Universidade Federal do Rio Grande Do Norte

Campus Central (Natal)

Relatório Técnico

Ênfase em Mecatrônica

Aluno: Mateus de Assis Silva.

Professor: Samaherni Morais Dias

Disciplina: Circuitos Digitais - Laboratório

Implementação de Somador Completo em Matriz De Contatos

Introdução:

Já sabemos que os “tijolos” dos circuitos lógicos, as portas lógicas, são abstrações de uma circuitaria baseada em transistores. Com essas podemos criar etapas mais altas de abstrações, como multiplexadores, decodificadores, dentre outros [1].

Uma das abstrações bastante utilizadas são os blocos de controle, capazes de computar entradas e saídas de dados. Tais elementos consegue interpretar eventos, como perceber que um botão foi apertado, por exemplo. Outro constituinte importante é o bloco operacional, capaz de operar dados de entrada ou saída. Nesse documento, descreverá-se o somador completo [1].

O objetivo dessa atividade é implementar um circuito eletrônico numa matriz de contatos capaz de somar dois números binários (A e B) de dois dígitos cada. Por fim, pretende-se exibir o resultado em um *display* de 7 segmentos, com um *led* externo representando o resultado extrapolado do cálculo (*carry-out*, em inglês).

Desenvolvimento:

De forma a poder construir em *hardware* tal sistema, esse foi dividido em módulos menores, seguindo as instruções contidas na atividade disponibilizada pelo professor. Trata-se da estratégia de emular a soma tal qual seres humanos fazem, combinando bits de importância semelhante [2]. Tal decomposição do sistema pode ser vista abaixo, na Figura 1.

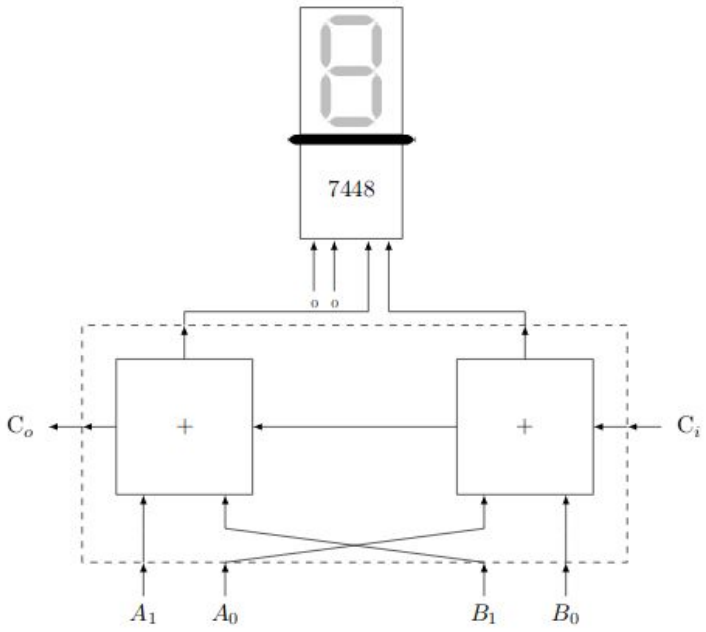


Figura 1: Sistema somador de dois números binários.

Como pode ser visto na figura acima, o sistema é composto de módulos que somam bits de magnitude semelhante. Isto é, considere um número A = A_1A_0 e um número B = B_1B_0. Soma-se, nesse cenário, A_1 com B_1, e A_0 com B_0. Além disso, considera-se que um bit resultante de uma soma anterior (*carry in* , do inglês, “vem um”) pode ser inserido no sistema, através de C_i. Por fim, caso exista algum bit resultante, ele é levado adiante (*carry out*, do inglês, “vai um”), através da variável C_o. Cada sub módulo somador de dois bits pode ser visto abaixo.

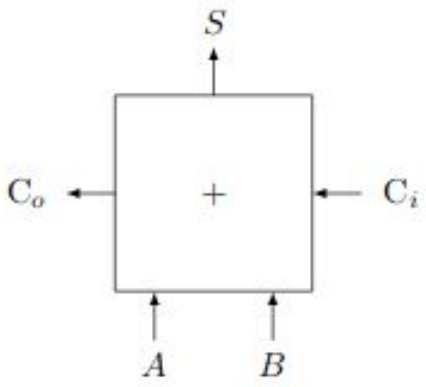


Figura 2: Somador completo de dois bits.

