

Altera Quartus II Tutorial

Part II

(For ECE 465 Students at UIC)

Sajjad Rahaman

TA for ECE 465, Spring 2009

Department of Electrical and Computer Engineering

University of Illinois at Chicago

mrahaman@ece.uic.edu

The first part of Quartus[®] II tutorial illustrates schematic diagram based entry for the desired circuit. The second part of Quartus[®] II tutorial is aimed at introducing hierarchical schematic based design entry method. In hierarchical based design approach, commonly used logic elements (i.e. 1-bit full adder) are compiled and synthesized separately and later, these smaller logic elements are integrated in a larger module (i.e. 4-bit ripple carry adder). As a result, top-level design file becomes more readable. In this tutorial, an 1-bit full adder (FA) will be compiled and synthesized using schematic based design entry and then it will be used to create an 4-bit ripple carry adder.

This tutorial will also introduce two types of simulation, namely, functional simulation and timing simulation, to assess the behavior and performance of the desired design.

Please note that this tutorial is based on Altera Quartus[®] II 8.1 web edition version. Altera just released Quartus[®] II 9 web edition version on March 09, 2009. It is available at

https://www.altera.com/support/software/download/altera_design/quartus_we/dnl-quartus_we.jsp?WT.mc_id=qe_ie_al_cp_ht_1_111&contactID=104314847

Content

- 1. Hierarchical Based Design**
- 2. Functional Simulation**
- 3. Timing Simulation**

1. Hierarchical Based Design

In this section, a new Quartus® II project will be created to design an 4-bit ripple carry adder, the top level module. Please follow the steps shown in part I of this tutorial to open a new project using **New Project Wizard** in Quartus® II.

We will start by designing an 1-bit FA using schematic editor. Figure 1 illustrates an 1-bit full adder.

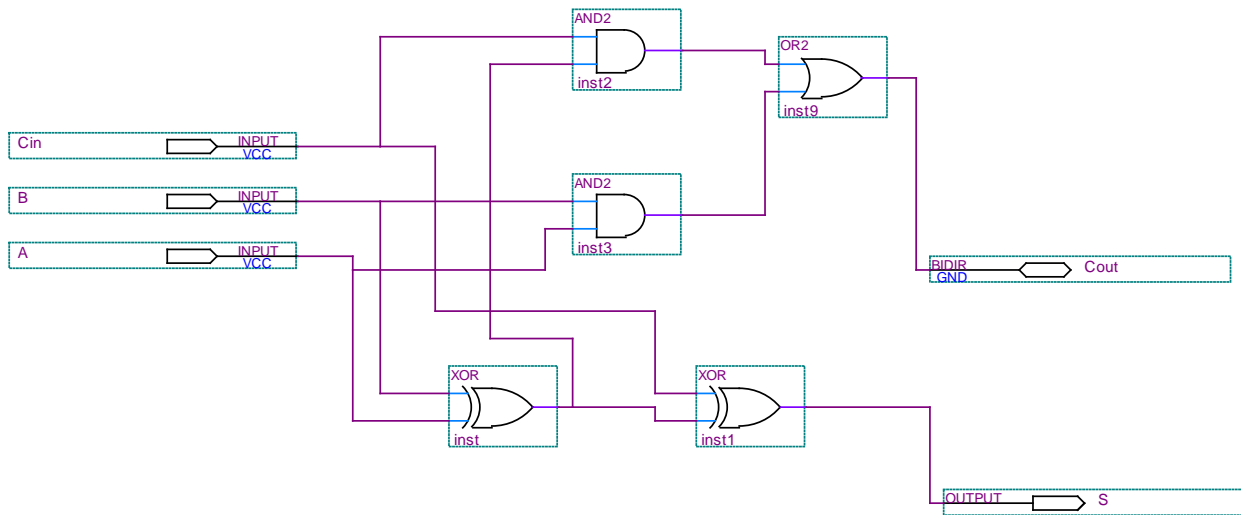


Figure 1. 1-bit full-adder (FA) design

After saving this file as **FA_1bit** we will run the following command for checking any design error.

Processing > Analyzing Current File.

If the design is free of errors, then we will create a symbol for this 1-bit FA block for later use in an 4-bit ripple carry adder. This is done so by selecting **File > Create/Update > Create Symbol File for Current File**. Figure 2 shows the screen capture of this symbol creation command.

