Projeto VLSI
Eletrônica II – Engenharia de Computação
(revisado em julho de 2018)

Ezequiel Orloski, Guilherme Corrêa Camargo, Mateus Felipe da Silva Junges

Abstract—Since the computer was invented, the function of performing calculations is among the major functions of this tool, and it is, of such great importance that many of the calculations made by computers today are impossible to perform in a timely manner by humans. The process of solving calculations in a computer is performed in an Arithmetic Logic Unit, or ALU. This process is made by receiving information in binary form. When received, this information is manipulated by arithmetic logic circuits. For ALU implementation are used adders and subtractors, logical functions, comparison operators, among others. In this work, we will discuss the implementation and design of a complete adder.

Index Terms— Full Adder, VHDL, layout, electric, transitor

1 Introdução

Neste relatório será realizada a implementação de um somador completo de 1 bit e um somador completo de 8 bits, mostrando a síntese e as simulações de cada um dos somadores, além de fazer a representação física dos somadores e portas lógicas utilizadas, fazendo uso do software electric, em sua versão 9.07.

Desde que o computador foi inventado, a função de realizer cálculos está entre as principais funções desta ferramenta, e tem uma importância tão grande que, muitas das contas realizadas hoje em dia por computadores são impossíveis de serem realizadas em tempo hábil por seres humano.

O processo de resolução de cálculos em um computador é realizado em uma Unidade Lógica Aritmética, ou ALU. Esse processo é feito a partir do recebimento de informações em forma binária. Quando recebida, esta informação é manipulada por circuitos lógicos aritméticos. Para implementação da ALU são utilizados somadores e subtratores, funções lógicas, operadores de comparação, entre outros. Neste trabalho, vamos discutir a implementação e projeto de um somador completo.

O modulo do somador tem a função de realizer as operações de soma, tendo como entrada uma cadeia de números binários. Um somador é composto por uma unidade básica, que é responsável por somar dois bits de entrada, A e B, e de uma unidade maior, que é composta por um conjunto de somadores básicos. Os processos realizados neste trabalho são descritos e detalhados ao decorrer do presente relatório.

Um somador é um circuito lógico usado em computadores para somar digitos binários. O somador completo aceita três entradas digitais (bits): os dois bits que serão somados e (A e B) e um bit de carry de entrada, ou carry in, que são somados e resultam em um sinal de saída S e carry de saída, ou carry out.

2 Somador completo de 1 bit

2.1 Introdução teórica

Um somador, conforme mencionado anteriormente, é um circuito lógico utilizado em computadores para somar digitos binários. O somador completo aceita três entradas digitais, que serão somadas: os dois bits de entrada, A e B, e um bit de carry de entrada, ou carry in, que são somados e resultam em um sinal de saída S, além de um carry de saída, ou carry out.

O somador completo de 1 bit surgiu da necessidade de modularizar as operações de soma, dada a complexidade de tais operações quando realizadas de forma conjunta. Assim, modularizando o processo, temos o somador completo de 1 bit, que se encarrega apenas da soma de dois bits de entrada e mostrá-lo na saída. O processo de soma, leva em conta o que é chamado, confome citado antes, de carry in, que pode ou não estar presente. Este bit é o resultado de somas anteriores, que acabaram levando ao estouro base, chamado de overflow, e, por consequência, há a necessidade de inclusão da soma do mesmo em uma próxima soma.

O somador completo de 1 bit é compost por:

- Entradas:
  - A: Primeiro bit a ser somado;
  - B: Segundo bit a ser somado.
  - Carry in: Bit resultante de uma soma anterior, que é resultado de um overflow.
- Saídas:
  - S: Bit resultante da operação de soma;
  - Carry out: Bit resultante do estouro da soma na base 2.

2.2 Tabelas verdade e mapas de Karnaugh

Uma função lógica geralmente é simplificada para obter um circuito mais simples e econômico, usando menos componentes. É possível fazer esta simplificação utilizando teoremas, postulados e identidades da álgebra de boole, mas nem sempre este trabalho é fácil, sendo que nem sempre é possível saber com certeza se a equação obtida é a menor possível.

O uso do mapa de Karnaugh é um método mais simples, uma vez que é baseado na tabela verdade da função lógica.

O procedimento realizaodo para obter as funções lógicas é apresentado abaixo.

A seguir, a tabela verdade do somador completo de 1 bit.

A	B	Carry In	S	Carry Out
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

Tabela 1: Tabela verdade do somador completo de 1 bit

A partir da tabela verdade acima, é possível obter a função logica que melhor representa as saídas digitais deste circuito. A soma será resultado da operação de ou exclusive, ou, XOR, entre os bits de entrada A, B e carry in. O bit de carry out é o resultado da operação de OR de três resultados diferentes. São eles: a operação AND entre A e B, operação AND entre A e o bit de carry in, e, a operação AND entre os bits B e carry in.

A seguir, é apresentado os mapas de Karnaugh que representam a função que gera a saída **S** do modulo:

AB / Cin	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Tabela 2: Mapa de karaugh para a função lógica da saída do circuito.

Portanto, a função lógica que representa o circuito será:

S = A xor B xor CarryIn.

O mapa de Karnaugh para a função lógica que representa a saída do bit carry out é apresentado abaixo:

AB / Cin	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Tabela 3: Mapa de Karnaugh para a saída do bit de carry out.

Portanto, a função lógica que representa a saída no bit carry out é a seguinte:

CarryOut = (A and B) or (A and CarryIn) or (B and CarryIn).

2.3 Código VHDL e Síntese RTL do somador completo de 1 bit

Para o somador completo de 1 bit, realizamos a implentação utilizando a liguagem VHDL, e, então, conseguimos a síntese RTL, que é apresentado abaixo.

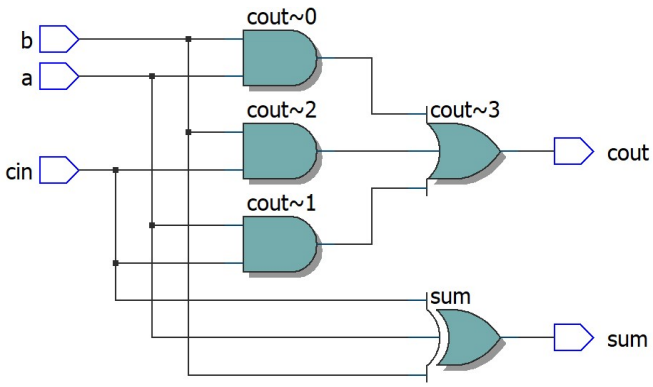

Figura 1: Síntese RTL do somador completo de 1 bit.

