Final Project
ENSC 350

Group Members
Davis, Ramazani & Samurai

Project submitted in partial fulfillment of the requirements for ENSC 350 towards a Bachelor Degree in Engineering Science

Table of Contents

Introduction **2**

Software Used **2**

- Quartus 2
- ModelSim Alberta 2

Arithmetic Unit **2**

- Brief 2
- **Design** **3**
  - Port signals 3
- Simulation 4
  - Functional Verification 4
  - Timing Verification 4

Shift Unit **5**

- Brief 5
- Design 5
  - Port signals 5
- Simulation 6
  - Functional Verification 6
  - Timing Verification 6

Logic Unit **7**

- Brief 7
- Design 7
- Simulation 8
  - Functional Simulation 8
  - Timing Simulation 8
- Netlist View 9

Execution Unit **9**

- Brief 9
- Design 9
- Simulation 11
  - Functional Simulation 11
  - Timing Simulation 11

Appendix **12**

- **Appendix A - Listing.pdf** **12**

Introduction

For our final project, we were charged with the duty to design a complete RISC-V Execution Unit, compatible with RV64I, capable of executing instructions such as addition, subtraction, logical and arithmetic shifting, and logical operations. The project proved to be very engaging and brought the theory learned in class into practical terms, which enforced our landing and understanding of how an execution unit operates.

Our design is broken down into 4 main units. These units are the Arithmetic Unit, Shift Unit, Logic Unit, and Execution Unit. Each unit's purpose and design are discussed further in the report.

Software Used

Quartus

We used Quartus to perform Functional and Timing Verifications on our entities. Our entities were synthesized for the Cyclone IV FPGA. Compilation of our design produces standard netlist files, with file extension .vho, which are generated for simulation tools and timing analysis tools with software such as ModelSim Alberta[1]. Another important file used for simulations is the .sdo file, which is the industry-standard Standard Delay Format. .sdo files allow us to back-annotation for simulations in software such as ModelSim Alberta.[2]

ModelSim Alberta

ModelSim Alberta was used to run both our Functional and Timing simulations. We were able to measure propagation delays produced by our design with the aid of the .vho and .sdo files that Quartus automatically created when running compilations on our designs.

Arithmetic Unit

Brief

The Arithmetic Unit (ArithUnit) can be seen as the core of the execution unit as many instructions require a calculation to be made and the ArithUnit makes this possible.

[1] Intel Glossary - VHDL Output File (.vho) Definition

[2] Intel Glossary - Standard Delay Format Output File (.sdo) Definition

Design

Port signals

- Input Signals
    - A, B - 64-bit input signals to the ShiftUnit. A contains the operand in RS1(*Source Register 1*), where B is either from the 12-bit immediate RS2(*Source Register 2*).
    - AddnSub- 1-bit Control Signal for CarryIn '0' for addition, '1' for subtraction
    - ExtWord - 1-bit Control Signal determining whether output should be Sign Extended
- Output Signals
    - AddY - 64-bit output signal passed to the ShiftUnit
    - Y - 64-bit output (disconnected)
    - Cout - 1-bit status signal indicating the carry out (disconnected)
    - Ovfl - 1-bit status signal indicating whether overflow has occurred (disconnected)
    - Zero - 1-bit status signal indicating whether A = B
    - AltB - Status Signal indicating A < B (signed)
    - AltBu - Status Signal indicating A < B (unsigned)

