

Experimento 4

Circuitos Combinacionais: Comparador de Palavras

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

¹Dep. Ciéncia 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 off the elaboration of comparative circuits using concepts already presented in previous reports, like Karnaugh maps, boolean functions e their simplifications, as well as analyze the achieved results and delays due to mechanical factors inherent to the digital micro-architecture.*

Resumo. *O presente relatório tem como objetivo a elaboração de circuitos comparadores utilizando conceitos já apresentados como mapa de Karnaugh e funções booleanas e suas simplificações, assim como a análise de seus resultados e atrasos obtidos por fatores mecânicos existentes na microarquitetura digital.*

1. Introdução

Portas lógicas possuem a característica de poderem se combinar de diferentes maneiras para a formulação de novos circuitos, a depender da maneira a qual se combinam temos um circuito combinacional, que não retém memória de seus estados anteriores. Contudo, ao ocorrer a associação destas portas lógicas, não será obtido um sistema inafetado por erros, pois é necessária a consideração de fatores mecânicos e como eles prejudicam a execução de um dado circuito, seja ele combinacional ou não.

1.1. Objetivos

O atual experimento busca não somente criar e demonstrar novos tipos de combinações para a obtenção de comportamentos específicos e desejados, como também fatores a serem considerados referentes ao tempo de resposta de um dado sistema e porque ocorrem atrasos ao associar inumeras portas lógicas.

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 Deeds.

A seguir estão enumerados os materiais utilizados:

- Componentes de Entrada(Gerador de clock e chave de estado)
- Componentes de Saída
- Portas Lógicas **NAND, OR, AND, NOT**

2. Procedimentos

Passaremos a apresentar os experimentos requeridos.

2.0. Atraso de Propagação em portas lógicas

A) Equações Lógicas de L0 e L1

Começaremos com a equação lógica **L1**, pois **L0** depende dessa. Temos que, como o número de portas **NOT** que ligam o sinal de clock **A** à **L1** é *ímpar*, a saída terá o sinal de **A** negado. Sendo assim, concluímos que a equação para **L1** será:

$$L1 = \overline{A} \quad (1)$$

Agora, considerando a saída **L0**, podemos verificar que existe uma porta **AND** tomando o resultado de **A** e **L1**. Portanto, a equação de **L0** será:

$$L0 = A \cdot L1 \quad (2)$$

Dessa forma, podemos expressar ambos **L0** e **L1** por meio de uma tabela verdade, em função de **A**.

Tabela 1. L0 e L1 em função de A

Entrada	Saídas	
A	L0	L1
0	0	1
1	0	0

B) Comportamento de L0 e L1

Utilizando a ferramenta Deeds para a simulação do circuito lógico, podemos observar que as saídas de **L0** e **L1** são exatamente as esperadas quando leva-se em conta a tabela verdade 1 apresentada acima.

Como representado nas figuras a seguir, o output de **L0** é *sempre* igual a 0, e o output de **L1** é *sempre* o oposto de **A**, assim como esperado.

