UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMATICA

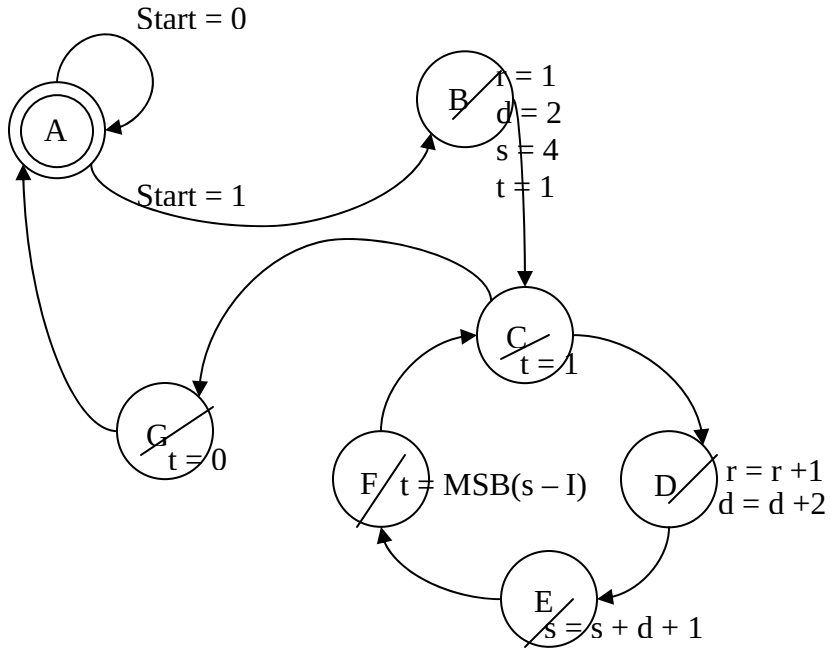
LISTA DE EXERCÍCIOS DE SISTEMAS DIGITAIS

Prof. Fernanda Gusmão de Lima Kastensmidt
E Marcelo Porto (aluno mestrado PPGC)

Descreva em VHDL, simule no simulador logico e sintetize usando uma ferramenta de CAD para FPGA :

1- Um multiplexador 8-1 de 16 bits.
2- Um demultiplexador 1-8 16bits.
3- Um somador carry look ahead de 4 bits usando componentes de somador de 1 bit
4- Um buffer ping-pong de 8 bits.
5- Um código que calcule o MDC para elementos de entrada de 4 bits.
6- Uma ULA de 8 bits com as seguintes operações:
   - saída = A + B
   - saída = A - B
   - saída = A or B
   - saída = A and B
   - saída = A xor B
   - saída = inv(A) inverte o valor de A
   - saída = 0 (coloca zero na saída)

7- Descreva uma parte operativa para a máquina de estados abaixo, com um operador para cada registrador. Esta máquina de estados descreve os passos do algoritmo do cálculo da raiz quadrada, onde r,d,s e t são sinais internos e I é o sinal de entrada (número que se deseja obter a raiz). Ao final do processo o sinal r possui o resultado da raiz para a entrada I.



8- Descreva um código VHDL que represente a máquina de estados do exercício 7, gerando todos os sinais de controle necessários para a parte operativa descrita no exercício 7.

9- Desenvolva uma segunda arquitetura para o exercício 7 utilizando apenas 1 operador (somador/subtrator), visando redução da utilização de recursos. Analise os resultados de síntese gerados pela ferramenta e compare os resultados.

10- Um contador de 16 bits UP/Down com reset, e carga paralela.

