

Experimento 6

Implementação de Circuitos Combinacionais com Multiplexadores

Matheus Cardoso de Souza, 202033507
Ualiton Ventura da Silva, 202033580
Grupo G42

¹Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0231 - Laboratório de Circuitos Lógicos

matheus-cardoso.mc@aluno.unb.br, 202033580@aluno.unb.br

Abstract. *The current report aims to show the elaboration, implementation and analysis of logic circuits using multiplexers and decoders as main the main logic components. Furthermore, it was explored the use of more advanced tools to describe the hardware, using the SystemVerilog language for more assertive, practical and scalable representation of complex systems.*

Resumo. *O presente relatório tem como objetivo a elaboração, implementação e análise de circuitos lógicos utilizando-se multiplexadores e decodificadores como os principais componentes lógicos. Além disso, foi explorado o uso de ferramentas mais avançadas para a descrição de hardware, valendo-se da linguagem SystemVerilog para uma representação assertiva, prática e escalável de sistemas complexos.*

1. Introdução

Como nesse relatório faremos uso de forma extensa de multiplexadores e decodificadores, cabe inicialmente uma explanação breve sobre esses circuitos lógicos, bem como suas propriedades e aplicações.

Multiplexadores (ou *mux*) são circuitos combinacionais capazes de mapear 2^N entradas para apenas 1 saída, sendo, portanto, conhecidos também como *seletores de dados*. No caso, quando existem 2^N dados de entrada, é requerido um total de N seletores para satisfatoriamente cumprir o mapeamento $2^N \rightarrow 1$.

Na figura 1 (fonte: [Lamar 2021a]) é possível entender conceitualmente o funcionamento de um multiplexador, e, na figura 2 (fonte: [Wikipedia 2021]) é possível visualizar com maiores detalhes a implementação por circuitos lógicos de um multiplexador.

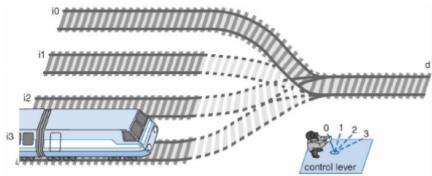


Figura 1. Como um multiplexador funciona

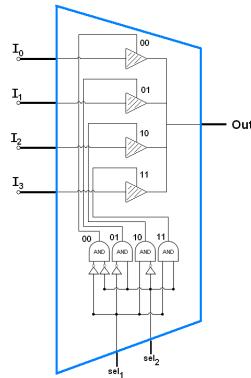


Figura 2. Implementação de um multiplexador

Um dos usos de multiplexadores é na economia de conexões ao longo de um único canal, conectando o único output do multiplexador com o único input do demultiplexador. Dessa forma, o benefício é que seria possível reduzir o custo de implementação física de múltiplos canais para cada uma das fontes de dados no circuito. Pode-se observar como funcionaria um circuito desse na figura 3



Figura 3. Uma das aplicações de um Multiplexador

Considerando agora os demultiplexadores, podemos afirmar que eles são basicamente a “*função inversa*” dos multiplexadores, convertendo 1 input em 2^N outputs, e possuindo N seletores. O diagrama básico de um demultiplexador pode ser visto na figura 4 (fonte: [Electrically4u 2021]).

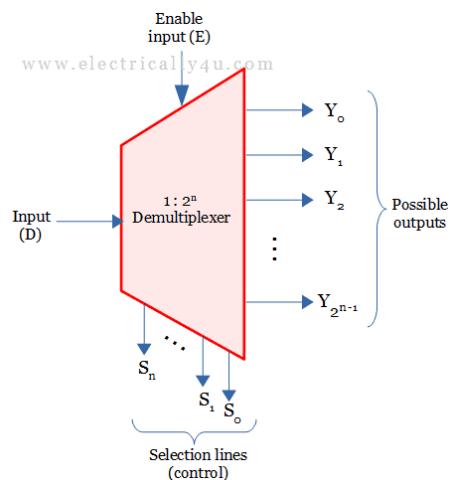


Figura 4. Diagrama de um demultiplexador

Por fim, os decodificadores. Podemos dizer que são muito semelhantes aos demultiplexadores, cabendo a ressalva de que uma diferença significativa entre os dois é que decodificadores não possuem o seletor entre ativo e inativo, já os demultiplexadores podem estar no estado ativo ou inativo. Um diagrama de um codificador pode ser visto na figura 5 (fonte: [steve 2021]).

