



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



“EEL7020 – Sistemas Digitais”

Prof. Eduardo Augusto Bezerra

Eduardo.Bezerra@eel.ufsc.br

Florianópolis, agosto de 2011.

Sistemas Digitais

Circuitos sequenciais, latches e flip-flops

Arquivo: lab3_VHDL.pdf
parte IV e parte V

Descrição de circuitos sequenciais em VHDL

```
library ieee;
use ieee.std_logic_1164.all;
entity Sinais is port (
    C: in std_logic;
    D: in std_logic;
    Q: out std_logic
);
end Sinais;

architecture behv of Sinais is
    signal A, B: std_logic;
begin
    A <= D;
    Q <= B;

    P1: process (C, D)
    begin
        B <= '0';
        if (C = '1') then
            B <= D;
        end if;
    end process P1;
end behv;
```


Process

- Define uma **SEQUÊNCIA DE COMANDOS** a ser realizada pelo circuito.
- O **processo é acionado**, e sua sequência de comandos é executada, sempre que ocorrer uma **alteração** em algum elemento da sua **LISTA DE PARÂMETROS** (Ex. **C** ou **D**).
- **Um processo nunca termina - CÍCLICO.**
- Dessa forma, após a execução do último comando, o primeiro comando da sequência é executado, **SEMPRE** que ocorrer uma nova alteração em algum parâmetro.
- Obs. Comando **IF .. THEN .. ELSE** é utilizado **APENAS** dentro de um *process*.

Descrição de circuitos sequenciais em VHDL

Process

- A lista de parâmetros de um processo é denominada **SENSITIVITY LIST**.



```
process (A, B, C, D)
begin
```

```
    A <= '1';
    B <= '1';
    B <= D;
    A <= not B;
    C <= A and '1';
    D <= C;
```

```
end process;
```

Sequencial

- Os valores atribuídos aos sinais pelos comandos do processo, **só serão válidos após a execução do último comando**, ou seja, após “sair” do processo.
- Se existirem **várias atribuições** a um mesmo sinal, **APENAS a última atribuição será válida** (ex. sinal B no corpo do processo).
- Não é permitido declarar sinais dentro de um processo.

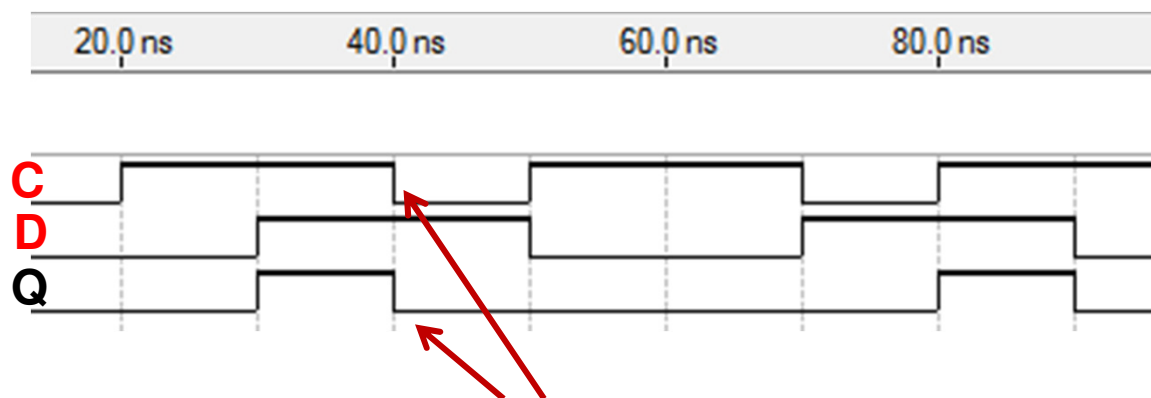
Descrição de circuitos sequenciais em VHDL

```
library ieee;
use ieee.std_logic_1164.all;
entity Sinais is port (
    C: in std_logic;
    D: in std_logic;
    Q: out std_logic
);
end Sinais;

architecture behv of Sinais is
    signal A, B: std_logic;
begin
    A <= D;
    Q <= B;

    P1: process (C, D)
    begin
        B <= '0';
        if (C = '1') then
            B <= D;
        end if;
    end process P1;
end behv;
```

