Universidade Federal de Santa Catarina
Centro Tecnológico – CTC
Departamento de Engenharia Elétrica

“Circuitos e Técnicas Digitais”

Prof. Eduardo Augusto Bezerra

Eduardo.Bezerra@ufsc.br

Florianópolis, agosto de 2015.

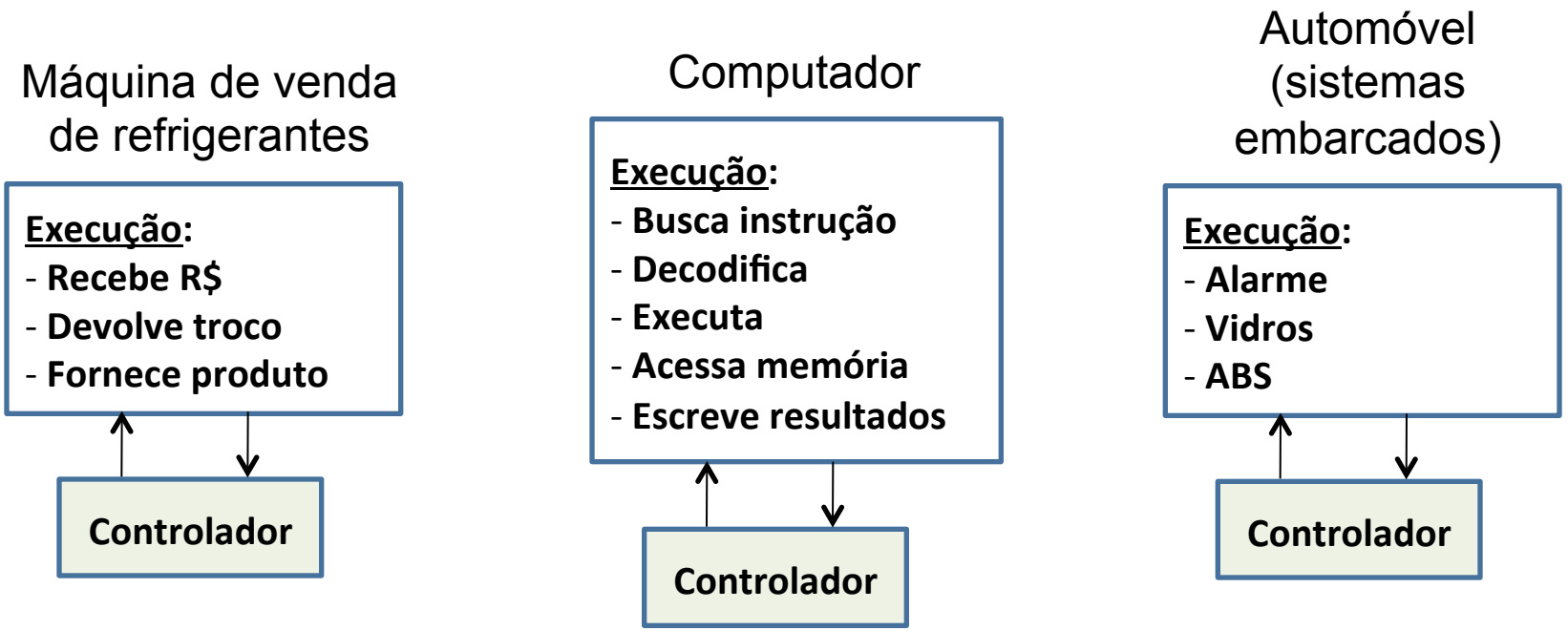
Objetivos do laboratório

1. Entender o conceito de máquinas de estados (FSM).
2. Entender o conceito de circuito sequencial controlando o fluxo de atividades de circuito combinacional.
3. Entender o processo de síntese de FSMs em VHDL.
4. Entender o funcionamento de contadores.
5. Estudo de caso: projeto e implementação em VHDL de um contador baseado em máquinas de estados.

“Síntese de máquinas de estado (FSM)”

Finite State Machine (FSM)

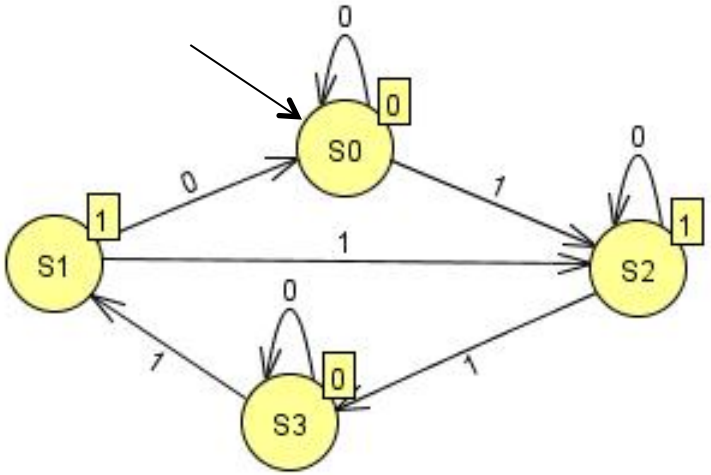
- Sistemas computacionais, normalmente, são compostos por um módulo de "controle" e um módulo para "execução das operações".



