



**UNIVERSITI KUALA LUMPUR
MALAYSIAN SPANISH INSTITUTE**

SAB36603

ADVANCE DIGITAL DESIGN AND FPGA

LAB 5

VERILOG DESIGN AND SIMULATION WITH QUARTUS II

PREPARED BY:

STUDENT'S NAME	ID NUMBER
MUHAMMAD ADIEL ADAM BIN MOHD NOOR	54222121212
KHAIRUL ANWAR BIN KHAIRUL SALLEH	54215121186

LECTURER'S NAME: DR. MOHD REZAL BIN MOHAMED

1. Introduction

The ME2200 Digital Systems Lab 5 delves into the realm of Verilog Design and Simulation using Quartus II software. This laboratory session aims to equip students with the essential skills required to create, analyze, and simulate Verilog designs for programmable logic devices, specifically field-programmable gate arrays (FPGAs). Through a series of hands-on exercises, students will explore the intricacies of Verilog code, design entry processes, synthesis, functional simulation, fitting, timing analysis, and programming within the Quartus II environment. By immersing themselves in this practical experience, students will gain a deeper understanding of digital systems design and implementation, paving the way for proficiency in FPGA-based projects and applications. This laboratory report encapsulates the journey of discovery and learning undertaken during the ME2200 Digital Systems Lab 5, shedding light on the fundamental concepts and methodologies essential for modern digital design practices.

2. Objectives

The objectives of this laboratory are:

- To get started on using Quartus II software to create Verilog designs, analyze, and perform simulations.

Equipment Required:

- One PC running Microsoft® Windows 2000/XP/Vista® with a minimum of 512 MB RAM with USB ports, and installed with Altera Quartus II v13.0 SP1 software

3. Schematic/HDL Language

3.1 Creating a New Project

- Create a New Project Wizard, name the project and top-level entity as *addersubtractor*.
- Click on the **Finish** button to create the **New Project Wizard**. Then, proceed to create a new Verilog HDL file name as *addersubtractor.v*.
- Use the Verilog template as shown below.

```
//Top-level module
module addersubtractor (A, B, Clock, Reset, Sel, AddSub, Z, Overflow);
    parameter n = 16;
    input [n-1:0] A, B;
    input Clock, Reset, Sel, AddSub;
    output [n-1:0] Z;
    output Overflow;
    reg SelR, AddSubR, Overflow;
    reg [n-1:0] Areg, Breg, Zreg;
    wire [n-1:0] G, H, M, Z;
    wire carryout, over_flow;

    //Define combinational logic circuit
    assign H = Breg ^ {n{AddSubR}};
    mux2to1 multiplexer (Areg, Z, SelR, G);
    defparam multiplexer.k = n;
    adderk nbit_adder (AddSubR, G, H, M, carryout);
    defparam nbit_adder.k = n;
    assign over_flow = carryout ^ G[n-1] ^ H[n-1] ^ M[n-1];
    assign Z = Zreg;

    //Define flip-flops and registers
    always @(posedge Reset or posedge Clock)
        if (Reset == 1)
            begin
                Areg <= 0; Breg <= 0; Zreg <= 0;
                SelR <= 0; AddSubR <= 0; Overflow <= 0;
            end
        else
            begin
                Areg <= A; Breg <= B; Zreg <= M;
                SelR <= Sel; AddSubR <= AddSub; Overflow <= over_flow;
            end
    end

endmodule

//k-bit 2-to-1 multiplexer
module mux2to1 (V, W, Sel, F);
```



```

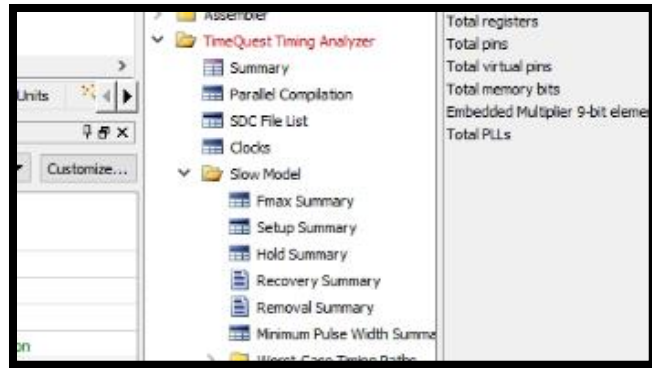
        parameter k = 8;
        input [k-1:0] V, W;
        input Sel;
        output [k-1:0] F;
        reg [k-1:0] F;

        always @(V or W or Sel)
            if (Sel == 0)
                F = V;
            else
                F = W;
    endmodule
//k-bit adder
module adderk (carryin, X, Y, S, carryout);
    parameter k = 8;
    input carryin;
    input [k-1:0] X, Y;
    output [k-1:0] S;
    output carryout;
    reg [k-1:0] S;
    reg carryout;

    always @(X or Y or carryin)
        {carryout, S} = X + Y + carryin;
endmodule

```

- Compiling the Verilog Code by selecting **Processing > Start Compilation**, or click the toolbar  icon to begin with compilation.
- After completed compiling the Verilog Code, click on the  symbol next to the **TimeQuest Timing Analyzer** to expand the section of the report. Select on **TimeQuest Timing Analyzer > Slow Model > Fmax Summary** to display the table as shown below.



