MSc in Computer Engineering

Electronics and Communications Systems

Project report: Timer

Edoardo Pantè

Academic Year: 2022/2023

Summary

1. INTRODUCTION 3
1.1 BACKGROUND 3
1.2 PROJECT REQUIREMENTS 3
2. DESCRIPTION OF THE ARCHITECTURE 4
3. VHDL CODE 5
3.1 PRESCALER 5
3.2 COUNTER 6
3.3 COMPARATOR 6
3.4 TIMER 7
4. TESTING 9
4.1 TEST PLAN 9
4.2 TESTBENCH 9
4.3 TEST RESULTS 10
5. SYNTHESIS 12
5.1 VIVADO'S ELABORATED DESIGN 12
5.2 TIMING CONSTRAINTS 12
5.3 SETUP AND HOLD SLACKS 12
5.4 CRITICAL PATH 13
5.5 UTILIZATION REPORT 13
5.6 POWER REPORT 14
6. IMPLEMENTATION 15
6.1 VIVADO'S ELABORATED DESIGN 15
6.2 WARNINGS AND DRC VIOLATIONS 15
6.3 SETUP AND HOLD SLACKS 16
6.4 CRITICAL PATH 16
6.5 UTILIZATION REPORT 17
6.6 POWER REPORT 17
7. CONCLUSIONS 18

1. Introduction

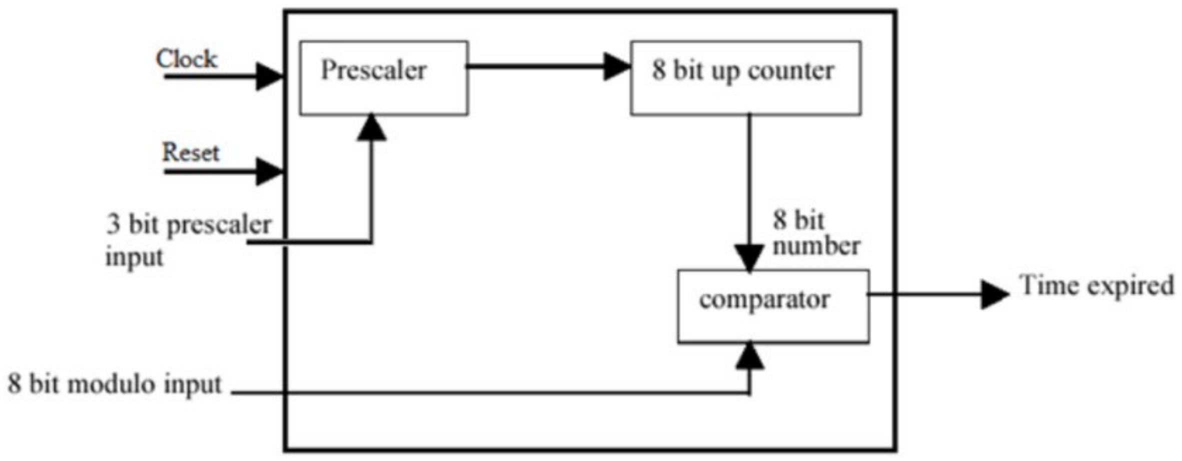
1.1 Background

A timer is a specialized type of clock used for measuring specific time intervals. Over the years this object has found different implementations going from mechanical to electromechanical up to present days electronic ones. There are plenty application fields where electronic timers are useful. Let's think for example about: microwaves, ovens, alarms and so on. All these devices need timers to properly function.

1.2 Project Requirements

The requirements for this project were to design a Timer composed of a Prescaler a Counter and a Comparator by using the VHDL descriptive language. The specifics for the behaviour of the timer needed to be as follows: The timer operates simply by setting the Prescaler to a value from 1 to 8, setting the module value to a number from 0 to 255, and simply supplying the circuit with the clock. The Timer expired output is set to 1 for a clock cycle at intervals determined by the Prescaler and module value. In particular, the Prescaler counts the number of input clock cycles and outputs a pulse, of one clock cycle duration, when the number of clock cycles is a multiple of the Prescaler input value. The counter counts the number of input pulses. The comparator provides a high output when the two input data coincide.

2. Description of the architecture



Inputs:

- Clock: external clock signal to synchronize the device;
- Reset: external reset signal to initialize/reset the device;
- 3-bit prescaler input: prescaler input value [1, 8];
- 8-bit modulo input: modulo input value [0, 255].

Outputs:

- Time expired: triggered when the time expires.

3. VHDL code

3.1 Prescaler

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
USE ieee.numeric_std.all;

entity prescaler is
    port (
          clock : in std_logic;
          reset : in std_logic;
          presc : in  std_logic_vector(3 downto 0);
          pulse : out  std_logic
        );
end prescaler;

architecture struct of prescaler is

signal count : integer;
signal tmp : std_logic;
signal num : integer;

begin
  num <= to_integer(signed(presc));
  process(clock, reset, tmp)
  begin
      if (reset = '0') then
          count <= 1;
          tmp <= '0';
      elsif (clock'event and clock='1') then
        count <= count + 1;
      if count = num then
        tmp <= not tmp;
        count <= 1;
      end if;
      end if;
      pulse <= tmp;
  end process;
end struct;
```

3.2 Counter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
  port (
    reset   : in  std_logic;
    pulse : in  std_logic;
    num_8bit  : out  std_logic_vector(7 downto 0)
  );

end counter;

architecture struct of counter is

signal pulse_count: std_logic_vector(7 downto 0);

  begin
  process(pulse, reset)
  begin
    if (reset='0') then
      pulse_count<="00000000";
    elsif pulse = '1' then
      pulse_count <= pulse_count + 1;
    end if;
    num_8bit<=pulse_count;

  end process;
end struct;
```