Finite State Machine (FSM)

- O “controlador” é responsável por coordenar a sequência de atividades a ser realizada em um determinado processo (ou sistema)
- Em sistemas digitais são utilizados “circuitos sequenciais” na geração de sinais de controle
- Um circuito sequencial transita por uma série de estados e, a cada estado (a cada momento), poderá fornecer uma determinada saída
- As saídas são utilizadas no controle da execução de atividades em um processo
- A lógica sequencial utilizada na implementação de uma FSM possui um número “finito” de estados.

Finite State Machine (FSM)



Modelo de comportamento composto por:

- Estados
- Transições
- Ações

Estado

Armazena informação sobre o passado refletindo as modificações das entradas do início até o presente momento

Transição

Indica uma troca de estado e é descrita por uma condição que habilita a modificação de estado

Ação

Descrição da atividade que deve ser executada em um determinado instante

Tarefa a ser realizada na aula prática

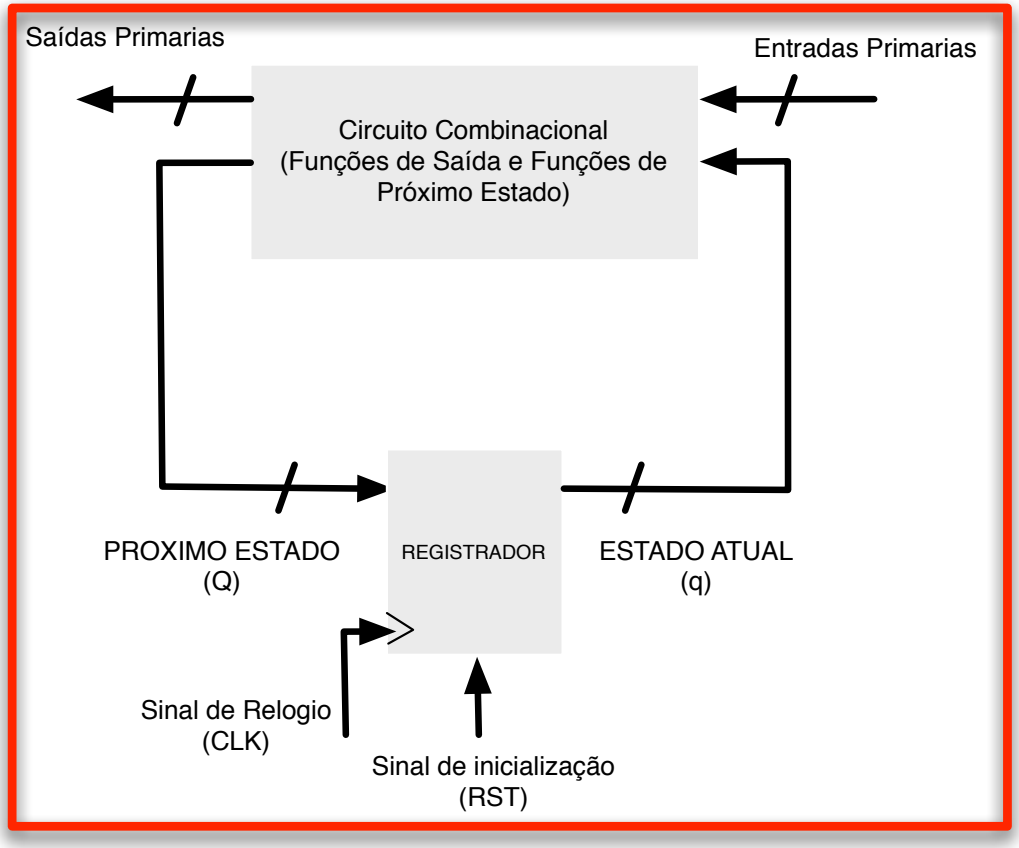
Solução I – FSM estrutural

Solução II – FSM comportamental

Solução I: FSM estrutural

Solução I: Estrutura da FSM

- Dois módulos:
  - Armazenamento do “estado atual”; e
  - Cálculo da “saída” e do “próximo estado”



Solução I: Estrutura da FSM

- Armazenamento do “estado atual”
  - Registrador construído a partir de flip-flops
- Cálculo da “saída” e do “próximo estado”
  - Circuito combinacional; ou
  - Tabela verdade da lógica de saída e da lógica de próximo estado armazenada em uma memória (ROM, Flash, RAM, ...)

