



Laboratório 3: Projeto e montagem de redes iterativas.

Bernardo Hoffmann da Silva

Marcos Vinicius Pereira Veloso

5: Montagens

De modo a facilitar o entendimento experimental e ratificar os resultados obtidos em laboratório, a equipe armazenou os resultados em *google drive*, conforme segue na seção de Referência [1].

5.1: Montagem de um circuito combinacional com portas NAND2

Inicialmente, montou-se em protoboard, utilizando uma unidade CI 7400, a função combinacional apresentada na Figura 1.

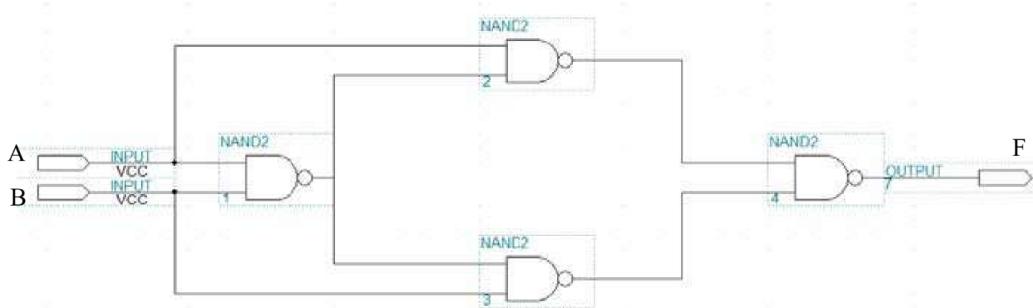


Figura 1: Referência da função combinacional montada em protoboard

A Figura 2 apresenta o esquema elétrico da montagem e a Figura 3 a montagem em si, conforme feita em laboratório.

Com a montagem, pôde-se construir experimentalmente a tabela verdade para a função. A Tabela 1 representa os resultados encontrados.

Observando-se a tabela verdade montada, pode-se escrever diretamente a Relação (1).

$$A\bar{B} + \bar{A}B = F \quad (1)$$

Portanto, percebe-se que a saída F é uma função XOR aplicada às entradas A e B. Note-se, portanto, que é possível implementar essa função utilizando apenas 4 portas NAND.

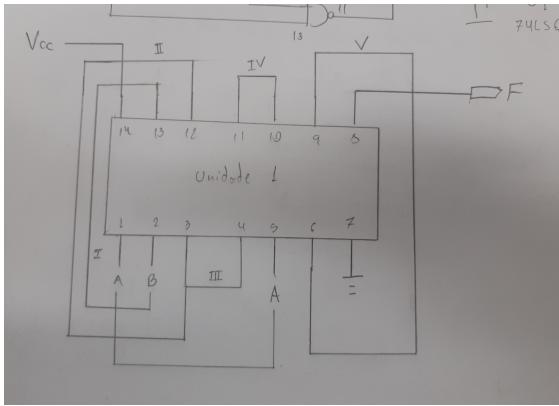


Figura 2: Esquema elétrico da função combinacional montada utilizando a unidade CI 7400

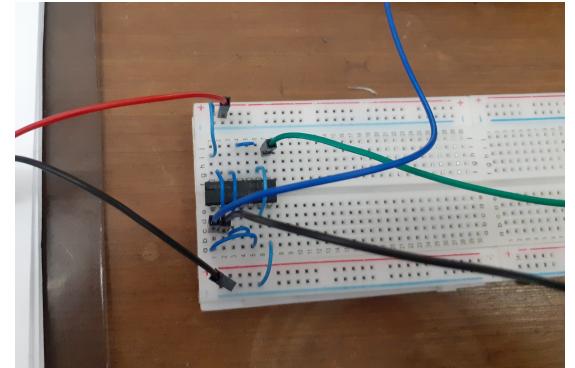


Figura 3: Montagem da função na protoboard

Tabela 1: Tabela verdade construída experimentalmente a partir do resultado de saída do circuito.

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

5.2: Projeto e montagem de um subtrator completo de 1 bit

Para o subtrator completo de 1 bit, tem-se a tabela verdade apresentada na Tabela 2, conforme especificado pelas variáveis de saída.