3.3 Comparator

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity comparator is
    port(
            reset   : in  std_logic;
            num_8bit : in  std_logic_vector (7 downto 0);
            module_input : in  std_logic_vector(7 downto 0);
            time_expired : out  std_logic
        );
end comparator;

architecture struct of comparator is

begin
```

```vhdl
    process(num_8bit, module_input)
    begin
        if (num_8bit = module_input) then
            time_expired <= '1';
        else
            time_expired <= '0';
        end if;
    end process;
end struct;
```

3.4 Timer

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity timer is
    port(
            clock : in  std_logic;
            reset : in  std_logic;
            presc : in  std_logic_vector(3 downto 0);
            module_input : in  std_logic_vector(7 downto 0);
            time_expired : out  std_logic
        );
end timer;

architecture struct of timer is

signal num_8bit : std_logic_vector (7 downto 0);
signal pulse : std_logic;

component prescaler is
    port(
            clock : in std_logic;
            reset : in std_logic;
            presc : in  std_logic_vector(3 downto 0);
            pulse : out  std_logic
        );
end component;

component counter is
    port(
            reset   : in  std_logic;
            pulse : in  std_logic;
            num_8bit  : out  std_logic_vector(7 downto 0)
        );
end component;

component comparator is
```

```vhdl
    port(
            reset   : in  std_logic;
            num_8bit : in  std_logic_vector (7 downto 0);
            module_input : in  std_logic_vector(7 downto 0);
            time_expired : out  std_logic
        );
end component;

begin
    u1: prescaler
        port map(
                clock => clock,
                reset => reset,
                presc => presc,
                pulse => pulse
            );


    u2: counter
        port map(
                pulse => pulse,
                reset => reset,
                num_8bit => num_8bit
            );

    u3: comparator
        port map(
                num_8bit => num_8bit,
                module_input => module_input,
                time_expired => time_expired,
                reset => reset
            );
end struct;
```

4. Testing

4.1 Test plan

The system behaviour must be correct to different input cases:

#	Prescaler value	Module value	Expected clock cycles
1	0001	00000101	10
2	0010	00000101	20
3	0011	00000101	30
4	0010	00001010	40

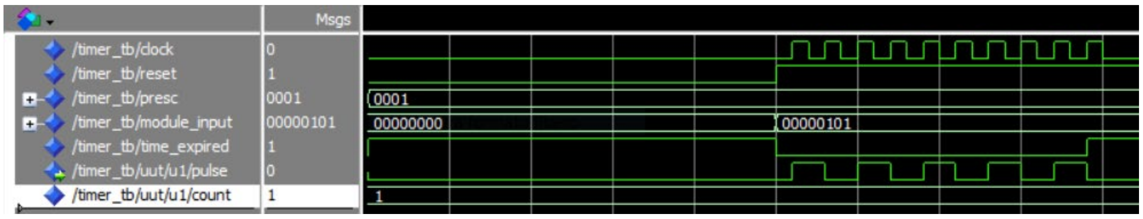
4.2 Testbench

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity timer_tb is
end timer_tb;

architecture struct of timer_tb is

  -- Component declaration for the Unit Under Test (UUT)
  component timer
    port(
        clock : in  std_logic;
        reset : in  std_logic;
        presc : in  std_logic_vector(3 downto 0);
        module_input : in  std_logic_vector(7 downto 0);
        time_expired : out  std_logic
      );
  end component;

  --Inputs
  signal clock : std_logic := '0';
  signal reset : std_logic := '0';
  signal presc : std_logic_vector(3 downto 0) := (others => '0');
  signal module_input : std_logic_vector(7 downto 0) := (others => '0');

  -- Output
  signal time_expired : std_logic := '0';

  -- Clock period definition (500 MHz)
  constant clock_period : time := 2 ns;
  constant t_reset : time := 25 ns;

begin

  -- Instantiate the Unit Under Test (UUT)
  uut: timer port map (
       clock => clock,
       reset => reset,
       presc => presc,
       module_input => module_input,
       time_expired => time_expired
      );

  clock <= (not(clock) and not(time_expired)) after clock_period/2;
  reset <= '1' after t_reset;

  stim_proc : process(clock, reset)
  begin
    if (reset = '0') then
       presc <= "0001";
       module_input <= "00000000";
    else
       presc <= "0010";
       module_input <= "00001010";
    end if;
  end process;
end;
```

4.3 Test results

- **Case 1:**

Prescaler value	Modulo value
0001	00000101

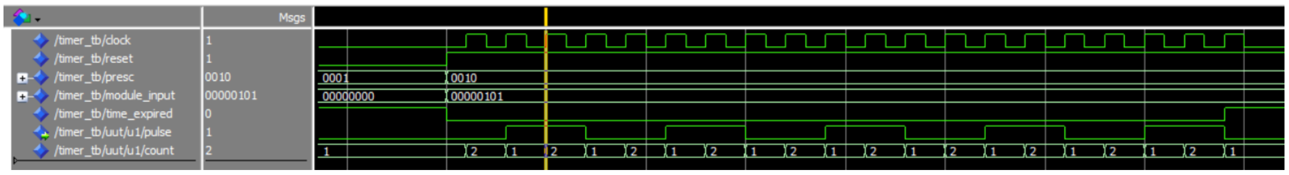