adder_subtractor.v

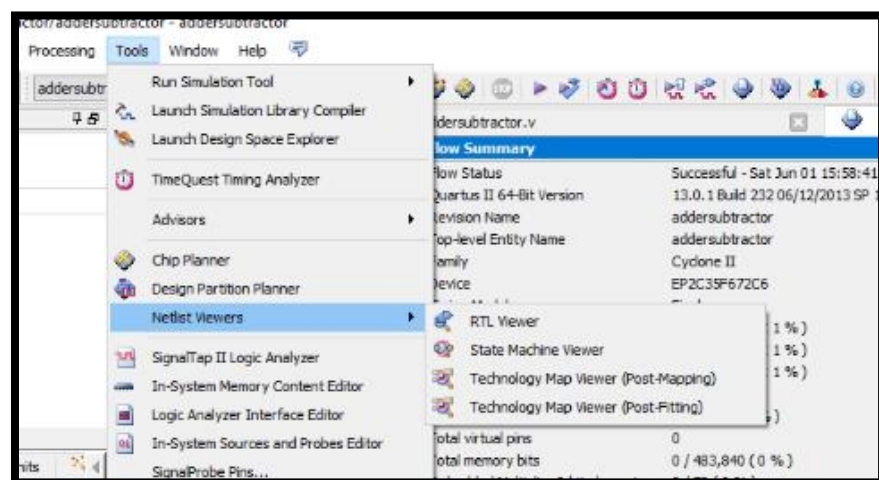
Slow Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	251.51 MHz	251.51 MHz	Clock	

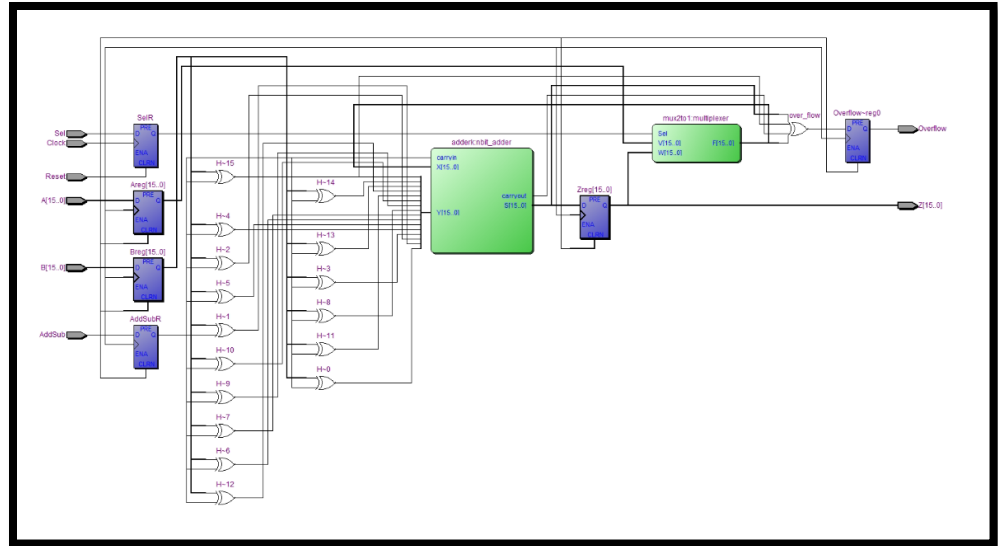
- To see the paths in the circuit that limit the fmax, select **TimeQuest Timing Analyzer > Slow Mode > Worst-Case Timing Paths > Setup 'Clock'** which obtain in the table shown as below.

10	1.018
11	1.048
12	1.055
13	1.093
14	1.127
15	1.133
16	1.135
17	1.161
18	1.170
19	1.191
20	1.212
21	1.223
22	1.247
23	1.250
24	1.262
25	1.270
26	1.277
27	1.277
28	1.278
29	1.282

