File

RA

RB

BusA

RegWrite

BusB

RW

5

5

5

32

32

32

BusW

Clock

32

5

5

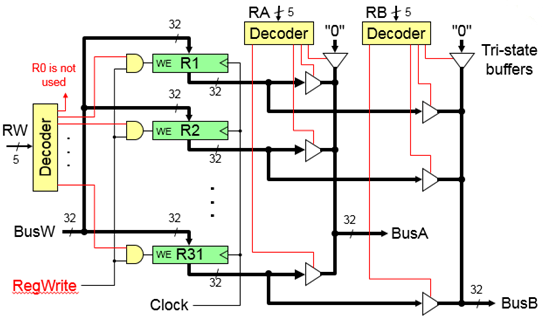
32

5

32

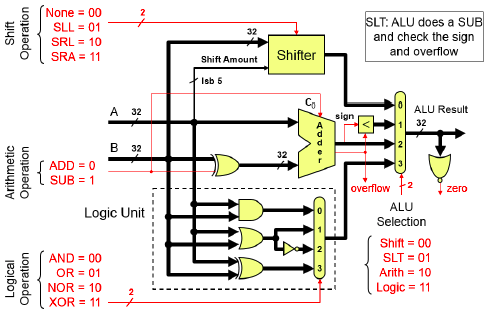
* 1. BusA and BusB: 32-bit data output busses for reading 2 registers. During read operation, the register file behaves as a combinational block and once the RA or RB have valid data, the content of the read register will appear on BusA or BusB after a certain access time.
  2. RegWrite: control signal to enable/disable register writing operation. When RegWrite is 1, register write is enabled; otherwise, is disabled.
  3. BusW: 32-bit data input bus for writing a register when RegWrite is 1
  4. RA: 5-bits selection lines to select the register to be read on BusA.
  5. RB: 5-bits selection lines to select the register to be read on BusB
  6. RW: 5-bits selection lines to select the register to be to be written with BusW.

The details of the designed 32 x 32 register file is shown in the following picture. In MIPS, register $0 is always 0, so, register 0 is not used. In this picture, each of BusA and BusB is connected to 32 tri-state buffers. Each tr-state buffer is connected to one of the 32 registers. The tri-state buffers enable signals are driven by the outputs of two 5x32 decoders, one with Ra input and the other with Rb input, to select which register puts its value on the corresponding bus. To complete the writing operation, RW is connected as the input to another 5 x 32 decoder, and the outputs of this decoder provides the register selection signals for 31 registers (register 1 to 31). These register selection signals are ANDed with RegWrite to provide the writable enable signals for individual registers.



1. **Arithmetic and Logic Unit (ALU) Design**

The designed ALU should have four main units as shown in the following picture: arithmetic unit, comparator unit, logic operation unit and shifter unit to perform arithmetic, comparison, and logic and shift operations, respectively. In order to select the output from either the arithmetic unit, the comparator unit, the logical operation unit for the shifter unit, a 32-bit 4 x 1 multiplexer is used with 2-bit ALU selection signals.



* Arithmetic Unit

1. Input signals: two input 32-bit integers A and B, one control signal ADD/SUB.
2. Output signals: A+B or A-B, Carry-Out (Cout) and Overflow signal
3. Building Components: 32-bit full adder, 32-XOR gates

The arithmetic Unit is composed of a 32-bit adder that can perform 32-bit addition and subtraction. Its internal implementation can be designed using a 32 ripple carry full adders. The inputs A and B are two 32-bit integers and the output F is A+B or A-B. The arithmetic block has a control signal, ADD/SUB, to determine whether the operation to be performed is addition or subtraction. If this signal is 0, the adder will perform addition, otherwise it will perform subtraction. The subtraction is performed using 2's complement representation as A – B = A + B' + 1. B' is computed by the 32-XOR gates in the arithmetic unit. The adder also generates Carry-Out (Cout) and Overflow signal that can be used to test for comparison purposes for unsigned and signed operations and for correctness of the obtained result. The overflow signal can be generated by XORing the carry-outs of bits 30 and 31.

* Comparator Unit

The comparator unit is mainly used for comparing signed and unsigned numbers and it shares the 32-bit full adder with the Arithmetic Unit. For the MIPS ISA implementation, we only need to compare if a number is less than another number for implementing the set on less than instructions: SLT, SLTU, SLTI, and SLTU.

In order to compare signed numbers A and B, we perform the subtraction operation A – B and then we check both the Sign of the result and the Overflow signal. The Sign of the result is the most significant bit of the result. From the flowing table, we can see that if the Sign value is not equal to the Overflow value, then A < B, otherwise, A ≥ B. This can be done by XORing the Sign and the Overflow signals. If the result is 1, this means that A < B, otherwise, means A ≥ B.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sign** | **Overflow** | **Results** | **Comments** |
| 0 | 0 | A ≥ B | No overflow and positive difference. |
| 0 | 1 | A < B | A < 0, B > 0 and overflow |
| 1 | 0 | A < B | No overflow and negative difference. |
| 1 | 1 | A ≥ B | A > 0, B < 0 and overflow |

For unsigned comparison of two numbers A and B, we need to perform a subtraction operation, A - B, and then check whether we have a Carry-Out (Cout) or not. Note, we use signed number full-adder to perform unsigned number subtraction. If Carry-Out=0, this means that there was a borrow, thus A < B. However, if Carry-Out=1 then this implies that there was no borrow and hence A ≥ B.

* Logical Operation Unit

The designed logical operation unit should perform AND, OR, NOR, and XOR logical operations for MIPS logic instruction. Functions in this unit are performed using the basic logic gates. AND, OR, NOR and XOR logical gates outputs will be forwarded to the inputs to a 32-bit 4 x 1 multiplexer. The logical operation unit has two control bits to determine the logical operation is AND, OR, NOR, or XOR, which will be connected to the input selection bits of the multiplexer.

* Shifter Block

The shifter block is used to implement SLL (shift left logic), SRL (shift right logic), SRA (shift right arithmetic) MIPS shift instructions). A shift operation takes a binary number and shifts it left or right by a specified number of bits. Logisim provides blocks for performing shift operations that can be used in the design of the Shifter unit.

* Zero Flag Detector Block

The role of this block is to set the zero-flag bit to 1 whenever the output of any operation is equal to zero. This could easily be designed using a NOR gate at the output.