Primeira tarefa a ser realizada: Solução I

- **Implementar uma FSM em VHDL, a partir da metodologia apresentada nas aulas teóricas, para geração dos caracteres 'A' a 'E' da tabela ASCII (http://asciitable.com), apresentando os caracteres (em binário) nos LEDs vermelhos (LEDR).**

- **FSM com *reset* assíncrono, usando o botão KEY(0) para inicializar um contador com o valor do primeiro caractere a ser gerado ('A' = 41H).**

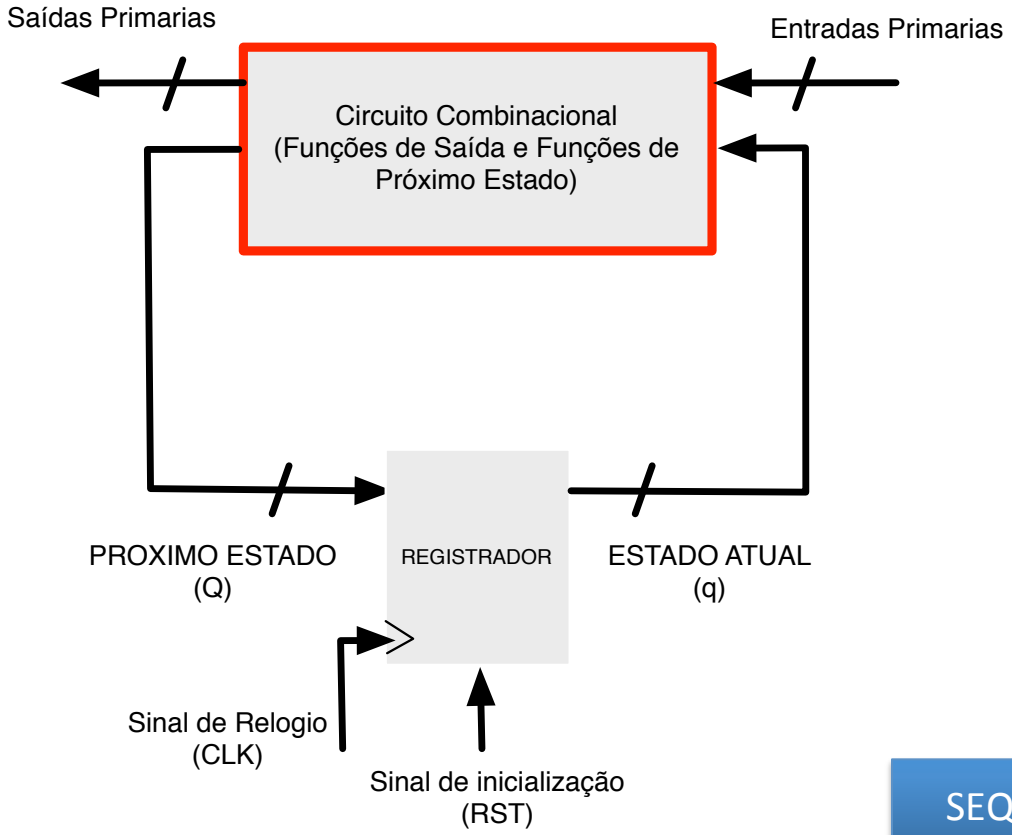
- **Usar KEY(1) como relógio manual, para avançar para o próximo estado, gerando assim o próximo caractere da tabela ASCII.**

- **Ao atingir o último caractere da tabela ASCII, a FSM deverá voltar ao início da sequencia, gerando novamente o caractere 'A'.**

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Solução I - Abordagem Estrutural

Determinação "manual" do circuito combinacional (lógica) para geração da saída e próximo estado.



SEQUENCIADOR: 1→ 2→ 3→4→5→1→…

Solução I: Diagrama de estados e codificação de estados

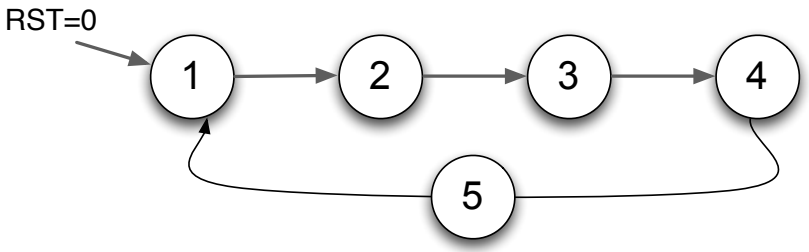


Diagrama de estados

ESTADOS	q_3	q_2	q_1	q_0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1

Tabela de codificação de estados

Solução I: Tabela de transição de estados e lógica nas entradas dos FFs

Tarefa a ser realizada:

1.- Preencher a tabela de transição de estados

2.- Obter a lógica nas entradas dos FFs tipo D (D_1 e D_0).