Conforme representado na figura 1, podemos perceber a implementação da função lógica do bit de carry de saída e também do resultado da soma através do uso de portas lógicas AND, OR e XOR.

As entradas são ligadas a três portas AND, de modo que na primeira porta, cout~0, são ligadas as entradas de bit A e B, na porta cout~2, é ligada a entrada o bit B e a entrada de transporte de bit de carry in, cin. Da mesma maneira, é ligada, na terceira porta AND (cout~1), os bits A e a entrada de transporte de bit, carry in.

Com o circuito RTL montado, podemos realizar a simulação do mesmo, conforme demonstrado abaixo.

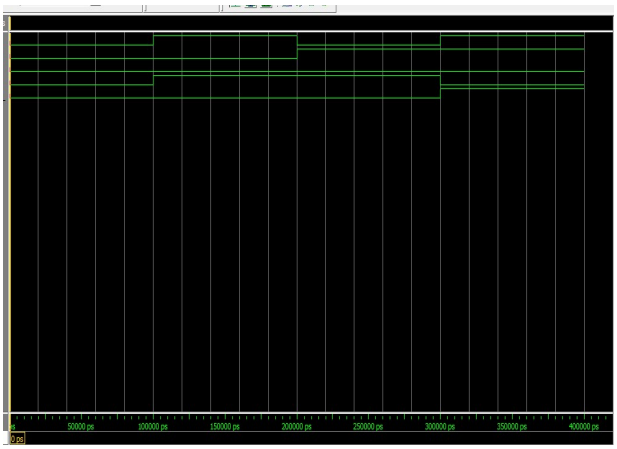

Figura 2: Simulação do circuito RTL para o somador completo de 1 bit

Conforme demonstrado na figura 5, podemos perceber o funcionamento correto do somador completo de 1 bit.

O código para implementação em linguagem VHDL é representado abaixo:

```
library ieee;                                           -- biblioteca ieee
use ieee.std_logic_1164.all;                            -- biblioteca 1164
entity somadorlbit is                                   -- declaração da entidade
   port(                                                -- declaração das portas
       a:    in std_logic;                              -- entrada a
       b:    in std_logic;                              -- entrada b
       cin:  in std_logic;                              -- carry in
       sum:  out std_logic;                             -- resultado da soma
       cout: out std_logic);                            -- carry out
end somadorlbit;                                        -- fim da entidade
architecture somador1 of somadorlbit is                 -- inicio da arquitetura
   begin                                                -- inicio
       sum <= a xor b xor cin;                          -- resultado da soma
       cout <= (a and b) or (cin and a) or (cin and b);-- carry out
end somador1;                                           -- fim da arquitetura
```

Figura 3: Representação do código VHDL para implementação do somador completo de 1 bit

Além de permitir o projeto de circuitos digitais de forma mais rápida, a linguagem VHDL também apresenta formas de testar o código em diversos níveis, o que garante uma maior confiabilidade no código. Uma dessas formas é o testbench. O testbench para o circuito do somador completo de 1 bit é apresentado a seguir.

```
LIBRARY ieee;                                           -- biblioteca ieee
USE ieee.std_logic_1164.all;                            -- biblioteca 1164
ENTITY somadorlbit_vhd_tst IS END;                      -- entidade vazia
ARCHITECTURE somadorlbit_arch OF somadorlbit_vhd_tst IS -- inicio da arquitetura
   SIGNAL a, b, cin : STD_LOGIC;                        -- entradas
   SIGNAL cout, sum : STD_LOGIC;                        -- saidas
   COMPONENT somadorlbit PORT (                         -- istancia do componente
      a, b, cin : IN STD_LOGIC;                         -- entradas do componente
      cout, sum : OUT STD_LOGIC);                       -- saidas do componente
   END COMPONENT;                                       -- fim do componente
   BEGIN                                                -- inicio
      i1 : somadorlbit PORT MAP(                        -- mapeamento das portas
      a => a, b => b, cin => cin, cout => cout, sum => sum);--mapeamento das portas
   processo : PROCESS                                   -- inicio do processo
      BEGIN                                             -- inicio
         a <= '0'; b <= '0'; cin <= '0';                -- entradas "000"
         wait for 100 ns;                               -- pausa 100 nanosegundos

         a <= '1'; b <= '0'; cin <= '0';                -- entrada "100"
         wait for 100 ns;                               -- pausa 100 nanosegundos

         a <= '0'; b <= '1'; cin <= '0';                -- entrada "010"
         wait for 100 ns;                               -- pausa 100 nanosegundos

         a <= '1'; b <= '1'; cin <= '0';                -- entrada "110"
         wait for 100 ns;                               -- pausa 100 nanosegundos
         WAIT;                                          -- espera longa
   END PROCESS always;                                  -- fim do processo
END somadorlbit_arch;                                   -- fim da arquitetura
```

Figura 4: Testbench para o circuito somador completo de 1 bit

O testbench é, na verdade, uma especificação VHDL que é simulada por um simulador VHDL. Ao final da realização do testbench, verificamos que o circuito está funcionando de maneira correta.

3 SOMADOR COMPLETOD E 8 BITS

Como demonstrado anteriormente, os somadores são constituídos de unidades básicas, que são os somadores de 1 bit. Assim, realizamos a implementação de um somador completo de 8 bits, fazendo uso do somado completo de 1 bit, que foi desenvolvido nas páginas que antecedem este item.

O processo realizado para implementação do somador completo de 8 bits são descritos nas páginas que seguem.

3.1 Introdução teórica

Os computadores fazem uso de unidades básicas para compor um somador para mais que 1 bit. Assim, tendo implementado o somador completo de 1 bit, podemos utilizá-lo para implementar o de 8 bits, apenas realizando o mapeamento das portas do mesmo em nosso código VHDL. Este processo é descrito a seguir.

