2025 SPRING ELEC-650
ADVANCED DIGITAL SYSTEM DESIGN
TERM PROJECT / ADJUSTABLE PWM GENERATOR

1. PROJECT STRUCTURE

1.1 Top Level:

I've got two (2) sub-modules which are PWM generator and debouncing filter that are connected at top module. This module implements an adjustable PWM generator. We've got control over frequency and duty time also over the mode-selection that manages the mode switch between center-aligned and edge-aligned.

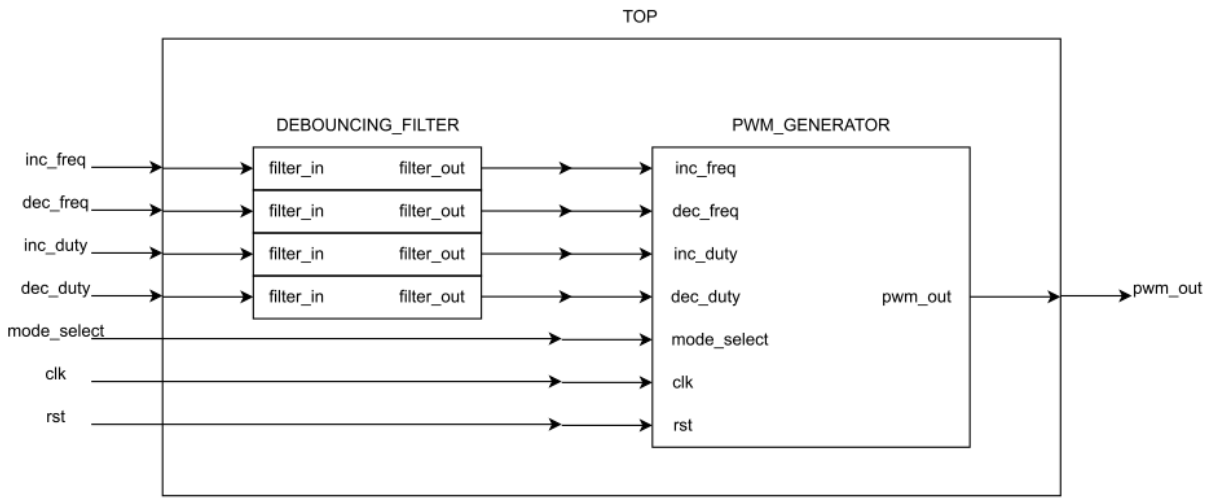


Figure 1: Top Design

Also debouncing filter has "clk"and "rst" pins, however, they are common with the top design's so I've not mentioned them at the Figure 1.

1.2 PWM Generator:

PWM generator allows us to generate different types of square wave using our FPGA's clock. Additionally, we are able to tune the duty time, frequency and mode.

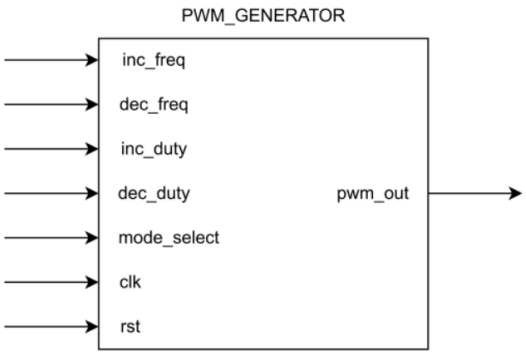


Figure 2: PWM Generator

In this design, I've used ACTIVE and DUTY_PERCENTAGE registers to change the duty time. Duty percantage changes with ±5 and re-assigns the active register for the counter

A a result of scaling problems, I could not decided for a frequency step, it doesn't change linear. So I've decided to create a look-up-table with some period values. After this I was able to reach 100KHz step size by only changing the index of look-up-table.

Specifications of PWM generator:

- **inc_duty / dec_duty:** Changes the duty time duration by changing ACTIVE register and step size is 5%.
- **inc_freq / dec_freq:** Manages the changes with selecting period values from look-up-table, and step size is approximately 100 KHz. (When we reach the higher frequencies, our sensitivity faces with some scaling issues due to that the changes become roughly 100KHz).
- **mode_select:** 0: edge-aligned / 1: center-aligned
- **rst:** The design uses a common reset signal for each module, which drives the outputs to '0'.
- **clk:** This one is also common for the system, and I've choosen 40ns for the operating period.

1.3 Debouncing Filter:

In physical implementation, we encounter some input glitches caused by push-buttons. Therefore, we typically use a debouncing filter on the inputs to filter this noise.