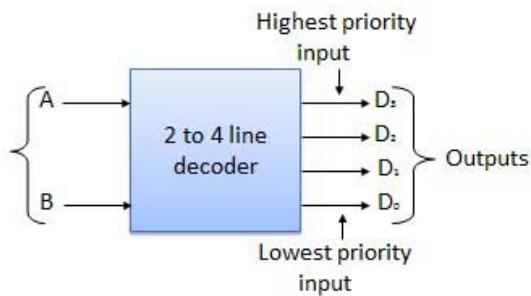


Figura 5. Diagrama de um decodificador

Agora, após uma breve explicação das propriedades de multiplexadores, demultiplexadores e decodificadores, podemos considerar o uso deles para a implementação de circuitos lógicos mais complexos, e bem importantes no dia a dia. Um exemplo de circuito lógico muito importante, e de presença praticamente ubíqua nos circuitos computacionais atuais são os circuitos somadores, responsáveis por operações simples mas básicas e necessárias a toda computação de números. Tais circuitos são responsáveis por somar dois bits **A** e **B**, (possivelmente considerando um carry-in “**Cin**”), e resultando na soma **S** e, possivelmente, no carry-out “**Cout**”. Esse circuito simples é a base para muitos cálculos algébricos realizados em praticamente todo ciclo de clock de um computador moderno. Outra aplicação importante que multiplexadores, demultiplexadores e decodificadores permitem é a implementação de funções genéricas complexas de forma relativamente simples. Essas duas aplicações serão abordadas de forma mais técnica e profunda nos tópicos seguintes.

1.1. Objetivos

Os textos subsequentes deste presente relatório tem por finalidade a elaboração de circuitos lógicos para implementação de somadores e funções lógicas arbitrárias valendo-se de circuitos combinacionais como multiplexadores e decodificadores. A montagem desses circuitos se dará tanto no nível de implementação manual dos circuitos e portas lógicas, bem como o uso da linguagem *SystemVerilog*.

1.2. Materiais

Em função da natureza do ensino a distância, os presentes experimentos não foram realizados usando-se materiais e equipamentos físicos, mas sim emulados por meio do Quartus-II.

A seguir estão enumerados os materiais utilizados:

- Software Quartus-II versão 13.0 SP1
- Multiplexadores

- Decodificadores
- Portas Lógicas **NOT, Buffer**

2. Procedimentos

Passaremos a apresentar os experimentos requeridos.

2.1. Elaboração de Somador Completo desenhandando o circuito

Para a elaboração do somador requerido no enunciado, precisamos inicialmente criar o circuito de um multiplexador. Como requisitado, implementaremos um multiplexador 8x1 usando *SystemVerilog*. O código necessário para criar tal multiplexador é representado em 2.1.

```

1 module mux8 (
2     input [7:0] dado,
3     input [2:0] escolha,
4     output saida
5 );
6
7     assign saida = dado[escolha];
8 endmodule

```

Agora que dispomos do multiplexador necessário para a construção de um somador, precisamos analisar a tabela verdade da função desejada para construirmos um circuito adequado.

A tabela verdade desejada do somador está representada em 2:

Tabela 1. Tabela Verdade para o Somador

Entradas			Saídas	
A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

E, de posse da tabela verdade final, podemos agora criar dois circuitos, um para a função que originará o output de **Cout** e outra função para o output de **S**.

Para o output de **Cout**, basta implementarmos a seguinte função:

$$f(A, B, Cin) = \overline{A} \cdot B \cdot Cin + A \cdot \overline{B} \cdot Cin + A \cdot B \cdot \overline{Cin} + A \cdot B \cdot Cin \quad (1)$$

E, para o output de **S**, implementaremos a função:

$$g(A, B, Cin) = \overline{A} \cdot \overline{B} \cdot Cin + \overline{A} \cdot B \cdot \overline{Cin} + A \cdot \overline{B} \cdot \overline{Cin} + A \cdot B \cdot Cin \quad (2)$$

Seria possível tentarmos minimizar as funções acima, mas, visto que utilizaremos um multiplexador para a construção do circuito, tal otimização torna-se desnecessária.

