Isabela Bella Bortoleto
Nícolas Auersvalt Marques

Relatório 3

Relatório à disciplina de Circuitos Digitais,
como requisito parcial para aprovação.

Docente: Jamil de Araujo Farhat

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Agosto de 2025

Sumário

1 Introdução

Nesta atividade, utiliza-se um *Complex Programmable Logic Device* (*CPLD*) da família *CPLD* Altera MAX II (*EPM240T100C5N*) para a implementação de circuitos digitais. O *CPLD* está integrado em uma placa de circuito impresso com tecnologia *THT* (*Through-Hole Technology*), que inclui recursos como um display de 7 segmentos para a visualização das saídas.

Para o planejamento e validação dos circuitos lógicos, foi empregado o *software Logisim-Evolution (v3.7.2)*, que também foi utilizado para gerar um bloco de descrição de hardware (*HDL*). O código exportado no formato *VHDL* (*VHSIC Hardware Description Language*) foi então compilado e embarcado no *CPLD* com o auxílio do *software Quartus II*.

2 Exercício 1 (Decodificador e RA)

2.1 Simulação

O circuito proposto no exercício foi modelado e teve seu funcionamento validado por meio de simulação, em conformidade com a aula [1]. Tendo em vista que o hexadecimal E representa o número 6 e o hexadecimal F mostra a letra P. A análise do comportamento lógico do circuito seguiu a Tabela 1.

Tabela 1: Tabela verdade do decodificador de 7 segmentos

K	L	M	N	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0	1	1	1

Portanto, a simulação feita gerou os seguintes resultados:

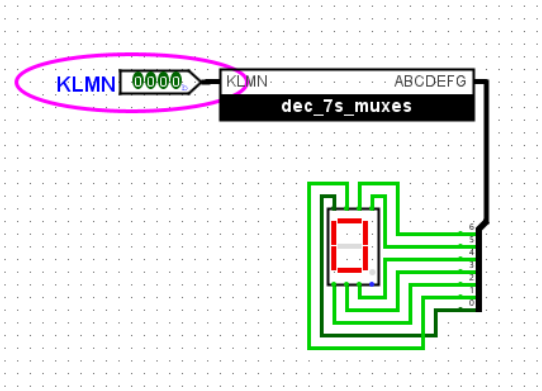

Figura 1: *Input* 0000 mostrando 0 no display.

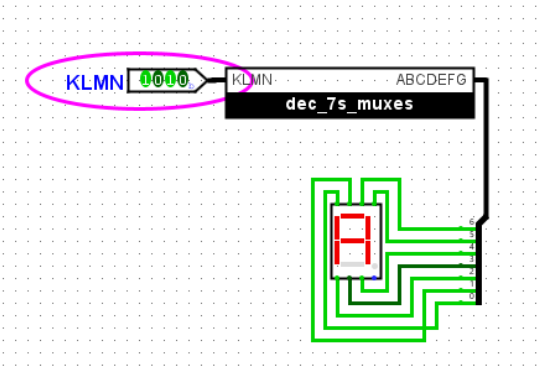

Figura 2: *Input* 1010 mostrando A no display.

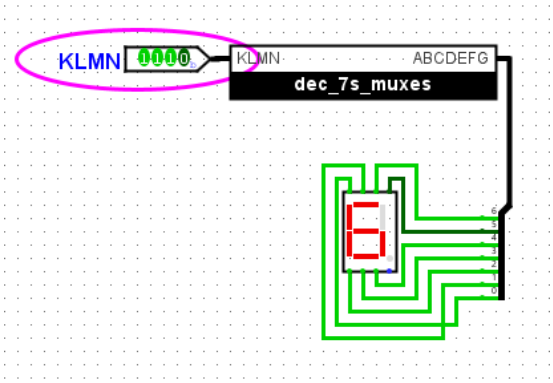

Figura 3: *Input* 1110 mostrando o último dígito do RA (6) no display.

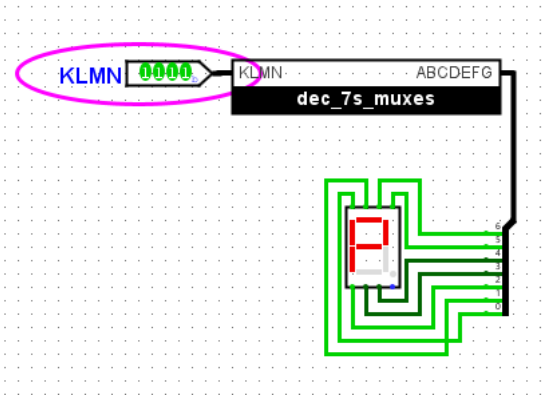

Figura 4: *Input* 1111 mostrando a letra P, pois 6 é par.

