**Departamento de Engenharia Elétrica e de Computação**

# SEL 384 – Lab. de Sistemas Digitais I

**Profa. Luiza Maria Romeiro Codá**

**PRÁTICA Nº10b**

**Contador Binário – Usando Template do QUARTUS II**

Utilizando o software QUARTUS II, faça os projetos em VHDL dos itens a seguir. Para tal, utilize o t*emplate* do Quartus abaixo. O contador do *template* tem os seguintes pinos:

Obs: criar as saídas como OUT e usar sinal auxiliar para trabalhar internamente ao PROCESS.

**DATA**: Entrada que corresponde ao valor carregado nas saídas dos FFs dos contadores para iniciar a contagem a partir do valor “Data”.

**LOAD**: Entrada que habilita o carregamento da entrada “Data”.

**CLEAR**: Entrada que zera a saída.

**CLOCK**: Entrada. A cada pulso a contagem é incrementada.

**ENABLE**: Entrada habilitadora do *clock*.

**CONT\_OUTPUT**: Saídas Q dos FFs.

1. Utilizando o template acima faça o projeto em VHDL de um contador binário de (0000)b até (1111)b com apenas as entradas *clear* assíncrona e *clock*. Desabilite as entradas que não são desejadas, deixando na descrição como comentário, caso necessite destas entradas posteriormente. Simule e gere os RTL.
2. Inicie outro projeto, aproveitando o projeto do item 1, e o modifique para que o contador se torne um contador de década, ou seja, a saída apresente números binários de 0 a 9. Simule e gere os RTL.
3. Faça o projeto de um divisor de frequência para obter 1 Hz a partir da frequência do *clock* da placa, que é de 50 MHz. Simule no Modelsim externo ao Quartus e gere o RTL

Como relatório entregar o texto do arquivo VHDL com comentários suficientes para a compreensão do projeto, contendo o nome e nº USP do alun@, as figuras do RTL e das formas de onda da simulação

Obs: Verifique o funcionamento do divisor de frequência considerando uma frequência de entrada de 10Hz. ( frequências altas , 50MHz, não podem ser simuladas pelo University Program VWF

**TEMPLATE CONTADOR:**

ENTITY \_\_entity\_name IS

 PORT

 (

 \_\_data\_input\_name : IN INTEGER RANGE 0 TO \_\_count\_value;

 \_\_clk\_input\_name : IN BIT;

 \_\_clrn\_input\_name : IN BIT;

 \_\_ena\_input\_name : IN BIT;

 \_\_ld\_input\_name : IN BIT;

 \_\_count\_output\_name : OUT INTEGER RANGE 0 TO \_\_count\_value

 );

END \_\_entity\_name;

ARCHITECTURE a OF \_\_entity\_name IS

 SIGNAL \_\_count\_signal\_name : INTEGER RANGE 0 TO \_\_count\_value;

BEGIN

 PROCESS (\_\_clk\_input\_name, \_\_clrn\_input\_name)

 BEGIN

 IF \_\_clrn\_input\_name = '0' THEN

 \_\_count\_signal\_name <= 0;

 ELSIF (\_\_clk\_input\_name'EVENT AND \_\_clk\_input\_name = '1') THEN

 IF \_\_ld\_input\_name = '1' THEN

 \_\_count\_signal\_name <= \_\_data\_input\_name;

 ELSE

 IF \_\_ena\_input\_name = '1' THEN

 \_\_count\_signal\_name <= \_\_count\_signal\_name + 1;

 ELSE

 \_\_count\_signal\_name <= \_\_count\_signal\_name;

 END IF;

 END IF;

 END IF;

 END PROCESS;

END a;