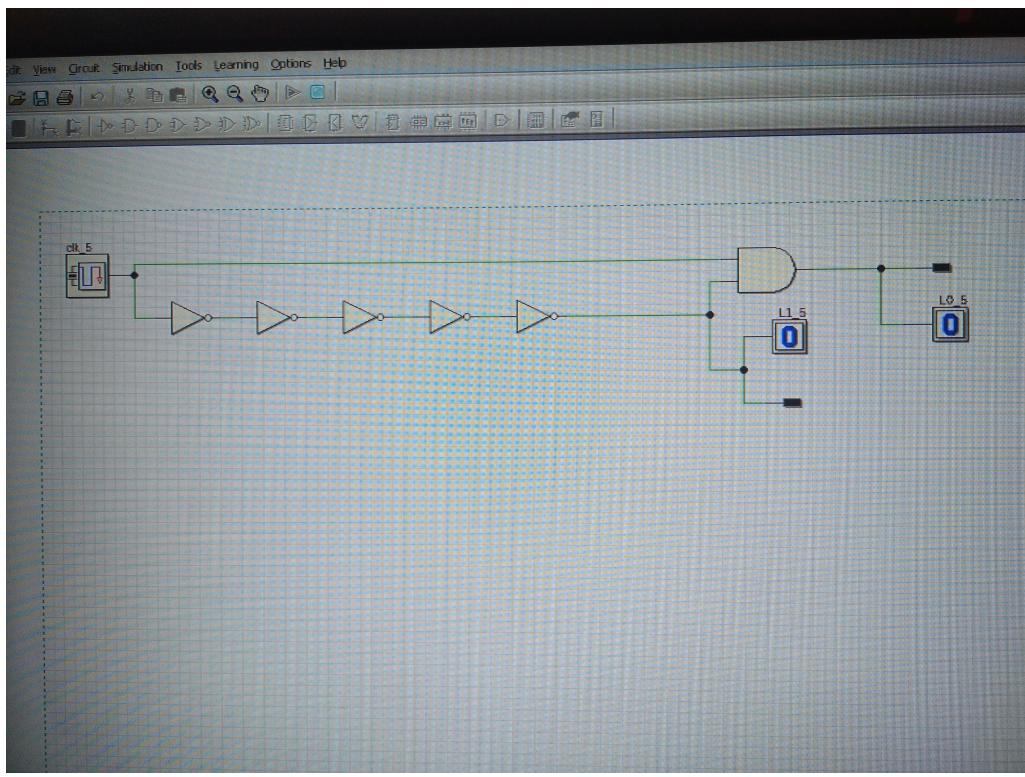


Figura 1. Clock Up

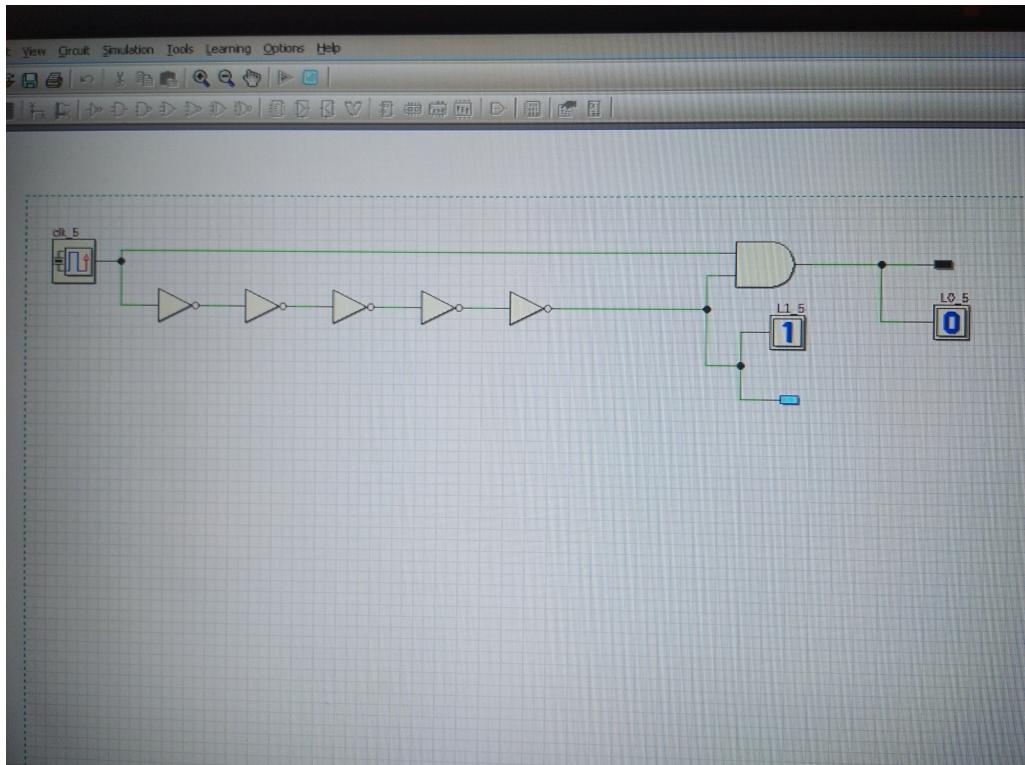


Figura 2. Clock Down

Para conferir o vídeo deste experimento, acesse o seguinte link:
<https://youtu.be/xzFPo8TH368>.

C) Formas de onda para L0 e L1

Utilizando a ferramenta *Timing Diagram Simulation* podemos gerar as formas de onda para **L0** e **L1**. A seguir reproduzimos as mesmas.

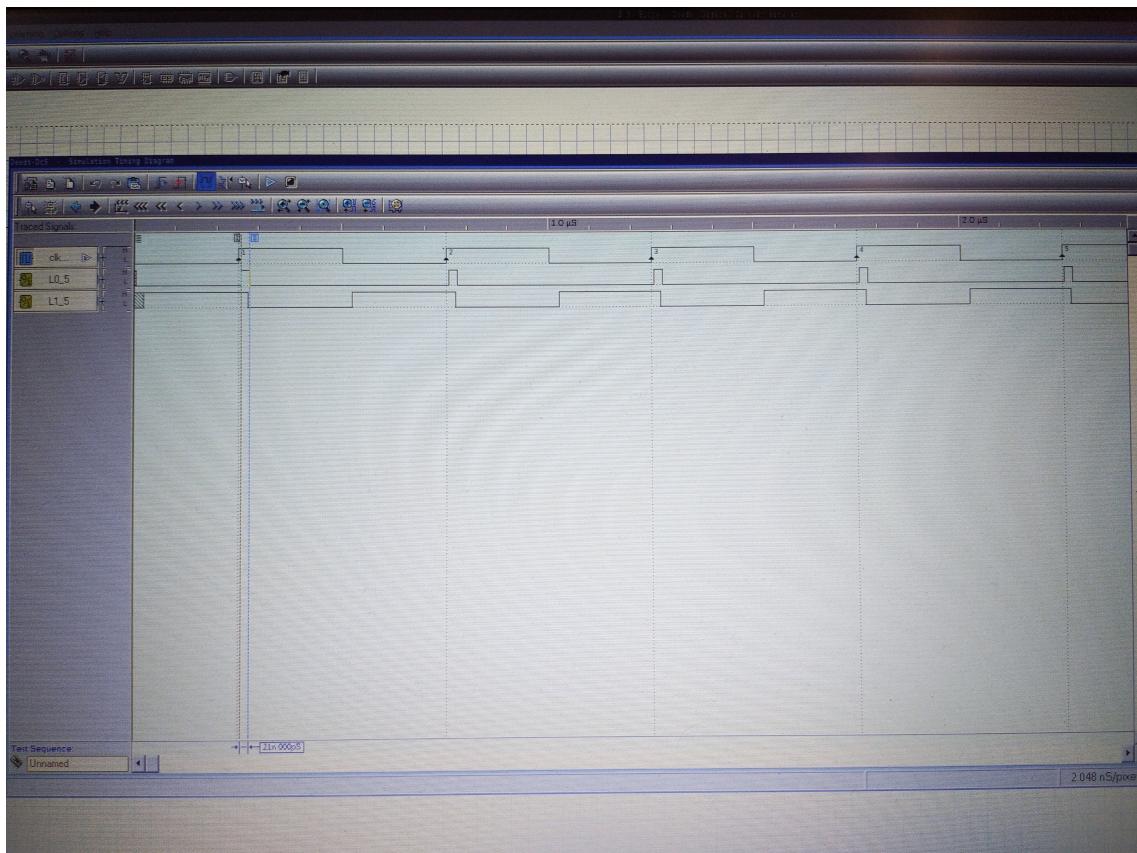


Figura 3. Onda e comprimento de pulso em L0

Como pode-se observar, a saída **L0** possui um pequeno pulso quando o clock varia. Podemos explicar esse comportamento levando em consideração o delay que as portas lógicas possuem. Como a porta lógica **L0** possui como um de seus inputs a porta **L1**, que por sua vez tem um delay de 5 portas **NOT**, a porta **L0** soma um delay de 5 portas **NOT** e 1 porta **AND**. Dessa forma, a soma total resulta em um delay de 21ns, como pode-se ver na imagem a seguir.