Após a delimitação desses componentes, passou-se à dedução do comportamento do somador completo de dois bits. Para tanto, foi construída a tabela verdade relativa ao componente e a dedução subsequente das equações para C_o e para S. Tais cálculos podem ser vistos no Anexo A. É importante notar que, para esse sub módulo, A e B são números de um bit cada.

Tabela 1: Expressões para C_o e S

Saídas	**Expressões**
C_o	$AB + BC_i + AC_i$
S	$A'B'C_i+A'BC_i'+ABC_i + AB'C_i'$

A tarefa ordenada também exigia a utilização de um tipo de porta lógica apenas. Assim, precisou-se utilizar a universalidade das portas de saída negada (NAND ou NOR) [3] para simularmos o comportamento das outras portas (AND, NOT e OR). Partindo dessa premissa, considerou-se as seguintes relações.

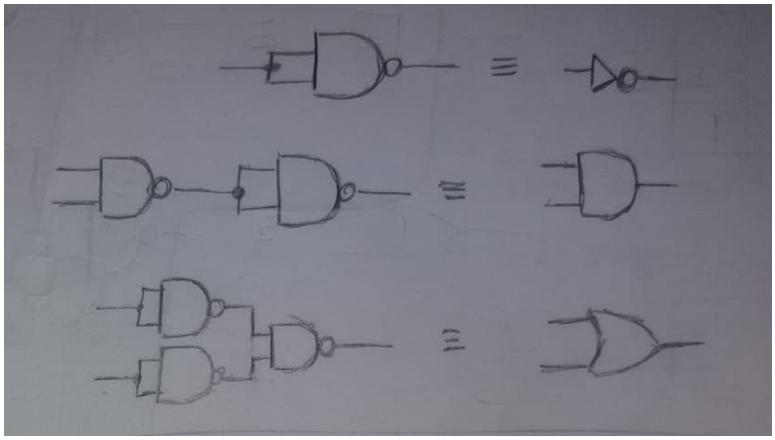


Figura 3: Universalidade da porta NAND.

Após considerar tais relações, o circuito final ficou da seguinte forma.

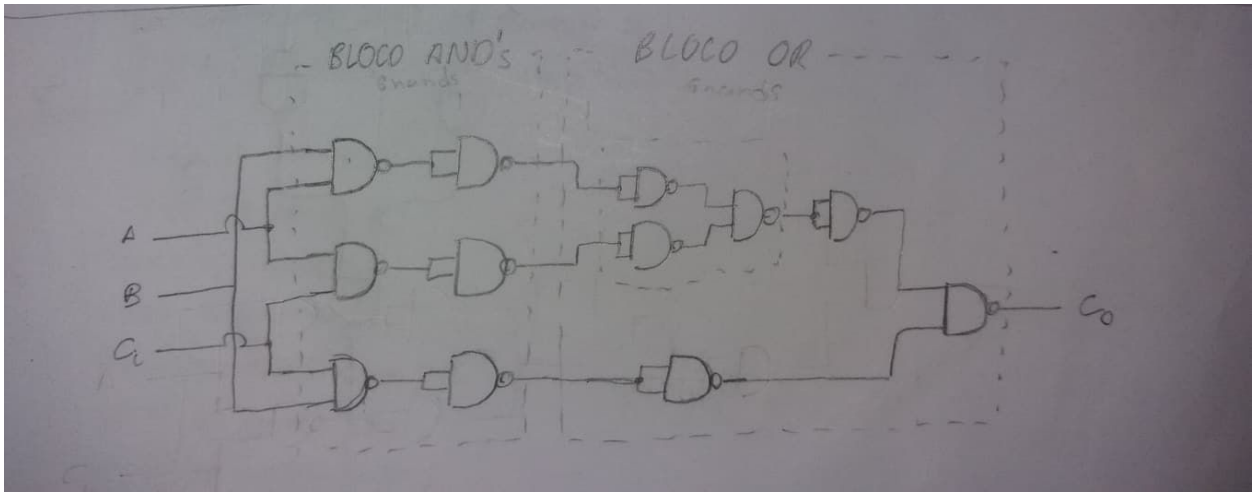


Figura 4: Circuito C_o Inicial.

