Worcester Polytechnic Institute

ECE 5723: Methodologies for System Level Design and Modeling

Logic Design, VHDL Modeling, and Testbench

by

Vladimir Vakhter

Laboratory Report

09/12/2020

Table of Contents

1. Introduction

This work aims to implement in VHDL a frequency divider circuit that divides the input clock frequency by 462 with a 50 percent duty cycle. For this, an 8-bit up-counter and a toggle flip-flop were implemented first, and then interconnected in the top-level module of the frequency divider.

2. Part 1 – 8-bit Up Counter

2.1. Description of the Task

Write RT-level VHDL description for an 8-bit up counter with rising-edge clock, asynchronous reset, count-enable, load-enable, carry-in, carry-out, 8-bit parallel input, and 8- bit parallel output. All control inputs are active high.

2.2. Diagram of Hardware Module

The diagram of the 8-bit up-counter is demonstrated in Fig. 1.

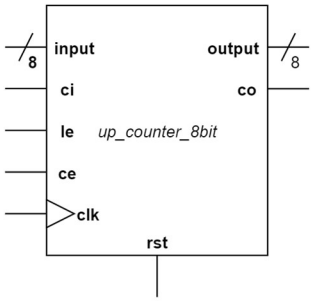


Fig. 1 – Diagram of 8-bit up-counter

In the above diagram, the input signals are denoted as following: *input* – 8-bit parallel input, *ci* – carry-in, *le* – load-enable, *ce* – count-enable, *clk* – rising-edge clock, and *rst* – asynchronous reset. The output signals embrace: *output* – 8-bit parallel output, *co* – carry-out.

2.3. Simulation of Hardware Module

A testbench was written for the designed 8-bit up-counter. The results of simulation are shown in Fig. 2 - Fig. 7.

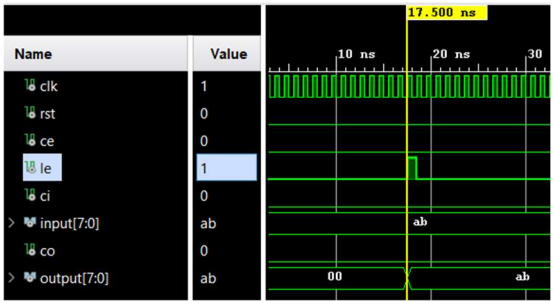


Fig. 2 – *le = '1'* loads the 8-bit *input* added with *ci* to the output (here, ci = '0')

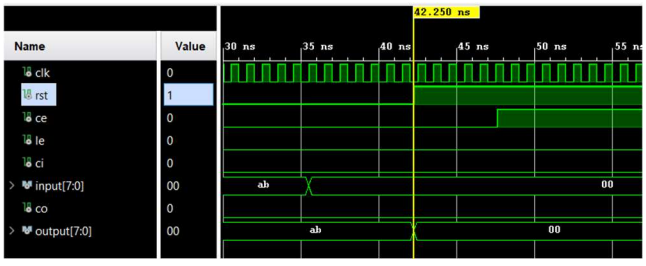


Fig. 3 – Asynchronous *rst = '1'* resets the counter (it stays reset even when *ce = '1'*)

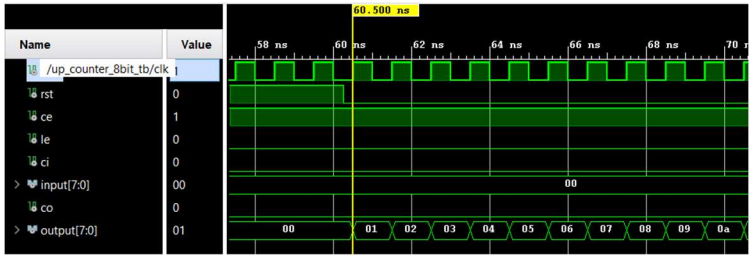


Fig. 4 – When *ce = '1'* (*rst = '0'*), the counter starts to count

Fig. 5 – When the counter counts to 'FF', *co = '1'* for one clock period, and the count begins again

