Universidade de Brasília – UNB

Curso: Engenharia de Redes de Comunicação
Disciplina: Laboratório de Sistemas Digitais
Turma: 08

Relatório da Disciplina Laboratório de Sistemas Digitais
Tema: Experimento 04 – Projeto Modular

Aluno: Pedro Henrique Dias Avelar
Matrícula: 241037112
Professor: Eduardo Paiva

Lista de Referências

Introdução

O presente experimento tem os seguintes objetivos:

- Utilizar multiplexadores e decodificadores para implementar circuitos lógicos combinacionais.
- Estudar técnicas de projeto modular em VHDL, desenvolvendo sistemas grandes a partir de circuitos menores interligados entre si

Experimento 01

Escreva em VHDL e simule no ModelSim uma entidade com 3 bits de entrada (A,B,C) e 2 bits de saída (X,Y) que implemente as funções lógicas

$$X = \bar{A}BC + A\bar{B}\bar{C} + AB$$

$$Y = \bar{A}\bar{B} + \bar{A}B\bar{C} + ABC$$

Sua arquitetura poderá usar somente dois multiplexadores 4x1 e uma porta inversora. Os multiplexadores e a porta devem ser incluídos no código da entidade principal como "componente". A arquitetura da entidade principal deve apenas fazer a conexão entre os componentes, sem usar operações lógicas adicionais.

Podemos modelar uma função lógica que envolva uma soma de produtos com 3 variáveis com o uso de um multiplexador 4x1 por meio da técnica de variável induzida. Das duas equações acima, podemos observar que os bits A e B estão presentes em todos os termos. Dessa forma, podemos então montar a tabela verdade para as funções X e Y induzindo o valor de C:

	A	B	C	$\bar{A}BC$	$A\bar{B}\bar{C}$	AB	X	$X_{C\ induzido}$
$\bar{A}\bar{B}$	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	0	
$\bar{A}B$	0	1	0	0	0	0	0	C
	0	1	1	1	0	0	1	
$A\bar{B}$	1	0	0	0	1	0	1	$\bar{C}$
	1	0	1	0	0	0	0	
AB	1	1	0	0	0	1	1	1
	1	1	1	0	0	1	1	

Tabela 1:Tabela Verdade para a função X

	A	B	C	$\bar{A}\bar{B}$	$\bar{A}B\bar{C}$	ABC	Y	$X_{C\ induzido}$
$\bar{A}\bar{B}$	0	0	0	1	0	0	1	1
	0	0	1	1	0	0	1	
$\bar{A}B$	0	1	0	0	1	0	1	$\bar{C}$
	0	1	1	0	0	0	0	
$A\bar{B}$	1	0	0	0	0	0	0	0
	1	0	1	0	0	0	0	
AB	1	1	0	0	0	0	0	C
	1	1	1	0	0	1	1	

Tabela 2:Tabela Verdade para a função Y

Assim, podemos então utilizar um multiplexador para modelar as funções acima usando A e B como como os seletores do multiplexador. Representando no Logisim, o circuito teria a seguinte estrutura:

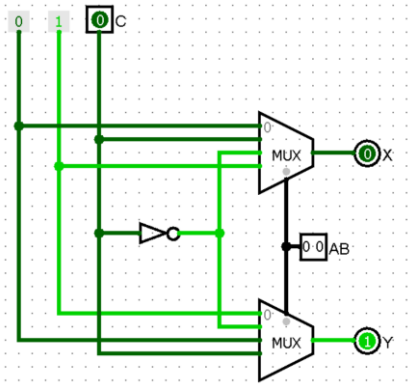

Figura 1:Representação gráfica do circuito que implementa as funções lógicas X e Y

O circuito foi então modelado no ModelSim usando o seguinte código:

```
-- Experimento 04 - Questão 01
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 27/11/2024

-- Entrada - 3 bits (A,B,C)
-- Saída   - 2 bits (X,Y)

-- X = !ABC + A!B!C + AB
-- Y = !A!B + !AB!C + ABC

-- usar APENAS 2 multiplexadores 4x1 e uma porta inversora

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY CIRCUITO_E04Q01 IS
        PORT(A,B,C: IN STD_LOGIC;
                   X,Y:  OUT STD_LOGIC);
END CIRCUITO_E04Q01;

ARCHITECTURE ARC_CIRCUITO_E04Q01 OF CIRCUITO_E04Q01 IS
--Multiplexador 4x1 do Experimento 02 Questão 02
COMPONENT MUX4X1 IS
         PORT (D: IN STD_LOGIC_VECTOR (3 DOWNTO 0);              --ENTRADA
               S: IN STD_LOGIC_VECTOR (1 DOWNTO 0);              --SELEÇÃO
           Y_MUX: OUT STD_LOGIC);
END COMPONENT;

COMPONENT PORTA_NOT IS
         PORT(ENTRADA: IN STD_LOGIC;
                  SAIDA:   OUT STD_LOGIC);
END COMPONENT;

SIGNAL BARRAMENTOAB: STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL BARRAMENTOC_X:  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL BARRAMENTOC_Y:  STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL NOT_C: STD_LOGIC;
BEGIN
UX: MUX4X1 PORT MAP(BARRAMENTOC_X,BARRAMENTOAB,X);
UY: MUX4X1 PORT MAP(BARRAMENTOC_Y,BARRAMENTOAB,Y);
NOT1: PORTA_NOT PORT MAP (C, NOT_C);
BARRAMENTOAB(0) <= B;
BARRAMENTOAB(1) <= A;
BARRAMENTOC_X(0) <= '0';
BARRAMENTOC_X(1) <= C;
BARRAMENTOC_X(2) <= NOT_C;
BARRAMENTOC_X(3) <= '1';
BARRAMENTOC_Y(0) <= '1';
BARRAMENTOC_Y(1) <= NOT_C;
BARRAMENTOC_Y(2) <= '0';
BARRAMENTOC_Y(3) <= C;
END ARC_CIRCUITO_E04Q01;
```

Código 1: Modelagem das funções lógicas X e Y

Os componentes MUX4X1 e PORTA_NOT foram modelados com os códigos abaixo:

```
-- Experimento 02 - Questão 02
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 02/11/2024

-- Função lógica do multiplexador 4 para 1:
-- Y = D0!S1!S0 + D1!S1S0 + D2S1!S0 + D3S1S0

-- ALTERADO O NOME DA ENTITY E DA SAIDA Y

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY MUX4X1 IS
        PORT (D: IN STD_LOGIC_VECTOR (3 DOWNTO 0);          --ENTRADA
              S: IN STD_LOGIC_VECTOR (1 DOWNTO 0);          --SELEÇÃO
          Y_MUX: OUT STD_LOGIC);                            --SAÍDA
END MUX4X1;

ARCHITECTURE ARC_MUX4X1 OF MUX4X1 IS
BEGIN
        Y_MUX <= (D(0) AND NOT(S(1)) AND NOT(S(0))) OR      --D0!S1!S0
             (D(1) AND NOT(S(1)) AND     S(0))  OR         --D1!S1S0
             (D(2) AND     S(1)  AND NOT(S(0))) OR         --D2S1!S0
             (D(3) AND     S(1)  AND     S(0));            --D3S1S0
END ARC_MUX4X1;
```

Código 2:Código do Multiplexador 4 para 1

```
-- Experimento 04 - Porta NOT
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 04/12/2024

-- PORTA NOT
-- ENTRADA X
-- SAIDA !X


LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY PORTA_NOT IS
        PORT(ENTRADA: IN STD_LOGIC;
             SAIDA:   OUT STD_LOGIC);
END PORTA_NOT;

ARCHITECTURE ARC_PORTA_NOT OF PORTA_NOT IS
BEGIN
        SAIDA <= NOT ENTRADA;
END ARC_PORTA_NOT;
```

Código 3:Código da porta NOT

A componentização envolve a ideia da “caixa preta”; isto é, não nos interessa o funcionamento interno de um determinado componente, e sim que suas entradas e saídas realizem um determinado comportamento esperado. Sua principal vantagem consiste na reutilização de código. No caso do multiplexador por exemplo, a partir do código 3, foi possível criar dois componentes MUX4X1 no código 1.

Observando o código 1, temos os sinais BARRAMENTOAB, BARRAMENTOC_X, BARRAMENTOC_Y e NOT_C. O BARRAMENTOAB consiste na ligação das entradas A e B aos seletores dos dois multiplexadores. Os barramentos BARRAMENTOC_X e BARRAMENTOC_Y consistem nas

entradas dos multiplexadores que irão representar, respectivamente, as funções lógicas X e Y. O sinal NOT_C consiste na ligação entre a entrada C e uma porta NOT.