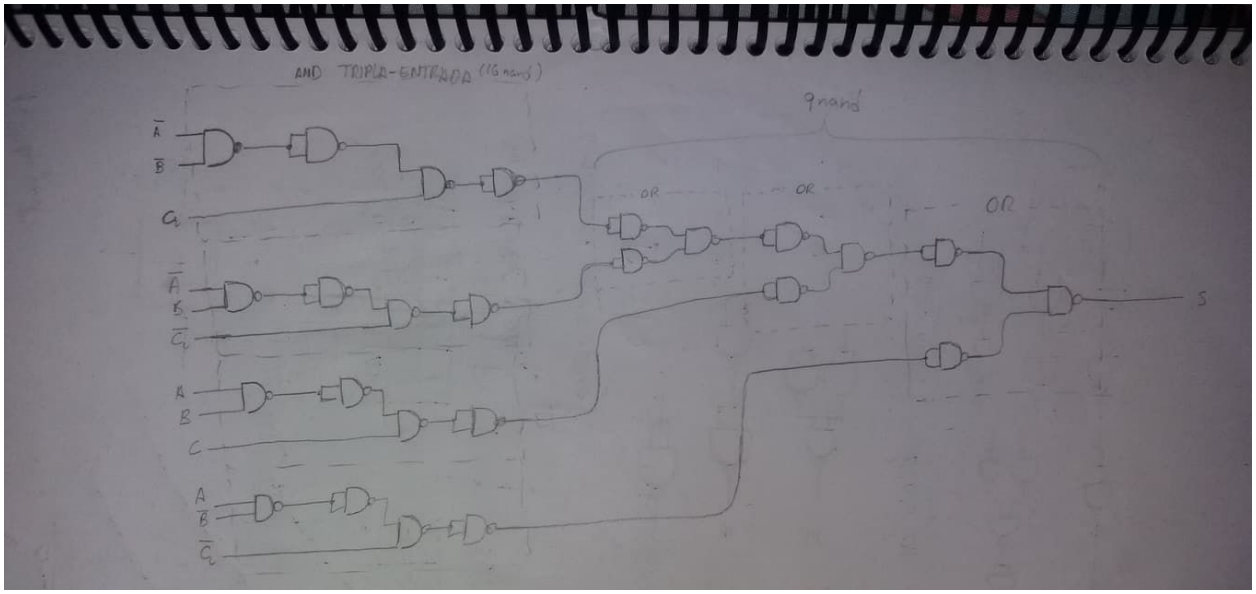


Figura 5: Circuito S Inicial.

Tal circuito completo possui em torno de três dezenas de portas lógicas. Após o conselho de alguns colegas, resolveu-se buscar alguma simplificação, e encontrou-se aquela apresentada no livro de Sistemas Digitais, de autoria de Frank Vahid [4]. Tal simplificação é sobre a saída S e se encontra representada abaixo. Como pode ser facilmente checado na Tabela Verdade do sub módulo (Figura apresentada no Anexo A), a saída S se comporta como uma XOR de três entradas.

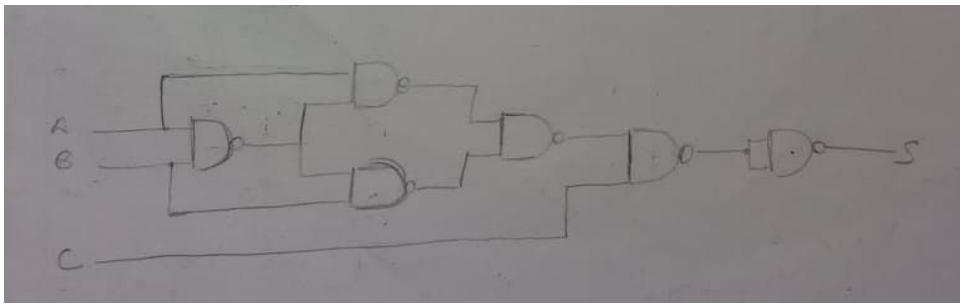


Figura 6: Simplificação sobre S.

Com essa simplificação, o número de portas lógicas diminuiu de trinta e seis (36) para doze (12). Iniciou-se a implementação desse comportamento, quando uma nove referência relativa a uma implementação mais sucinta do circuito foi descoberta [5]. Utilizando essa implementação sucinta, obtém-se o circuitos de ambas as saídas com apenas nove (9) portas lógicas.