2.2 *Quartus*

O código *VHDL* exportado do *Logisim-evolution* foi compilado no *Quartus*. A seguir, foram analisadas a lógica sintetizada (*RTL*) e a simulação funcional para verificação do projeto. A Figura 5 apresenta a visualização *RTL*, detalhando a estrutura do circuito implementado pelo software.

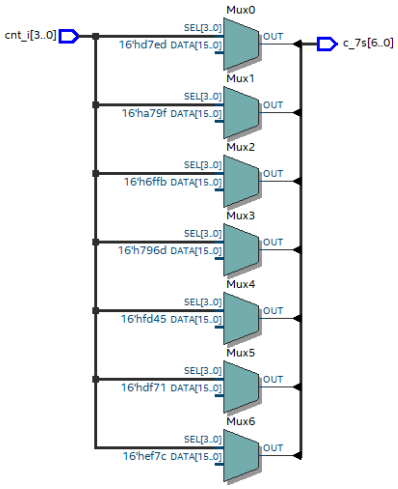

Figura 5: Visualização RTL do circuito.

Para validar o comportamento, a simulação funcional foi executada. As formas de onda resultantes (Figura 6) confirmam que a lógica está correta, pois as saídas correspondem aos valores de entrada esperados.

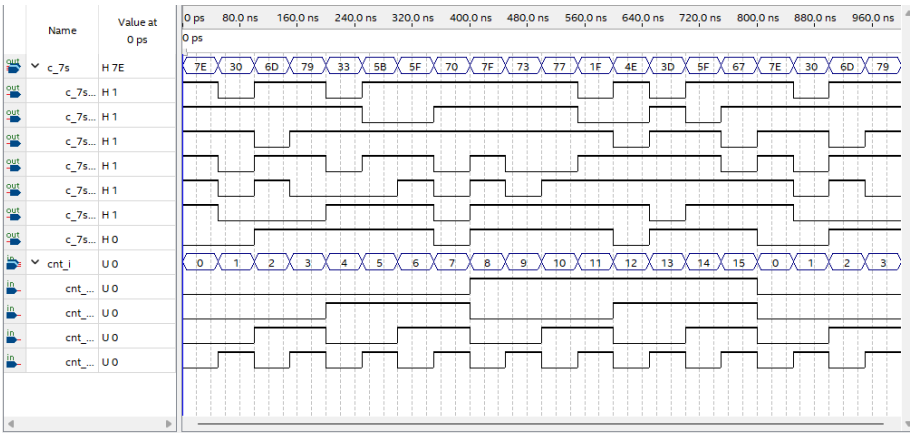

Figura 6: Visualização das formas de onda resultantes do circuito.

O código *VHDL* exportado da simulação foi compilado e embarcado no *CPLD*. Os testes na placa apresentaram os seguintes resultados:

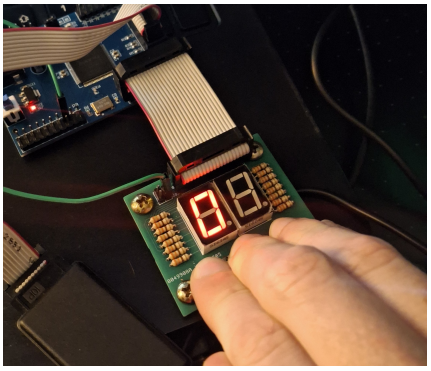

Figura 7: *Input* 0000 mostrando 0 no display.

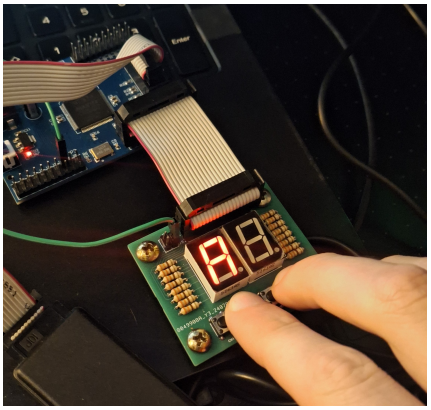

Figura 8: *Input* 1010 mostrando A no display.

