Timing Considerations with Verilog-Based Designs

This tutorial describes how Altera's Quartus® II software deals with the timing issues in designs based on the Verilog hardware description language. It discusses the various timing parameters and explains how specific timing constraints may be set by the user.

Contents:

Quartus II software includes a Timing Analyzer module which performs a detailed analysis of all timing delays for a circuit that is compiled for implementation in an FPGA chip. This tutorial discusses the types of analyses performed and shows how particular timing requirements may be specified by the user. The discussion assumes that the reader is familiar with the basic operation of Quartus II software, as may be learned from an introductory tutorial.

Doing this tutorial, the reader will learn about:

- Parameters evaluated by the Timing Analyzer
- Specifying the desired values of timing parameters
- Using timing simulation

The timing results shown in the examples in this tutorial were obtained using Quartus II version 8.0, but other versions of the software can also be used.

1 Example Circuit

Timing issues are most important in circuits that involve long paths through combinational logic elements with registers at inputs and outputs of these paths. As an example, we will use the adder/subtractor circuit shown in Figure 1. It can add, subtract, and accumulate n-bit numbers using the 2's complement number representation. The two primary inputs are numbers $A = a_{n-1}a_{n-2} \cdots a_0$ and $B = b_{n-1}b_{n-2} \cdots b_0$, and the primary output is $Z = z_{n-1}z_{n-2} \cdots z_0$. Another input is the *AddSub* control signal which causes $Z = A + B$ to be performed when *AddSub* = 0 and $Z = A - B$ when *AddSub* = 1. A second control input, *Sel*, is used to select the accumulator mode of operation. If *Sel* = 0, the operation $Z = A \pm B$ is performed, but if *Sel* = 1, then B is added to or subtracted from the current value of Z. If the addition or subtraction operations result in arithmetic overflow, an output signal, *Overflow*, is asserted.

To make it easier to deal with asynchronous input signals, they are loaded into flip-flops on a positive edge of the clock. Thus, inputs A and B will be loaded into registers *Areg* and *Breg*, while *Sel* and *AddSub* will be loaded into flip-flops *SelR* and *AddSubR*, respectively. The adder/subtractor circuit places the result into register *Zreg*.