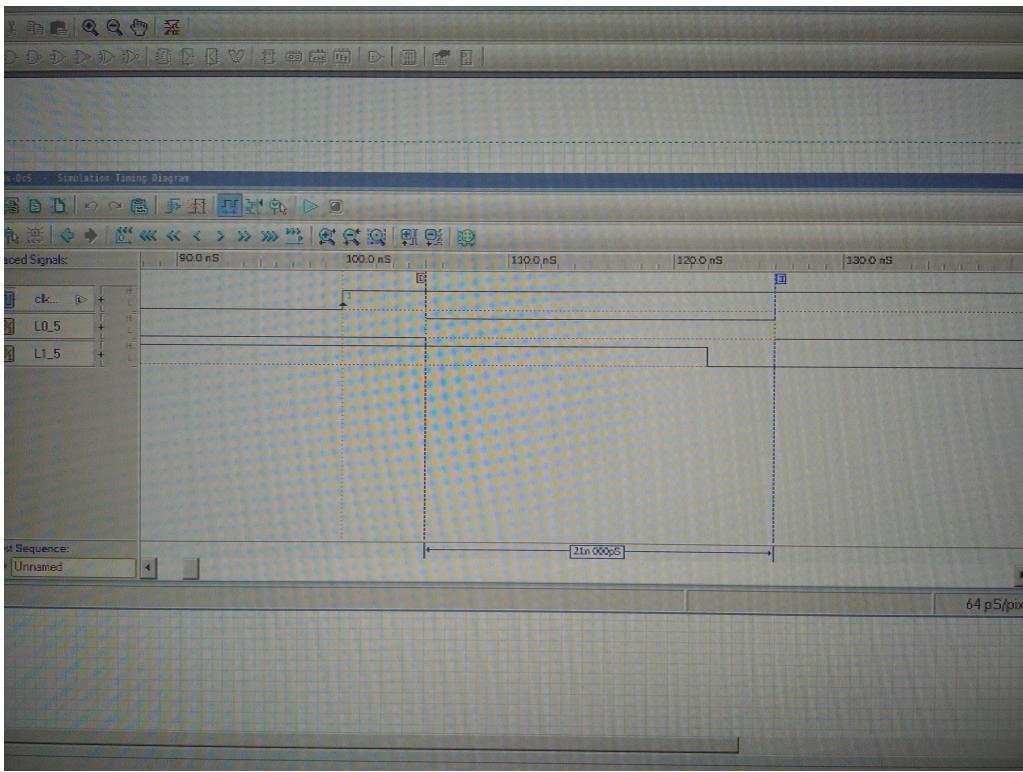


Figura 4. Comprimento do pulso em L0

D) Pulsos em L0 quando A retorna ao nível 0

De forma sucinta, podemos afirmar que, para o circuito específico deste exercício, contendo 5 portas **NOT** e 1 porta **AND**, não existe pulso em **L0** quando **A** retorna ao nível 0. O motivo para isso é simples: quando **A** está retornando de 1 para 0 o valor lógico de **L1** é 0, e assim a porta lógica **AND** que produz o output de **L0** sempre retornará o valor 0, justificando a ausência de pulso nesse momento do circuito.

Para justificar tal afirmação precisamos considerar qual a função lógica responsável por produzir o output em **L0**. Lembrando da equação 2, vemos que a única forma de termos um pulso em **L0** seria se **L1** fosse 1 no momento que **A** retorna ao nível 0. Podemos afirmar isso pois consideramos o delay da porta **AND** responsável pelo output de **L0**. Logo, o pulso seria consideravelmente menor que o existente na passagem de nível lógico **A** = 0 para **A** = 1, mas seria possível existir. Cabe ressaltar, entretanto, que o pulso em **L0** *não* aconteceria na montagem de apenas 5 portas **NOT**, pois o delay não seria suficientemente grande. Na imagem a seguir apresentamos uma possível configuração de circuito lógico, contendo um número ímpar de portas **NOT**, que tem de fato um pulso em **L0** quando **A** está retornando ao nível 0.