However, since simulations are not in the physical domain, we are simulating in Vivado, the top-level design simulation does not reflect real-world behavior.

For instance, when we intend to increment only once, the filter prolongs the signal duration, causing multiple increments.

Therefore, I do not recommend relying on the top-level simulation for precise results.

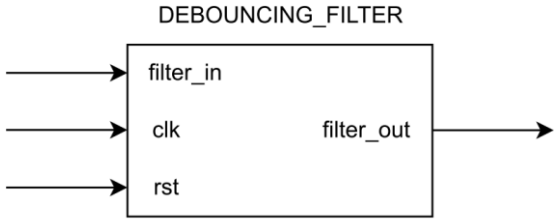


Figure 3: Debouncing Filter

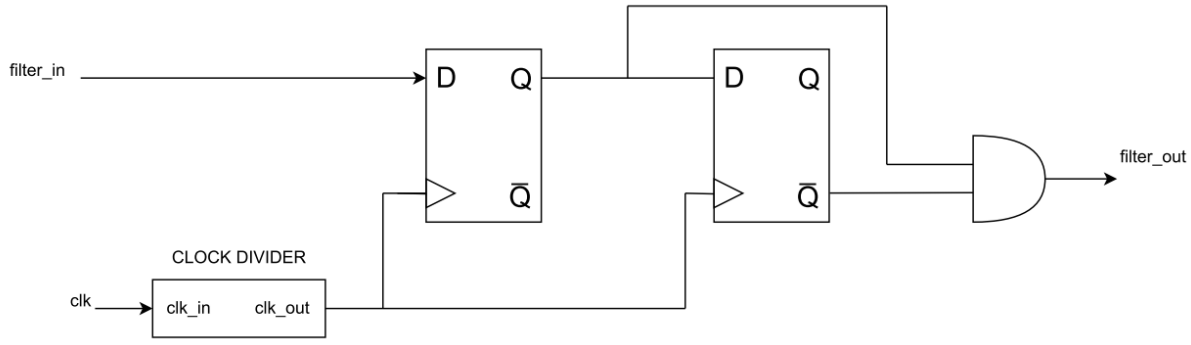


Figure 4: Debouncing Filter Circuit

2. VIVADO SCHEMATICS

2.1 Top Level:

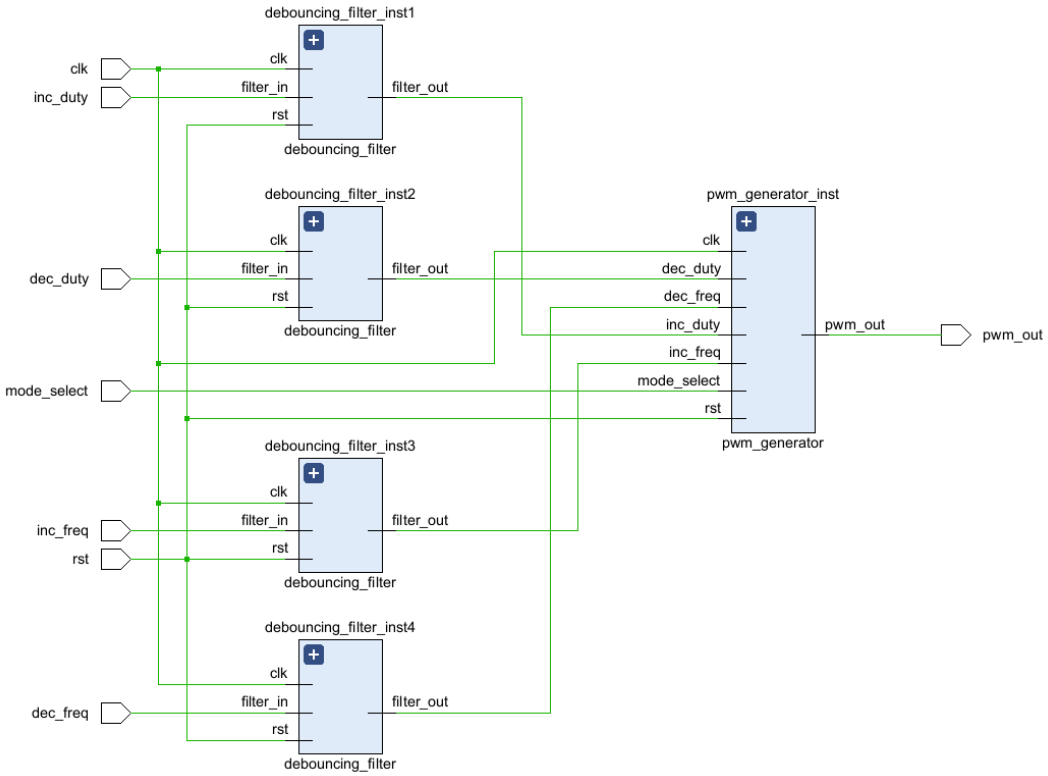


Figure 5: Top Level Schematic

2.2 Debouncing Filter:

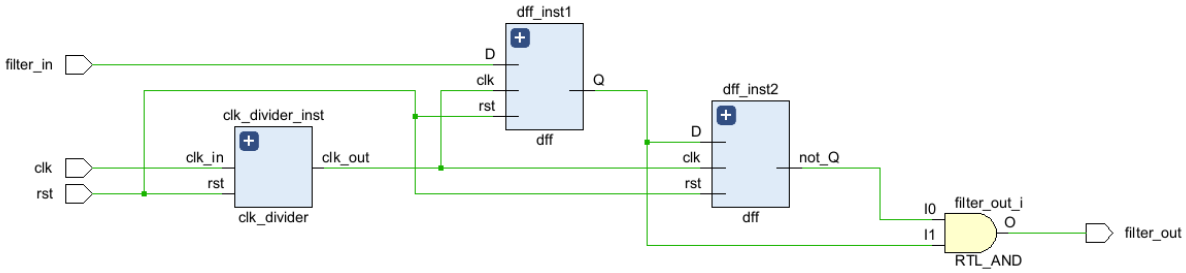


Figure 6: Debouncing Filter Schematic

2.3 PWM Generator:

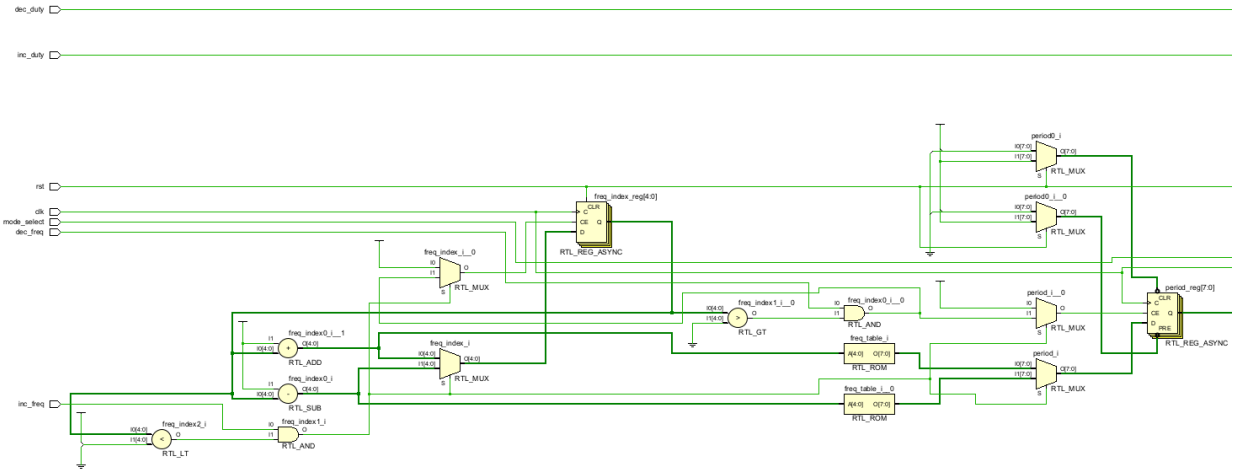


Figure 7: PWM Generator (1st Part)

