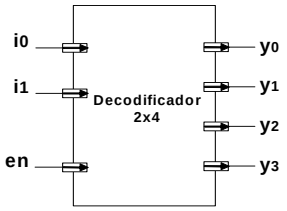
2.2.5 Decodificadores

Um decodificador é um circuito que converte códigos presentes nas entradas em um outro código nas saídas. Geralmente a entrada codificada tem menos bits do que a saída codificada. O decodificador mais usado é o decodificador n para 2^n, ou decodificador binário.

No Anexo E está descrito um decodificador que tem como entrada código BCD[12] e como saída o código para um display de sete segmentos.

Um decodificador binário completo é um módulo que tem n entradas 2^n saídas e uma entrada especial (*enable*) para habilitar as saídas correspondentes ao código binário nas entradas.

A cada instante uma única saída será ativada baseada no valor da entrada de n Bits. Por exemplo, o código binário das entradas (`i0, i1`) determina qual saída é ativada, caso a entrada de habilitação esteja ativa (`en <= '1'`), conforme representado pelo bloco diagrama e respectiva tabela verdade da Figura 2.29.



Entradas			Saídas			
i1	i0	en	y3	y2	y1	y0
x	x	0	0	0	0	0
0	0	1	0	0	0	1
0	1	1	0	0	1	0
1	0	1	0	1	0	0
1	1	1	1	0	0	0

Figura 2.29 - Decodificador binário, bloco diagrama e tabela verdade.

A Figura 2.30 ilustra o decodificador binário completo 2x4 em nível de portas lógicas obtido da tabela verdade da Figura 2.29.

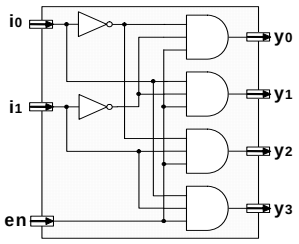


Figura 2.30 – Decodificador binário 2x4 em nível de portas lógicas.

Um decodificador binário é comumente utilizado para identificar e habilitar um elemento dentre um conjunto de dispositivos, com base num código ou endereço. Por exemplo, um banco de memórias RAM utilizadas por um processador, que são selecionadas individualmente conforme o barramento de endereços é decodificado para acessar e habilitar o dispositivo de memória correspondente a sua faixa de endereços, conforme ilustra a Figura 2.31.

[12] BCD – *Binary Coded Decimal*, representação binária dos dez símbolos decimais (0 a 9).

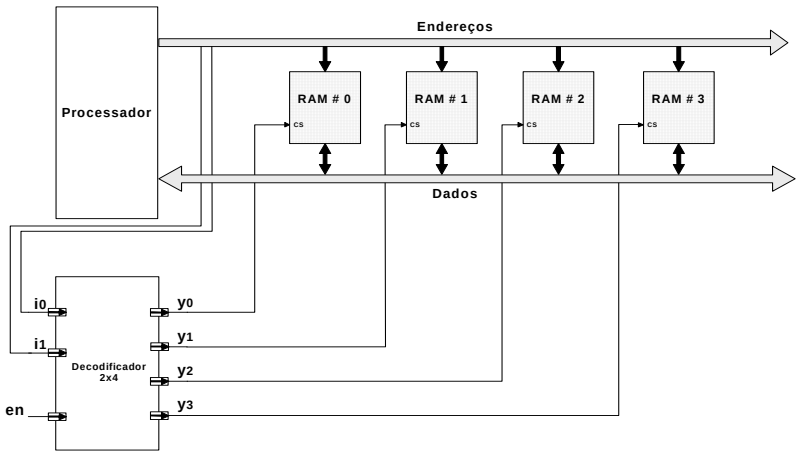


Figura 2.31 – Decodificador binário completo 2x4 utilizado como seletor de dispositivos.

Além de inúmeras aplicações, este tipo de decodificador é utilizado para sintetizar circuitos combinados. O circuito resultante, em geral, não é uma boa alternativa do ponto de vista de custos, porém seu projeto é imediato e alterações são fáceis de implementar.