Agora que dispomos de todas as informações necessárias para a construção do circuito lógico, resta-nos apenas montar o circuito com as portas lógicas, fios e multiplexadores necessários.

A implementação do circuito requisitado no enunciado pode ser visualizada na figura 6.

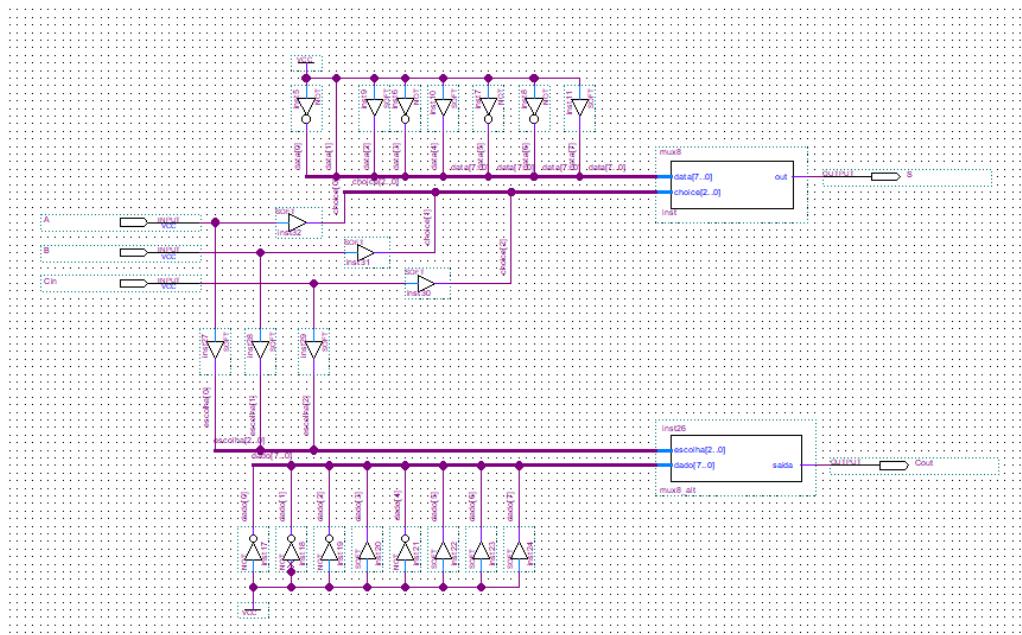


Figura 6. Circuito final do somador montado a partir de mapeamento de portas lógicas

Por fim, com o circuito implementado e devidamente demonstrado, passaremos a apresentar o comportamento efetivo do circuito para todos os possíveis valores de input de **A**, **B** e **Cin**. O diagrama da simulação funcional do circuito acima pode ser visto na figura 7, e o diagrama da simulação de tempo do circuito, na figura 8.

