Contador BCD genérico

Eletrônica Digital II

Bernardo Souza Muniz

18 de Maio de 2025

Engenharia de Telecomunicações - IFSC-SJ

Sumário

1. Introdução

O objetivo deste documento é apresentar um código em ***VHDL (VHSIC Hardware Description Language)*** que realiza a contagem genérica, permitindo definir qualquer número de dígitos BCD de 1 até N. O código foi estruturado estruturado em dois segmentos: um de lógica sequencial e outro de lógica combinacional. Além do desenvolvimento do código, são apresentados dados de verificação de desempenho, como a frequência máxima (Fmax), número de pinos utilizados e quantidade de elementos lógicos consumidos. A plataforma de desenvolvimento utilizada foi o Quartus Prime, e a simulação foi realizada com o ModelSim.

2. Código BCD

O código de contagem genérica é dividido em dois blocos principais: um responsável pela lógica sequencial (State Register) e outro pela lógica combinacional (Next State Logic). A parte sequencial tem o objetivo de atualizar todas as mudanças feitas nas casas de unidade, dezena, centena, unidade de milhar e etc. Já a parte combinacional tem objetivo de calcular as mudanças em cada casa de contagem, fazendo incrementos a partir de um condicionais. O objetivo final é que o contador realize a contagem de acordo com a quantidade de casas arbitrada pelo usuário na variável genérica.

2.1. Parte Sequencial

O primeiro segmento do código se trata do State register, que é sensível ao clock e ao reset. O objetivo do State Register é resetar todas as unidades do vetor de contagem para "0" através de um loop quando a variável ***enable*** for igual a "1". Caso não estiver ativo, o processo só executa durante uma borda de subida (rising edge) do clock. Durante o clock os valores do próximo estado são armazenados nos registradores.

```
SEQ: process(clk, reset)
begin
    if reset = '1' then
        for i in estado_atual'range loop
            estado_atual(i) <= 0;
        end loop;
    elsif rising_edge(clk) then
        if enable = '1' then
            estado_atual <= proximo_estado;
        end if;
    end if;
end process SEQ;
```

2.2. Parte Combinacional

O segundo segmento se baseia no cálculo do próximo estado. Dentro desete processo, a a variável ***proximo estado*** é atualizada quando o enable estiver ativo. A atualização de variáveis utiliza uma função descrita no arquivo ***package***.

```
COMB: process(estado_atual, enable)
begin
    proximo_estado <= estado_atual;

    if enable = '1' then
        proximo_estado <= incrementa_bcd(estado_atual);
    end if;

    bcd_out <= estado_atual;
end process COMB;
```

3. Contador BCD genérico

3.1. Contador para 3 dígitos

Para fazer a contagem de um número de 3 dígitos, basta alterar a variável genérica NUM_DIGITOS para 3. Dessa forma o contador irá fazer a contagem de 000 até 999.

O código abaixo exemplifica as alterações feitas no arquivo principal.

```
entity bcd_generico is
generic (
    NUM_DIGITOS : positive := 3
);
```

Ao fazer a análise de síntese, observar a quantidade de elementos lógicos, o tempo de FMAX e o diagrama do RTL Viewer, foi obtido os seguintes resultados:

Figura 1: Relatório de compilação para 3 dígitos

Flow Status	Successful - Sun May 18 11:35:04 2025
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Standard Edition
Revision Name	pkg_contador_bcd
Top-level Entity Name	bcd_generico
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	16 / 6,272 (< 1 %)
Total registers	12
Total pins	15 / 92 (16 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