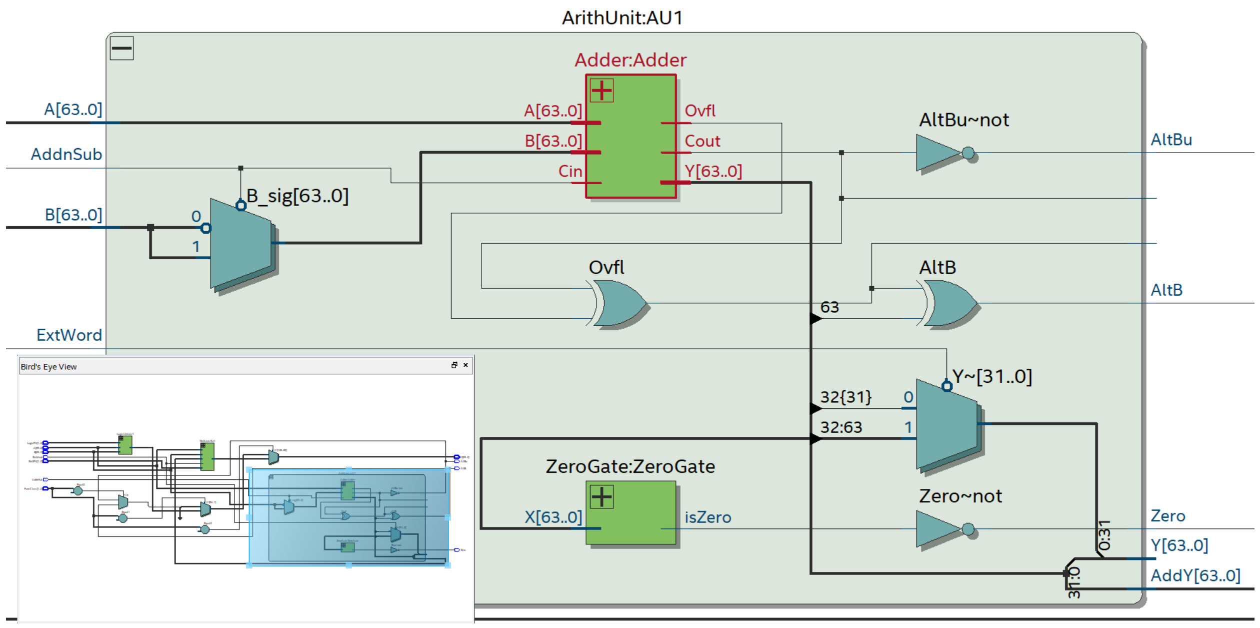


Figure 1 : Post Synthesis RTL view - Arith Unit

The Arithmetic Unit consists of two separate entities, the Adder (*Adder.vhd*), and the ZeroGate (*ZeroGate.vhd*). Adder is implemented by simply extending A and B to 65-bits to allow for the carry bit in the MSB. ZeroGate is a Tree structure instantiating primitive or gates and negating that final bit to determine whether A - B = 0.

Simulation

Functional Verification

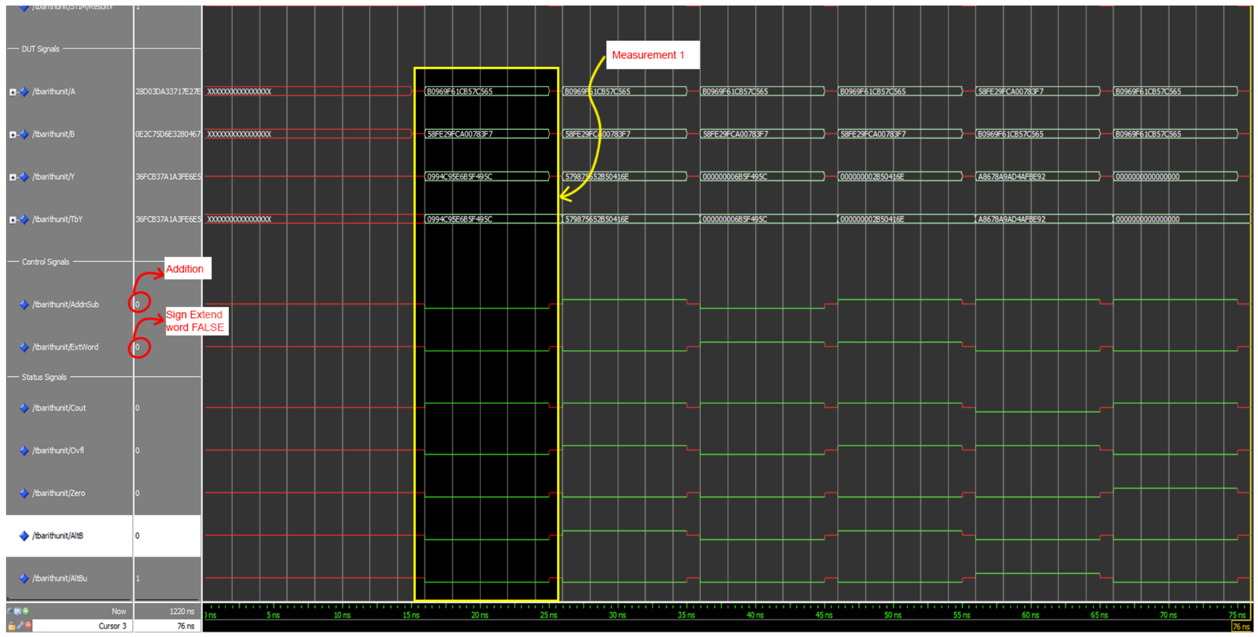


Figure 2: Functional SImulation - ArithUnit

From *Figure 2* shows Measurement 1 performing a 64-bit addition operation on A and B, then

