MILESTONE 1

Quartus Familiarization

Objectives

- Familiarization with Altera Quartus for entering, compiling and simulating a logic design.
- Describing a complex digital system using hierarchy, modularity and regularity
- Creation of several low-level reusable subsystems to build a module library

In this milestone, you will build a ripple carry adder, a multiplexer and a register.

1.1 4-bit Ripple Carry Adder

1.1.1 Half-Adder

Step 1

Use Quartus schematic entry to input the half adder design.

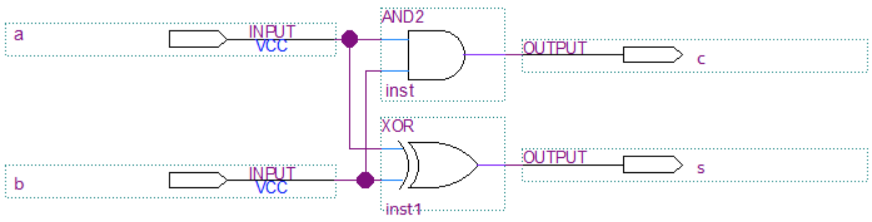


Figure 1.1: Half-adder schematic.

Step 2

Compile the design. If there are no errors, print compilation report as PDF.

Step 3

Simulate by giving all 4 input combinations for a and b : 00, 01, 10 and 11. Maintain each input combination for 100ns. That means the End Time for the simulation should be 400ns. Check the result with the expected output. When the circuit is error-free, proceed to Step 4.

Step 4

Convert the module into symbol file. Call it ha.bsf.

Step 5

Print the following 3 pages as PDF. You must choose landscape orientation.
Page 1: Schematic of half adder: Printscreen then convert to PDF.
Page 2: Compilation report

Page 3: Annotated simulation output waveform: This is not the same as the simulation input. Printscreen first, then in MSPaint or similar software, highlight the important information. Save as PDF.

1.1.2 Full Adder

Step 1

Use Quartus schematic entry to input two instances of half adder symbols from Part 1 Step 4. Combine the two half adders with an or2 gate to build a full adder.

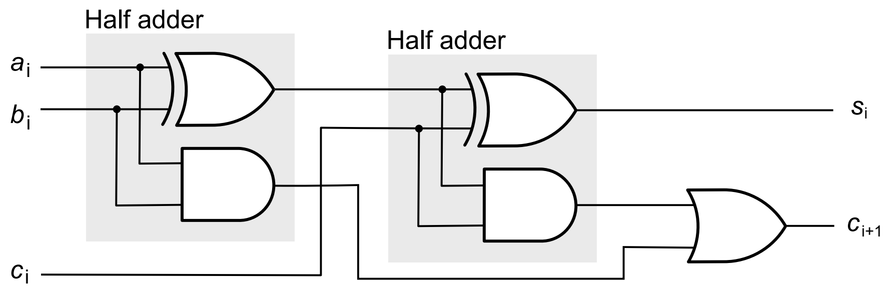


Figure 1.2: Full adder schematic.

Step 2

Compile the design. If there are no errors, print compilation report as PDF.

Step 3

Simulate using all 8 input combinations 000 through 111. Maintain each input combination for 100ns. That means the End Time for the simulation should be 800ns. Check the result with the expected output. When the circuit is error-free, proceed to Step 4.

Step 4

Convert the module into symbol file. Call it fa.bsf.

Step 5

Print the following 3 pages as PDF. You must choose landscape orientation.
Page 4: Schematic of full adder..
Page 5: Compilation report
Page 6: Annotated simulation output waveform..

1.1.3 Ripple Carry Adder

Step 1

Use Quartus schematic entry to input four instances of full adder symbols from Part 2 Step 4. Combine to build the ripple carry as follows.

Step 2

Compile the design. If there are no errors, print compilation report as PDF.

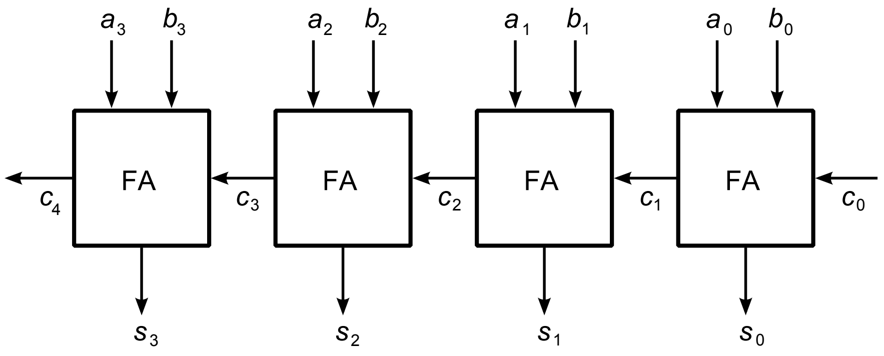


Figure 1.3: Full adder schematic.