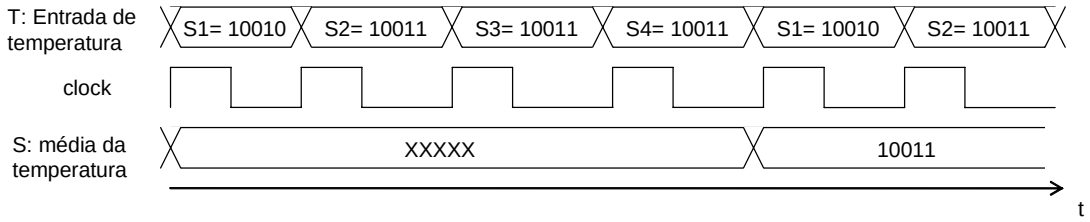
11- Um registrador de 8 bits com deslocamento para esquerda, direira, hold e carga paralela.

12- Conversor BCD para 7 segmentos

Agora alguns exercicios de Projeto

13-: Projete um sistema digital baseado em parte de controle e parte operativa capaz de calcular a média de temperatura de uma câmera frigorífica com 4 sensores instalados.
Cada sensor informa a temperatura medida em 5 bits. A temperatura medida pelos sensores é transmitida a entrada do sistema T_{5-1} de forma serial como mostra o diagrama de tempos a baixo. Note que de quatro em quatro pulsos de relógio deve haver uma nova média de temperatura na saída S_{5-1} do sistema.

A câmara pode operar em temperaturas positivas e negativas, representadas em complemento de 2.



Pede-se:

a) descrição do algoritmo em alto nível
b) alocação de registradores
c) parte de controle implementada através de diagramas de estados
d) parte operativa implementada através de um datapath com transferência entre registradores baseada em multiplexadores

14: Projete uma máquina de estados finitos (FSM) que detecte a seqüência “1011” na entrada A. Toda a vez que ocorrer essa seqüência, a saída T, que controla o acendimento de uma lâmpada, deve permanecer em “1” por dois ciclos de relógio e após retornar a “0”.

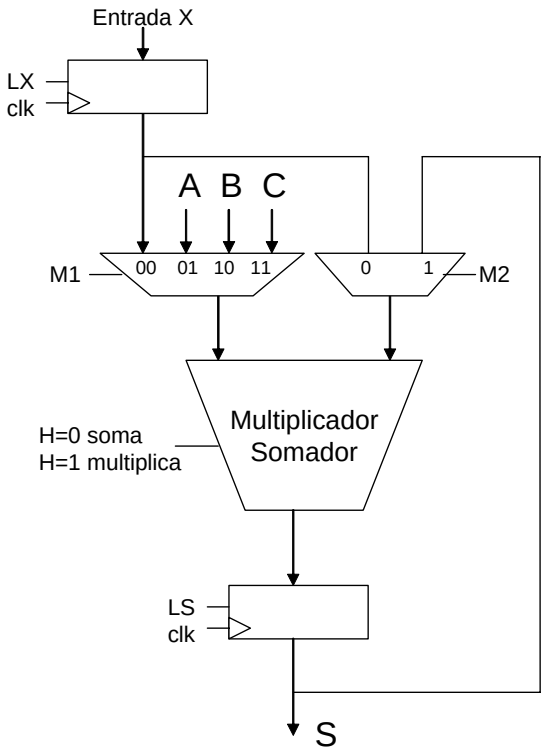
Pede-se:

a) Desenhe o diagrama de estados da máquina projetada.
b) A máquina implementada é de Mealy ou de Moore? Explique. _____
_____
c) Quantos flip-flops são necessários para a implementação dessa máquina de estados em um circuito digital? _____

15: Projete um bloco de controle (represente na forma de diagrama de estados) que realize a seguinte operação no datapath a seguir:

$$S= A.X^2 + B.X + C$$

.



