Escola Politécnica da USP

Departamento de Engenharia de Computação e Sistemas Digitais

PCS3635 – Laboratório Digital I

Turma 01 – Prof. Edson Midorikawa

Relatório da Experiência 2

Circuito em VHDL

Bruno Yukio Fujita Saito **9349631**

Daniel Oliveira de Azambuja **8910198**

Bancada: A-2

Data: 23/01/2017

Sumário

1. Objetivos

Este experimento visa a aplicação do aprendizado de projeto de circuitos lógicos em VHDL realizado em Sistemas Digitais I e II.

Será utilizada uma placa de desenvolvimento FPGA para que seja possível implementar o código e visualizá-lo em operação. Os cógidos e components estudados na experiência anterior, “Primeiro Circuito Digital”, serão também utilizados aqui.

Para realizar as simulações e síntese do circuito, o software para projeto de circuitos digitais Altera Quartus Prime será utilizado. Juntamente com o software Waveforms para gerar o sinal de clock

2. Atividades Pré-Laboratório

As atividades pré-laboratório consistiram no estudo e elaboração dos códigos VHDL que serão utilizados durante a experiência.

2.1. Contador Hexadecimal 74163 em VHDL

Aqui foi feito um estudo mais aprofundado do contador hexadecimal 74163 utilizado na experiência anterior. Desta vez, o foco maior foi na análise de seu código VHDL.

a) Na Figura 1, encontra-se o código VHDL comentado, e a Figura 2 representa o símbolo lógico desse circuito.

```
library IEEE; -- Bibilioteca principal, contem os "packets" de std_logic 1164 e arith.
use IEEE.std_logic_1164.all; -- Traz informações sobre os tipos de variaveis definidas por usuarios, que
-- sao utilizadas no programa ( linha 11 por exemplo, em que RCO é definida como STD_LOGIC ).
use IEEE.std_logic_arith.all; -- Traz informações sobre operações aritmeticas definidas por usuarios, que
-- sao utilizadas no programa ( linha 21 por exemplo, o operador '+' ).

entity V74xl63 is -- Define a entidade, o black box a ser desenvolvido no programa
     port ( CLK, CLR_L, LD_L, ENP, ENT: in STD_LOGIC; -- Define as portas de entrada dessa entidade,
-- definindo as entradas de 1 bit.
    D: in UNSIGNED (3 downto 0); -- Define o barramento de entrada do contador, com o MSB sendo o
-- da esquerda devido a enumeracao (3 downto 0).
    Q: out UNSIGNED (3 downto 0); -- O mesmo que a linha acima, mas define a saida.
    RCO: out STD_LOGIC ); -- Define mais uma saida do contador, a porta RCO.
end V74xl63; -- Fecha a definicao da entidade.

architecture V74xl63_arch of V74xl63 is -- Define a arquitetura (comportamento) da entidade definida
signal IQ: UNSIGNED (3 downto 0); -- Define um sinal interno (intermediario) de mesmo tipo que as entradas e saidas
begin -- Comeca a arquitetura. Grupo de processos, funcoes, eventos e sinais que irao interagir
-- de forma a traduzir o comportamento da entidade.
process (CLK, ENT, IQ) -- Processo sequencial definindo os sinais envolvidos nesse processo
    begin -- Comeca o processo sequencial definido anteriormente.
        if (CLK1'event and CLK='1') then -- Define um laco condicional ativado na borda de subida do clock
            if CLR_L='0' then IQ <= (others => '0'); -- Realiza a acao do clear, zerando IQ quando o CLR_L (ativo baixo)
-- esta em nivel logico baixo.
            elsif LD_L='0' then IQ <= D; -- Realiza a acao do load, passando os valores do barramento de entrada para IQ
-- quando o LD_L (ativo baixo) esta em nivel logico baixo. Caso a operacao de clear nao esteja sendo realizada.
            elsif (ENT and ENP)='1' then IQ <= IQ + 1; -- Define a operacao de contagem quando ambos enables estao ativados.
-- Ou seja, define alem da operacao de contagem, a funcao dos enables. Caso ambas as operacoes de clear e load nao estejam
-- sendo realizadas.
            end if; -- Fecha o laco condicional ativado pelas operacoes definidas anteriormente.
        end if; -- Fecha o laco condicional ativado pela borda de clock, assim definindo um ciclo de operacao do clock.
        if (IQ=15) and (ENT='1') then RCO <= '1'; -- Define quando RCO e ativado (no fim da contagem) e associa o sinal
-- ENT a essa operacao, diferenciando este sinal de ENP.
        else RCO <= '0'; -- Retorna o valor de RCO ao original, quando ja nao ha carry a ser sinalizado.
        end if; -- Fecha o laco condicional definido pelo fim da contagem.
        Q <= IQ; -- Transmite o sinal final interno ao barramento de saida.
    end process; -- Conclui o processo sequencial.
end V74xl63_arch; -- Conclui a arquitetura da entidade, finalizando a definicao de seu comportamento.
```