```
                         Msgs
/timer_tb/clock          0
/timer_tb/reset          1
/timer_tb/presc          0001
/timer_tb/module_input   00000101
/timer_tb/time_expired   1
/timer_tb/uut/u1/pulse   0
/timer_tb/uut/u1/count   1
```

As **expected**, the number of clock cycles is **10**.

- **Case 2:**

Prescaler value	**Module value**
0010	00000101

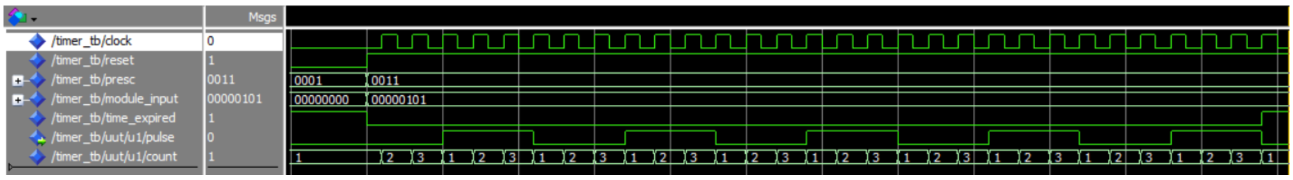


	Msgs
/timer_tb/clock	1
/timer_tb/reset	1
/timer_tb/presc	0010
/timer_tb/module_input	00000101
/timer_tb/time_expired	0
/timer_tb/uut/u1/pulse	1
/timer_tb/uut/u1/count	2

As **expected**, the number of clock cycles is **20**.

- **Case 3:**

Prescaler value	**Module value**
0011	00000101

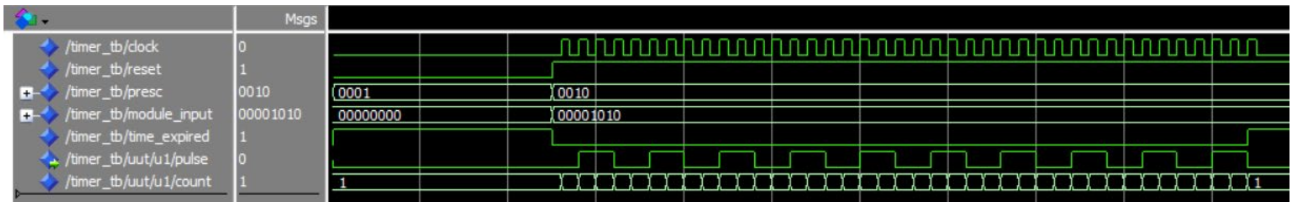


	Msgs
/timer_tb/clock	0
/timer_tb/reset	1
/timer_tb/presc	0011
/timer_tb/module_input	00000101
/timer_tb/time_expired	1
/timer_tb/uut/u1/pulse	0
/timer_tb/uut/u1/count	1

As **expected**, the number of clock cycles is **30**.

- **Case 4:**

Prescaler value	**Module value**
0010	00001010

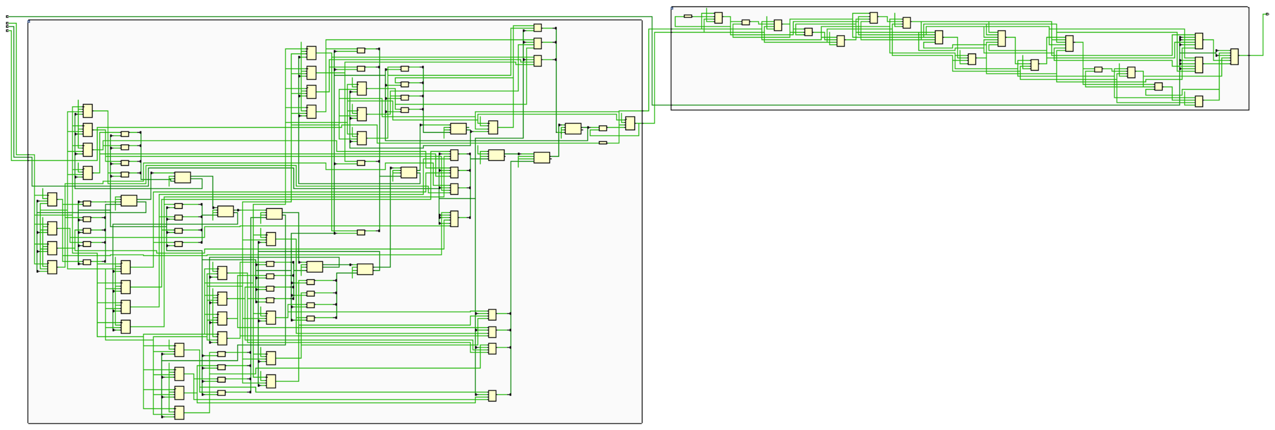


	Msgs
/timer_tb/clock	0
/timer_tb/reset	1
/timer_tb/presc	0010
/timer_tb/module_input	00001010
/timer_tb/time_expired	1
/timer_tb/uut/u1/pulse	0
/timer_tb/uut/u1/count	1

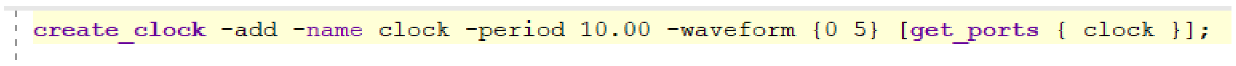
As **expected**, the number of clock cycles is **40**.

5. Synthesis

5.1 Vivado's Elaborated Design

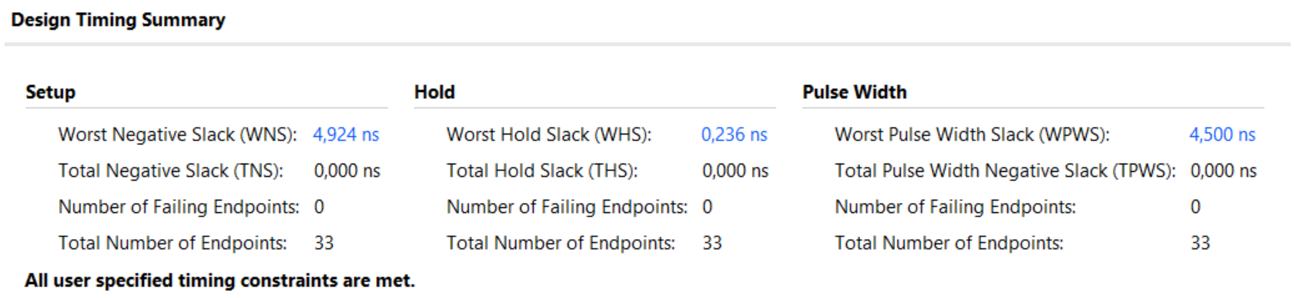


5.2 Timing Constraints