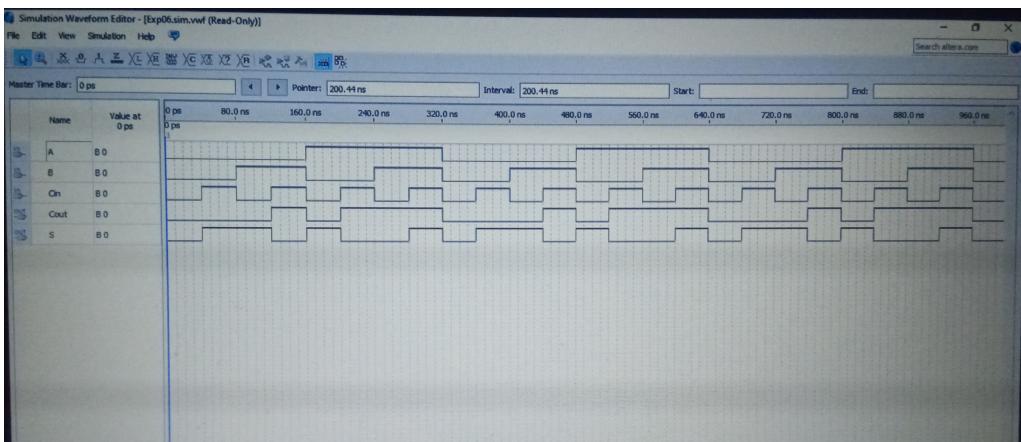


Figura 7. Diagrama Funcional do Somador

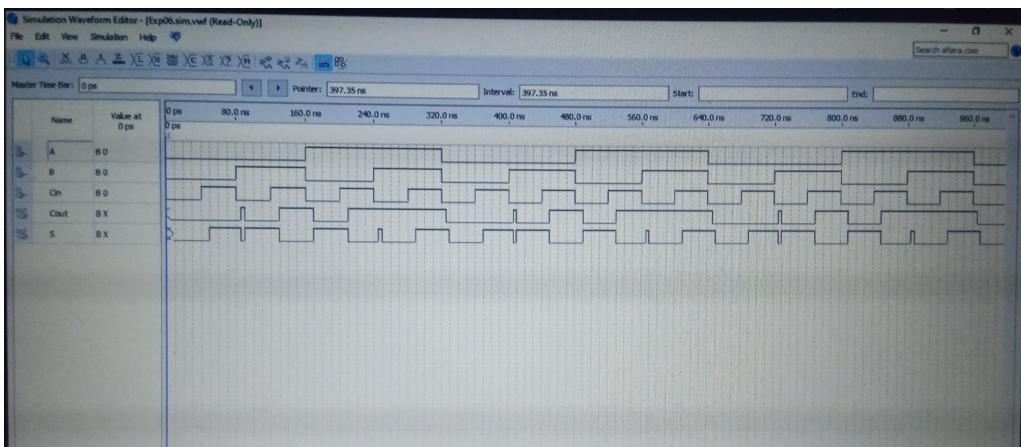


Figura 8. Diagrama de Tempo do Somador

2.2. Elaboração de Somador Completo apenas com Verilog

Para este tópico, que devemos elaborar um somador usando apenas a descrição em verilog, vemos quão mais fácil é o desenvolvimento de circuitos lógicos mais complexos valendo-se de uma linguagem de descrição de hardware adequada.

Para conseguirmos elaborar o somador, basta escrevermos a descrição de um módulo com o seguinte código:

```

1   module full_adder(
2     input A,
3     input B,
4     input Cin,
5     output Cout,
6     output S
7   );
8
9   assign {Cout,S} = A + B + Cin;
10

```

Com esse código, temos a certeza que a implementação final será a de um somador totalmente funcional, pois isso é garantido pela linguagem de descrição de hardware *SystemVerilog*.

Para avaliarmos os diagramas funcional e de tempo do circuito descrito acima, podemos analizar respectivamente as figuras 9 e 10.