Timing Verification

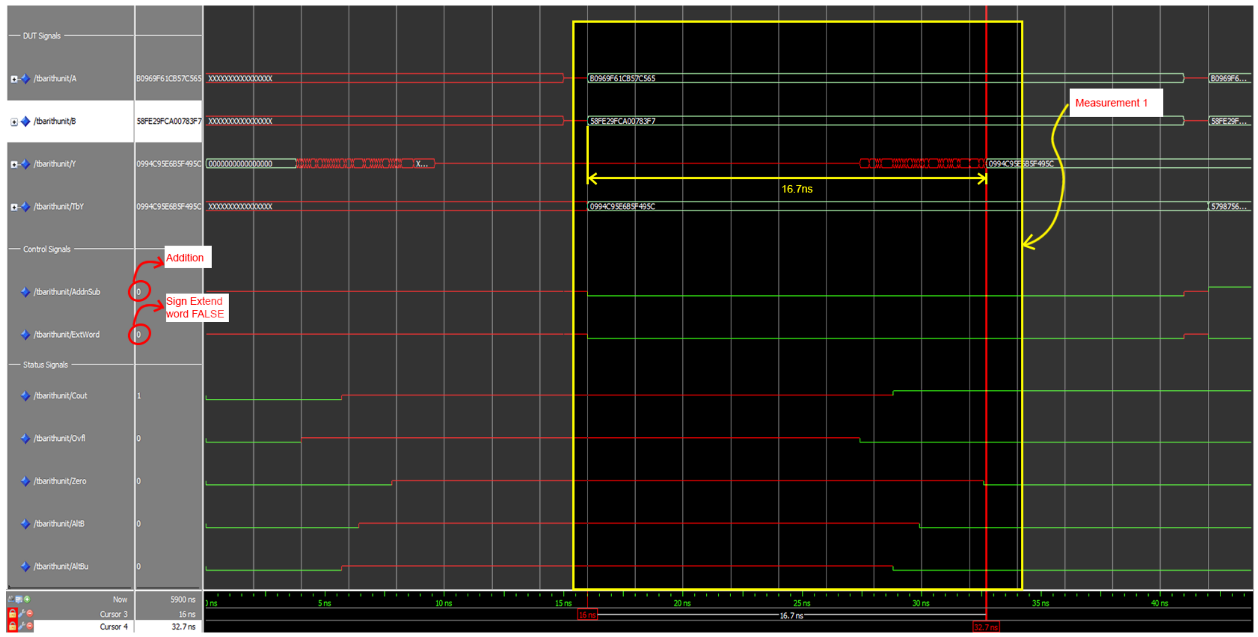


Figure 3: Timing SImulation - ArithUnit

From *Figure 3* shows the same Measurement 1 performing a 64-bit addition operation on A and B with the propagation delay of 16.7ns

- Which measurement has the largest propagation delay? Does this make sense?

Shift Unit

Brief

The Shift Unit as the name implies is responsible for the various shift instructions described in the *RISC-V Instruction Set* such as the Shift Left Logical (***SLL***), Shift Right Logical (***SRL***), and Shift Right Arithmetic (***SRA***). In addition, the ShiftUnit is also responsible for passing through and Sign Extending (if required) the output from the *ArithUnit.*

Design

The Shift Unit (*ShiftUnit.vhd*) consists of three separate entities for the three shift instructions (*SLL64.vhd*, *SRL64.vhd*, and *SRA.vhd*). Each Shift Function is made up of three 4-channel, 64-bit MUXs in cascade where the Selection Lines are chosen from the lower 6-bits from the Input ‘B’. The Choice of using three 4-channel MUXs instead of one single 64-channel MUX is to reduce the fan-in. Instead of having each bit from the operand connected to 64 device inputs, they only need to connect to 4 now.

Port signals

- Input Signals
  - A, B, and C - 64-bit input signals to the ShiftUnit. A contains the operand in RS1(*Source Register 1*) to be shifted, where the lower 5 / 6 bits of B are either from the 12-bit immediate RS2(*Source Register 2*) and contain the ”ShiftCount”. C contains the output from the ArithUnit and is passed through the ShiftUnit.
  - ShiftFN - 2-bit Control Signal for selecting between the three Shift Functions
  - ExtWord - 1-bit Control Signal determining whether output should be Sign Extended
- Output Signals
  - Y - 64-bit output signal from the ShiftUnit