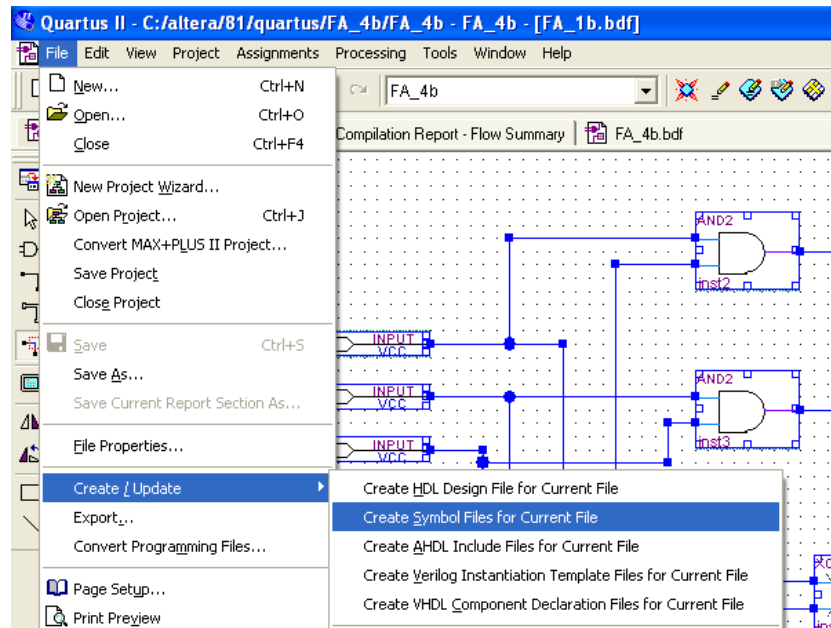


Figure 2. Symbol Creation Command

At this point we are ready to design our top level module, an 4-bit ripple carry adder (RCA), by cascading 4(four) 1-bit FAs. We need to import 1-bit FA symbol in the new schematic file. Figure 3 shows the screen capture for this step.

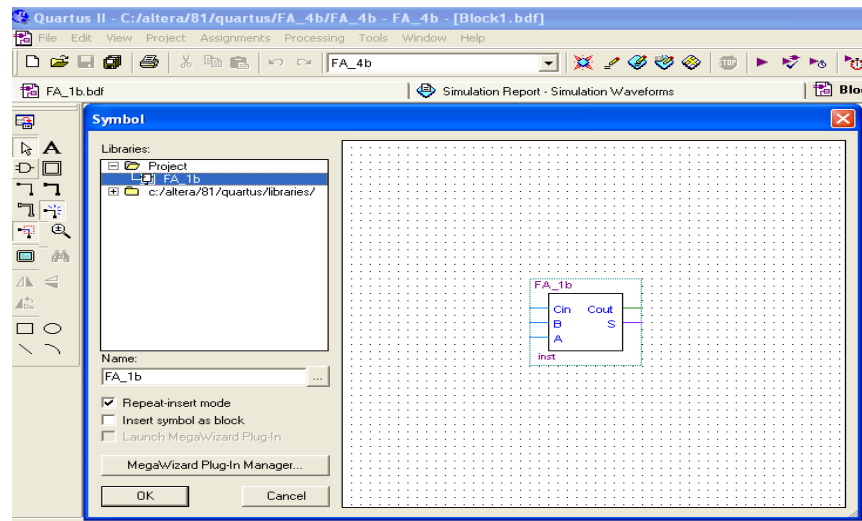


Figure 3. Importing 1-bit FA symbol in 4-bit RCA

Figure 4 shows schematic diagram for an 4-bit RCA consisting of cascaded 1-bit FAs. This top-level design file name must be the same as Quartus® II project name.