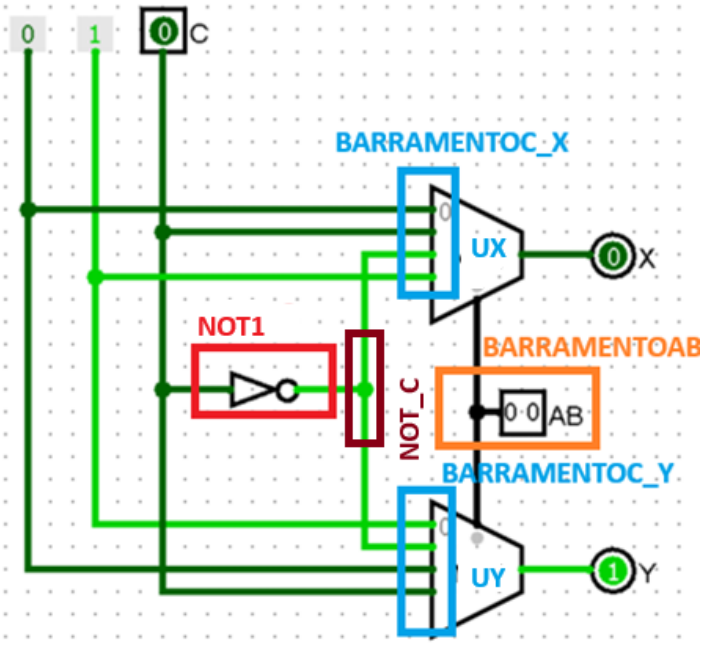

Figura 2:Representação gráfica dos sinais e componentes do código que implementa as funções lógicas X e Y

Para fazer a simulação no ModelSim, foi utilizado um clock de 100ps para a entrada A, 50ps para a entrada B e 25 os para a entrada C, em um período de 100ps. A simulação teve o seguinte resultado:

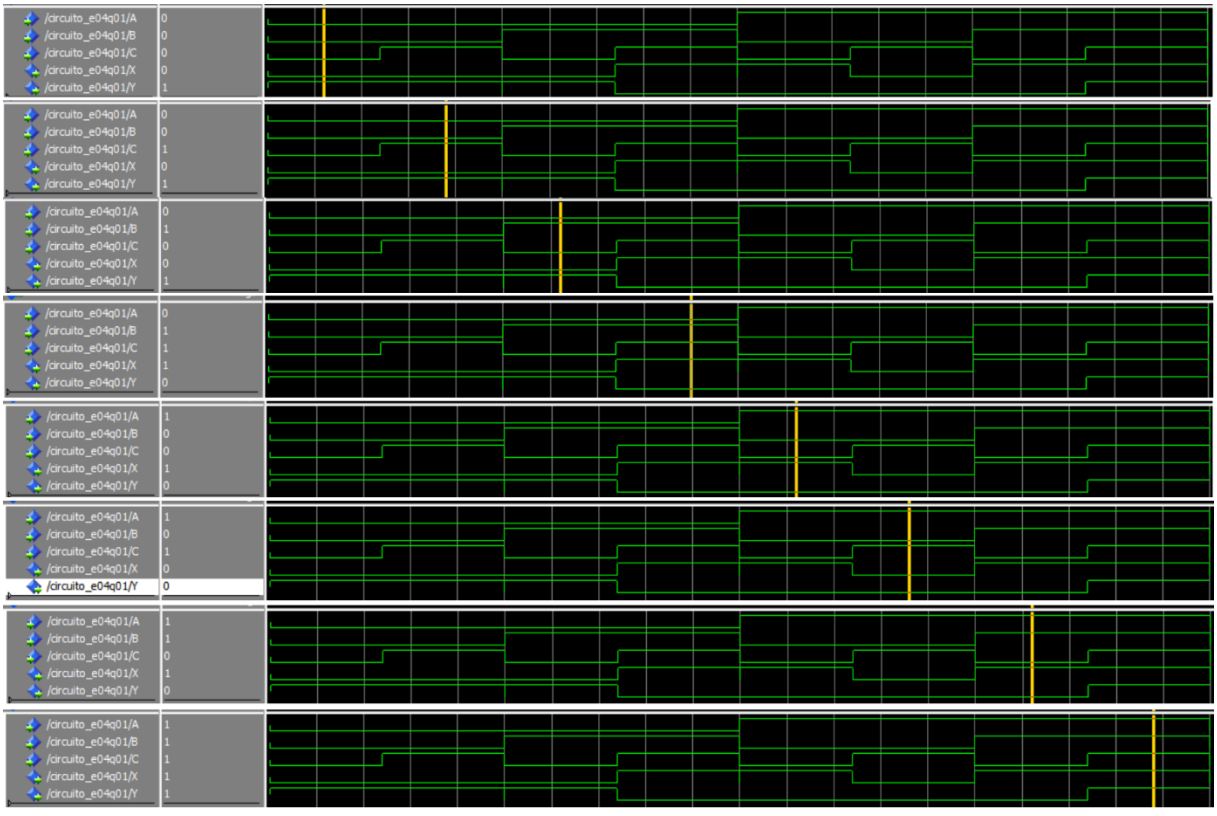

Figura 3:Resultado da Simulação para o Experimento 01

A partir do resultado da simulação podemos então montar a tabela verdade:

A	B	C	X	$X_{C\ induzido}$	Y	$Y_{C\ induzido}$
0	0	0	0	0	1	1
0	0	1	0		1	
0	1	0	0	C	1	$\bar{C}$
0	1	1	1		0	
1	0	0	1	$\bar{C}$	0	0
1	0	1	0		0	
1	1	0	1	1	0	C
1	1	1	1		1	

Tabela 3: Resultado da simulação para o experimento 01

Podemos ver que os resultados para X e Y da tabela 3 são, por indução perfeita, iguais aos respectivos resultados de X na tabela 1 e Y na tabela 2 e, com isso, podemos concluir que o circuito representado pelo código 1 representa com sucesso as funções lógicas X e Y.

Experimento 02

Escreva em VHDL e simule no ModelSim uma entidade com 7 bits de entrada (A,B,C,D,E,F,G) e um bit de saída (Z) que implemente a função lógica

$$Z = FG + ABCD\bar{E}\bar{F}G + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}G + A\bar{B}CEF\bar{G} + \bar{A}BCD\bar{E}F\bar{G} + ABCDE\bar{F}\bar{G} + A\bar{B}\bar{C}DE\bar{F}\bar{G}$$

Sua arquitetura deve usar somente um decodificador de 4 para 16, um multiplexador de 8 para 1 e quantas portas OU forem necessárias. Mais uma vez, os elementos permitidos devem ser incluídos no código da entidade principal como componentes e a arquitetura da entidade principal deve fazer apenas a conexão entre elas. Dica: use as variáveis E,F,G como entradas de seleção do multiplexador.

Inicialmente precisamos associar as entradas do multiplexador aos respectivos mintermos obtidos pelo decodificador. Seguindo a sugestão do enunciado, as variáveis E,F e G foram utilizadas no seletor do multiplexador; então usaremos as variáveis A,B,C e D no decodificador.

A	B	C	D	# saída do decodificador	Mintermo
0	0	0	0	d_0	$\bar{A}\bar{B}\bar{C}\bar{D}$
0	0	0	1	d_1	$\bar{A}\bar{B}\bar{C}D$
0	0	1	0	d_2	$\bar{A}\bar{B}C\bar{D}$
0	0	1	1	d_3	$\bar{A}\bar{B}CD$
0	1	0	0	d_4	$\bar{A}B\bar{C}\bar{D}$
0	1	0	1	d_5	$\bar{A}B\bar{C}D$
0	1	1	0	d_6	$\bar{A}BC\bar{D}$
0	1	1	1	d_7	$\bar{A}BCD$
1	0	0	0	d_8	$A\bar{B}\bar{C}\bar{D}$
1	0	0	1	d_9	$A\bar{B}\bar{C}D$
1	0	1	0	d_{10}	$A\bar{B}C\bar{D}$
1	0	1	1	d_{11}	$A\bar{B}CD$
1	1	0	0	d_{12}	$AB\bar{C}\bar{D}$
1	1	0	1	d_{13}	$AB\bar{C}D$
1	1	1	0	d_{14}	$ABC\bar{D}$
1	1	1	1	d_{15}	$ABCD$

Tabela 4:Lista de mintermos produzidos pelo decodificador 4x16

Grifando a equação para visualizar melhor, temos que:

$$Z = FG + ABCD\bar{E}\bar{F}\bar{G} + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}\bar{G} + A\bar{B}C\bar{E}\bar{F}\bar{G} + \bar{A}BCD\bar{E}\bar{F}\bar{G} + ABCD\bar{E}\bar{F}\bar{G} + A\bar{B}\bar{C}D\bar{E}\bar{F}\bar{G}$$

E usando a propriedade distributiva podemos simplificar a equação:

$$Z = FG + \bar{E}\bar{F}\bar{G}(ABCD + \bar{A}\bar{B}\bar{C}\bar{D}) + A\bar{B}C\bar{E}\bar{F}\bar{G} + \bar{A}BCD\bar{E}\bar{F}\bar{G} + \bar{E}\bar{F}\bar{G}(ABCD + A\bar{B}\bar{C}D)$$

E com isso podemos então fazer as seguintes relações abaixo:

E	F	G	# entrada	Entrada	Mintermos do Decoder associados
0	0	0	m_0	0	-
0	0	1	m_1	$ABCD + \bar{A}\bar{B}\bar{C}\bar{D}$	$d_{15} + d_0$
0	1	0	m_2	$\bar{A}BCD$	d_7
0	1	1	m_3	1	-
1	0	0	m_4	$ABCD + A\bar{B}\bar{C}D$	$d_{15} + d_9$
1	0	1	m_5	0	-
1	1	0	m_6	$A\bar{B}C$	$d_{10} + d_{11}$
1	1	1	m_7	1	-

Tabela 5: Relação entre as portas do Multiplexador e as saídas do Decodificador

Os termos m_0 e m_5 do multiplexador não estão associados a um termo da função lógica, por isso sempre terão valor zero.