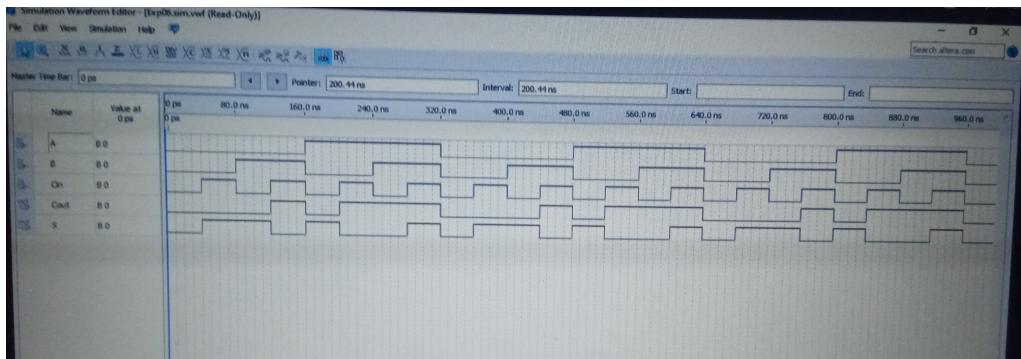


Figura 9. Diagrama Funcional do Somador implementado com *SystemVerilog*

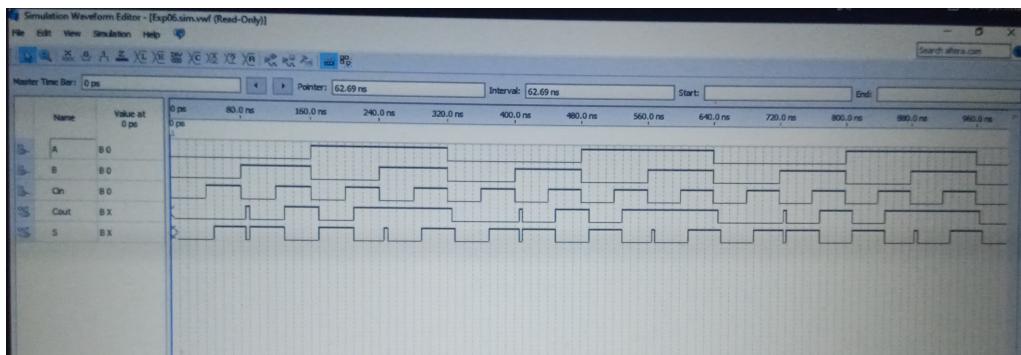


Figura 10. Diagrama de Tempo do Somador implementado com *SystemVerilog*

2.3. Obtendo as funções booleanas para o decodificador com multiplexador

Para o respectivo experimento utiliza-se uma implementação de decodificador com base no demultiplexador apresentado. Sendo em *SystemVerilog* expresso por:

```

1  module decod16(
2      input [3:0] Escolha,
3      output [15:0] Saída
4  );
5
6      assign Saída[0] = Escolha==4'd0? 1'b1 : 1'b0;
7      assign Saída[1] = Escolha==4'd1? 1'b1 : 1'b0;
8      assign Saída[2] = Escolha==4'd2? 1'b1 : 1'b0;
9      ...
10     assign Saída[14] = Escolha==4'd14? 1'b1 : 1'b0;
11     assign Saída[15] = Escolha==4'd15? 1'b1 : 1'b0;
12 endmodule

```

Temos que a função analisada trata-se de:

$$f(A, B, C, D, E, F, G) = FG + A \cdot B \cdot C \cdot D \cdot \bar{E} \cdot \bar{F} \cdot G + \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} \cdot \bar{E} \cdot \bar{F} \cdot G + \\ A \cdot \bar{B} \cdot \bar{C} \cdot E \cdot F \cdot \bar{G} + \bar{A} \cdot B \cdot C \cdot D \cdot \bar{E} \cdot F \cdot \bar{G} + A \cdot B \cdot C \cdot D \cdot E \cdot \bar{F} \cdot \bar{G} + A \cdot \bar{B} \cdot \bar{C} \cdot D \cdot E \cdot \bar{F} \cdot \bar{G}$$

A princípio define-se **A, B, C** e **D** como sendo os bits que irão compor o decodificador e **E, F, G** para o multiplexador. Também temos que **A** será o bit mais significativo e **G** o menos significativo.

Analizando somente os locais que possuem saída 1, sua tabela verdade será descrita por:

Tabela 2. Tabela Verdade para $f(A, B, C, D, E, F, G)$

Entradas							Saídas
A	B	C	D	E	F	G	S
0	0	0	0	0	0	1	1
0	1	1	1	0	1	0	1
1	0	0	1	1	0	0	1
1	0	1	*	1	1	0	1
1	1	1	1	0	0	1	1
1	1	1	1	1	0	0	1
*	*	*	*	*	*	1	1

Deve-se atentar ao fato de que pelo uso da expressão **FG** temos que para todas as combinações que possuem $F = 1$ e $G = 1$ o resultado será 1, utilizando o processo de soma de produtos. Assim, temos que o circuito será descrito através da ferramenta Quartus II como:

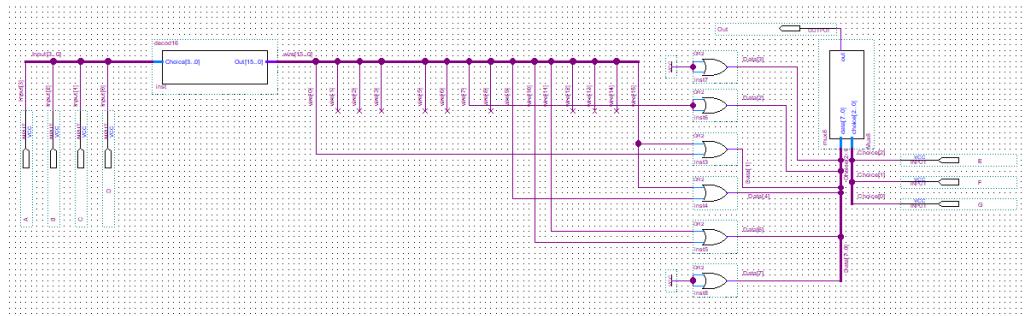


Figura 11. Decodificador e Multiplexador p/ Min Termos

Simulando obtem-se em seu diagrama funcional:

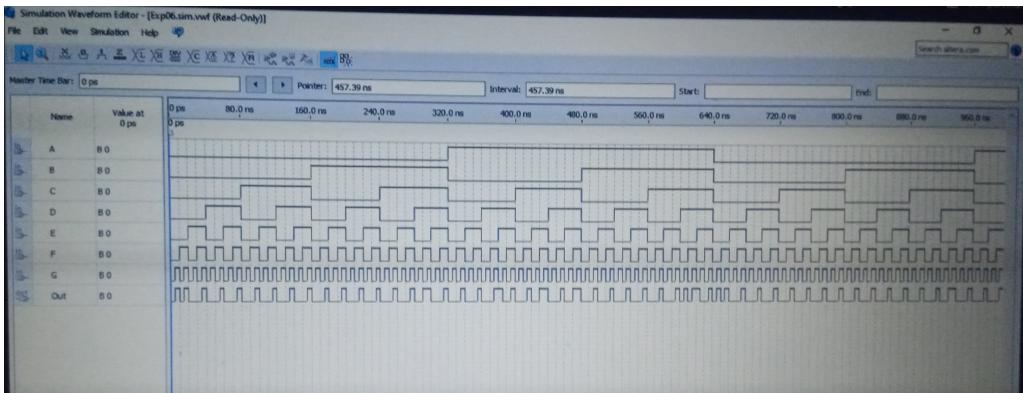


Figura 12. Simulação Funcional 2.3

Observa-se que de fato para $F = 1$ e $G = 1$ temos que saída resulta em 1, como previsto, além dos casos adicionais que compõe a função.

Simulando obtém-se para seu diagrama com análise de tempo:

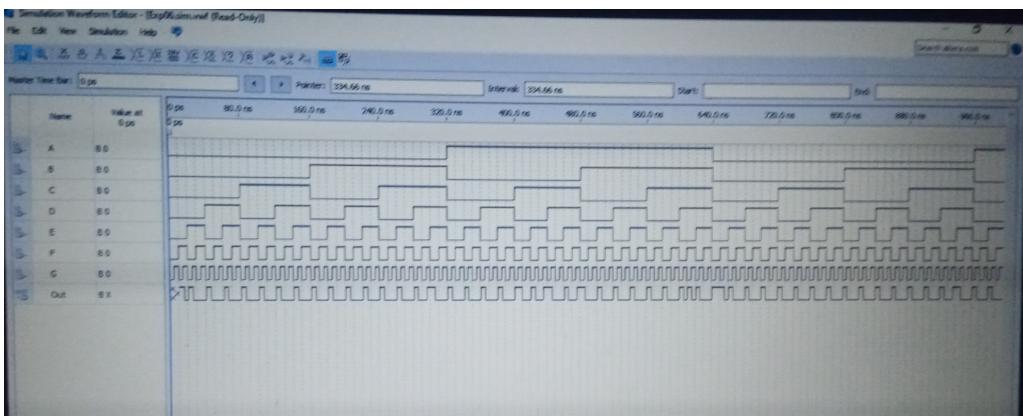


Figura 13. Simulação Temporal 2.3

Observa-se que diferentemente dos experimentos prévios apesar de não haver Hazards de maneira tão evidente tem-se atrasos nos resultados a serem obtidos.