Tabela de transição de estados

Entradas				Saídas							
Estado Atual				Próximo Estado				Reg. de estado			
q3	q2	q1	q0	Q3	Q2	Q1	Q0	D3	D2	D1	D0
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	1	0	0	0	1		
0	1	0	0	0	1			0	1		
0	1	0	1	0	0			0	0		

Lógica nas estradas dos FFs (tipo D)

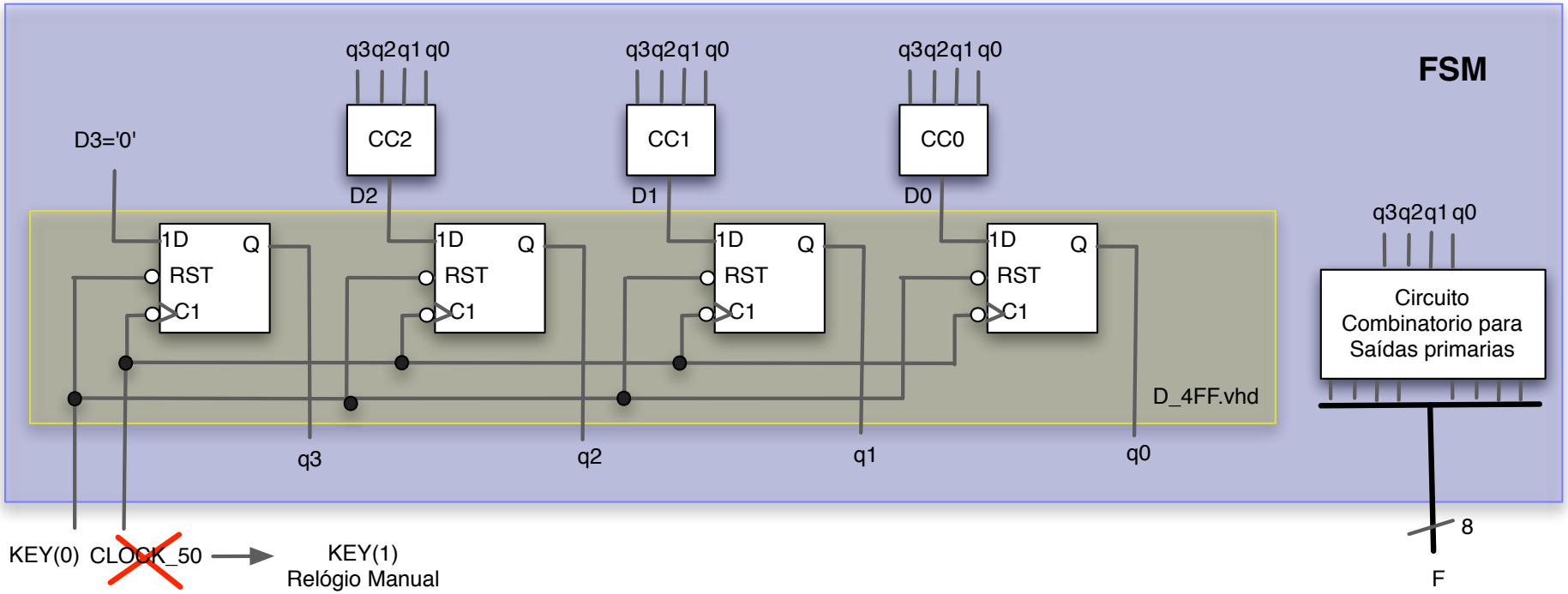
D3 = 0

D2 = $\overline{q3}\ \overline{q2}$ q1q0+$\overline{q3}$ q2 $\overline{q1}\overline{q0}$

D1=?

D0=?

Solução I: Diagrama de blocos do circuito a ser implementado



Tarefa a ser realizada:

1. Completar o código VHDL do slide 17 com a lógica obtida para as entradas D_1 e D_0 (ver slide 15);
2. Criar um componente D_4FF.vhd com o VHDL do registrador de 4 bits do lab. 6 (ver slide 14 do lab. 6), porém sensível à borda de descida.
3. Criar o sinal de 8 bits a partir das saídas dos FFDs e o valor $4_{(Hex)}$ requerido para o primeiro caractere da contagem.