A implementação do decodificador 2x4 é desenvolvido com as declarações WHEN/ELSE, para implementar o circuito da Figura 2.30. Na seqüência é apresentado código VHDL correspondente, transcrito como segue:

```
-----------------------------------------------
-- Circuito: decodificador 2x4:(deco1_2x4.vhd)
--                       en habilita saida
--                        i Entrada, i = (00:11)
--                        y Saída (WHEN/ELSE)
-----------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------
ENTITY deco2x4 IS
      PORT (en     : IN STD_LOGIC;
            I      : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
            Y      : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
END deco2x4 ;
-------------------------------------------
 ARCHITECTURE deco1_2x4 OF deco2x4 IS
     BEGIN
            y <=   "0001" WHEN i ="00" And en = '1' ELSE
                   "0010" WHEN i ="01" And en = '1' ELSE
                   "0100" WHEN i ="10" And en = '1' ELSE
                   "1000" WHEN i ="11" And en = '1' ELSE
                   "0000";
END deco1_2x4;
```

Uma descrição de *testbench* (testbench7) para validação do decodificador 2x4 é desenvolvida utilizando-se um sinal contador (`tb_i`) para selecionar a saída ativa. O sinal de habilitação inicia desabilitado e após 10 ns é habilitado. No código VHDL foi acrescentado o componente decodificador (`deco2x4`) que é instanciado (`deco1: deco2x4 PORT MAP (en => tb_en, i => tb_i, y => open);`) e interconectado ao processo de estímulo, conforme código VHDL transcrito como segue:

```
-- ****************************************
-- Testbench para simulação Funcional do
-- Circuito: decodificador 2x4:(deco1_2x4.vhd)
--                       en habilita saida
--                         i Entrada, i = (00:11)
--                         y Saída (WHEN/ELSE)
```

```
-- ***************************************
ENTITY testbench7 IS END;
-----------------------------------------------
-- Testbench para deco1_2x4.vhd.vhd
-- Validação assincrona
-----------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;
USE std.textio.ALL;

ARCHITECTURE tb_deco1_2x4 OF testbench7 IS
-----------------------------------
-- Declaração do componente deco2x4
-----------------------------------
component  deco2x4
      PORT (  en   : IN STD_LOGIC;
               I   : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
               Y   : OUT STD_LOGIC_VECTOR (3 DOWNTO 0));
end component;

signal tb_en      : std_logic;
signal tb_i       : STD_LOGIC_VECTOR (1 DOWNTO 0);

Begin

deco1: deco2x4 PORT MAP (en => tb_en, i => tb_i, y => open);

estimulo: PROCESS
   begin
      tb_i <= "00";
      tb_en<= '0';
      wait for 10 ns; tb_en <= '1';
      loop
        wait for 5 ns;
        tb_i <= tb_i + '1';
        end loop;
    end PROCESS estimulo;
end tb_deco1_2x4;
```

Os resultados obtidos na saída do decodificador 2x4 são apresentados na Figura 2.32. Neste *testbench* não são utilizados sinais de relógio para estímulo das entradas, apenas os estímulos gerados pelo sinal `tb_i`, que após inicializado em `"00"` é incrementado em uma unidade (`tb_i <= tb_i + '1';`) a cada 5 ns (linhas 40, 41, 42 e 43 do código VHDL - Figura 2.32).

