PREVIOUS STUDY SESSION 4

Sonia Navas Rutete & Jorge Rodrigo Leclercq

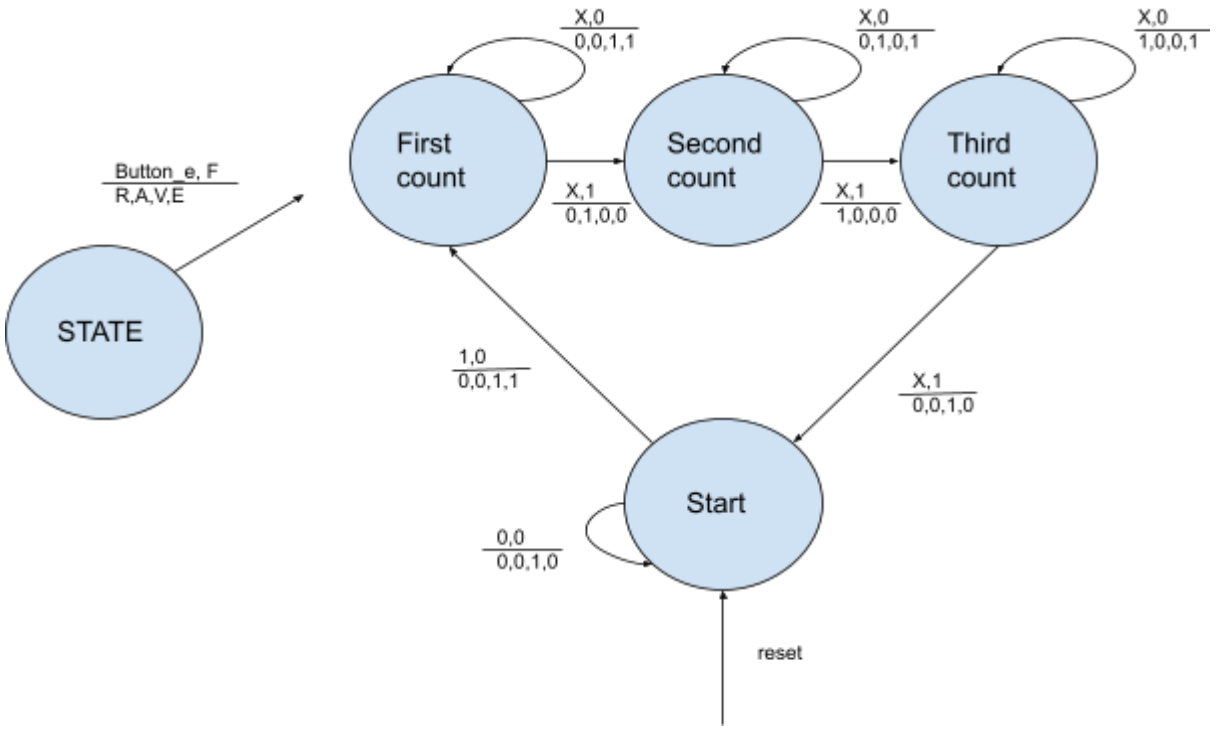
EXERCISE 1

As the period is T = 1/f = 1/1 = 1s, that means that in each rising edge of clk we will have 1 second, so we need 5 cycles to get 5 seconds(=5 rising-edges). So the maximum value for count will be 100 in binary(if we count the 000).

EXERCISE 2

We have chosen a mealy machine as the output is affected by the state and the input.

As the button is only used in the start state, in the rest of situations it doesn't matter (button: x). In addition The enable will always be 1 and change to 0 automatically, to reset the count.



EXERCISE 3

We have just implemented the FSM, that suppose that variables that should be signals, here they are inputs and outputs(e,f,button_e). Moreover, f will be changed by the timer, as well as, the button_e will be changed by the edge_detector.

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

entity state_machine is
	port(
		reset, clk, button_e, f: in STD_LOGIC;
		e: out STD_LOGIC;
		RAG: out STD_LOGIC_VECTOR(2 downto 0)
		);
end state_machine;

architecture functional of state_machine is
	type state is(start, first, second, third);
	signal current_state, next_state: state;

	begin
	sequential: process(clk, reset):
		begin
		if reset = '1' then
			current_state <= start;
		elsif clk'EVENT AND clk='1' then
			current_state <= next_state;
		end if;
	end process sequential;

		fsm: process(current_state, button_e, f):
			begin
			case current_state is
				when start =>
					RAG <= "001";
					f <= '0';
					e <= '0';
					if button_e = '0' then
						next_state <= current_state;
					else
						e <= '1'
						next_state <= first;
					end if;

				when first =>
					if f = '0' then
						next_state <= current_state;
```

```
                    else
                        next_state <= second;
                    end if;

                when second =>
                    RAG <= "010";
                    if f = '0' then
                        next_state <= current_state;
                    else
                        next_state <= third;
                    end if;

                when third =>
                    RAG <= "100";
                    if f = '0' then
                        next_state <= current_state;
                    else
                        next_state <= start;
                end if;
            end case;
        end process fsm;
end functional;
```