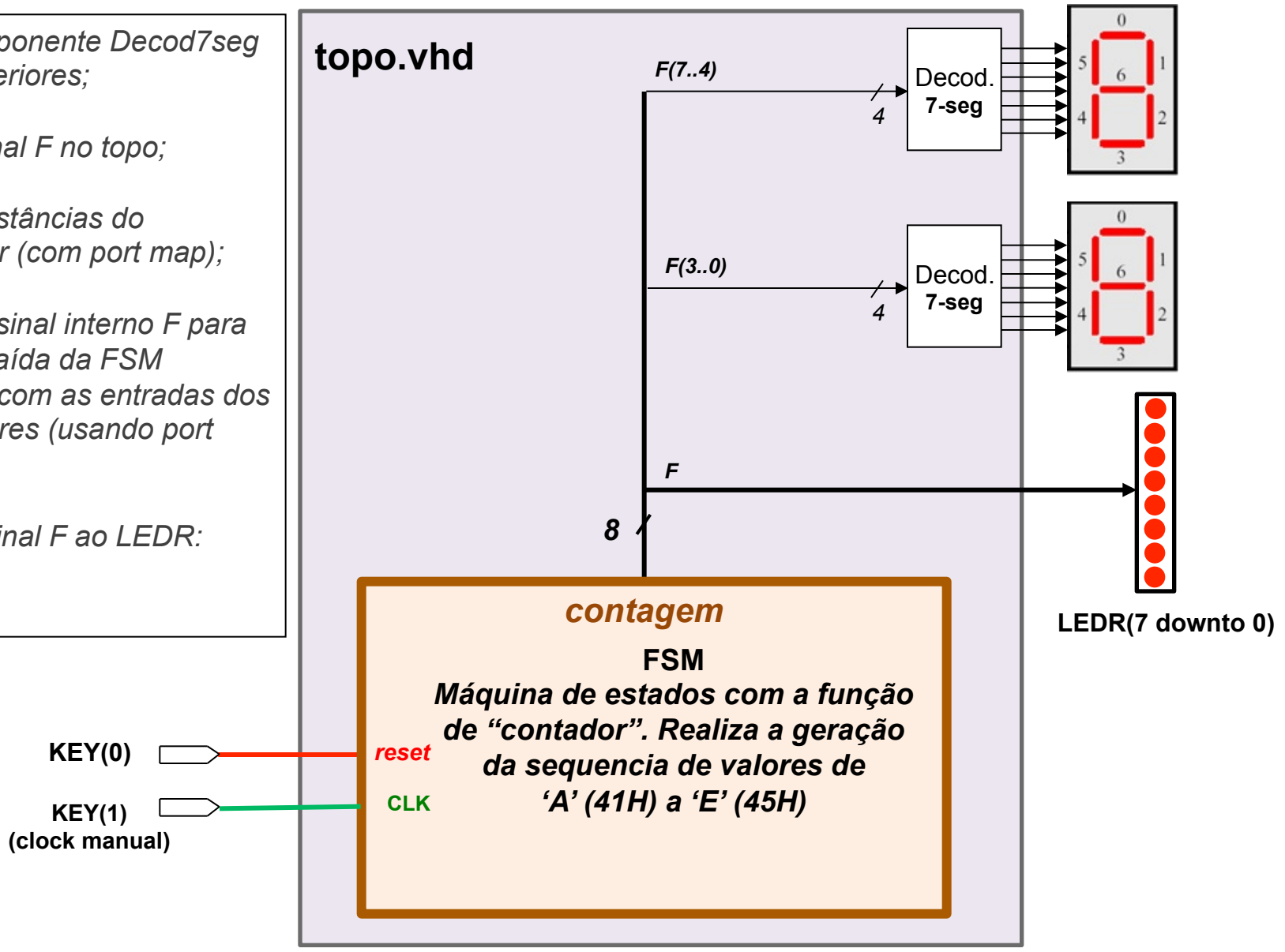
Solução I: Codigo VHDL da hierarquia superior “topo”

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

entity topo is
port (KEY : IN std_logic_vector(1 downto 0);
HEX0: out std_logic_vector(6 downto 0);
HEX1: out std_logic_vector(6 downto 0);
LEDR : OUT STD_LOGIC_VECTOR(7 downto 0)
);
end topo;

architecture topo_estru of topo is
  signal D, q: std_logic_vector(3 downto 0);
  signal F: std_logic_vector (7 downto 0);

component D_4FF
port (
       CLK, RST: in std_logic;
       D: in std_logic_vector(3 downto 0);
       Q: out std_logic_vector(3 downto 0)
);
end component;

component decod7seg
port (
         C: in std_logic_vector(3 downto 0);
         Y: out std_logic_vector(6 downto 0)
);
end component;
```