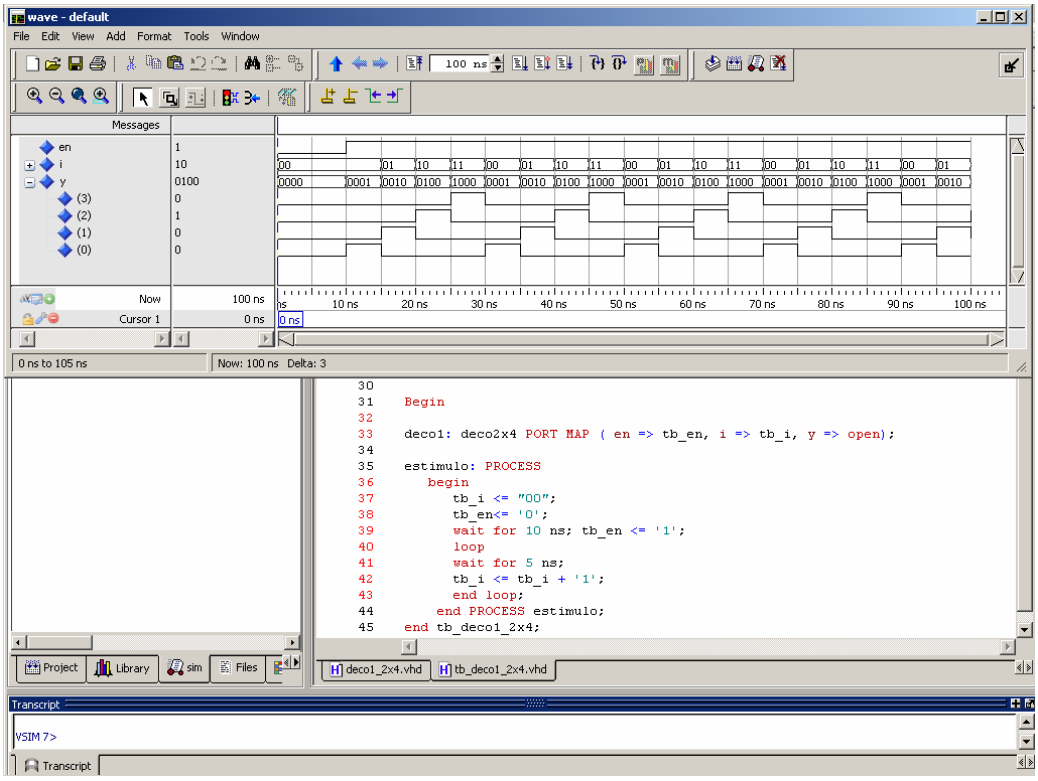


Figura 2.32 - Resultados da simulação do decodificador 2x4.

Na Figura 2.33 está apresentado um detalhamento da simulação do decodificador. Como pode ser observado, até 10 ns do início da simulação (cursor 1) todas as saídas (`y(3),y(2),y(1), y(0)`) permanecem desabilitadas, pois a entrada (`en`) de habilitação apresenta o valor lógico '0'. Após 10 ns (`wait for 10 ns; tb_en <= '1';`) observa-se que cada uma das saídas é ativada na seqüência correspondente ao valor lógico binário das entradas (`i(1), i(0)`).

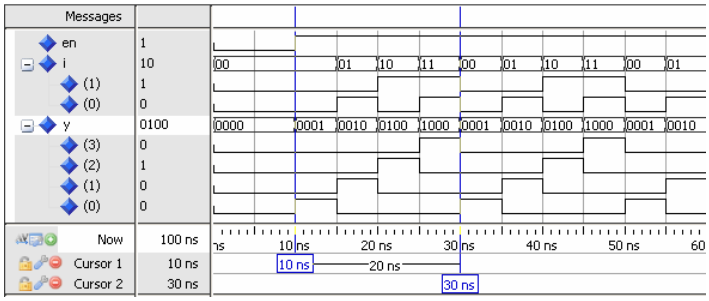


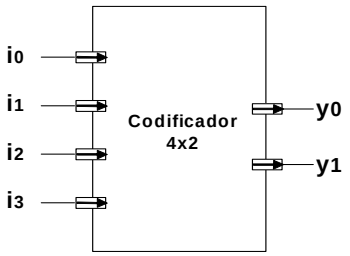
Figura 2.33 – Detalhamento dos resultados da simulação na saída do decodificador 2x4.