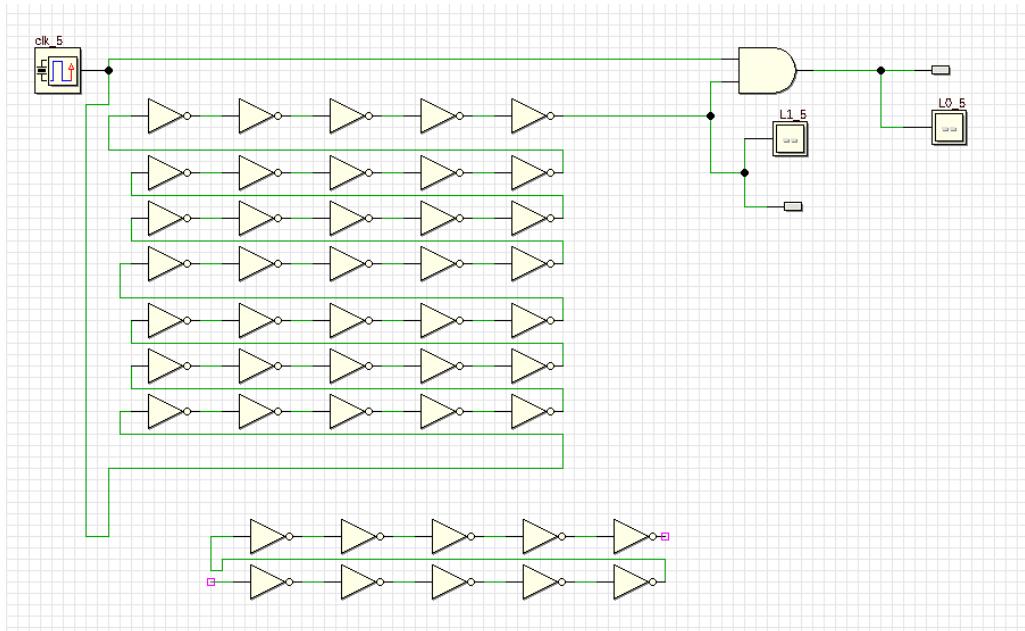


Figura 5. Circuito que apresenta pulso quando A retorna ao nível 0

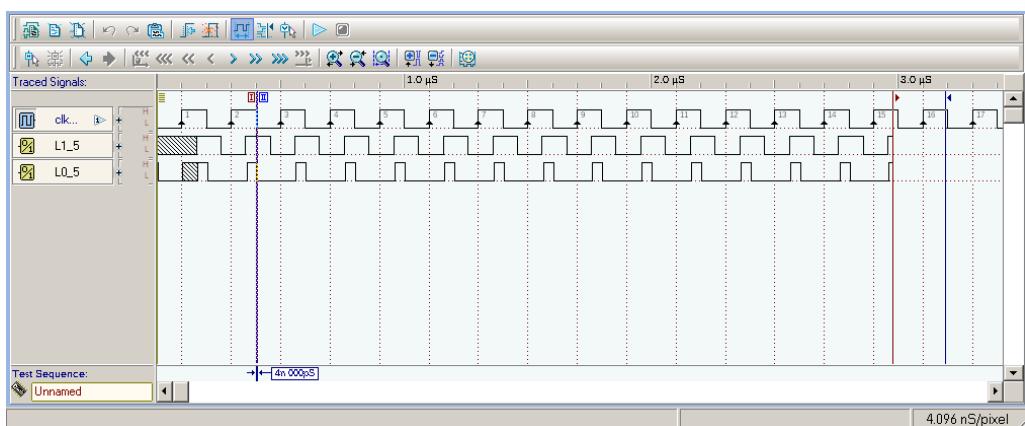


Figura 6. Formato de onda desse circuito

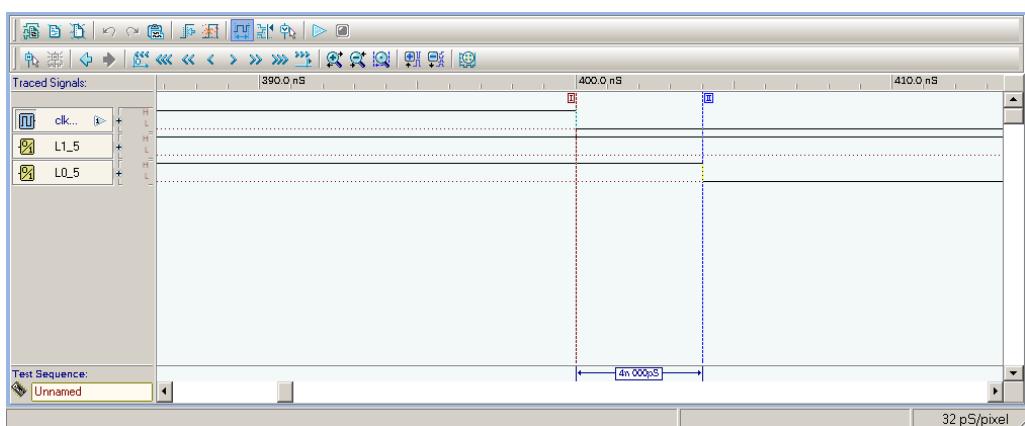


Figura 7. Tamanho do pulso existente quando A retorna ao nível 0

E) Haveria pulso em L0 caso houvesse um número par de portas NOT?

Sim, poderia existir um pulso em **L0**. O único limitante para a observação de pulsos é a quantidade de portas **NOT** existentes no circuito. O motivo é o mesmo do item anterior: quanto maior o número de portas, maior será o delay no circuito lógico; Caso o delay acarrete em uma superposição entre o nível lógico 1 de **A** e **L1**, então haverá um pulso em **L0**. Ou seja, o pulso é apenas limitado à introdução de um delay suficientemente grande no circuito.

Como prova dessa afirmação, é apresentado um circuito com número *par* de portas **NOT**, e que ainda assim possui pulsos em **L0**.