A partir da Tabela 1, podemos montar as funções booleanas relacionadas às saídas do subtrator, conforme o uso de Mapas de Karnaugh. Deles decorrem que

$$F = A \oplus B \oplus B_{in} \quad (2)$$

$$B_{out} = \overline{A}(B_{in} + B) + B_{in}B \quad (3)$$

Percebe-se que a operação XOR foi implementada com portas NAND na Seção 5.1 deste relatório. Assim, a transcrição da Relação (2) em portas NAND será omitida. Para a Relação (3), pode-se transformá-la para utilizar apenas portas NAND, via Teorema de De Morgan.

Tabela 2: Tabela verdade construída a partir da definição das variáveis de saída de um subtrator.

A	B	Bin	F	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$\begin{aligned}
 B_{out} &= \overline{AB}_{in} + \overline{A}B + BB_{in} \\
 &= \overline{\overline{AB}_{in}\overline{A}B} + B_{in}B \\
 &= \overline{\overline{\overline{AB}_{in}} \cdot \overline{\overline{A}B}} + EB \\
 &= \overline{\overline{\overline{AB}_{in}}} \cdot \overline{\overline{\overline{A}B}} \cdot \overline{B_{in}B}
 \end{aligned} \tag{4}$$

Portanto, para a implementação de B_{out} , são necessárias 7 portas NAND. O esquema elétrico para a função combinatória de Cout está apresentado na Figura 4, a partir do uso das duas unidades implementadas na Seção 5.1.

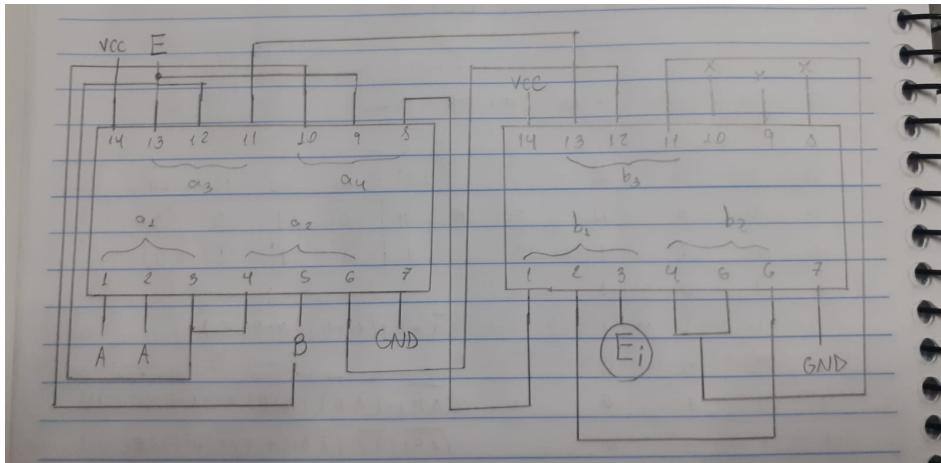


Figura 4: Esquema elétrico para a implementação de Cout utilizando duas unidades CI 7400.

Para a implementação da função F, temos a utilização de duas portas XOR em associação em série. Portanto, com dois esquemas apresentados na Figura 2, podemos construir a função, utilizando como entradas da segunda unidade CD 7400 o B_{in} e o resultado de saída da unidade 1. A montagem do subtrator completo de 1 bit se encontra na figura 5, em que há a implementação da saída *borrow* usando portas NAND conforme consta na Relação (7).

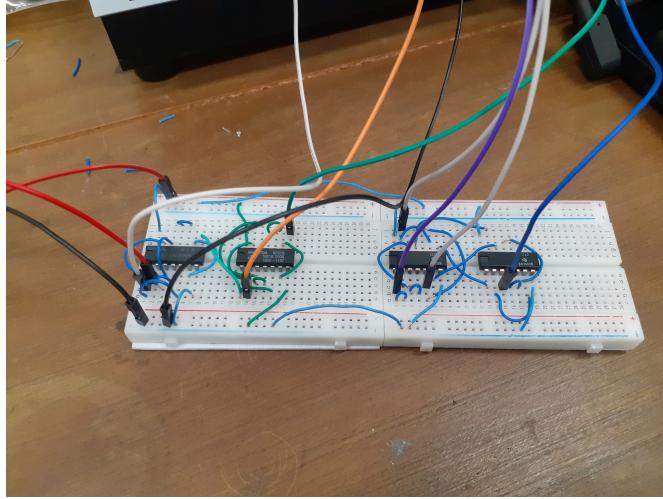


Figura 5: Montagem do subtrator completo de 1 bit utilizando 4 unidades CI 7400

Concluída a montagem, foi possível testar as saídas variando-se as entradas A , B e B_{in} , no qual foi verificado que o resultado de soma F e *borrow* B_{out} coincidiu conforme o denotado pela tabela verdade, já construída na Tabela 2. O resultado de saída experimental deste e dos demais experimentos, conforme já mencionado, estão armazenados na nuvem para eventuais consultas conforme segue na Referência [1].

