[redacted]

Please make sure that the radix for all vectors in your simulation timing diagrams is switched to “Signed Decimal”!

The following VHDL Entities must be tested. At this point, you should have written and tested all these entities. Please adapt VHDL-testbenches to the following requirements and run the simulations.

- ***32-Bit_Register***

  Load the following sequence of values into the register: 0, StudentID (HEX value), 0, StudentID (HEX value).

- ***32-Bit_5-to-32_Decoder***

  Demonstrate that the decoder generates all 32 correct output signals.

- ***32-Bit_32-to-1_Multiplexer***
  Your simulation should provide the following 32bit values on all 32 inputs of the multiplexer: StudentID (HEX value) onto the first input, StudentID - 1 (HEX value) onto the second input, StudentID - 2 (HEX value onto the third input, … please continue until all 32 multiplexer inputs have a value. Show that you can map these 32 inputs to the output by changing the 5-bit_select_vector.

- ***Register_File*** (32 x ***32-Bit_Register***, 1 x ***32-Bit_5-to-32_Decoder***, 2 x ***32-Bit_32-to-1_Multiplexer***, 32 x AND_Gate)

  Load your StudentID (HEX value) into the first of the 32 registers. Then load your StudentID - 1 (HEX value) into the 2nd register, your StudentID - 2 (HEX value) into the 3rd register ... please continue until all 32 registers have a value.

  Please simulate that this only works if the Write signal is set (for the implementation of the write signals (Load enable) please see page 5 in lecture notes 7).

  Demonstrate that you can make the register content on “A data” and “B date” by changing “A address” and “B address”.

- ***Full_Adder***

  Show that your full adder implements the correct truth table for a full adder.

- ***32-Bit_Ripple_Carry_Adder*** (32 x ***Full_Adder***, C_V_Logic)

  Simulate the following:

  All numbers are in 2’s complement.

  1. positive(StudentID (HEX value) + positive value
  2. positive(StudentID (HEX value) + negative value
  3. negative(StudentID (HEX value)) + positive value
  4. negative(StudentID (HEX value)) + negative value

  5. Demonstrate worst case propagation delay. Also how long is the propagation delay?

  6. If you add a 2’s complement number to your StudentID (HEX value), what number would set the C flag and), what number would set the V flag?

- ***1-Bit_4-to-1_Multiplexer***

  Show that the multiplexer works.

- ***B_Input_Logic***

  Simulation your StudentID (HEX value) as input to the B_Input_Logic and demonstrate that you can output all 0’s, your StudentID, 1’s complement, and all 1’s by changing the select signals S_0 and S_1.

- ***32-Bit_Ripple-Carry_Adder–Subtractor*** (1 x ***B_Input_Logic***, 1 x ***32-Bit_Ripple_Carry_Adder***)

  Your simulation should provide your StudentID (HEX value) on the "A" 32-bit input vector. You should also select a suitable value for the "B" 32-bit input vector.

  Demonstrate that by changing the signals S_0, S_1, and C_{in}, you generate the following outputs:

  A, A + 1, A + B, A + B + 1, A + 1's complement(B), A + 1's complement(B) + 1, A – 1, and A

- ***1-Bit_4-to-1_Multiplexer***

  Show that the multiplexer works.

- ***1-Bit_Logic_Circuit*** (AND, OR, XOR, NOT, ***1-Bit_4-to-1_Multiplexer***)

  Demonstrate that our code can perform the following bitwise operations: AND, OR, XOR, NOT

- ***32-Bit_Logic_Circuit*** (32 x ***1-Bit_Logic_Circuit***)

  Your simulation should provide your StudentID (HEX value) on the "A" 32-bit input vector. You should also select a suitable value for the "B" 32-bit input vector.

  Demonstrate that by changing the signals S_0, and S_1, you generate the following bitwise operations: AND, OR, XOR, NOT.

- ***32-Bit_2-to-1_Multiplexer***

  Your simulation should show that the multiplexer works.

- ***32-Bit_ALU*** (1 x ***32-Bit_Ripple-Carry_Adder–Subtractor***, 1 x ***32-Bit_Logic_Circuit***, 1 x ***32-Bit_2-to-1_Multiplexer***)

  Your simulation should provide your StudentID (HEX value) on the "A" 32-bit input vector. You should also select a suitable value for the "B" 32-bit input vector.

  Demonstrate that by changing the signals S_0, S_1, S_2, and C_{in} you generate the following outputs:

  A, A + 1, A + B, A + B + 1, A + 1's complement(B), A + 1's complement(B) + 1, A – 1, A, A AND B, A OR B, A XOR B, and NOT A.

- ***32-Bit_SR-SL_Shifter_Unit*** (32 x ***1-Bit_4-to-1_Multiplexer***)

  Your simulation should provide your StudentID (HEX value) as 32-bit input vector.

Demonstrate that you can shift your StudentID by one bit to the right or one bit to the left or leave it unchanged by changing the signals S_0 and S_1.

- **Negative_Detect**

  Demonstrate that the negative-detection logic works.

- ***Zero_Detect***

  Demonstrate that the zero-detection logic works.