```
begin

-- Inicio da FSM --
D(3) <= '0';
D(2) <= (not(q(3)) and not(q(2)) and q(1) and q(0)) or
(not(q(3)) and q(2) and not(q(1)) and not(q(0)));
D(1) <= -- A preencher
D(0) <= -- A preencher

L0: D_4FF port map (KEY(1), KEY(0), D(3 downto 0), q(3 downto 0));
F <= -- A preencher
-- Fim da FSM –

LEDR <= F;
L2: decod7seg port map (F(7 downto 4), HEX1);
L3: decod7seg port map (F(3 downto 0), HEX0);

end topo_estru;
```

Solução I: Diagrama de blocos do circuito a ser implementado "Uso dos displays de 7-segmentos como saída"

1. *Incluir o componente Decod7seg dos labs anteriores;*
2. *Criar um signal F no topo;*
3. *Criar duas instâncias do decodificador (com port map);*
4. *Usar o novo sinal interno F para conectar a saída da FSM (Contagem) com as entradas dos decodificadores (usando port map);*
5. *Conectar o sinal F ao LEDR: LEDR <= F.*



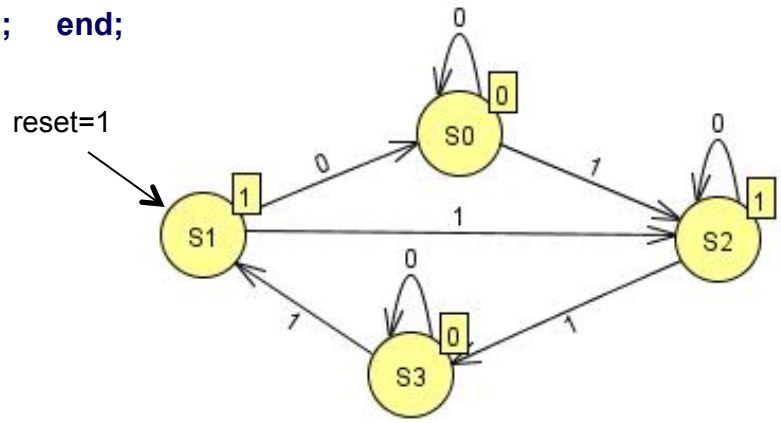
Solução II: FSM comportamental

Solução II: Uso de VHDL para descrever uma FSM

VHDL
1 proc

Máquina de estados – 1 processo

```
entity MOORE is    port(X, clock, reset : in std_logic;     Z: out std_logic);    end;