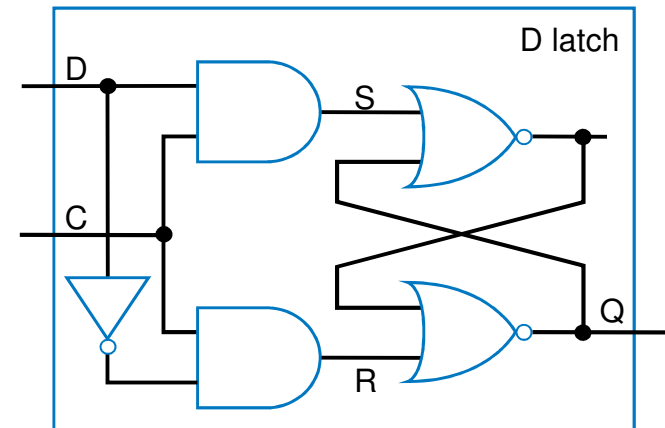
Resultado da simulação do VHDL



C foi alterado para '0', logo o processo é executado novamente, e B receberá '0', pois o teste do *IF* será falso. Logo, Q será zero (combinacional).

Latch D

```
library ieee;
use ieee.std_logic_1164.all;
entity D_latch is port (
    C: in std_logic;
    D: in std_logic;
    Q: out std_logic
);
end D_latch;
architecture behv of D_latch is
begin
    process(C, D)
    begin
        if (C = '1') then
            Q <= D;
        end if;
    end process;
end behv;
```



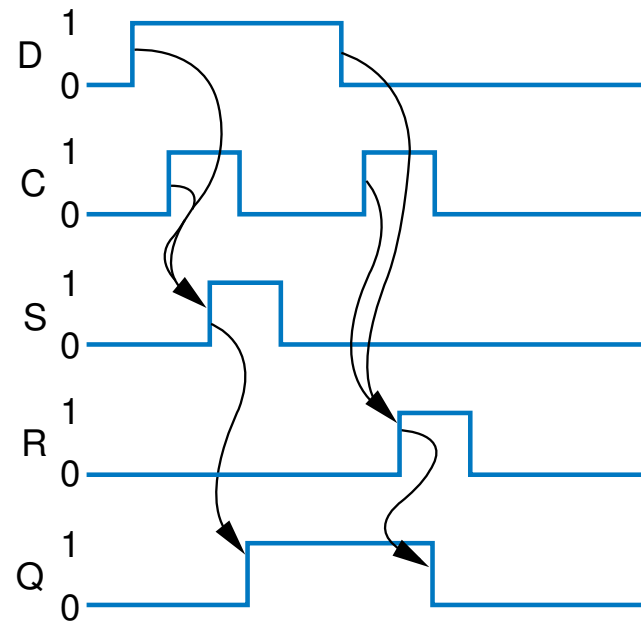
C	D	Q_{t+1}
0	X	Q_t
1	0	0
1	1	1

Mantém estado

Latch D

```
library ieee;
use ieee.std_logic_1164.all;
entity D_latch is port (
    C: in std_logic;
    D: in std_logic;
    Q: out std_logic
);
end D_latch;
architecture behv of D_latch is
begin
    process(C, D)
    begin
        if (C = '1') then
            Q <= D;
        end if;
    end process;
end behv;
```