Figura 1 – Cógido VHDL comentado do contador hexadecimal 74163.

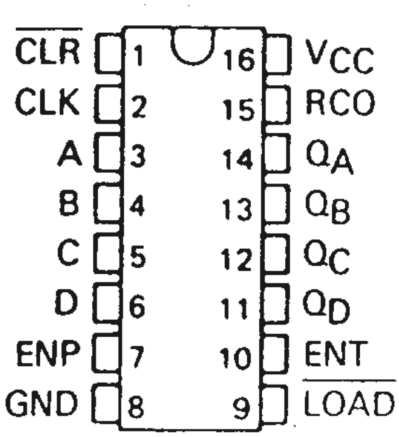

Figura 2 – Símbolo lógico do contador hexadecimal 74163.

b)

1) O UNSIGNED é uma coleção de STD_LOGIC que representa conjuntamente um inteiro, ou seja, possui operações aritméticas definidas para seu tipo, além de utilizar os bits da forma usual e não em um tipo como complemento de 2 em suas operações. Enquanto que o STD_LOGIC_VECTOR é uma coleção de elementos STD_LOGIC que não possuem operações aritméticas definidas e sim operações lógicas.
2) Analisando as linhas 22 e 23, ocorre um laço condicional na linha 22, que avalia se a borda de subida ocorre, e dentro desse laço, na linha 23, ocorre a operação do clear, acionado pelo sinal CLEAR. Portanto o sinal é síncrono.
3) A função que carrega um sinal no 74163 é realizada após afirmar o sinal de LD_L como nível baixo (pois é ativo baixo), como pode ser visto na linha 25, e ela está dentro do laço condicional de borda de subida do clock (linha 22). Assim deve-se carregar o valor desejado no barramento de entrada, ativar LD_L atribuindo nível lógico baixo neste e ativar uma borda de subida de clock.
4) Este componente é sensível à borda de subida do clock, como pode ser avaliado na linha 22, que aguarda um evento de clock e analisa se este está no nível lógico alto.
5) RCO é acionado quando IQ = “15” e ENT está em nível alto, ou seja, no fim da contagem, quando se deseja sinalizar que existe um carry out, como pode ser observado na linha 32 que executa o seguinte comando:

```
if (IQ=15) and (ENT='1') then RCO <= '1';
```