```
create_clock -add -name clock -period 10.00 -waveform {0 5} [get_ports { clock }];
```

5.3 Setup and Hold Slacks



Design Timing Summary

Setup

Worst Negative Slack (WNS):	4,924 ns
Total Negative Slack (TNS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	33

Hold

Worst Hold Slack (WHS):	0,236 ns
Total Hold Slack (THS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	33

Pulse Width

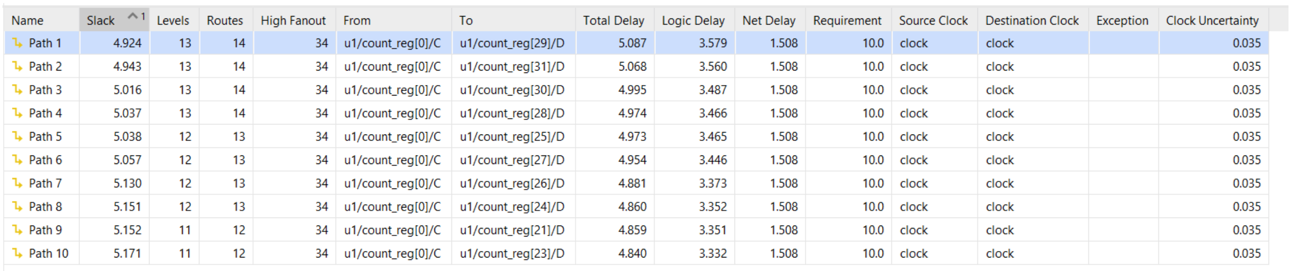
Worst Pulse Width Slack (WPWS):	4,500 ns
Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	33

All user specified timing constraints are met.

As it's clear from the timing summary of the synthesis, since all the time slacks are positive, there are no setup nor hold time violations.

5.4 Critical Path

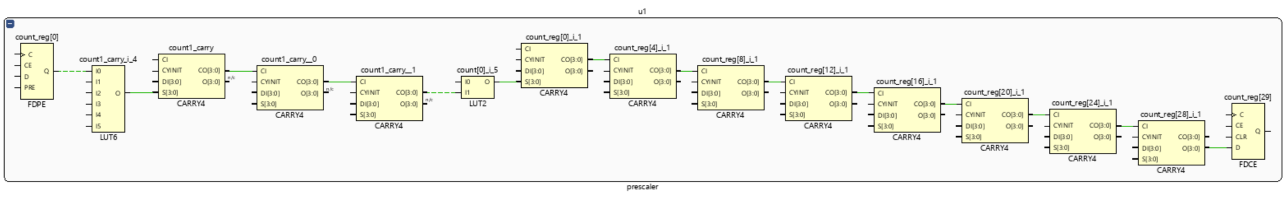
These ones are the register-logic-register paths sorted by tsetup slack time.



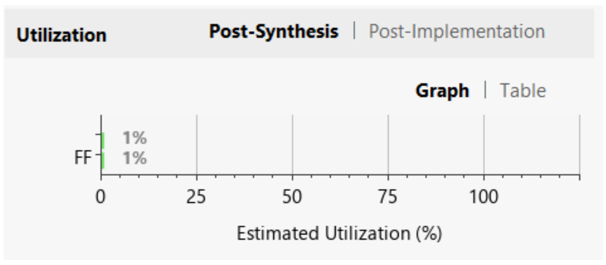
Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	4.924	13	14	34	u1/count_reg[0]/C	u1/count_reg[29]/D	5.087	3.579	1.508	10.0	clock	clock		0.035
Path 2	4.943	13	14	34	u1/count_reg[0]/C	u1/count_reg[31]/D	5.068	3.560	1.508	10.0	clock	clock		0.035
Path 3	5.016	13	14	34	u1/count_reg[0]/C	u1/count_reg[30]/D	4.995	3.487	1.508	10.0	clock	clock		0.035
Path 4	5.037	13	14	34	u1/count_reg[0]/C	u1/count_reg[28]/D	4.974	3.466	1.508	10.0	clock	clock		0.035
