Bachelor of Technology

in

Electronics and Communication Engineering by

Name: - Arpit Jain
Roll No: 23UEC523

Course Coordinator

Dr. Nikhil Raj

Topic: -Frequency divider using T flip-flops, where the input clock frequency is divided by 2^N, with N being the number of flip-flops on FPGA

Department of Electronics and Communication Engineering The LNM Institute of Information Technology, Jaipur

November 2024

CONTENTS

AIM OF THE PROJECT :-

Implement a frequency divider using T flip-flops, where the input clock frequency is divided by 2^N, with N being the number of flip-flops.

INTRODUCTION :-

What is T Flip Flop ?

- T flip flop or to be more precise is known as Toggle Flip Flop because it is capable of toggling its output based on the input.
- T there, stands for Toggle.
- Toggle basically means that the bit will flip that is either from 1 to 0 or from 0 to 1.

In this design we are using T Flip Flops to divide the input clock frequency by 2, and every subsequent division is performed by another T flip flop in series.

If we chain together in series, two T-type flip-flops the input frequency first will get "divided-by-two" by the first flip-flop $f \div 2$ and then "divided-by-two" again by the second flip-flop $\div 2$, giving an output frequency which has effectively been divided four times, then its output frequency becomes one quarter value 25% of the original clock frequency, ($f \div 4$).

Each time we add yet another toggle or "T-type" flip-flop to the chain, the output clock frequency is halved or divided-by-2 again and so on, giving an output frequency of 2n where "n" is the number of flip-flops used in the sequence. The T flip flop requires 2 clock cycle for toggling 1 for high and other for low. That’s why this property of toggling of T flip flops helps to divide frequency by factor of 2.

So we can now see that the output from the T-type flip-flop is at half the frequency of the input, in other words it counts in 2's. By cascading together more Toggle FlipFlops, we can produce a divide-by-2, divide-by-4, divideby-8, etc. circuit which will divide the input clock :-frequency by 2, 4 or 8 times, in fact any value to the powerof2 we want making a binary counter circuit.

CODE FOR THE PROJECT :-

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FrequencyDivider is     Generic (N : integer := 3);
    Port ( clk : in STD_LOGIC;
           T   : in STD_LOGIC;
           Y   : inout STD_LOGIC_VECTOR(N-1 downto 0) := (others => '0'));
end FrequencyDivider;

architecture Behavioral of FrequencyDivider is   signal Q :
STD_LOGIC_VECTOR(N-1 downto 0) := (others => '0'); begin

    process(clk,T)     begin
if rising_edge(clk) then
if T = '1' then
                Q(0) <= not Q(0); -- First flip-flop toggles with each clock
pulse                              for i in 1 to N-1 loop
Q(i) <= Q(i) xor Q(i-1);                end loop;                        end if;
end if;          end process;
 Y <= Q;