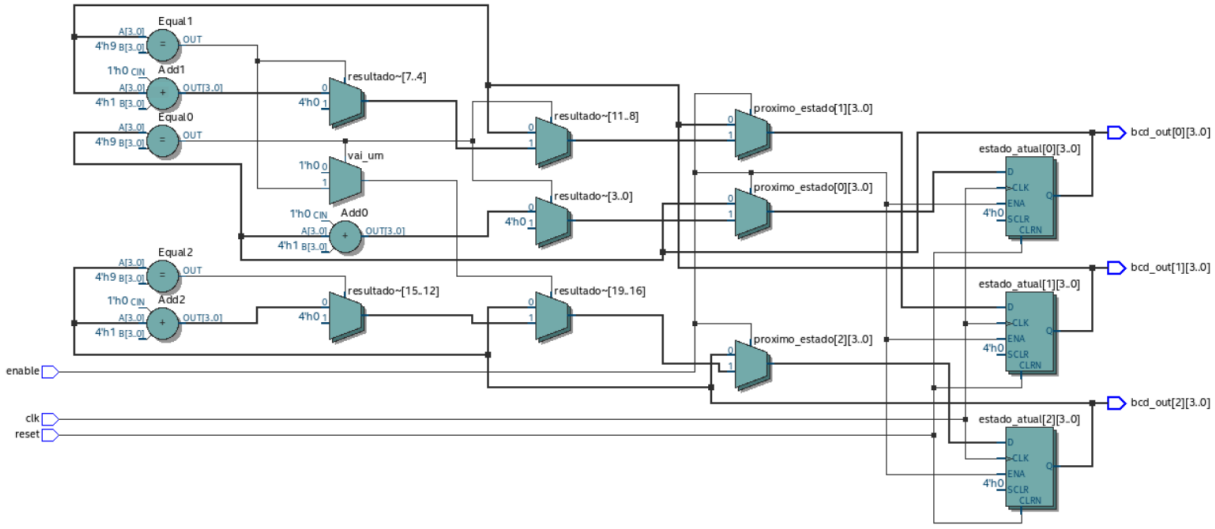
Fonte: Elaborado pelo autor

Figura 2: Tempo de Fmax para 3 dígitos

	Fmax	Restricted Fmax	Clock Name	Note
1	353.48 MHz	250.0 MHz	clk	limit due to minimum pe...n (max I/O toggle rate)

Fonte: Elaborado pelo autor

Figura 3: RTL Viewer do contador de 3 dígitos



Fonte: Elaborado pelo autor

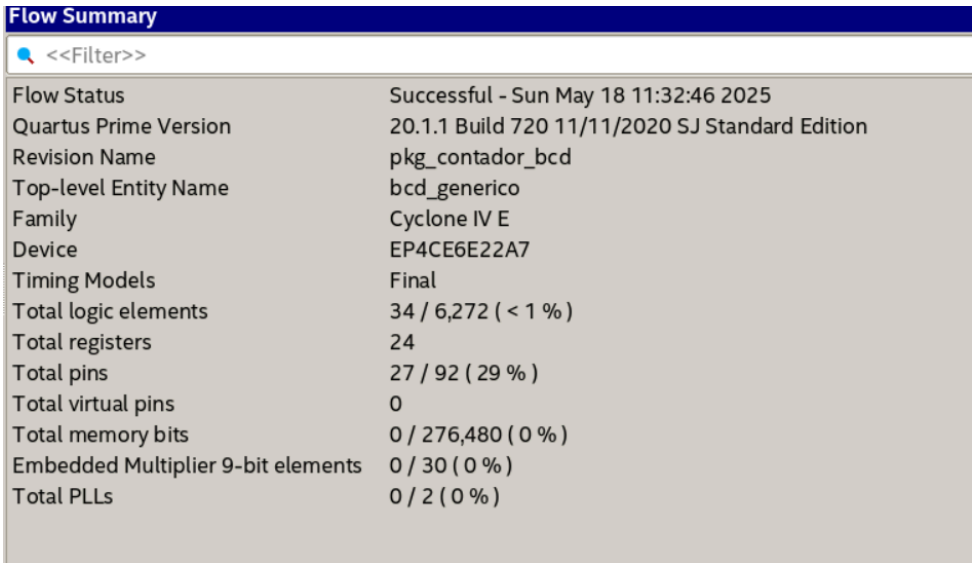
3.2. Contador para 6 dígitos

Para fazer a contagem de um número de 6 dígitos, basta alterar a variável genérica NUM_DIGITOS para 6. Dessa forma o contador irá fazer a contagem de 000000 até 999999.