Durante a seleção das saídas o sinal de habilitação (`en`) permanece em estado lógico '1' permitindo que as saídas sejam ativadas na seqüência. Em 30 ns (cursor 2) o processo de estimulo irá se repetir deste ponto em diante a cada 20 ns.

2.2.6 Codificadores

Como exemplo da descrição do decodificador binário completo, um codificador realiza a função inversa de decodificador. O codificador é um circuito cujo código de saída tem normalmente menos bits do que o código de entrada. O codificador mais simples é o 2^n para n ou codificador binário. Ele tem função oposta ao decodificador binário, também denominado de Codificador de Prioridade.

Por exemplo, um codificador de prioridade 4x2 é um dispositivo combinacional com 4 entradas (i_3, i_2, i_1, i_0) e duas saídas (y_1, y_0). As saídas y_1 e y_0, na forma de número binário, indicam qual a entrada de maior prioridade está ativa em "1", conforme ilustra a Figura 2.34 no bloco diagrama e a tabela verdade de um codificador de prioridade (4x2).



Entradas				Saídas	
i3	i2	i1	i0	y1	y0
0	0	0	0	Z	Z
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Figura 2.34 - Codificador de prioridade 4x2, bloco diagrama e tabela verdade.

Este circuito atribui à entrada i_0 a maior prioridade e à entrada i_3 a menor prioridade. Neste tipo de codificador apenas uma entrada é ativada de cada vez. Se mais de uma entrada for ativada no codificador de prioridade ele assume que, apenas uma das entradas é responsável por ativar a saída correspondente à entrada de mais alta prioridade. Se na entrada o bit menos significativo for "1" a saída é "00", se o bit seguinte for "1", a saída é dependentemente do anterior, e neste caso a saída permanece em "00", pois o bit anterior (menos significativo) possui maior prioridade. Caso o anterior não estivesse em "1" então a saída seria "01", e assim sucessivamente.

A Figura 2.35 ilustra o codificador de prioridade 4x2 em nível de portas lógicas obtido da tabela verdade da Figura 2.34. Neste codificador foi arbitrado que quando nenhuma das entradas estiver ativa, a saída assume alta impedância "ZZ". Esta condição e implementada pelo circuito representado pela porta NOR de quatro entradas, que é responsável pela habilitação dos *buffers tri-state* (Anexo C) para as saídas (y_0, y_1) do codificador.

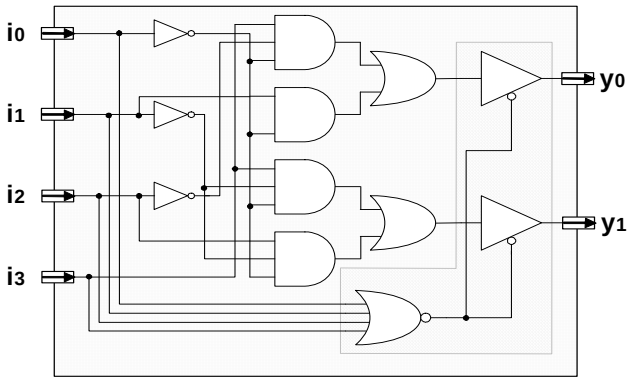


Figura 2.35 – Codificador de prioridade 4x2 em nível de portas lógicas.

Os codificadores são utilizados como conversores de código. Os mais comuns utilizados são os códigos: Binário; BCD; Octal e Hexadecimal.