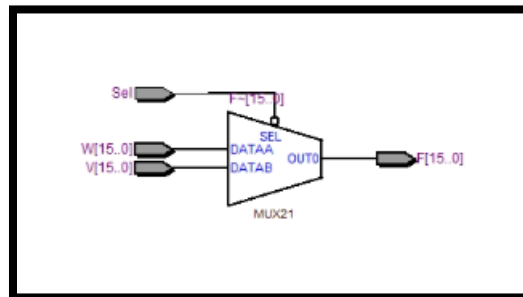
Slow Model Setup: 'Clock'								
	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	0.524	AddSubR	Overflow~reg0	Clock	Clock	4,500	-0.002	4.010
2	0.589	SelR	Overflow~reg0	Clock	Clock	4,500	-0.002	3.945
3	0.708	Zreg[5]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.826
4	0.722	Zreg[0]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.812
5	0.749	Zreg[2]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.785
6	0.850	Zreg[4]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.684
7	0.910	Breg[1]	Overflow~reg0	Clock	Clock	4,500	-0.003	3.623
8	0.952	Zreg[6]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.582
9	0.990	Zreg[1]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.544
10	1.018	Breg[0]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.516
11	1.048	Areg[0]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.486
12	1.055	Breg[3]	Overflow~reg0	Clock	Clock	4,500	-0.003	3.478
13	1.093	Zreg[7]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.441
14	1.127	Breg[4]	Overflow~reg0	Clock	Clock	4,500	-0.003	3.406
15	1.133	Areg[1]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.401
16	1.135	Zreg[3]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.399
17	1.161	Breg[2]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.373
18	1.170	Areg[4]	Overflow~reg0	Clock	Clock	4,500	0.002	3.368
19	1.191	Areg[2]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.343
20	1.212	AddSubR	Zreg[15]	Clock	Clock	4,500	-0.002	3.322
21	1.223	Zreg[10]	Overflow~reg0	Clock	Clock	4,500	0.000	3.313
22	1.247	Areg[5]	Overflow~reg0	Clock	Clock	4,500	0.002	3.291
23	1.250	Zreg[13]	Overflow~reg0	Clock	Clock	4,500	0.000	3.286
24	1.262	Zreg[8]	Overflow~reg0	Clock	Clock	4,500	0.000	3.274
25	1.270	Breg[5]	Overflow~reg0	Clock	Clock	4,500	-0.003	3.263
26	1.277	Areg[6]	Overflow~reg0	Clock	Clock	4,500	0.002	3.261
27	1.277	SelR	Zreg[15]	Clock	Clock	4,500	-0.002	3.257
28	1.278	Areg[3]	Overflow~reg0	Clock	Clock	4,500	-0.002	3.256
29	1.283	AddSubR	Zreg[14]	Clock	Clock	4,500	-0.002	3.251
30	1.313	Breg[6]	Overflow~reg0	Clock	Clock	4,500	-0.003	3.220
31	1.348	SelR	Zreg[14]	Clock	Clock	4,500	-0.002	3.186
32	1.354	AddSubR	Zreg[13]	Clock	Clock	4,500	-0.002	3.180
33	1.369	Zreg[9]	Overflow~reg0	Clock	Clock	4,500	0.000	3.167
34	1.385	Areg[7]	Overflow~reg0	Clock	Clock	4,500	0.002	3.153
35	1.395	Breg[7]	Overflow~reg0	Clock	Clock	4,500	-0.003	3.138
36	1.396	Zreg[5]	Zreg[15]	Clock	Clock	4,500	-0.002	3.138
37	1.410	Zreg[0]	Zreg[15]	Clock	Clock	4,500	-0.002	3.124
38	1.419	SelR	Zreg[13]	Clock	Clock	4,500	-0.002	3.115
39	1.425	AddSubR	Zreg[12]	Clock	Clock	4,500	-0.002	3.109
40	1.437	Zreg[2]	Zreg[15]	Clock	Clock	4,500	-0.002	3.097
41	1.443	Zreg[11]	Overflow~reg0	Clock	Clock	4,500	0.000	3.093
42	1.467	Zreg[5]	Zreg[14]	Clock	Clock	4,500	-0.002	3.067
43	1.481	Zreg[0]	Zreg[14]	Clock	Clock	4,500	-0.002	3.053
44	1.490	SelR	Zreg[12]	Clock	Clock	4,500	-0.002	3.044


- In Quartus II software, there are features to display a schematic diagram of the designed circuit at the Register Transfer Level. The tool called **RTL Viewer**. Click on **Tools > Netlist Viewer > RTL Viewer**, to open the window shown as below.



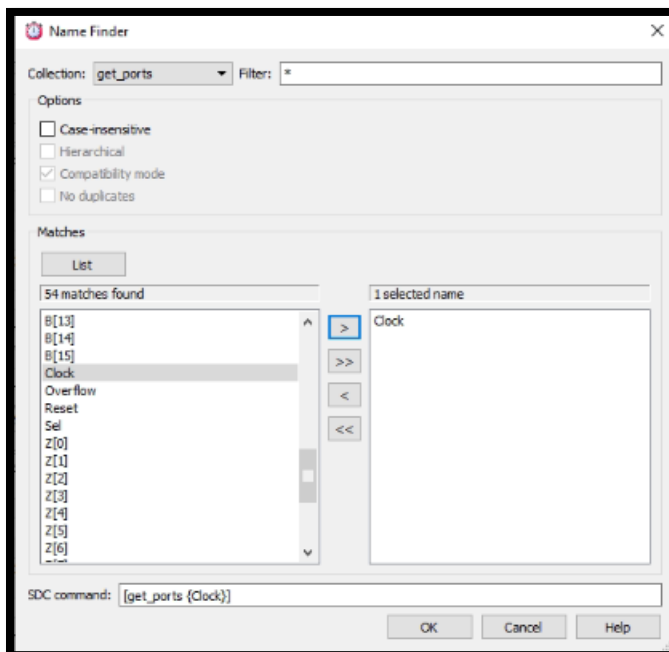
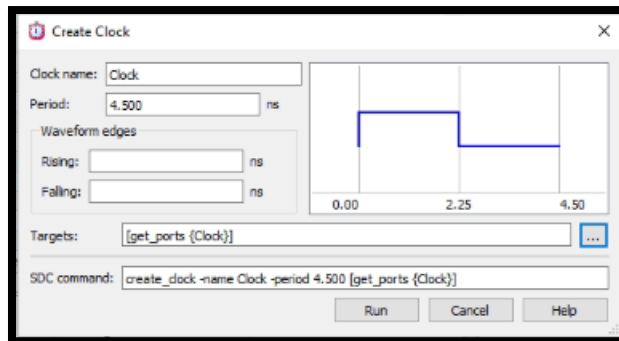
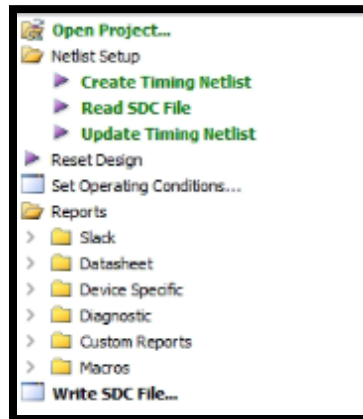


- The RTL Viewer can also be accessed through the **Project Navigator** window. To locate the mux2to1 subcircuit in the **Hierarchy** view, right-click on it and choose **Locate > Locate in RTL Viewer** from the pop-up menu. This presents the right-to-left (RTL) perspective of the subcircuit.