O código abaixo exemplifica as alterações feitas no arquivo principal.

```
entity bcd_generico is
generic (
    NUM_DIGITOS : positive := 6
);
```

Figura 4: Relatório de compilação para 6 dígitos



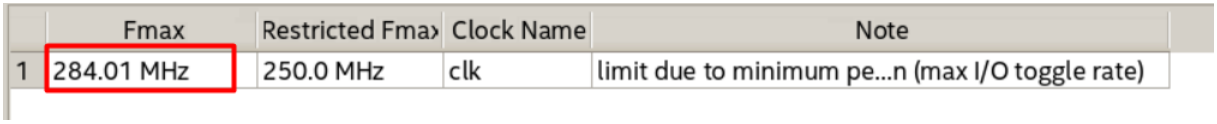
Flow Summary

<<Filter>>

Flow Status	Successful - Sun May 18 11:32:46 2025
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Standard Edition
Revision Name	pkg_contador_bcd
Top-level Entity Name	bcd_generico
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	34 / 6,272 (< 1 %)
Total registers	24
Total pins	27 / 92 (29 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

Fonte: Elaborado pelo autor

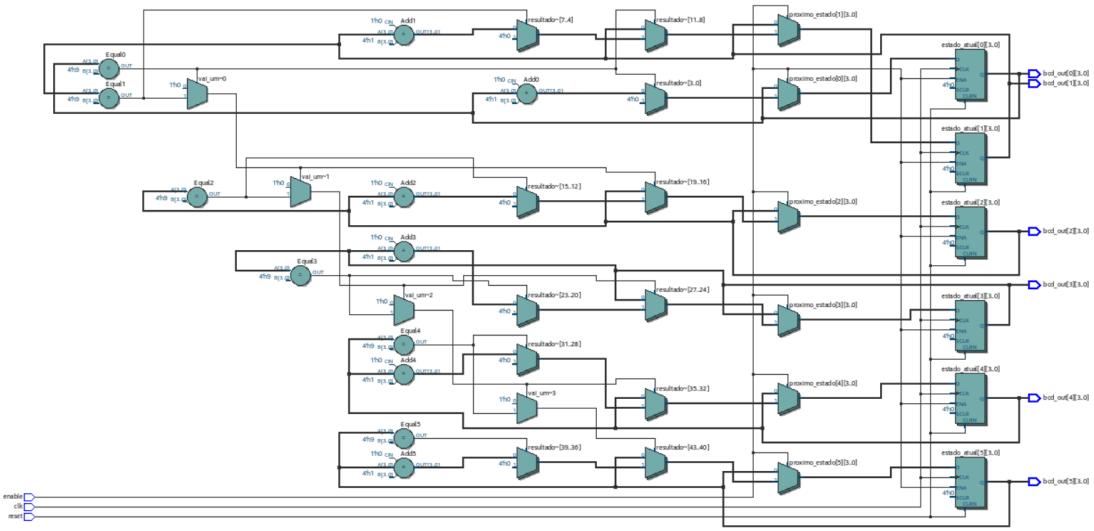
Figura 5: Tempo de Fmax para 6 dígitos



	Fmax	Restricted Fmax	Clock Name	Note
1	284.01 MHz	250.0 MHz	clk	limit due to minimum pe...n (max I/O toggle rate)

Fonte: Elaborado pelo autor

Figura 6: RTL Viewer do contador de 6 dígitos



Fonte: Elaborado pelo autor

3.3. Tabela de comparação entre BCD de 3 e 6 dígitos

A Tabela 1 apresenta um resumo das principais características obtidas após a compilação dos dois circuitos BCD desenvolvidos. Esses dados foram extraídos diretamente do relatório de compilação do Quartus, com o objetivo de facilitar a análise comparativa quanto à utilização de recursos lógicos e temporização.