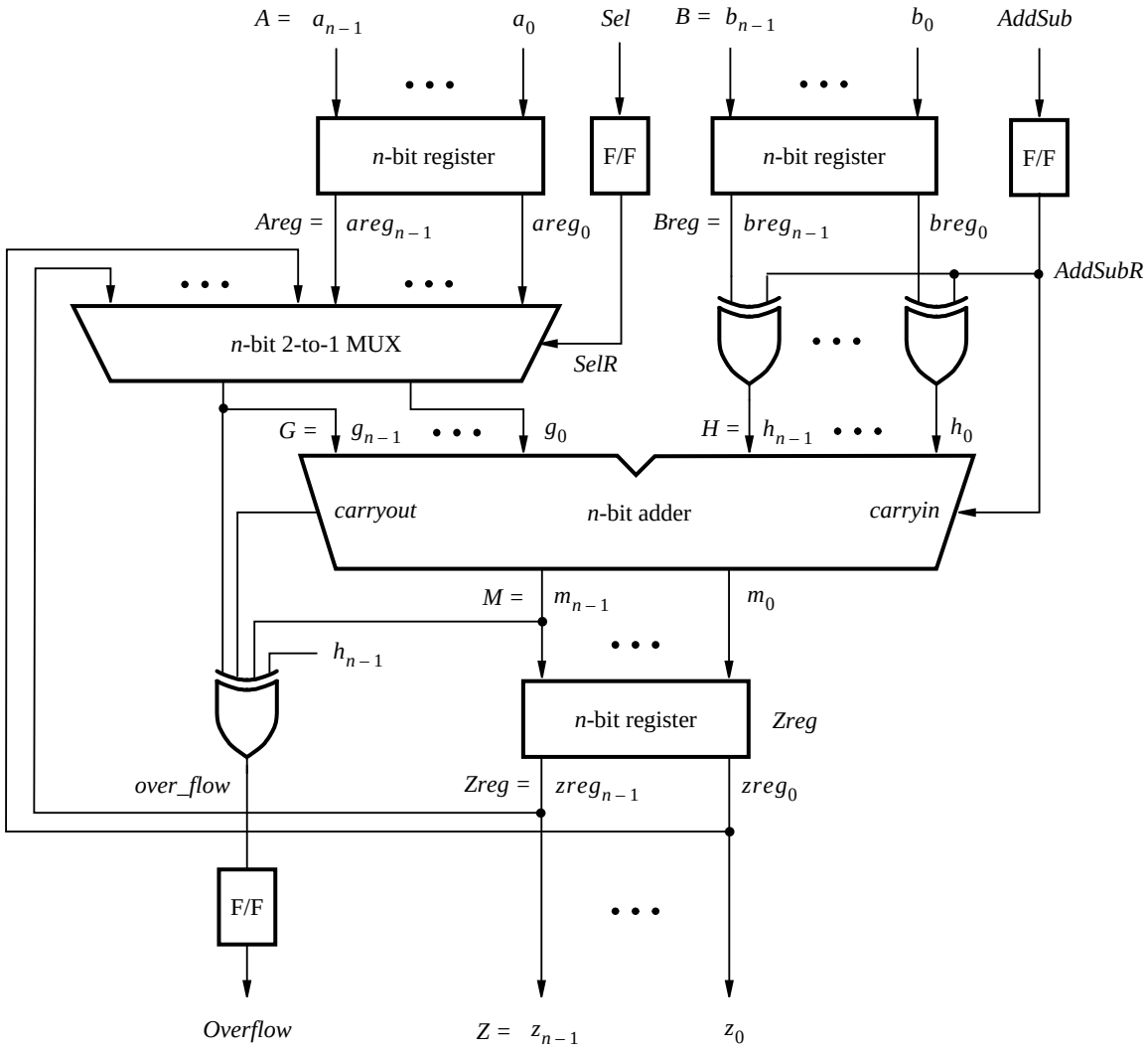


Figure 1. The adder/subtractor circuit.

The required circuit is described by the Verilog code in Figure 2. For our example, we use a 16-bit circuit as specified by $n = 16$. Implement this circuit as follows:

- Create a project *addersubtractor*.
- Include a file *addersubtractor.v*, which corresponds to Figure 2, in the project. For convenience, this file is provided in the directory *DE2_tutorials\design_files*, which is included on the CD-ROM that accompanies the DE2 board and can also be found on Altera's DE2 web pages.
- Choose the Cyclone II EP2C35F672C6 device, which is the FPGA chip on Altera's DE2 board.
- Compile the design.

```
// 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
endmodule

// k-bit 2-to-1 multiplexer
module  mux2to1 (V, W, Selm, F);
    parameter k = 8;
    input  [k−1:0] V, W;
    input  Selm;
    output  [k−1:0] F;
    reg  [k−1:0] F;

    always @(V or W or Selm)
        if (Selm == 0)  F = V;
        else  F = W;
endmodule
```

... continued in Part *b*

Figure 2. Verilog code for the circuit in Figure 1 (Part *a*).

```
// k-bit adder
module   adderk (carryin, X, Y, S, carryout);
    parameter k = 8;
    input  [k−1:0] X, Y;
    input  carryin;
    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
```

Figure 2. Verilog code for the circuit in Figure 1 (Part *b*).

2 Timing Analyzer Report

Successful compilation of our circuit generates the Compilation Report in Figure 3. This report provides a lot of useful information. It shows the number of logic elements, flip-flops (called registers), and pins needed to implement the circuit. It gives detailed information produced by the Synthesis and Fitter modules. It also indicates the speed of the implemented circuit. A good measure of the speed is the maximum frequency at which the circuit can be clocked, referred to as *fmax*. This measure depends on the longest delay along any path, called the *critical path*, between two registers clocked by the same clock. Quartus II software performs a timing analysis to determine the expected performance of the circuit. It evaluates several parameters, which are listed in the Timing Analyzer section of the Compilation Report. Click on the small + symbol next to Timing Analyzer to expand this section of the report, and then click on the Timing Analyzer item Summary which displays the table in Figure 4. The last entry in the table shows that the maximum frequency for our circuit implemented on the specified chip is 251.38 MHz. You may get a different value of *fmax*, dependent on the specific version of Quartus II software that you are using. To see the paths in the circuit that limit the *fmax*, click on the Timing Analyzer item Clock Setup: 'Clock' in Figure 4 to obtain the display in Figure 5. This table shows that the critical path begins at bit 4 of register *Z* and ends at the flip-flop *Overflow*.