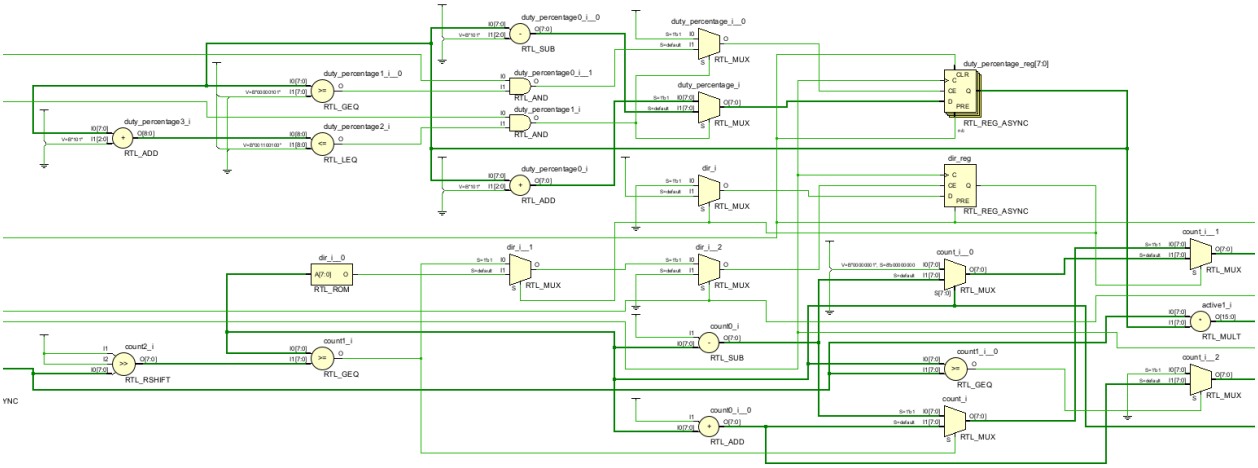


Figure 8: PWM Generator (2nd Part)

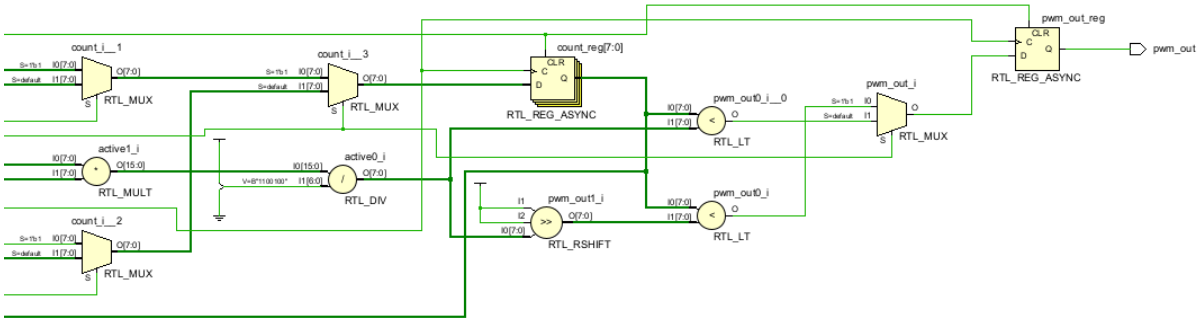


Figure 9: PWM Generator (3rd Part)

3. SIMULATIONS

First of all, I've used several tasks in the testbenches to improve readability, accelerate the verification process, and keep the code clean. Below is a explanation of these tasks:

- **delay(integer)** : Adds a delay based on the specified number of clock cycles.
- **up_duty() :** Generates a one-clock-cycle pulse to increase the duty cycle.
- **down_duty() :** Generates a one-clock-cycle pulse to decrease the duty cycle.
- **up_freq():** Generates a one-clock-cycle pulse to increase the frequency.
- **down_freq():** Generates a one-clock-cycle pulse to decrease the frequency.
- **reset(integer):** Sends a reset signal for the specified number of clock cycles.

Secondly, I have prepared several separate testbenches and wave config files to verify each module and its functionality individually. Also , I've created a top-level testbench to observe the overall functionality of the system.

Additionally, I've choosen clock period for 40 ns to get 100 kHz when the period equals to the highest.

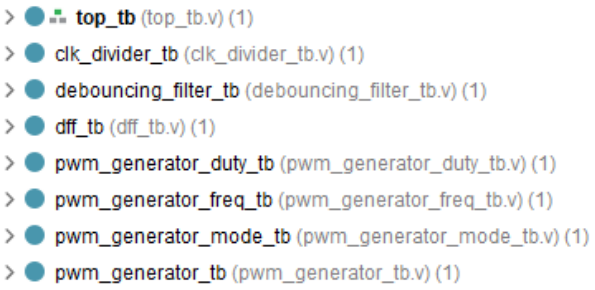


Figure 10: List of Testbenches

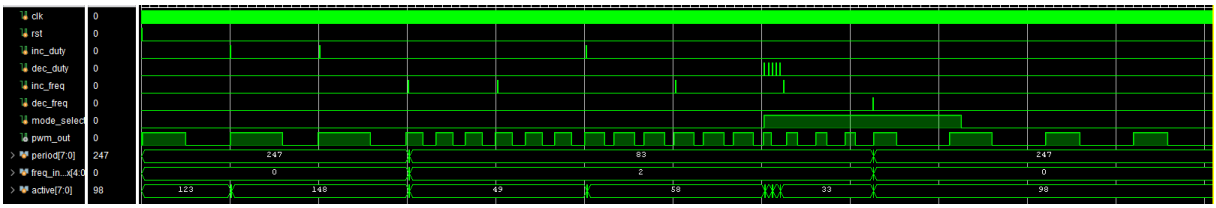
3.1 Top-Level:

File: top_tb.v

Tcl commands after run:
-> open_wave_config ./wave_outputs/top_tb_behav.wcfg

-> relaunch_sim

-> run 120 us



As I mentioned before, we can clearly see the issue that debouncing filters caused in this simulation. When we sent a one-clock-cyle signal, it just expands it and the system increases/decreases multiple times and skips the periods or duty percentages.

3.2 PWM Generator:

I’ve got four (4) seperated testbenches for this module, and I’ll explain them in a order.

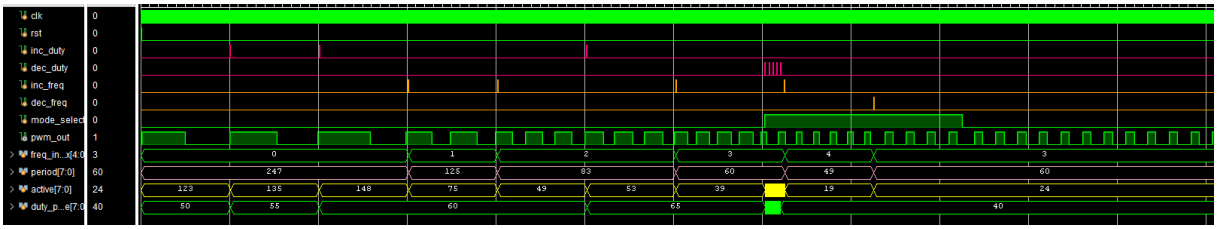
File: pwm_generator_tb.v

Tcl commands after run:
-> open_wave_config ./wave_outputs/pwm_generator_tb_behav.wcfg

-> relaunch_sim

-> run 120 us

(This one is better to simulate than the previous top_tb.v to see the functionality)

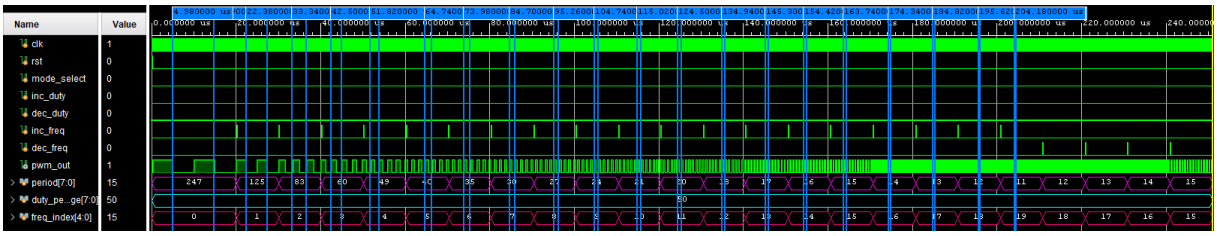


File: pwm_generator_freq_tb.v

Tcl commands after run:
-> open_wave_config ./wave_outputs/pwm_generator_freq_tb_behav.wcfg

-> relaunch_sim

-> run 1000 us

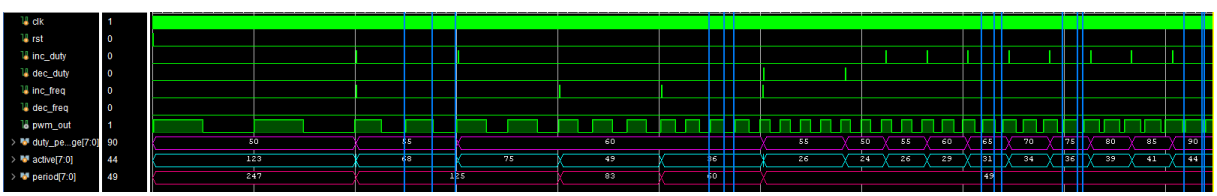


File: pwm_generator_duty_tb.v

Tcl commands after run:
-> open_wave_config ./wave_outputs/pwm_generator_duty_tb_behav.wcfg