5.3: Projeto e montagem de um somador completo de 1 bit

Tal como visto no projeto para subtrator completo de 1 bit, a função lógica de resultado de saída e de *carry* são dados por:

$$F = A \oplus B \oplus C_{in} \quad (5)$$

$$C_{out} = A(C_{in} + B) + C_{in}B \quad (6)$$

Note-se que a saída F de ambos, subtrator e somador, possuem a mesma função booleana. Sendo assim, pode-se aproveitar o termo construído na protoboard para F implementada no subtrator. Ademais, note-se que a única diferença entre B_{out} e C_{out} consiste no uso ou não do literal complementar de A . Posto isso, pode-se montar a tabela verdade presente na Tabela 3.

Para a implementação da saída de resultado da soma, basta utilizar a mesma saída de resultado da subtração. No que diz respeito ao *carry*, como a única diferença consiste no uso de A no lugar de \bar{A} , basta que façamos a entrada ser A na implementação do *borrow* visto na seção do subtrator. Com isso em mente, visto que o circuito é o mesmo do subtrator a exceção da entrada A no lugar de \bar{A} , o circuito é omitido aqui. Com base em (4), o *carry* para o somador tem como função em portas NAND o resultado

$$C_{out} = \overline{\overline{(AC_{in})} \cdot \overline{(AB)} \cdot \overline{C_{in}B}} \quad (7)$$

Mais uma vez, o resultado do circuito denota o mesmo resultado presente na Tabela 3.

Tabela 3: Tabela verdade construída a partir da definição das variáveis de saída de um somador.

A	B	Cin	F	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

6: Simulações

6.1: Projeto e simulação da célula básica de um comparador

Para o comparador iterativo de n bits, são necessárias quatro entradas para a célula básica, isto é, duas variáveis secundárias e duas primárias, e duas saídas secundárias, sendo as entradas $Y_{1,i}, Y_{3,i}$, que são as saídas da célula anterior, e A_i, B_i , que são os bits atuais da entrada. As saídas são $Y_{1,i-1}, Y_{3,i-1}$. A tabela verdade para essa célula é demonstrada na tabela 4. Pode-se construir uma célula com três saídas, porém, torna-se redundante para um comparador de 1 bit, já que com apenas duas saídas pode-se deduzir o resultado da terceira.

Tabela 4: Tabela verdade construída a partir da definição das variáveis de saída de um comparador de 1 bit.

A_i	B_i	$Y_{1,i}$	$Y_{3,i}$	$Y_{1,i-1}$	$Y_{3,i-1}$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0

Nota-se que entradas com $Y_{1,i}, Y_{3,i}$ iguais a 1 e 1 simultaneamente são omitidas pois, pela definição da célula comparadora, não é possível a célula receber essas entradas. A partir da tabela 4, pode-se, então construir as funções lógicas. Outro ponto a se destacar é que a entrada inicial (primeira rede da célula) são os bits mais significativos de A e de B. Logo, as células seguintes têm menos prioridade que a as anteriores nas comparações. Portanto, se o bit A_n for maior que B_n , por exemplo, não importam os $n-1$ bits restantes, pois A será maior do que B.

$$\begin{aligned}
 Y_{1,i-1} &= Y_{1,i}\overline{Y_{3,i}} + (\overline{Y_{1,i}})(\overline{Y_{3,i}})AB \\
 Y_{3,i-1} &= \overline{Y_{1,i}}Y_{3,i} + (\overline{Y_{1,i}})(\overline{Y_{3,i}})\overline{AB}
 \end{aligned} \tag{8}$$

Logo, pode-se realizar a simulação do circuito no simulador Quartus II, a fim de atestar as funções encontradas (8). O circuito montado a partir das relações apresentadas é mostrado na figura 6.