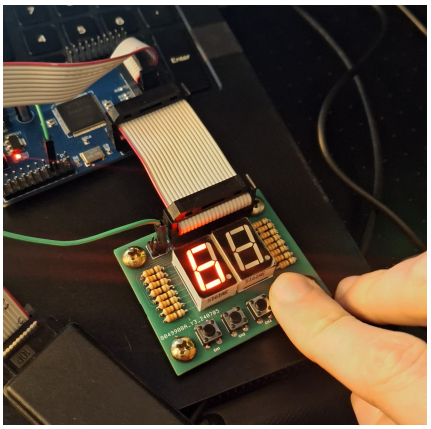

Figura 9: *Input* 1110 mostrando o último dígito do RA (6) no display.

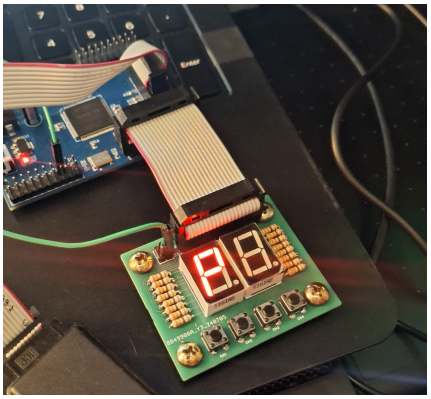

Figura 10: *Input* 1111 mostrando a letra P, pois 6 é par.

3 Exercício 2

3.1 Simulação

O circuito proposto no exercício foi modelado e teve seu funcionamento validado por meio de simulação, em conformidade com a aula [1]. Para este exercício, o decodificador desenvolvido anteriormente foi complementado para avançar uma posição a cada pulso.

O objetivo da simulação foi validar o comportamento sequencial do novo circuito. A análise confirmou que, a cada pulso de *clock* (”clk”), a saída do contador era incrementada e o *display* avançava para a próxima posição, conforme o esperado.

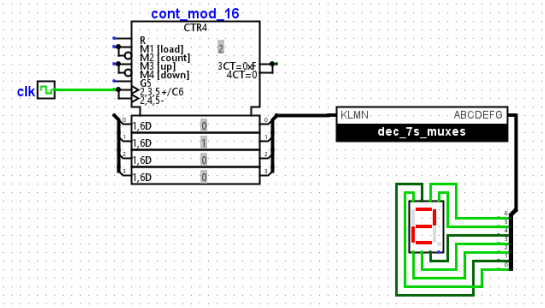

Figura 11: Circuito em funcionamento no *Logisim-evolution*.

3.2 Prática

Na implementação VHDL, o decodificador anterior foi integrado a um contador de 4 bits gerado pelo *IP Catalog* da Altera (módulo *LPM_COUNTER*). O componente resultante foi conectado via *port map* ao decodificador, e o projeto completo foi compilado para os testes.

A Figura 12 apresenta a visualização *RTL*, detalhando a estrutura do circuito implementado pelo software.

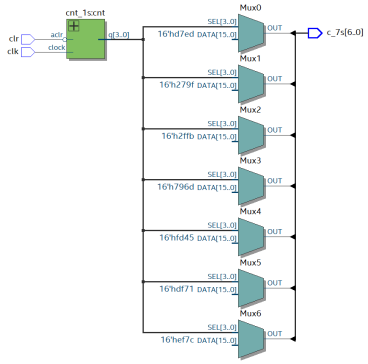

Figura 12: Visualização RTL do circuito.

Para validar o comportamento, a simulação funcional foi executada. As formas de onda resultantes (Figura 13) confirmam que a lógica está correta, pois as saídas correspondem aos

valores de entrada esperados.

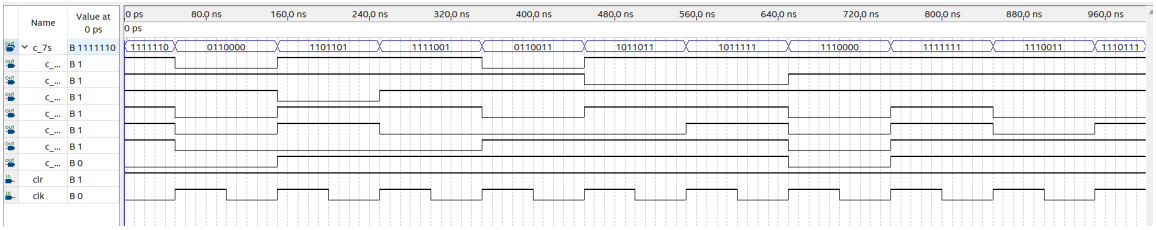

Figura 13: Visualização das formas de onda resultantes do circuito.