- ***Function_Unit*** (1 x ***32-Bit_ALU***, 1 x ***32-Bit_SR-SL_Shifter_Unit***, 1 x ***32-Bit_ra2-to-1_Multiplexer***)

  The order in with you must demonstrate the various operations of your Funtion-Unit is determined by the last digit of your student number (ID). Please see the following table on the next page for details.

  The simulation timing diagram should show these operations in the correct order and on a single screenshot.

  Furthermore, your testbench should provide your StudentID (HEX value) as 32-bit input vector on the “A” input of your Function Unit and your StudentID (HEX value) plus the last digit of your StudentID on the “B” input of your Function Unit.

		Last Digit of your Student Number (ID)									
		0	1	2	3	4	5	6	7	8	9
Order of micro-operations	1st	F=A (FS=00000)	F=A+B	F=A+B+1	F=A+1's c B	F=A+1's c B+1	F=A-1	F=A AND B	F=A OR B	F=A XOR B	F=1's c A
	2nd	F=slB	F=A OR B	F=A XOR B	F=A OR B	F=1's c A	F=A (FS=00111)	F=A OR B	F=A XOR B	F=1's c A	F=A+1
	3rd	F=A+1	F=A (FS=00000)	F=A+B	F=A+B+1	F=A+1's c B	F=A+1's c B+1	F=A-1	F=A AND B	F=A OR B	F=A XOR B
	4th	F=A AND B	F=srB	F=A AND B	F=A XOR B	F=A (FS=00111)	F=A XOR B	F=srB	F=B	F=A+1	F=A (FS=00000)
	5th	F=A+B	F=A+1	F=A (FS=00000)	F=A+B	F=A+B+1	F=A+1's c B	F=A+1's c B+1	F=A-1	F=A AND B	F=A OR B
	6th	F=B	F=1's c A	F=B	F=slB	F=srB	F=B	F=1's c A	F=slB	F=A (FS=00111)	F=slB
	7th	F=A+B+1	F=A+1's c B	F=A+1	F=A (FS=00000)	F=A+B	F=A+B+1	F=A+1's c B	F=A+1's c B+1	F=A-1	F=A AND B
	8th	F=srB	F=slB	F=1's c A	F=B	F=slB	F=srB	F=B	F=A (FS=00111)	F=srB	F=B
	9th	F=A+1's c B	F=A+B+1	F=A+1's c B+1	F=A+1	F=A (FS=00000)	F=A+B	F=A+B+1	F=A+1's c B	F=A+1's c B+1	F=A-1
	10th	F=1's c A	F=B	F=srB	F=A+1's c B+1	F=B	F=1's c A	F=slB	F=srB	F=B	F=A (FS=00111)
	11th	F=A (FS=00111)	F=A+1's c B+1	F=A+1's c B	F=A-1	F=A+1	F=A (FS=00000)	F=A+B	F=A+B+1	F=A+1's c B	F=A+1's c B+1
	12th	F=A OR B	F=A XOR B	F=slB	F=1's c A	F=A XOR B	F=slB	F=A (FS=00111)	F=1's c A	F=slB	F=srB
	13th	F=A-1	F=A (FS=00111)	F=A-1	F=A AND B	F=A OR B	F=A+1	F=A (FS=00000)	F=A+B	F=A+B+1	F=A+1's c B
	14th	F=A+1's c B+1	F=A AND B	F=A (FS=00111)	F=srB	F=A-1	F=A AND B	F=A+1	F=A (FS=00000)	F=A+B	F=A+B+1
	15th	F=A XOR B	F=A-1	F=A OR B	F=A (FS=00111)	F=A AND B	F=A OR B	F=A XOR B	F=A+1	F=A (FS=00000)	F=A+B

- ***Datapath*** (1 x ***Register_File***, 1 x ***Function_Unit***, 2 x ***32-Bit_2-to-1_Multiplexer***)

  Please provide a clock signal for the registers in your register-file. The clock signal must be appropriate for the worst-case propagation delay of your Function Unit.

  Load your StudentID (HEX value) into the first of the 32 registers. Then load your StudentID - 1 (HEX value) into the 2nd register, your StudentID - 2 (HEX value) into the 3rd register ... please continue until all 32 registers have a value.

  The load operation should be via “Data in” port and “MUX D”. See Figure 1: Register File and Functional Unit on Project 1 -Datapath Design -Part B assignment.

  Once all 32 registers are loaded, use the last digit of your student number to select the destination-register (D address).

  Use the (last digit of your student number + 5) to select the source-register A (A address).

  Use the (last digit of your student number + 15) to select the source-register B (B address).

  The order in with you must demonstrate the various operations of your Datapath is determined by the last digit of your student number (ID). Please see the table on the previous page for details. The same order as for the simulation of the Function Unit.

  Maintain the current configuration but switch the “MUX B” to “Constant in”. Your simulation should provide your StudentID (HEX value) on the “Constant in” port. Execute the various operations of your Datapath in the order determined by the last digit of your student number (ID) but only those 10 operations that require data on the “B” input of your Function Unit.