Folha de Respostas -Aula 5
Nome do Aluno: Vinícius de Barros Silva
No. USP: 10335913
Instruções:
Completar com apenas as informações e dados solicitados, e apenas nos campos adequados. Siga a enumeração de acordo com a apostila prática.

2) Rc_adder
a) Insira no quadro abaixo a descrição da arquitetura em VHDL do novo RC_adder (com generate)

```
Library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity rc_adder_2 is
	generic
	(
	WIDTH		: natural := 8
	);
	port
	(
	a_i, b_i	:	in STD_LOGIC_VECTOR (WIDTH-1 downto 0);
	z_out		:	out STD_LOGIC_VECTOR (WIDTH-1 downto 0);
	c_i			:	in STD_LOGIC;
	c_o			:	out STD_LOGIC
	);

end rc_adder_2;


architecture structural of rc_adder_2 is


	COMPONENT full_adder_1
		port ( a_in, b_in, c_in		:	in STD_LOGIC;
			z_out, c_out		:	out STD_LOGIC);
	END COMPONENT;



signal carry	: STD_LOGIC_VECTOR (WIDTH-1 downto 0);  -- auxiliary signal carry(x) means carry_out of stage x

begin
```

```
-- for generate para todos os WIDTH bits
R1 : FOR N IN WIDTH-1 downto 0 GENERATE


--      if generate para caso LSB
        R12: IF (N=0) GENERATE
                    F1 :full_adder_1
                    PORT MAP(a_i(N),b_i(N),c_i,z_out(N),carry(N));
                    END GENERATE R12;


--      if generate para caso MSB
        R13 :IF(N=WIDTH-1) GENERATE
                    F2 : full_adder_1
                    PORT MAP (a_i(N),b_i(N),carry(N-1),z_out(N),c_o);
                    END GENERATE R13;

--                  if generate para demais casos
       R14: IF(N>0 AND N<WIDTH-1) GENERATE
                    F3 : full_adder_1
                    PORT MAP(a_i(N),b_i(N),carry(N-1),z_out(N),carry(N));
                    END GENERATE R14;



-- fechar for
END GENERATE R1;


end structural;

-- full_adder_1
Library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity full_adder_1 is
          port
          (
          a_in, b_in, c_in                  :         in STD_LOGIC;
          z_out, c_out              :         out STD_LOGIC
          );

end full_adder_1;
```

```
architecture dataflow of full_adder_1 is

signal aux_xor, aux_and_1, aux_and_2, aux_and_3          : STD_LOGIC;

begin

        z_out <= aux_xor xor c_in AFTER 4 ns;
    aux_xor <= a_in xor b_in AFTER 4 ns;
    aux_and_1 <= a_in and b_in AFTER 2 ns;
    aux_and_2 <= a_in and c_in AFTER 2 ns;
        c_out <= aux_and_1 or aux_and_2 or aux_and_3 AFTER 4 ns;
        aux_and_3 <= b_in and c_in AFTER 2 ns;

end dataflow;
```

b) Insira no quadro abaixo o trecho da arquitetura do módulo stimuli_module, em que fiquem identificadas as mudanças por você realizadas para a verificação do completo funcionamento do rc adder.

```
-- test vectors application
--teste soma carry in 0 com propagação carry
assign_input_words(50, 40);
assign_carry_in('0');
--teste soma carry in 1
assign_input_words(50,40);
assign_carry_in('1');
-- teste para carry out = 1
assign_input_words(184,199);
assign_carry_in('0');
```

c) Faça o upload da imagem da carta de tempos da simulação feita, com os sinais necessários, onde fique evidente o completo funcionamento do rc adder (com generate)- ver item seguinte.

d) No campo abaixo, interprete a imagem do item c), mostrando como fica evidente o completo funcionamento do rc adder (com generate)

Para os primeiros vetores aplicados, já indicado no arquivo recebido, ocorre uma soma básica em que carry in = 0, como visto na simulação a soma de ‘00110010’ com ‘00101000’ resulta em ‘01011010’, vale observar que a propagação de vai um ocorreu corretamente. Para a segunda aplicação, utilizando os mesmos vetores, porém utilizando carry in = 1, o resultado obtido foi ‘01011011’ mudando apenas o ultimo bit como esperado.

Para a terceira aplicação utilizou-se vetores de modo a ocorrer uma propagação de carry out, assim somando ‘10111000’ com ‘11000111’ obtivemos ‘ 01111111’ com carry out = 1, provando o funcionamento esperado.