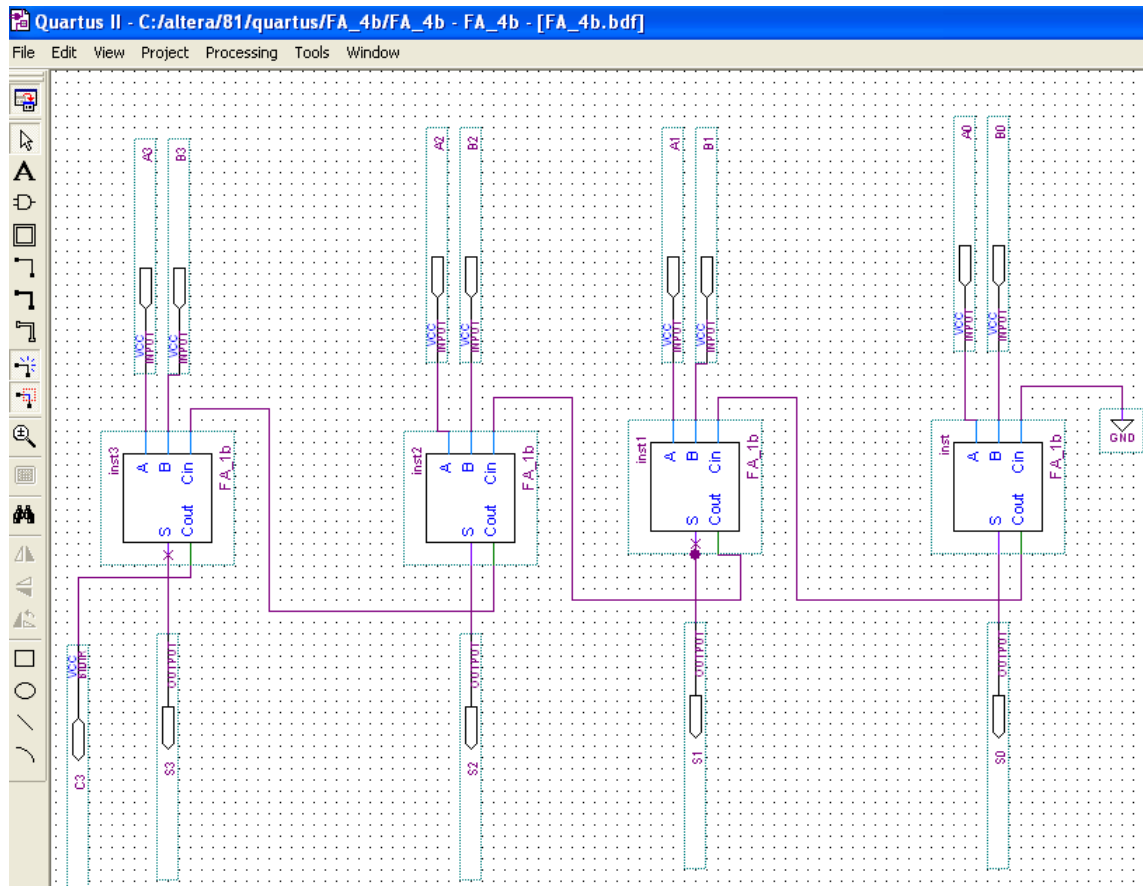


Figure 4. Schematic diagram for 4-bit RCA

After saving the above schematic file, we will again run the following command for checking any design error.

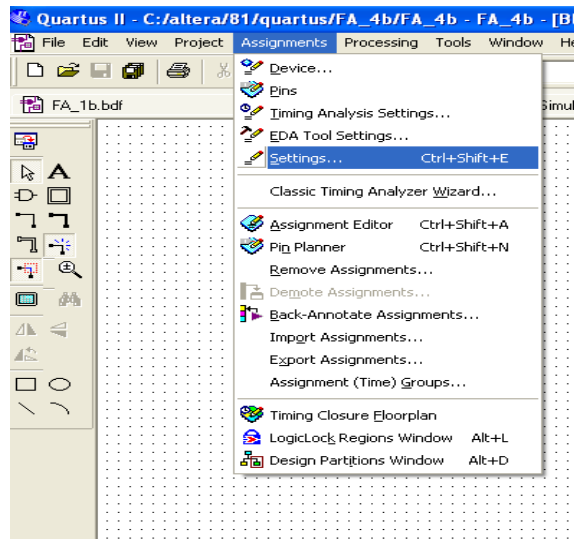
Processing > Analyzing Current File.

Having completed our design without any errors, we are now ready to compile and simulate to see the functional correctness and timing analysis of the synthesized circuit. We need to create a vector waveform file (*.vwf) to simulate the design. Please follow the steps shown in part I of this tutorial for creating such file.