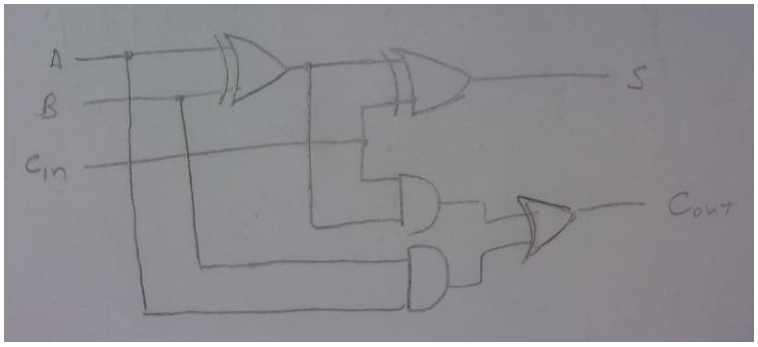


Figura 7: Circuito Sucinto.

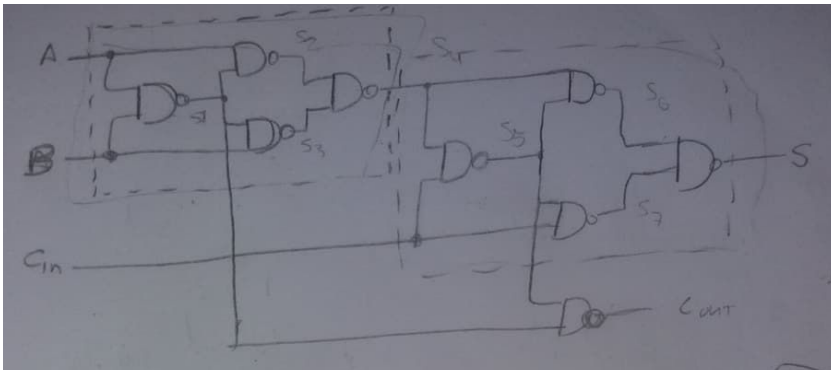


Figura 8: Circuito em Portas NAND.

Antes da implementação do circuito em matriz de contatos, foi-se simulado em *software* via arquivos *.vhd* e *.do*, e seu resultado se encontram abaixo. Os códigos podem ser visualizados no anexo B. Como pode ser observado, os resultado desse circuito são iguais ao da tabela verdade do somador completo de dois bits.