end Behavioral;
```

SCHEMATIC DIAGRAM :-

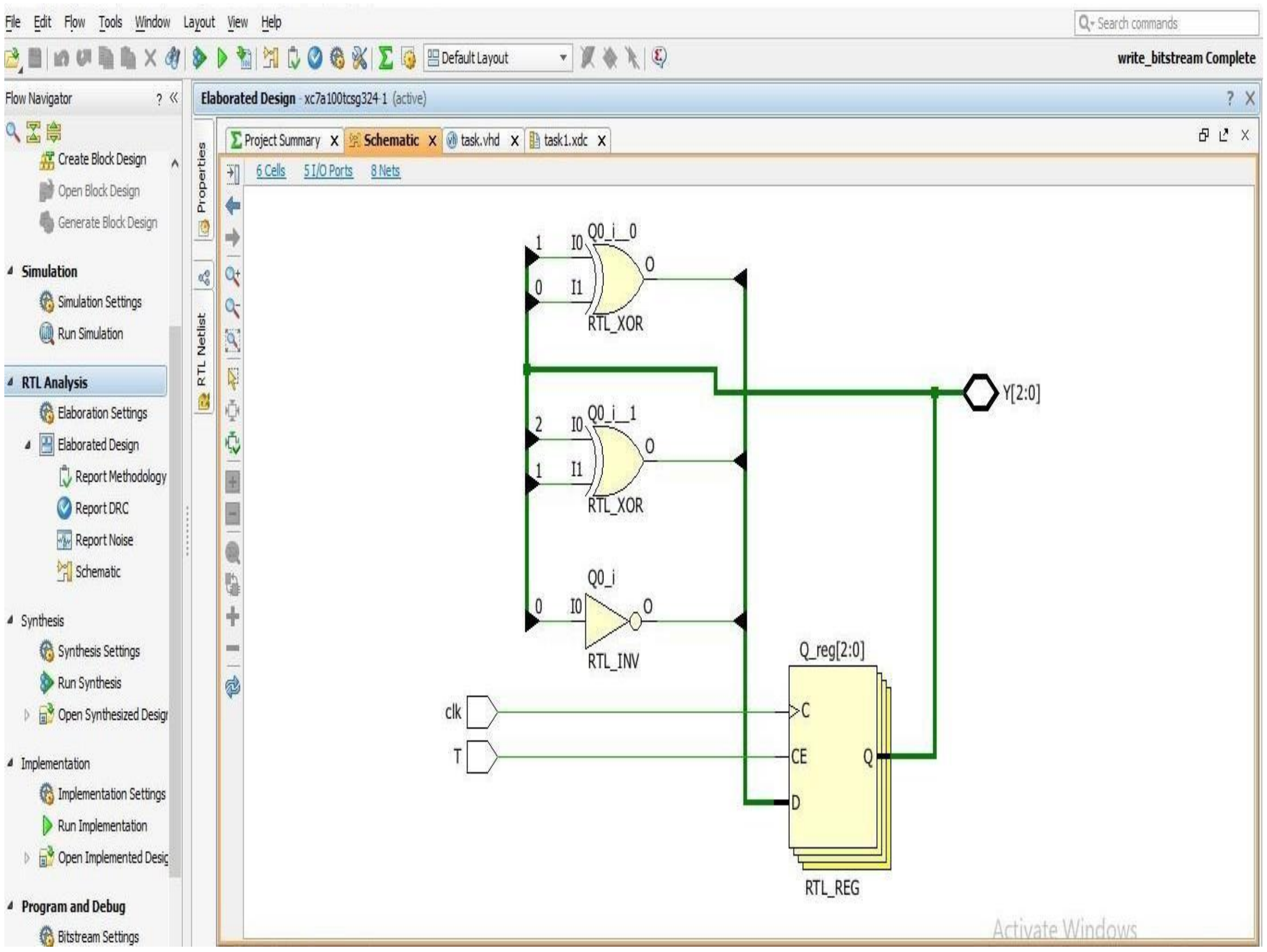
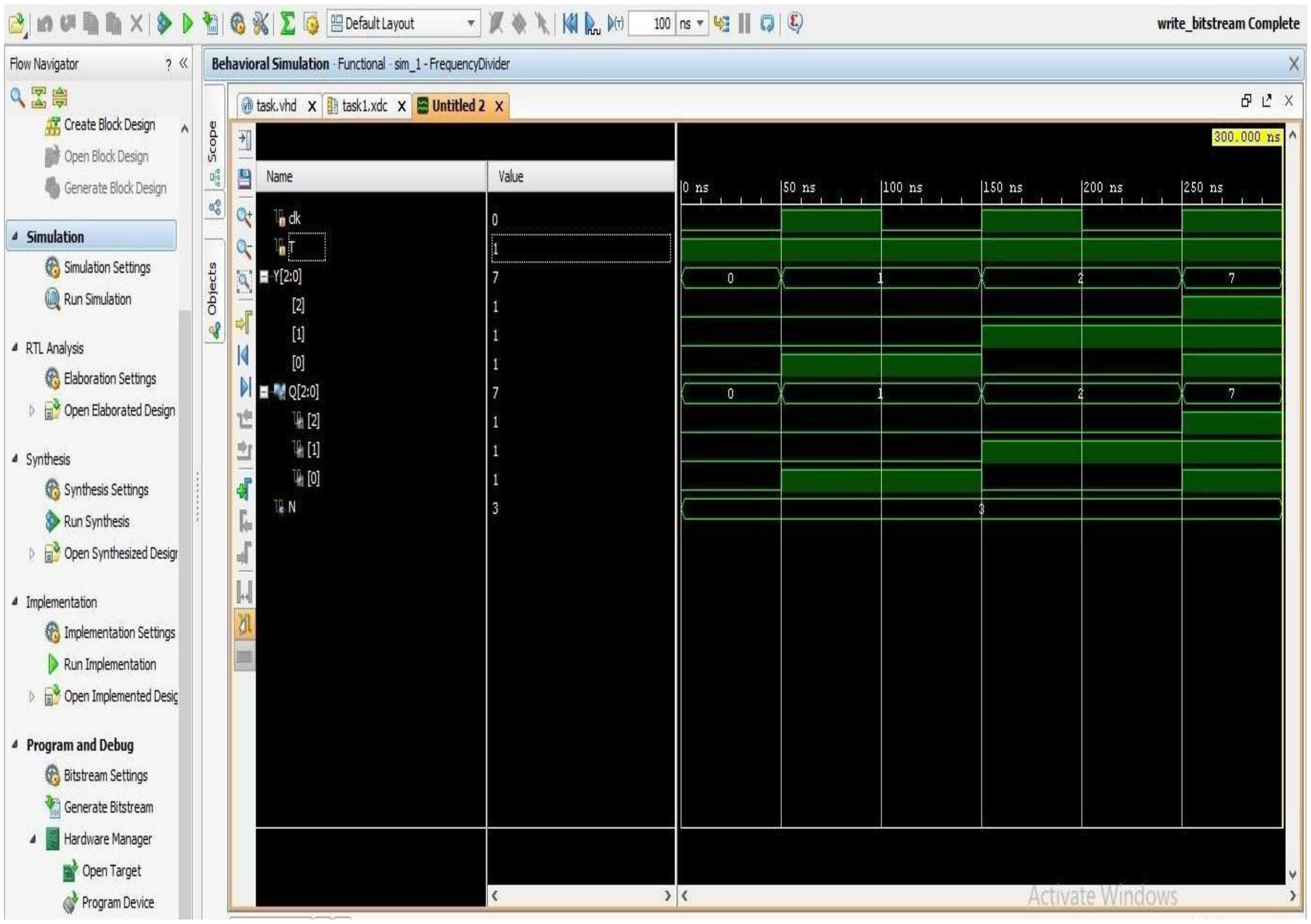


FIGURE 1 : Schematic diagram for frequency divider using T flip flops

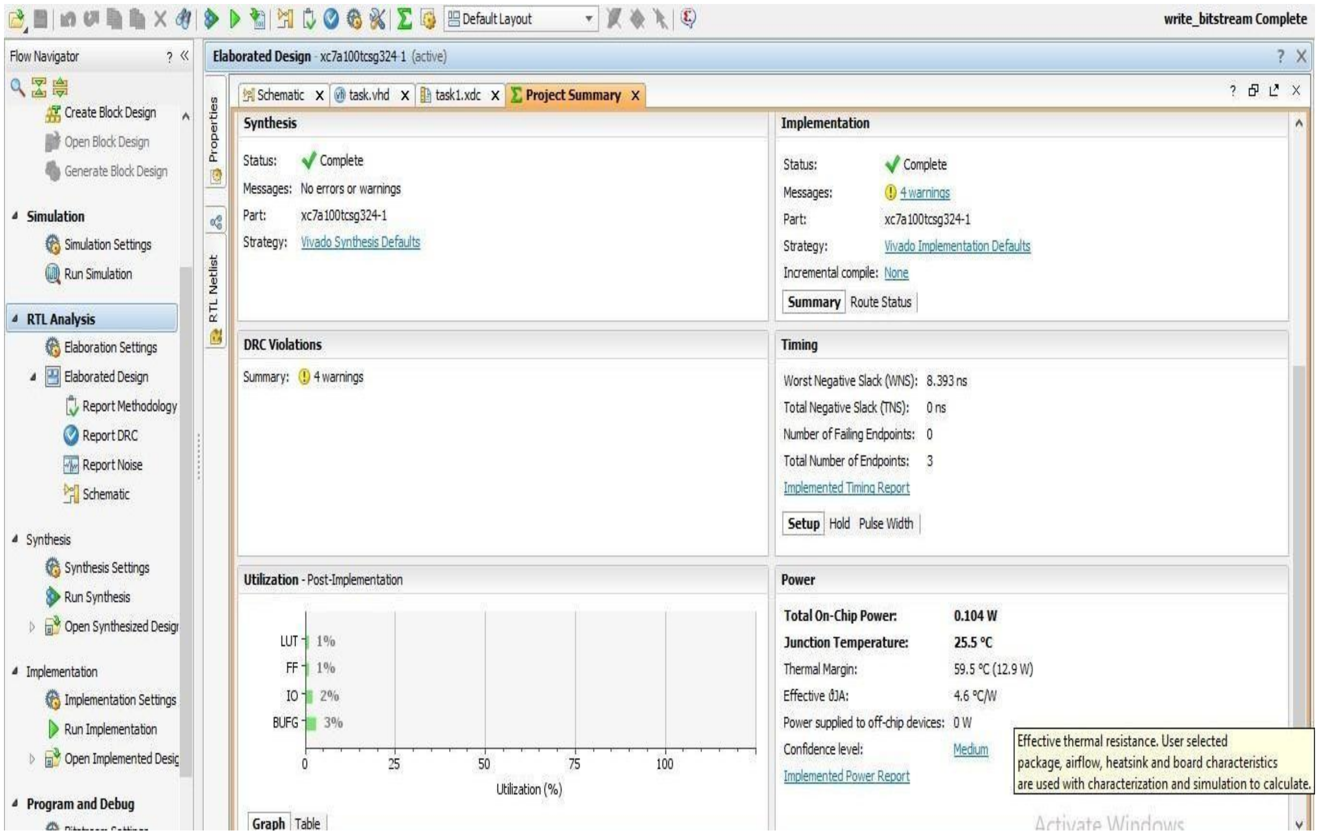
<u>SIMULATION TIMING DIAGRAM</u> :-



Name	Value
clk	0
T	1
Y[2:0]	7
[2]	1
[1]	1
[0]	1
Q[2:0]	7
[2]	1
[1]	1
[0]	1
N	3

FIGURE 2 Simulation diagram for frequency divider using T flip flops

Project Summary:


Default Layout

write_bitstream Complete

Flow Navigator

Create Block Design
Open Block Design
Generate Block Design

Simulation
Simulation Settings
Run Simulation

RTL Analysis
Elaboration Settings
Elaborated Design
Report Methodology
Report DRC
Report Noise
Schematic

Synthesis
Synthesis Settings
Run Synthesis
Open Synthesized Design

Implementation
Implementation Settings
Run Implementation
Open Implemented Design

Program and Debug

Elaborated Design - xc7a100tcsg324-1 (active)

Schematic | task.vhd | task1.xdc | Project Summary