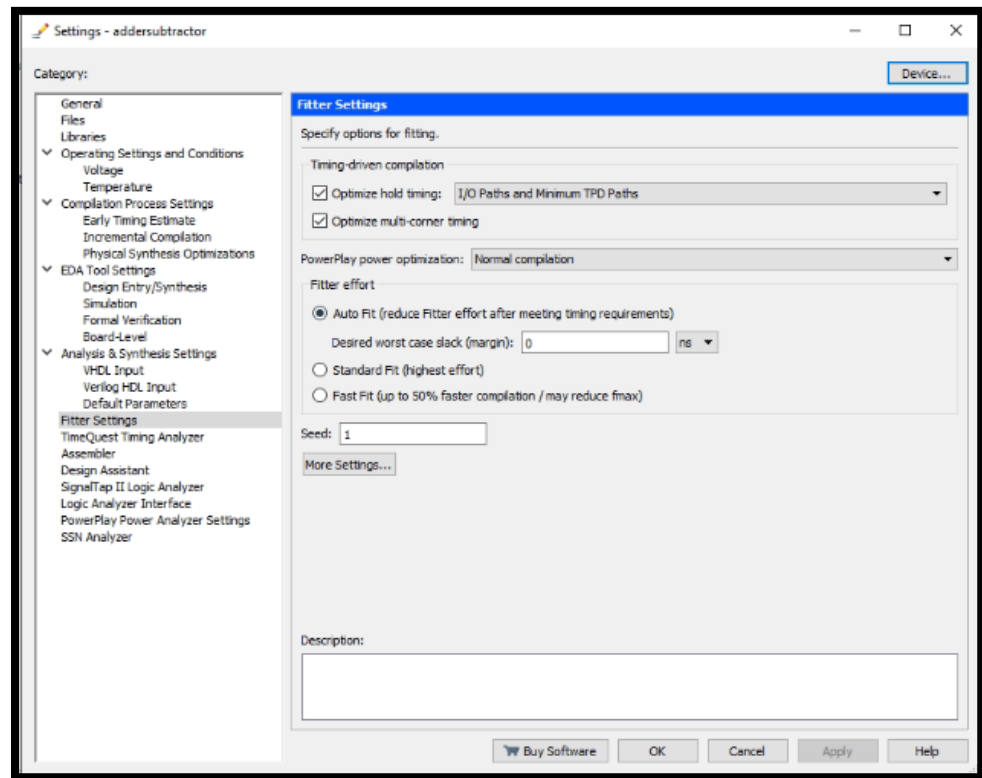


- To access the TimeQuest Timing Analyzer, you may either go to **Tools > TimeQuest Timing Analyzer** or click on the  toolbar icon. Assume that we require a circuit capable of functioning with a clock period of **4.5 ns**. The TimeQuest Timing Analyzer tool can be utilized to generate a new SDC (Synopsys Design Constraints) file that includes the clock constraint.
- In the TimeQuest Analyzer, double-click **Create Timing Netlist** under Tasks. Then go to **Constraints > Create Clock**. In the dialog box, name the clock **Clock**, set the **Period** to **4.5 ns**, and select the port **Clock** from the list of targets. Click **OK**.
- View the SDC command, which creates a 4.5 ns clock named **CLOCK** and links it to the port **Clock**. Click **Run. Select Constraints > write SDC file** and save the SDC file as *addersubtractor.out.sdc* in the project directory. Close the TimeQuest Analyzer.

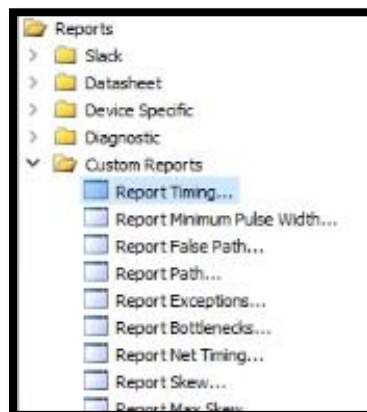
- In Quartus II, go to **Assignments > Settings**, select **TimeQuest Timing Analyzer**, and **add** the SDC file to the project by selecting *addersubtractor.out.sdc*. Click **OK**.



- In the Settings dialog, select **Fitter Settings**. Choose **Auto Fit** for a quick, adequate implementation. **Fast Fit** reduces compilation time but may lower fmax. **Standard Fit** aims for the best implementation, possibly taking longer. Click **OK** and recompile the circuit.



- The fmax of a circuit is limited by its longest delay path. To view this critical path, open the TimeQuest Timing Analyzer. Under Tasks, select **Reports > Custom Reports > Report Timing**. Choose **CLOCK** for both **From clock** and **To clock** fields. Set **Report number of paths** to 10.



Report Timing

Clocks

From clock:

Clock

To clock:

Clock

Targets

From:

...

Through:

...

To:

...

Analysis type

Setup

Hold

Recovery

Removal

Paths

Report number of paths:

10

Maximum number of paths per endpoint:

Maximum slack limit:

ns

Pairs only

Output

Detail level:

Full path

Set Default

Show routing

Report panel name:

Report Timing

File name:

...

File options

Overwrite

Append

Open

Console

Td command:

k } -to_clock { Clock } -setup -npaths 10 -detail full_path -panel_name {Report Timing}

Report Timing

Close

Help

- Click **Report Timing** to start the analysis and view the timing report.