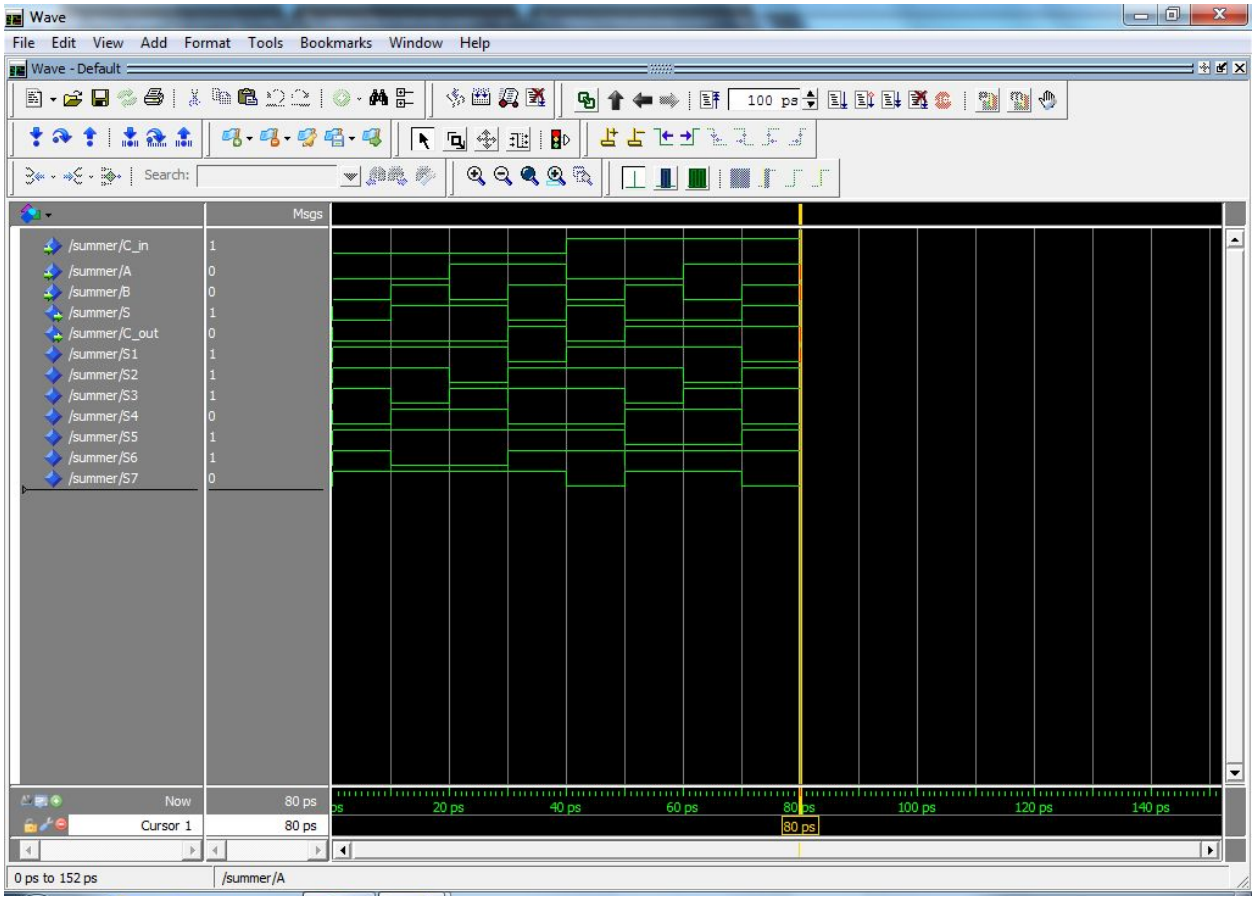


Figura 9: Simulação do Circuito Simplificado.

A simulação demonstrou a capacidade do circuito mais sucinto em implementar o comportamento do somador. Dado que essa construção da circuitaria não foi demonstrada formalmente, após visualizada sua eficiência, partiu-se para a implementação.

Ora, os módulos foram construídos e conectados conforme consta na Figura 1. Daí, faltou apenas a conexão do conversor BIN-BCD. Para tanto, utilizou-se o Circuito Integrado (chip) 7448, capaz de gerenciar essa conversão numérica [67]. O chip teve suas entradas 7 e 1 conectadas às saídas das somas de menor importância e maior importância, respectivamente. As saídas 2 e 8 foram aterradas [7]. Outras configurações foram adotadas, relativas ao funcionamento interno do CI, e fogem ao escopo desse relatório. Por fim, foi adicionado um LED à saída C_o do último módulo.

Após a implementação do sistema, bastou-se testar o sistema completo para algumas combinações específicas de entradas.

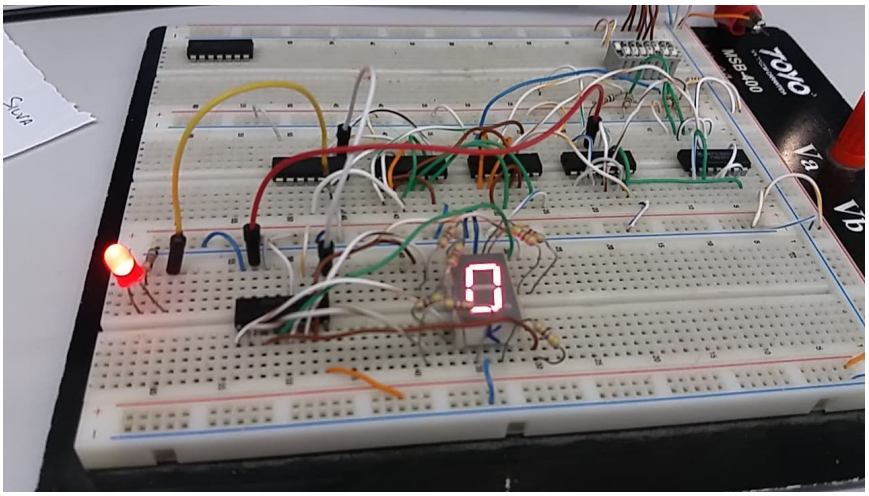


Figura 10: A = 10, B = 10 e C_{in} = 0.