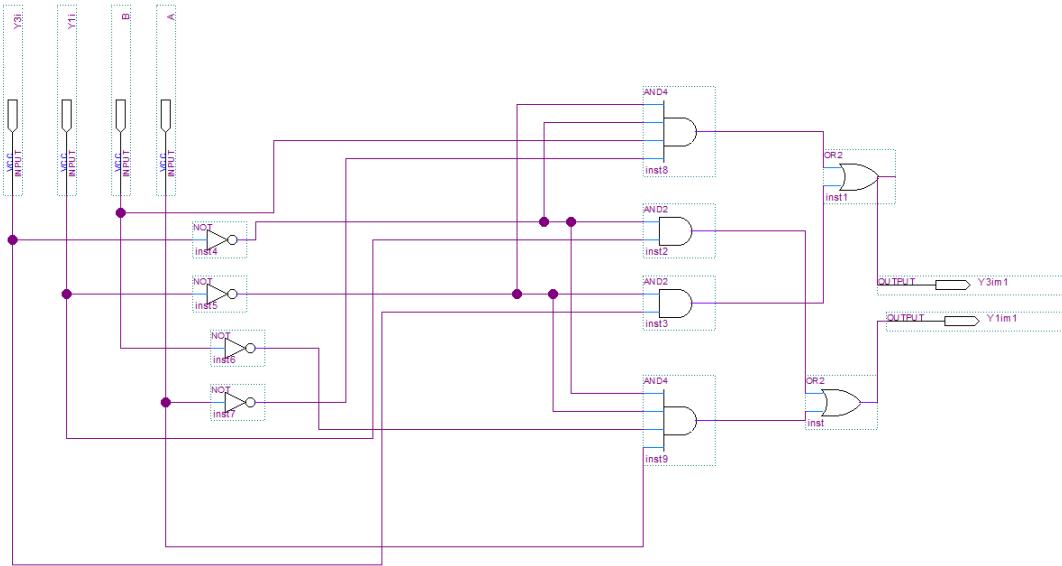


Figura 6: Circuito combinacional de célula básica de comparador montado no simulador

Por fim, realizou-se a simulação para obter o diagrama de temporização do circuito encontrado. A figura 7 apresenta o resultado encontrado. Nota-se como o resultado obtido é exatamente equivalente à tabela verdade encontrada em 4, e o resultado é o esperado.

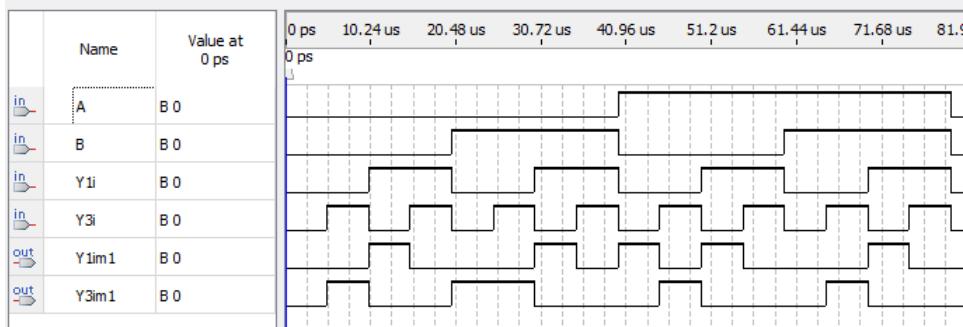


Figura 7: Diagrama de temporização para a célula básica de um comparador de 1 bit.

6.2: Projeto e simulação de um circuito multiplicador de 2 bits

Para a implementação do circuito, considere o mecanismo de multiplicação abaixo:

$$\begin{array}{r}
 & A_1 & A_0 \\
 X & B_1 & B_0 \\
 \hline
 P & Q & B_0A_1 & B_0A_0 & + \\
 & & B_1A_1 & B_1A_0 \\
 \hline
 M_3 & M_2 & M_1 & C_0
 \end{array}$$