Step 3

In Table 1.1, fill in columns $COUT$ and $S[3:0]$ so you know what is the expected output for each input combination. Columns A, B and S are in hex.

Table 1.1: Test data for ripple carry adder.

A[3..0]	B[3..0]	CIN	COUT	S[3..0]	Comply?
0	0	0			
F	F	1			
F	0	0			
0	F	0			
F	0	1			
0	F	1			
8	8	0			
A	5	0			
C	3	0			
3	C	1			

In the simulation waveform input, use bus groups for A, B and S signal. (A bunch of related signals are called a bus.) Use hexadecimal system. Simulate using the given test data. Then mark with the adder complies with the expected results. Simulate using the 10 input combinations given below. Maintain each input combination for 100ns. That means the End Time for the simulation should be 1000ns. When the circuit is error-free, proceed to Step 4.

Step 4

Print the following 4 pages as PDF. You must choose landscape orientation.
Page 7: Schematic of ripple carry adder..
Page 8: Compilation report
Page 9: Annotated simulation output waveform..
Page 10: References. On this page, list all web sites, articles or books you referred in completing the assignment. You must any citation format but be consistent.

Step 5

Combine all 10 PDF pages into 1 PDF document. Add the plagiarism declaration as the front cover. Upload to elearning.utm.my.

1.2 Bus Multiplexer

A bus multiplexer selects data from one out two buses.

1.2.1 2:1 Mux

1. Enter the 2:1 mux design.

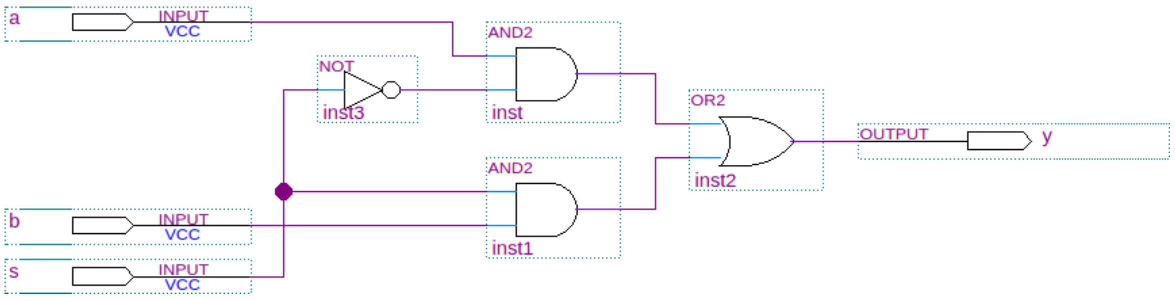


Figure 1.4: 2:1 mux schematic.

2. Compile the design.
3. Simulate by giving all 4 input combinations for *a*, *b* and *s*. Maintain each input combination for 100ns.
4. Convert the module into symbol file. Call it **mux21.bsf**.