2.2. Registrador com Sinal de Habilitação em VHDL

c) O cógido VHDL do registrador pode ser visto na Figura 3 e seu símbolo lógico correspondente, na Figura 4.

```vhdl
library IEEE; -- -- Bibilioteca principal, contem os "packets" de std_logic 1164.
use IEEE.std_logic_1164.all; -- Traz informações sobre os tipos de variaveis definidas por usuarios, que
-- sao utilizadas no programa ( linha 16 por exemplo, em que CLK é definida como STD_LOGIC ).

entity Vreg16 is -- Definição da entidade do registrador, seu black box.
    port (CLK, CLKEN, OE_L, CLR_L: in STD_LOGIC; -- Define as portas de entrada como STD_LOGIC.
          D: in STD_LOGIC_VECTOR (1 to 16); -- Define o barramento de entrada como STD_LOGIC_VECTOR de 16 bits
          Q: out STD_ULOGIC_VECTOR (1 to 16)); -- Define o barramento de saida como STD_LOGIC_VECTOR de 16 bits
end Vreg16; -- Conclui a definicao das entradas e saidas do registrador e seus tipos.

architecture Vreg16 of Vreg16 is -- Inicia a definicao do comportamento apresentado pela entidade acima definida.
signal CLR, OE: STD_LOGIC; -- Define sinais intermediarios que irao auxiliar na definicao do comportamento do componente.
signal IQ: STD_LOGIC_VECTOR (1 to 16); -- Define o barramento intermediario que ira guardar os valores a serem transferidos dos barramentos
-- de entrada para os de saida.
process(CLK, CLR_L, CLR, OE_L, OE, IQ) -- Define o inicio do processo sequencial que definira o comportamento do componente,
-- e que utilizara os sinais entre parenteses em sua operacao
begin -- Inicia o processo sequencial
    CLR <= not CLR_L; -- Define um sinal interno de CLR (clear) ativo alto.
    OE <= not OE_L; -- Define um sinal interno de OE (output enable) ativo alto.
    if (CLR = '1') then IQ <= (others => '0'); -- Define um laco condicional ativado no sinal de CLR que zera o barramento IQ.
    elsif (CLK'event and CLK='1') then -- Caso CLR nao seja ativado, esta linha busca avaliar se ocorre uma borda de subida do clock para
-- realizar a operacao em seu laco.
        if (CLKEN = '1') then IQ <= D; -- Define a operacao de que se o sinal CLKEN esta em nivel alto, entao o barramento IQ recebe
-- os dados do barramento D.
        end if; -- Fecha o laco condicional aberto na linha 23.
    end if; -- Fecha o laco condicional aberto na linha 20, e indica que ocorreu um ciclo do clock.
    if OE = '1' then Q <= To_StdULogicVector(IQ); -- Converte o resultado em IQ para std_ulogic_vector e carrega a saida Q com esse valor,
-- caso OE esteja ativado em nivel logico alto.
    else Q <= (others => 'Z'); -- Caso contrario a saida Q é carregada como modo Z, de alta impedancia.
    end if; -- Fecha o laco condicional da linha 27.
end process; -- Finaliza o processo sequencial aberto na linha 17.
end Vreg16; -- Fecha a arquitetura definida para a entidade Vreg16.
```

Figura 3 – Cógido VHDL comentado do registrador de 16 bits.

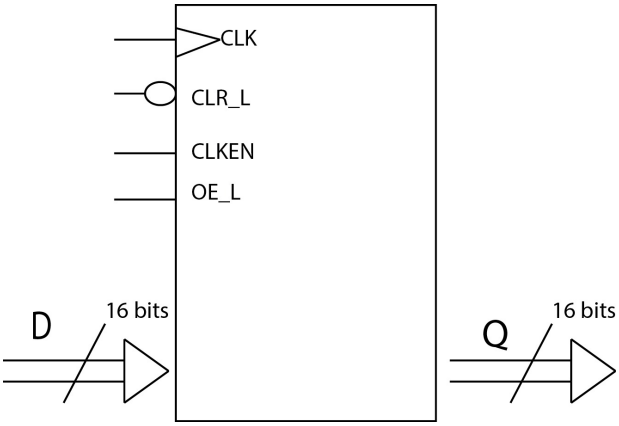

Figura 4 – Símbolo lógico do registrador de 16 bits.

d)

6) No registrador descrito existem 16 bits definidos.
7) Existem 4 sinais de controle:
   - CLK: Sinal que irá definir quando as operações de registro poderão ocorrer, garante a sincronização das operações e com outros circuitos que possam vir a ser associados a este. Demonstrado pela seguinte linha do código:

     elsif (CLK'event and CLK='1') then;
   - OE_L: Sinal que permite a transferência do sinal de barramento interno (IQ) para o barramento de saída (Q). Se não estiver ativado, isto é, se estiver em nível lógico alto, a saída entra em alta impedância. Seu comportamento é descrito pelas seguintes linhas de código:

     if OE = '1' then Q <= To_StdULogicVector(IQ);
     else Q <= (others => 'Z');

- CLKEN: Sinal que permite a transferência do barramento de entrada (D) para o barramento interno, caso ocorra uma borda de subida de clock. Demonstrado na seguinte linha:

  if (CLKEN='1') then IQ <= D;
- CLR_L: Sinal que zera o barramento interno do registrador de forma assíncrona. É o único sinal de controle assíncrono no registrador.

