Clock glitcher

The principal idea on this project is to investigate the behavior of high-level C functions running on a processor based on the RISC-V (ISA) microarchitecture under fault injection. In the second part of this report, the normal behavior of low-level C functions is discussed. In this section, A fault injector is introduced in the RISC-V processor to analyze and evaluate the behavior of these functions. For this purpose, To fault injection into the target system, we used a special type of physical attack known as clock glitching.

Structure of the fault injector (glitch generator) is shown in Figur.1. In the figure, structure of fault injection attack (clock glitch attack) is Shown. Generally, it can be divided into two general parts; Target and fault injector.

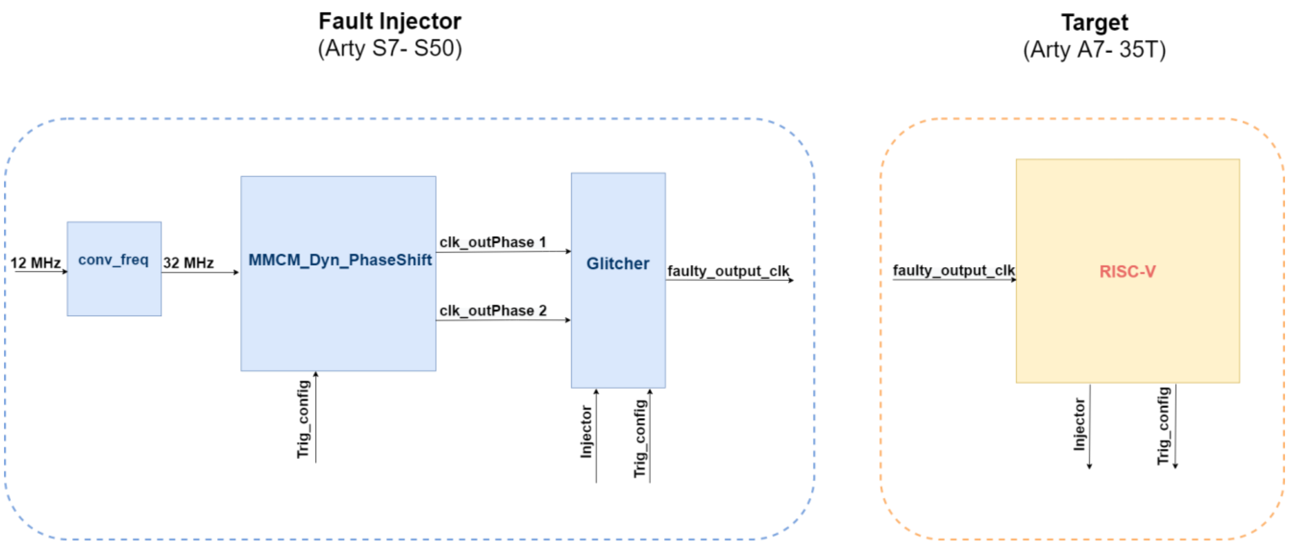


Figure. 1: General structure of the fault injector

1. Target

Since our goal is to examine the commonly used high-level C functions on RISC-based processors, So we implemented a RISC-V based processor on the Arty XC7A35T board. Then, for the clock glitching operation first, contents of the XDC file were manipulated so that the chip used an external clock, then a fault injection attack was performed with a glitch generator design that insert a faulty input clock.
The "faulty_output_clk" signal shown in the figure. 1 is the faulty clock, Which is connected to the RISC-V external clock pin by a fault injector. The two output signals("injector" and "Trig_config") are also described below.

2. Fault injector

In order to fault inject on RISC-V, this section provides a complete explanation of the clock glitch generator structure. For this purpose to implement the fault injector, board Arty xc7s50 has been used.The fault generator consists of three general parts and a number of input and output signals; in the following, we will examine the detailed structure of these parts:

2.1. Conv_freq

The desired clock frequency for the target device is 32Mhz. On the other hand, two clocks with frequencies of 100Mhz and 12Mhz are available in FPGA. Therefore, in the first step, the frequency of 12Mhz, which is provided by the FPGA, is converted to frequency 32Mhz by an MMCM IP in the "conv_freq" entity.

2.2. MMCM_Dyn_PhaseShift

Since our goal in fault injector is to insert glitches with different phases and widths on the output of the circuit ("faulty_output_clk" signal), This module produces two signals with an equal frequency of 32MHz and different phases at the output. These two signals are "CLK_shifted_Phase1" and "CLK_shifted_Phase2" in Fig. 3 which are shifted to phase1 and phase2 value, respectively. The schematic of "MMCM_Dyn_PhaseShift" entity can be seen in Fig. 3.