1.2.2 4-bit Bus Multiplexer

1. Instantiate four copies of **mux21** symbols from the previous step.

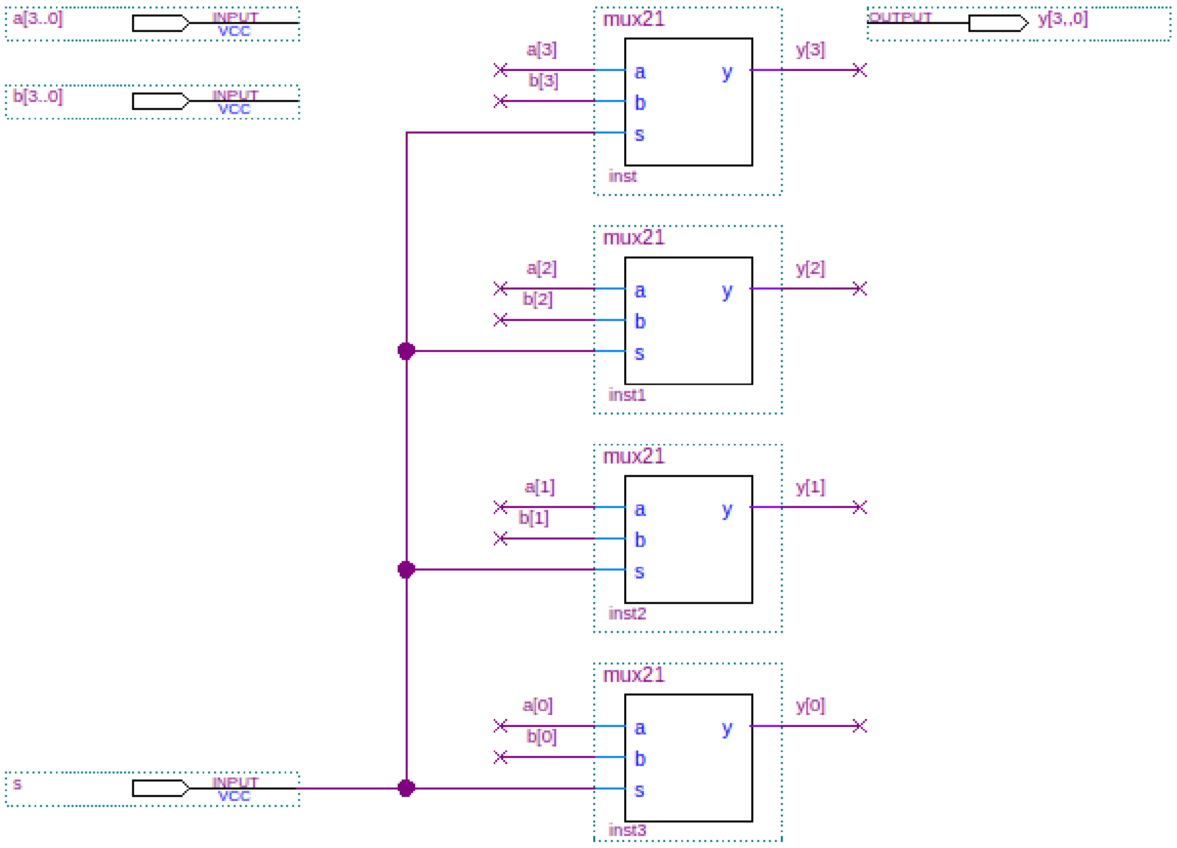


Figure 1.5: Bus multiplexer schematic.

2. Compile the design.
3. Simulate using the following test data.

Table 1.2: Test data for 4-bit bus multiplexer.

A[3..0]	B[3..0]	S	Y[3..0]	Comply?
F	0	0		
0	F	0		
A	5	1		
5	A	1		
8	8	0		
A	A	1		

4. Convert the module into symbol file. Call it **busmux4bit.bsf**. It should like Fig. 1.6.

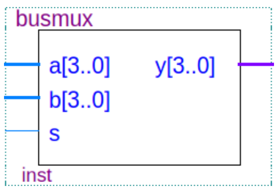


Figure 1.6: Bus multiplexer symbol.

1.3 Data Register

A data register contains a set of flip-flops that can be loaded on demand.

1.3.1 Register Cell

1. Enter the register cell design as shown Fig. 1.7. Get the DFF symbol from the Quartus library and mux21 symbol from the previous step of this assignment.

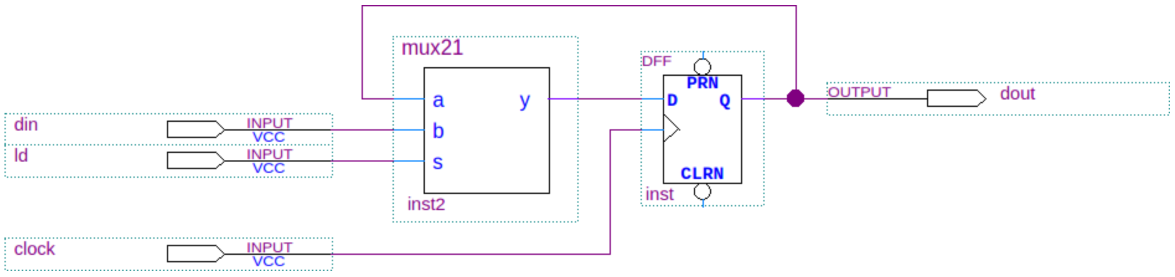
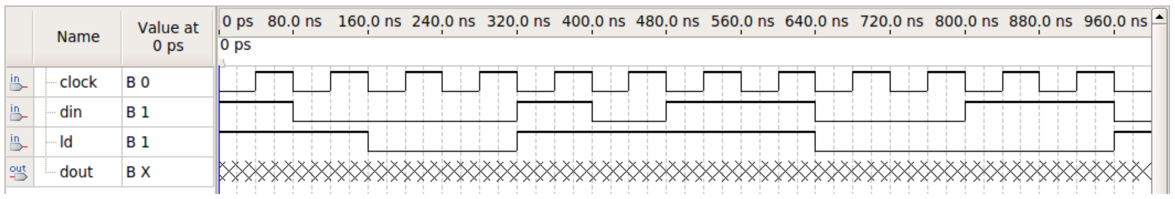


Figure 1.7: Register cell schematic.