Com efeito, podemos notar que

$$C_0 = B_0A_0 \quad (9)$$

Note-se que M_1 é a soma de dois números de 1 bit: se exclusivamente um deles for 1, o resultado é 1; se os dois são 0, o resultado é 0; se os dois forem 1, o resultado é 0 e Q é 1 (*carry*). Dessa forma, podemos concluir que

$$M_1 = B_0A_1 \oplus B_1A_0 \quad (10)$$

$$Q = B_0A_1(0 + B_1A_0) + 0B_1A_0 = (B_0A_1)(B_1A_0) \quad (11)$$

A Relação (11) decorre do fato de que C_{in} é zero para tal soma, já que B_0A_0 não gera *carry in*. Usando o mesmo raciocínio, pode-se obter expressões para M_2 e M_3 , de modo que

$$M_2 = (B_0A_1)(B_1A_0) \oplus B_1A_1 \quad (12)$$

$$M_3 = P = ((B_0A_1)(B_1A_0))(B_1A_1) \quad (13)$$

Com tais resultados, pode-se montar a função em circuito via simulador, cujo resultado consta na Figura 8.

Com efeito, o respectivo resultado do diagrama de temporização para as multiplicações desejadas, isto é, para as entradas dadas, é denotado na Figura 9.

Nota-se, por fim, que o resultado obtido pela simulação foi de acordo com as multiplicações propostas, confirmando, então, o projeto de multiplicador realizado.

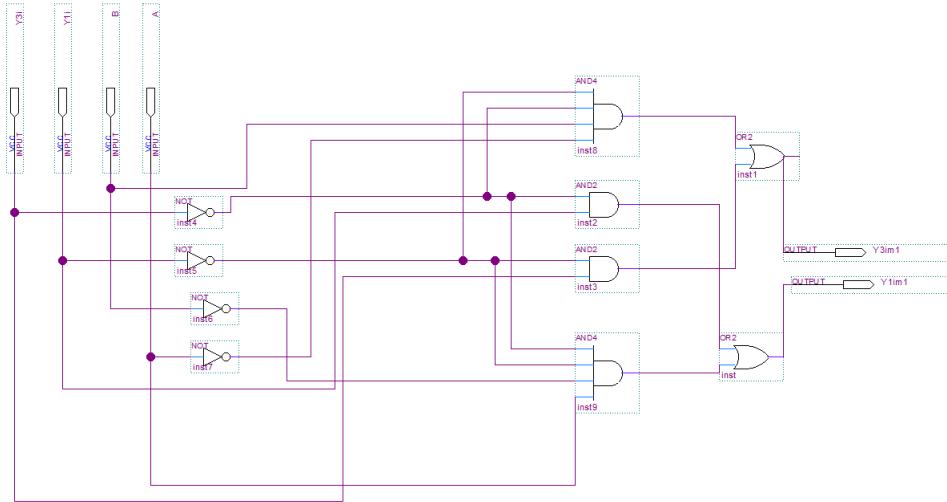


Figura 8: Construção de circuito que sintetiza um multiplicador de dois bits.

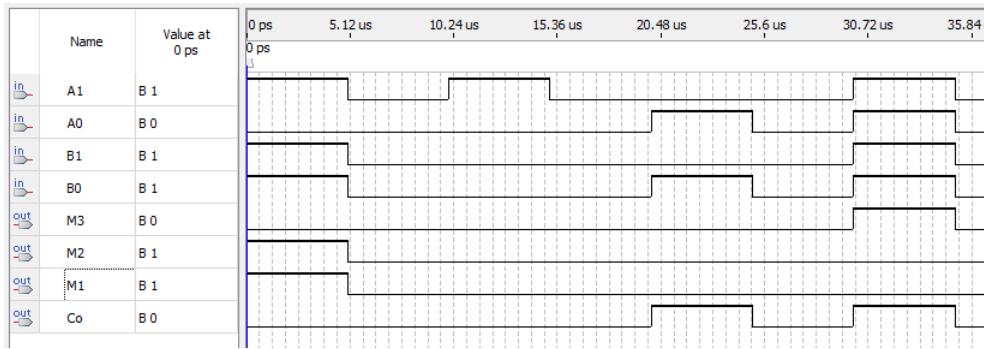


Figura 9: Diagrama de temporização para o multiplicador de 2 bits para as multiplicações determinadas.

7: Referências

1. Fotos e vídeos experimentais armazenados em *google drive*. Pode ser acessado em <http://drive.google.com/drive/folders/1UnXvj62163zG-k0ZmW_0KsiaUpy-1W0j?usp=sharing>. Acesso permitido para usuários de @ga.ita.br.