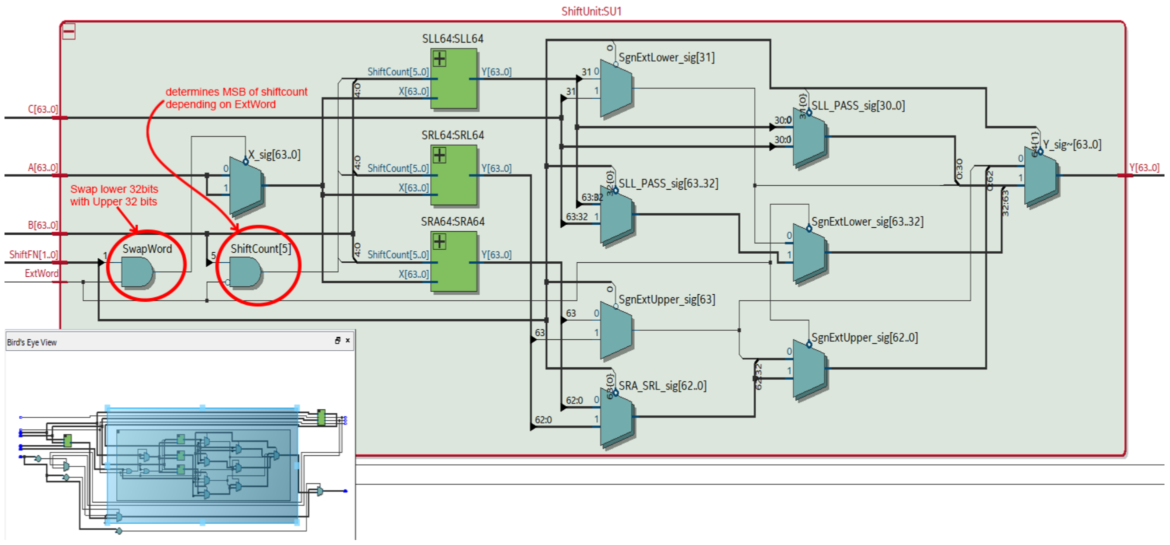


Figure 4: RTL viewer - Shift Unit

Simulation

Functional Verification

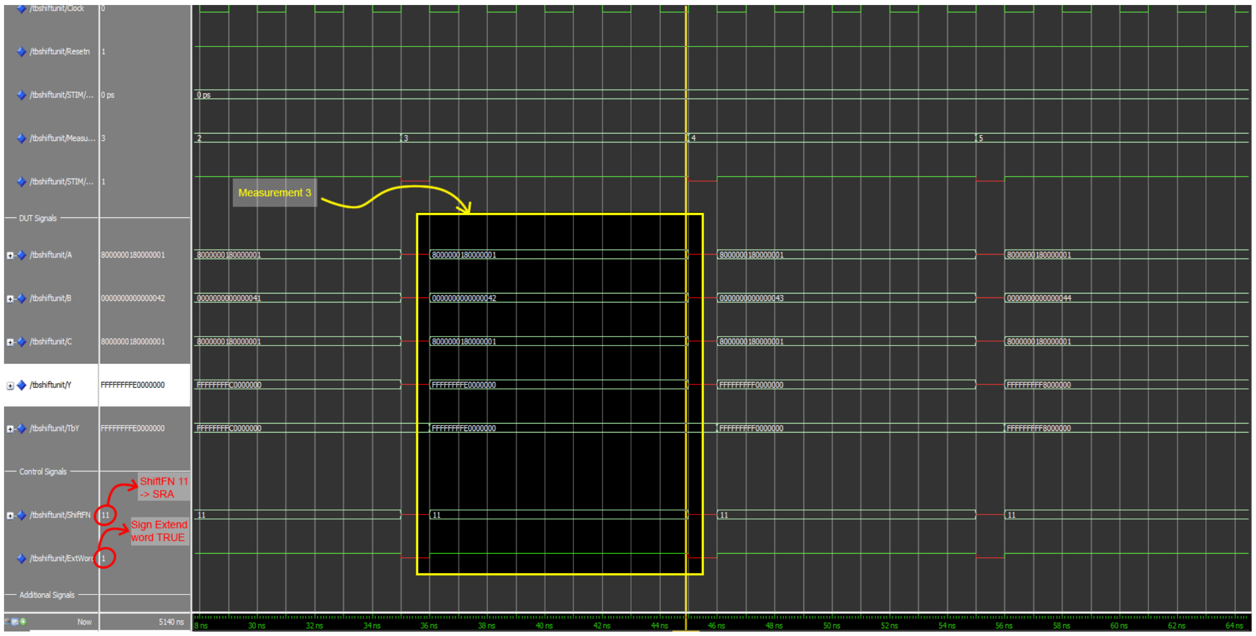


Figure 5: Functional Simulation - Shift Unit

From *Figure 5* shows Measurement 3 performing a 2-bit (Lower 5-bits of B) SRA operation on the lower 32-bits of A, and Sign Extends result to 64-bits

Timing Verification

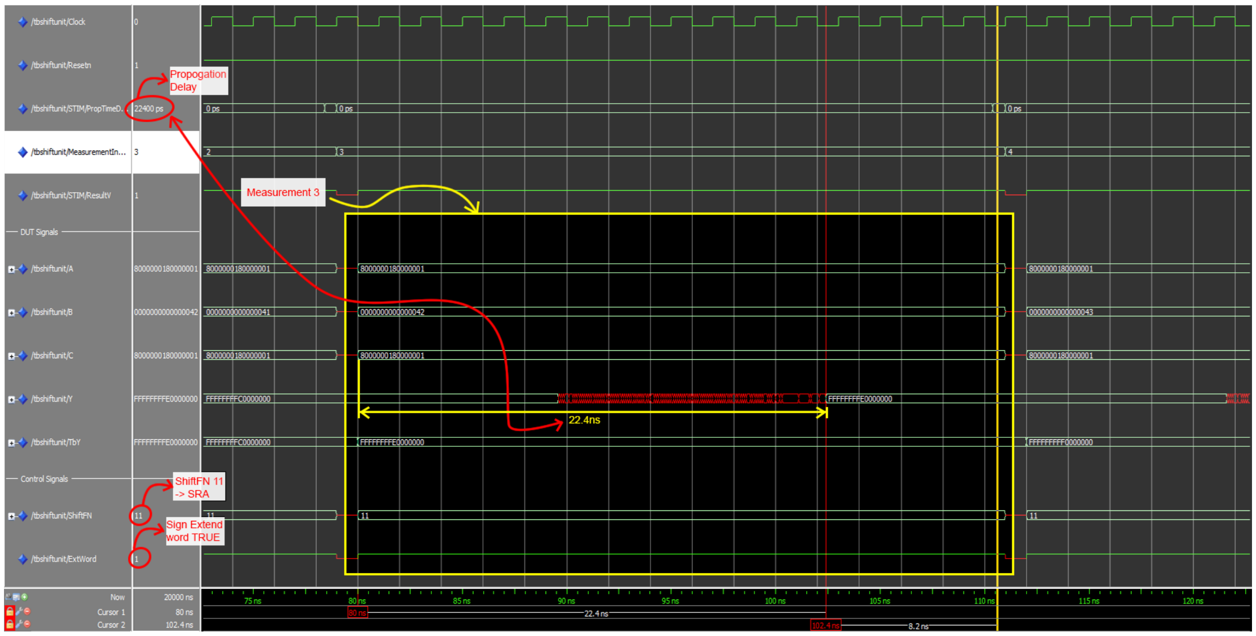


Figure 6: Timing Simulation - Shift Unit

From *Figure 6* shows the same Measurement 3 performing a 2-bit (Lower 5-bits of B) SRA operation on the lower 32-bits of A, and Sign Extends result to 64-bits. As the input A,B and C arrive, it takes 22.4ns for input to propagate and stabilize and arrive at Y.