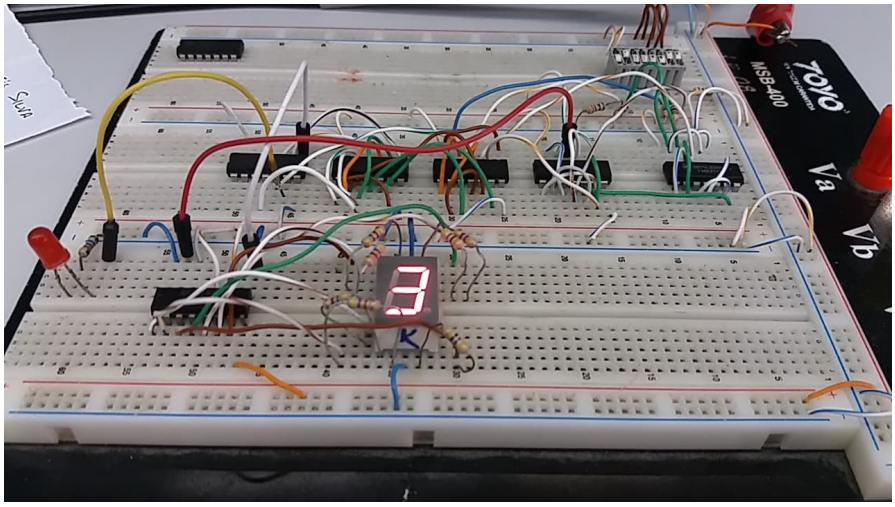


Figura 11: A = 01, B = 01 e C_{in} = 1.

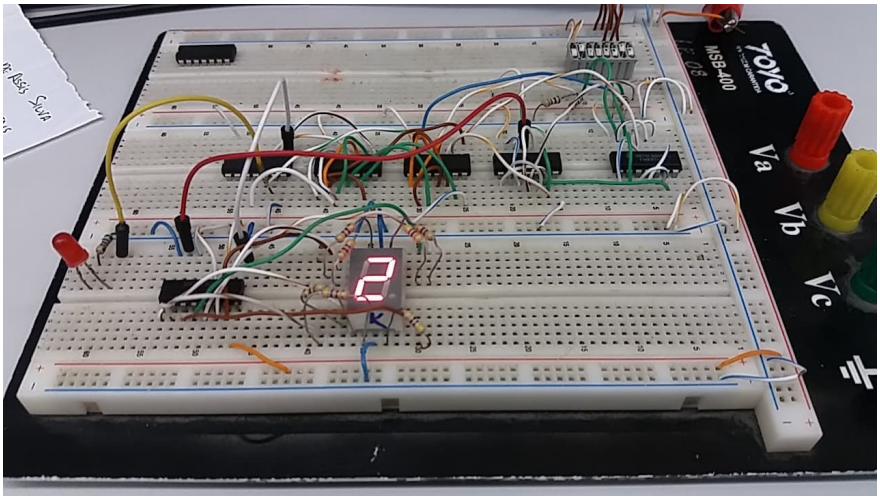


Figura 12: A = 01, B = 01 e C_{in} = 0.

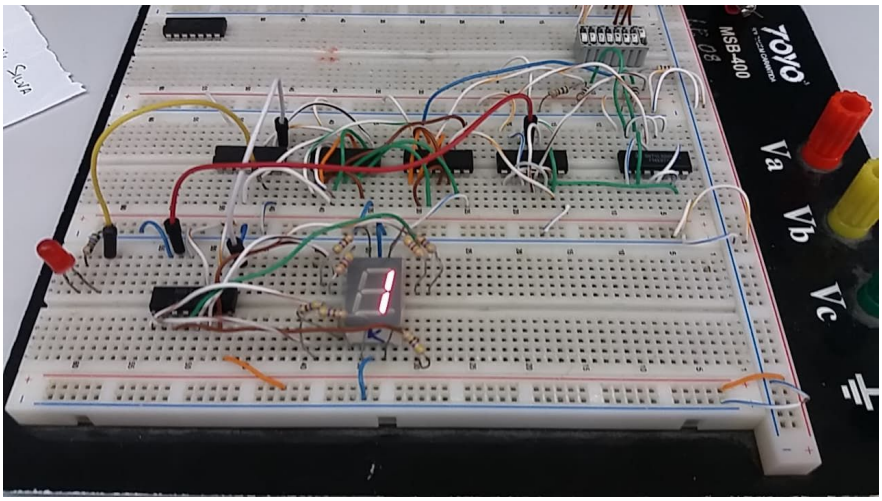


Figura 13: A = 00, B = 00, C_{in} = 1.