3.2 Síntese RTL e código VHDL para o somador completo de 8 bits

O código para um somador completo de 8 bits em VHDL é apresentado na figura, a seguir.

```
:ary ieee;                                              -- biblioteca ieee
 ieee.std_logic_1164.all;                               -- biblioteca 1164
ity somador8bit is                                      -- declaração da entidade
ort(                                                    -- declaração das portas
   a:    in std_logic_vector(7 downto 0);               -- entrada a
   b:    in std_logic_vector(7 downto 0);               -- entrada b
   cin:  in std_logic;                                  -- carry in
   sum:  out std_logic_vector(7 downto 0);              -- resultado da soma
   cout: out std_logic);                                -- carry out
somador8bit;                                           -- fim da entidade
itecture behavior of somador8bit is                    -- inicio da arquitetura
  component somadorlbit is                             -- instanciação do componente
    port(                                              -- portas do componente
      a:    in std_logic;                              -- entrada a
      b:    in std_logic;                              -- entrada b
      cin:  in std_logic;                              -- entrada carry in
      cout: out std_logic;                             -- saida carry out
      sum:  out std_logic);                            -- soma
  end component;                                       -- fim do componente
  signal aux : std_logic_vector(7 downto 0);           -- sinal auxiliar
  begin                                                -- inicio
     x0 : somadorlbit port map(a(0)   ,b(0),   cin,      aux(1),   sum(0));--mapeamento das portas
     x1 : somadorlbit port map(a(1)   ,b(1),   aux(1),   aux(2),   sum(1));
     x2 : somadorlbit port map(a(2)   ,b(2),   aux(2),   aux(3),   sum(2));
     x3 : somadorlbit port map(a(3)   ,b(3),   aux(3),   aux(4),   sum(3));
     x4 : somadorlbit port map(a(4)   ,b(4),   aux(4),   aux(5),   sum(4));
     x5 : somadorlbit port map(a(5)   ,b(5),   aux(5),   aux(6),   sum(5));
     x6 : somadorlbit port map(a(6)   ,b(6),   aux(6),   aux(7),   sum(6));
     x7 : somadorlbit port map(a(7)   ,b(7),   aux(7),   cout,     sum(7));
behavior;                                              -- fim da arquitetura
```

Figura 5: Código em VHDL para implementação de um somador completo de 8 bits

Com o código VHDL escrito e devidamente compilado, conseguimos a síntese RTL, com as portas utilizadas para sua implementação. A imagem a seguir apresenta a síntese do somador completo de 8 bits:

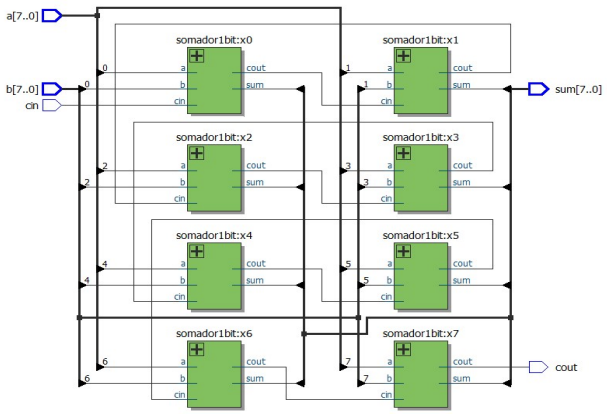

Figure 6: Síntese RTL para um somador completo de 8 bits, utilizando como componentes os somadores completos de 1 bit.

Como pode ser percebido na imagem acima, o componente somador completo de 8 bits utiliza módulos, que são os somadores completos de 1 bit. Isso facilita a implementação, uma vez que é mais fácil projetar o circuito em VHDL, apenas fazendo uso do mapeamento de portas.

Depois de escrever o código e obter a síntese RTL, foi possível realizar o teste do circuito, através de uma simulação RTL e, posteriormente, com o testbench. Os dois procedimentos são apresentados nas próximas imagens.

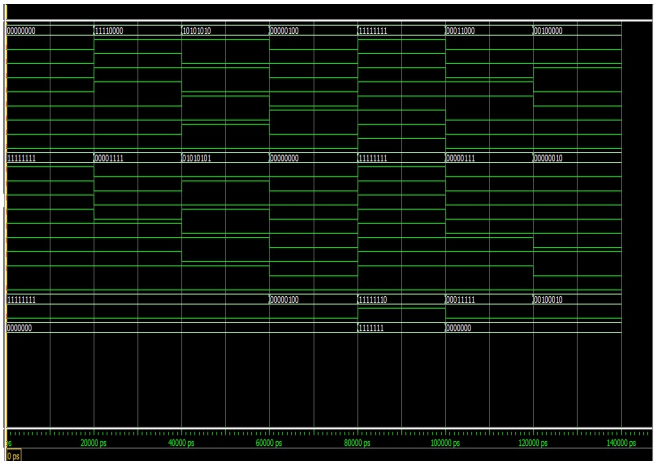

Figura 5: Simulação RTL para um somador completo de 8 bits

A próxima imagem apresenta o código que possibilitou a realização do testebench para o circuito.

```
LIBRARY ieee;                                                   -- biblioteca ieee
USE ieee.std_logic_1164.all;                                    -- biblioteca 1164
ENTITY somador8bit_vhd_tst IS END;                              -- entidade vazia
ARCHITECTURE somador8bit_arch OF somador8bit_vhd_tst IS         -- inicio a arquitetura
    SIGNAL a, b : STD_LOGIC_VECTOR(7 DOWNTO 0);                 -- sinais de entrada
    SIGNAL cin : STD_LOGIC;                                     -- carry in
    SIGNAL cout : STD_LOGIC;                                    -- carry out
    SIGNAL sum : STD_LOGIC_VECTOR(7 DOWNTO 0);                  -- vetor de soma
COMPONENT somador8bit PORT(                                     -- declaração do componente
    a, b : IN STD_LOGIC_VECTOR(7 DOWNTO 0);                     -- vetor de entrada
    cin : IN STD_LOGIC;                                         -- carry in
    cout : OUT STD_LOGIC;                                       -- carry out
    sum : OUT STD_LOGIC_VECTOR(7 DOWNTO 0));                    -- vetor de saida soma
END COMPONENT;                                                  -- fim do componente
BEGIN                                                           -- inicio
    i1 : somador8bit PORT MAP(a => a, b => b,cin => cin,        -- portas do componente
                              cout => cout, sum => sum);        -- portas do componente
calculo : PROCESS                                               -- inicio do processo
BEGIN                                                           -- inicio
        a <= "00000000"; b <= "11111111"; cin <= '0';           -- entrada "110"
        wait for 20 ns;                                         -- pausa 20 nanosegundos
        a <= "11110000"; b <= "00001111";                       -- entrada "110"
        wait for 20 ns;                                         -- pausa 20 nanosegundos
        a <= "10101010"; b <= "01010101";                       -- entrada "110"
        wait for 20 ns;                                         -- pausa 20 nanosegundos
        a <= "00000100"; b <= "00000000";                       -- entrada "110"
        wait for 20 ns;                                         -- pausa 20 nanosegundos
        a <= "11111111"; b <= "11111111";                       -- entrada "110"
        wait for 20 ns;                                         -- pausa 20 nanosegundos
        a <= "00011000"; b <= "00000111";                       -- entrada "110"
        wait for 20 ns;                                         -- pausa 20 nanosegundos
        a <= "00100000"; b <= "00000010"; cin <= '0';           -- entrada "110"
        wait for 20 ns;                                         -- pausa 20 nanosegundos
WAIT;                                                           -- espera longa
END PROCESS calculo;                                            -- fim do processo
```