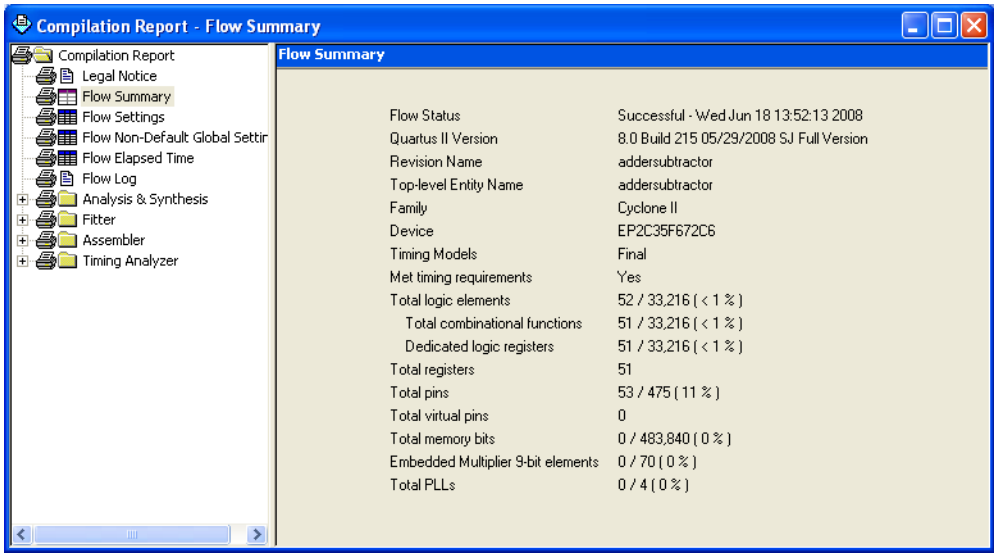


Figure 3. The Compilation Report.

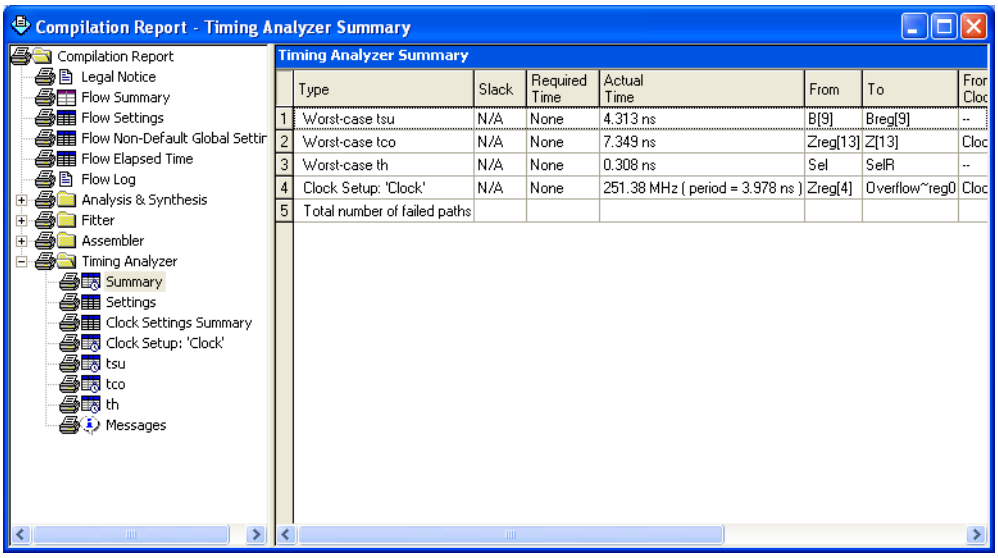


Figure 4. The Timing Analyzer Summary.