2.2.1. Phase_machine entity

In the "Phase_machine" entity in Fig. 3, the values of phase1 and phase2 are changing under 4005 different states whenever a rising edge is applied to the "trig_config" input. To be more specific, for each rising edge on "trig_config", the phases are configured to a new state.
In each state, appropriate controlling signals are prepared in the "phase_shift_controller" modules for the MMCM2 and MMCM3 IPs and are applied to dynamic phase shifting ports of these IPs. In this way, two shifted signals with the frequency of 32 MHz appear in the "CLK_shifted_phase1" and "CLK_shifted_phase2" outputs.
This unit is written as a state machine and determines the appropriate phase value for "clk_out1" and "clk_out2" when a rising edge is observed in "trig_config" signal (a new configuration for new state), and then these phase values are entered to two "Phase_shift_controller" modules. Initially, if the reset = `1`, the machine is on state1. When the reset = `0`, the machine enters state2 and the value of phase1 is equal to 1 degree and the value of phase2 is equal to 2 degrees(it is one degree more than phase 1; this phase difference leads to a glitch with a width of 1 degree).

Then, in the state3 the phase2 value is checked to be less than 91 degrees, if this condition is met, the machine enters State4 and stays there until a rising edge is applied to "trig_config".
Once this edge is detected, a new state (or a new configuration in other words) occurs by entering state5 and increasing phase2 by one degree (according to the attached table).
After that, the condition of phase2 in state 3 is checked again and then stays in State4 until the next rising edge is seen on "trig_config". This cycle continues between states 3, 4, and 5 until phase2 reaches 90, and then by entering State 6, the phase1 value is checked to be less than 90 degrees; If this condition is met, phase1 increases to 1 more degree. When the overall cycle for phase1 to 89 degrees is done, The machine enters state7 and the phases are set to zero, and then, the machine resumes the previous routine. Fig. 2 shows the state machine on the "Phase_machine" entity.

PHASE1	***1***	***2***	***3***	***4***	***...***	***87***	***88***	***89***
PHASE2	*2*	*3*	*4*	*5*	*...*	*88*	*89*	*90*
	3	*4*	*5*	*6*	*...*	*89*	*90*	
	4	*5*	*6*	*7*	*...*	*90*		
	...	...	...	...	...			
	86	*87*	*88*	*89*				
	87	*88*	*89*	*90*				
	88	*89*	*90*					
	89	*90*						
	90							