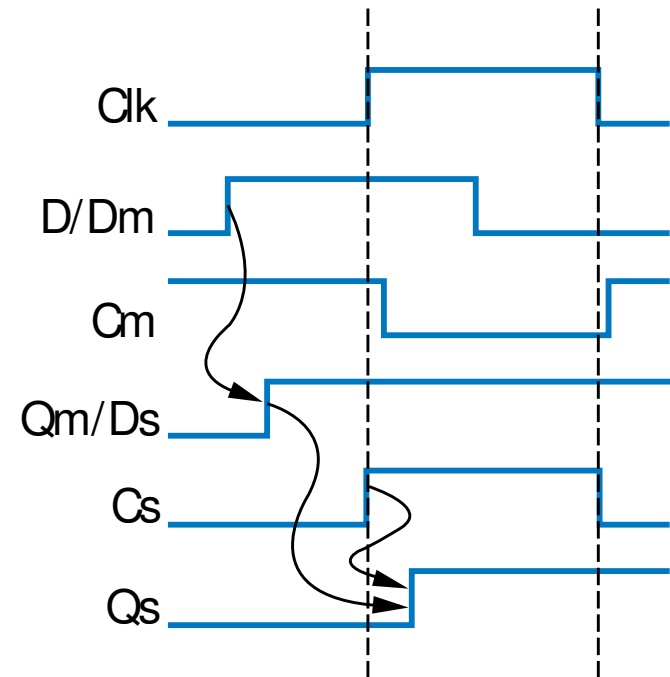
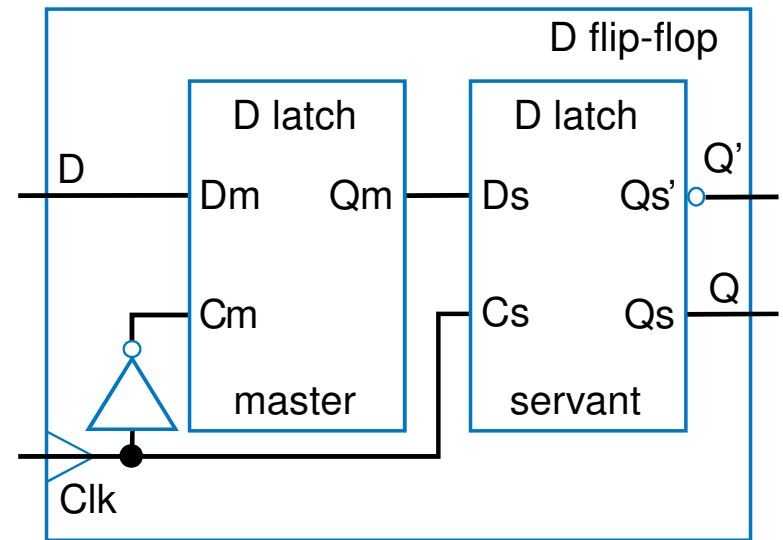
C	D	Q_{t+1}
0	X	Q_t Mantém estado
1	0	0
1	1	1



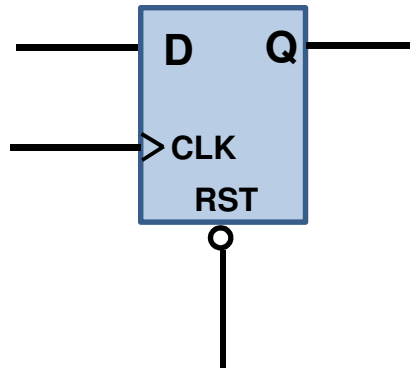
Flip-Flop D

```

library ieee;
use ieee.std_logic_1164.all;
entity D_FF is port (
    CLK: in std_logic;
    D: in std_logic;
    Q: out std_logic
);
end D_FF;
architecture behv of D_FF is
begin
    process(CLK, D)
    begin
        if (CLK'event and CLK = '1') then
            Q <= D;
        end if;
    end process;
end behv;
    
```



Flip-Flop D com RESET assíncrono



- Sempre que a entrada RST for Zero, a saída Q será Zero.
- Quando RST for diferente de Zero, o valor na saída Q vai depender da entrada CLK.
- Se CLK for '1' E for uma borda de subida, então Q receberá a entrada D.