Report Timing

Command Info

Summary of Paths

Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
0.524	AdtsAR	Overflow-req0	Clock	Clock	4,500	-0.002	4.030
0.589	SeR	Overflow-req0	Clock	Clock	4,500	-0.002	3.945
0.637	SeR	Overflow-req0	Clock	Clock	4,500	-0.002	3.857
0.664	AdtsAR	Overflow-req0	Clock	Clock	4,500	-0.002	3.870
0.668	SeR	Overflow-req0	Clock	Clock	4,500	-0.002	3.866
0.673	AdtsAR	Overflow-req0	Clock	Clock	4,500	-0.002	3.861
0.698	SeR	Overflow-req0	Clock	Clock	4,500	-0.002	3.838
0.708	2mg[0]	Overflow-req0	Clock	Clock	4,500	-0.002	3.826
0.712	SeR	Overflow-req0	Clock	Clock	4,500	-0.002	3.822
0.722	2mg[0]	Overflow-req0	Clock	Clock	4,500	-0.002	3.812

Path #1: Setup slack is 0.524

Path Summary

Statistics

Data Path

Waveform

Total	Intr	RP	Type	Fanout	Location	Element
0.000	0.000					launch edge time
2.675	2.675					data path
0.000	0.000					source latency
0.000	0.000					Clock
0.999	0.999	RR	CELL	1	IOC_X0_Y18_X2	Clockcombout
1.117	0.118	RR	IC	1	CLKCTRL_03	Clock-<ctrl>ind[0]
1.117	0.000	RR	CELL	51	CLKCTRL_03	Clock-<ctrl>ind[0]
2.138	1.021	RR	IC	1	LOP_X37_Y34_N13	AdtsARjck
2.675	0.537	RR	CELL	1	LOP_X37_Y34_N13	AdtsAR
6.685	4.010					data path
2.925	0.230	UTop	1		LOP_X37_Y34_N13	AdtsAR
2.925	0.000	RR	CELL	17	LOP_X37_Y34_N13	AdtsARjckout
1.465	0.560	RR	IC	1	CCOMB_X38_Y34_N16	set10del

Path #1: Setup slack is 0.524

Path Summary

Statistics

Data Path

Waveform

Total	Intr	RP	Type	Fanout	Location	Element
4.900	4.900					latch edge time
7.173	2.673					data path
4.900	0.000					source latency
4.900	0.000					Clock
5.499	0.999	RR	CELL	1	IOC_X0_Y18_X2	Clockcombout
5.617	0.118	RR	IC	1	CLKCTRL_03	Clock-<ctrl>ind[0]
5.617	0.000	RR	CELL	51	CLKCTRL_03	Clock-<ctrl>ind[0]
6.636	1.019	RR	IC	1	LOP_X37_Y33_N19	Overflow-reqjck
7.173	0.537	RR	CELL	1	LOP_X37_Y33_N19	Overflow-req0
7.209	0.036	UTop	1		LOP_X37_Y33_N19	Overflow-req0

Waveform

Launch Clock

Setup Relationship

Latch Clock

Data Arrival

Clock Delay

Slack

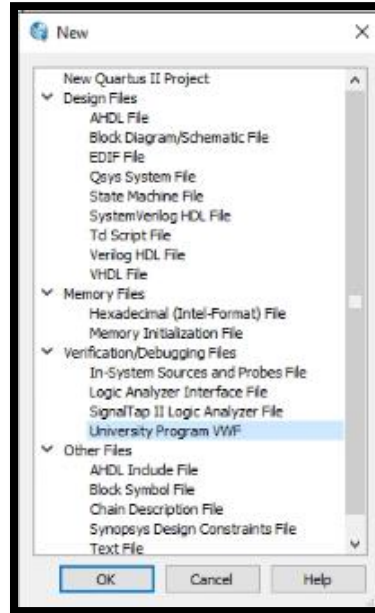
Data Required

Clock Delay

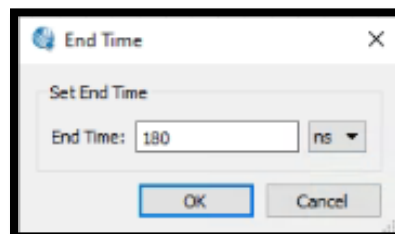
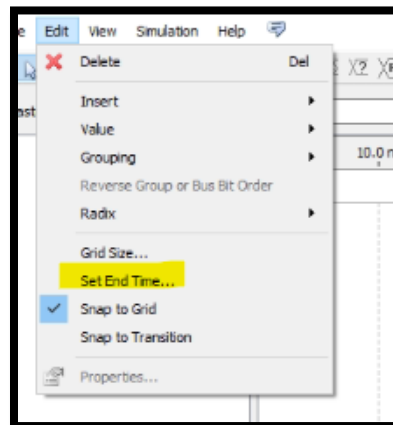
Time (ns)