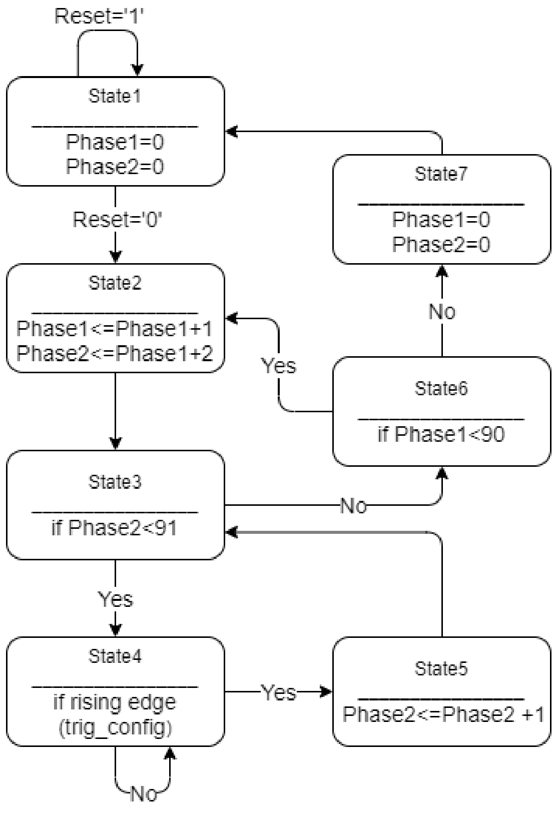


Figure. 2: Unit diagram Phase_machine

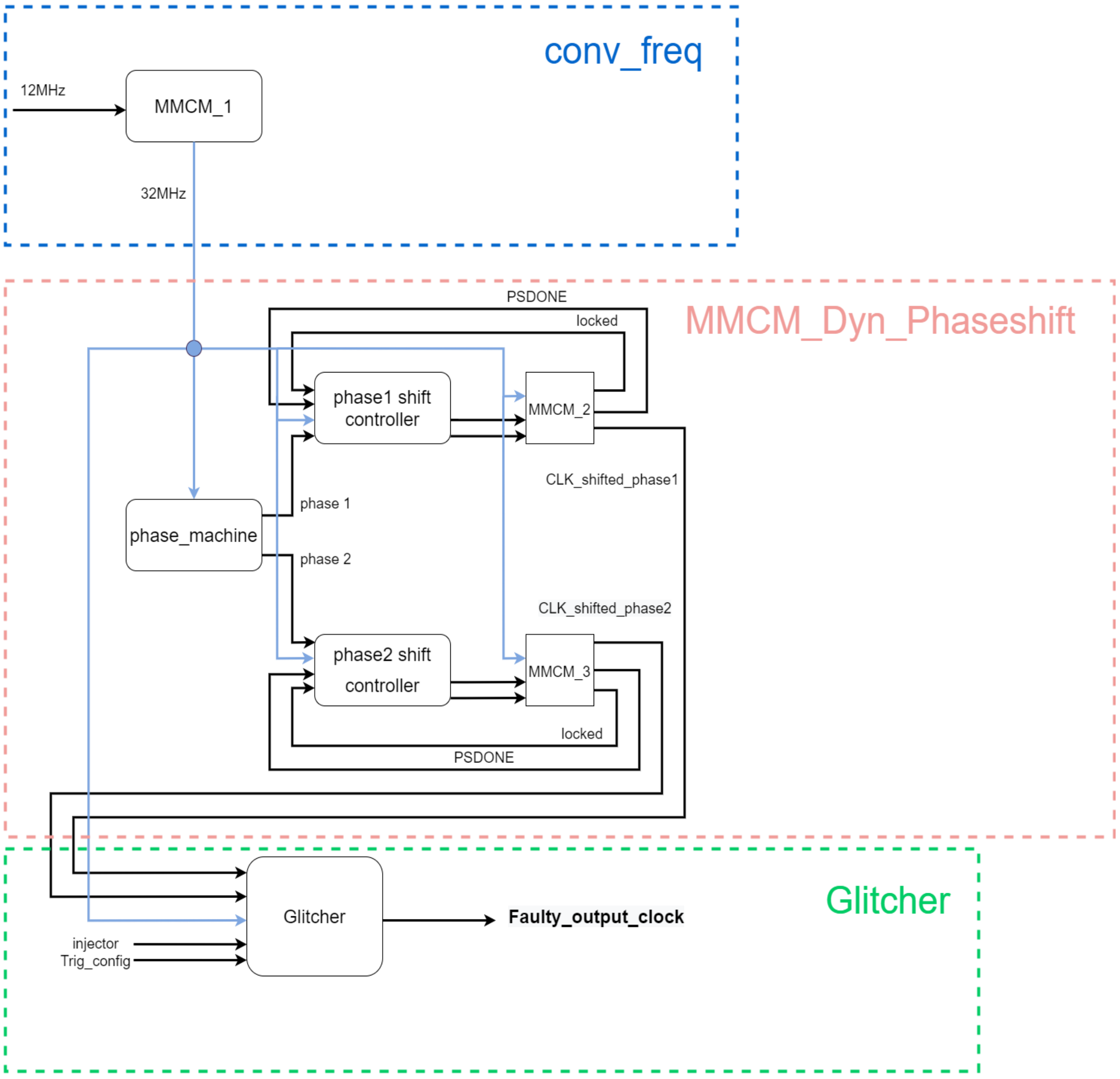


Figure. 3: Schematic of Fault Injector

2.3.Glitcher

In the "Glitcher" entity, the phase difference of the shifted clocks given to the input is extracted. Then, the glitches are generated in such a way that the width of the glitch is equal to the difference between "CLK_shifted_phase1" and "CLK_shifted_phase2" (phase1 and phase2 differences), its phase is equal to the phase of “CLK_shifted_phase1”. By applying a rising edge on injector input in this unit, the generated glitch is applied to the output clock signal, and this signal becomes faulty.

This module is written using the state machine, the diagram of which is shown in Figure 4.

According to the state machine, first it is in idle state, then with the occurrence rising edge of the injector signal (injector_edge = 1) it goes to count state. In count state, the generated glitch is inserted on the “faulty_output_clk” . After that, it updates the value of “sig_var” each time it sees a rising edge on the “injector signal” (injector_edge = 1). As long as it is in “count” state if Trig_edge = 1 occurs it goes to the initial state, this means that the glitch must be generated with new configuration. Figure. 5 shows the implementation of “Glitcher” Unit, which is implemented in the vhdl programming language.