Tabela 1: Elaborada pelo Autor

Parâmetros	Circuito 1 (BCD com 3 dígitos)	Circuito 2 (BCD com 6 dígitos)
Elementos lógicos	16	34
Pinos	15	24
Registers	12	27
Fmax	353,48 MHz	284,01 MHz

Tabela de resultados de compilação para o contador de 6 dígitos

Nota-se que houve uma grande diferença principalmente na quantidade de flip-flops (registers) , elementos lógicos e no tempo de Fmax de ambos os projetos.

4. Simulação funcional do circuito do contador BCD genérico

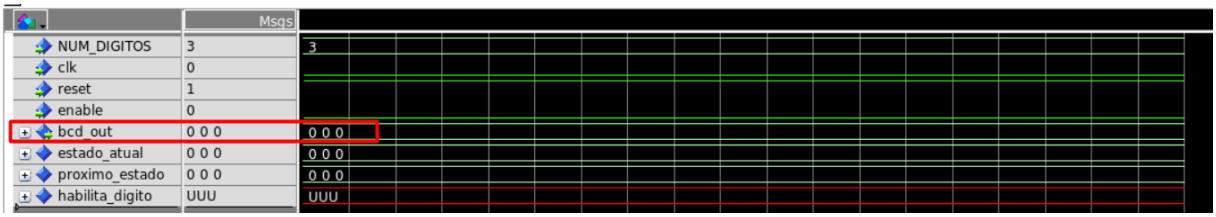
Para fazer a simulação funcional do circuito, foi utilizado um contador para N = 3 dígitos BCD. Além disso, foi feito a verificação de Overflow após 999.

O código a seguir demonstra os comandos utilizados para a configuração do circuito:

```
force clk 0
force reset 1
force enable 0
run 10 ns
force reset 0
force enable 1
force clk 0, 1 5 ns -repeat 10 ns
run 10000 ns
```

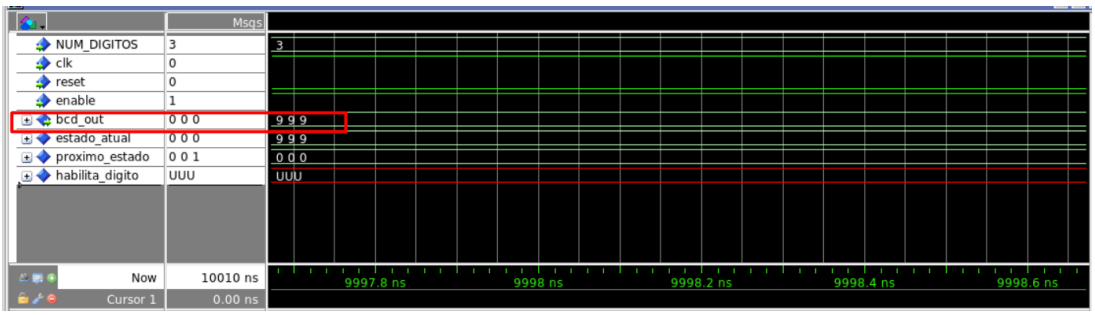
Uma vez configurado o compilador, foi feito a captura dos resultados obtidos. As figuras abaixo demonstram os detalhes da simulação que mostram as contagem das Unidades, Dezenas, Centenas e também o Overflow.