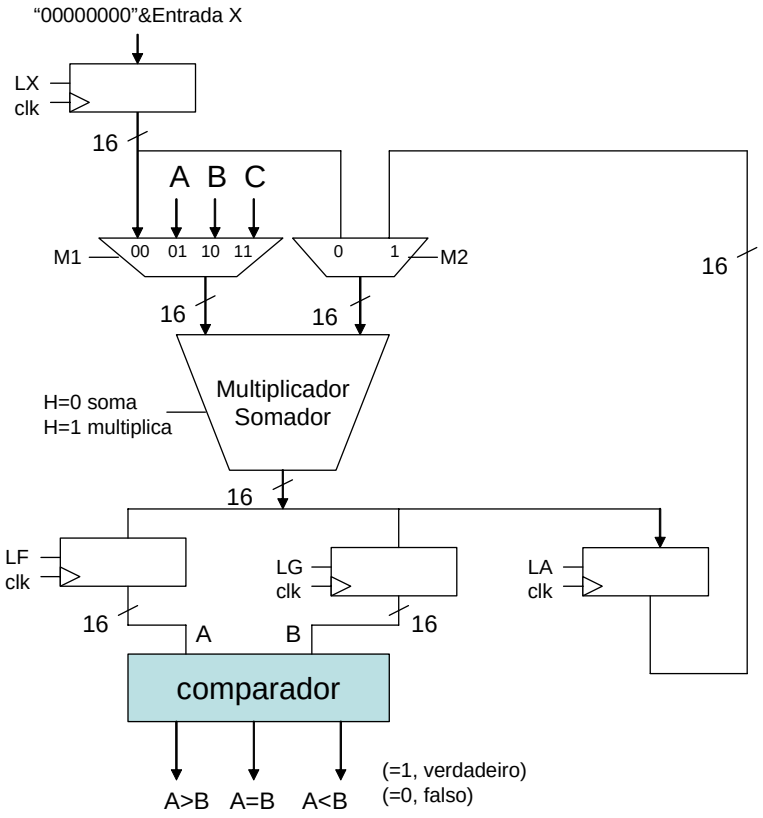
16: Dada as funções $F(x) = A.\ x^2 + B$, e $G(x) = C.x + A$, pede-se:

a) Projete um bloco de controle para a seguinte parte operativa a fim de determinar para um dado valor de x, se F(x) é maior, igual ou menor a G(x) e indique quantos ciclos de relógio são necessários para obter a resposta. (2 pontos)

b) Redesenhe uma nova parte operativa, mudando os recursos, a fim de aumentar o desempenho, ou seja, diminuir o numero de estados da parte de controle. Explique as principais mudanças na parte operativa e indique o numero de ciclos de relógio necessário para obter a resposta neste novo caso. Não é preciso refazer o bloco de controle. (1 ponto)

c) Determine a freqüência máxima de operação deste sistema digital com base nos atrasos presentes na tabela a seguir. Os atrasos referentes ao flip-flop valem tanto para os flip-flops que guardam os estados na parte de controle como para os registradores da parte operativa. (1 ponto)

Tempo de propagação do flip-flop	3 ns
Tempo de propagação da função de saída (PC->PO)	10 ns
Tempo de propagação do mux	15 ns
Tempo de propagação multiplicador/somador	25 ns
Tempo de propagação do comparador	22 ns
Tempo de propagação da função de próximo estado	8 ns
Tempo de setup do flip-flop	1 ns


"00000000"&Entrada X
LX
clk
16
A B C
M1
00 01 10 11
0 1
M2
16
16
16
Multiplicador
Somador
H=0 soma
H=1 multiplica
16
LF
clk
LG
clk
LA
clk
16
A
B
16
comparador
A>B A=B A<B
(=1, verdadeiro)
(=0, falso)

.