architecture B of MOORE is
   type STATES is (S0, S1, S2, S3);
   signal EA: STATES;
begin
   process(clock, reset)
   begin
         if reset= '1' then
                  EA  <= S0;
          elsif clock'event and clock='1' then
                  case EA is
                    when S0 => Z <= '0';
                                                  if X='0' then EA <=S0; else EA <= S2; end if;
                    when S1 => Z <= '1';
                                                  if X='0' then EA <=S0; else EA <= S2; end if;
                    when S2 =>  Z <= '1';
                                                   if X='0' then EA <=S2; else EA <= S3; end if;
            when S3 => Z <= '0';
                                                  if X='0' then EA <=S3; else EA <= S1; end if;
      end case;
          end if;
   end process;
end B;
```



- EA: mesmo comportamento
- Saída Z ficará defasada 1 ciclo de clock

Solução II: Uso de VHDL para descrever uma FSM

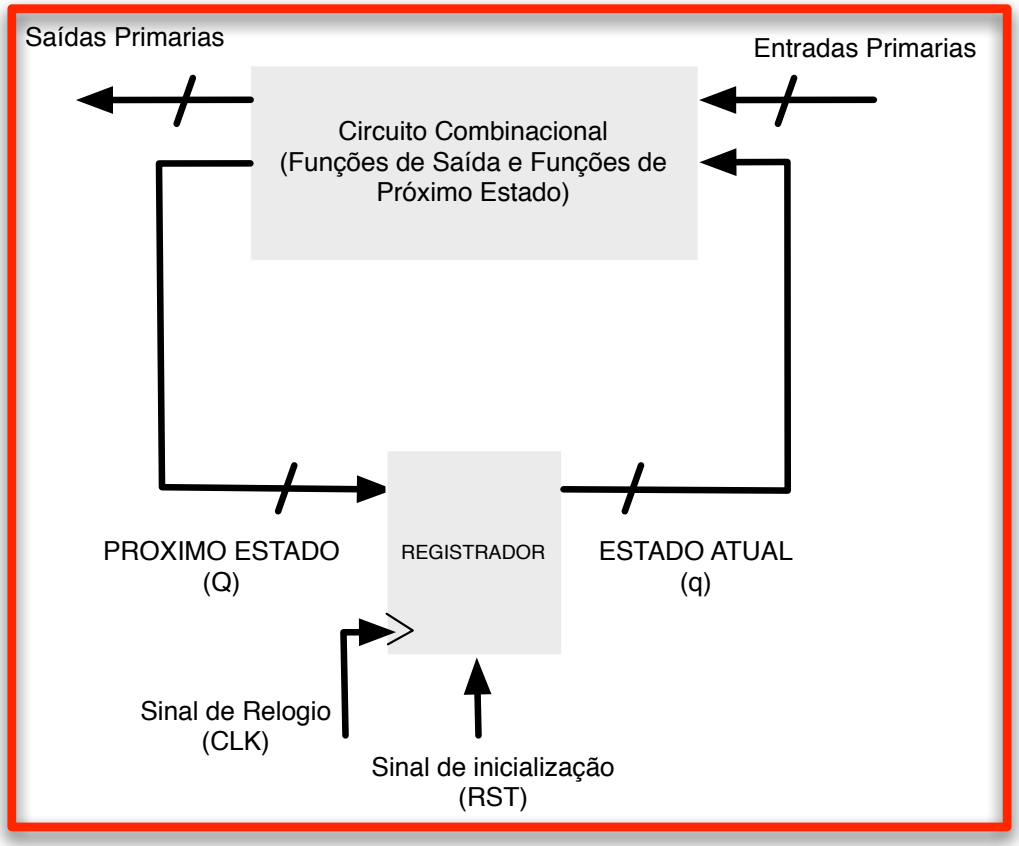
VHDL
1 proc

Máquina de estados – 1 processo