2. Compile the design.
3. Simulate using the following input waveform:



Name	Value at 0 ps
clock	B 0
din	B 1
ld	B 1
dout	B X

Figure 1.8: Register cell simulation input.

Verify the function of the register cell.

4. Convert the module into symbol file. Call it **regcell.bsf**.

1.3.2 4-bit Data Register

1. Instantiate four copies of **regcell** symbols and connect as shown in Fig. 1.9.

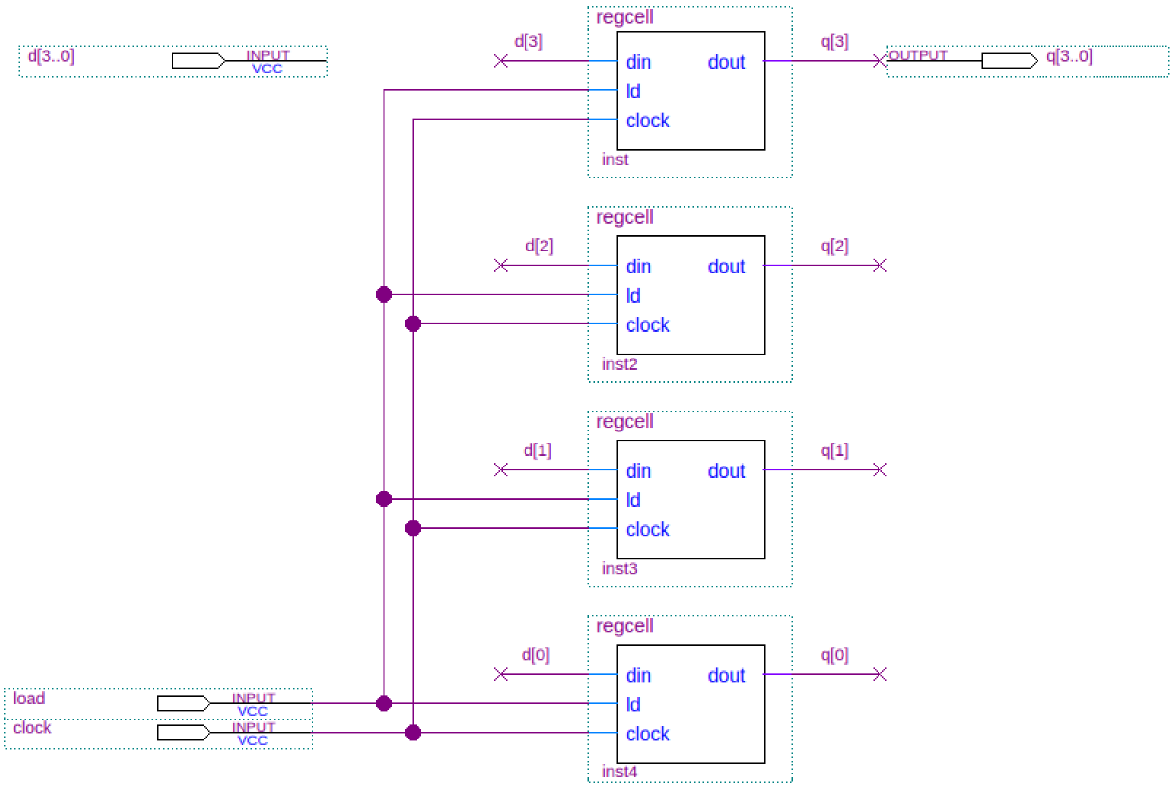


Figure 1.9: 4-bit data register schematic.

2. Compile the design.
3. Simulate using the following input waveform:

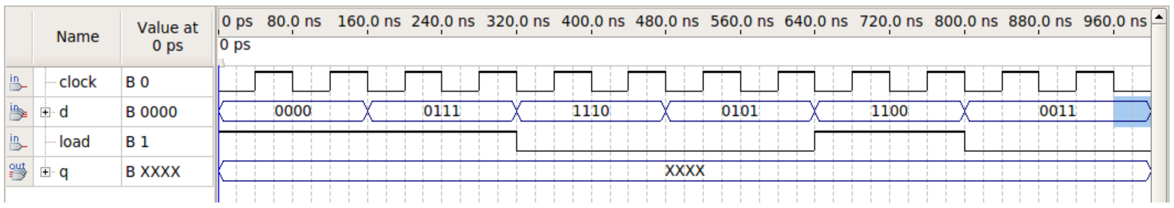


Figure 1.10: 4-bit data register simulation input.

4. Convert the module into symbol file. Call it **reg4bit.bsf**. It should like Fig. 1.11.

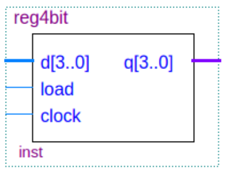


Figure 1.11: 4-bit data register symbol.