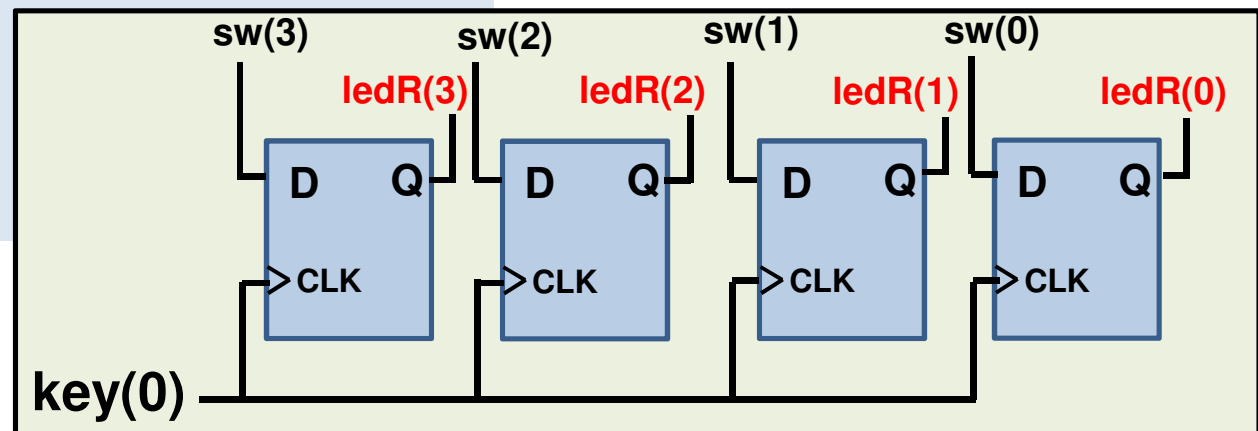
```
library ieee;
use ieee.std_logic_1164.all;
entity D_FF is port (
    CLK: in std_logic;
    RST: in std_logic;
    D: in std_logic;
    Q: out std_logic
);
end D_FF;
architecture behv of D_FF is
begin
    process(CLK, RST, D)
    begin
        if (RST = '0') then
            Q <= '0';
        elsif (CLK'event and CLK = '1') then
            Q <= D;
        end if;
    end process;
end behv;
```

Flip-Flop D (4 bits)

```
library ieee;
use ieee.std_logic_1164.all;
entity D_4FF is port (
    CLK: in std_logic;
    D: in std_logic_vector(3 downto 0);
    Q: out std_logic_vector(3 downto 0)
);
end D_4FF;
architecture behv of D_4FF is
begin
    process(CLK, D)
    begin
        if (CLK'event and CLK = '1') then
            Q <= D;
        end if;
    end process;
end behv;
```

Enquanto não ocorrer um novo evento no sinal CLK e enquanto esse evento for diferente de '1' (borda de subida), então a saída Q do flip-flop continuará armazenando o valor atual.

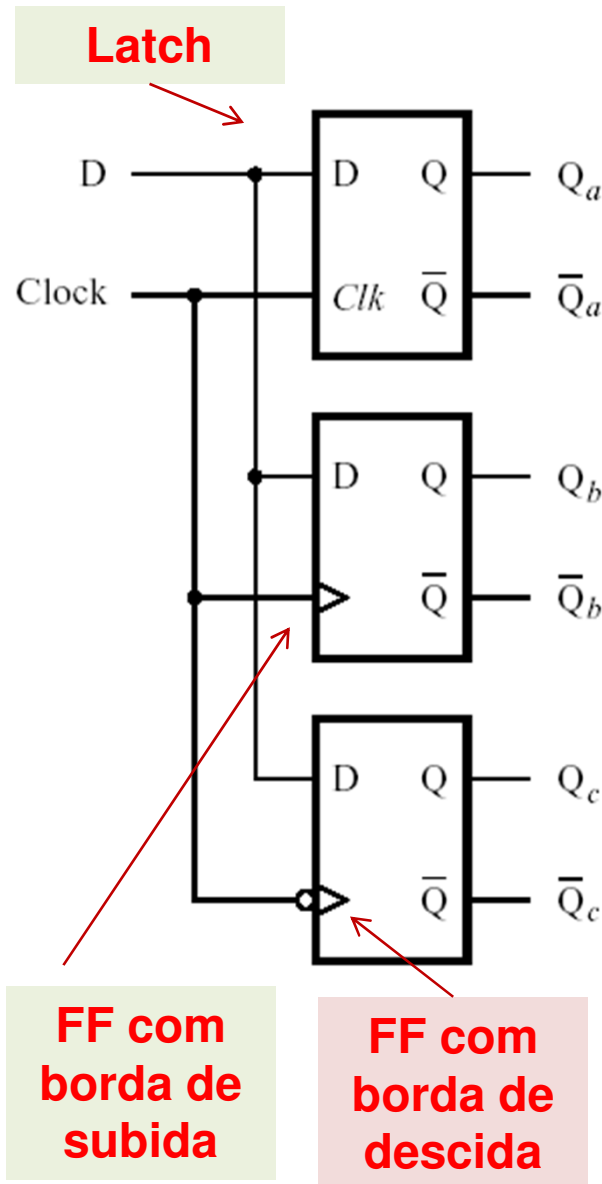
Ao ocorrer um novo evento em CLK, e se esse novo evento for uma transição de '0' para '1' (borda de subida), então a saída Q receberá o novo valor existente na entrada D.