3.2 Timing Diagram Setup

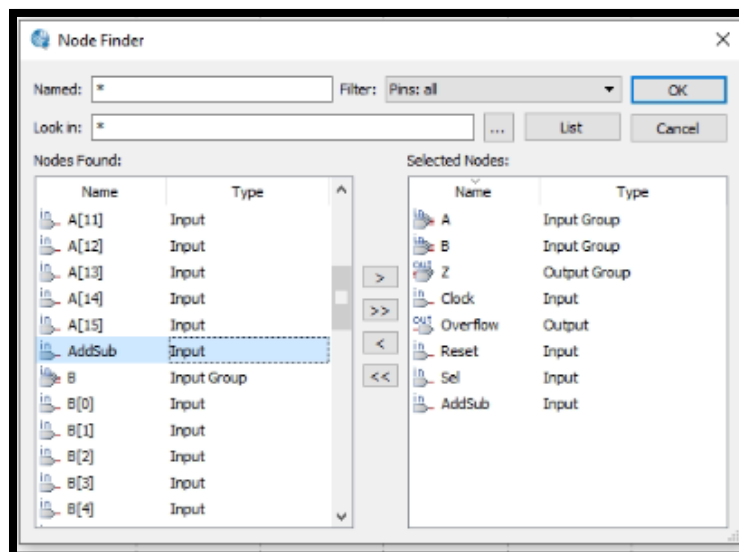
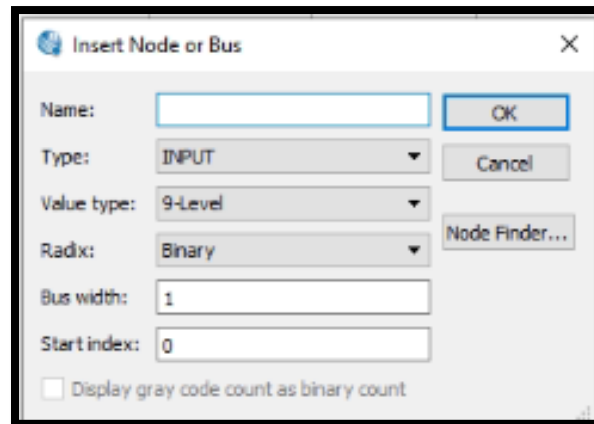
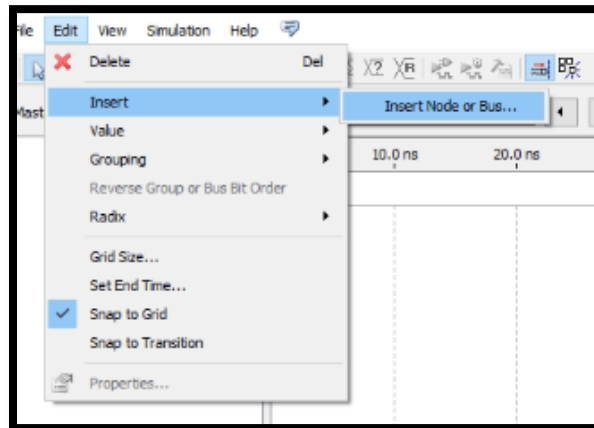
- Open the Waveform Editor window by selecting **File > New**, which gives the window shown in figure below. Choose **University Program VWF** and click **OK**.



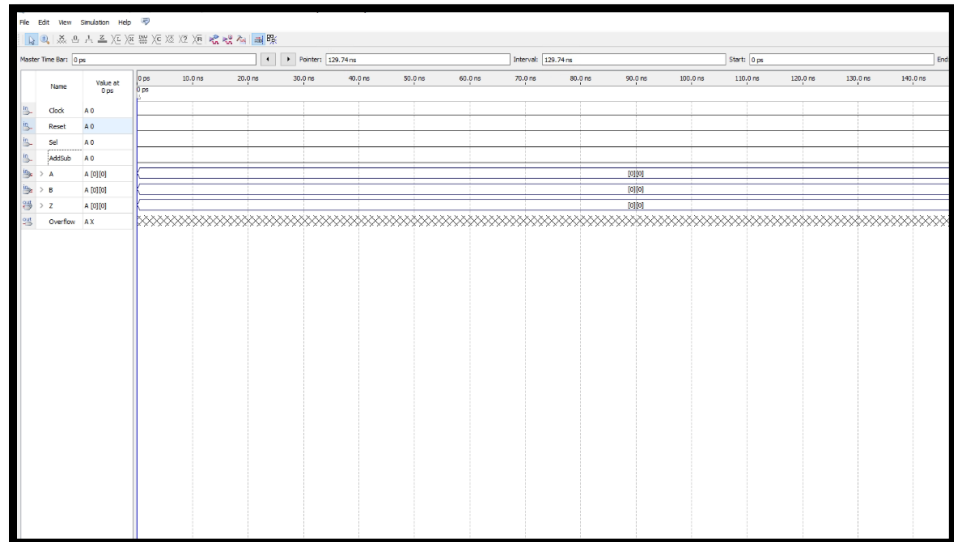
- In the Waveform Editor, save the file as addersubtractor.vwf. Set the simulation to run to 180 ns via **Edit > End Time**. Select **View > Fit in Window** to display the entire range.



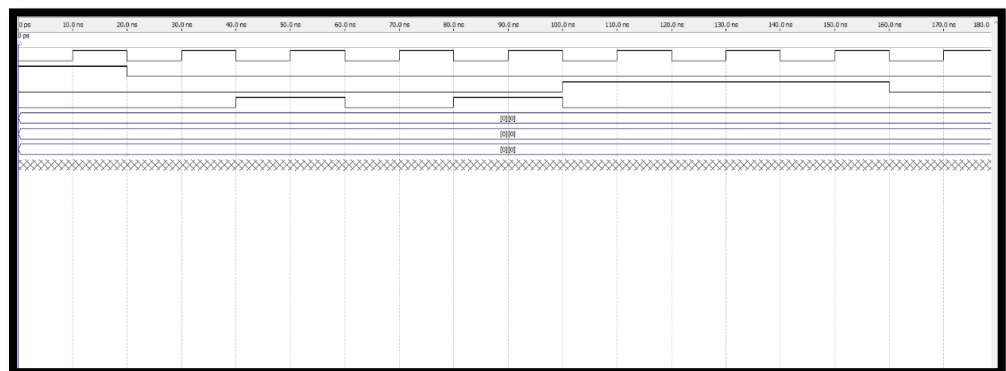
- To include input and output nodes for simulation, go to **Edit > Insert > Insert Node or Bus**. Click **Node Finder**, set the filter to **Pins: all**, and click **List**. Select the input and output signals **A**, **B**, and **Z** as 16-bit vectors for convenience.