```
process(clock, reset)
begin
        if reset= '1' then
                EA  <= S0;
        elsif clock'event and clock='1' then
                case EA is
                        when S0 =>      Z <= '0';
                                        if X='0' then
                                                EA <=S0;
                                        else
                                                EA <= S2;
                                        end if;
                        when S1 =>      Z <= '1';
                                        if X='0' then
                                ...
                        end case;
        end if;
end process;
```

Solução II - Abordagem Comportamental

A ferramenta de síntese (do Quartus II) realiza a geração automática de toda lógica necessária.



Segunda tarefa a ser realizada: Solução II

- **Implementar uma FSM em VHDL usando *process*, para geração dos caracteres 'A' a 'E' da tabela ASCII (http://asciitable.com), apresentando os caracteres (em binário) nos LEDs vermelhos (LEDR).**

- **FSM com *reset* assíncrono, usando o botão KEY(0) para inicializar um contador com o valor do primeiro caractere a ser gerado ('A' = 41H).**

- **Usar um botão, KEY(1), por exemplo, para avançar para o próximo estado, gerando assim o próximo caractere da tabela ASCII.**

- **Ao atingir o caractere 'E' da tabela ASCII, ou seja, 45H, a FSM deverá voltar ao início da sequencia, gerando novamente o caractere 'A'.**

Solução II
Os 4 passos para projetar uma FSM comportamental

PASSO 1: Descrever o sistema da forma mais completa possível (requisitos);

PASSO 2: Preparar uma representação gráfica da FSM, refinando/ redesenhando quando necessário;

PASSO 3: Criar uma tabela de transição de estados para a FSM, listando as entradas e as saídas, incluindo todos os estados (atual e próximo);

PASSO 4: Descrever em VHDL o comportamento da FSM a partir do diagrama e da tabela de estados, conforme slides 11 a 12;

Em uma metodologia clássica de projeto de FSM em sistemas digitais, mais passos são necessários, como apresentado na solução 1.

Solução II
Os 4 passos para projetar uma FSM comportamental

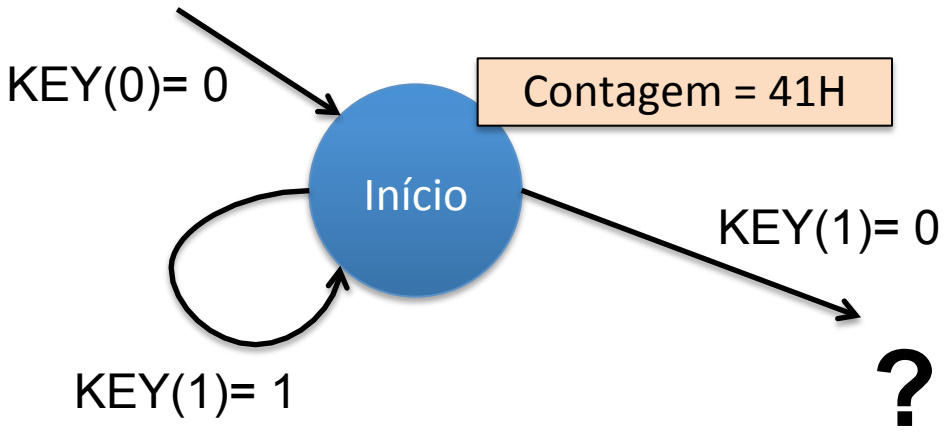
PASSO 1: Descrever o sistema da forma mais completa possível (requisitos).

Exemplo de descrição da FSM:

- No estado inicial, a FSM fornecerá a saída “01000001” (ou 41 em hexadecimal = “A” em ASCII)
- Haverá uma mudança de estado quando o pino clock sofrer uma transição (borda de descida)
- A transição de 1 para 0 no clock (KEY(1) pressionada), causará o incremento do valor de saída
- O último estado da FSM fornecerá a saída com o caractere “E” em ASCII (45 em hexadecimal)
- Após o último estado, a FSM deve retornar ao primeiro estado (apresentar saída “A”)
- A qualquer instante, quando o botão de reset for pressionado, a FSM também deverá retornar para o estado inicial, apresentando a saída “A” em ASCII.

Solução II
Os 4 passos para projetar uma FSM comportamental

Passo 2: Preparar uma representação gráfica da FSM, redesenhando quando necessário.



Inputs:
- Reset - KEY(0)
- Clock – KEY(1)

Output:
- Contagem

Quem é o próximo estado?
Qual a saída do próximo estado?
Quantos estados serão necessários?

Completar o diagrama de estados!

Solução II
Os 4 passos para projetar uma FSM comportamental

PASSO 3: Escrever a tabela de transição de estados para a FSM, listando entradas e saídas, incluindo todos os estados (atual e próximo).

ENTRADAS			SAÍDAS	
KEY(0) (Reset)	KEY(1) (Clock)	EA	PE	Contagem
0	X	Inicio	Inicio	41H
1	0	Inicio	S2	42H
1	1	Inicio		
1	0	S2	S3	43H
1	1	S2	S2	42H
1	0	S3		
1				
1				
1				
1		S5		
1		S5		

EA – Estado Atual
PE – Próximo Estado

Solução II
Os 4 passos para projetar uma FSM comportamental

PASSO 4: Descrever em VHDL o comportamento da FSM a partir do diagrama e da tabela de estados, conforme slides 26 e 27.

```
entity FSM_Conta is
        contagem: out std_logic_vector(7 downto 0);
                clock: in std_logic;
                reset: in std_logic);
end;
architecture bhv of FSM_Conta is
     type STATES is (Inicio, !!!!!completar com os estados que achar necessario!!!!);
     signal EA: STATES;
begin
     process(clock, reset)
     begin
             if reset= '0' then
                     EA  <= Inicio;
                     contagem <= "01000001";
             elsif clock'event and clock='0' then
                     case EA is
                             when Inicio =>
                                     contagem <= "01000001";
                                     EA <= ?????;
                             when ???? =>
                                     contagem <= ????????;
                                     EA <= ?????;
                             when ???? =>
                     end case;
             end if;
     end process;
end bhv;
```

Solução II: Dicas

Usar Topo.vhd com componente FSM
(sem utilizar os displays de 7-segmentos)

```
entity Topo is
    port (          LEDR: out std_logic_vector(7 downto 0);
                    KEY: in std_logic_vector(1 downto 0)
          );
end Topo;
architecture topo_beh of Topo is
    component FSM_Conta   -- Esse e' o componente FSM
        port (
                contagem: out std_logic_vector(7 downto 0);
                clock: in std_logic;
                reset: in std_logic
        );
    end component;
-- incluir component decodificador
begin

    L0: FSM_Conta port map ( LEDR, KEY(1), KEY(0) );
--incluir decodificador
end topo_beh;
```

Solução II: Diagrama de blocos do circuito a ser implementado
"Uso dos displays de 7-segmentos como saída"

1. *Incluir o componente Decod7seg dos labs anteriores;*
2. *Criar um signal F no topo;*
3. *Criar duas instâncias do decodificador (com port map);*
4. *Usar o novo sinal interno F para conectar a saída da FSM (Contagem) com as entradas dos decodificadores (usando port map);*
5. *Conectar o sinal F ao LEDR: LEDR <= F.*