Figura 6: Código utilizado para realização do testebench para o circuito somador completo de 8 bits

Depois de concluído o desenvolvimento do código para o testebench e sua simulação, verificamos que o código e o circuito está funcionando corretamente.

4 PORTAS LÓGICAS CMOS

4.1 Introdução teórica

Uma porta lógica é um dispositivo que implementa uma função lógica em uma ou mais entradas binárias e produz uma saída binária única.

Existem várias maneiras de implementar portas lógicas utilizando-se de transistores MOSFET. A masis usada é o CMOS, e ele utiliza redes complementares Pull-Up (PUN) e Pull-down, (PDN). Elas são responsáveis por abrir ou fechar os caminhos entre o 1 lógico e a saída e entre o 0 lógico e a saída, respectivamente.

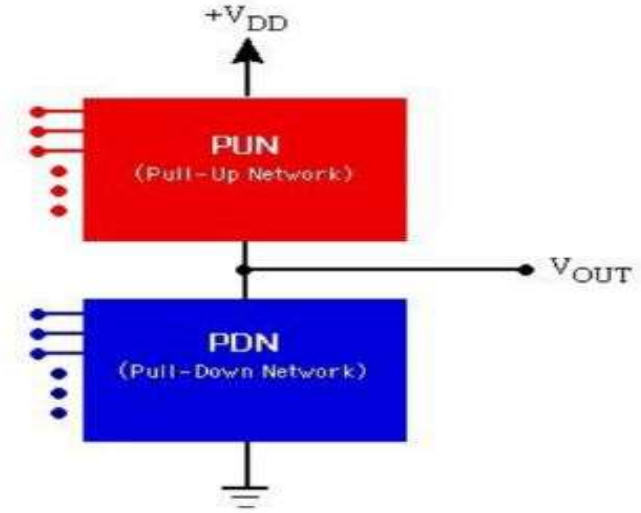

Figura 97: Redes Pull-Down e Pull-Up do CMOS

As redes PDN e PUN devem possuir lógica complementar, de modo que sempre que o sinal 1 é liberado para a saída, o 0 é bloqueado e vice-versa, fornecendo sempre uma saída definida impedindo um curto-circuito.

Em portas lógicas CMOS, são usados transistores PMOS na rede Pull-Up e NMOS na rede Pull-down, dado que os transistores PMOS conduzem bem o nível lógico baixo, enquanto que os transistores NMOS conduzem bem o nível lógico alto. Isso é o que faz com que seja possível implementar lógicas negadas, como NAND, NOR e NOT, sendo necessário colocar uma porta NOT na saída para implementar portas AND ou OR.

Após terminar a implementação do somador de 1 e de 8 bits no Quartus, em linguagem VHDL, prosseguimos para a implementação de cada uma das portas descritas na síntese RTL. O procedimento é descrito a seguir.

4.2 Porta lógica AND

O ícone que caracteriza a porta lógica AND é apresentado na figura abaixo.

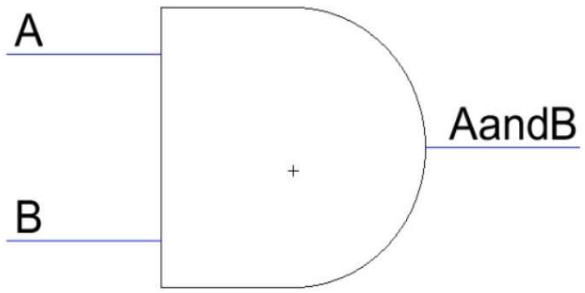

Figura 10: Ícone da porta lógica AND

A porta lógica AND utiliza-se do operador de produto lógico. A saída é igual a 1 se todas as entradas

forem 1. A saída será igual a 0 se uma ou mais entradas forem iguais a 0.

A tabela verdade da porta lógica AND é apresentada abaixo.

A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

Table 4: Tabela verade da porta lógica AND

Seguindo o objetivo deste trabalho, a porta lógica foi implementada utilizando a lógia CMOS. O esquemático da porta AND através de lógica CMOS é mostrado abaixo.

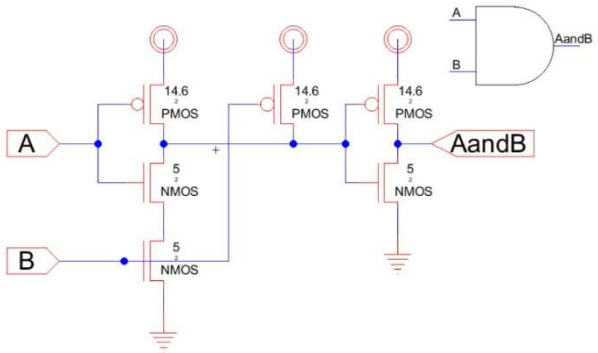

Figura 11: Esquemático da porta lógica AND com lógica CMOS

De acordo com o esquemático acima, utilizando o softwarere Electric, implementamos o layout da porta lógica AND com lógica CMOS. Este layout pode ser visto na próxima imagem.