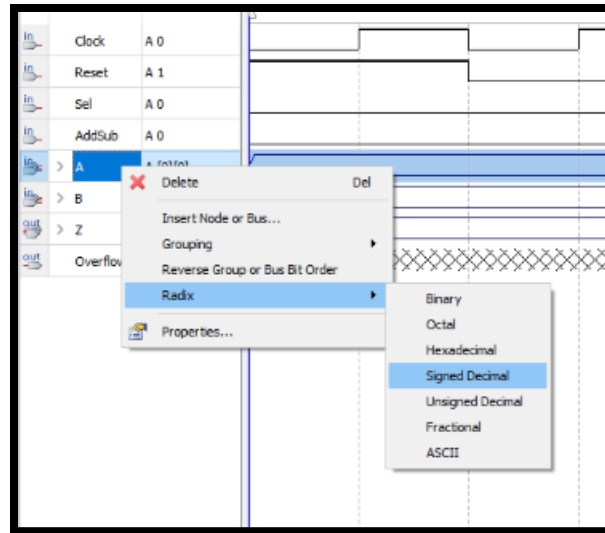
- Use the scroll bar in the Nodes Found box to find the Clock signal. Select it and click the > sign to add it to the **Selected Nodes** box. Repeat for **Reset**, **Sel**, **AddSub**, vectors **A**, **B**, **Z**, and the output **Overflow**. Click **OK** to close **Node Finder**, and **OK** again in the **Insert Node or Bus** window. Rearrange nodes in the **Waveform Editor** by dragging them up or down in the Name column.



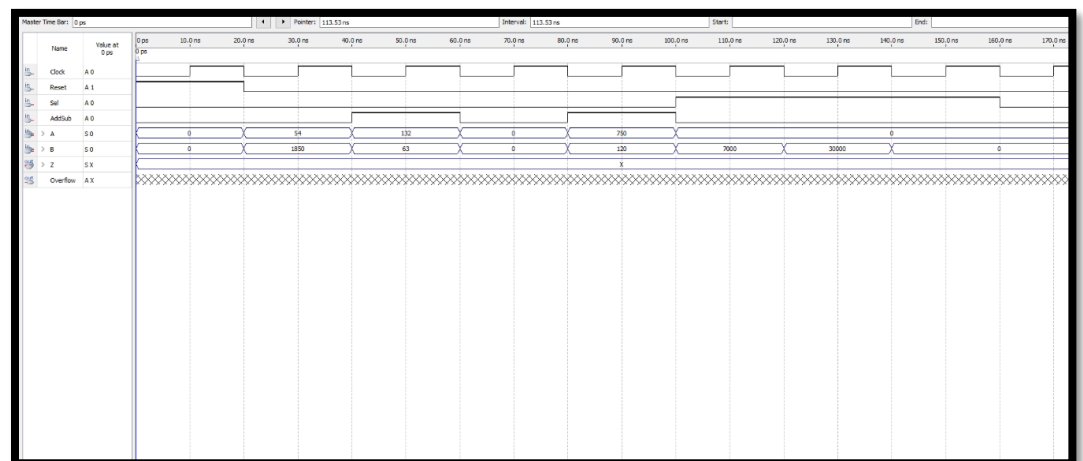
- For simplicity, assume input signals change at the clock's negative edges. To reset the circuit, set **Reset** = 1 from 0 to 20 ns by dragging the mouse over this interval and selecting logic value 1 from the toolbar. Set **Sel** = 1 from 100 to 160 ns, and **AddSub** = 1 from 40 to 60 ns and 80 to 100 ns. This should match the image in figure below.



- Vectors **A**, **B**, and **Z** are initially ASCII codes but should be treated as **signed decimal** numbers. Right-click on A, select **Properties**, set the **radix** to **signed decimal**, ensure the bus width is 16 bits, and click **OK**. Repeat for vectors B and Z.



- Click the **Arbitrary Value** icon in the toolbar to open the pop-up window. Set the start time to **20 ns** and the **end time to 40 ns**, enter **54** under **Numeric or named value**, and click **OK**. For subsequent 20-ns intervals, set A to 132, 0, 750, and then 0. Set B to 1850, 63, 0, 120, 7000, 30000, and 0. This generates the waveforms shown in figure below, with Z and Overflow displaying unknown values. **Save** the file.



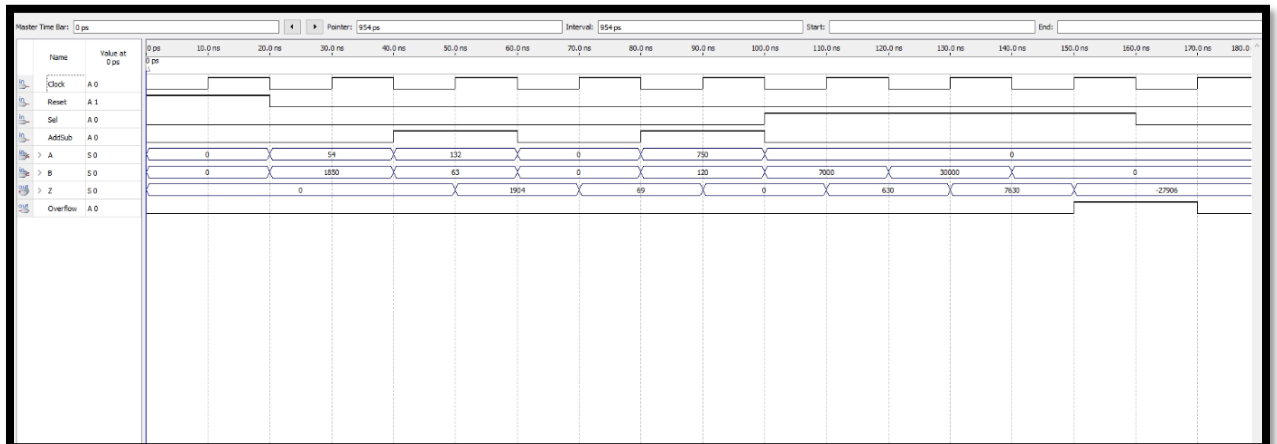
- Start the simulation by selecting **Processing > Start Simulation** or using the icon. After completion, Quartus II will display a Simulation Report, as shown in figure in *Result* part.