Path 5	5.038	12	13	34	u1/count_reg[0]/C	u1/count_reg[25]/D	4.973	3.465	1.508	10.0	clock	clock		0.035
Path 6	5.057	12	13	34	u1/count_reg[0]/C	u1/count_reg[27]/D	4.954	3.446	1.508	10.0	clock	clock		0.035
Path 7	5.130	12	13	34	u1/count_reg[0]/C	u1/count_reg[26]/D	4.881	3.373	1.508	10.0	clock	clock		0.035
Path 8	5.151	12	13	34	u1/count_reg[0]/C	u1/count_reg[24]/D	4.860	3.352	1.508	10.0	clock	clock		0.035
Path 9	5.152	11	12	34	u1/count_reg[0]/C	u1/count_reg[21]/D	4.859	3.351	1.508	10.0	clock	clock		0.035
Path 10	5.171	11	12	34	u1/count_reg[0]/C	u1/count_reg[23]/D	4.840	3.332	1.508	10.0	clock	clock		0.035

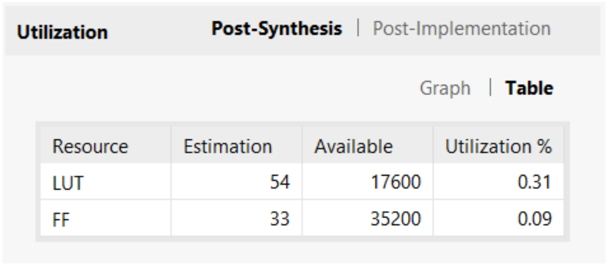
Since $\boldsymbol{Slack} = \boldsymbol{Tck} - \boldsymbol{Tc{-}q} - \boldsymbol{Tprop} - \boldsymbol{Tsetup}$ = **4.924** $\boldsymbol{ns}$, and $\boldsymbol{Tck}$ = **10** $\boldsymbol{ns}$, it's possible to speed up the clock period to reach $\boldsymbol{Tmax\text{-}ck} = \boldsymbol{Tck} - \boldsymbol{Slack}$ = **10** $\boldsymbol{ns}$ **- 4.924** $\boldsymbol{ns}$ **= 5.076** $\boldsymbol{ns}$, getting a maximum frequency of **1/**$\boldsymbol{Tmax\text{-}ck}$ **= 1/5.076 ≅ 197 MHz**. This is an estimation, an upper boundary of the real maximum frequency value, because in this part of the synthesis we are not considering the delay due to the capacitance of the wires that interconnect the inner blocks of the device.