Synthesis

Status: ✓ Complete
Messages: No errors or warnings
Part: xc7a100tcsg324-1
Strategy: Vivado Synthesis Defaults

Implementation

Status: ✓ Complete
Messages: 4 warnings
Part: xc7a100tcsg324-1
Strategy: Vivado Implementation Defaults
Incremental compile: None

Summary | Route Status

DRC Violations

Summary: 4 warnings

Timing

Worst Negative Slack (WNS): 8.393 ns
Total Negative Slack (TNS): 0 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 3
Implemented Timing Report

Setup | Hold | Pulse Width

Utilization - Post-Implementation

LUT	1%
FF	1%
IO	2%
BUFG	3%

Utilization (%)

Graph | Table

Power

Total On-Chip Power: 0.104 W
Junction Temperature: 25.5 °C
Thermal Margin: 59.5 °C (12.9 W)
Effective ϑJA: 4.6 °C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
Implemented Power Report

Effective thermal resistance. User selected package, airflow, heatsink and board characteristics are used with characterization and simulation to calculate.

Figure 3 Project summary for frequency divider using T flip flops

<u>CONSTRAINT FILE</u> :-

Clock signal set_property -dict { PACKAGE_PIN E3   IOSTANDARD LVCMOS33 } [get_ports {clk }];

#IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk }];
##Switches set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { T}]; #IO_L24N_T3_RS0_15 Sch=sw[0]

LEDs
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { Y[0] }];
#IO_L24P_T3_RS1_15 Sch=led[1]

set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { Y[1] }];

#IO_L17N_T2_A25_15 Sch=led[2] set_property -dict { PACKAGE_PIN N14
IOSTANDARD LVCMOS33 } [get_ports { Y[2] }]; #IO_L8P_T1_D11_14 Sch=led[3]

FPGA BOARD IMPLEMENTATION :-

- NEXYS 4 DDR

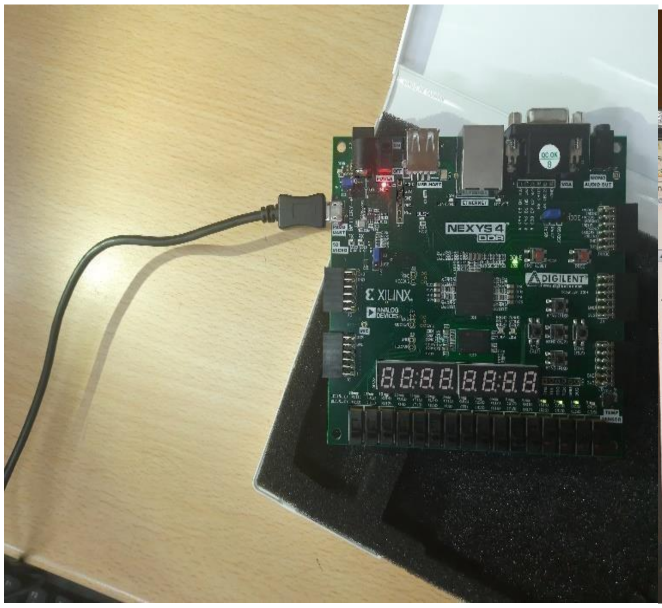


FIGURE 4

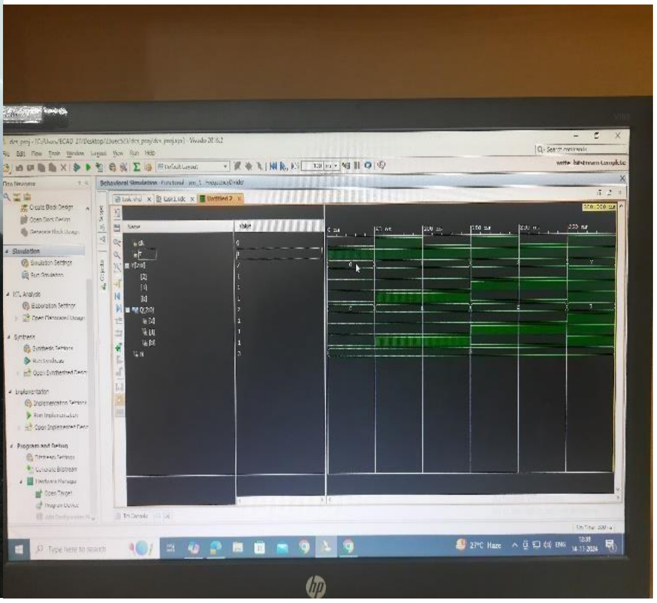


FIGURE 5

SUMMARY :-

In the project, a frequency divider was designed and successfully implemented using N = 3 T flip-flops- a division that would reduce the input clock frequency by 2^N. The design utilized the toggling feature of T flip-flops whereby the cascading arrangement allowed progressive division of the clock frequency.

The VHDL implementation demonstrated the flexibility of the design, as depicted by the parameterized N, which made the design applicable to any divisor factor. It also works efficiently to give a stable output clock signal that satisfies several demands of digital systems.

This frequency divider is a basic building block in any application involving clock management, timing control, or frequency synthesis. The project really brings out the simplicity and versatility of the use of T flip-flops for frequency division and, as such, is a worthy addition to digital circuit design.

We are using NEXYS 4 DDR FPGA (Field Programmable Gate Array) to implement the frequency divider circuit. Configuring and reconfiguring on the same device can be done as many times as we want. At every input, we can generate bit-stream according to code and implement that on board.