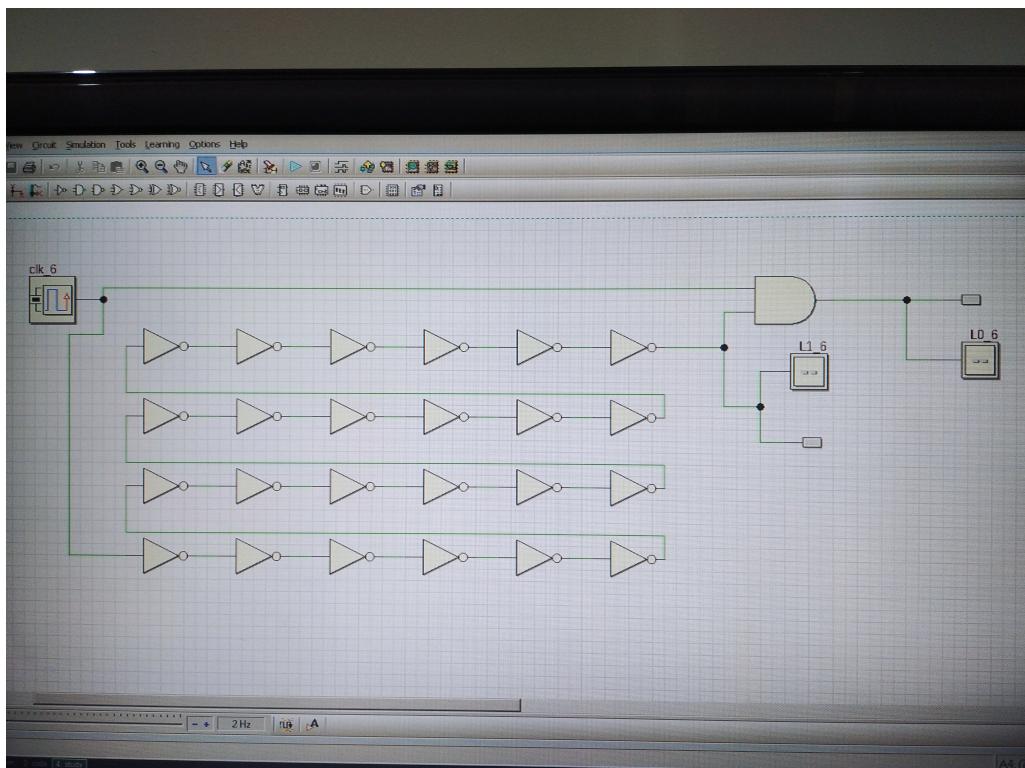


Figura 8. Circuito com pulso em L0 usando número *par* de portas NOT

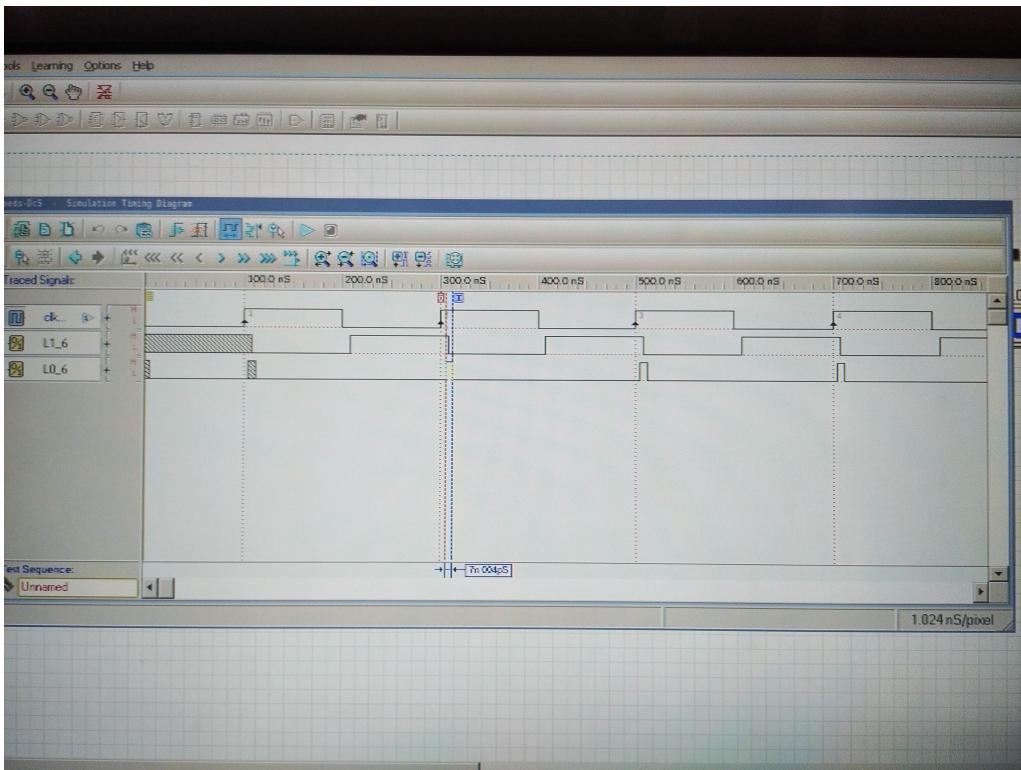


Figura 9. Formato de onda do circuito com pulso

2.1. Comparador de palavras de 3 bits

A) Completando a tabela verdade para um circuito XNOR

Para fazermos a tabela verdade do circuito proposto no enunciado, basta notarmos a própria definição de uma porta **XNOR**; Sendo assim, teremos:

Tabela 2. Tabela Verdade para o circuito XNOR

Entradas		Saída
A_i	B_i	Z_i
0	0	1
0	1	0
1	0	0
1	1	1

B) Implementação da função lógica Z_i apenas com portas NAND

Para implementarmos um circuito lógico **XNOR** usando apenas portas **NAND**, podemos usar o circuito a seguir:

$$\overline{\left(\overline{(A_i \cdot A_i)} \cdot \overline{(B_i \cdot B_i)} \right)} \cdot (A_i \cdot B_i) \quad (3)$$

C) Subcircuito de comparação de duas palavras de 1 bit

Como já dispomos de uma fórmula booleana para a implementação de um circuito que compare duas palavras de 1 bit, podemos agora implementar um circuito que de fato possua essa funcionalidade. Mostramos a seguir tal subcircuito:

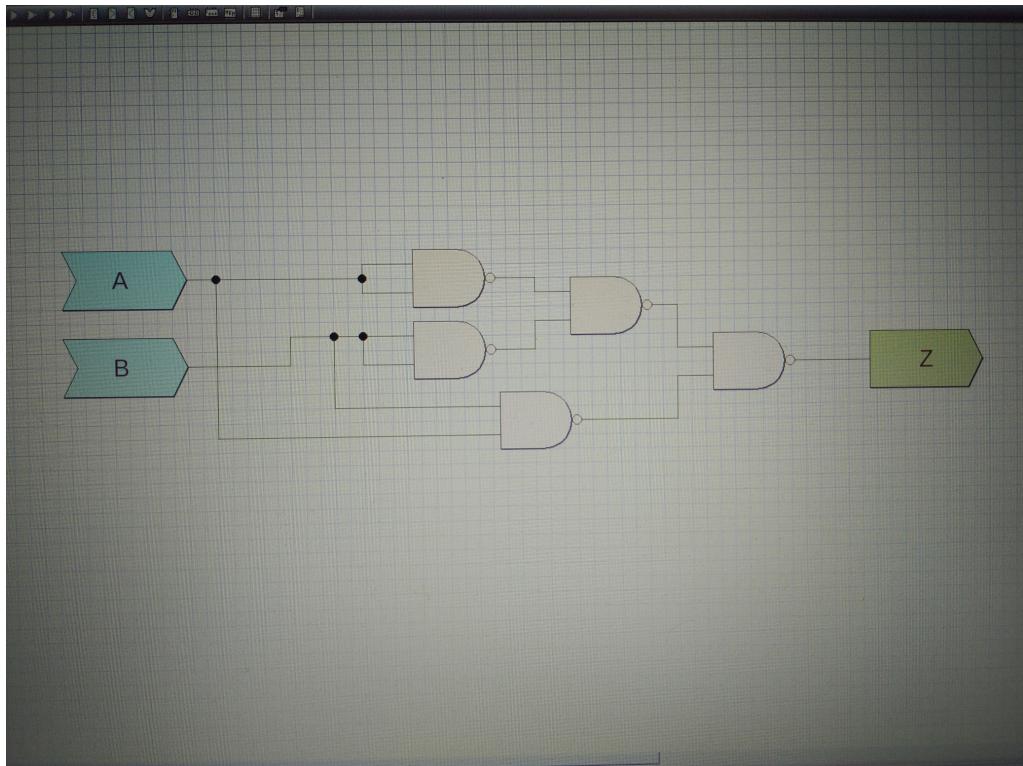


Figura 10. Subcircuito comparador de duas palavras de 1 bit

Para conferir o vídeo deste experimento, acesse o seguinte link:
<https://youtu.be/fElngmSb5IQ>.

D) Circuito comparador de palavras de 3 bits

Para montarmos o circuito total pedido no enunciado, basta usarmos 3 dos blocos implementados no item anterior, fazendo um **AND** entre cada uma das 3 saídas. Caso cada um dos blocos retorne o nível lógico 1, então teremos que o output final também será com nível lógico 1, e, caso contrário, o nível 0. Existe, entretanto, um requisito: devemos montar esse circuito apenas com portas **NAND** de 2 entradas; Sendo assim, apresentamos um circuito que satisfaz os requisitos necessários para o objetivo desejado:

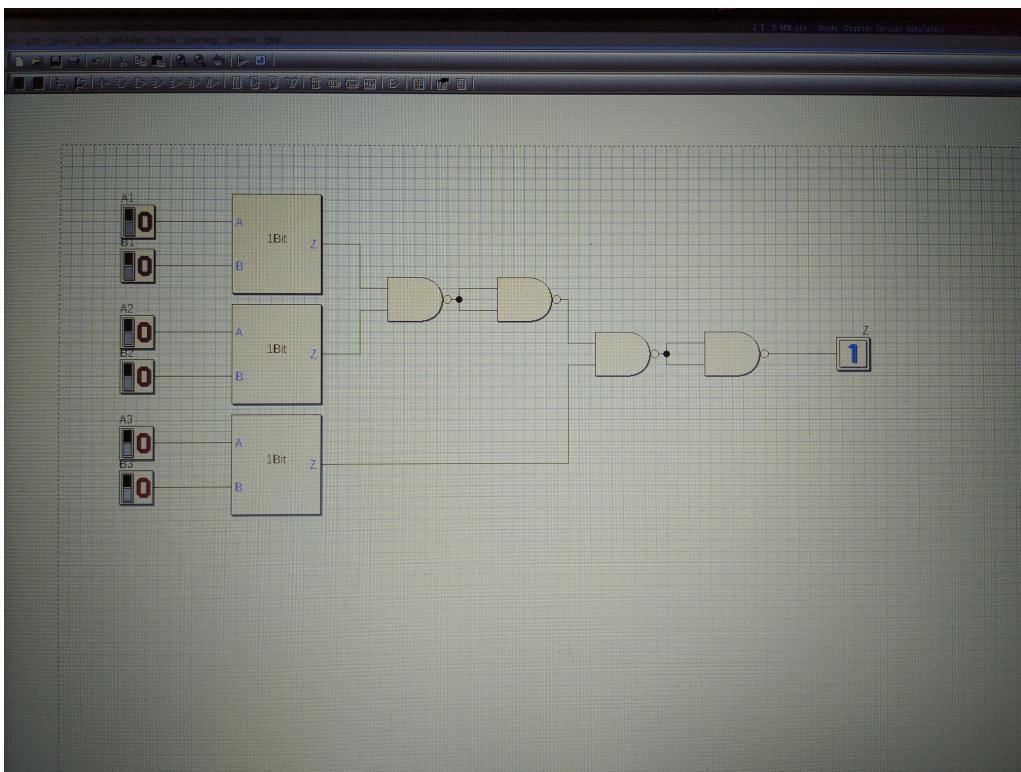


Figura 11. Circuito Comparador de palavras de 3 bits

Reproduzimos a seguir a tabela verdade do circuito final. Contudo, Faz-se necessário uma ressalva: como o circuito final possui 6 valores de input, isso acarreta em uma tabela com $2^6 = 64$ linhas. Uma tabela assim seria muito grande para reproduzirmos aqui. Dessa forma, optamos por somente mostrar os inputs que produzem um output com valor igual a 1. Todas as demais combinações são assumidas como produzindo o output de nível lógico 0.

Tabela 3. Tabela Verdade para o circuito geral

Entradas						Saída
A_1	A_2	A_3	B_1	B_2	B_3	Z_i
0	0	0	0	0	0	1
0	0	1	0	0	1	1
0	1	0	0	1	0	1
0	1	1	0	1	1	1
1	0	0	1	0	0	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	1

E) Circuito comparador de palavras de 3 bits

Com o curcuito completo em mãos, podemos analisar as formas de onda geradas por ele. A seguir é reproduzido tal gráfico.

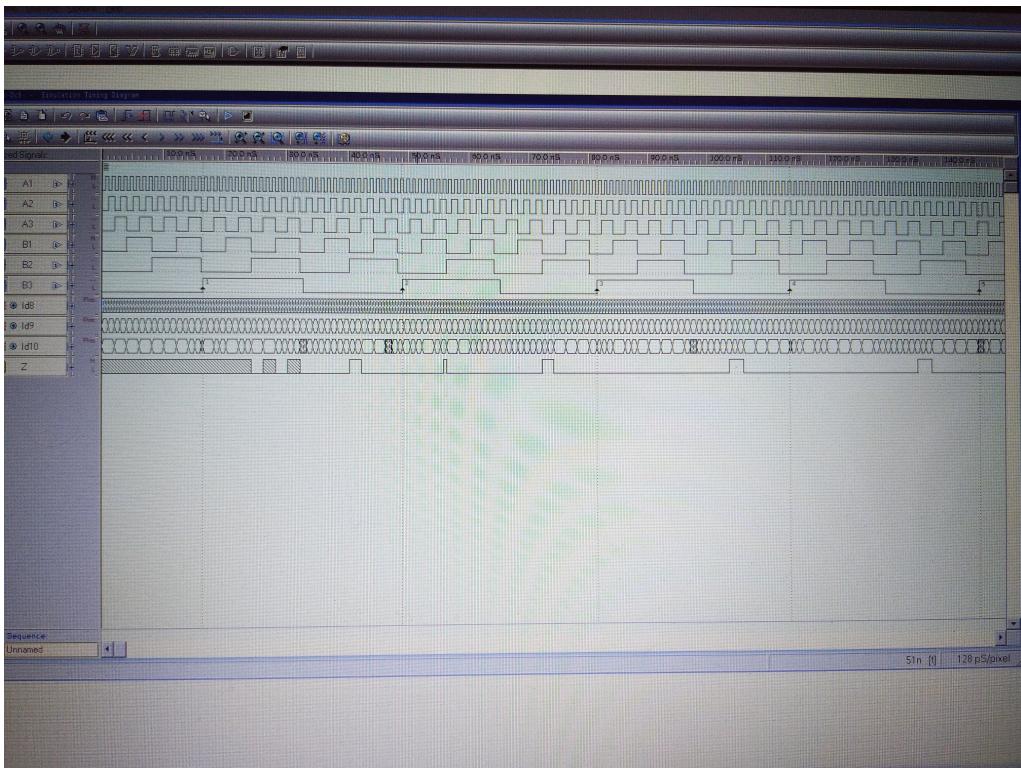


Figura 12. Comparador de palavras de 3 bits

Como podemos observar pela imagem, os pulsos obtidos em **Z** nem sempre correspondem a posição esperada deles. Por exemplo, o terceiro pulso existente, que ocorre por volta dos 70ns , não deveria existir. Podemos justificar essa afirmação com duas observações: **1)** O referido pulso é bem maior do que teoricamente deveria ser (havia vista que o pulso em **A1** é muito menor que ele) e **2)** o pulso em **B2** possui valor 1 nesse momento, e o esperado seria que tivesse o valor 0.

Considerando essas observações e o valor teoricamente esperado, podemos justificar o valor experimental observado levando em consideração o atraso que cada porta lógica insere no circuito. Dessa forma, algumas configurações que anteriormente não seriam esperadas de produzirem o valor lógico 1 acabam existindo.

2.2. Comparador de palavras de 2 bits

A) Tabela verdade de um comparador de 1 bit

Como trata-se de um comparador de 1 bit, para a tabela verdade teremos:

Tabela 4. Tabela Verdade para o circuito comparador de 1 bit

Entradas		Saída		
<i>A</i>	<i>B</i>	Y_1	Y_2	Y_3
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Assim, teríamos que para cada uma das saídas desejadas, suas equações lógicas seriam:

$$Y_1 = A \cdot \overline{B} \quad (4)$$

$$Y_2 = A \cdot B + \overline{A} \cdot \overline{B} \quad (5)$$

$$Y_3 = \overline{A} \cdot B \quad (6)$$

Observa-se que todas as equações já estão em uma forma minimizada, já que possuem poucos termos, e quando colocadas em mapa de Karnaugh não haverá uma maneira de reduzi-las mais do que já descrito. Contudo, a minimização poderá ocorrer ao implementar o circuito.