***Tarefa a ser realizada:
Circuito com dois flip-flops e um latch***

Arquivo: lab3_VHDL.pdf
parte IV

Tarefa: Circuito com dois FFs e um *latch*

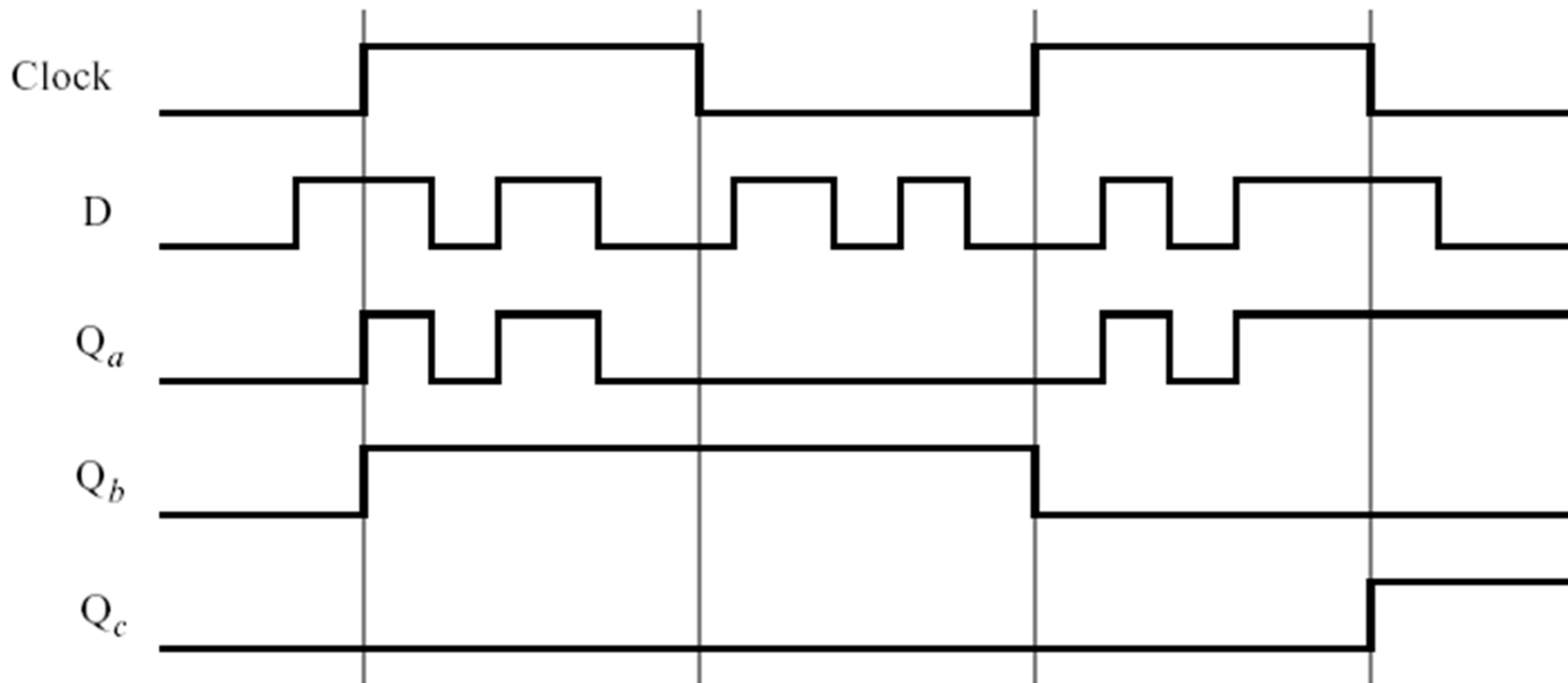


Etapas:

1. Implementar o circuito em VHDL, usando o Quartus II, e realizar a síntese.
2. Utilizar a ferramenta em *Tools -> Netlist Viewers -> Technology Map Viewer*, e examinar o circuito gerado pela síntese.
3. Verificar que o *latch* é implementado em uma *lookup table*, e para os *flip-flops* são aproveitados os *flip-flops* existentes no FPGA da DE2.
4. Realizar a simulação funcional de acordo com os dados descritos no diagrama de formas de onda do próximo slide.
5. Observar as diferenças no comportamento dos três elementos de armazenamento implementados.
6. Após a simulação, utilizar chaves (no lugar do D e clock) e LEDs (nas saídas), e testar o circuito no FPGA da DE2.

Tarefa: Circuito com dois FFs e um *latch*

Resultado esperado da simulação funcional.



Sugestão de implementação VHDL: criar um componente LATCH e um componente FLIP-FLOP e, na descrição VHDL principal (top), utilizar esses componentes para criar o LATCH e os dois FFs, e realizar o mapeamento dos pinos, conectando os diversos sinais com PORT MAP.

Tarefa opcional

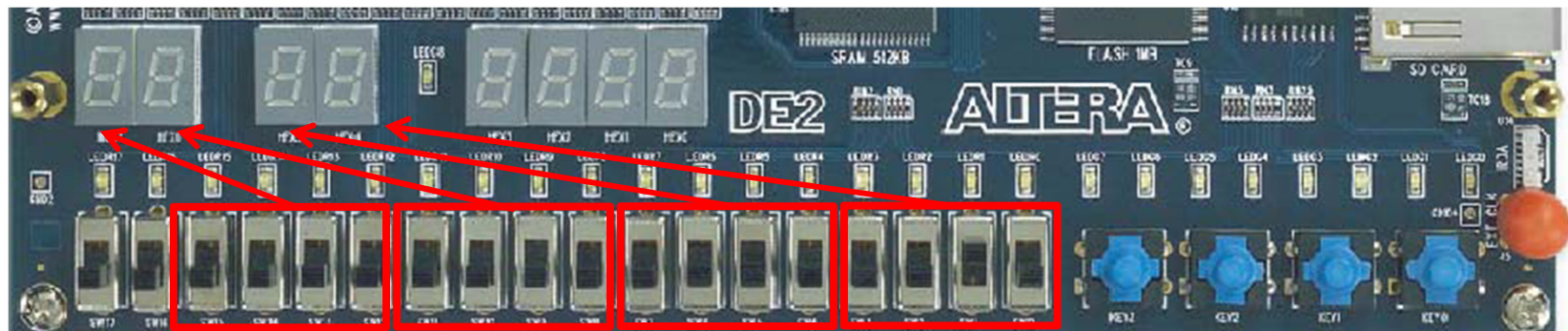
Para aqueles que terminaram a tarefa do lab. 7, e tiverem interesse em aprender um pouco mais sobre processos, *flip-flops*, e registradores em VHDL/FPGAs...