Os termos m_3 e m_7 estão associados ao termo FG. Podemos representar o termo como

$$1 \cdot FG,$$

Logo os termos m_3 e m_7 sempre serão 1.

Assim, a partir das associações da tabela 5, podemos então modelar o circuito:

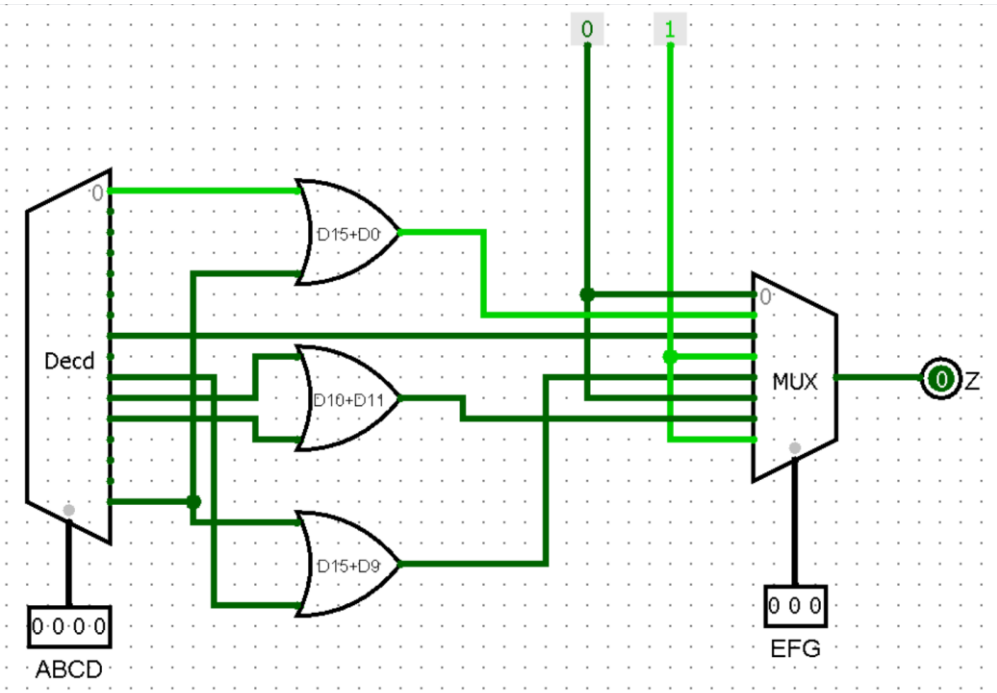

Figura 4:Representação gráfica do circuito que implementa a função lógica Z

O circuito foi então modelado no ModelSim com o seguinte código:

```
-- Experimento 04 - Questão 02
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 02/12/2024

-- Entrada - 7 bits (A,B,C,D,E,F,G)
-- Saída   - 1 bit  (Z)

-- Função - Z = FG + ABCD!E!FG + !A!B!C!D!E!FG + A!BCEF!G + ABCDE!F!G + A!B!CDE!F!G

-- Usar apenas UM decoder 4x16, UM mux 8x1 e quantas OU forem necessárias

-- 1a etapa - montar o mux 8x1 usando as entradas E,F e G como signal do Mux

-- S(2)=E;S(1)=F;S(0)=G;

-- D(7)=1 D(6)=A!BC  D(5)=0             D(4)=ABCD+A!B!CD
-- D(3)=1 D(2)=!ABCD D(1)=ABCD+!A!B!C!D D(0)=0
--
--
--

-- 2a etapa - associar saidas do decoder com o mux

-- A(3)=A   A(2)=B  A(1)=C  A(0)=D

-- Y_DEC(0 )=!A!B!C!D     Y_DEC(1 )=!A!B!CD    Y_DEC(2 )=!A!BC!D    Y_DEC(3 )=!A!BCD
-- Y_DEC(4 )=!AB!C!D      Y_DEC(5 )=!AB!CD     Y_DEC(6 )=!ABC!D     Y_DEC(7 )=!ABCD
-- Y_DEC(8 )=A!B!C!D      Y_DEC(9 )=A!B!CD     Y_DEC(10)=A!BC!D     Y_DEC(11)=A!BCD
-- Y_DEC(12)=AB!C!D       Y_DEC(13)=AB!CD      Y_DEC(14)=ABC!D      Y_DEC(15)=ABCD

-- D(7)=1  D(6)=A(10)+A(11) D(5)=0           D(4)=A(15)+A(9)
-- D(3)=1  D(2)=A(7)        D(1)=A(15)+A(0)  D(0)=0

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY CIRCUITO_E04Q02 IS
          PORT(A,B,C,D,E,F,G: IN STD_LOGIC;
               Z:             OUT STD_LOGIC);
END CIRCUITO_E04Q02;

ARCHITECTURE ARC_CIRCUITO_E04Q02 OF CIRCUITO_E04Q02 IS
-- Multiplexador 8x1 do Experimento 03 Questão 01
COMPONENT MUX8X1 IS
          PORT (S: IN STD_LOGIC_VECTOR (2 DOWNTO 0);       -- SELETOR 3 BITS
                D: IN STD_LOGIC_VECTOR (7 DOWNTO 0);       -- ENTRADA 8 BITS
            Y_MUX: OUT STD_LOGIC);                         -- SAÍDA
END COMPONENT;

--Decodificador 4x16 do Experimento 03 Questão 02
COMPONENT DEC4X16 IS
          PORT (A: IN STD_LOGIC_VECTOR    (3 DOWNTO 0);              -- ENTRADA 4 BITS
            Y_DEC: OUT STD_LOGIC_VECTOR (15 DOWNTO 0));          -- SAÍDA 16 BITS
END COMPONENT;

-- Porta OR
COMPONENT PORTA_OR IS
          PORT(ENTRADA_1, ENTRADA_2: IN STD_LOGIC;
                       SAIDA_OR:   OUT STD_LOGIC);
END COMPONENT;

SIGNAL SELETOR_MUX: STD_LOGIC_VECTOR(2 DOWNTO 0); -- Seletores do Multiplexador: E,F,G
SIGNAL SELETOR_DEC: STD_LOGIC_VECTOR(3 DOWNTO 0); -- Seletores do Decoder: A,B,C,D
SIGNAL BARRAMENTO_MUX: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL BARRAMENTO_DEC: STD_LOGIC_VECTOR(15 DOWNTO 0);
SIGNAL D10_OR_D11, D15_OR_D9, D15_OR_D0: STD_LOGIC;

BEGIN
DECODIFICADOR: DEC4X16 PORT MAP (SELETOR_DEC,BARRAMENTO_DEC);
MULTIPLEXADOR: MUX8X1  PORT MAP (SELETOR_MUX,BARRAMENTO_MUX,Z);

OR01: PORTA_OR PORT MAP (BARRAMENTO_DEC(10),BARRAMENTO_DEC(11),D10_OR_D11);
OR02: PORTA_OR PORT MAP (BARRAMENTO_DEC(15),BARRAMENTO_DEC(9),D15_OR_D9);
OR03: PORTA_OR PORT MAP (BARRAMENTO_DEC(15),BARRAMENTO_DEC(0),D15_OR_D0);

-- SELETOR DO DECODER: A,B,C,D
SELETOR_DEC(3) <= A;
SELETOR_DEC(2) <= B;
SELETOR_DEC(1) <= C;
SELETOR_DEC(0) <= D;

-- SELETOR DO MUX: E,F,G
SELETOR_MUX(2) <= E;
SELETOR_MUX(1) <= F;
SELETOR_MUX(0) <= G;

BARRAMENTO_MUX(7) <= '1';
BARRAMENTO_MUX(6) <= D10_OR_D11;
BARRAMENTO_MUX(5) <= '0';
BARRAMENTO_MUX(4) <= D15_OR_D9;
BARRAMENTO_MUX(3) <= '1';
BARRAMENTO_MUX(2) <= BARRAMENTO_DEC(7);
BARRAMENTO_MUX(1) <= D15_OR_D0;
BARRAMENTO_MUX(0) <= '0';
END ARC_CIRCUITO_E04Q02;
```

Código 4: Modelagem da função lógica Z

Os componentes MUX8X1, DEC4X16 e PORTA_OR foram modelados com os seguintes códigos:

```
-- Experimento 03 - Questão 01
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 13/11/2024

-- Multiplexador 8x1
-- Entrada: Vetor S (3 bits) e Vetor D (8 bits)
-- Saída: Y (1 bit)

--Tabela Verdade:
--  S   |   Y
-- 000      D0
-- 001      D1
-- 010      D2
-- 011      D3
-- 100      D4
-- 101      D5
-- 110      D6
-- 111      D7

-- D7: 10000000
-- D6: 01000000
-- D5: 00100000
-- D4: 00010000
-- D3: 00001000
-- D2: 00000100
-- D1: 00000010
-- D0: 00000001

-- Usar atribuições condicionais WHEN-ELSE

-- Alterado o nome da entity e da saida Y

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY MUX8X1 IS
          PORT (S: IN STD_LOGIC_VECTOR (2 DOWNTO 0);       -- SELETOR 3 BITS
                D: IN STD_LOGIC_VECTOR (7 DOWNTO 0);       -- ENTRADA 8 BITS
            Y_MUX: OUT STD_LOGIC);                         -- SAÍDA
END MUX8X1;

ARCHITECTURE ARC_MUX8X1 OF MUX8X1 IS
BEGIN
        Y_MUX <= D(0) WHEN (S = "000") ELSE     -- S=000 -> Y=D0
                 D(1) WHEN (S = "001") ELSE     -- S=001 -> Y=D1
                 D(2) WHEN (S = "010") ELSE     -- S=010 -> Y=D2
                 D(3) WHEN (S = "011") ELSE     -- S=011 -> Y=D3
                 D(4) WHEN (S = "100") ELSE     -- S=100 -> Y=D4
                 D(5) WHEN (S = "101") ELSE     -- S=101 -> Y=D5
                 D(6) WHEN (S = "110") ELSE     -- S=110 -> Y=D6
                 D(7) WHEN (S = "111") ELSE     -- S=111 -> Y=D7
                 '0';                           -- Demais casos - Y=0


END ARC_MUX8X1;
```