B) Implementação de um comparador de 1 bit

A princípio, seguindo as equações de maneira literal teríamos o seguinte esquema:

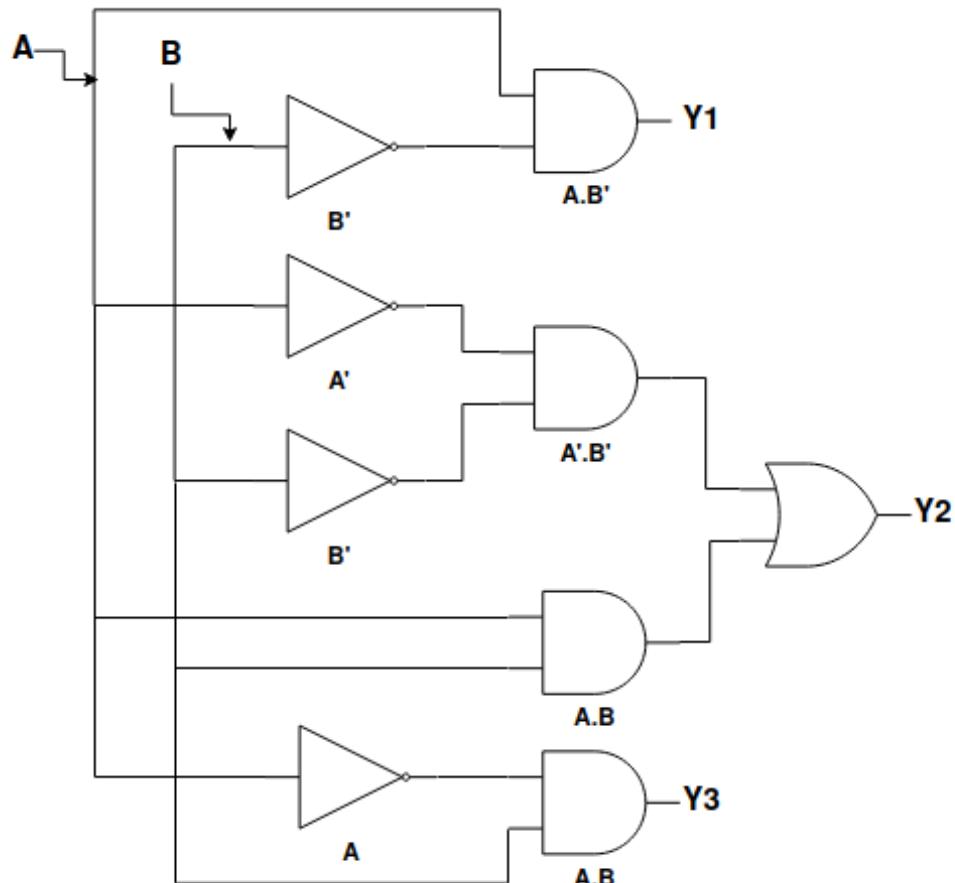


Figura 13. Comparador de 1 bit

Uma maneira de criar redução será:

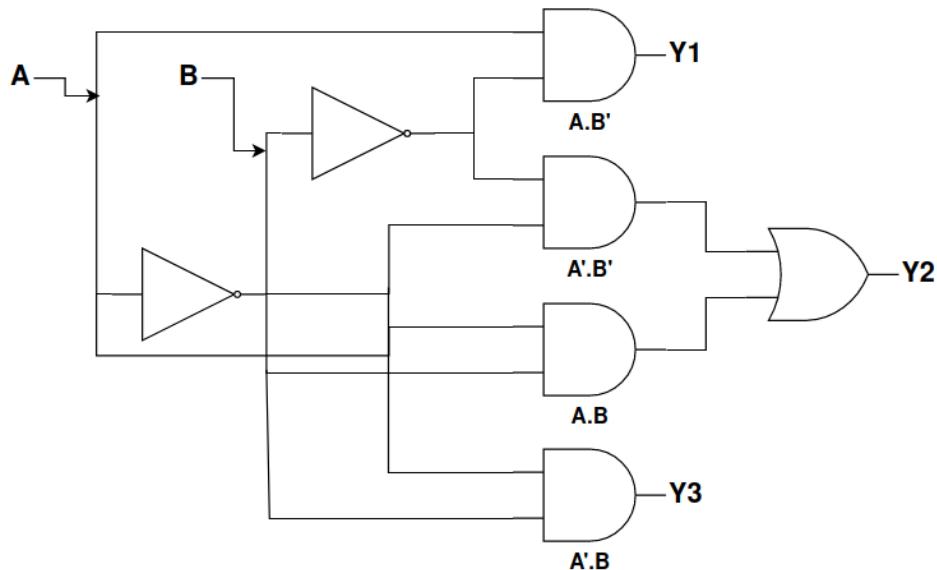


Figura 14. Comparador Simplificado de 1 bit

Ou seja, bastando reduzir as portas NOT implementadas. Assim, temos que o subcircuito implementado através do Deeds será:

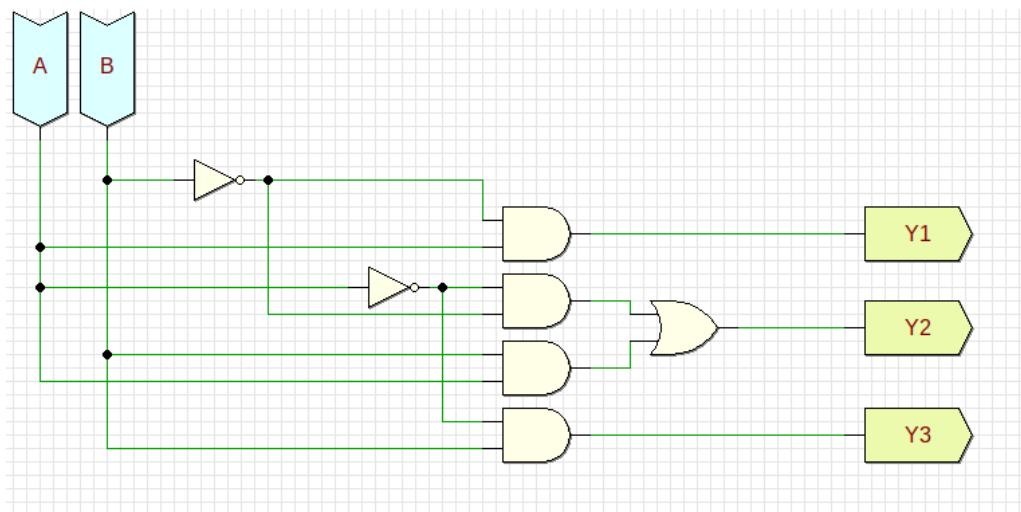


Figura 15. Comparador Simplificado de 1 bit - Deed

Deve-se ater também ao fato de que por a utilização de duas portas NOT fatores como fan-out serão considerados. Abaixo encontra-se o link de simulação:

Para conferir o vídeo deste experimento, acesse o seguinte link:
<https://youtu.be/Ac0u58QqDo>.