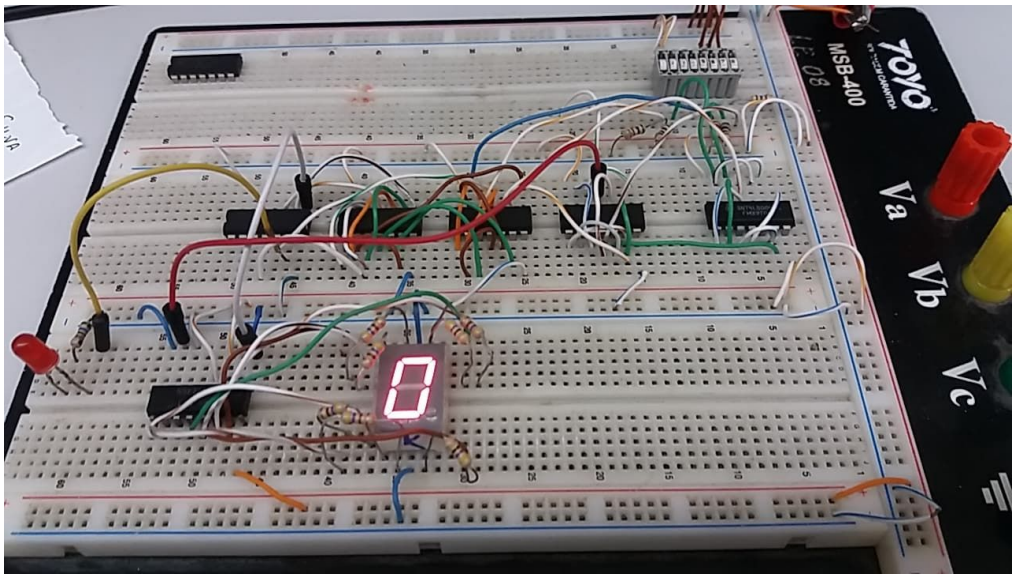


Figura 14: A = 00, B = 00 e C_{in} = 0.

Conclusão:

Ao longo do experimento, notou-se, novamente, a necessidade de modularizar-se o comportamento do circuito mais global. Também ficou claro a necessidade de simplificar ao máximo a expressão obtida, muitas vezes precisando-se pesquisar implementações conhecidas.

Também é importante registrar as etapas do experimento, além de testar cada nova conexão separadamente. Para tanto, uma simulação computacional pode ser de grande auxílio. Por fim, notou-se a necessidade de simular algum submódulo, caso seu comportamento não tenha sido demonstrado, antes da integração deste ao sistema completo, de forma a tornar mais claro onde um possível erro se localiza.

Referências:

[1] Sistemas Digitais - Projetos de Otimização e HDLs. Vahid, Frank - Bookman Editora. 2008. P.166 - 167.

[2] Sistemas Digitais - Projetos de Otimização e HDLs. Vahid, Frank - Bookman Editora. 2008. P.183.

[3] **Teoremas da Álgebra de Boole**, IFSC, Campus São José, Curso Técnico Integrado em Telecomunicações. Acesso em 13 ago 2019. Disponível em:<https://wiki.sj.ifsc.edu.br/wiki/images/b/bd/TEOREMAS-POSTULADOS_com_circuitos.pdf>.