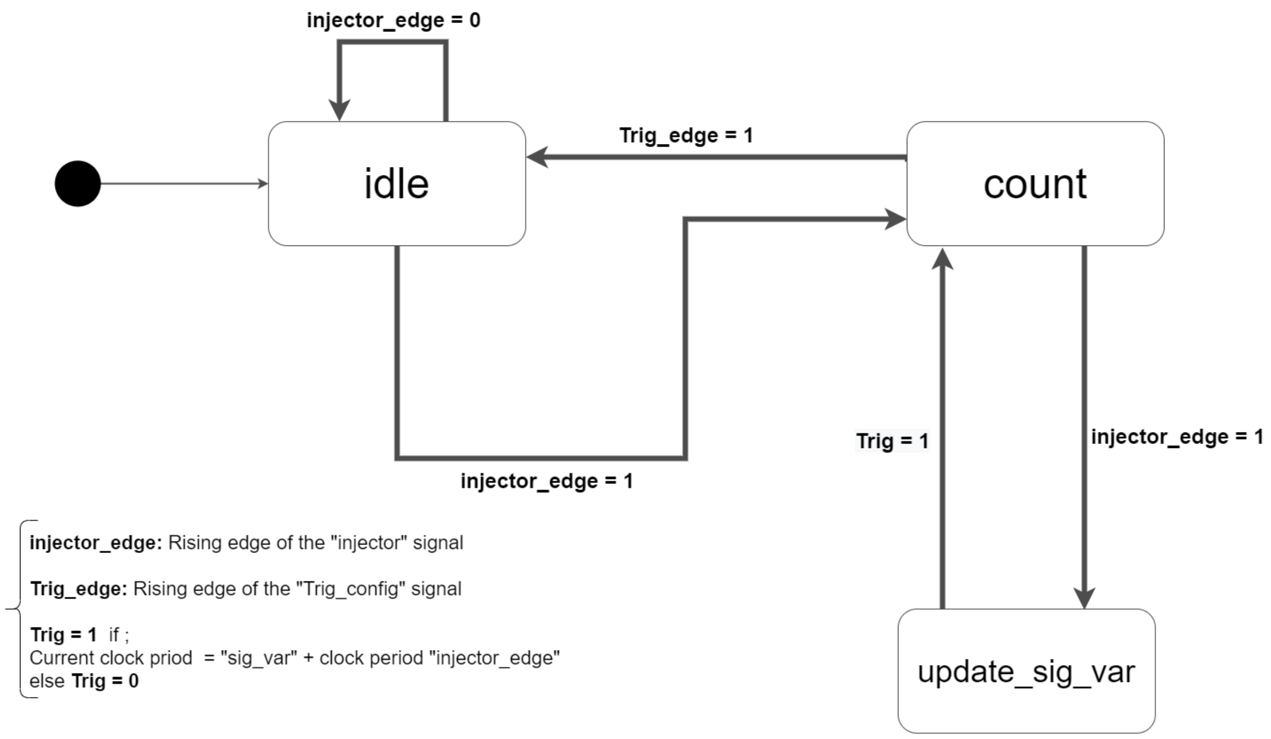
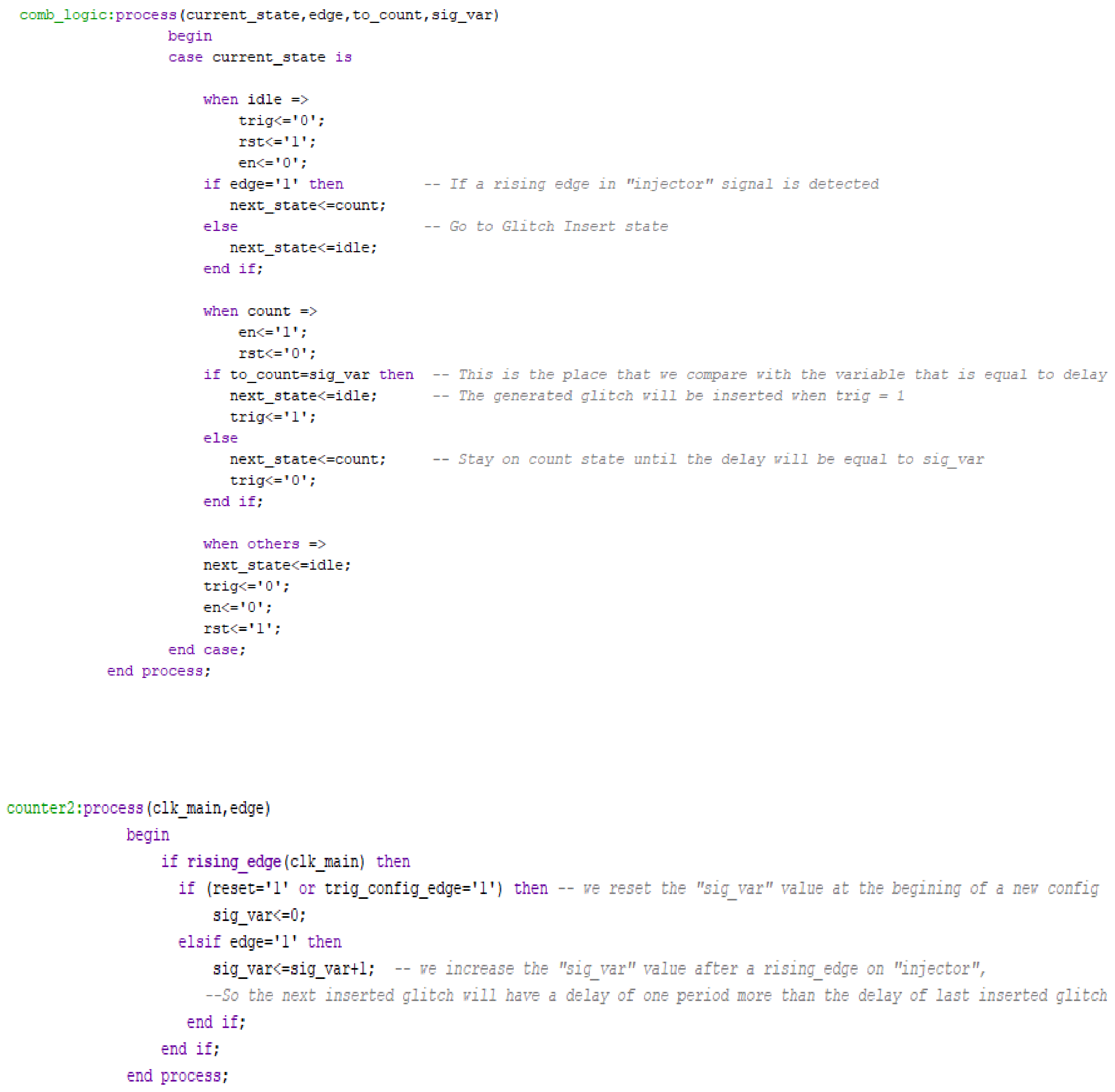