A simulação desta porta foi feita no software LTSpice, utilizando a seguinte programação:

```
vdd vdd 0 DC 5
Va a 0 pulse 5 0 0 1n 1n 2u 4u
Vb b 0 pulse 5 0 0 1n 1n 1u 2u
.tran 4u
```

Figura 12: Código utilizado para simulação da porta lógica AND

Vale ressaltar que, para simulação, é necessário imcluir uma linha no código acima, especificando o caminho para o seu arquivo com os modelos, da seguinte forma:

.include <path>

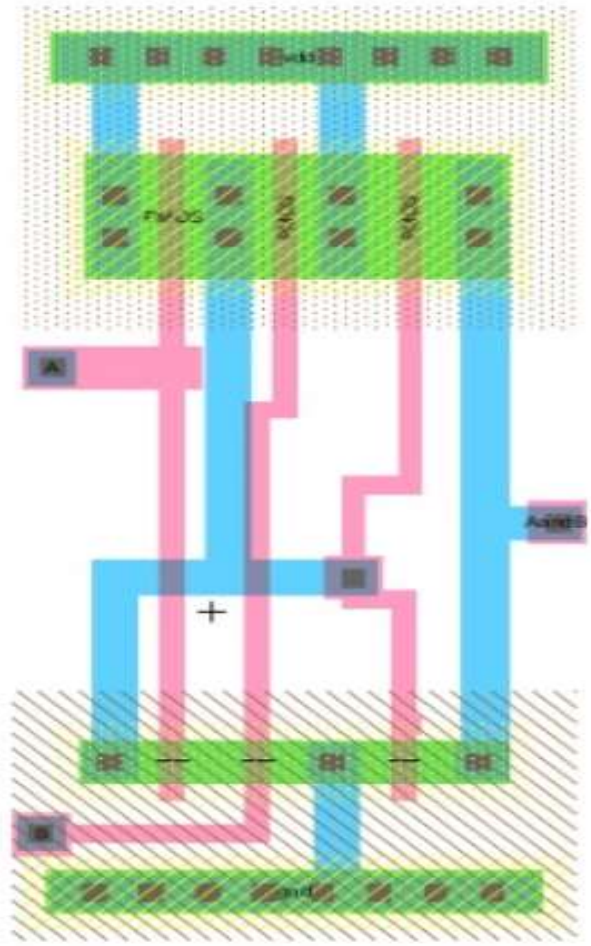

Figura 13: Layout em Electric para uma porta lógica AND

O resultado da simulação desta porta é mostrada noa próxima imagem:

Figura 8: Resultado da simulação da porta lógica AND no software LTSpice

Através desta simulação, podemos perceber que aporta lógica AND está funcionando corretamente, uma vez que a saída é 1 apenas quando as duas entradas tem nível lógico 1.

4.3 Porta lógica OR

O ícone característico de uma porta lógica OR é mostrado a seguir:

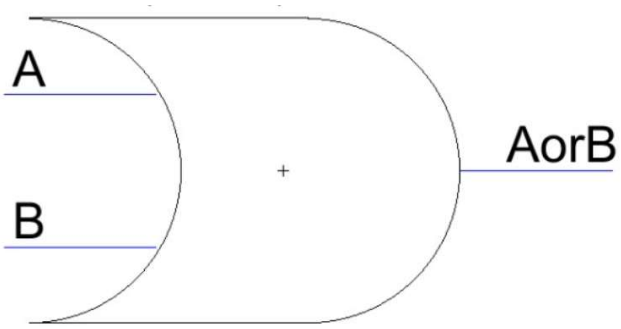

Figure 9: Ícone característico de uma porta lógica OR

A porta lógica OR funciona com a utilização do operador OU, que se caracteriza por apresentar a saída lógica em nível alto quando uma das duas entradas for 1. A tabela verdade desta porta lógica é apresentado a seguir:

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 5: Tabela verdade da função lógia OR

A seguir, é apresentado o esquema da montagem de uma porta lógica OU utilizando a lógica CMOS.

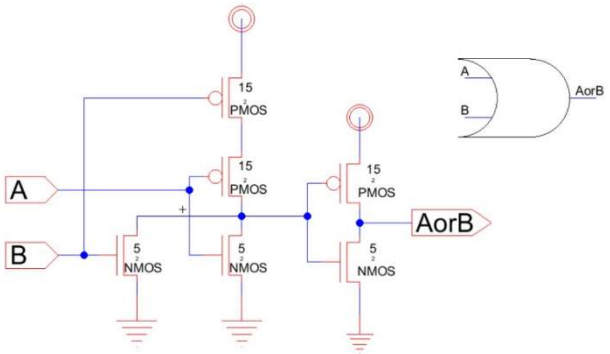

Figura 16: Esquemático para implementação de uma porta lógica OR com lógica CMOS

Através do esquemático representado acima, fizemos o layout do OR CMOS. Além do layout, também foi realizado a simulação do circuito, com o seguinte código:

```
vdd vdd 0 DC 5
Va a 0 pulse 5 0 0 1n 1n 2u 4u
Vb b 0 pulse 5 0 0 1n 1n 1u 2u
.tran 4u
```