The following screenshot shows the synthetized registers involved in Path 1:



5.5 Utilization Report





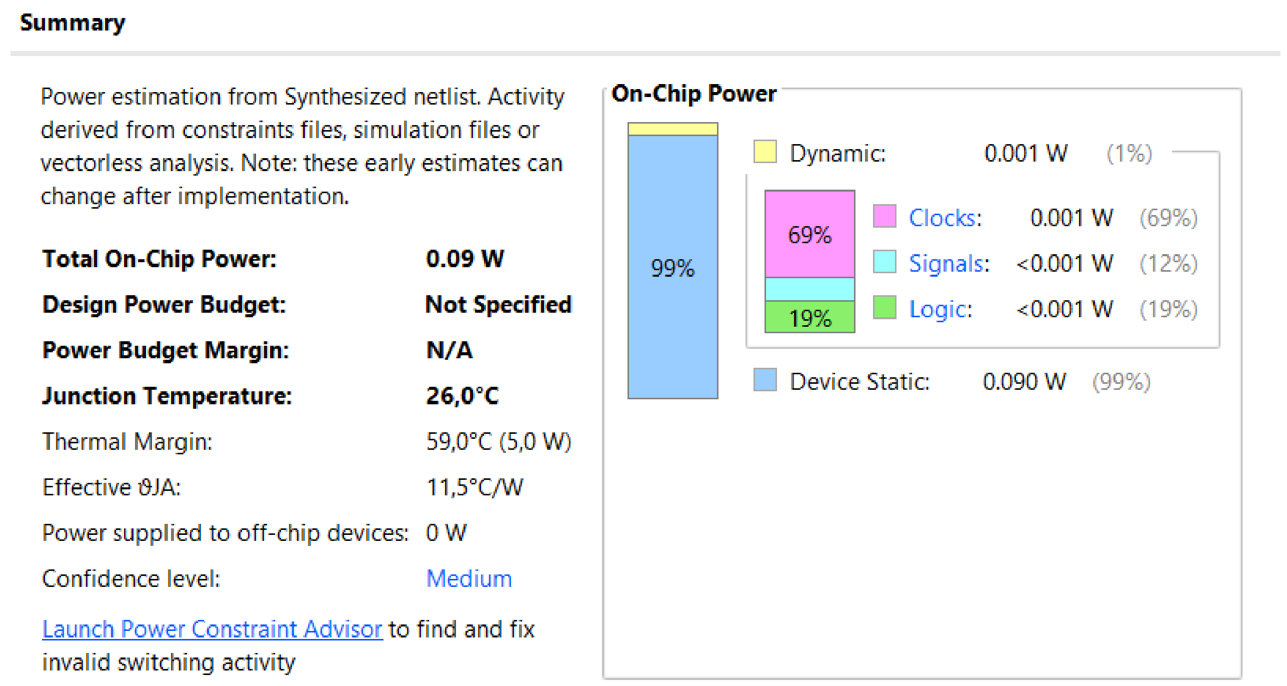
Utilization   **Post-Synthesis** | Post-Implementation

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	54	17600	0.31
FF	33	35200	0.09

5.6 Power Report

Given the default parameters for the environment, the synthesis returns the following power report:



Summary

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.09 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26,0°C**
Thermal Margin:	59,0°C (5,0 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

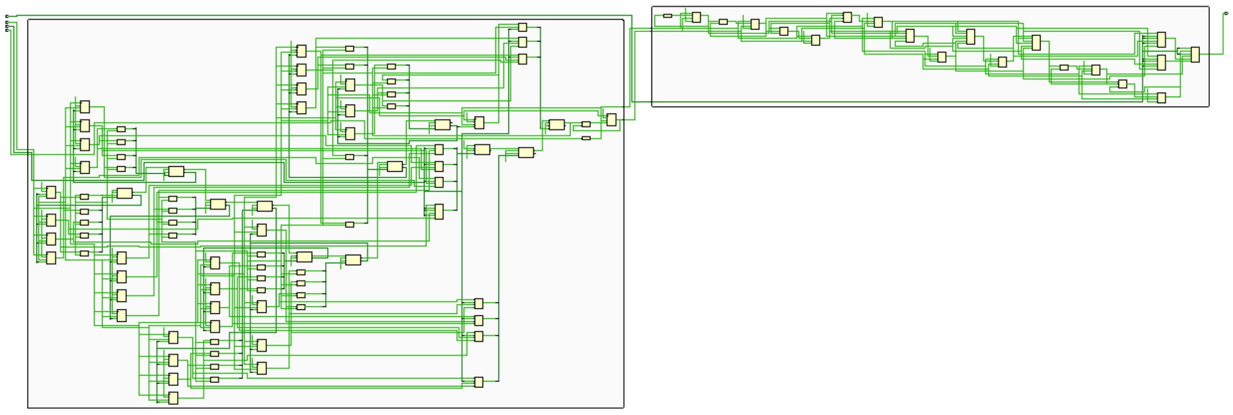
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

🟨	Dynamic:	0.001 W	(1%)
🟪	Clocks:	0.001 W	(69%)
🟦	Signals:	<0.001 W	(12%)
🟩	Logic:	<0.001 W	(19%)
🟦	Device Static:	0.090 W	(99%)

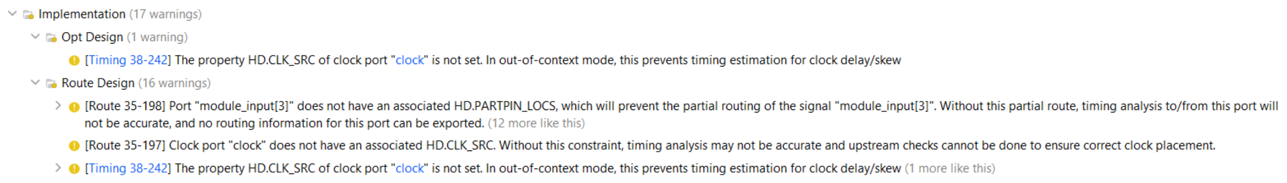
6. Implementation

6.1 Vivado’s Elaborated Design

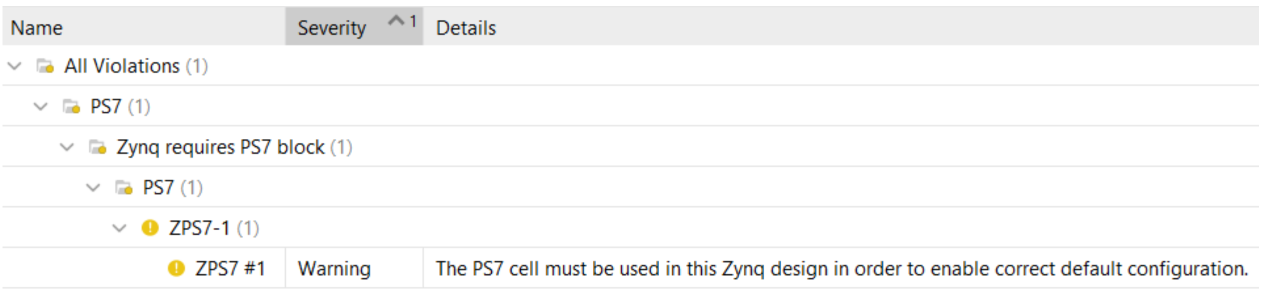


6.2 Warnings and DRC Violations

The following warnings are due to the Out of Context mode, in fact they are related to inputs and outputs that have no information on the placement. They can be ignored.

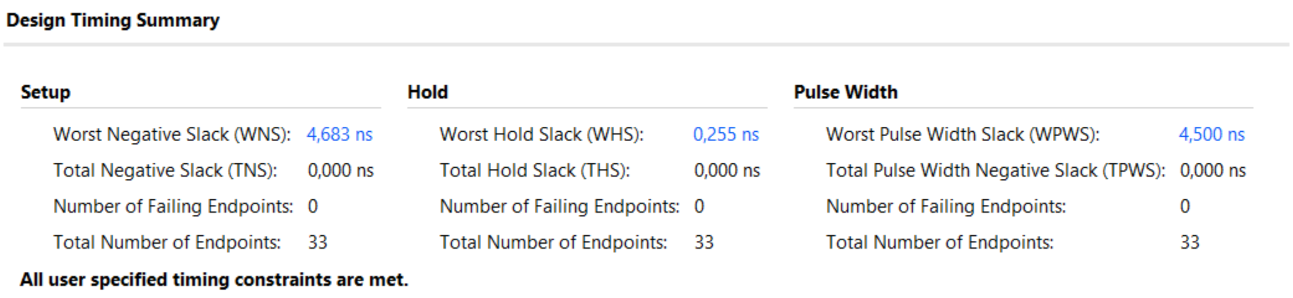