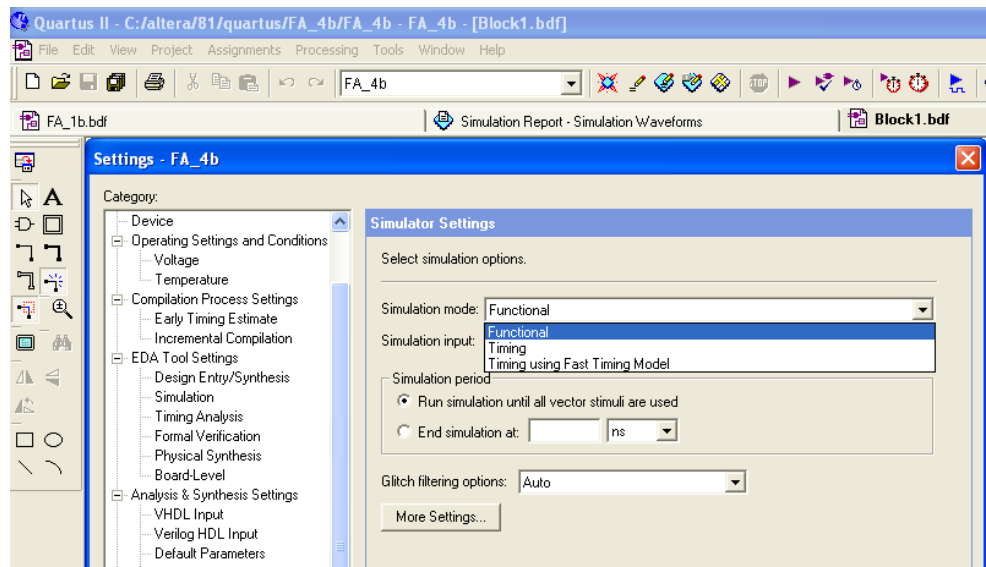
2. Functional Simulation

Functional simulation verifies the correctness of logic operation of the synthesized circuits. It does not take timing issue into consideration.

Quartus II carries out timing simulation by default. So, the setting needs to be changed to run functional simulation. This is done by selecting **Assignment>Setting** and then selecting functional simulation mode. Figure 5 illustrates the changes necessary in default setting to run function simulation.



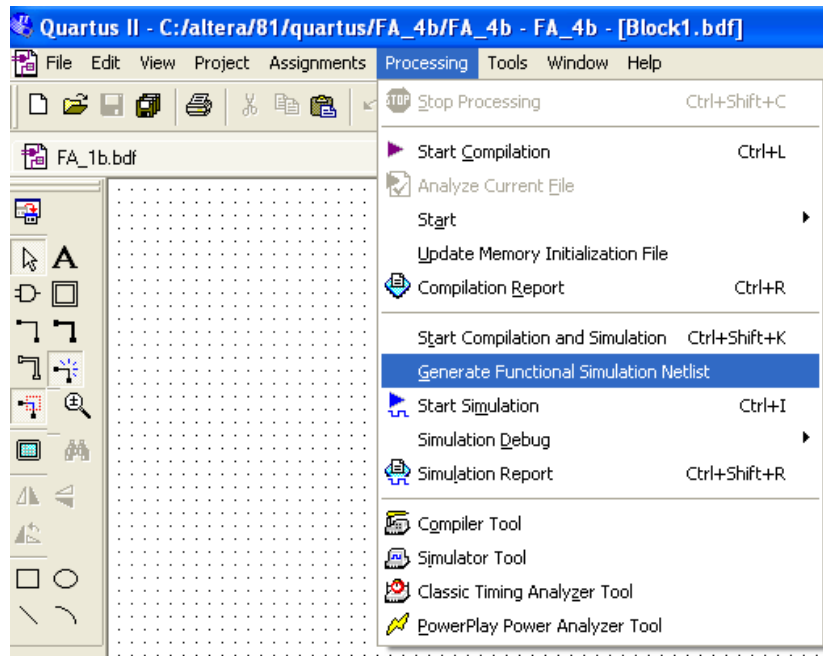
(a)