Logic Unit

Brief

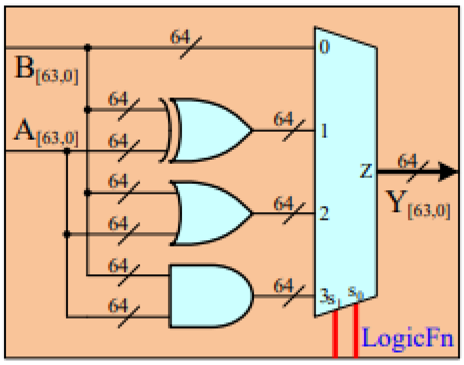
The Logic Unit is very primitive as it is essentially created using the 3 basic logic gates to perform the *RISC-V Instruction Sets* (**XOR**), (**OR**) and (**AND**). The unit can also pass through the input signal B to the output where it will be used for the **LUI** instruction.

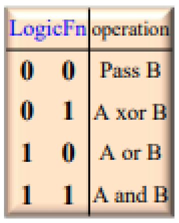
Design

The Logic Unit (*LogicUnit.vhd*) starts performing the 4 execution instructions by passing in the two 64-bit inputs A and B. Next, the two inputs pass through a xor gate, an or gate as well as an and gate. Finally, the outputs from these gates and a pass through from input B form the inputs to a 4 to 1 MUX. Lastly, the LogicFn control signal will determine the final operation to be passed through to the output.

<u>Port signals</u>

- Input signals
    - A, B - These 64-bit signals are the inputs to the overall unit and depend on what the instruction being executed is. For XOR, OR and AND, A would be RS1 (Source register 1) value and B would be RS2 (Source register 2) value. For LUI, A would be the PC (Program counter) value and B would be SextU (Sign extended U-type format instruction) value.
    - LogicFn - This 2-bit signal is crucial as it forms the input to the MUX ultimately deciding which instruction is performed.
- Output signals
    - Y - This 64-bit signal acts as the output from this unit and later goes on to become the RD (Destination register) value in the Register File.





LogicFn		operation
0	0	Pass B
0	1	A xor B
1	0	A or B
1	1	A and B

Figure 7: Circuit Diagram for the Logic Unit entity

Simulation

Functional Simulation

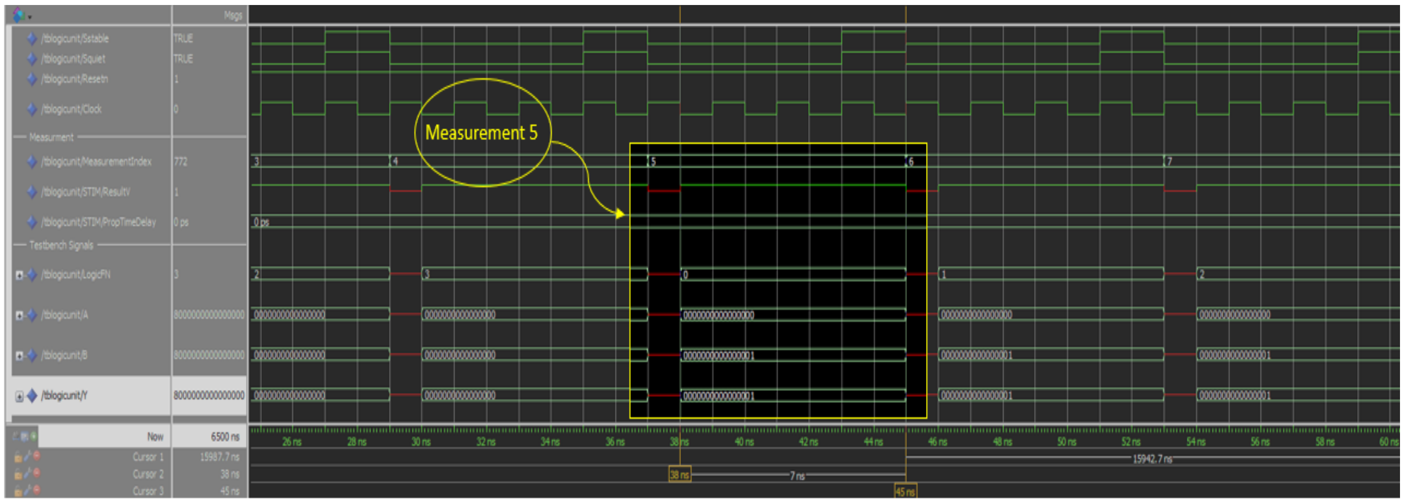


Figure 8: Functional Simulation - Logic Unit

From Figure 8, we see that measurement 5 performs the LUI operation since LogicFn is “00”. In this case, the 64-bit input value B gets passed into the output Y. The cursors above measure the duration of each measurement where in this case it is 7ns.