4. Result

The result obtain in the figure below is the timing report setup displays critical information about the designed circuit's timing analysis. This includes details on the 10 paths with the longest delays, highlighting the critical path that determines the maximum delay within the circuit.

Specifically, it shows the slack for each path, indicating the amount of delay that can be added to a path without violating the specified timing constraint. A positive slack value implies that the path meets timing requirements, while a negative slack value suggests a potential timing violation that needs attention.

Furthermore, this presents the actual elements involved in the selected critical path and the incremental delay contributed by each stage along the path. This breakdown helps to pinpoint areas where optimization may be necessary to enhance overall circuit performance and ensure compliance with timing constraints.



5. Analysis

The Verilog code is to implement a parameterized 16-bit adder-subtractor module (addersubtractor) that can perform both addition and subtraction of two 16-bit binary numbers. This module also incorporates overflow detection and uses two submodules: a 2-to-1 multiplexer (mux2to1) and a k-bit adder (adderk).

- The top-level module (addersubtractor) is parameterized with $n = 16$, allowing flexibility for different bit-widths if modified.
- The module takes two 16-bit inputs (A and B) and performs either addition or subtraction based on the control signal AddSub. When AddSub is 0, the module performs addition; when AddSub is 1, it performs subtraction.
- Subtraction is achieved by adding the 2's complement of B to A. This is done by XORing B with AddSubR and adding AddSubR as the carry-in.
- A 2-to-1 multiplexer (mux2to1) is used to select between Areg and the previous output Z based on the Sel signal.
- The combinational logic calculates the intermediate results (G, H, M) required for the arithmetic operations.
- Overflow is detected using the formula $\text{carryout} \wedge G[n-1] \wedge H[n-1] \wedge M[n-1]$.
- Sequential logic updates the registers (Areg, Breg, Zreg, SelR, AddSubR, Overflow) on the rising edge of the clock or when a reset signal is received.
- On reset, all registers are cleared to 0.
- The module includes logic to detect arithmetic overflow and outputs an Overflow signal to indicate if overflow occurred during the operation.

Submodules

1) mux2to1

A k-bit 2-to-1 multiplexer that selects between two inputs (V, W) based on the select signal (Sel).

2) adderk

A k-bit adder that adds two k-bit inputs (X, Y) along with a carry-in (carryin) and produces a k-bit sum (S) and a carry-out (carryout).

Usage

1) Inputs

A, B: 16-bit input operands for the arithmetic operation.

2) Clock, Reset

Control signals for synchronous operation and reset functionality.

3) Sel

Select signal for the multiplexer.

4) AddSub

Control signal to choose between addition (0) and subtraction (1).

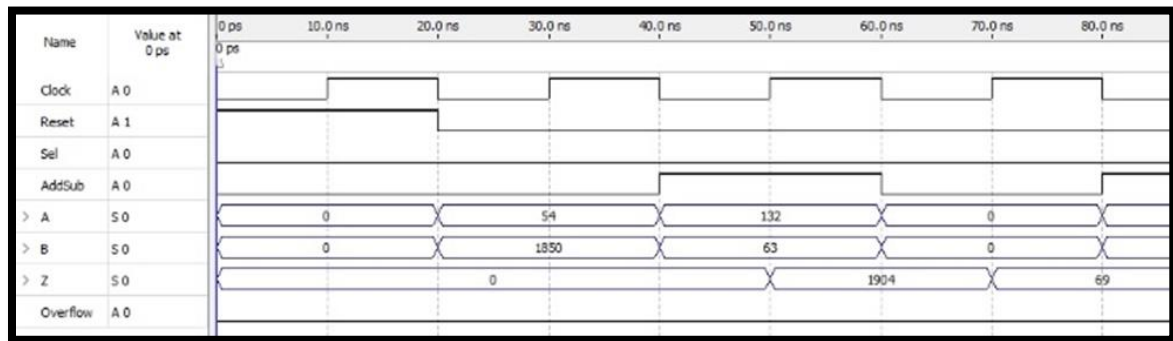
Outputs

1) Z

16-bit result of the arithmetic operation.

2) Overflow

Signal indicating if an overflow occurred.



The Verilog code defines a versatile arithmetic module capable of performing 16-bit addition and subtraction with overflow detection. It leverages modular design principles by using separate multiplexer and adder submodules. This design can be efficiently implemented in FPGA or ASIC environments, providing a flexible and robust solution for arithmetic operations in digital systems.

6. Conclusion

The Verilog code for the parameterized 16-bit adder-subtractor module presents an efficient and flexible design for performing addition and subtraction of 16-bit binary numbers, with built-in overflow detection for robust arithmetic operations. Key features include its parameterized top-level module, control signal for switching operations, 2's complement subtraction mechanism, modular design with multiplexer and adder submodules, reliable overflow detection, and a combination of combinational and sequential logic for synchronous operation. The module's versatile outputs and suitability for FPGA or ASIC environments, alongside the comprehensive design and simulation flow facilitated by Quartus II software, provide a solid foundation for practical digital system design and implementation using Verilog.