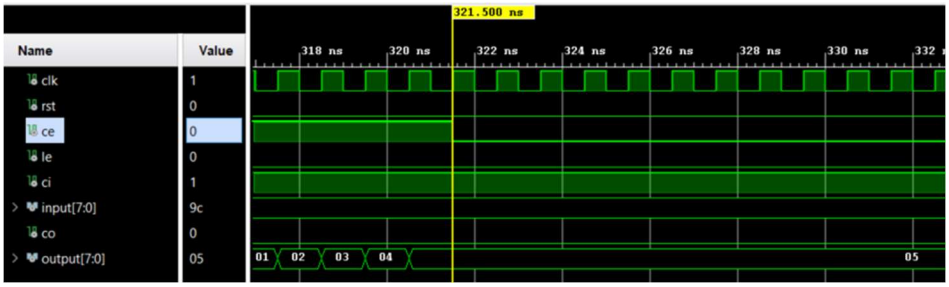


Fig. 6 – When *ce = '0'*, the counter stops to count and keeps its last output value

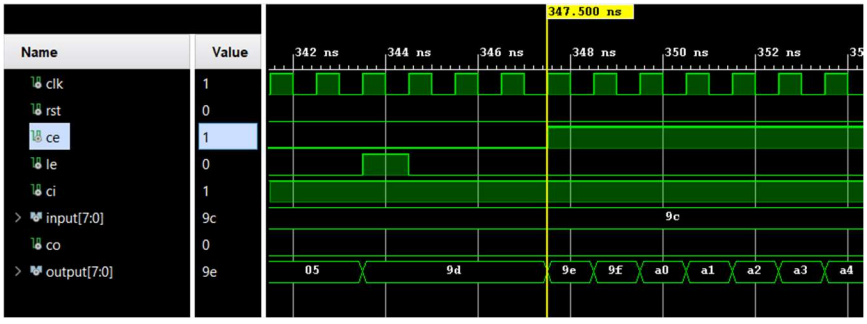


Fig. 7 – When *le = '1'*, the counter loads the input value (here, *input = '9C'*) added with ci (here, *ci = '1'*); when *ce = '1'*, the counter starts to count since the loaded value (here, '9D') until it reaches 'FF'

3. Part 2 – Clock Divider

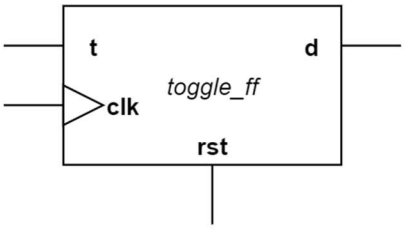
3.1. Description of the Task

Use a Toggle Flip-Flop (T-FF) and the counter in Part 1 to build a circuit that divides the clock frequency by 462 with a 50 percent duty cycle. For dividing the clock frequency (*cf*) by number *d* (here *d* is 462), you are to use a modulo-*n* counter (here *n* is 256) that starts counting from *n-(d/2)* and goes up to *n-1*. In this case, the frequency of the carry-out signal becomes *cf/(d/2)*. To achieve this, take the carry-out and run it into the counter load input. For the parallel input use *n-(d/2)*. When the count reaches *n-1*, carry-out becomes 1 that causes *n-(d/2)* to be loaded into the counter on the next clock. Then the count continues until the count reaches *n-1* again. In the next step, you should add a T-FF to the carry-out signal of the counter to produce the frequency *cf/d* with a 50 percent duty cycle. As you know, T-FF behaves as a "divide-by-2" counter.

3.2. Diagrams of Hardware Modules

3.2.1. Toggle Flip-Flop

The diagram of the toggle flip-flop (T-FF) is demonstrated in Fig. *8*.



clk	t	d	
		Current State	Next State
↗	0	0	0
↗	0	1	1
↗	1	0	1
↗	1	1	0

Fig. 8 – Diagram and truth-table of toggle flip-flop: *t* – toggle input, *clk* – rising-edge clock, *rst* – asynchronous reset, and *d* – output

If the t-input is kept at logic high and the initial clock is used as the T-FF clock, the d-output will change its value once per clock period (since the flip-flop is sensitive only to the rising edge). The output clock will be divided by two.

3.2.2. Clock Divider

The diagram of the clock divider is demonstrated in Fig. *9*.

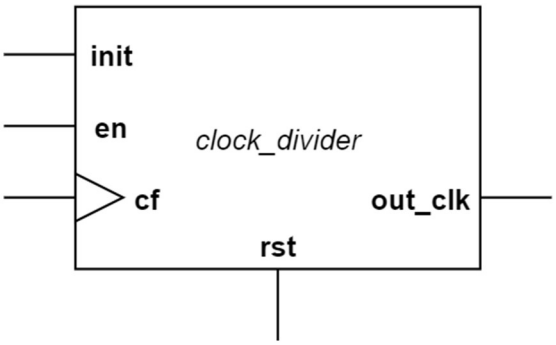


Fig. 9 – Diagram of clock divider: init – input initialization signal, cf – input clock frequency, en – divide-enable, rst – asynchronous reset, and out_clk – output clock frequency

The schematic of the clock divider is demonstrated in Fig. 10.