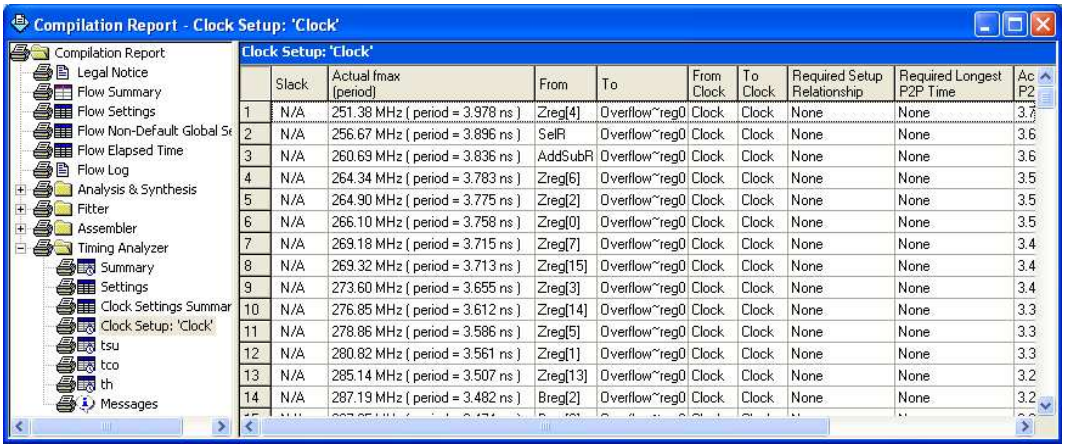


Figure 5. Critical paths.

The table in Figure 4 also shows other timing results. While *fmax* is a function of the longest propagation delay between two registers in the circuit, it does not indicate the delays with which output signals appear at the pins of the chip. The time elapsed from an active edge of the clock signal at the clock source until a corresponding output signal is produced (from a flip-flop) at an output pin is denoted as the *tco* delay at that pin. In the worst case, the *tco* in our circuit is 7.349 ns. Click on tco in the Timing Analyzer section to view the table given in Figure 6. The first entry in the table shows that it takes 7.349 ns from when an active clock edge occurs until a signal propagates from bit 13 in register *Zreg* to the output pin z_{13}. The other two parameters given in Figure 4 are setup time, *tsu*, and hold time, *th*.

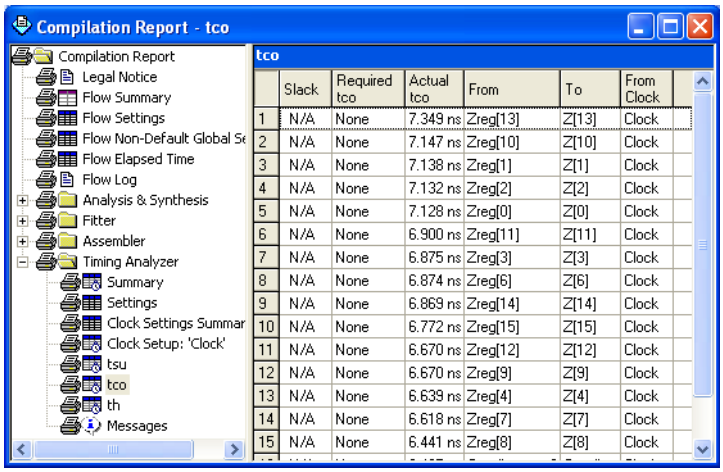


Figure 6. The *tco* delays.