Código 5:Código do Multiplexador 8 para 1

```
-- Experimento 03 - Questão 02
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 13/11/2024

-- Decodificador 4x16
-- Entrada: Vetor A (4bits)
-- Saída: Vetor Y (16 bits)

--Tabela Verdade:
--  A   |          Y
-- 0000    0000 0000 0000 0001  -- 1
-- 0001    0000 0000 0000 0010  -- 2
-- 0010    0000 0000 0000 0100  -- 3
-- 0011    0000 0000 0000 1000  -- 4
-- 0100    0000 0000 0001 0000  -- 5
-- 0101    0000 0000 0010 0000  -- 6
-- 0110    0000 0000 0100 0000  -- 7
-- 0111    0000 0000 1000 0000  -- 8
-- 1000    0000 0001 0000 0000  -- 9
-- 1001    0000 0010 0000 0000  -- 10
-- 1010    0000 0100 0000 0000  -- 11
-- 1011    0000 1000 0000 0000  -- 12
-- 1100    0001 0000 0000 0000  -- 13
-- 1101    0010 0000 0000 0000  -- 14
-- 1110    0100 0000 0000 0000  -- 15
-- 1111    1000 0000 0000 0000  -- 16


-- Usar atribuições seletivas WITH-SELECT
-- Alterado o nome da entity e da saida Y

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY DEC4X16 IS
          PORT (A: IN STD_LOGIC_VECTOR   (3 DOWNTO 0);           -- ENTRADA 4 BITS
            Y_DEC: OUT STD_LOGIC_VECTOR (15 DOWNTO 0));          -- SAÍDA 16 BITS
END DEC4X16;

ARCHITECTURE ARC_DEC4X16 OF DEC4X16 IS
BEGIN
    WITH A SELECT
          Y_DEC <= "0000000000000001" WHEN "0000",  -- 1
                   "0000000000000010" WHEN "0001",  -- 2
                   "0000000000000100" WHEN "0010",  -- 3
                         "0000000000001000" WHEN "0011",  -- 4
                         "0000000000010000" WHEN "0100",  -- 5
                         "0000000000100000" WHEN "0101",  -- 6
                         "0000000001000000" WHEN "0110",  -- 7
                         "0000000010000000" WHEN "0111",  -- 8
                         "0000000100000000" WHEN "1000",  -- 9
                         "0000001000000000" WHEN "1001",  -- 10
                         "0000010000000000" WHEN "1010",  -- 11
                         "0000100000000000" WHEN "1011",  -- 12
                         "0001000000000000" WHEN "1100",  -- 13
                         "0010000000000000" WHEN "1101",  -- 14
                         "0100000000000000" WHEN "1110",  -- 15
                         "1000000000000000" WHEN "1111",  -- 16
                         "0000000000000000" WHEN OTHERS;
END ARC_DEC4X16;
```

Código 6:Código do Decodificador 4 para 16

```
-- Experimento 04 - Porta OR
-- Aluno: Pedro Henrique Dias Avelar 241037112
-- Turma 08
-- Data: 04/12/2024

-- PORTA OR

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY PORTA_OR IS
          PORT(ENTRADA_1, ENTRADA_2: IN STD_LOGIC;
                    SAIDA_OR:   OUT STD_LOGIC);
END PORTA_OR;

ARCHITECTURE ARC_PORTA_OR OF PORTA_OR IS
BEGIN
          SAIDA_OR <= ENTRADA_1 OR ENTRADA_2;
END ARC_PORTA_OR;
```

Código 7:Código da porta OR

Associamos os sinais SELETOR_DEC e SELETOR_MUX respectivamente as entradas A,B,C,D e E,F,G. Os sinais BARRAMENTO_MUX e BARRAMENTO_DEC foram associados respectivamente a entrada do multiplexador e a saída do decodificador. Os sinais D10_OR_D11, D15_OR_D9 e

D15_OR_D0 foram utilizados para representar a conexão das saídas do decodificador com as portas OR.

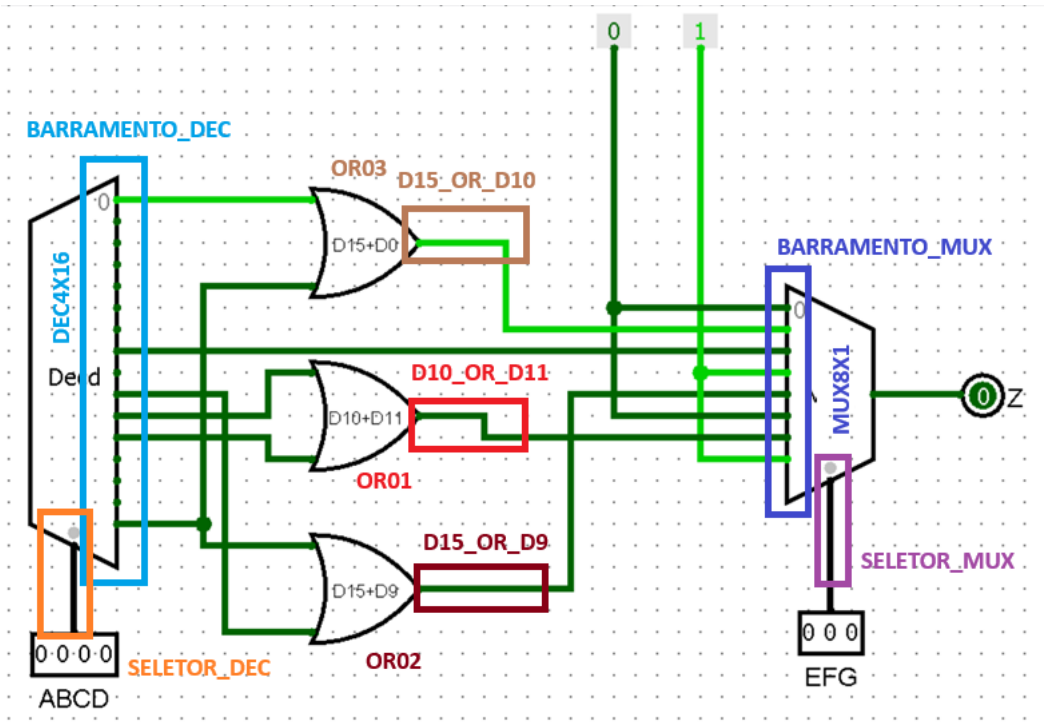

Figura 5:Representação gráfica dos sinais e componentes do código que implementa a função lógica Z

Para realizar a simulação, foram utilizados os seguintes clocks:

Entrada	Clock
A	1600
B	800
C	400
D	200
E	100
F	50
G	25

Tabela 6:Relação de clocks utilizados na simulação do experimento 02

A configuração acima permite que a simulação itere por todas as combinações possíveis das entradas A,B,C,D,E,F,G iterando do bit menos significativo ao bit mais significativo. A simulação foi realizada com um período de 1600ps e teve o seguinte resultado:

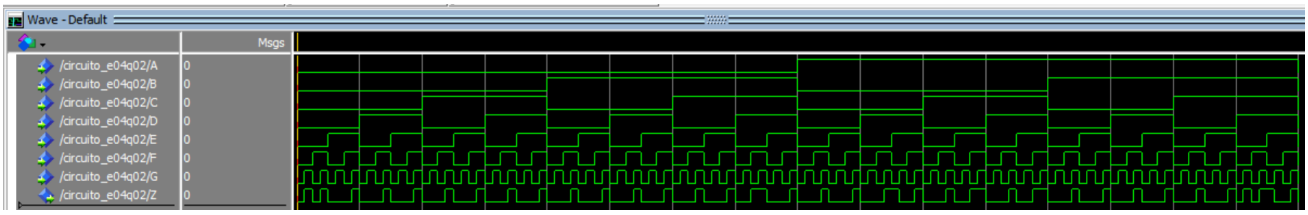

Figura 6:Resultado da simulação do experimento 02

Para facilitar a visualização dos resultados a tabela verdade foi montada para cada período de 100 ps:

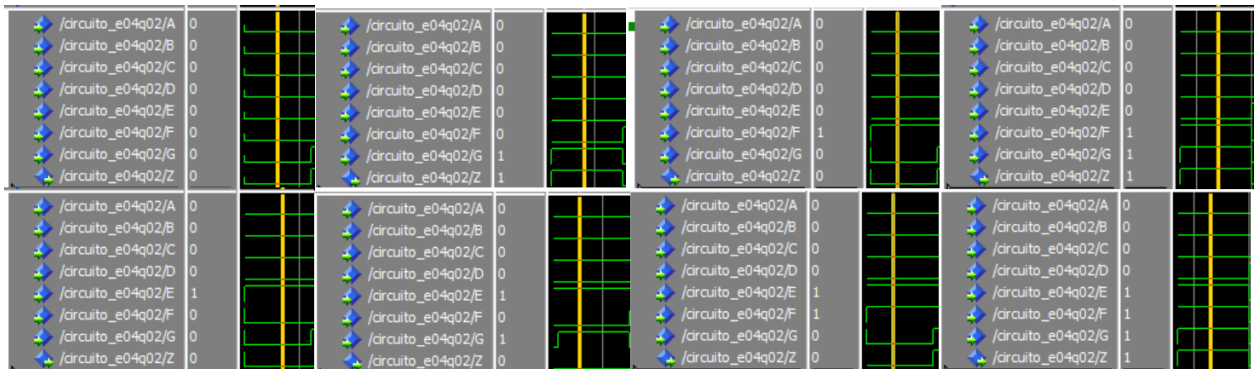

Figura 7:Resultados para 0000000 a 000111

#	A	B	C	D	E	F	G	Z
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1
2	0	0	0	0	0	1	0	0
3	0	0	0	0	0	1	1	1
4	0	0	0	0	1	0	0	0
5	0	0	0	0	1	0	1	0
6	0	0	0	0	1	1	0	0
7	0	0	0	0	1	1	1	1

Tabela 7:Resultados para 0000000 a 000111

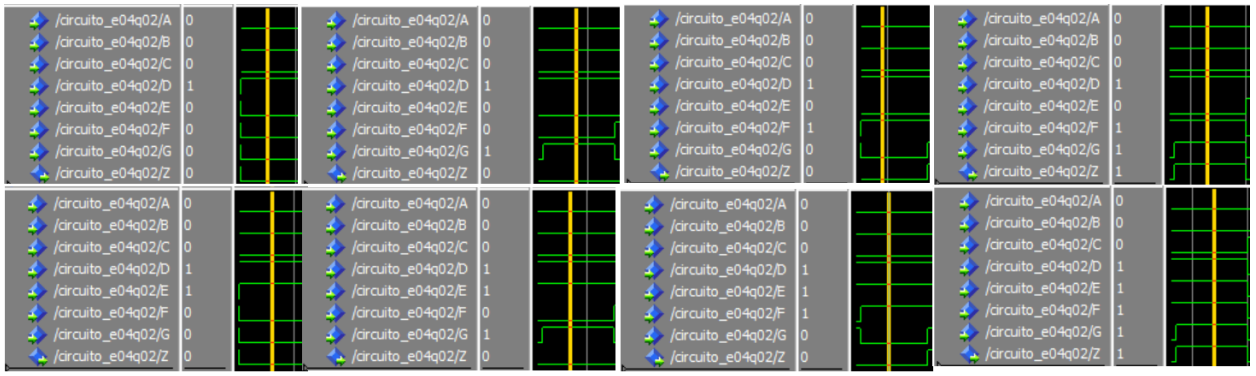

Figura 8:Resultados para 0001000 a 0001111

#	A	B	C	D	E	F	G	Z
8	0	0	0	1	0	0	0	0
9	0	0	0	1	0	0	1	0
10	0	0	0	1	0	1	0	0
11	0	0	0	1	0	1	1	1
12	0	0	0	1	1	0	0	0
13	0	0	0	1	1	0	1	0
14	0	0	0	1	1	1	0	0
15	0	0	0	1	1	1	1	1

Tabela 8:Resultados para 0001000 a 0001111

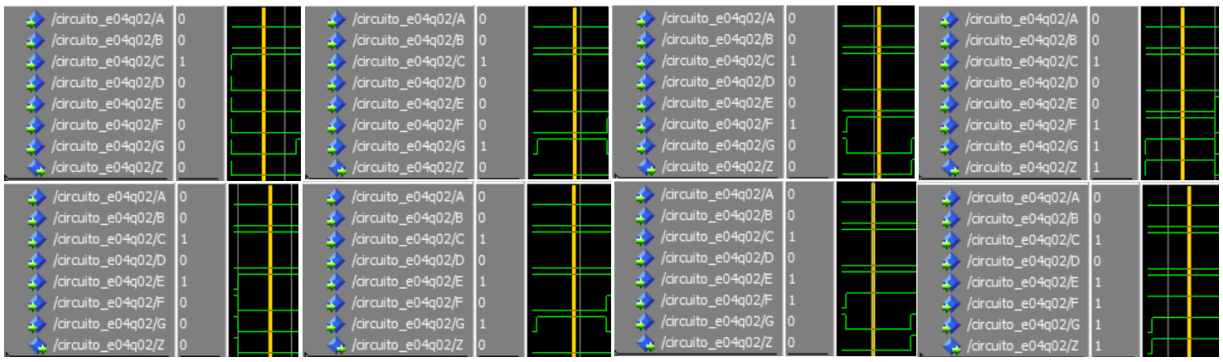

Figura 9:Resultados para 0010000 a 0010111

#	A	B	C	D	E	F	G	Z
16	0	0	1	0	0	0	0	0
17	0	0	1	0	0	0	1	0
18	0	0	1	0	0	1	0	0
19	0	0	1	0	0	1	1	1
20	0	0	1	0	1	0	0	0
21	0	0	1	0	1	0	1	0
22	0	0	1	0	1	1	0	0
23	0	0	1	0	1	1	1	1

Tabela 9:Resultados para 0010000 a 0010111

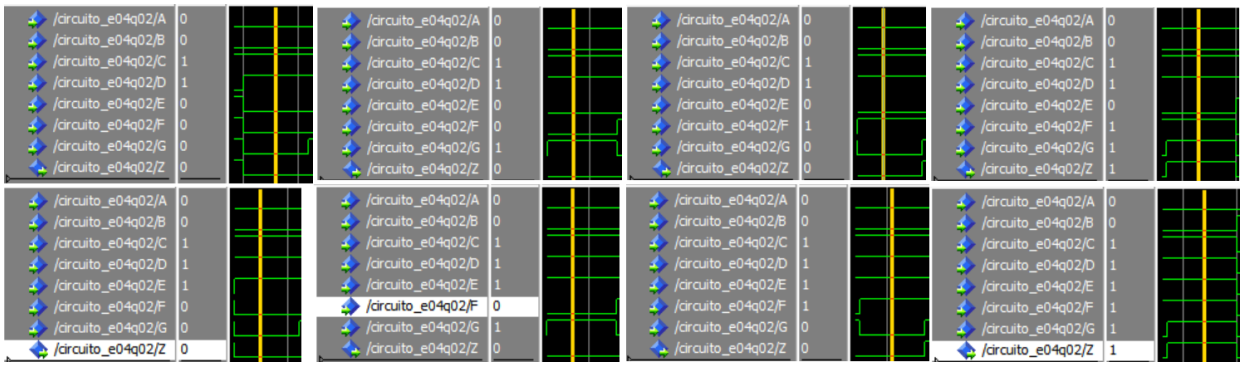

Figura 10:Resultados para 0011000 a 0011111

#	A	B	C	D	E	F	G	Z
24	0	0	1	1	0	0	0	0
25	0	0	1	1	0	0	1	0
26	0	0	1	1	0	1	0	0
27	0	0	1	1	0	1	1	1
28	0	0	1	1	1	0	0	0
29	0	0	1	1	1	0	1	0
30	0	0	1	1	1	1	0	0
31	0	0	1	1	1	1	1	1

Tabela 10:Resultados para 0011000 a 0011111

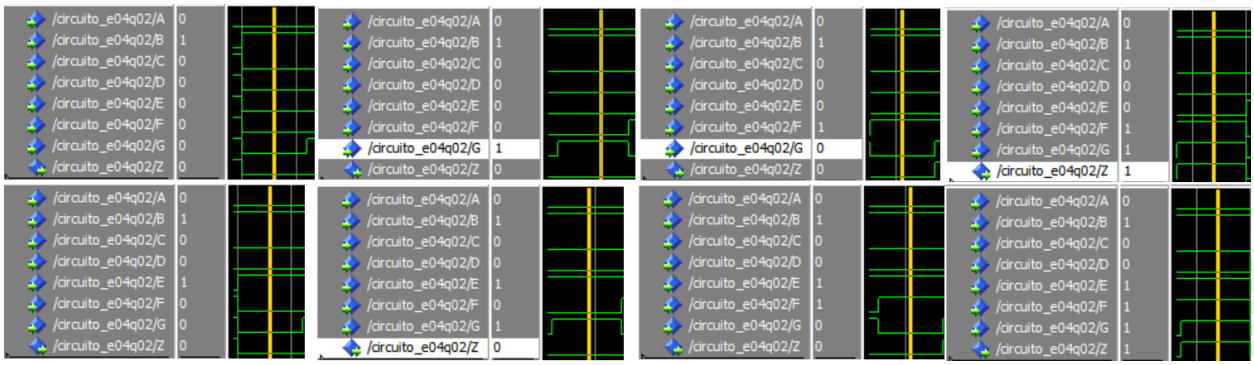

Figura 11:Resultados para 0100000 a 0100111

#	A	B	C	D	E	F	G	Z
32	0	1	0	0	0	0	0	0
33	0	1	0	0	0	0	1	0
34	0	1	0	0	0	1	0	0
35	0	1	0	0	0	1	1	1
36	0	1	0	0	1	0	0	0
37	0	1	0	0	1	0	1	0
38	0	1	0	0	1	1	0	0
39	0	1	0	0	1	1	1	1

