Design: 4-to-2 encoder

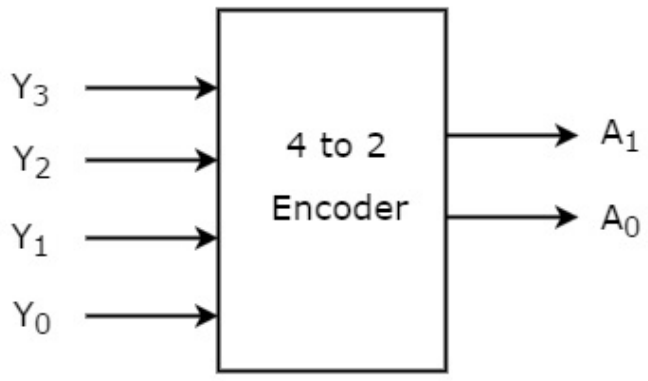
Description

In general an encoder is a device or process that converts data from one format to another. In Digital Logic, an encoder is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and ‘n’ output lines, hence it encodes the information from 2^n inputs into an n-bit code. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with ‘n’ bits. A 4 to 2 encoder takes 4 input lines and produces 2 output lines.

Truth Table

INPUT				OUTPUT	
i_3	i_2	i_1	i_0	o_1	o_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Diagram



Implementation

Using Structural Modelling

```
libraryIEEE;
useIEEE.STD_LOGIC_1164.ALL;

entity fourtwoenc is
```

```
Port(i:in    STD_LOGIC_VECTOR(3downto0);
    o:out    STD_LOGIC_VECTOR(1downto0));
end fourtwoenc;

architecture Behavioral of fourtwoenc is
begin
    o(0) <= i(1) or i(3);
    o(1) <= i(2) or i(3);
end Behavioral;
```

Using behaviour modelling using concurrent statements

```
libraryIEEE;
useIEEE.STD_LOGIC_1164.ALL;

entity fourtwoenc_conc is
    Port(i:in    STD_LOGIC_VECTOR(3downto0);
         o:out    STD_LOGIC_VECTOR(1downto0));
end fourtwoenc_conc;

architecture Behavioral of fourtwoenc_conc is
begin
    with i select o<=
    "00"when"0001",
    "01"when"0010",
    "10"when"0100",
    "11"when"1000",
    "ZZ"when others;
end Behavioral;
```

Using behaviour modelling using sequential statements

```
libraryIEEE;
useIEEE.STD_LOGIC_1164.ALL;

entity fourtwoenc_seq is
    Port(i:in    STD_LOGIC_VECTOR(3downto0);
         o:outSTD_LOGIC_VECTOR(1downto0));
end fourtwoenc_seq;

architecture Behavioral of fourtwoenc_seq is
    begin
        Process(i)
            begin
                case i is
                    when"0001" => o <= "00";
                    when "0010" => o <= "01";
                    when"0100" => o <= "10";
                    when"1000" => o <= "11";
                    when others => o <= "ZZ";
                end case;
```

```
        end Process;
end Behavioral;
```