8) A expressão (others=>'0') serve para avisar que todos os bits não nomeados de algum vetor devem receber o nível lógico baixo. Caso fosse necessário que apenas alguns bits recebessem nível alto, poderia ser usada a seguinte expressão:

   (0=>'1', 8=>'1', others=>'0').

   Esta expressão imprime nível lógico alto apenas nos bits 0 e 8.
9) O tipo STD_LOGIC_VECTOR é o padrão da indústria, pois ele resolve os valores existentes no STD_ULOGIC_VECTOR para valores que melhor representam os sinais que ocorrem no circuito real quando há mais de um 'driver' na entrada (mais de um tipo de sinal sendo aplicado na mesma entrada).
10) O comando To_StdULogicVector() é uma função que aceita um vetor STD_LOGIC e devolve o vetor no tipo STD_ULOGIC_VECTOR.
11) O valor Z representa um valor de alta impedância, ou seja, como se o fio estivesse 'cortado'.

e) Baseando-se no código VHDL do item c), na Figura 5 e na Tabela 1, foi desenvolvido o código VHDL da Figura 6, que representa o registrador de 4 bits com sinal de habilitação.

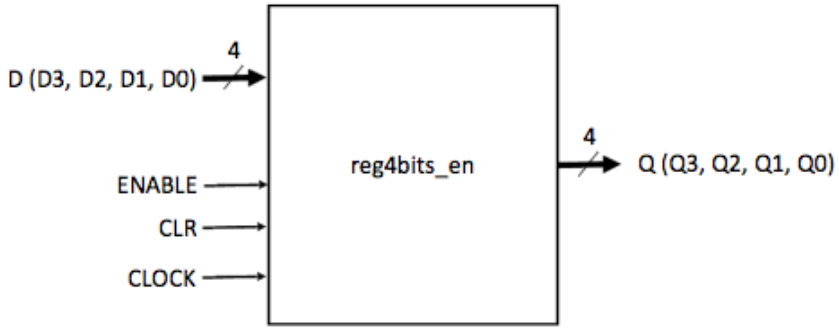

Figura 5 – Símbolo lógico do registrador de 4 bits.

CLR	ENABLE	Q* (próxima saída)	descrição
1	X	0	zera saída
0	0	Q	nenhum sinal de controle ativado
0	1	D	carrega entrada paralela D

Tabela 1 - Descrição dos sinais de controle do registrador de 4 bits.

```vhdl
library IEEE;  -- Bibilioteca principal, contem os "packets" de std_logic 1164.
use IEEE.std_logic_1164.all; -- Traz informações sobre os tipos de variaveis definidas por usuarios, que
-- sao utilizadas no programa.

entity reg4bits_en is -- Define a entidade que representa o registrador
    port (CLOCK, ENABLE, CLR: in STD_LOGIC; -- Define as portas de entrada como STD_LOGIC
          D: in STD_LOGIC_VECTOR (1 to 4); -- Define o barramento de entrada como um STD_LOGIC_VECTOR de 4 bits
          Q: out STD_ULOGIC_VECTOR (1 to 4)); -- Define o barramento de saída como STD_ULOGIC_VECTOR de 4 bits.
end reg4bits_en; -- Conclui a definicao das portas de entrada e saida do componente.

architecture reg4bits_en of reg4bits_en is -- Inicio da definicao do comportamento do registrador.
signal IQ: STD_LOGIC_VECTOR (1 to 4);
begin -- Definicao do sinal interno intermediario, ou seja, os sinais dos flip flops.
process(CLOCK, CLR, ENABLE, IQ) -- Define os sinais que serao utilizados no processo sequencial.
begin -- Inicia o processo descrito anteriormente.
    if (CLOCK'event and CLOCK='1') then -- Verifica se houve borda de subida do clock.
        if (CLR = '1') then IQ <= (others=>'0'); -- Se CLR for 1, o vetor IQ recebe 0.
        elsif (ENABLE = '1') then IQ <= D; -- Se ENABLE for 1, o vetor IQ recebe os sinais do barramento de entrada.
        end if; -- Fecha o laço condicional aberto na linha 16.
        if (CLR = '1' or ENABLE = '1') then Q <= To_StdULogicVector(IQ); -- Caso CLR ou ENABLE forem 1 na saída, Q recebe
-- O valor do barramento intermediario.
        end if; -- Fecha o laço condicional aberto na linha 19.
    end if; -- Fecha o laço condicional aberto na linha 15.
end process; -- Fecha o processo que descreve o comportamento do registrador.
end reg4bits_en; -- Fecha a arquitetura do componente.
```