Timing Simulation

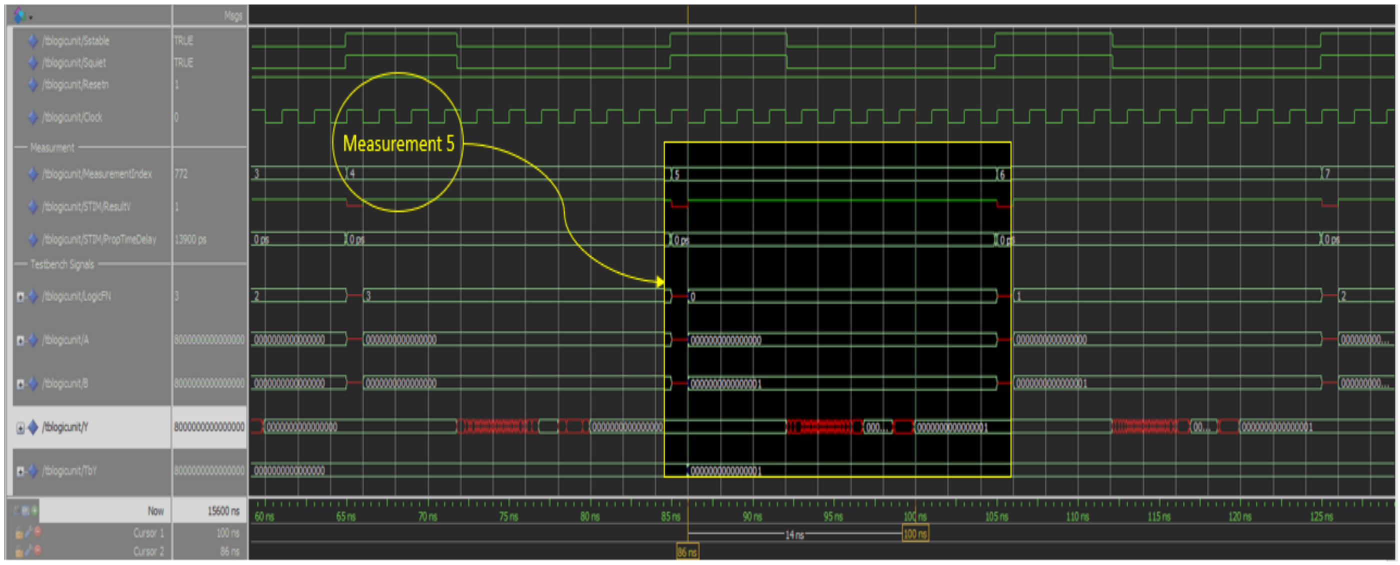


Figure 9: Timing Simulation - Logic Unit

From Figure 9, we see the same measurement 5 where it takes into account propagation delay when the instruction is executing. In this case, the propagation delay was measured to be about 14 ns.

Netlist View

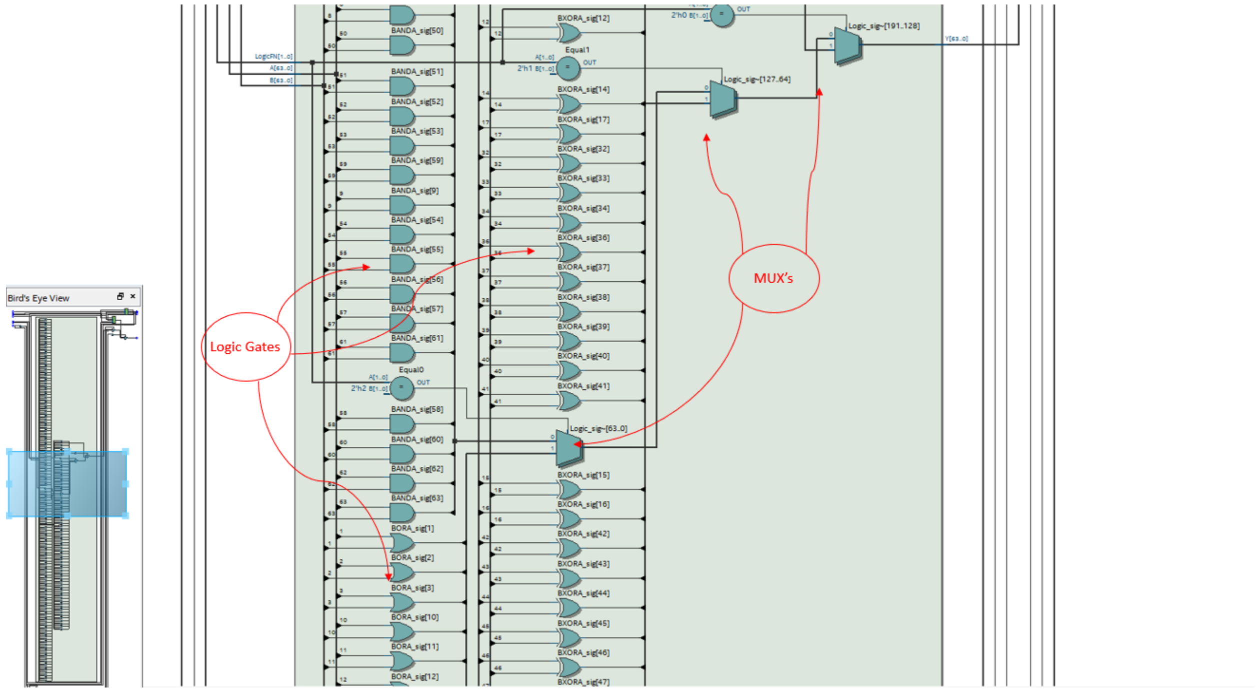


Figure 10: Post-Synthesis RTL view - Logic Unit

From Figure 10, we can see the internal view of the logic unit showing the 3 logic gates for each of the 64-bit inputs and the MUX's selecting the desired instruction.