Após a implementação do projeto em VHDL, o código foi compilado e embarcado no *CPLD*. A análise confirmou que, a cada pulso de *clock* (”clk”), a saída do contador era incrementada e o *display* avançava para a próxima posição, conforme o esperado. Todavia, foi observado que, ao pressionar o botão de pulso por tempo prolongado, o contador pode avançar múltiplas unidades em vez de apenas uma, comportamento que pode ser atribuído ao efeito de *bouncing* do componente físico. Para resolver a problemática, é possível implementar um circuito anti-repique, conforme [2].

A escolha de uma entrada de *clear* assíncrona (‘aclr‘) para o contador garante que o reset do circuito ocorra imediatamente ao ser acionado, sem esperar por um pulso de *clock*. Embora sistemas assíncronos possam introduzir complexidades em projetos maiores, para uma função de reset manual, essa abordagem oferece uma resposta instantânea e eficaz.

4 Exercício 3

Utilizando-se do que foi realizado nos exercícios anteriores, o objetivo é criar um divisor de *clock* em *VHDL* para converter a frequência de 50 MHz do *EPM240* para 1 Hz, fazendo com que o *display* do exercício anterior avance automaticamente a cada segundo. A implementação sugerida é usar um *LPM_COUNTER* com módulo 50.000.000, utilizando sua saída de *Carry-out* para gerar o pulso de 1 Hz.

A Figura 14 apresenta a visualização *RTL*, detalhando a estrutura do circuito implementado pelo software.

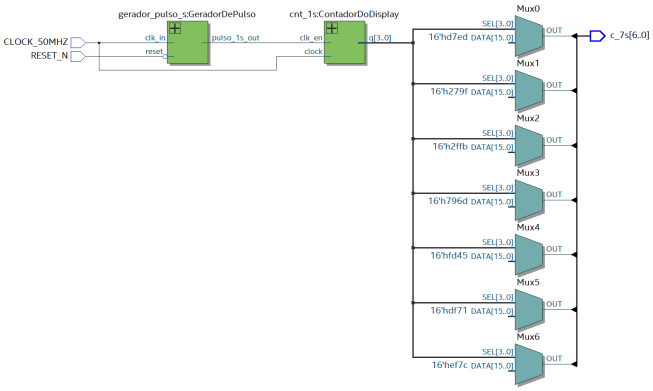

Figura 14: Visualização RTL do circuito.

Para validar o comportamento, a simulação adaptada foi executada. As formas de onda resultantes (Figura 15) mostram que o *VHDL* foi alterado para contar apenas de 1 até 2, pois a contagem de 1 até 50 milhões ultrapassava a escala do simulador, limitada ao intervalo de 0 até 100 μs. Como o período do clock é $T = \frac{1}{50 \times 10^6} = 20\ ns$, esse valor não era suficiente para atualizar o `c_7s`.

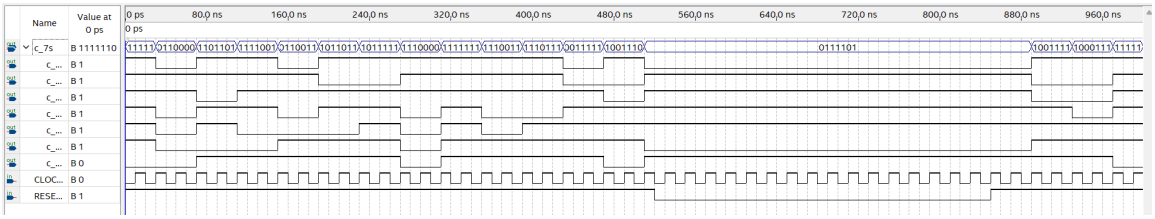

Figura 15: Visualização das formas de onda resultantes do circuito.

Após a implementação do projeto em VHDL, o código foi compilado e embarcado no *CPLD* e apresentou os resultados esperados.

Afinal, o divisor de frequência funciona, essencialmente, como um contador que vai de 1 até 50.000.000. Como o clock do CPLD é de 50 MHz, este processo leva exatamente um segundo para ser concluído. Ao final da contagem, o contador emite um único pulso em sua saída, que serve como o sinal de *clock* para o circuito do exercício anterior, atualizando o valor no display.

Alterar o limite de contagem impactaria diretamente o período do pulso; por exemplo, um limite de 60.000.000 resultaria em um período de 1,2 segundos.

Referências

[1] Jamil Farhat. *Circuitos Digitais - Laboratório 3*. Material de aula. Baseado nos slides elaborados pelos Profs. Peron e Gortan. Paraná, Brasil, 2024.

[2] Antonio Gortan. *Tutorial – Circuito Anti-repique em VHDL*. Tutorial de apoio. Material para a disciplina de Circuitos Digitais. 2024.