Figura 17: Código utilizado para simulação da porta lógica OR

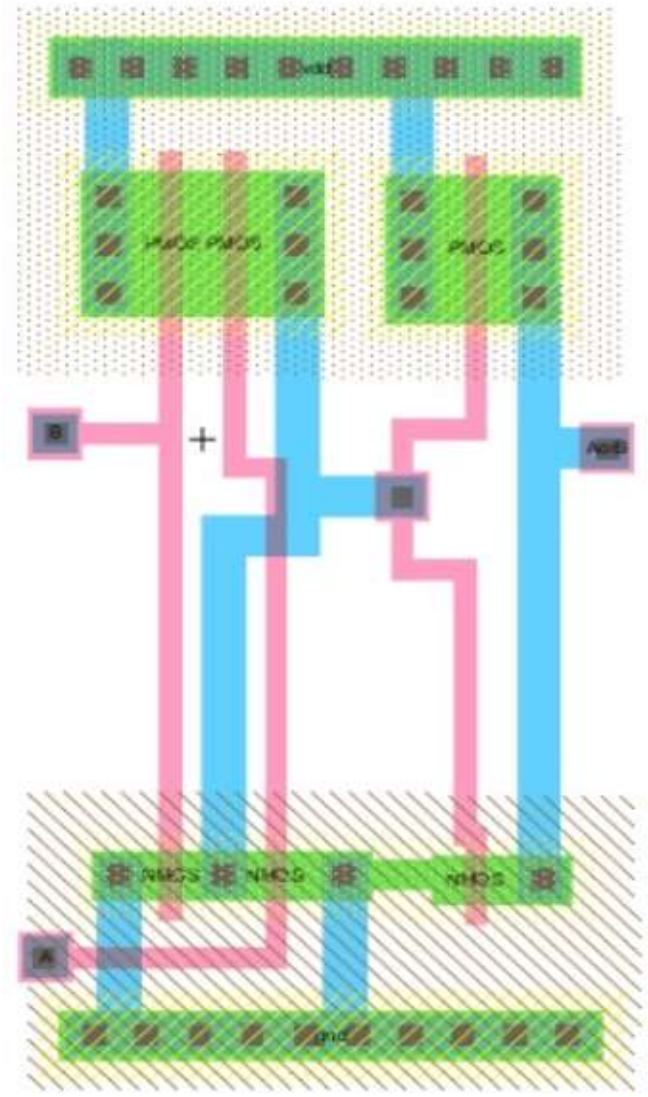

Figura 18: Layout em electric para uma porta lógica OR

Com o códio mostrado anteriormente e fazendo a inclusão do arquivo com os modelos, realizamos a simulação da porta no software LTSpice. O resultado é mostrado a seguir.

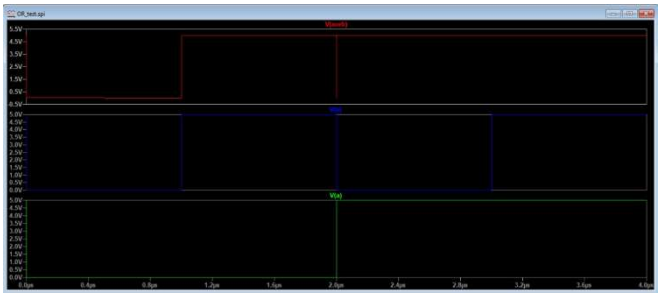

Figura 10: Resultado da simulação da porta lógica OR

Analisando a última imagem, é possível perceber que a porta lógica funcionada maneira correta, uma vez que apresenta a saída em nível lógico alto sempre que uma das entradas está em nível lógico 1.

4.4 Porta lógica XOR

O ícone característico de uma porta XOR é mostrado abaixo:

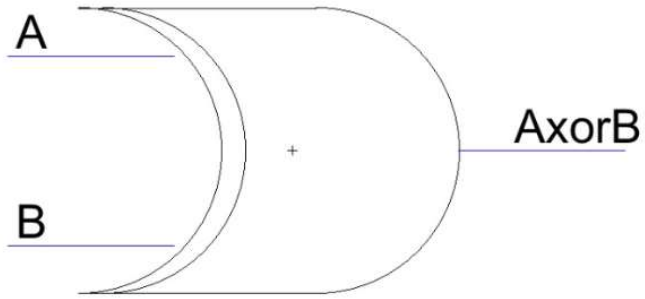

Figura 11: Ícone característico da porta lógica XOR

A seguir, é apresentado o esquemático da porta lógica XOR, utilizando lógica CMOS:

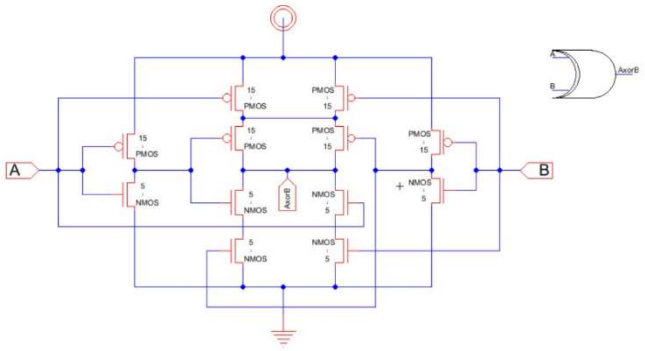

Figura 12: Esquemático para implementação da porta lógica XOR com lógica CMOS

A partir do esquemático descrito acima, realizamos o projeto do layout desta porta, e o mesmo é apresentado a seguir:

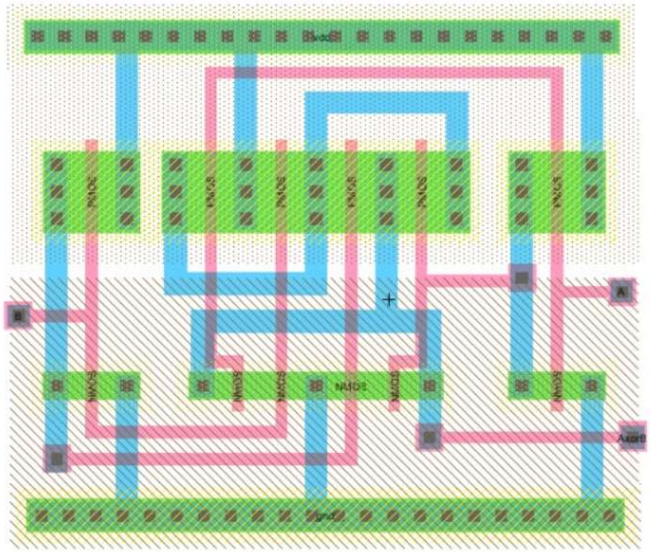

Figura 13: Layout para porta lógica XOR com lógica CMOS

Com o layout acima, realizamos a simulação do circuito no LTSpice. Conforme dito anteriormente, é necessário a inclusão da linha especificando o caminho para o seu arquivo com os modelos, como informado nas páginas que antecedem.