Respostas:

Exercício 1:

```
----------------------------------------------------------------------
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Mux8 IS
  PORT(
    Sel : In std_logic_vector(2 downto 0);
    A0, A1, A2, A3 : In std_logic_vector(15 downto 0);
    A4, A5, A6, A7 : In std_logic_vector(15 downto 0);
    output : Out std_logic_vector(15 downto 0)
    );
END Mux8;

ARCHITECTURE behavior OF Mux8 IS

BEGIN

PROCESS (Sel, A0, A1, A2, A3, A4, A5, A6, A7)
BEGIN
        case Sel is
                when "000" => output <= A0;
                when "001" => output <= A1;
                when "010" => output <= A2;
                when "011" => output <= A3;
                when "100" => output <= A4;
                when "101" => output <= A5;
                when "110" => output <= A6;
                when "111" => output <= A7;
                when others =>
        end case;
END PROCESS;

END behavior;
----------------------------------------------------------------------
```

Exercício 2:

```
----------------------------------------------------------------------
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY Dmux8 IS
        PORT(
          Sel : In std_logic_vector(2 downto 0);
          S : In std_logic_vector(15 downto 0);
          output0, output1, output2, output3 : Out std_logic_vector(15 downto 0)
          );
END Dmux8;

ARCHITECTURE behavior OF Dmux8 IS

BEGIN

output0 <= S WHEN sel = "000";
output1 <= S WHEN sel = "001";
output2 <= S WHEN sel = "010";
output3 <= S WHEN sel = "011";
output4 <= S WHEN sel = "100";
output5 <= S WHEN sel = "101";
output6 <= S WHEN sel = "110";
output7 <= S WHEN sel = "111";

END behavior;
----------------------------------------------------------------------
```

Exercício 3:

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

entity add4cla is
        port ( cin              : in std_logic;
                        input0  : in std_logic_vector(3 downto 0);
                        input1          : in std_logic_vector(3 downto 0);
                        output          : out std_logic_vector(3 downto 0);
                        gen     : out std_logic;
                        prop    : out std_logic
                );
end add4cla;
architecture behavior of add4cla is

signal c0, c1, c2 : std_logic;
signal p, g : std_logic_vector(3 downto 0);

begin

output(0) <= p(0) xor cin;
output(1) <= p(1) xor c0;
output(2) <= p(2) xor c1;
output(3) <= p(3) xor c2;

p(0) <= input0(0) xor input1(0);
p(1) <= input0(1) xor input1(1);
p(2) <= input0(2) xor input1(2);
p(3) <= input0(3) xor input1(3);

g(0) <= input0(0) and input1(0);
g(1) <= input0(1) and input1(1);
g(2) <= input0(2) and input1(2);
g(3) <= input0(3) and input1(3);

c0 <= g(0) or (p(0) and cin);
c1 <= g(1) or (p(1) and g(0)) or (p(1) and p(0) and cin);
c2 <= g(2) or (p(2) and g(1)) or (p(2) and p(1) and g(0)) or (p(2) and p(1) and p(0) and
cin);

prop <= p(0) and p(1) and p(2) and p(3);
gen  <= g(3) or (p(3) and g(2)) or (p(3) and p(2) and g(1)) or (p(3) and p(2) and p(1) and
g(0));

end behavior;
```

Exercício 4:

```
Library ieee;
Use ieee.std_logic_1164.all;

ENTITY ping_pong IS
  PORT(

    clk: In std_logic;
    En : In std_logic;
    Ain : In std_logic_vector(15 downto 0);

    A0, A1, A2, A3 : Out std_logic_vector(15 downto 0);
    A4, A5, A6, A7 : Out std_logic_vector(15 downto 0);
    A8, A9, A10, A11 : Out std_logic_vector(15 downto 0);
    A12, A13, A14, A15 : Out std_logic_vector(15 downto 0)

    );
END ping_pong;

ARCHITECTURE behavior OF ping_pong IS

signal Aa0, Aa1,  Aa2, Aa3, Aa4, Aa5,  Aa6, Aa7, Aa8, Aa9,  Aa10, Aa11, Aa12, Aa13,
Aa14, Aa15 : std_logic_vector(15 downto 0);
signal Ab0, Ab1,  Ab2, Ab3, Ab4, Ab5,  Ab6, Ab7, Ab8, Ab9,  Ab10, Ab11, Ab12,
Ab13,  Ab14, Ab15 : std_logic_vector(15 downto 0);