Figura 6 – Código VHDL comentado do registrador de 4 bits.

2.3. Projeto do Circuito Digital Simples em VHDL

Neste item, foi feita a associação entre o componente umFSM.vhd (máquina de estados descrita pelo código VHDL do Relatório de Planejamento anterior) e o contador hexadecimal 74163 como na última experiência, porém, agora esta associação foi feita através da descrição estrutural em VHDL.

g) A Figura 7 apresenta o código VHDL estrutural e a Figura 8, o esquema interno para a associação já citada anteriormente.

```vhdl
library IEEE;  -- Bibilioteca principal, contem os "packets" de std_logic 1164.
use IEEE.std_logic_1164.all; -- Traz informações sobre os tipos de variaveis definidas por usuarios, que
-- sao utilizadas no programa.

entity ComponenteAssociado is -- Declara a entidade da associacao.
    port ( liga, sinal, RESET, CLOCK : in STD_LOGIC; -- Declara as entradas e saidas, junto de seus tipos.
           RCO : out STD_LOGIC;
           Q : out STD_LOGIC_VECTOR(1 to 4));
end ComponenteAssociado; -- Fecha a declaracao de entidade.

architecture ComponenteAssociado of ComponenteAssociado is -- Inicia a definicao da arquitetura, que ira descrever
-- como o componente se comporta.
signal Resetc, Enablec, LD_L, ENP: STD_LOGIC; -- Indicam os "fios" que iram ligar um componente ao outro.
-- Descreve o componente 74163 que utilizamos na aula passa, como um elemento a ser associado.
component V74x163 port ( CLK, CLR_L, LD_L, ENP, ENT: in STD_LOGIC; RCO : out STD_LOGIC; Q : out STD_LOGIC_VECTOR ); end component;
-- Descreve o componente, maquina de estados, que ira controlar o CI. Declarando as portas de entrada e saida.
component umFSM port ( CLOCK, RESET, liga, sinal: in STD_LOGIC; enablec, resetc: out STD_LOGIC ); end component;
-- Inicia o processo de mapeamento das entradas e saidas.
begin
-- Mapeia as entradas e saidas de um componente ao outro, utilizando os sinais declarados
-- como fios, para ligar estes.
    U1: umFSM port map (CLOCK, RESET, liga, sinal, Enablec, Resetc);
-- Mapeia as entradas e saidas do CI
    U2: V74x163 port map (CLOCK, Resetc, LD_L, ENP, Enablec, RCO, Q);
-- Fecha a descricao do comportamento da associacao
end ComponenteAssociado; -- Fecha a descricao da arquitetura do ComponenteAssociado.
```

Figura 7 – Código VHDL comentado da associação dos components umFSM e contador 74163.