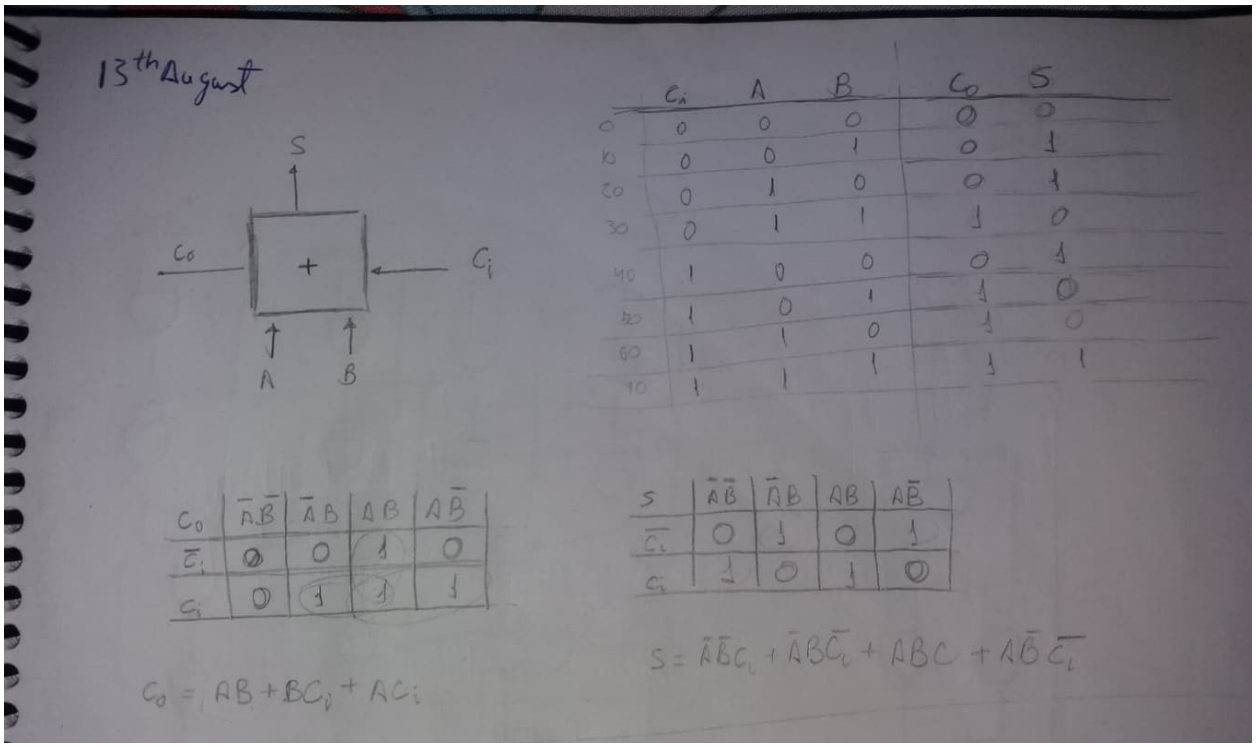
[4] Sistemas Digitais - Projetos de Otimização e HDLs. Vahid, Frank - Bookman Editora. 2008. P.185.

[5] EEWEB, *Full-Adder NAND Equivalent*. Acesso em 15 ago 2019. Disponível em: <https://www.eeweb.com/quizzes/full-adder-nand-equivalent>.

[6] *Electronics Tutorials*, *Binary Coded Decimal*. Acesso em 13 ago 2019. Disponível em <https://www.electronics-tutorials.ws/binary/binary-coded-decimal.html?fbclid=IwAR0bPDtT6NivNQDkMXVctmrHwBb1Ip9ycs2RXY59gVLok96g5O7R3Hsz0YQ>.

[7] *Make Your Own Chip*, *BCD to 7-Segment Decoder-Driver (High - On) For Common Cathode Displays*. Acesso em 12 ago 2019. Disponível em <http://makeyourownchip.tripod.com/7448.html?fbclid=IwAR2qaW-WzXWrIlgSb8zDTcFEBjk_1IbvlxwLCOPitezG9bpcgkCNBDpCDCo>.

Anexo A: Simplificações



Anexo B: Códigos

Script do Módulo.

```
-- simulation of the circuit
-- complete summer of two bits
-- Code by Mateus de Assis Silva
-- for more info, please go to github.com/mtxslv/DigitalCircuits

entity summer is
  port(A,B,C_in: in bit;
      S, C_out: out bit);
end summer;

architecture summer_ckt of summer is
  signal S1, S2, S3, S4, S5, S6, S7: bit;
begin
  S1 <= A NAND B;
  S2 <= S1 NAND A;
  S3 <= S1 NAND B;
  S4 <= S2 NAND S3;
  S5 <= S4 NAND C_in;
  S6 <= S4 NAND S5;
  S7 <= S5 NAND C_in;
  S <= S6 NAND S7;
  C_out <= S1 NAND S5;
end summer_ckt;
```

Simulação do Módulo.

```
vsim summer

add wave *
force C_in 0 0, 1 40
force A 0 0, 1 20 -repeat 40
force B 0 0, 1 10 -repeat 20
```

```
run 80
```