Figura 7: Varíaveis iniciais para a contagem de dígitos



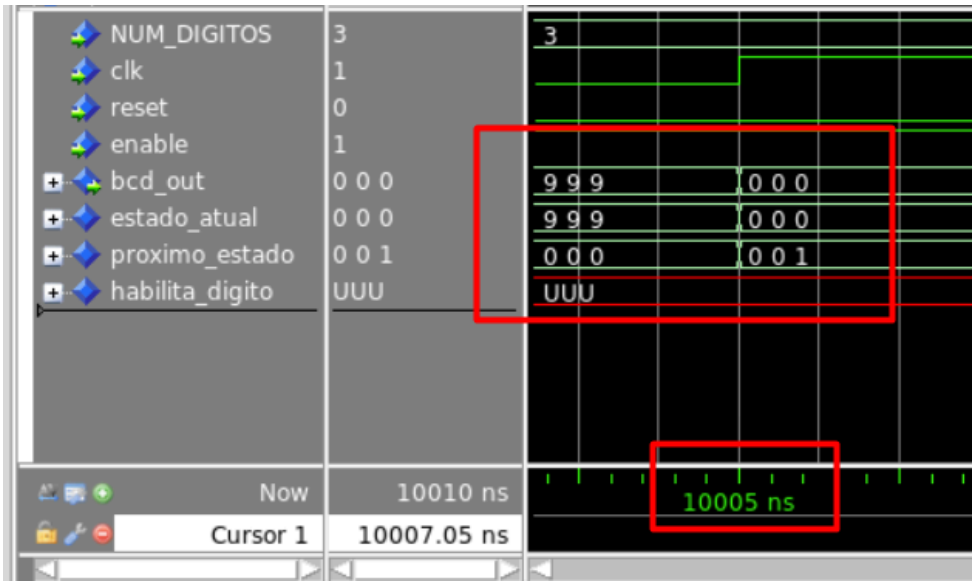
Fonte: Elaborado pelo autor

Figura 8: Varíaveis finais para a contagem de dígitos



Fonte: Elaborado pelo autor

Figura 9: Overflow



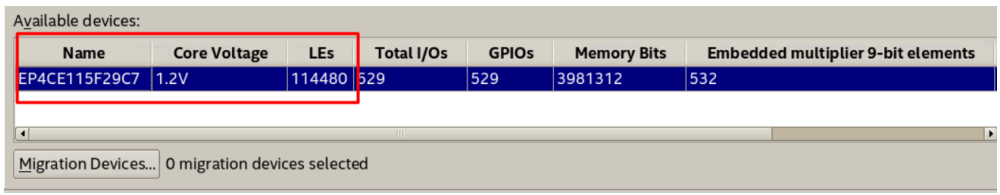
Fonte: Elaborado pelo autor

Como nota-se na figura 9, a mudança de 999 para 000 novamente ocorreu exatamente em 10005 ns na linha de tempo do simulador. Além disso as figuras 7 e 8 mostram que o contador iniciou sua contagem em 000 e terminou em 999, o que confirma os resultados esperados para o contador.

5. Implemente o contador com o divisor de clock no kit

Para a implementação do circuito feito no Kit DE2-115 foi utilizado a família **Cyclone IV E** e Filter **EP4CE115F29C7**.

Figura 10: Device Family



Fonte: Elaborado pelo autor

Foi observado no que a família escolhida possui uma tensão de 1.2V de Core Voltage e 114480 de LEs.