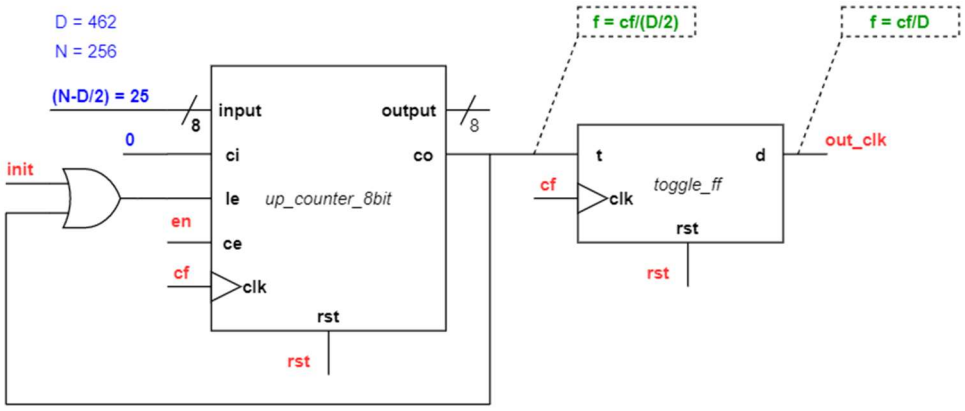


Fig. 10 – Schematic of clock divider: input signals are drawn in red, constant values/drivers are drawn in blue, and the output frequencies in control points are drawn in green

When the counter first starts, it starts with 0. The first time around, divide by *d* would not happen and the counter would count *n*. To mitigate this situation, an additional control signal called '*init*' was introduced for initialization. Then, *le* = *co* OR *init* as it is shown in the above figure.

For the design of frequency divider, a hierarchical structure was implemented. First, all the necessary modules were implemented, and then they were instantiated in the top-level module as shown in Fig. 11.

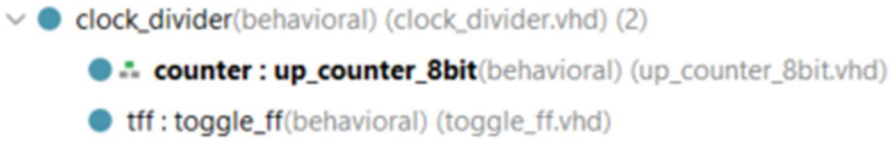


Fig. 11 – Hierarchical structure of the design

3.3. Simulation of Hardware Module

3.3.1. Toggle Flip-Flop

A testbench was written for the T-FF. The results of simulation are shown in Fig. 12.

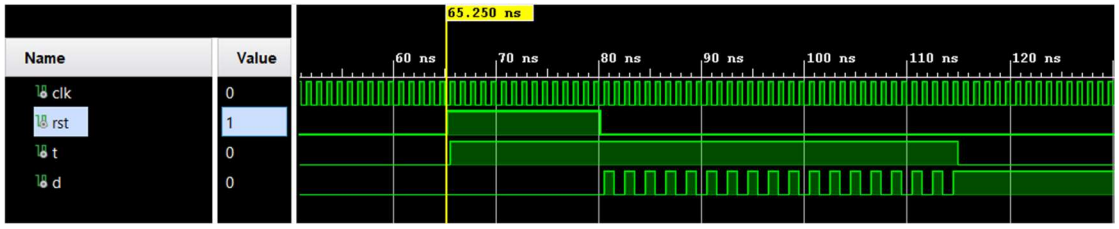


Fig. 12 – T-FF with asynchronous reset

3.3.2. Clock Divider

A testbench was written for the designed clock divider. The results of simulation are shown in Fig. 13 - Fig. 17.