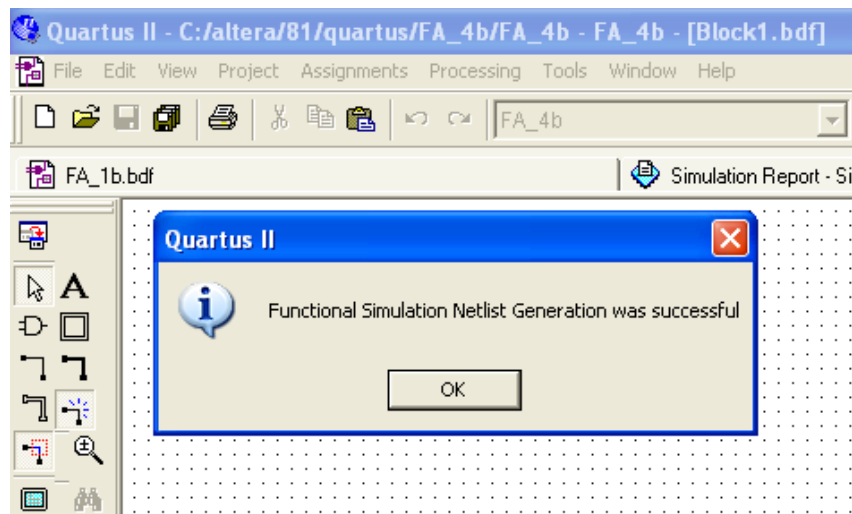
(b)

Figure 5. Setting for functional simulation

Before running functional simulation, a functional simulation netlist for the synthesized circuit needs to be generated. This is done by selecting **Processing > Generate functional Simulation Netlist**. Figure 6 shows the screen capture of the command and output.



(a)



(b)

Figure 6. Functional simulation netlist generation for 4-bit RCA

At this point, we are ready to run functional simulation. Simulation will use a vector waveform file created earlier. We will select **Process > Start Simulation** to begin the simulation. Successful simulation will generate simulation waveform (Figure 7)

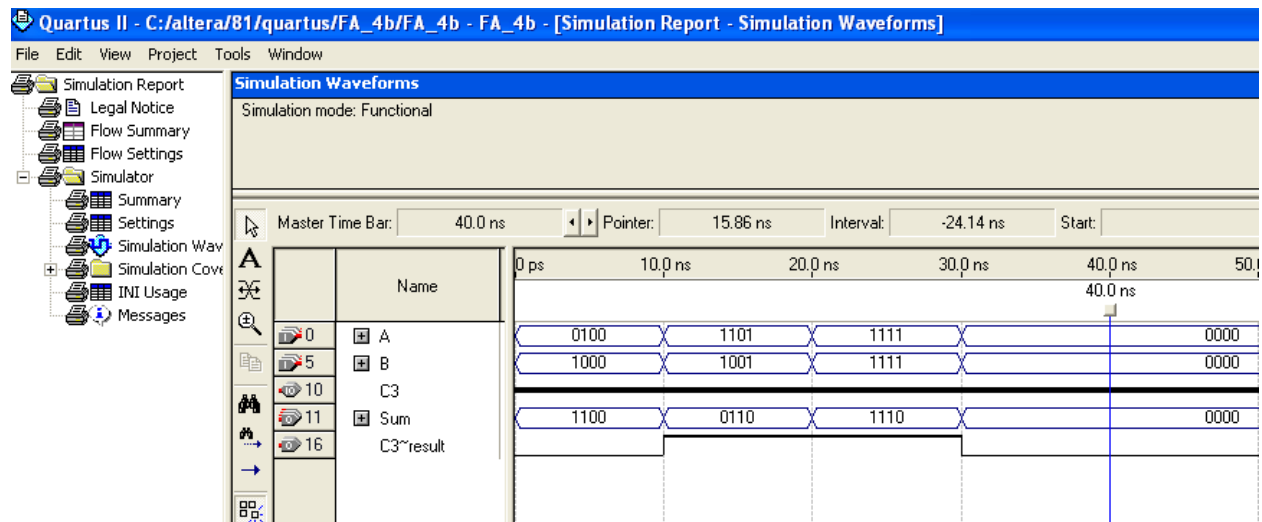


Figure 7. Simulation output

As shown in the above figure, our design file correctly generates the output variables. It is also seen that there is no time delay between input and output switching.

3. Timing Simulation

Having verified the logical correctness of the design file, it is necessary to determine whether the synthesized circuit meets the delay constraint. Timing simulation verifies both the logical correctness and timing. We will be able to find the propagation delay along various paths in the synthesized circuits.