A descrição em VHDL do codificador de prioridade 4x2 é desenvolvido com as declarações WHEN/ELSE, de forma a implementar o circuito da Figura 2.35. Na seqüência é apresentado código VHDL correspondente, transcrito como segue:

```
-----------------------------------------------
-- Circuito: codificador 4x2:(code1_4x2.vhd)
--                i Entradas i = (3:0)
--                y Saídas    y = (00:11)
-----------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
-----------------------------------------
ENTITY code4x2 IS
      PORT (i :IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            y :OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
END code4x2;
-----------------------------------------
 ARCHITECTURE code1_4x2 OF code4x2 IS
     BEGIN
            y <=   "00" WHEN i(0) ='1' ELSE
                   "01" WHEN i(1) ='1' ELSE
                   "10" WHEN i(2) ='1' ELSE
                   "11" WHEN i(3) ='1' ELSE
                   "ZZ";
END code1_4x2;
```

A descrição do *testbench* (testbench8) para validação do codificador de prioridade 4x2 é desenvolvida utilizando-se de um sinal contador (`tb_i <= tb_i + '1';`) de 4 bits para estimular as entradas i(3:0). O sinal inicia desabilitando todas as entradas (`tb_i <= "0000";`) e após 5 ns é incrementado em uma unidade a cada 5 ns (em *loop*) sucessivamente, para configurar todas as combinações possíveis nas entradas do codificador, conforme código VHDL transcrito como segue:

```
-- ****************************************
-- Testbench para simulação Funcional do
-- Circuito: codificador 4x2:(code1_4x2.vhd)
--                       i Entradas i = (3:0)
--                       y Saídas    y = (00:11)
-- ****************************************
ENTITY testbench8 IS END;
-----------------------------------------------
-- Testbench para code1_4x2.vhd
-- Validação assincrona
-----------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_signed.all;
USE std.textio.ALL;

ARCHITECTURE tb_code1_4x2 OF testbench8 IS
-----------------------------------
-- Declaração do componente deco2x4
-----------------------------------
component  code4x2
      PORT (i     : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            y : OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
end component;

signal tb_i       : STD_LOGIC_VECTOR (3 DOWNTO 0);

Begin

code1: code4x2 PORT MAP (i => tb_i, y => open);

estimulo: PROCESS
   begin
      tb_i <= "0000";
      loop
        wait for 5 ns;
        tb_i <= tb_i + '1';
        end loop;
    end PROCESS estimulo;
end tb_code1_4x2;
-----------------------------------
```

Os resultados simulados na saída do codificador 4x2 são apresentados pela Figura 2.36. No testbench8 também não são utilizados sinais de relógio para estímulo das entradas, apenas

os estímulos gerados pelo sinal `tb_i`, que após ser inicializado em `"0000"` é incrementado em uma unidade (`tb_i <= tb_i + '1';`) a cada 5 ns (linhas 34, 35, 36 e 37 do código VHDL - Figura 2.36).