3 Specifying Timing Constraints

So far we have compiled our Verilog code without indicating to the Quartus II software the required speed performance of the circuit. In the absence of such timing constraints the Quartus II software implements a designed circuit in a good but not necessarily the best way in order to keep the compilation time short. If the result does not meet the user's expectations, it is possible to specify certain timing constraints that should be met. For example, suppose that we want our example circuit to operate at a clock frequency of at least 255 MHz, rather than the 251.38 MHz as indicated by the value of *fmax* in Figure 4. To see if this can be achieved we can set the *fmax* constraint as follows:

1. Select Assignments > Timing Analysis Settings > Classic Timing Analyzer Settings to reach the window in Figure 7. In this window it is possible to specify the requirements for a number of different parameters.

2. In the box Clock Settings specify that the required value of *fmax* is 255 MHz. Click OK.

3. Recompile the circuit.

4. Open the Timing Analyzer Summary to see that the new *fmax* is 256.08 MHz, as indicated in Figure 8. You may get a slightly different result depending on the version of the Quartus II software used.

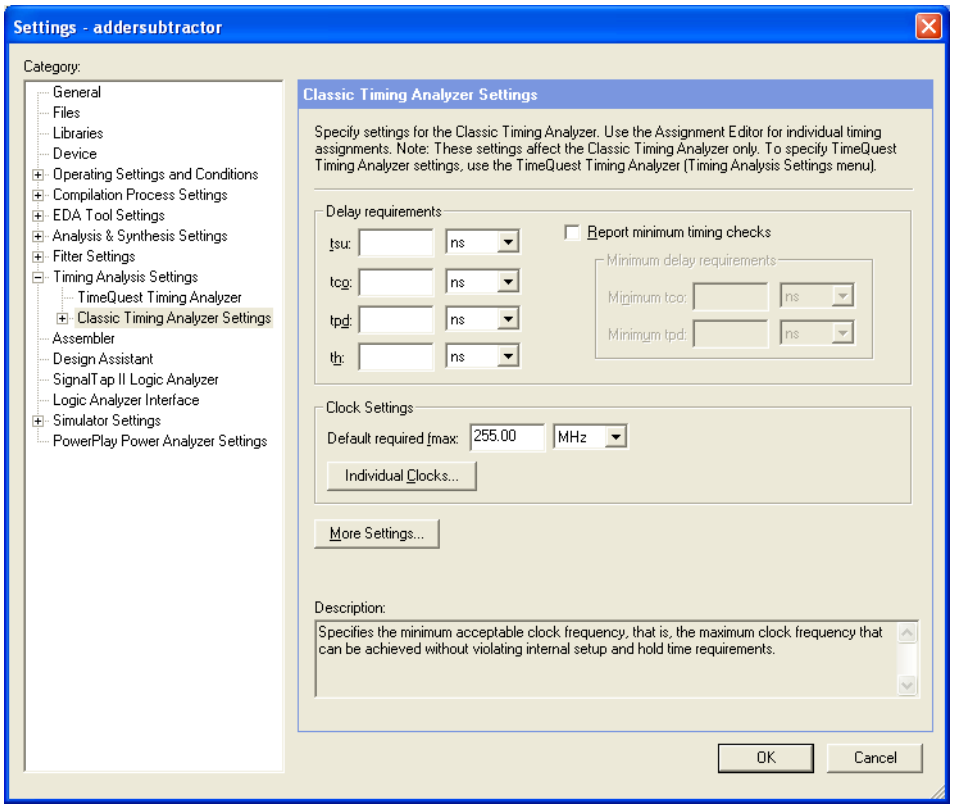


Figure 7. Specify the timing constraints in the Settings window.