Tabela 11:Resultados para 0100000 a 0100111

Figura 12:Resultados para 0101000 a 0101111

#	A	B	C	D	E	F	G	Z
40	0	1	0	1	0	0	0	0
41	0	1	0	1	0	0	1	0
42	0	1	0	1	0	1	0	0
43	0	1	0	1	0	1	1	1
44	0	1	0	1	1	0	0	0
45	0	1	0	1	1	0	1	0
46	0	1	0	1	1	1	0	0
47	0	1	0	1	1	1	1	1

Tabela 12:Resultados para 0101000 a 0101111

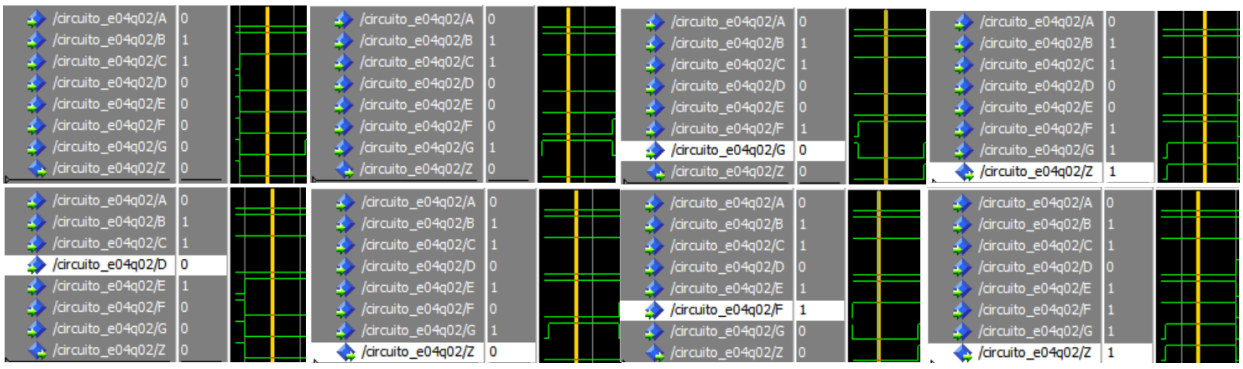

Figura 13:Resultados para 0110000 a 0110111

#	A	B	C	D	E	F	G	Z
48	0	1	1	0	0	0	0	0
49	0	1	1	0	0	0	1	0
50	0	1	1	0	0	1	0	0
51	0	1	1	0	0	1	1	1
52	0	1	1	0	1	0	0	0
53	0	1	1	0	1	0	1	0
54	0	1	1	0	1	1	0	0
55	0	1	1	0	1	1	1	1

Tabela 13:Resultados para 0110000 a 0110111

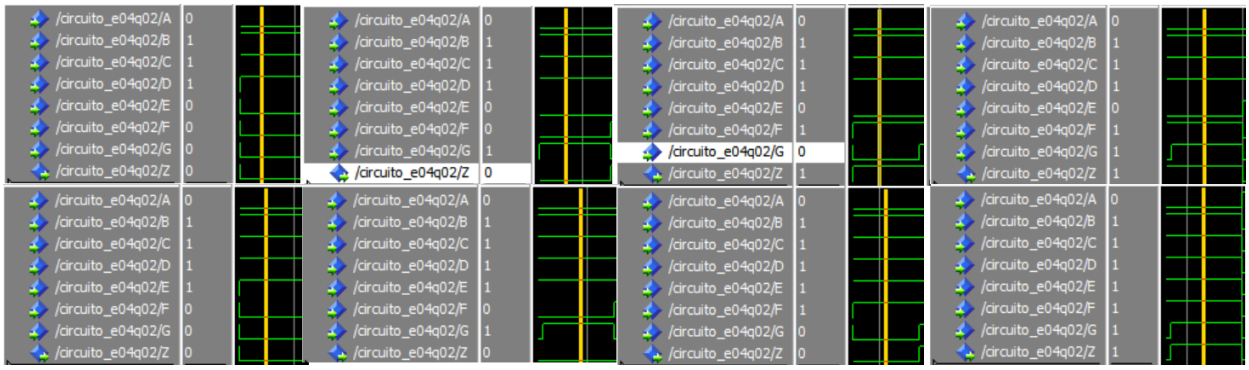

Figura 14:Resultados para 0111000 a 0111111

#	A	B	C	D	E	F	G	Z
56	0	1	1	1	0	0	0	0
57	0	1	1	1	0	0	1	0
58	0	1	1	1	0	1	0	1
59	0	1	1	1	0	1	1	1
60	0	1	1	1	1	0	0	0
61	0	1	1	1	1	0	1	0
62	0	1	1	1	1	1	0	0
63	0	1	1	1	1	1	1	1

Tabela 14:Resultados para 0111000 a 0111111

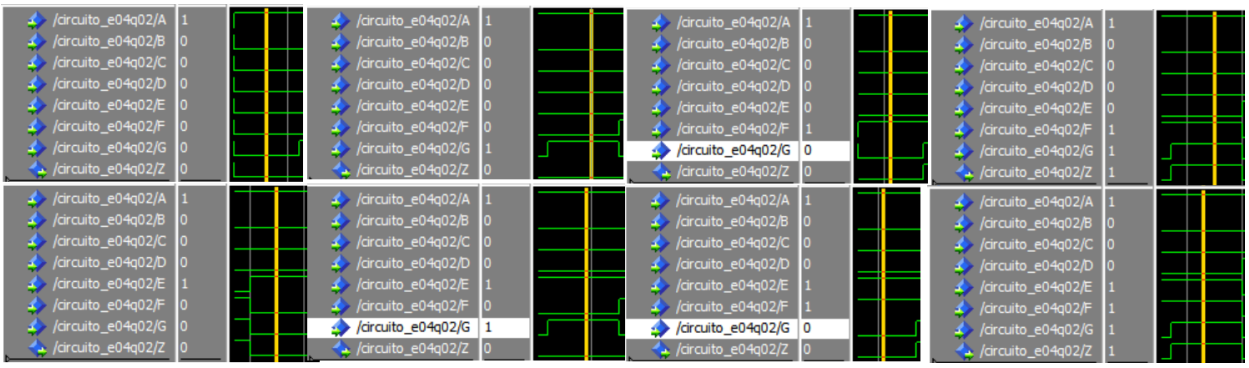

Figura 15:Resultados para 1000000 a 1000111

#	A	B	C	D	E	F	G	Z
64	1	0	0	0	0	0	0	0
65	1	0	0	0	0	0	1	0
66	1	0	0	0	0	1	0	0
67	1	0	0	0	0	1	1	1
68	1	0	0	0	1	0	0	0
69	1	0	0	0	1	0	1	0
70	1	0	0	0	1	1	0	0
71	1	0	0	0	1	1	1	1

Tabela 15:Resultados para 1000000 a 1000111

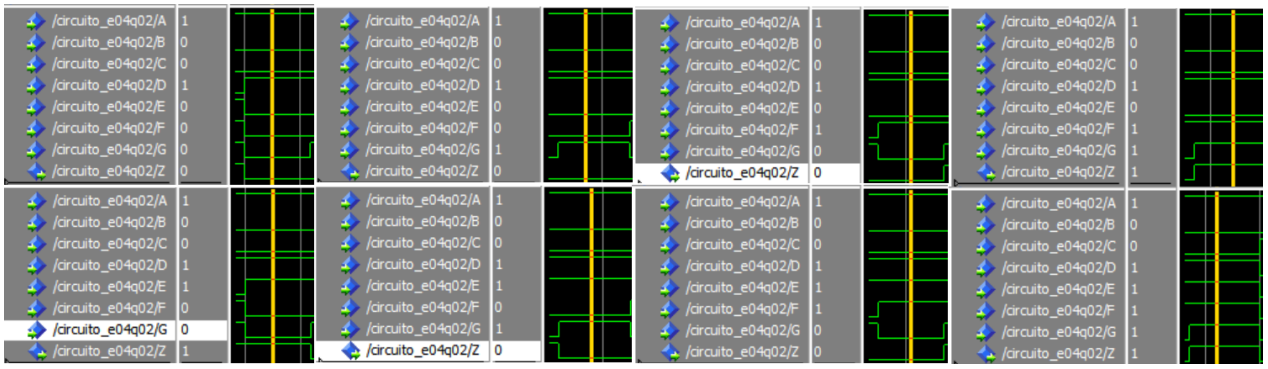

Figura 16:Resultados para 1001000 a 1001111

#	A	B	C	D	E	F	G	Z
72	1	0	0	1	0	0	0	0
73	1	0	0	1	0	0	1	0
74	1	0	0	1	0	1	0	0
75	1	0	0	1	0	1	1	1
76	1	0	0	1	1	0	0	1
77	1	0	0	1	1	0	1	0
78	1	0	0	1	1	1	0	0
79	1	0	0	1	1	1	1	1