Execution Unit

Brief

The execution unit puts everything together. It consists of 3 output signals and 5 input signals. The 5 input signals are given as execution functions, namely FuncClass, LogicFN, ShiftFN, AddnSub and ExtWord. We use std_logic and std_logic_vector for its interface signals to make them compatible with the types created by Quartus's netlist.

Design

The Execution Unit (*ExecUnit.vhd*) consists of three separate entities (*LogicUnit.vhd*, *ArithUnit.vhd*, and *ShiftUnit.vhd*).

<u>Port signals</u>

- Input signals
    - A, B - These 64-bit signals are the inputs to the Execution Unit and depend on what the instruction being executed is. A contains the operand in RS1(*Source Register 1*), where B is either from the 12-bit immediate or RS2(*Source Register 2*).
    - FuncClass - This 2-bit control signal is crucial as it forms the selection to the MUX deciding between the output of the Logic, Shift or Arithmetic functions.

- LogicFN - 2-bit Control signal controls the function of Logic Unit (*LUI, XOR, OR, AND*)
- ShiftFN - 2-bit Control signal controls the function of Shift Unit (*PASS, SLL, SRL, SRA*)
- AddnSub - 1-bit Control Signal into ArithUnit choosing between addition and subtraction
- ExtWord - 1-bit Control Signal used to sign extend operand

- Output signals
  - Y - 64-bit signal acts as the output from the ExecUnit.
  - Zero - 1-bit Status signal indicating A = B
  - AltB - Status Signal indicating A < B (signed)
  - AltBu - Status Signal indicating A < B (unsigned)