BEGIN

PROCESS (clk)
BEGIN
  if (clk'event and clk = '1') then
        Aa1 <= Aa2;
        Aa2 <= Aa3;
        Aa3 <= Aa4;
        Aa4 <= Aa5;
        Aa5 <= Aa6;
        Aa6 <= Aa7;
        Aa7 <= Aa8;
        Aa8 <= Aa9;
        Aa9 <= Aa10;
        Aa10 <= Aa11;
        Aa11 <= Aa12;
        Aa12 <= Aa13;
```

```
        Aa13 <= Aa14;
        Aa14 <= Aa15;
        Aa15 <= Ain;
    if (en = '1') then
        Ab0 <= Aa1;
        Ab1 <= Aa2;
        Ab2 <= Aa3;
        Ab3 <= Aa4;
        Ab4 <= Aa5;
        Ab5 <= Aa6;
        Ab6 <= Aa7;
        Ab7 <= Aa8;
        Ab8 <= Aa9;
        Ab9 <= Aa10;
        Ab10 <= Aa11;
        Ab11 <= Aa12;
        Ab12 <= Aa13;
        Ab13 <= Aa14;
        Ab14 <= Aa15;
        Ab15 <= Ain;
    end if;
  end if;

END PROCESS;

A0 <= Ab0;
A1 <= Ab1;
A2 <= Ab2;
A3 <= Ab3;
A4 <= Ab4;
A5 <= Ab5;
A6 <= Ab6;
A7 <= Ab7;
A8 <= Ab8;
A9 <= Ab9;
A10 <= Ab10;
A11 <= Ab11;
A12 <= Ab12;
A13 <= Ab13;
A14 <= Ab14;
A15 <= Ab15;

END behavior;
-------------------------------------------------------------------
```

Exercício 5:

```
Library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

ENTITY mdc_outro IS
   PORT(
        clk, start, reset: in std_logic;
        x, y: in std_logic_vector(3 downto 0);
        p: out std_logic;
        mdc: out std_logic_vector(3 downto 0)
   );
END mdc_outro;

ARCHITECTURE behavior OF mdc_outro IS

signal a, b: std_logic_vector(3 downto 0);
BEGIN

process (clk)
begin
        if (reset='1' or start='1') then
                p<='0';
                mdc<="0000";
        elsif (a=b) then
                p<='1';
                mdc<=a;
        end if;
end process;

process (a, b, clk, start, reset)
begin
        if (reset='0') then
                if (clk'event and clk='1') then
                        if (start='1') then
                                a<=x;
                                b<=y;
                        end if;
                        if (a < b) then
                                b<=b-a;
                        elsif (a > b) then
                                a<=a-b;
                        end if;
                end if;
```

```
        end if;
end process;

END behavior;
-------------------------------------------------------------------
```

Exercício 6:

```
----------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;

ENTITY ULA IS
PORT (
        s : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
        A, B : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
        F : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
);
END ULA;

ARCHITECTURE comportamento OF ULA IS

BEGIN

PROCESS (op, A, B)
BEGIN
        CASE op IS
                WHEN “000” => F <= “0000”;
                WHEN “001” => F <= A + B;
                WHEN “010” => F <= A – B;
                WHEN “011” => F <= A or B;
                WHEN “100” => F <= A and B;
                WHEN “101” => F <= A xor B;
                WHEN “110” => F <= not(A);
                WHEN OTHERS => F <= “1111”;
        END CASE;
END PROCESS;
END comportamento;
----------------------------------------------------------------------
```

Exercício 7/8:

```
----------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity Raiz_V1S is
	--generic (N : positive:= 8); -- valor padrão
	port(
		clk, start: in std_logic;