We have changed the default setting to run the functional simulation. So, we have to change it back to Timing simulation mode to run timing simulation. Timing simulation is carried out by selecting **Processing > Start Compilation and Simulation**. After successful compilation and simulation, simulation waveform will be displayed. Figure 8 shows the simulation waveform from timing analysis.

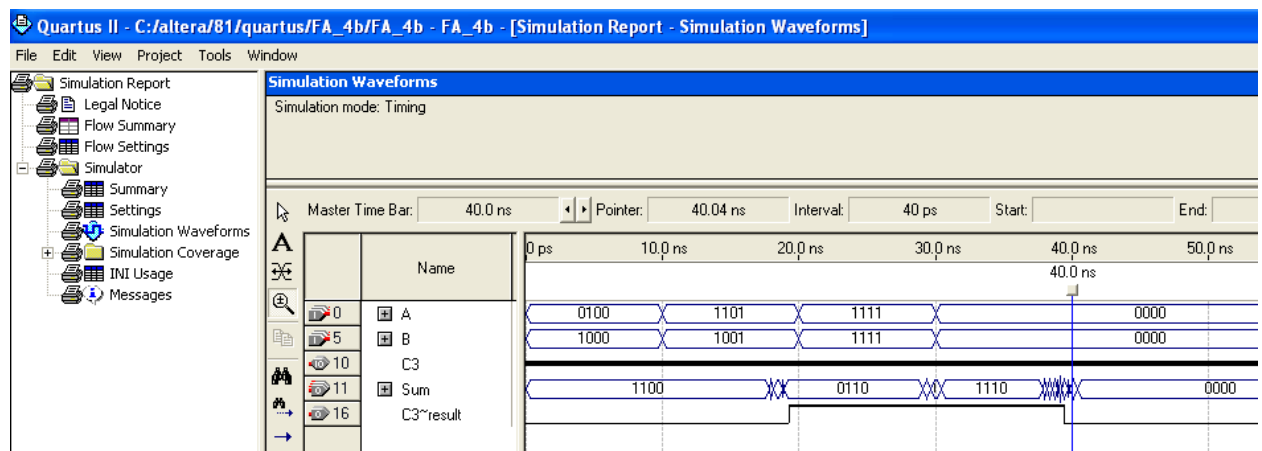


Figure 8. Timing simulation output

It is seen in Figure 8 that there are propagation delays between inputs and outputs. Timing Analyzer tool will provide propagation delays along all the paths and the worst case propagation delay. Figure 9 illustrates the timing parameter of our design file. Please note that this delay depends on the device chosen for the simulation. This tutorial uses CYCLONE for simulation purpose.

tpd						
	Slack	Required P2P Time	Actual P2P Time	From	To	
3	N/A	None	10.158 ns	A1	C3	
4	N/A	None	10.135 ns	B1	C3	
5	N/A	None	10.107 ns	B3	C3	
6	N/A	None	10.014 ns	A0	S2	
7	N/A	None	9.816 ns	A1	S3	
8	N/A	None	9.793 ns	B1	S3	
9	N/A	None	9.761 ns	B3	S3	
10	N/A	None	9.743 ns	A0	C3	
11	N/A	None	9.677 ns	B0	S2	
12	N/A	None	9.654 ns	B2	S2	
13	N/A	None	9.406 ns	B0	C3	
14	N/A	None	9.401 ns	A0	S3	
15	N/A	None	9.384 ns	B2	C3	
16	N/A	None	9.300 ns	A3	C3	
17	N/A	None	9.271 ns	A1	S1	
18	N/A	None	9.249 ns	B1	S1	
19	N/A	None	9.202 ns	A2	S2	
20	N/A	None	9.064 ns	B0	S3	
21	N/A	None	9.042 ns	B2	S3	
22	N/A	None	8.954 ns	A3	S3	
23	N/A	None	8.931 ns	A2	C3	
24	N/A	None	8.853 ns	A0	S1	
25	N/A	None	8.589 ns	A2	S3	
26	N/A	None	8.519 ns	B0	S1	
27	N/A	None	8.168 ns	A0	S0	
28	N/A	None	7.831 ns	B0	S0	

(a)

Timing Analyzer Summary									
	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	10.429 ns	A1	S2	--	--	0
2	Total number of failed paths								0

(b)

Figure 9. Timing analyzer output