Tabela 16:Resultados para 1001000 a 1001111

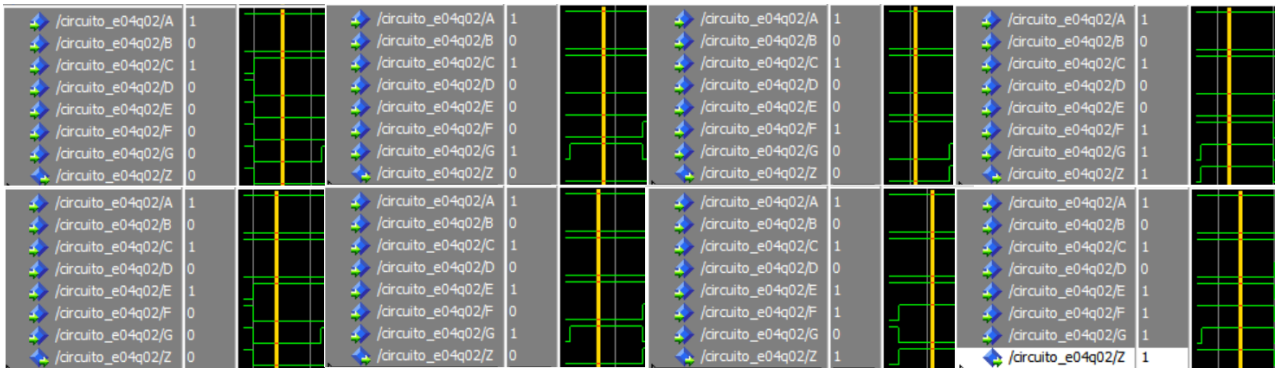

Figura 17:Resultados para 1010000 a 1010111

#	A	B	C	D	E	F	G	Z
80	1	0	1	0	0	0	0	0
81	1	0	1	0	0	0	1	0
82	1	0	1	0	0	1	0	0
83	1	0	1	0	0	1	1	1
84	1	0	1	0	1	0	0	0
85	1	0	1	0	1	0	1	0
86	1	0	1	0	1	1	0	1
87	1	0	1	0	1	1	1	1

Tabela 17:Resultados para 1010000 a 1010111

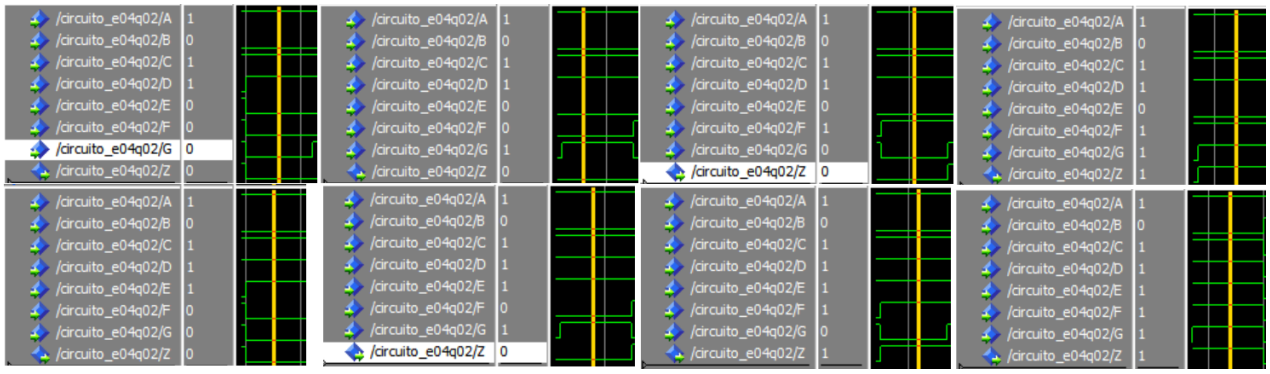

Figura 18:Resultados para 1011000 a 1011111

#	A	B	C	D	E	F	G	Z
88	1	0	1	1	0	0	0	0
89	1	0	1	1	0	0	1	0
90	1	0	1	1	0	1	0	0
91	1	0	1	1	0	1	1	1
92	1	0	1	1	1	0	0	0
93	1	0	1	1	1	0	1	0
94	1	0	1	1	1	1	0	1
95	1	0	1	1	1	1	1	1

Tabela 18:Resultados para 1011000 a 1011111

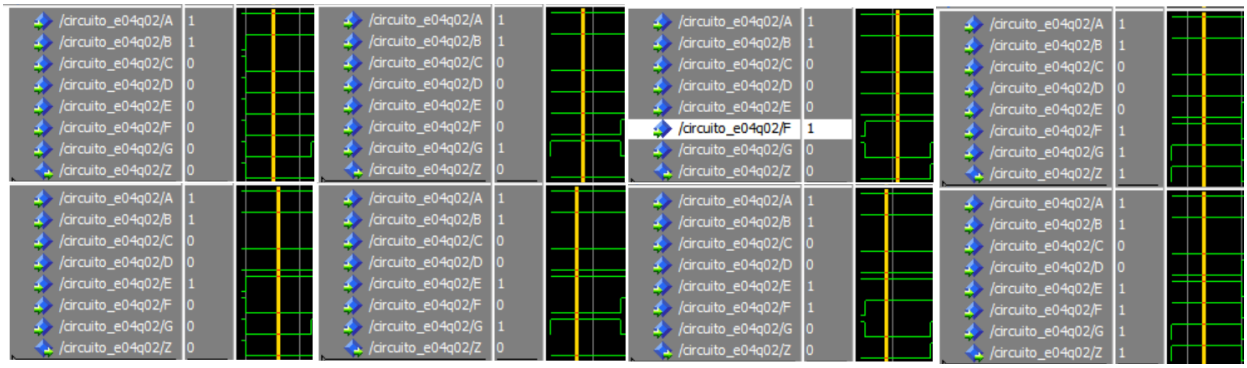

Figura 19:Resultados para 1100000 a 1100111

#	A	B	C	D	E	F	G	Z
96	1	1	0	0	0	0	0	0
97	1	1	0	0	0	0	1	0
98	1	1	0	0	0	1	0	0
99	1	1	0	0	0	1	1	1
100	1	1	0	0	1	0	0	0
101	1	1	0	0	1	0	1	0
102	1	1	0	0	1	1	0	0
103	1	1	0	0	1	1	1	1

Tabela 19:Resultados para 1100000 a 1100111

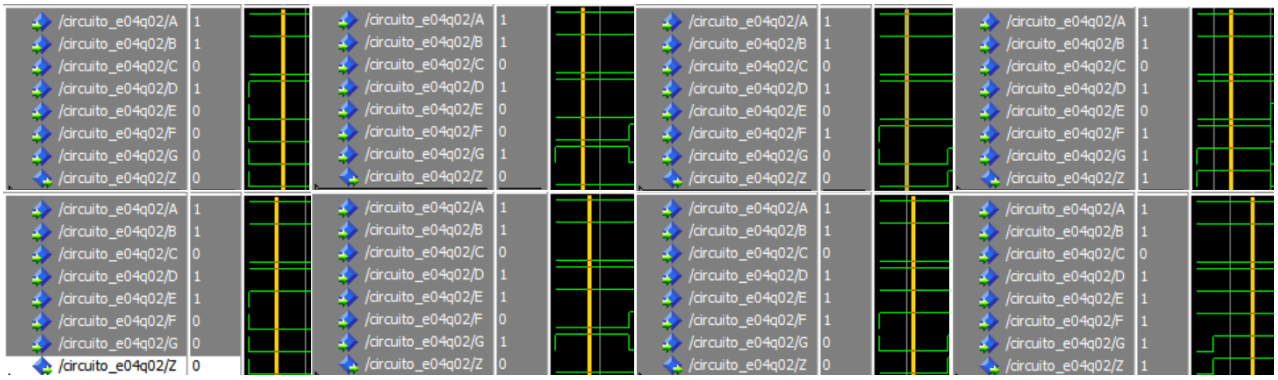

Figura 20:Resultados para 1101000 a 1101111

#	A	B	C	D	E	F	G	Z
104	1	1	0	1	0	0	0	0
105	1	1	0	1	0	0	1	0
106	1	1	0	1	0	1	0	0
107	1	1	0	1	0	1	1	1
108	1	1	0	1	1	0	0	0
109	1	1	0	1	1	0	1	0
110	1	1	0	1	1	1	0	0
111	1	1	0	1	1	1	1	1

Tabela 20:Resultados para 1101000 a 1101111

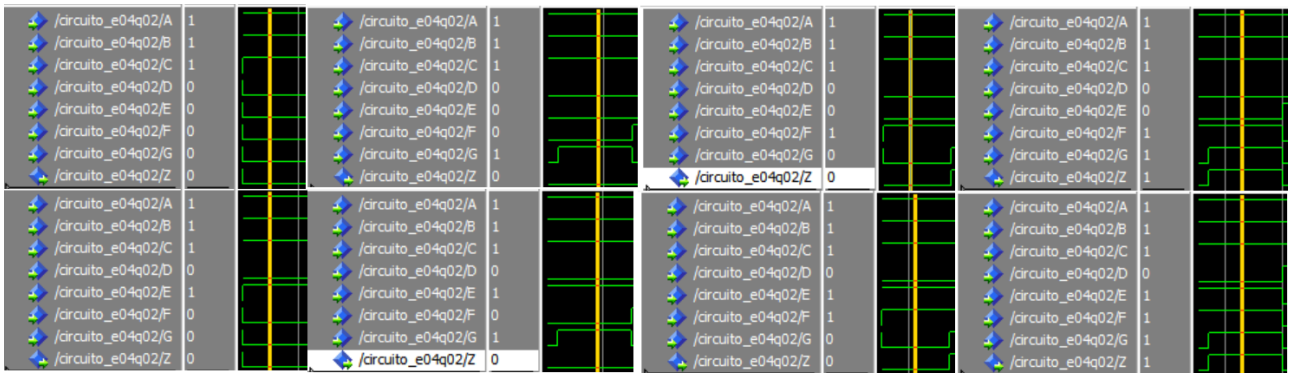

Figura 21:Resultados para 1110000 a 1110111

#	A	B	C	D	E	F	G	Z
112	1	1	1	0	0	0	0	0
113	1	1	1	0	0	0	1	0
114	1	1	1	0	0	1	0	0
115	1	1	1	0	0	1	1	1
116	1	1	1	0	1	0	0	0
117	1	1	1	0	1	0	1	0
118	1	1	1	0	1	1	0	0
119	1	1	1	0	1	1	1	1