-> relaunch_sim

-> run 1000 us

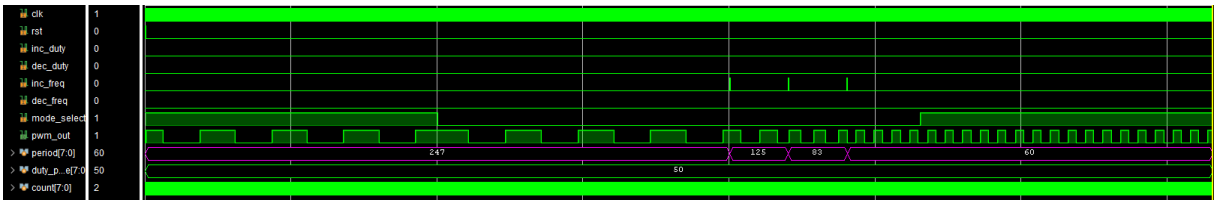


File: pwm_generator_mode_tb.v

Tcl commands after run:
-> open_wave_config ./wave_outputs/pwm_generator_mode_tb_behav.wcfg

-> relaunch_sim

-> run 40 us



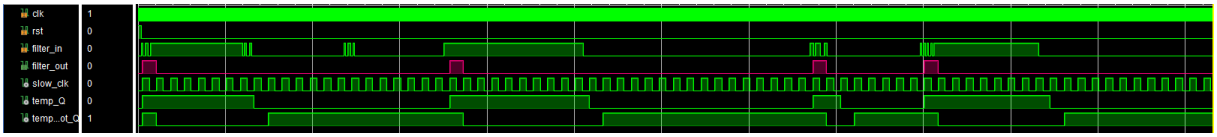
3.3 Debouncing Filter:

File: debouncing_filter_tb.v

Tcl commands after run:
-> open_wave_config ./wave_outputs/debouncing_filter_tb_behav.wcfg

-> relaunch_sim

-> run 10 us



4. UTILIZATION AND POWER REPORTS

4.1 Utilization Report:

Name	Slice LUTs (20800)	Slice Registers (41600)	Slice (8150)	LUT as Logic (20800)	Bonded IOB (106)	BUFGCTRL (32)
N top	241	47	78	241	8	1
debouncing_filter_inst1 (debouncing_filter)	2	4	3	2	0	0
clk_divider_inst (clk_divider_10)	1	2	2	1	0	0
dff_inst1 (dff_11)	1	1	1	1	0	0
dff_inst2 (dff_12)	0	1	1	0	0	0
debouncing_filter_inst2 (debouncing_filter_0)	2	4	3	2	0	0
clk_divider_inst (clk_divider_7)	1	2	2	1	0	0
dff_inst1 (dff_8)	1	1	1	1	0	0
dff_inst2 (dff_9)	0	1	1	0	0	0
debouncing_filter_inst3 (debouncing_filter_1)	2	4	3	2	0	0
clk_divider_inst (clk_divider_4)	1	2	2	1	0	0
dff_inst1 (dff_5)	1	1	1	1	0	0
dff_inst2 (dff_6)	0	1	1	0	0	0
debouncing_filter_inst4 (debouncing_filter_2)	3	4	4	3	0	0
clk_divider_inst (clk_divider)	2	2	3	2	0	0
dff_inst1 (dff)	1	1	1	1	0	0
dff_inst2 (dff_3)	0	1	1	0	0	0
pwm_generator_inst (pwm_generator)	232	31	72	232	0	0

4.2 Power Reports:

Utilization	Name	Clocks (W)	Signals (W)	Data (W)	Clock Enable (W)	Logic (W)	I/O (W)
0.002 W (3% of total)	N top						
0.001 W (2% of total)	pwm_generator_inst (pwm_generator)	<0.001	<0.001	<0.001	<0.001	0.001	<0.001
0.001 W (1% of total)	Leaf Cells (9)						
<0.001 W (<1% of total)	debouncing_filter_inst2 (debouncing_filter_0)	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
<0.001 W (<1% of total)	debouncing_filter_inst4 (debouncing_filter_2)	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
<0.001 W (<1% of total)	debouncing_filter_inst1 (debouncing_filter)	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001
<0.001 W (<1% of total)	debouncing_filter_inst3 (debouncing_filter_1)	<0.001	<0.001	<0.001	<0.001	<0.001	<0.001

Summary

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

Total On-Chip Power:	**0.074 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.4°C**
Thermal Margin:	59.6°C (11.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity