Universidade Estadual de Maringá

Ciência da Computação – Circuitos Digitais II

Discente: Thiago Rodrigo Bucalão

1. **Introdução:**

Definição: A *ULA* **(**Unidade Lógica da Aritmética) é a seção do computador que executa todas as operações lógicas e aritméticas. As funções aritméticas requeridas, como adição e subtração e todas as operações lógicas, tais como: AND, NAND, OR, NOR, XOR e XNOR.

O conceito da ULA foi proposto pelo matemático John Von Neumann em 1945 quando escreveu um relatório sobre os fundamentos para um novo computador chamado EDVAC, que usava o sistema binário para suas operações. O circuito apresentado a seguir é uma ULA de 1 bit, que contém as seguintes operações:

Lógicas: AND, NAND, OR, NOR, XOR, XNOR;

Aritméticas: Adição binária completa, Subtração binária completa.

Além da parte lógica e a aritmética da ULA temos ainda que criar um decodificador de 3 entradas e 8 saídas, para que seja possível a escolha de qual resultado queremos na saída, se de uma operação lógica ou aritmética.

2. **Componentes da ULA e Códigos**

Portanto para se criar a ULA foram criados os seguintes componentes:

- Decodificador 3x8;
- Portas lógicas;
- Somador;
- Subtrator;
- Seleção de saída.

E em seguida um pacote contendo todos esses componentes ao qual será utilizado na ULA de 1 bit.

2.1.**Decodificador 3x8**

O decodificador foi feito utilizando o comando “CASE WHEN”, este é um comando sequencial quando usados dentro de procedimento, funções e processos. Segundo (AMORE) este comando não permite que mais de uma condição verdadeira na relação de valores que a expressa pode assumir, e que todas as condições possuem a mesmo prioridade.

Sedo assim a figura 1 apresenta o código do decodificador utilizado para experimento:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DECODIFICADOR_3x8 IS
    PORT(SEL: BIT_VECTOR (2 DOWNTO 0);
         S : OUT BIT_VECTOR(7 DOWNTO 0));
END DECODIFICADOR_3x8;

ARCHITECTURE TEST_DECO_3x8 OF DECODIFICADOR_3x8 IS
    BEGIN
        PROCESS(SEL)
            BEGIN
                CASE SEL IS
                    WHEN "000" => S <= "00000001";
                    WHEN "001" => S <= "00000010";
                    WHEN "010" => S <= "00000100";
                    WHEN "011" => S <= "00001000";
                    WHEN "100" => S <= "00010000";
                    WHEN "101" => S <= "00100000";
                    WHEN "110" => S <= "01000000";
                    WHEN "111" => S <= "10000000";
                END CASE;
            END PROCESS;
    END TEST_DECO_3x8;
```

Figura 1: Código Decodificador 3x8 com CASE WHEN

Na entidade foram declarados “SEL” e “S” sendo as entradas e saídas respectivamente do Decodificador, portanto como temos 8 opções de saídas é necessário um seletor de 3 bits, sendo para cada fez que esses bits entram iremos ter resultados diferentes nas saídas, isso é claro se as entradas forem diferentes em cada estado.

Sabendo que S é vetor que armazena até 8 bit, quando a entrada de seleção é “000” S terá como saída “00000001” sendo a posição zero do vetor com o bit “1”, ou seja, é a descrição da tabela verdade deste componente porém, escrita em VHDL, podemos ver melhor através da forma de onda apresentada na figura 2.

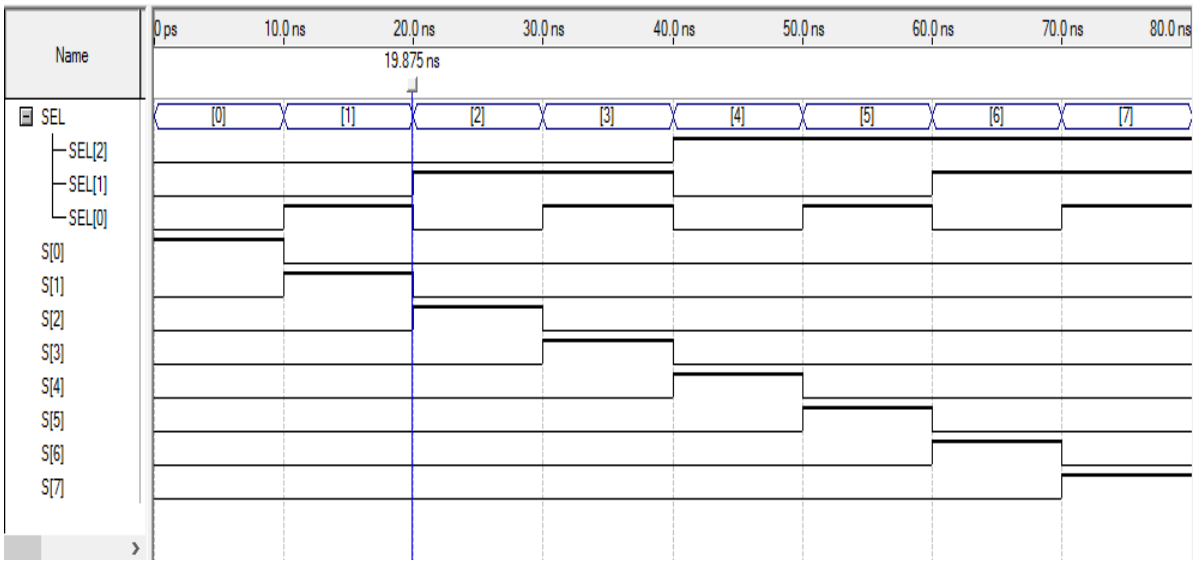

Figura 2: Forma de Onda Código Decodificador 3x8

Portanto percebemos pela forma de onda que quando entra “000” somente a posição zero do vetos “S” irá ter saída alta o restante todas são saídas em zero, confirmando o que esperávamos do código.

2.2.Portas Lógicas

As portas lógicas de uma ULA é onde é permitido fazer as seguintes operações lógicas, AND, NAND, OR, NOR, XOR, XNOR e para se fazer esse “componente” utilizamos o comando “IF THEN ELSE”, este comando é sequencial, quando utilizado dentre de processos a figura 3 nos apresentas o código para a criação desse componente.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY PORTAS_LOGICAS IS
    PORT(A_B_Cin : IN BIT_VECTOR (2 DOWNTO 0);
        Saida_deco : IN BIT_VECTOR (7 DOWNTO 0);
        S: OUT BIT);
END PORTAS_LOGICAS;
ARCHITECTURE COMPORTAMENTAL OF PORTAS_LOGICAS IS
BEGIN
        PROCESS(A_B_Cin, saida_deco)
        BEGIN
            IF (saida_deco(0)= '1') THEN
                S <= A_B_Cin(2) AND A_B_Cin(1); -- s0 estará recebendo A e B
            ELSIF (saida_deco(1)= '1') THEN
                S <= A_B_Cin(2) NAND A_B_Cin(1);
            ELSIF (saida_deco(2)= '1') THEN
                S <= A_B_Cin(2) OR A_B_Cin(1);
            ELSIF (saida_deco(3)= '1') THEN
                S <= A_B_Cin(2) NOR A_B_Cin(1);
            ELSIF (saida_deco(4)= '1') THEN
                S <= A_B_Cin(2) XOR A_B_Cin(1);
            ELSIF (saida_deco(5)= '1') THEN
                S <= A_B_Cin(2) XNOR A_B_Cin(1);
            ELSE
                S <= '0';
            END IF;
        END PROCESS;
END COMPORTAMENTAL;
```

Figura 3: Código das Portas Lógicas com IF THEN ELSE

Esse componente utiliza as entras A e B que estão armazenadas no vetor “A_B_Cin” e desse vetor os elementos de posição 2 e 1 apenas são utilizados nas operações. “As condições são avaliadas na ordem da apresentação no código” (AMORE, 2005). Dessa forma somente uma das portas lógicas será ativada por vez, sendo a condição que irá determinar isso é a entrada “SAIDA_DECO”, caso nenhuma das condições seja satisfeitas então a saída do componente é zero conforme será mostrado na figura 4.

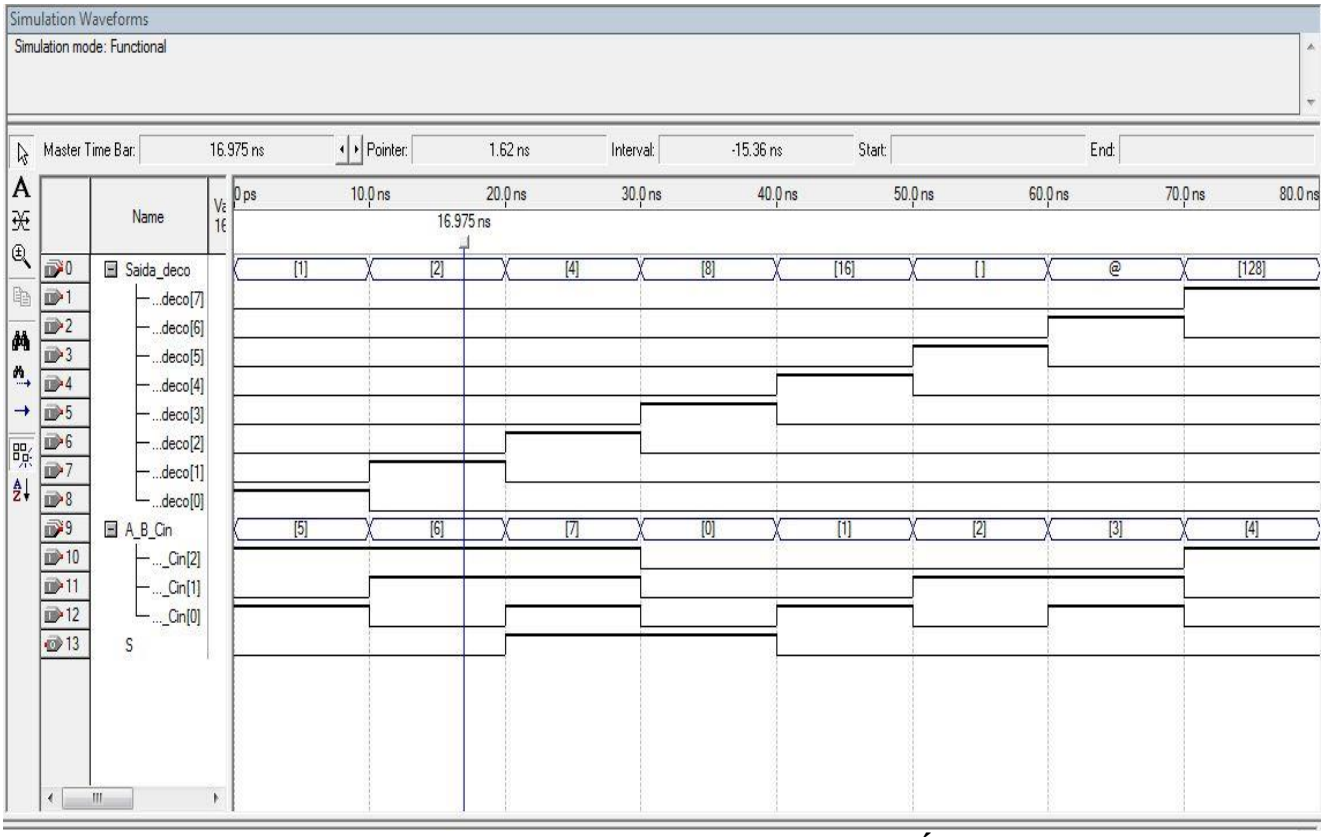

Figura 4: Formas de Ondas Código PORTAS LÓGICAS

2.3. Somador

"O Somador Completo é um circuito para efetuar a soma completa de uma coluna, considerando o transporte de entrada" (CAPUANO, p.212). Dessa forma, ao criarmos o somador na linguagem VHDL utilizamos o comando "WITH SELECT WHEN", ao qual segundo (AMORE) esse comando transfere uma valor a um sinal de destino segundo a relação de opções, onde todas as condições são verificadas, e não podem ocorrer condições iguais. O código do somador pode ser visto na figura 5.

```
ENTITY SOMADOR IS
    PORT (A_B_Cin: IN BIT_VECTOR (2 downTO 0);   -- A, B e Cin
          SAIDA_DECO : IN BIT_VECTOR(7 DOWNTO 0);-- Saidas do decodificador
          S_SOM,Cout_SOM : OUT BIT; -- Saídas do somador
          S,Cout: BUFFER BIT); -- Auxiliares
END SOMADOR;
ARCHITECTURE TEST_SOMADOR OF SOMADOR IS
BEGIN
        WITH A_B_Cin SELECT
            S <= '0' WHEN "000", -- Da mesma forma que o subtrator, foi utilizado duas
                 '1' WHEN "001", -- vezes o with select when para selecionar a saída ao final
                 '1' WHEN "010",
                 '0' WHEN "011",
                 '1' WHEN "100",
                 '0' WHEN "101",
                 '0' WHEN "110",
                 '1' WHEN "111";
        WITH A_B_Cin SELECT
            Cout <= '0' WHEN "000",
                    '0' WHEN "001",
                    '0' WHEN "010",
                    '1' WHEN "011",
                    '0' WHEN "100",
                    '1' WHEN "101",
                    '1' WHEN "110",
                    '1' WHEN "111";
        S_SOM <= (S AND SAIDA_DECO(6)); -- Análogamente ao subtrator, foi feito uma operação and
        Cout_SOM <= (Cout AND SAIDA_DECO(6));-- para selecionar a saída
END TEST_SOMADOR;
```

Figura 5: Somador Completo com WITH SELECT WHEN

Temos nesse código portanto a entras A, B e Cin onde estão todas “agrupas” em um vetor e declarado como “A_B_Cin”, além disso as entras do decodificador 3x8 onde apenas uma de suas saídas é utilizada no somador, isso acontece pois, caso queira realizar uma opção aritmética de soma é necessário que entre como seleção no decodificador “110” para que a saída 6 do mesmo possa liberar bit 1, e assim ativar o somador.

Caso entre qualquer outro valor no seletor do decodificador não irá ativar o somador e assim as saídas “S_SOM” e “Cout_SOM” irá ter valor zero.

Podemos comprovar isso através da forma de ondas do componente demostrado na figura 6.

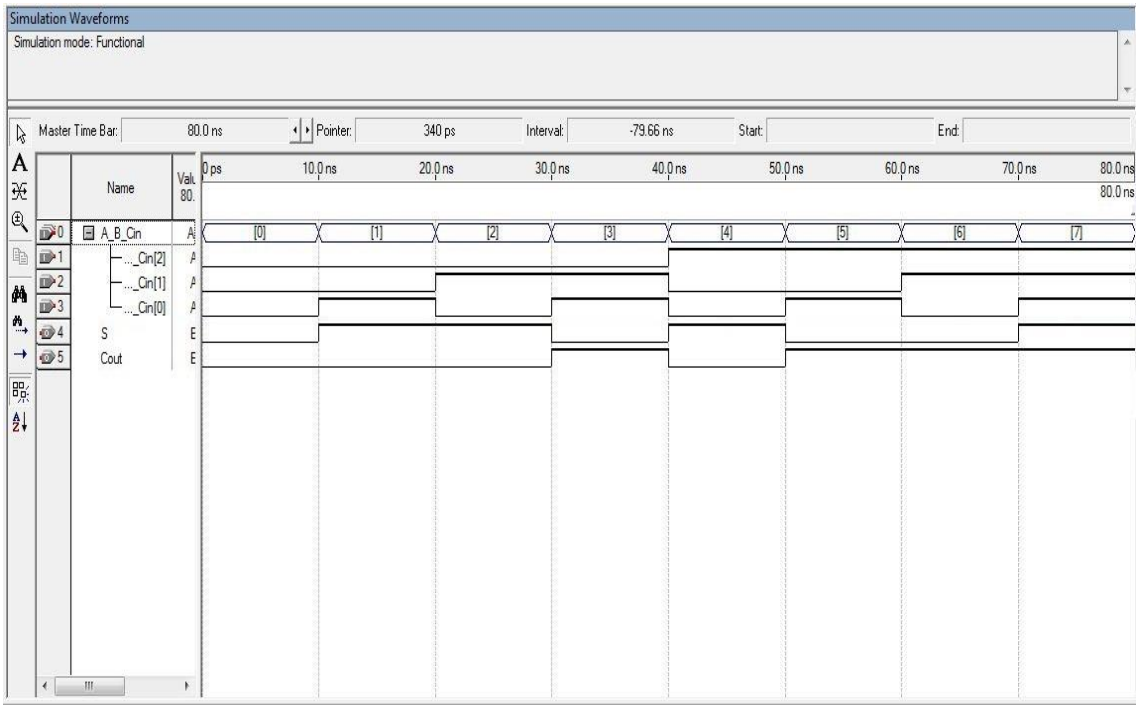

Figura 6: Forma de Onda do código Somador

2.4.Subtrator

"O Subtrator Completo é um circuito que efetua a subtração completa de uma coluna, ou seja, considera o transporte de entrada proveniente da coluna anterior" (CAPUANO, p.218).

Para criarmos esse componente da ULA utilizamos o comando "WHEN ELSE", ao qual é um comando concorrente e é bastante útil para expressar funções lógicas em forma de tabela verdade.

"Nesta construção, uma lista de opções é apresentada estabelecendo qual o valor de uma expressão deve ser transferido a um sinal de destino. A primeira condição que retorna o valor verdadeiro define o valor que é transferido para o sinal de destino." (AMORE, p.31). Dessa forma o nosso Subtrator é descrito conforme o código da figura 7.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SUBTRATOR IS
    PORT(EN_SUB: IN BIT_VECTOR (2 DOWNTO 0); -- A, B  e Cin
         SAIDA_DECO : IN BIT_VECTOR(7 DOWNTO 0); -- Saída do decodificador
         S,Cout: BUFFER BIT; -- Auxiliares
         S_SUB,Cout_SUB : OUT BIT); -- Saídas do Subtrator
END SUBTRATOR;
ARCHITECTURE TEST_SUBTRATOR OF SUBTRATOR IS
    BEGIN
        S <=      '0' WHEN EN_SUB = "000" ELSE   -- Foi utilizado duas vezes
                  '1' WHEN EN_SUB = "001" ELSE   -- o when else para que pudéssemos
                  '1' WHEN EN_SUB = "010" ELSE   -- selecionar a saída ao final
                  '0' WHEN EN_SUB = "011" ELSE
                  '1' WHEN EN_SUB = "100" ELSE
                  '0' WHEN EN_SUB = "101" ELSE
                  '0' WHEN EN_SUB = "110" ELSE
                  '1' WHEN EN_SUB = "111";
        Cout <=   '0' WHEN EN_SUB = "000" ELSE
                  '1' WHEN EN_SUB = "001" ELSE
```

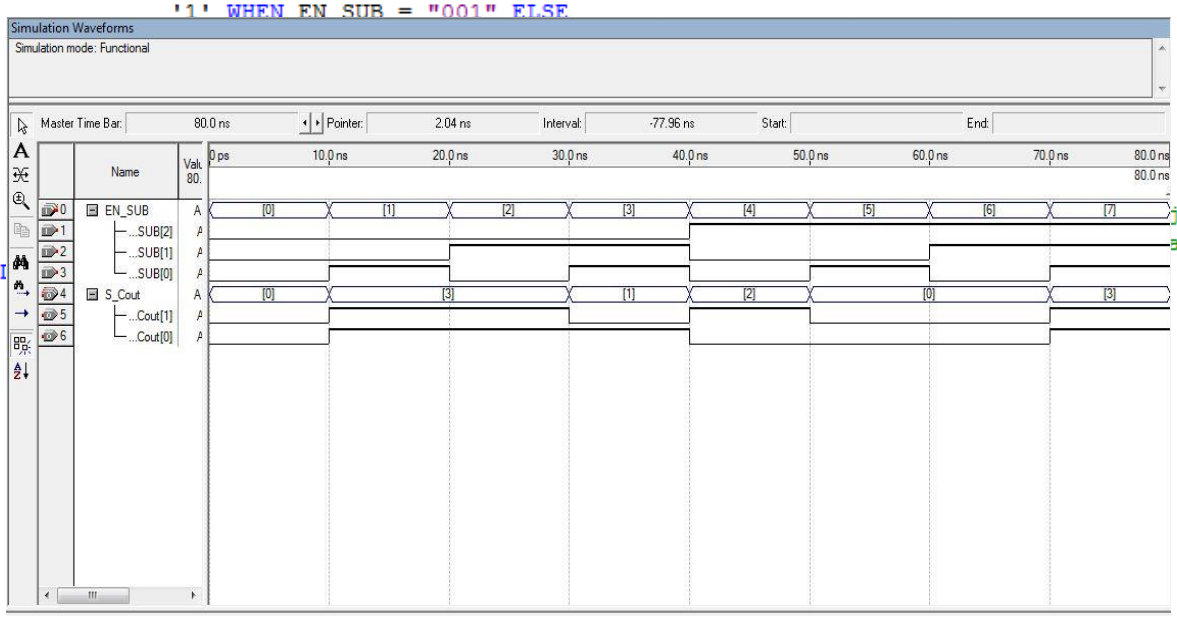

Figura 7: Subtrator com comando WHEN ELSE

De forma equivalente que ocorre no somador, o componente do Subtrator só será ativado quando a saída do decodificador for conforme a condição descrita no código, ou seja, somente quando a posição 7 da saída do decodificador for igual a 1 irá ativar o Subtrator, sendo assim só quando a entrada do seletor for “111”, qualquer outra entrada de seleção não irá ativar e com isso as saídas “S_SUB” e “Cout_SUB” serão zero. Para comprovar isso temos a forma de onda do Subtrator na figura 8.

Figura 8: Forma de Onda do código Subtrator

2.5.Seleção de Saída

Para que obetermos os resultados desejáveis da ULA foi criado um componente ao qual irá receber as saídas da parte lógica, outra da aritmetéca, tendo assim cinco entradas esse pois na parte aritimética temos as saídas “S_SOM, S_SUB” do somador e Subtrator respectivamente e os “Cout” de ambos. O código do componente é descrito na figura 9.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SELECAO_SAIDA IS
    PORT( S_SOM,S_SUB,Cout_SOM,Cout_SUB,S_LOGICA : IN BIT; -- Saídas dos componentes portas_logicas, somador e subtrator
          S,Cout : OUT BIT); -- Saídas finais da ULA
END SELECAO_SAIDA;
ARCHITECTURE COMPORTAMENTAL OF SELECAO_SAIDA IS
BEGIN
    S <= (S_SOM OR S_SUB OR S_LOGICA); -- como somente uma das saídas sairá com bit '1' devido
    Cout <= (Cout_SOM OR Cout_SUB); -- ao decodificador, pode--se usar uma or, pois S só receberá
END COMPORTAMENTAL; -- '1' caso alguma saída tenha sido ativada e liberado '1', da mesma forma com o Cout
```

Figura 9: Código do componente Selecao_saida

Pelo fato de ter sido criado apenas como um componente e não um projeto então não é possível obter a forma de onda mais sabermos que esse componente se comporta como uma porta OR de cinco entradas e portanto, somente no caso das entradas “S_SOM, S_SUB, S_LOGICA” forem de valor lógico 0 terá saída zero em “S”, de forma equivalente ocorre pra saída Cout.

3. Pacote dos Componentes

Para

4. ULA 1 bit

Portanto após ter feito todos os componentes necessários para o funcionamento da ULA, é necessário interligar todos eles, para isso utilizamos uma arquitetura estrutural ao qual com a utilização de Sinais que por sua vez são “fios” levando informação da saída de um componente para entrada de outro componente, é assim possível fazer essa ligação, a figura 10.

```
LIBRARY WORKTRABALHO;
USE WORKTRABALHO.blocos_package.all;
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ULA IS
    PORT(A_B_Cin, SEL_ULA : IN BIT_VECTOR (2 DOWNTO 0); -- entradas A, B e Cin e Entradas de Seleção
         S, Cout : OUT BIT); --Saídas da ULA
END ULA;
ARCHITECTURE TEST_ULA OF ULA IS
SIGNAL T0 : BIT_VECTOR(7 DOWNTO 0);
SIGNAL T1,T2,T3,T4,T5 : BIT;
BEGIN
    P1: DECODIFICADOR_3X8 PORT MAP (SEL_ULA,T0); -- t0 é a saída da ULA que servirá de entrada aos outros componentes
    P2: SOMADOR PORT MAP (A_B_Cin,T0,T1,T2); -- t1 e t2 são as saídas S e Cout do somador
    P3: SUBTRATOR PORT MAP (A_B_Cin,T0,T3,T4); -- t3 e t4 são as saídas S e Cout do subtrator
    P4: PORTAS_LOGICAS PORT MAP(A_B_Cin,T0,T5); -- t5 é a saída S das portas lógicas
    P5: SELECAO_SAIDA PORT MAP (T1,T3,T2,T4,T5,S,Cout); -- de t1 a t5 estão as saídas a serem selecionadas na OR
END TEST_ULA;
```

Figura 10: Código ULA

Dessa forma utilizando como sinal “T0” que ao qual é responsável por levar os bits das saídas de Decodificador e ele é utilizado tanto como entrada para as portas lógicas como para o somador e o Subtrator, por fim as saídas desses componentes são transmitidas para entrada de “SELECAO_SAIDA” através dos sinais “T1,T2,T3,T4,T5”. Para comprovar o funcionamento da ULA a figura 11 nos mostra a forma de onda.