injector_edge = 0
idle
Trig_edge = 1
count
injector_edge = 1
Trig = 1
injector_edge = 1
update_sig_var
injector_edge: Rising edge of the "injector" signal
Trig_edge: Rising edge of the "Trig_config" signal
Trig = 1 if ;
Current clock priod = "sig_var" + clock period "injector_edge"
else Trig = 0

Figure. 4: Unit diagram glitch



```vhdl
comb_logic:process(current_state,edge,to_count,sig_var)
             begin
             case current_state is

                when idle =>
                    trig<='0';
                    rst<='1';
                    en<='0';
                if edge='1' then         -- If a rising edge in "injector" signal is detected
                   next_state<=count;
                else                     -- Go to Glitch Insert state
                   next_state<=idle;
                end if;

                when count =>
                    en<='1';
                    rst<='0';
                if to_count=sig_var then  -- This is the place that we compare with the variable that is equal to delay
                   next_state<=idle;      -- The generated glitch will be inserted when trig = 1
                   trig<='1';
                else
                   next_state<=count;     -- Stay on count state until the delay will be equal to sig_var
                   trig<='0';
                end if;

                when others =>
                next_state<=idle;
                trig<='0';
                en<='0';
                rst<='1';
             end case;
       end process;
```

```vhdl
counter2:process(clk_main,edge)
            begin
                if rising_edge(clk_main) then
                  if (reset='1' or trig_config_edge='1') then -- we reset the "sig_var" value at the begining of a new config
                      sig_var<=0;
                  elsif edge='1' then
                      sig_var<=sig_var+1;  -- we increase the "sig_var" value after a rising_edge on "injector",
                     --So the next inserted glitch will have a delay of one period more than the delay of last inserted glitch
                   end if;
                end if;
            end process;
```

Figure. 5: VHDL code of Glitcher

2.4. Trig_config signal

By changing the “Trig_config” input as mentioned earlier the phases of Phase1 and Phase2 get a new value in new state, thus by changing the shift value of the “CLK_shifted_phase1” and “CLK_shifted_phase2” signals, the configuration of the produced glitches change.

By increasing phase1 from 1 degree to more degrees until 89 degree, and changing phase2 relative to it (each time the values of one unit more than phase1 to 90 degrees) according to the attached table in 4005 different states, 4005 different types of glitches can be generated on output clock(faulty_output_clk).

Thus, for each configuration that is created by “trig_config” signal, rising edges of the “injector” signal inserts glitches to the output clock signal after an offset period (The sig_var variable must be used to determine the offset, we will explain in the following). Each time the edge is created on trig_config signal, the configuration changes. then, by activating the injector, new glitch is inserts to the output clock.

2.5. Injector signal

This signal, which is set by the user from the target, by applying injector input to the glitcher unit, the generated glitch is applied to the output clock signal, and the output clock signal will be faulty. this signal is edge sensitive, Each time this signal is activated, a glitch is applied to the output clock(faulty_output_clock) signal.

In order to be able to apply the glitch with different offset, we will have the need for the sig_var signal, the value of this signal is increased by observing each edge of the injector. Therefore, as long as the value of Trig_config does not change, the distance between the glitch and the edge of the injector increases with each clock, this distance is called offset.

with each edge of the injector, until the Trig_config does not change, glitch with different offset and fixed width are applied to the output clock.

When Trig_config changes, it means that phase1 and phase2 change and that glitch is produced with a new width. until the value of Trig_config does not change, the glitch created with the same width is applied to the different offsets of the output clock each time the value of the injector is changed. Figure. 6 shows the injection of glitch in different offset, as shown in this figure, each time the injector edge is raised, the value of the “edge” signal is equal to 1. Then according to the value of “sig_var” glitch is inserted in different clock periods.

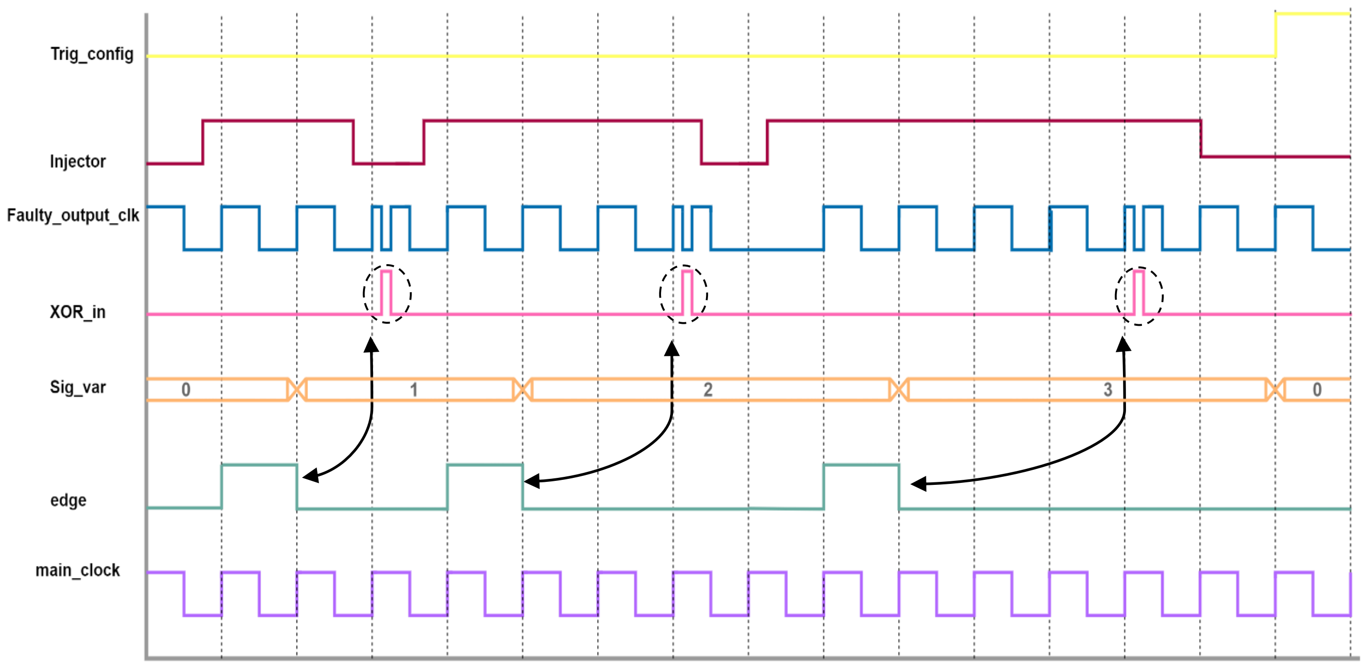
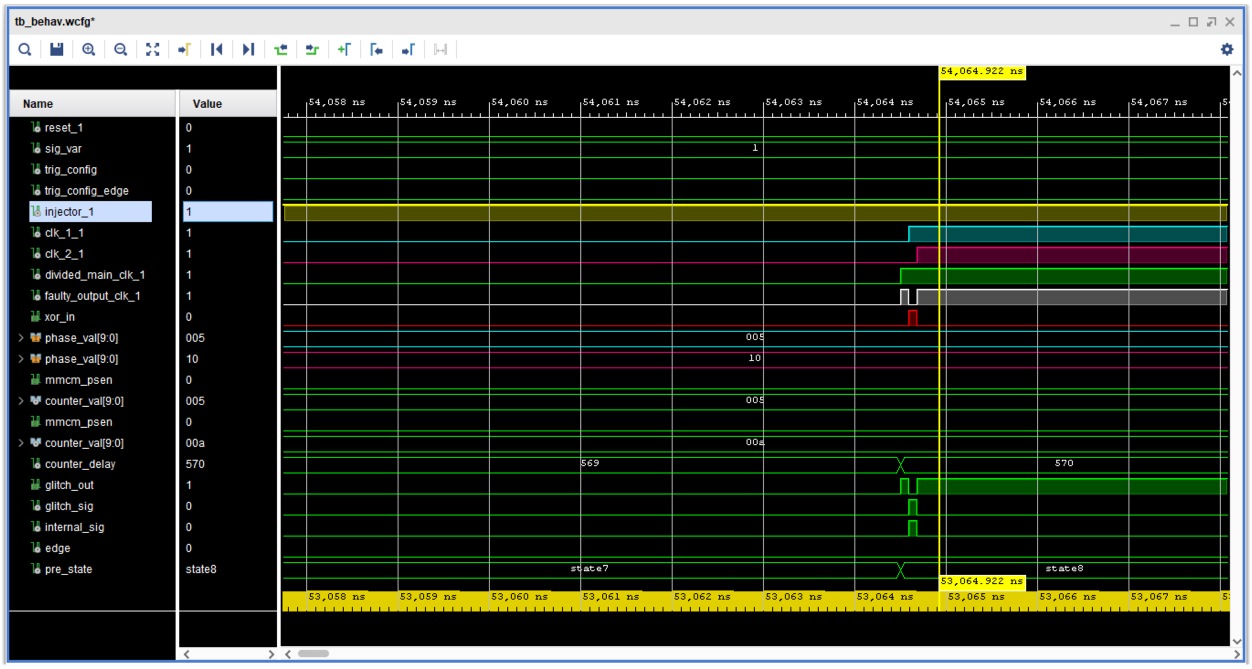


Figure. 6: Insertion of glitch in different offset

Simulation

In this section, the simulation results in the Vivado tool are shown :



Name	Value
reset_1	0
sig_var	1
trig_config	0
trig_config_edge	0
injector_1	1
clk_1_1	1
clk_2_1	1
divided_main_clk_1	1
faulty_output_clk_1	1
xor_in	0
phase_val[9:0]	005
phase_val[9:0]	10
mmcm_psen	0
counter_val[9:0]	005
mmcm_psen	0
counter_val[9:0]	00a
counter_delay	570
glitch_out	1
glitch_sig	0
internal_sig	0
edge	0
pre_state	state8

Figure. 7: Simulation results

In Figure 7, According to the input values :

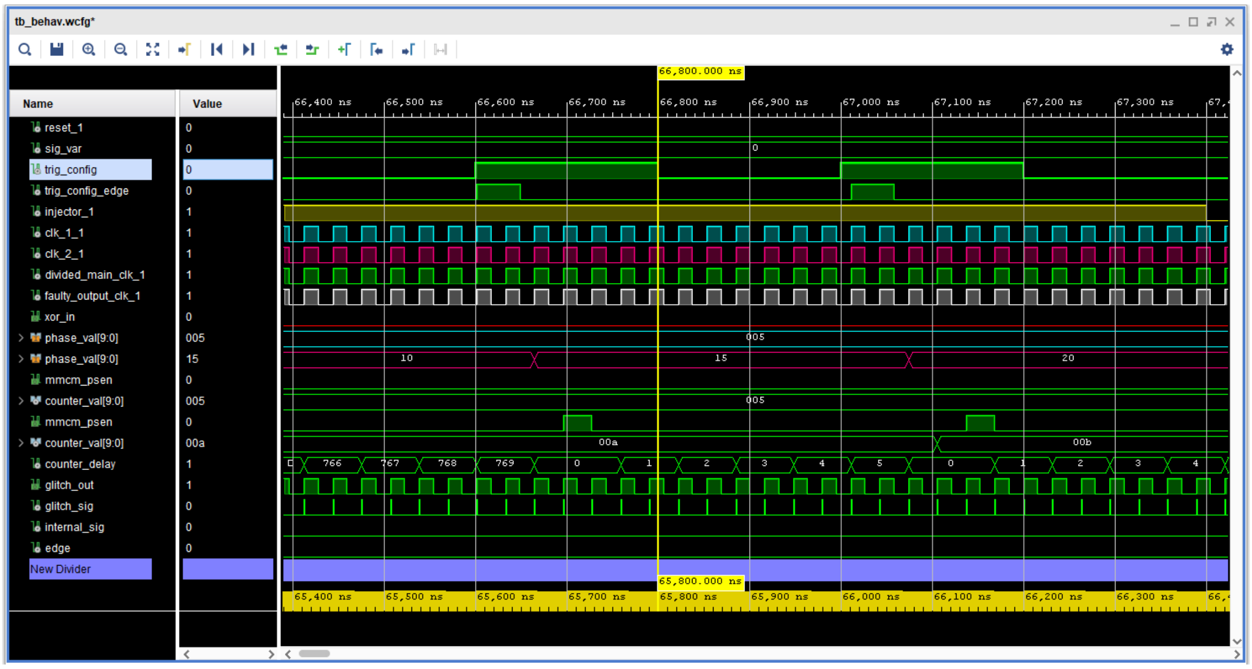
- Trig_config = 0
  Because the trig_config is zero, there is no change in the glitch configuration(glitch-width)

- Injector = 1
  Because the injector=1 and sig_var is equal to 1, a glitch is inserted into the output clock at a distance of one clock after the positive edge of the injector.

- Phase1 = 5 and phase2 = 10, so the width of the glitch obtained from the difference between these two phases will be equal to 5 degrees.

- As can be seen, phase 2 is always greater than phase1.

- The glitch inserted with width=5 and offset=1 in the faulty_output_clk signal is visible in the figure.4 .