Tabela 21:Resultados para 1110000 a 1110111

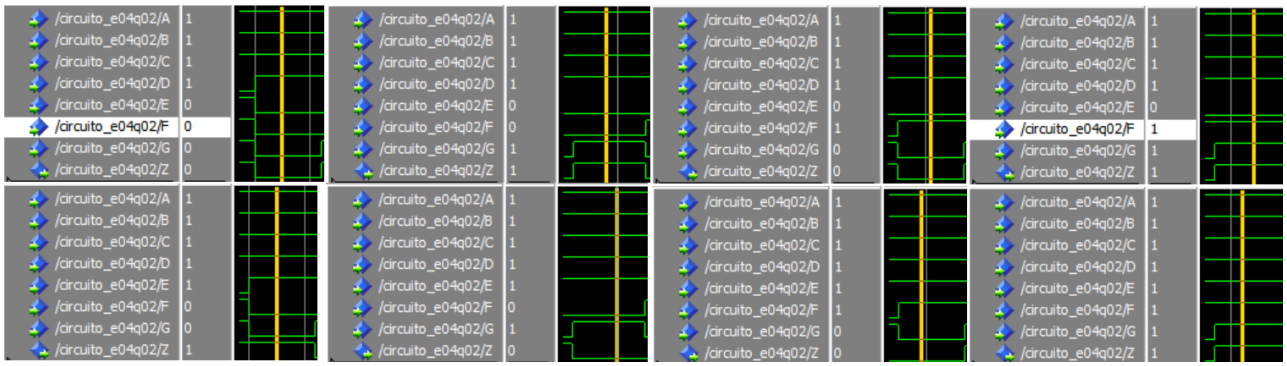

Figura 22:Resultados para 1111000 a 1111111

#	A	B	C	D	E	F	G	Z
120	1	1	1	1	0	0	0	0
121	1	1	1	1	0	0	1	1
122	1	1	1	1	0	1	0	0
123	1	1	1	1	0	1	1	1
124	1	1	1	1	1	0	0	1
125	1	1	1	1	1	0	1	0
126	1	1	1	1	1	1	0	0
127	1	1	1	1	1	1	1	1

Tabela 22:Resultados para 1111000 a 1111111

Juntando as tabelas 7 a 22 teremos então a tabela verdade completa para a função lógica Z:

#	A	B	C	D	E	F	G	Z	#	A	B	C	D	E	F	G	Z
0	0	0	0	0	0	0	0	0	64	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	1	65	1	0	0	0	0	0	1	0
2	0	0	0	0	0	1	0	0	66	1	0	0	0	0	1	0	0
3	0	0	0	0	0	1	1	1	67	1	0	0	0	0	1	1	1
4	0	0	0	0	1	0	0	0	68	1	0	0	0	1	0	0	0
5	0	0	0	0	1	0	1	0	69	1	0	0	0	1	0	1	0
6	0	0	0	0	1	1	0	0	70	1	0	0	0	1	1	0	0
7	0	0	0	0	1	1	1	1	71	1	0	0	0	1	1	1	1
8	0	0	0	1	0	0	0	0	72	1	0	0	1	0	0	0	0
9	0	0	0	1	0	0	1	0	73	1	0	0	1	0	0	1	0
10	0	0	0	1	0	1	0	0	74	1	0	0	1	0	1	0	0
11	0	0	0	1	0	1	1	1	75	1	0	0	1	0	1	1	1
12	0	0	0	1	1	0	0	0	76	1	0	0	1	1	0	0	1
13	0	0	0	1	1	0	1	0	77	1	0	0	1	1	0	1	0
14	0	0	0	1	1	1	0	0	78	1	0	0	1	1	1	0	0
15	0	0	0	1	1	1	1	1	79	1	0	0	1	1	1	1	1
16	0	0	1	0	0	0	0	0	80	1	0	1	0	0	0	0	0
17	0	0	1	0	0	0	1	0	81	1	0	1	0	0	0	1	0
18	0	0	1	0	0	1	0	0	82	1	0	1	0	0	1	0	0
19	0	0	1	0	0	1	1	1	83	1	0	1	0	0	1	1	1
20	0	0	1	0	1	0	0	0	84	1	0	1	0	1	0	0	0
21	0	0	1	0	1	0	1	0	85	1	0	1	0	1	0	1	0
22	0	0	1	0	1	1	0	0	86	1	0	1	0	1	1	0	1
23	0	0	1	0	1	1	1	1	87	1	0	1	0	1	1	1	1
24	0	0	1	1	0	0	0	0	88	1	0	1	1	0	0	0	0
25	0	0	1	1	0	0	1	0	89	1	0	1	1	0	0	1	0
26	0	0	1	1	0	1	0	0	90	1	0	1	1	0	1	0	0
27	0	0	1	1	0	1	1	1	91	1	0	1	1	0	1	1	1
28	0	0	1	1	1	0	0	0	92	1	0	1	1	1	0	0	0
29	0	0	1	1	1	0	1	0	93	1	0	1	1	1	0	1	0
30	0	0	1	1	1	1	0	0	94	1	0	1	1	1	1	0	1
31	0	0	1	1	1	1	1	1	95	1	0	1	1	1	1	1	1
32	0	1	0	0	0	0	0	0	96	1	1	0	0	0	0	0	0
33	0	1	0	0	0	0	1	0	97	1	1	0	0	0	0	1	0
34	0	1	0	0	0	1	0	0	98	1	1	0	0	0	1	0	0
35	0	1	0	0	0	1	1	1	99	1	1	0	0	0	1	1	1
36	0	1	0	0	1	0	0	0	100	1	1	0	0	1	0	0	0
37	0	1	0	0	1	0	1	0	101	1	1	0	0	1	0	1	0
38	0	1	0	0	1	1	0	0	102	1	1	0	0	1	1	0	0
39	0	1	0	0	1	1	1	1	103	1	1	0	0	1	1	1	1
40	0	1	0	1	0	0	0	0	104	1	1	0	1	0	0	0	0
41	0	1	0	1	0	0	1	0	105	1	1	0	1	0	0	1	0
42	0	1	0	1	0	1	0	0	106	1	1	0	1	0	1	0	0
43	0	1	0	1	0	1	1	1	107	1	1	0	1	0	1	1	1
44	0	1	0	1	1	0	0	0	108	1	1	0	1	1	0	0	0
45	0	1	0	1	1	0	1	0	109	1	1	0	1	1	0	1	0
46	0	1	0	1	1	1	0	0	110	1	1	0	1	1	1	0	0
47	0	1	0	1	1	1	1	1	111	1	1	0	1	1	1	1	1
48	0	1	1	0	0	0	0	0	112	1	1	1	0	0	0	0	0
49	0	1	1	0	0	0	1	0	113	1	1	1	0	0	0	1	0
50	0	1	1	0	0	1	0	0	114	1	1	1	0	0	1	0	0
51	0	1	1	0	0	1	1	1	115	1	1	1	0	0	1	1	1
52	0	1	1	0	1	0	0	0	116	1	1	1	0	1	0	0	0
53	0	1	1	0	1	0	1	0	117	1	1	1	0	1	0	1	0
54	0	1	1	0	1	1	0	0	118	1	1	1	0	1	1	0	0
55	0	1	1	0	1	1	1	1	119	1	1	1	0	1	1	1	1
56	0	1	1	1	0	0	0	0	120	1	1	1	1	0	0	0	0
57	0	1	1	1	0	0	1	0	121	1	1	1	1	0	0	1	1
58	0	1	1	1	0	1	0	1	122	1	1	1	1	0	1	0	0
59	0	1	1	1	0	1	1	1	123	1	1	1	1	0	1	1	1
60	0	1	1	1	1	0	0	0	124	1	1	1	1	1	0	0	1
61	0	1	1	1	1	0	1	0	125	1	1	1	1	1	0	1	0
62	0	1	1	1	1	1	0	0	126	1	1	1	1	1	1	0	0
63	0	1	1	1	1	1	1	1	127	1	1	1	1	1	1	1	1

Tabela 23:Tabela verdade para a função lógica Z

Pela tabela verdade, podemos então expressar Z na forma de sua soma canônica:

$$Z(A,B,C,D,E,F,G) = \sum_{A,B,C,D,E,F,G} m(1,3,7,11,15,19,23,27,31,35,39,43,47,51,55,58,59,63,67,71,75,76,79,86,87,91,94,95,99,103,107,111,115,119,121,123,124,127)$$