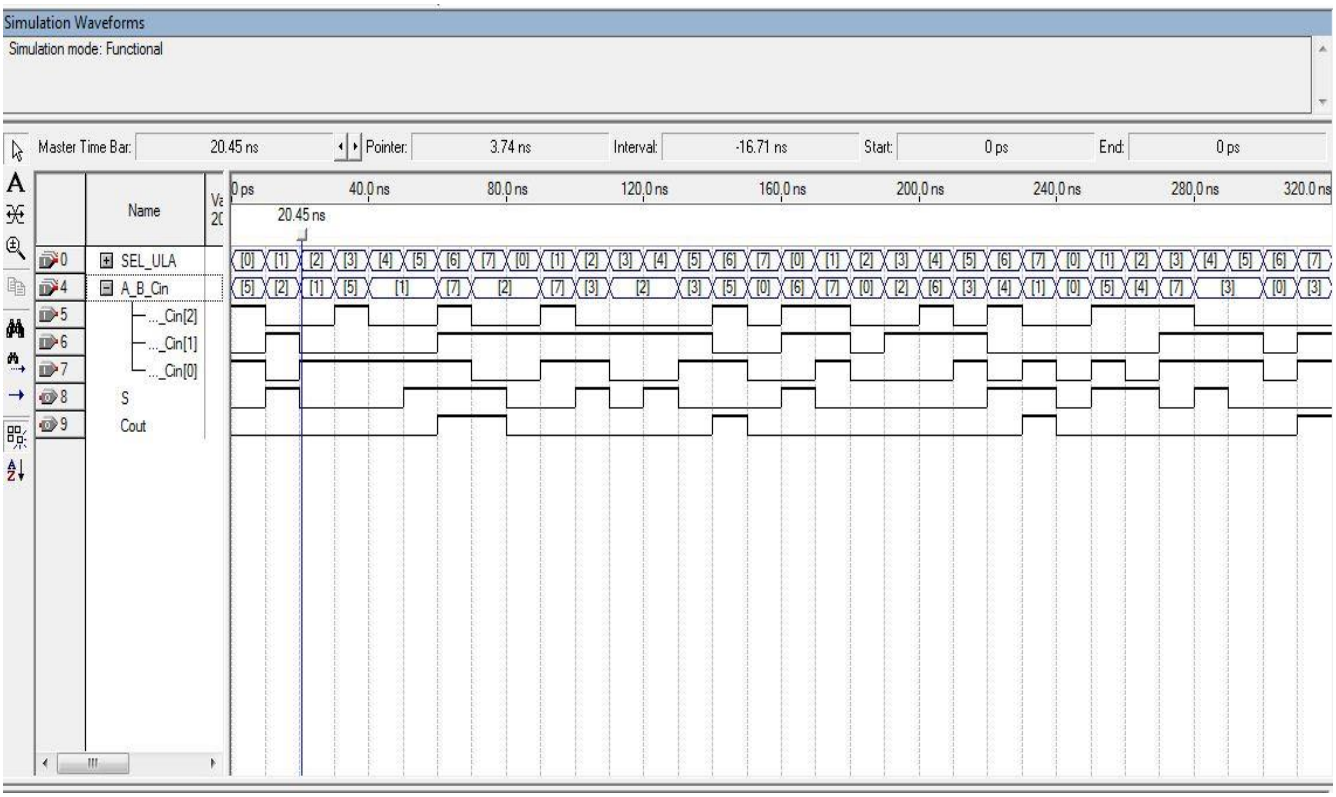

Figura 11: Forma de Onda da ULA

Dessa forma podemos observar que no primeiro instante a porta lógica AND é ativada e temos saída “0” em seguida temos a saída “S” e “Cout” recebendo os valores das operações NAND(1), OR(2), NOR(3), XOR(4), XNOR(5), SOMADOR(6) e por último SUBTRATOR(7).