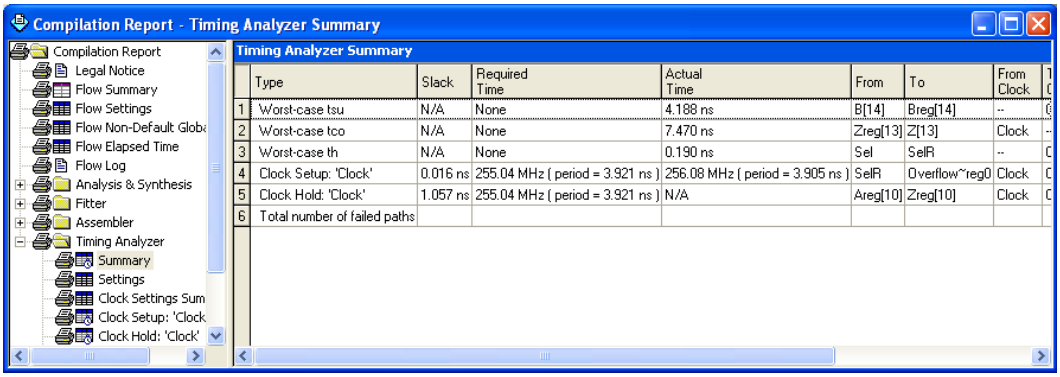


Figure 8. New timing results.

If the specified constraint is too high, the Quartus II compiler will not be able to satisfy it. For example, set the *fmax* constraint to 300 MHz and recompile the circuit. Now, the Timing Analyzer Summary will show that this constraint cannot be met, as seen in Figure 9.

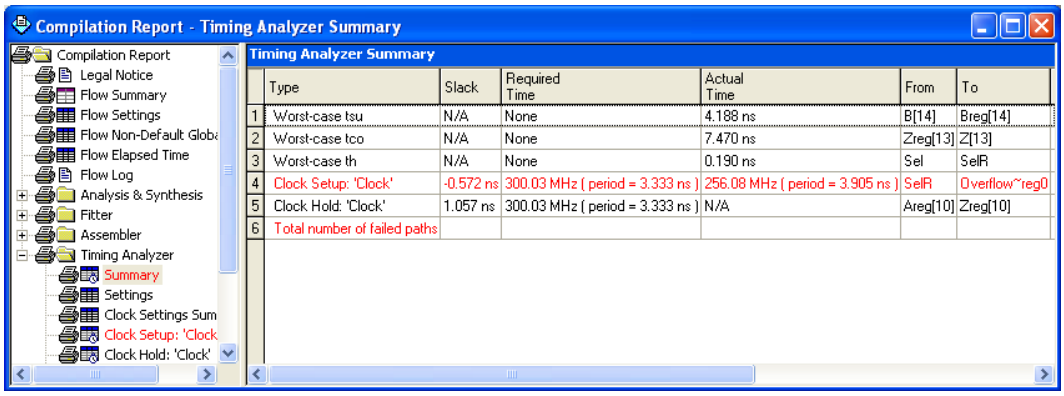


Figure 9. The timing constraint cannot be met.

The specified *fmax* of 300 MHz cannot be achieved because one or more paths in the circuit have long propagation delays. To locate the most critical path highlight the Clock Setup entry in the table by clicking on it. Then, right-click to get the pop-up menu shown in Figure 10. Select Locate > Locate in RTL Viewer which will cause the RTL Viewer to display the critical path as presented in Figure 11. Note that this path begins at flip-flop *SelR* and ends at the *Overflow* flip-flop.