“Display de 7-segmentos com registrador”

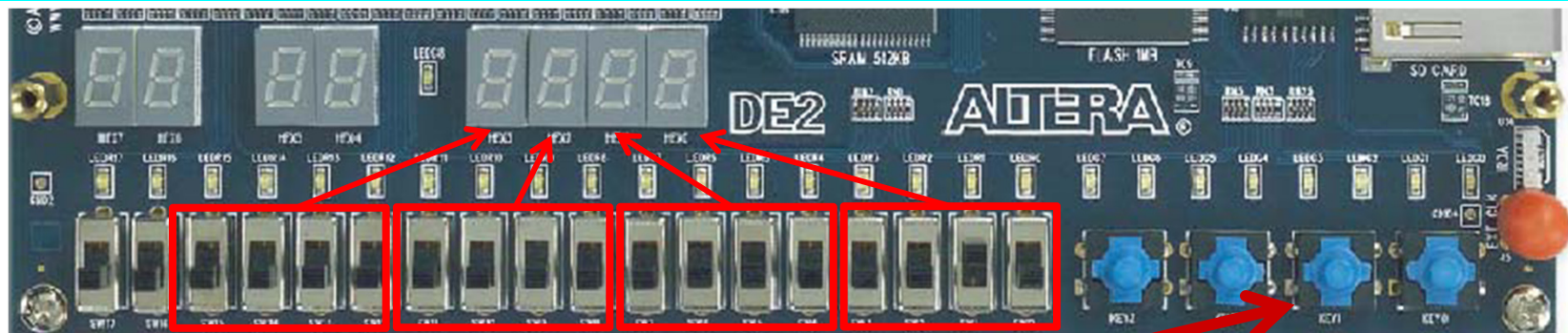
Arquivo: lab3_VHDL.pdf
parte V

Tarefa opcional: Descrição do problema

Escrever em HEX7 a HEX4 o “valor A” de 16 bits das chaves SW0 a SW15:

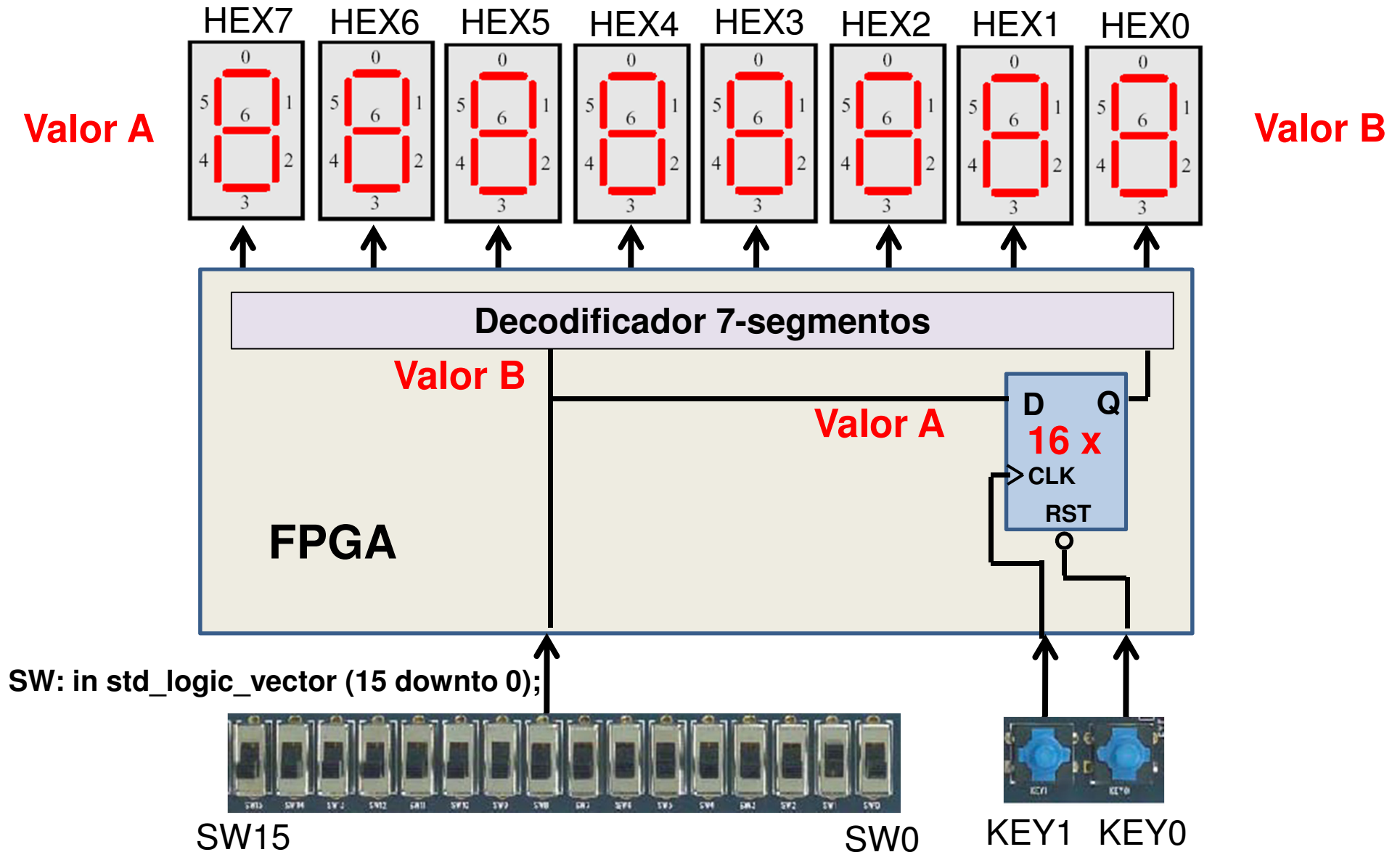


Escrever em HEX3 a HEX0 o “valor B” de 16 bits das chaves SW0 a SW15:



Usar o botão **KEY1** para armazenar o “valor A”, antes de fornecer o “valor B” nas chaves SW0 a SW15.

Tarefa opcional: Sugestão de circuito



Simulação com *ModelSim*

Simulação do projeto com ModelSim

1. Criar uma nova pasta dentro da pasta do projeto.
2. Copiar os **scripts de simulação** disponíveis na página da disciplina para dentro da nova pasta.
3. Entrar na nova pasta, editar o arquivo "*compila.do*", e alterar **lab7.vhd** para o nome do seu arquivo VHDL a ser simulado.
4. Copiar APENAS o seu arquivo VHDL (FFs e *latch*) a ser simulado para a nova pasta, que já deve possuir os **scripts** de simulação copiados da página da disciplina.
5. Executar o *ModelSim-Altera*, que se encontra no menu *Iniciar* do Windows, pasta "*Altera*".
6. No menu "*File*" do *ModelSim*, definir a pasta do projeto (opção "*Change Directory*"), selecionando a nova pasta.

Simulação do projeto com ModelSim (cont.)

7. Execução da simulação (arquivo *compila.do*)
 - a) No menu "Tools" do ModelSim, selecionar "*Tcl*" -> "*Execute Macro*".
 - b) Selecionar o arquivo "*compila.do*", e "*Open*".
8. O *ModelSim* irá compilar os arquivos VHDL e iniciar a simulação.
9. A janela com as formas de onda irá abrir, apresentando o resultado da simulação.

Obs. Se desejar, editar o arquivo *tb.vhd* para alterar a simulação a ser realizada, e repetir o passo 7.

Simulação do projeto com ModelSim (cont.)

Obs. Se o resultado da simulação não estiver de acordo com o esperado, alterar o seu VHDL, salvar, e executar novamente a simulação (arquivo *compila.do*).

Obs. A simulação só irá funcionar se o seu projeto possuir EXATAMENTE a seguinte *entity*:

```
entity lab7 is
    port (D : in std_logic;
          CLK : in std_logic;
          QA : out std_logic;
          QB : out std_logic;
          QC : out std_logic );
end lab7;
```

Simulação do projeto com ModelSim (cont.)

Obs. O arquivo "*compila.do*" contém os comandos do *ModelSim* necessários para realizar a simulação, incluindo:

- a) Criação da biblioteca de trabalho - comando *vlib*.
- b) Compilação dos arquivos VHDL para a biblioteca de trabalho - comando *vcom*.
- c) Inicialização do simulador com o arquivo *testbench* - comando *vsim*.
- d) Execução da janela de formas de onda (*waveform*) - comando *wave*.
- e) Adição dos sinais na janela de formas de onda - comando *wave*.
- f) Execução da simulação - comando *run*.