```
Implementation (17 warnings)
    Opt Design (1 warning)
        [Timing 38-242] The property HD.CLK_SRC of clock port "clock" is not set. In out-of-context mode, this prevents timing estimation for clock delay/skew
    Route Design (16 warnings)
        [Route 35-198] Port "module_input[3]" does not have an associated HD.PARTPIN_LOCS, which will prevent the partial routing of the signal "module_input[3]". Without this partial route, timing analysis to/from this port will
        not be accurate, and no routing information for this port can be exported. (12 more like this)
        [Route 35-197] Clock port "clock" does not have an associated HD.CLK_SRC. Without this constraint, timing analysis may not be accurate and upstream checks cannot be done to ensure correct clock placement.
        [Timing 38-242] The property HD.CLK_SRC of clock port "clock" is not set. In out-of-context mode, this prevents timing estimation for clock delay/skew (1 more like this)
```



Name	Severity ^1	Details
All Violations (1)		
PS7 (1)		
Zynq requires PS7 block (1)		
PS7 (1)		
ZPS7-1 (1)		
ZPS7 #1	Warning	The PS7 cell must be used in this Zynq design in order to enable correct default configuration.

6.3 Setup and Hold Slacks



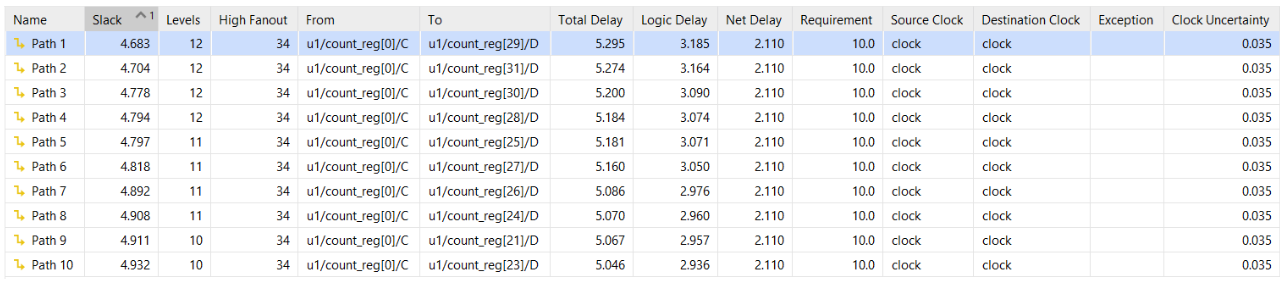
Design Timing Summary

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	4,683 ns	Worst Hold Slack (WHS):	0,255 ns	Worst Pulse Width Slack (WPWS):	4,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	33	Total Number of Endpoints:	33	Total Number of Endpoints:	33

All user specified timing constraints are met.

As it's clear from the timing summary of the implementation, since all the time slacks are positive, there is no setup nor hold time violations. As we can see the Worst Negative Slack is lower than the synthesis's one.

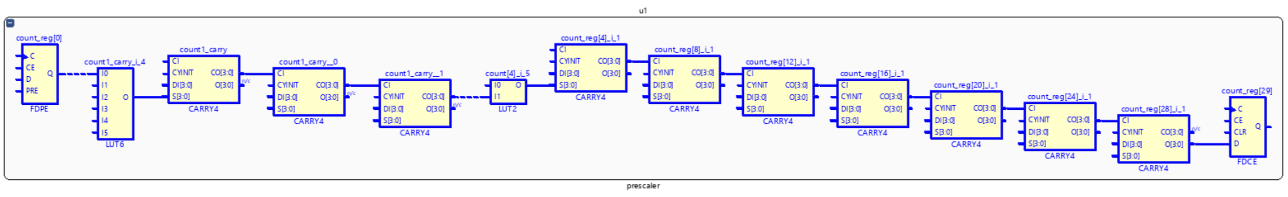
6.4 Critical Path



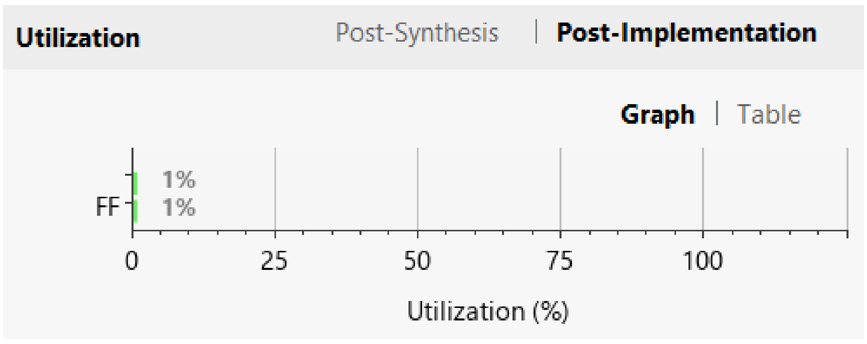
Name	Slack ^1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	4.683	12	34	u1/count_reg[0]/C	u1/count_reg[29]/D	5.295	3.185	2.110	10.0	clock	clock		0.035
Path 2	4.704	12	34	u1/count_reg[0]/C	u1/count_reg[31]/D	5.274	3.164	2.110	10.0	clock	clock		0.035