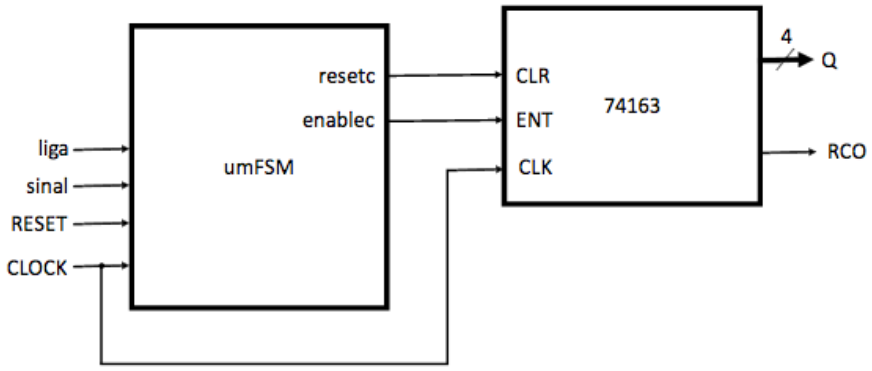

Figura 8 – Esquema interno do circuito interligando os components umFSM e contador 74163.

3. Planejamento da Aula Prática

Neste experimento, há duas atividades a serem realizadas: a simulação do circuito e a síntese na placa FPGA, e a análise de funcionamento do circuito em VHDL.

Estas atividades serão realizadas na ordem listada.

3.1. Simulação do Circuito e Síntese na Placa FPGA

Será feita a simulação no Quartus do circuito digital projetado no item 2.3 para compreender melhor como este software funciona e a forma que ele realiza testbenches. Alguns dos testes feitos na experiência anterior serão repetidos agora para verificar seu comportamento e comparar os resultados obtidos. Tais testes estão listados a seguir:

- LIGA = 1 → SINAL = 1 → SINAL = 0 → LIGA = 1 → SINAL = 1
- LIGA = 1 → SINAL = 1 → SINAL = 0 → LIGA = 0
- LIGA = 0 → LIGA = 1 → SINAL = 0 → SINAL = 1 → SINAL = 1 → SINAL = 0 → LIGA = 0
- LIGA = 1 → SINAL = 1 → SINAL = 0 → LIGA = 1 → SINAL = 0 → SINAL = 1

As formas de ondas obtidas com os testes realizados serão anotadas no Relatório.

Em seguida, será feita a preparação para síntese do circuito na placa de desenvolvimento FPGA. Para isso, a Tabela 2 indica a designação dos pinos a serem conectados, de forma que seja possível avaliar as saídas e controlar seus valores a partir das chaves e LEDs.

Designação		Sinal externo
sinal	pino	
CLOCK	GPIO_0_D0	saída digital 0 (Analog Discovery)
RESET	chave SW9	-
liga	chave SW0	-
sinal	chave SW1	-
resetc	*led* LEDR9	-
enablec	*led* LEDR8	-
Q	*leds* LEDR0 a LEDR3	-
RCO	*led* LEDR4	-

Tabela 2 – Indicação das ligações a serem feitas para preparação para a síntese do circuito.

Por fim, iremos verificar a saída do Quartus a fim de analisar se a síntese foi feita corretamente.

3.2. Análise de Funcionamento do Circuito em VHDL

Com a síntese concluída no item anterior, o circuito será implementado na placa FPGA utilizando o Quartus. Também será usado o software Waveforms para gerar um clock periódico de frequência 1 Hz no sinal de saída digital 0.

Agora que o circuito está implementado, os testes citados anteriormente e mais alguns serão realizados novamente e todos seus resultados serão anotados no Relatório:

- LIGA = 1 → SINAL = 1 → SINAL = 0 → LIGA = 1 → SINAL = 1
- LIGA = 1 → SINAL = 1 → SINAL = 0 → LIGA = 0
- LIGA = 0 → LIGA = 1 → SINAL = 0 → SINAL = 1 → SINAL = 1 → SINAL = 0 → LIGA = 0
- LIGA = 1 → SINAL = 1 → SINAL = 0 → LIGA = 1 → SINAL = 0 → SINAL = 1
- Acionar chave SW1 (SINAL) por 5 segundos e verificar a saída nos LEDs LEDR0 a LEDR3.
- Desativar chave SW0 (LIGA) antes da chave SW1 e verificar o que ocorre.
- Verificar a saída do contador quando a chave SW1 fica acionada por mias de 15 segundos.
- Verificar o que acontece quando a chave SW1 é acionada várias vezes por intervalo de tempos diferentes.