3) ALU

a) Insira no quadro abaixo a descrição da arquitetura VHDL do novo testbench topo (para a ULA)

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

entity tb_alu is

GENERIC (WIDTH: natural :=8);

end tb_alu;

architecture test of tb_alu is

component stimuli_module
	generic
	(
	WIDTH		: natural := 8
	);

	port
	(
	a_i	:	out STD_LOGIC_VECTOR(WIDTH-1 downto 0));
end component ;

component alu
	generic
	(
	WIDTH			: NATURAL	:= 8
	);

	port
	(
	rb_op			: in STD_LOGIC_VECTOR(WIDTH-1 downto 0);
	alu_result		: out STD_LOGIC_VECTOR(WIDTH-1 downto 0)
	);
end component ;

	signal a_i_s, z_o_s : STD_LOGIC_VECTOR(WIDTH-1 downto 0);
```

```
begin

-- Instantiate DUT
        dut : alu
                generic map(WIDTH => WIDTH)
                port map(rb_op                          => a_i_s,
                                alu_result                      => z_o_s);

-- Instantiate test module
        test : stimuli_module
                generic map(WIDTH => WIDTH)
                port map(a_i                    => a_i_s);


end architecture test;
```

b) Insira no quadro abaixo o trecho do módulo stimuli_module novo, com o processo para a aplicação dos estímulos à ULA

```
simulation : process

-- procedure for vector generation

procedure assign_input_words(constant a: in integer) is
begin
-- Assign values to estimuli_module´s outputs.
a_i <= std_logic_vector(to_unsigned(a,WIDTH));-- representa a com width bits

wait for TIME_DELTA;
end procedure assign_input_words;
```

. c) Insira no quadro abaixo o trecho do módulo stimuli_module em que o processo acima é utilizado.

```
begin

-- test vectors application
--teste iniciando da posição 6
assign_input_words(6);
--teste se tiver na posição 7
assign_input_words(7);
--teste para overflow, somar uma posição na posição max para 8 bits
```

```
assign_input_words(255);


wait;
```

d) Faça o upload da imagem da carta de tempos da simulação feita, com os sinais necessários, onde fique evidente o completo funcionamento da ULA- ver item seguinte.
e) Explique como no quadro abaixo como pode-se verificar, pela imagem subida, que todas as condições de funcionamento da ULA foram testadas.

Foram aplicados dois vetores para testar a soma de ‘1’ bit e apresentou funcinamento correto, também foi aplicado o vetor limite ‘11111111’ para testar overflow, como o resultado excede a capacidade da ULA, obtivemos o vetor ‘00000000’, quando deveriamos ter ‘100000000’.

5)

a) Insira no quadro abaixo os trechos da arquitetura do módulo stimuli_module, em que fiquem identificadas e comentadas as mudanças por você realizadas para a verificação do completo funcionamento do contador.

```
sim : process

    procedure check_counter(go_value      : in STD_LOGIC)       is
            begin
                -- Assign values to estimuli_module´s outputs.
                go <= go_value;

                wait until rising_edge (clk_s);
                -- Events at the rising edge of next clock cycle
                end procedure check_counter;

            procedure reset_activate is    -- reset activation procedure
            begin
                wait until falling_edge(CLK_s);
                rst <= '1';
                wait for CLK_PERIOD;
                rst <= '0';
            end procedure reset_activate;

            procedure assign_input_words(constant a: in SPEED_FACTOR) is
                begin
                -- Assign values to speed.
                speed_sync <= a;
```

```
                wait for CLK_PERIOD;
        end procedure assign_input_words;


    begin
        -- Apply test vectors

        check_counter('0');    -- initialize

        reset_activate;   -- counter will start

        assign_input_words (SPEED_FACTOR'VAL(0));                -- É equvivalente
a "assign_input_words (x1)"


        for k in 0 to 3 loop

            if (k > 0) then

            assign_input_words (SPEED_FACTOR'VAL(k));    -- É equvivalente a
"assign_input_words (x2,x3,x4) para k = 1,2,3"

            check_counter('1');     -- inicia nova contagem

            end if;

            wait for 1*CLK_PERIOD;

            check_counter('0');    --resetting go

            while (cnt_guru_rdy /= '1')
                loop
                    wait for 1*CLK_PERIOD;
            end loop;


            wait for 10*CLK_PERIOD;

        end loop;

    end process sim;
```

c) Faça o upload da(s) imagem(ns) da carta de tempos da simulação feita, com os sinais adequados, onde fique evidente o funcionamento do contador para todas as velocidades- ver item seguinte.

d) No campo abaixo, interprete a(s) imagem(ns) do item c), mostrando como fica evidente o completo funcionamento do contador

Inicialmente aplicado um reset , parte-se da velocidade x1 até que o cont guru seja 1 e possa zerar a contagem e iniciá la para velocidade x2, o momento de transição de x1 para x2 é registrado na imagem 1com cont guru e cont discípulo atingindo ‘1’ e a contagem no limite.
A imagem 2 evidencia o funcionamento do go para iniciar nova contagem assim que há uma troca de velocidade. Na imagem a troca evidenciada é de x1 para x2.
A imagem 3 evidencia a troca de x2 para x4 com as mesmas observações.
A imagem 4 evidencia um momento em que o cont discípulo atinge 1 fora do limite do guru.
A imagem 5 evidencia a troca de x4 para x8.
E por fim a imagem 6 evidencia o fim da contagem para velocidade x8 e o zeramento da contagem novamente.