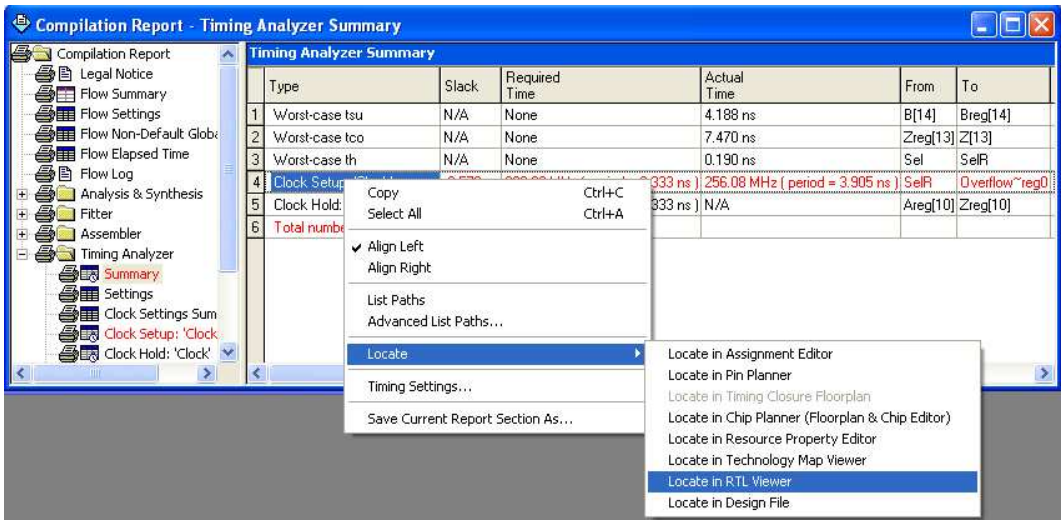


Figure 10. Locate the critical path.

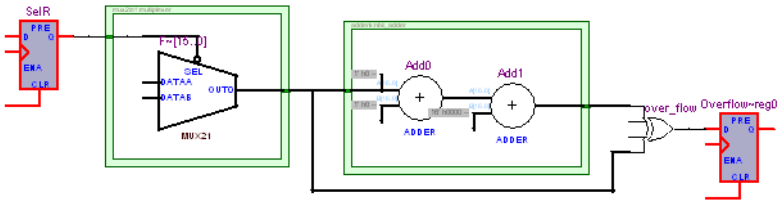
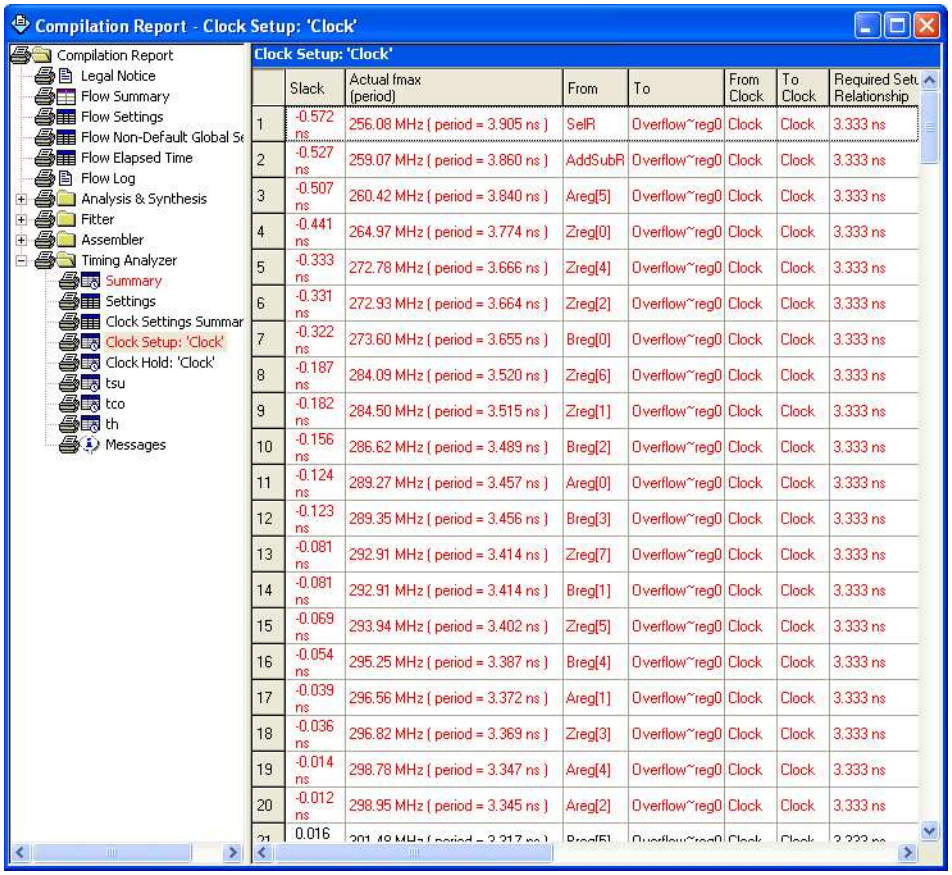


Figure 11. Path for which the timing constraint cannot be met.