O código para simulação é apresentado a seguir:

```
vdd vdd 0 DC 5
Va a 0 pulse 5 0 0 1n 1n 2u 4u
Vb b 0 pulse 5 0 0 1n 1n 1u 2u
.tran 4u
```

Figura 14: Código para simulação da porta XOR no software LTSPice

Com o códico acima, foi realizada a simulação da porta lógica, e estes podem ser visualizados abaixo:

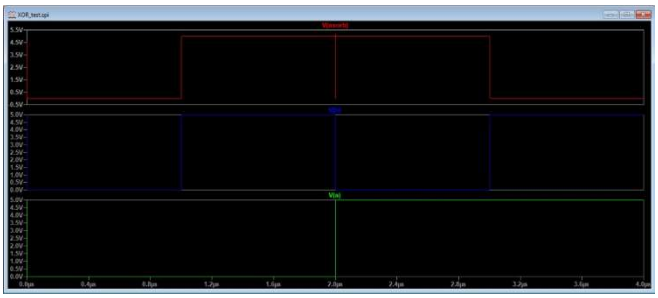

Figura 15: SImulação da porta lógica XOR no software LTSpice

Com base na imagem acima, concluimos que a porta lógica funciona de maneira correta, visto que, apresenta saída em nível lógico alto apenas quando somente uma das saídas está com nível lógico alto.

5. Lógica CMOS – Somador completo de 1 bit

Com as portas lógicas OR, AND e XOR, é possível implementar um somador completo de 1 bit. O procedimento é descrito a seguir.

5.1 Ícone

O ícone característico de um somador completo é mostrado abaixo:

Figura 16: Ícone de um somador completo de 1 bit

De acordo com o ícone acima, montamos o esquemático do somador completo de 1 bit, utilizando as portas lógicas que foram implementadas até agora, no decorrer deste trabalho.

5.2 Esquemático

O esquemático para este somador é apresentado a seguir.

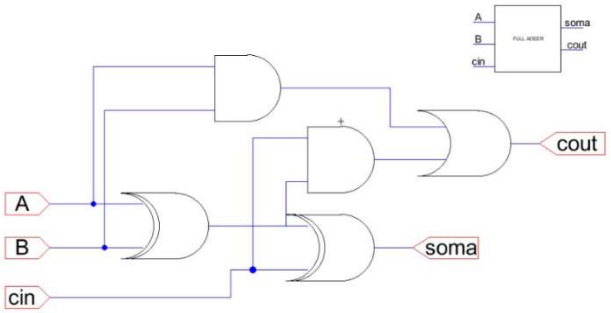

Figura 17: Esquemático para um somador completo de 1 bit

Com este esquemático, montamos o layout deste somador. O mesmo é mostrado no próximo item.

5.3 Layout

O layout do somador completo de 1 bit é mostrado a seguir:

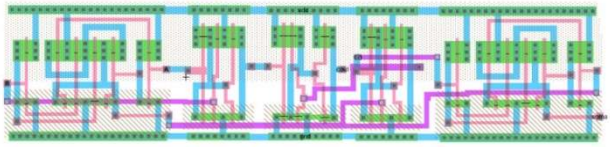

Figura 18: Layout para um somador completo de 1 bit

Com este layout, podemos realizar a simulação do somador. Esta simulação é mostrada nos próximos itens.

5.4 Simulações

O código para simulação do somador completo de 1 bit é mostrado na próxima imagem. Novamente, lembramos que é necessário incluir a última linha, com um include especificando o caminho para o arquivo com os modelos.

```
vdd vdd 0 DC 5
Va A 0 pulse 5 0 0 1n 1n 4u 8u
Vb B 0 pulse 5 0 0 1n 1n 2u 4u
Vcin cin 0 pulse 5 0 0 1n 1n 1u 2u
.tran 8u
```

Figura 19: Código para simulação do somador completo de 1 bit no software LTSpice

Com este código, podemos realizar a simulação do somador completo de 1 bit, e esta simulação é mostrada abaixo.

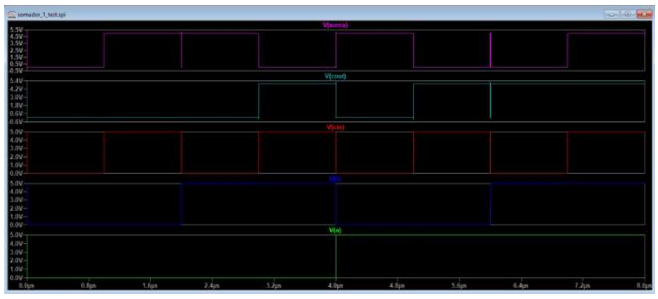

Figura 20: Resultado da simulação do somador completo de 1 bit.

Através da simulação acima, podemos verificar que o somador completo de 1 bit está funcionando da maneira correta.

6. LÓGICA CMOS – SOMADOR COMPLETO DE 8 BITS

Com o somador completo de 1 bit devidamente implementado e testado, podemos seguir agora para a implementação de um somador completo de 8 bits, uma vez que este ultimo é compost por oito somadores completos de 1 bit.

O procedimento é descrito a seguir.

6.1 Ícone

O ícone para um somador completo de 8 bits é mostrado a seguir:

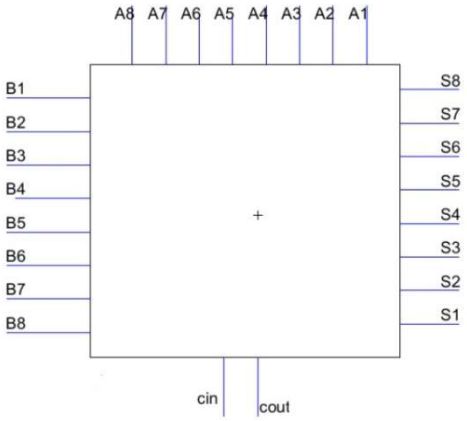

Figura 21: Ícone para um somador completo de 8 bits

Seguindo a descrição do ícone acima, implementamos o esquemático para um somador completo de 8 bits, que é, basicamente, a junção de 8 somadores completos de 1 bit.

Este esquemático é mostrado a seguir.

6.2 Esquemático

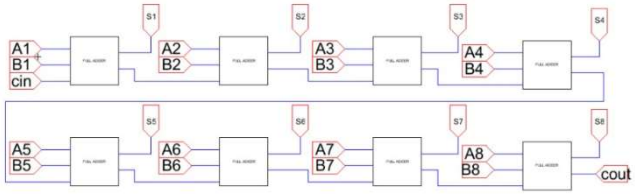

Figura 22: Esquemático para implementação de um somador completo de 8 bits

Utilizando o esquemático acima, realizamos o projeto do layout do somador completo de 8 bits, uma vez que este também é composto de 8 layouts do somador completo de 1 bit. Este layout é mostrado a seguir.