		I: in std_logic_vector(15 downto 0);

		pronto: out std_logic;
		Result: out std_logic_vector(7 downto 0)

    );
end Raiz_V1S;

architecture behavior of Raiz_V1S is

signal t,p, Mux_t : std_logic;
type tipo_estado is (A, B, C, Di, E, F, G);
signal estado_atual, proximo_estado: tipo_estado;
signal SelMuxr,SelMuxd,SelMuxs, SelMuxt: std_logic_vector(1 downto 0);
signal r, Mux_r: std_logic_vector(7 downto 0);
signal t_tmp,d,s, Mux_d, Mux_s: std_logic_vector(15 downto 0);

begin

--------------------------- Parte Operativa ----------------------------
---------------- r -----------------
process(clk)
begin
	if(clk'event AND clk = '1') then
		--r <= Mux_r;
		with SelMux select
		Mux_r <= "00000001" when "00",
		Mux_r <= r when "01",
		Mux_r <= r +1 when "10",
		unaffected when others;
	end if;
end process;
-------------------------------------

----------------- d -----------------
```

```
process(SelMuxd, d)
begin
	case SelMuxd is
		when "00" => Mux_d <= "0000000000000010";
		when "01" => Mux_d <= d;
		when "10" => Mux_d <= d + 2;
		when others =>
	end case;
end process;


process(clk)
begin
	if(clk'event AND clk = '1') then
		d <= Mux_d;
	end if;
end process;
-----------------------------------------

------------------- s ----------------------
process(SelMuxs, s, d)
begin
	case SelMuxs is
		when "00" => Mux_s <= "0000000000000100";
		when "01" => Mux_s <= s;
		when "10" => Mux_s <= s + d + 1;
		when others =>
	end case;
end process;


process(clk)
begin
	if(clk'event AND clk = '1') then
		s <= Mux_s;
	end if;
end process;
-----------------------------------------

----------------- t ---------------------------
t_tmp <= s - I;

process(clk)
begin
	if(clk'event AND clk = '1') then
		t <= Mux_t;
```

```
        end if;
end process;

process(SelMuxt, t_tmp, t)
begin
        case SelMuxt is
                when "00" => Mux_t <= '1';
                when "01" => Mux_t <= t;
                when "10" => Mux_t <= t_tmp(15);
                when others =>
        end case;
end process;
-----------------------------------------

result <= r;
pronto <= p;

------------------  Parte de Controle --------------------

------------ Maquina de estado --------------
process(clk, proximo_estado)
begin
        if(clk'event and clk = '1') then
                estado_atual <= proximo_estado;
        end if;
end process;

process(estado_atual, start, t)
begin
        case estado_atual is

                        when A => if(start = '0') then
                                                proximo_estado <= A;
                                        elsif(start = '1') then
                                                proximo_estado <= B;

                                        end if;

                        when B => SelMuxr <= "00";
                                        SelMuxd <= "00";
                                        SelMuxs <= "00";
                                        SelMuxt <= "00";
                                        p <= '0';
                                        proximo_estado <= C;

                        when C => if(t='0') then
```

```
                                        proximo_estado <= G;
                                    elsif(t = '1') then
                                        proximo_estado <= Di;
                                    end if;

                when Di => SelMuxr <= "10";
                                     SelMuxd <= "10";
                                     SelMuxs <= "01";
                                     SelMuxt <= "01";
                                     proximo_estado <= E;

                when E => SelMuxr <= "01";
                                    SelMuxd <= "01";
                                    SelMuxs <= "10";
                                    SelMuxt <= "01";
                                    proximo_estado <= F;

                when F => SelMuxr <= "01";
                                    SelMuxd <= "01";
                                    SelMuxs <= "01";
                                    SelMuxt <= "10";
                                    proximo_estado <= C;

                when G => p <= '1';
                                proximo_estado <= A;
        end case;
end process;
-----------------------------------------------

end behavior;
-----------------------------------------------------------------------
```