3. Análise dos Resultados

3.1. Análise em 2.1 e 2.2

Dos experimentos 2.1 e 2.2 podemos fazer duas observações interessantes e pertinentes.

A primeira observação é que o circuito criado em 2.1 é praticamente o mesmo que o feito em 2.2, havia vista que os diagramas em 8 e 10 são iguais. E, uma vez que 2.1 e 2.2 são iguais, vemos como uma linguagem de descrição de hardware como o *SystemVerilog* pode ser uma ferramenta útil, pois o tempo de desenvolvimento dos dois circuitos é muito diferente. Enquanto em um o programador precisa se preocupar em ligar corretamente todos os fios e portas lógicas, no outro o programador só precisa especificar o comportamento esperado do sistema, aumentando muito a produtividade no desenvolvimento de novos circuitos.

Outra observação pertinente de se fazer é que em ambos os circuitos há presença de *hazards*, que são ocasionados pelo fato de que os circuitos não serem ideias (eles apresentam atrasos, fade-ins, fade-outs, etc.), e, dessa forma, é importante notar que há momentos que os canais de saída do circuito final (os pinos de **S** e **Cout**) apresentarão, nessa configuração, um resultado errado durante um curto período de tempo.

3.2. Análise em 2.3

Apesar do experimento 2.3 não possuir Hazards evidentes como os outros observados, deve-se atentar ao tempo de propagação. Para visualização do experimento utilizou-se um período de 5ns , para uma análise funcional não temos muitos problemas na adoção deste período, contudo, é errôneo a adoção de um período tão pequeno em projetos práticos, este problema é extremamente evidente na simulação temporal onde temos que a cada novo estado a saída capta o resultado anterior, tal fato é o que justifica uma longa indefinição inicial.

Para este relatório apesar de errônea esta adoção, escolheu-se para a visualização de todos os resultados possíveis e também sermos capazes de analisar o respectivo erro e abordá-lo.

4. Conclusão

O presente relatório foi capaz de além da utilização de multiplexadores, demultiplexadores e decodificadores para a geração de variados tipos de circuitos, como também pudemos ser capazes de analisar os erros associados a suas composições e a importância de ser definido um tempo de mudança ideal para circuitos, caso contrário poderá ser grande o erro obtido nos resultados.

Importante mencionar que circuitos como “*half-adder*” e “*full-adder*” são de extrema importância para a computação, por serem estes elementos que compõe componentes eletrônicos como processadores, pois, com eles somos capazes de contar tempo passado, ciclos realizados, deslocamento entre instruções, assim como também operações de soma no geral, porém, não limita-se a somente as possibilidades mencionadas.

Referências

- [Electrically4u 2021] Electrically4u (2021). Demultiplexers. <https://www.electrically4u.com/what-is-demultiplexer/>.
- [Koike and Mandelli 2021] Koike, C. and Mandelli, M. G. (2021). Codificadores e decodificadores. <https://aprender3.unb.br/mod/url/view.php?id=386056>.
- [Lamar 2021a] Lamar, M. V. (2021a). Introdução a sistemas computacionais - módulo 3 - slides.
- [Lamar 2021b] Lamar, M. V. (2021b). Laboratório de circuitos lógicos.
- [Mandelli 2021a] Mandelli, M. G. (2021a). Codificadores. <https://aprender3.unb.br/mod/resource/view.php?id=386054>.
- [Mandelli 2021b] Mandelli, M. G. (2021b). Multiplexadores. <https://aprender3.unb.br/mod/url/view.php?id=386051>.

[steve 2021] steve, D. (2021). Decoders. <https://wiringschemas.blogspot.com/2019/09/decoder-logic-diagram-and-truth-table.html>.

[Wikipedia 2021] Wikipedia (2021). Multiplexers. <https://en.wikipedia.org/wiki/Multiplexer>.

Auto-Avaliação

Respostas:

Questão	Resposta
1	V
2	V
3	F
4	V
5	F
6	V
7	V
8	F
9	V
10	F
11	V