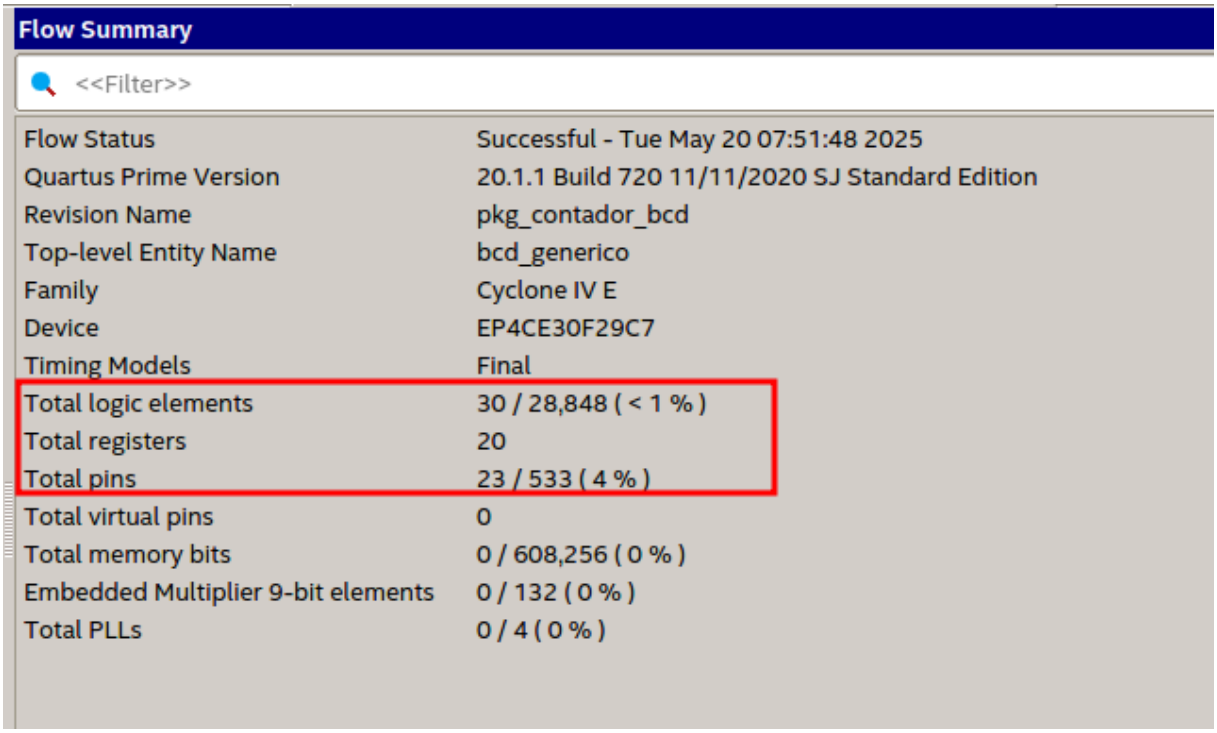
5.1. Restringindo a frequência máxima de Clock

Para restringir a frequência máxima de clock, foi utilizado um arquivo **.sdc** contendo o seguinte comando:

```
1 create_clock -name CLK50MHz -period 50MHz [get_ports -no_case {clk*}]
```

Após fazer a compilação do código, foi obtido os seguintes resultados:

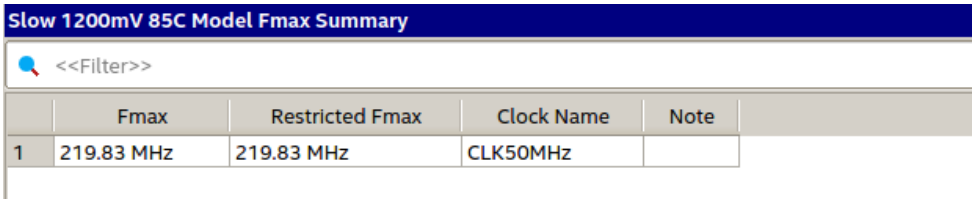
Figura 11: Sumário de compilação



Fonte: Elaborado pelo autor

Uma vez verificado a quantidade de pinos e elementos lógicos do circuito, foi observado a frequência do Fmax do projeto:

Figura 12: Fmax do projeto



Fonte: Elaborado pelo autor

A tabela abaixo faz uma relação com as principais informações presentes no circuito:

Tabela 2: Elaborada pelo Autor

Compilação	Fmax (MHz)	Elementos lógicos	Pinos	Registers
Primeira compilação	219,83	30	23	20

Tabela de resultados de compilação

6. Conclusão

Conclui-se que a utilização de uma estrutura de projeto segmentada proporcionou resultados significativamente mais consistentes em comparação à abordagem com apenas um segmento. Os objetivos definidos no início do projeto foram plenamente alcançados, resultando na implementação bem-sucedida de um contador BCD genérico de 1 até N.