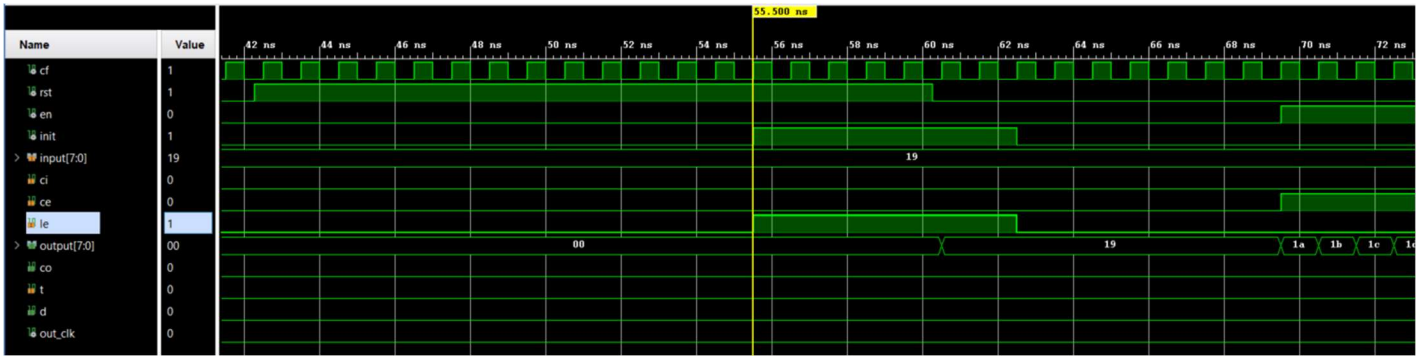


Fig. 13 – While *rst = '1'*, the clock divider is reset. When *rst = '0'*, the circuit is sensitive to the input signals. Here, *init = '1'*, and therefore, *input = 0x19* added with *ci = '0'* are loaded to the 8-bit up-counter

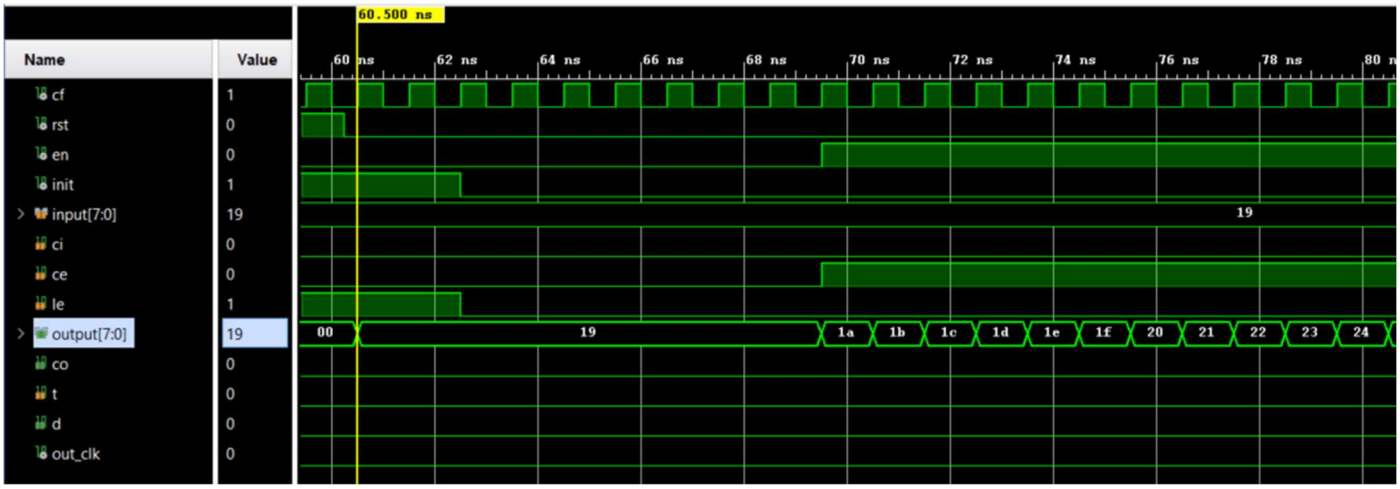


Fig. 14 – When *en = '1'*, the 8-bit up-counter, as a part of the clock divider, starts to count from the preloaded value of 0x19