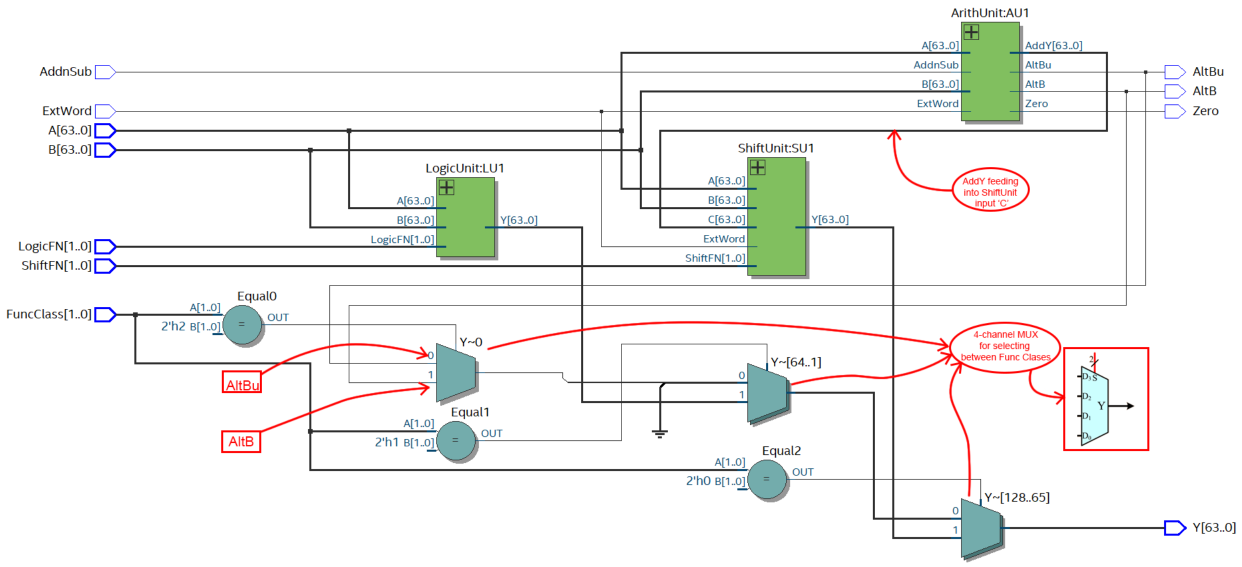


Figure 11 : Post-Synthesis RTL view - ExecUnit

Simulation

Functional Simulation

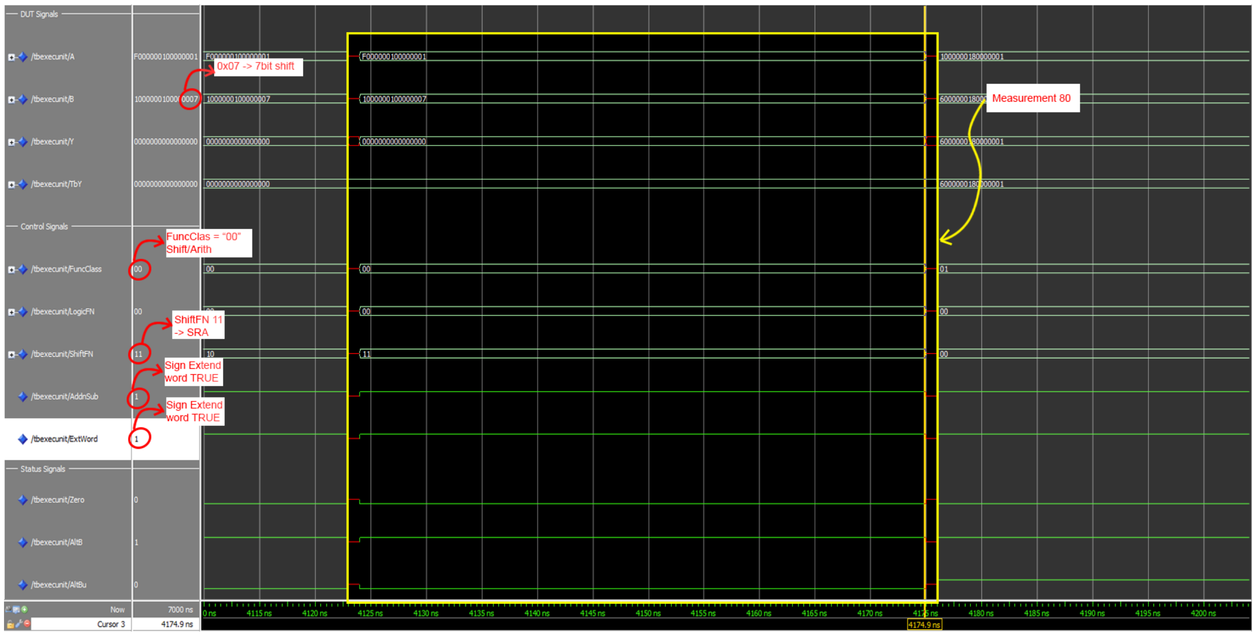


Figure 12 : Functional Simulation - ExecUnit

From *Figure 12* shows Measurement 80 performing a 7-bit (Lower 5-bits of B) SRA operation on the lower 32-bits of A, and Sign Extends result to 64-bits

Timing Simulation

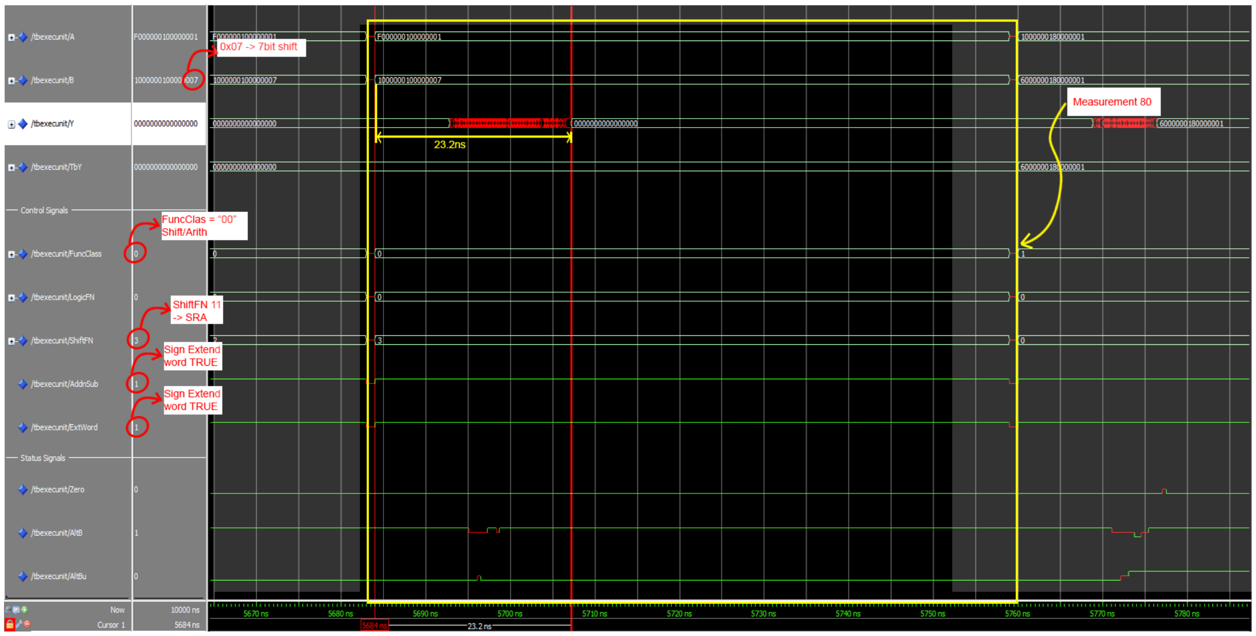


Figure 13 : Timing Simulation - ExecUnit

From *Figure 13* shows the same Measurement 80 performing a 7-bit (Lower 5-bits of B) SRA operation on the lower 32-bits of A, and Sign Extends result to 64-bits. As the input A,B and C arrive, it takes 22.3ns for input to propagate and stabilize and arrive at Y

Appendix

Appendix A - Listing.pdf