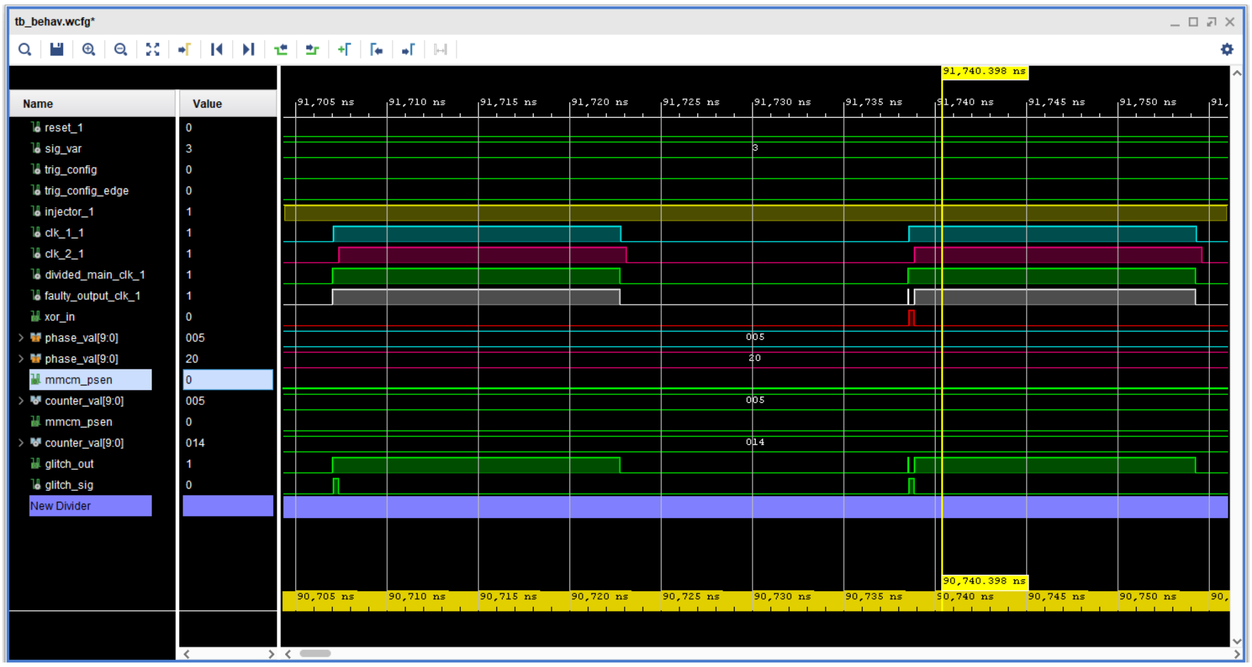


Name	Value
reset_1	0
sig_var	0
trig_config	0
trig_config_edge	0
injector_1	1
clk_1_1	1
clk_2_1	1
divided_main_clk_1	1
faulty_output_clk_1	1
xor_in	0
phase_val[9:0]	005
phase_val[9:0]	15
mmcm_psen	0
counter_val[9:0]	005
mmcm_psen	0
counter_val[9:0]	00a
counter_delay	1
glitch_out	1
glitch_sig	0
internal_sig	0
edge	0
New Divider	

Figure. 8: Simulation results

According to Figure. 8:

- Each time a change in the trig_config is detected by trig_config_edge signal, the glitch configuration then changes, this means that phase2 is increased and because the width is obtained from the difference between phase 2 and phase1, so with the increase of phase2, the width of the glitch increases. In Figure. 8, the glitch width is initially 5(10-5=5) and increases to 10 (15-5=10)with the first change in trig_config signal.
  As can be seen, when trig_config=1, the shift value of phase2 changes from 10 to 15, and as the difference between phase 1 and phase 2 increases(15-5=10), the glitch width increases.
  This created glitch is applied to the faulty_output_clk with the first change in the injector signal. With the second positive edge in trig_config, it can be seen that again the width of the glitch has changed to 15.
- In this figure injector=1, but the glitch is not applied because with the first observation of the change in signal injector the glitch is applied and with the first subsequent change in that glitch is applied to the faulty_output_clk.
- At each edge in the trig_config signal, the value of sig_var is zero, because the function of this signal is to inject glitch with different offset (distance from the injector edge to the glitch) to the output signal, therefore, in order for all offset modes of a glitch to be tested with specific configuration, the value of the sig_var signal must be zero after each change of specifications, that's mean the edge is seen in trig_config.



Name	Value
reset_1	0
sig_var	3
trig_config	0
trig_config_edge	0
injector_1	1
clk_1_1	1
clk_2_1	1
divided_main_clk_1	1
faulty_output_clk_1	1
xor_in	0
phase_val[9:0]	005
phase_val[9:0]	20
mmcm_psen	0
counter_val[9:0]	005
mmcm_psen	0
counter_val[9:0]	014
glitch_out	1
glitch_sig	0
New Divider	

Figure. 9: Simulation results

- In Figure 9, the value of trig_con is zero, which means that the glitch configuration does not change, And the produced glitch has a width of 15 (phase2 – phase1: 20-5=15).

- Since the value of sig_var = 3, the glitch enters the output signal after passing 3 clock clocks from the edge of the injector, and where trig_con is not equal to 1, any change in the injector signal executes the entire glitch offset mode.

- In the faulty_output_clk signal, which is displayed in white, you can see the created glitch, which is 15 degrees wide and its offset is 3.