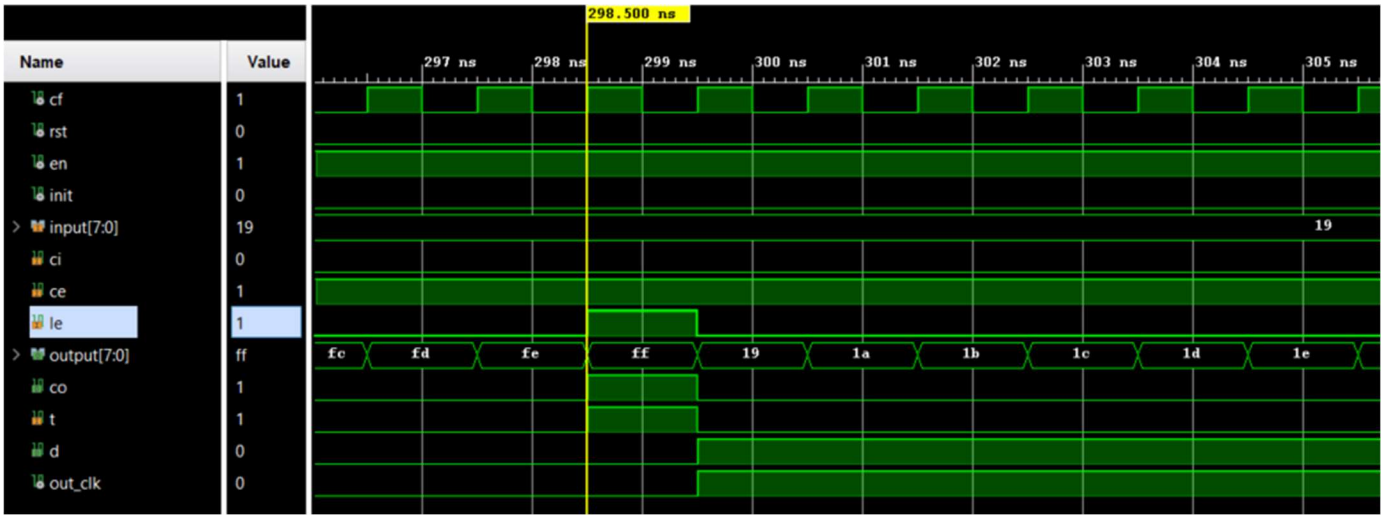


Fig. 15 – When the counter reaches its maximum value (0xFF), *co = '1'*. This signal also feeds the toggle input *t* of T-FF which causes the output *d* of T-FF to trigger and form the output clock *out_clk*

Fig. 16 – The initial frequency *cf* = 1 GHz (T = 1ns) is down sampled 462 times to *out_clk* (T = 761.5ns – 299.5ns = 462ns => *out_clk = 1/462*cf*; duty cycle = (530.5ns – 299.5ns)/462ns = 0.5) (Fig.16a shows the rising edge of out_clk, Fig.16b shows the falling edge of out_clk, and Fig.16c shows the next rising edge of out_clk)

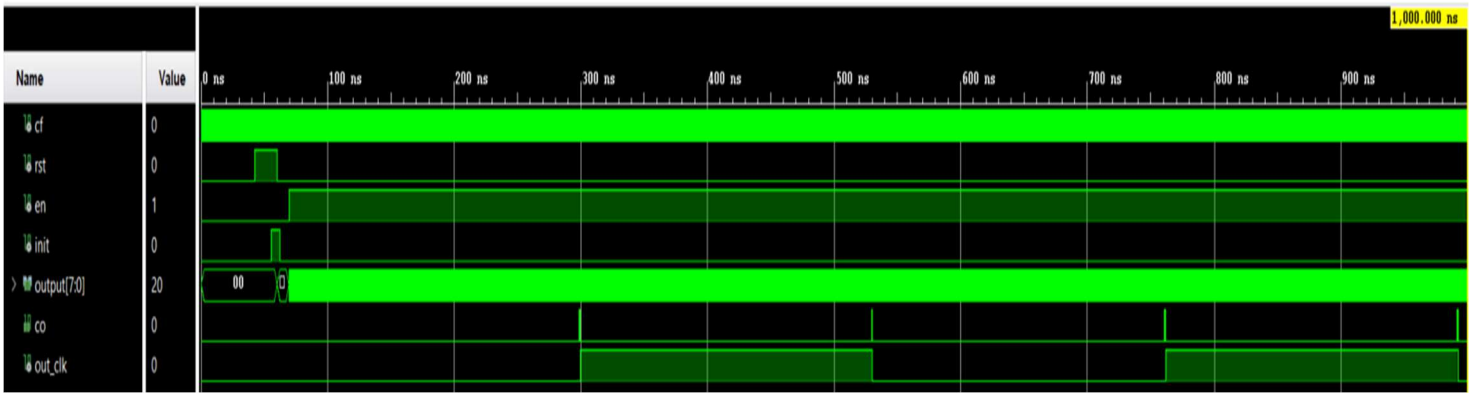


Fig. 17 – The overall picture of the waveforms generated in the testbench

4. Conclusions

In this work, a frequency divider circuit that divides the input clock frequency by 462 with a 50 percent duty cycle was implemented in VHDL. All VHDL codes were simulated and verified with testbenches.