C) Implementação de um comparador de 2 bits

Utilizando o comparador de 1 bit construído, temos que a implementação para um comparador de 2 bits será:

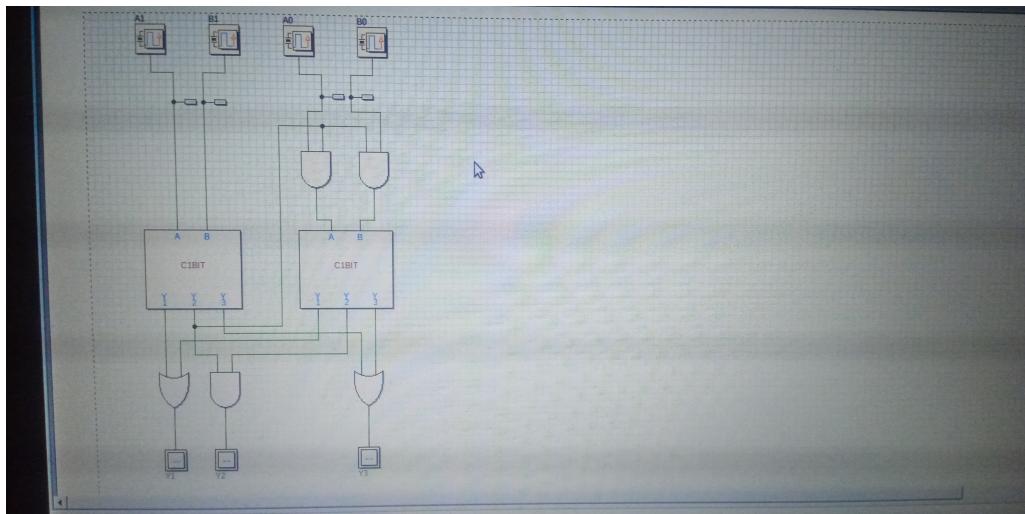


Figura 16. Comparador 3 Bits

Abaixo segue a imagem de sua simulação:

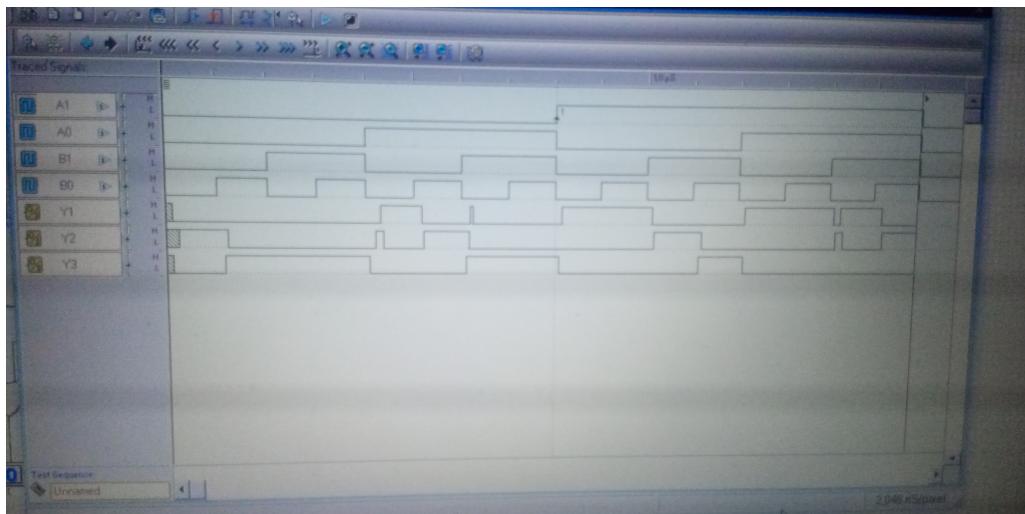


Figura 17. Diagram de Comparador de 2 bits

Portanto, temos para sua tabela verdade o seguinte resultado:

Tabela 5. Tabela Verdade para o circuito comparador de 2 bits

Entradas				Saída		
A_1	A_0	B_1	B_0	Y_1	Y_2	Y_3
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

D) Atrasos Obtidos

Devido a estrutura de circuito feita, ocorrem longos atrasos, e que para o computadores utilizados contidianamente já seria um atraso significativo, percebe-se não somente problemas em seus atrasos como a produção de Hazards. Portanto, seria necessário uma análise mais detalhada que visa a redução destes erros.

3. Análise dos Resultados

Como pudemos notar através de todos os experimentos realizados erros e atrasos ocorrem, provando a falha mecânica e a necessidade de técnicas que reduzam estes mesmos erros. Percebe-se também como da transição de um estado booleano a outro não irá produzir uma saída imediata, por fatores como o da própria borda de subida e descida na mudança de estados. Importante notar que apesar do uso de um simulador para os circuitos, ao utilizarmos geradores de onda no mundo físico, deve-se considerar que não é possível a criação de estados de ondas que ocorrem ao mesmo tempo, ou seja, simultâneos, pois, como já mencionado, sempre há fatores mecânicos a serem considerados.

4. Conclusão

A elaboração de técnicas que lidam com a álgebra booleana possibilitam facilidade ao lidar com problemas do mundo real, de um ponto de vista teórico foi possível a construção de modelos práticos. A princípio poderia ser percebidos como problemas futeis ou sem muitas necessidades, contudo, deve-se notar que todos os computadores necessitam de tais operações, ou seja, igualdade e inequações, assim, o presente relatório apesar de tratar destes problemas de maneira mais simplificada ainda foi capaz de descrever comportamentos tão desejados na implementação de máquinas utilizadas no dia a dia. Apesar de sermos capazes de descrever tais comportamentos, deve-se notar que vêm com um preço, sendo os erros associados a estes mesmos circuitos.

Referências

[Lamar 2021] Lamar, M. V. (2021). Laboratório de circuitos lógicos.

[Mandelli 2021] Mandelli, M. (2021). Circuitos lógicos.

Auto-Avaliação

Respostas:

A	B
1	c
2	a
3	d
4	a
5	a