It is likely that there are other paths that make it impossible to meet the specified constraint. To identify these paths choose Clock Setup: 'Clock' on the left side of the Compilation Report in Figure 9. As seen in Figure 12, there are 20 paths with propagation delays that are too long. Observe a column labeled Slack. The term *slack* is used to indicate the margin by which a timing requirement is met or not met. In the top row in Figure 12 we see that the timing delays along the path from the *SelR* flip-flop to the *Overflow* flip-flop are 0.572 ns longer than the maximum of 4 ns that is the period of the 250-MHz clock specified as the *fmax* constraint.



	Slack	Actual fmax (period)	From	To	From Clock	To Clock	Required Setu Relationship
1	-0.572 ns	256.08 MHz (period = 3.905 ns)	SelR	Overflow~reg0	Clock	Clock	3.333 ns
2	-0.527 ns	259.07 MHz (period = 3.860 ns)	AddSubR	Overflow~reg0	Clock	Clock	3.333 ns
3	-0.507 ns	260.42 MHz (period = 3.840 ns)	Areg[5]	Overflow~reg0	Clock	Clock	3.333 ns
4	-0.441 ns	264.97 MHz (period = 3.774 ns)	Zreg[0]	Overflow~reg0	Clock	Clock	3.333 ns
5	-0.333 ns	272.78 MHz (period = 3.666 ns)	Zreg[4]	Overflow~reg0	Clock	Clock	3.333 ns
6	-0.331 ns	272.93 MHz (period = 3.664 ns)	Zreg[2]	Overflow~reg0	Clock	Clock	3.333 ns
7	-0.322 ns	273.60 MHz (period = 3.655 ns)	Breg[0]	Overflow~reg0	Clock	Clock	3.333 ns
8	-0.187 ns	284.09 MHz (period = 3.520 ns)	Zreg[6]	Overflow~reg0	Clock	Clock	3.333 ns
9	-0.182 ns	284.50 MHz (period = 3.515 ns)	Zreg[1]	Overflow~reg0	Clock	Clock	3.333 ns
10	-0.156 ns	286.62 MHz (period = 3.489 ns)	Breg[2]	Overflow~reg0	Clock	Clock	3.333 ns
11	-0.124 ns	289.27 MHz (period = 3.457 ns)	Areg[0]	Overflow~reg0	Clock	Clock	3.333 ns
12	-0.123 ns	289.35 MHz (period = 3.456 ns)	Breg[3]	Overflow~reg0	Clock	Clock	3.333 ns
13	-0.081 ns	292.91 MHz (period = 3.414 ns)	Zreg[7]	Overflow~reg0	Clock	Clock	3.333 ns
14	-0.081 ns	292.91 MHz (period = 3.414 ns)	Breg[1]	Overflow~reg0	Clock	Clock	3.333 ns
15	-0.069 ns	293.94 MHz (period = 3.402 ns)	Zreg[5]	Overflow~reg0	Clock	Clock	3.333 ns
16	-0.054 ns	295.25 MHz (period = 3.387 ns)	Breg[4]	Overflow~reg0	Clock	Clock	3.333 ns
17	-0.039 ns	296.56 MHz (period = 3.372 ns)	Areg[1]	Overflow~reg0	Clock	Clock	3.333 ns
18	-0.036 ns	296.82 MHz (period = 3.369 ns)	Zreg[3]	Overflow~reg0	Clock	Clock	3.333 ns
19	-0.014 ns	298.78 MHz (period = 3.347 ns)	Areg[4]	Overflow~reg0	Clock	Clock	3.333 ns
20	-0.012 ns	298.95 MHz (period = 3.345 ns)	Areg[2]	Overflow~reg0	Clock	Clock	3.333 ns
21	0.016						

Figure 12. The longest delay paths.

We have shown how to set the *fmax* constraint. The other constraints depicted in the window in Figure 7 can be set in the same way.

4 Timing Simulation

Timing simulation provides a graphical indication of the delays in the implemented circuit, as can be observed from the displayed waveforms. For a discussion of simulation see the tutorial *Quartus II Simulation with Verilog Designs*, which uses the same *addersubtractor* circuit as an example.