6.3 Layout

O layout para um somador completo de 8 bits é mostrado na próxima imagem:

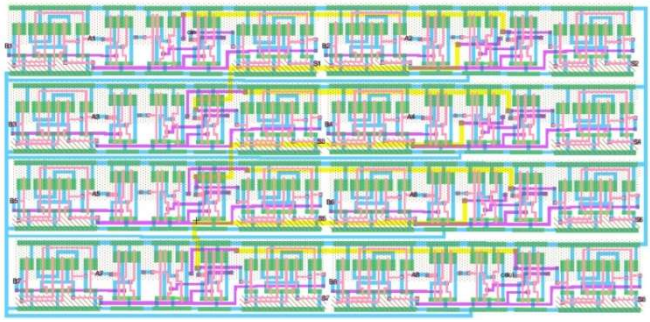

Figura 23: Layout para um somador completo de 8 bits

Com o layout devidamente implementado, seguimos para a simulação do somador. Este procedimento é mostrado no próximo item.

6.4 Simulações

Para a simulação do somador completo de 8 bits, utilizamos a seguinte configuração:

```
vdd vdd 0 DC 5
vnd gnd 0 DC 0
Va1 A1 0 DC 0
Va2 A2 0 DC 0
Va3 A3 0 DC 0
Va4 A4 0 DC 0
Va5 A5 0 DC 0
Va6 A6 0 DC 0
Va7 A7 0 DC 0
Va8 A8 0 DC 0
Vb1 B1 0 DC 5
Vb2 B2 0 DC 5
Vb3 B3 0 DC 5
Vb4 B4 0 DC 5
Vb5 B5 0 DC 5
Vb6 B6 0 DC 5
Vb7 B7 0 DC 5
Vb8 B8 0 DC 5
.tran 10u
```

Figura 24: Configuração utilizada para realizara simulação do somador completo de 8 bits

Com o código mostrado anteriormente, obtemos o resultado da simulação, que é mostrado abaixo:

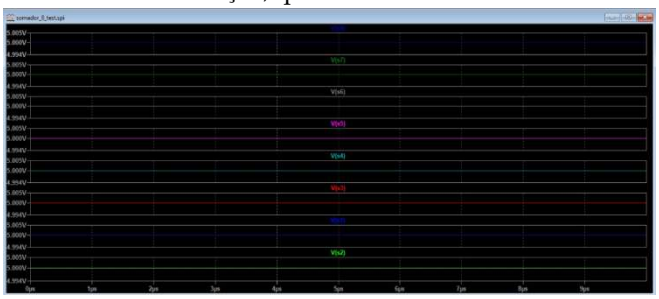

Figura 25: Resultados da simulação do somador completo de 8 bits

Com este resultado, percebemos o correto funcionamento do somador completo de 8 bits.

7. Procedimento de projeto

Para realizar este projeto, foi necessário a realização de alguns cálculos. Os mesmos são mostrado no anexo 01.

8. Conclusões

O somador é um dos componentes mais importantes de uma unidade lógica artimética (ALU), que é a unidade responsável por todo o processamento dos cálculos quando são solicitados ao processador.

Neste relatório, foram descritos todos os processos para realização de um somador completo de 8 bits, partindo da implementação do mesmo em linguagem VHDL até o layout de um circuito físico, fazendo uso do software electric.

Com a realização deste trabalho, aprendemos muito sobre o processo de fabricação e implementação deste circuito, possibilitando um maior entendimento sobre o processo de fabricação de um circuito tão pequeno quanto os transistores e as portas lógicas implementadas a partir deles.

Ao concluir o projeto dos dois somadores, tanto de 1 quanto de 8 bits, podemos pensar em algumas aplicações para o mesmo, como a unidade lógica aritmética, conforme já citado anteriormente, que é totalmente dependente de um circuito somador para realizar as operações necessárias.

9. Bibliografia

[1] SEDRA, S.; SMITH, K.. Microeletrônica. 4ª. Edição, Pearson Makron Books, São Paulo, Brasil, 2005.

[2] Allen, P.E. and Holberg, D.R., 1987. *CMOS analog circuit design*. Oxford university press.

[3] Weste, N.H. and Eshraghian, K., 1994. Principles of CMOS VLSI design: A systems perspective second edition. *Addision-Wesley Publishing, California, l994*.

[4] Baker, R.J., 2008. *CMOS: circuit design, layout, and simulation* (Vol. 1). John Wiley & Sons.

[5] CMOSedu.com. Electric VLSI Tutorials from CMOSedu.com. Disponível em: http://cmosedu.com/videos/electric/electric_videos.htm

[6] PEDRONI, V. Circuit Design and Simulation with VHDL, 2ed, MIT, 2010

10. ANEXO 01

$$V_{th} = \frac{V_{DD} - |V_{tp}| + V_{tn} * \beta}{1 + \beta}$$

$K_n = K_n(\frac{W}{L})_n$
$K_p = K_p(\frac{W}{L})_p$
$V_{th} = 2{,}5V$
$V_{DD} = 5V$
$V_{tp} = 0{,}669V$
$C_{ox} = 1{,}39 * 10^{-8}$
$V_{dp} = 458{,}43$
$V_{dn} = 212{,}09$

$$K_n = V_{dn} * C_{ox} = 458{,}43 * 1{,}39 * 10^{-8}$$
$$K_n = 6{,}37\mu A/V^2$$
$$K_n = V_{dp} * C_{ox} = 212{,}01 * 1{,}39 * 10^{-8}$$
$$K_p = 2{,}94\mu A/V^2$$

$$2{,}5 = \frac{5 - 0{,}92 + 0{,}669 * \beta}{1 + \beta}$$
$$\beta = 0{,}86$$

Escolhendo L como mínimo:

$$0{,}86 = \sqrt{\frac{K_n\left(\frac{W}{L}\right)_n}{K_p\left(\frac{W}{L}\right)_p}}$$

$$0{,}86^2 = \frac{6{,}37 * \left(\frac{W}{L}\right)_n}{2{,}94 * \left(\frac{W}{L}\right)_P}$$

Escolhendo 5 para W_n

$$W_p 0{,}74 = 2{,}16 * 10{,}8$$
$$W_p = 14{,}60$$