Path 3	4.778	12	34	u1/count_reg[0]/C	u1/count_reg[30]/D	5.200	3.090	2.110	10.0	clock	clock		0.035
Path 4	4.794	12	34	u1/count_reg[0]/C	u1/count_reg[28]/D	5.184	3.074	2.110	10.0	clock	clock		0.035
Path 5	4.797	11	34	u1/count_reg[0]/C	u1/count_reg[25]/D	5.181	3.071	2.110	10.0	clock	clock		0.035
Path 6	4.818	11	34	u1/count_reg[0]/C	u1/count_reg[27]/D	5.160	3.050	2.110	10.0	clock	clock		0.035
Path 7	4.892	11	34	u1/count_reg[0]/C	u1/count_reg[26]/D	5.086	2.976	2.110	10.0	clock	clock		0.035
Path 8	4.908	11	34	u1/count_reg[0]/C	u1/count_reg[24]/D	5.070	2.960	2.110	10.0	clock	clock		0.035
Path 9	4.911	10	34	u1/count_reg[0]/C	u1/count_reg[21]/D	5.067	2.957	2.110	10.0	clock	clock		0.035
Path 10	4.932	10	34	u1/count_reg[0]/C	u1/count_reg[23]/D	5.046	2.936	2.110	10.0	clock	clock		0.035

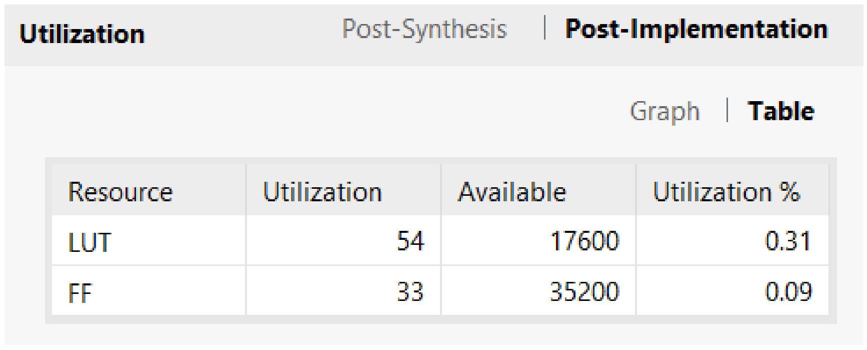
Since $Slack = Tck - Tc\text{–}q - Tprop - Tsetup$ = **4.683** $\boldsymbol{ns}$, and $\boldsymbol{Tck}$ = **10** $\boldsymbol{ns}$, it's possible to speed up the clock period to reach $\boldsymbol{T\text{max-}ck = Tck - Slack}$ **= 10** $\boldsymbol{ns}$ **- 4.683** $\boldsymbol{ns}$ **= 5.317** $\boldsymbol{ns}$, getting a maximum frequency of $\boldsymbol{1/T\text{max-}ck}$ **= 1/5.317 ≅ 188 MHz**.

The following screenshot shows the implemented registers involved in Path 1:



6.5 Utilization Report

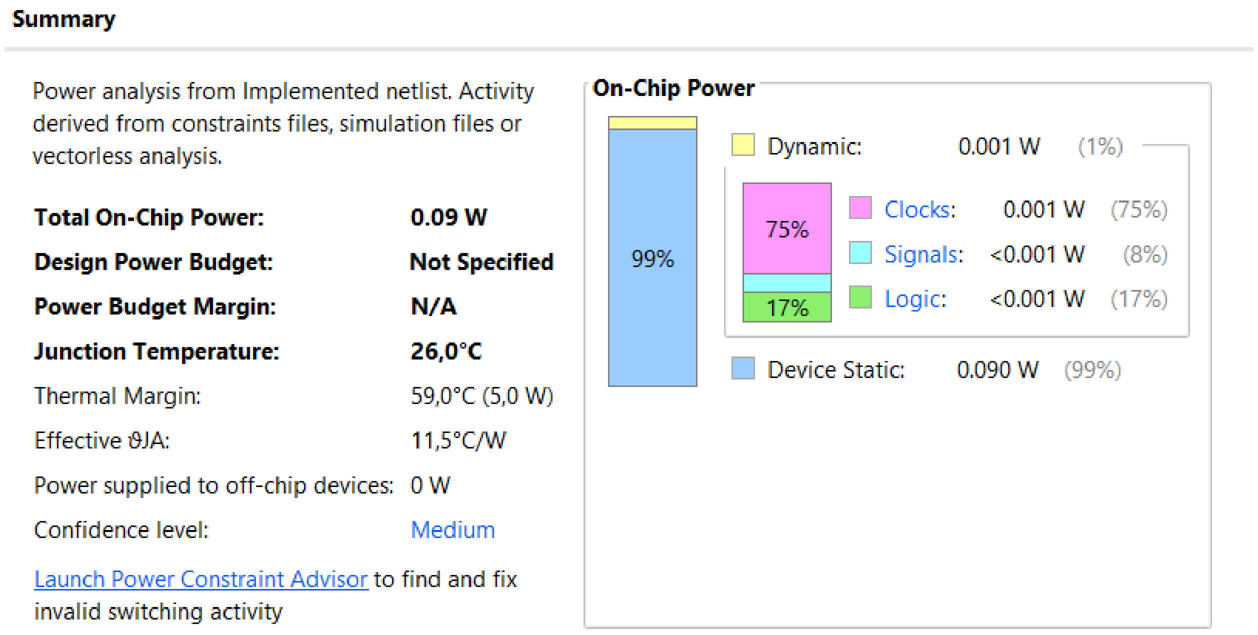




Resource	Utilization	Available	Utilization %
LUT	54	17600	0.31
FF	33	35200	0.09

6.6 Power Report

Given the default parameters for the environment, the implementation returns the following power report:



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.09 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26,0°C**
Thermal Margin:	59,0°C (5,0 W)
Effective ϑJA:	11,5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.001 W	(1%)
Clocks:	0.001 W	(75%)
Signals:	<0.001 W	(8%)
Logic:	<0.001 W	(17%)
Device Static:	0.090 W	(99%)

7. Conclusions

The Timer during this work meets all the requirements. The simulation of the planned test cases proved that the VHDL code of the device works correctly, the behaviour of the system is as expected. After the automatic implementation stage, we can assert that the final product is a robust, low- power consuming device.