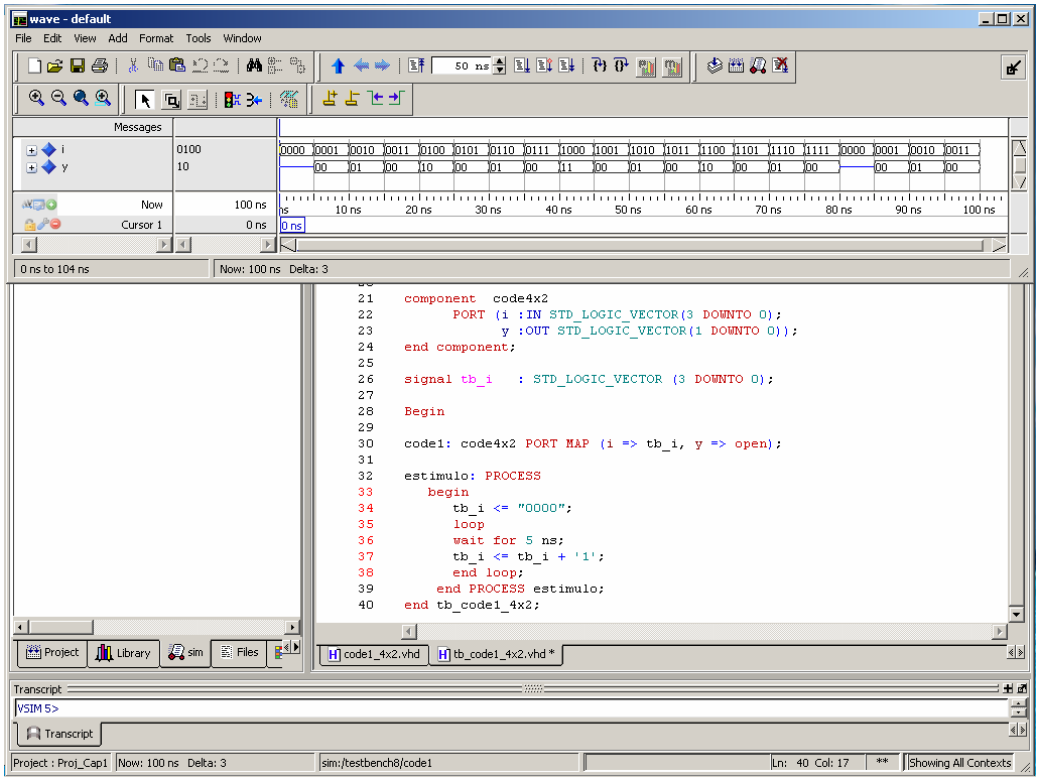


Figura 2.36 - Resultados da simulação do codificador 4x2.

Na Figura 2.37 está apresentado um detalhamento da simulação do codificador. Como pode ser observado, até 5 ns (cursor 1) a partir do início da simulação todas as entradas (`i(3),i(2),i(1),i(0)`) permanecem desabilitadas. Desta forma a saída do codificador apresenta alta impedância "ZZ". Após 5 ns (`wait for 5 ns;`) observa-se que os bits da saída `y` são ativados com o valor "00" correspondente à prioridade da entrada, pois para este período de tempo a entrada (`i(0)`) de maior prioridade encontra-se em estado lógico "1".

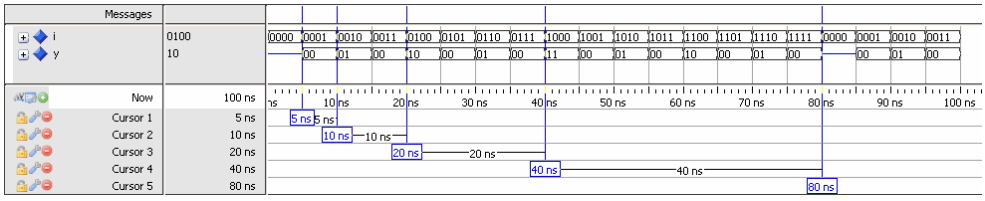


Figura 2.37 – Detalhamento dos resultados da simulação na saída do codificador 4x2.

De acordo com os estímulos do testbench8 as condições das entradas são incrementadas de um a cada 5 ns, desta forma impondo todas as condições possíveis paras as entradas do codificador.

Observa-se na simulação, entre os tempos de 10 ns até 20 ns (cursores 1 e 2), que as duas primeiras entradas (`i(1)` e `i(0)`) de mais alta prioridade são estimuladas com o valor lógico "0011" e a saída do codificador apresenta o valor "00" correspondente à entrada (`i(0)`) de prioridade "0" (mais alta).

Na simulação de 20 ns até 40 ns (cursores 3 e 4) semelhante situação ocorre para as três primeiras entradas (`i(2),i(1),i(0)`) e o decodificador de prioridade determina o valor de saída de acordo com a entrada de maior prioridade. Na seqüência da simulação, de 40 ns até 80 ns (cursores 4 e 5), as quatro entradas (`i(3),i(2),i(1),i(0)`) são combinadas de

forma binária crescente até que todas quatro tenham o valor lógico "1". Desta forma são validandos aos resultados na saída `y` do codificador que apresenta o valor "00" correspondente à entrada (`i(0)`) de prioridade "00" (mais alta).