Design: 8-to-3 encoder

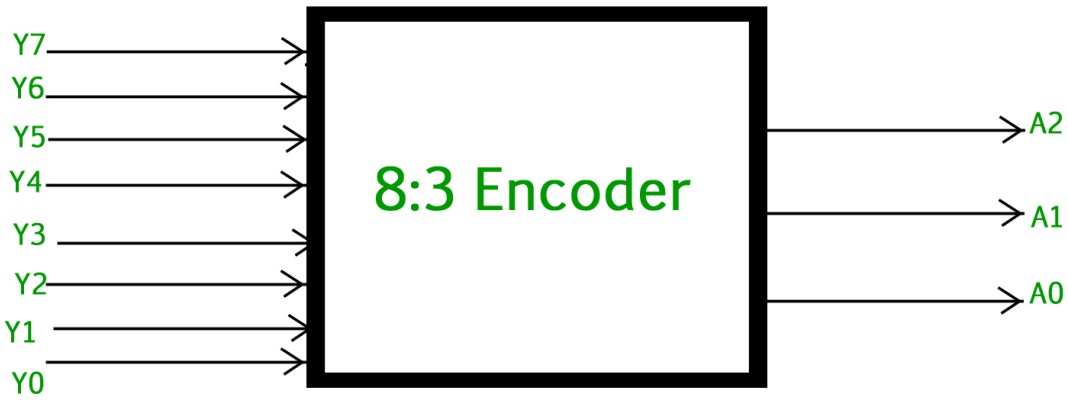
Description

In general an encoder is a device or process that converts data from one format to another. In Digital Logic, an encoder is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and ‘n’ output lines, hence it encodes the information from 2^n inputs into an n-bit code. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with ‘n’ bits. A 8 to 3 encoder has 8 input lines and 3 output lines

Truth Table

INPUTS										
i_7	i_6	i_5	i_4	i_3	i_2	i_1	i_0	o_2	o_1	o_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Diagram



Implementation

Using Structural modelling

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity eighttothreeenc is
    Port ( i : in    STD_LOGIC_VECTOR (7 downto 0);
           o : out    STD_LOGIC_VECTOR (2 downto 0));
end eighttothreeenc;

architecture Behavioral of eighttothreeenc is
begin
    o(0)<= i(1) or i(3) or i(5) or i(7);
    o(1)<= i(2) or i(3) or i(6) or i(7);
    o(2)<= i(4) or i(5) or i(6) or i(7);
end Behavioral;
```

Using Behavioral modelling using Select statements

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity eighttothree_select is
    Port ( i : in    STD_LOGIC_VECTOR (7 downto 0);
           o : out    STD_LOGIC_VECTOR (2 downto 0));
end eighttothree_select;

architecture Behavioral of eighttothree_select is
begin
    o(0)<= i(1) or i(3) or i(5) or i(7);
    o(1)<= i(2) or i(3) or i(6) or i(7);
```

```
    o(2)<= i(4) or i(5) or i(6) or i(7);
end Behavioral;
```

Using Behavioral modelling using Case statements

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity eighttothreee_case is
    Port ( i : in    STD_LOGIC_VECTOR (7 downto 0);
           o : out    STD_LOGIC_VECTOR (2 downto 0));
end eighttothreee_case;

architecture Behavioral of eighttothreee_case is
begin
    Process(i)
    begin
        case i is
            when "00000001" => o <= "000";
            when "00000010" => o <= "001";
            when "00000100" => o <= "010";
            when "00001000" => o <= "011";
            when "00010000" => o <= "100";
            when "00100000" => o <= "101";
            when "01000000" => o <= "110";
            when "10000000" => o <= "111";
        end case;
    end Process;
end Behavioral;
```

Design: Decimal-to-BCD encoder

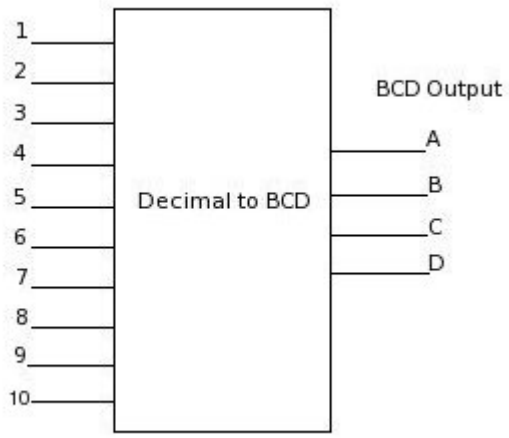
Description

A Decimal to BCD encoder has 10 input lines and 4 output lines. If i_x is set to 1 the output is the binary equivalent of x.

Truth Table

INPUTS										OUTPUTS			
i_9	i_8	i_7	i_6	i_5	i_4	i_3	i_2	i_1	i_0	o_3	o_2	o_1	o_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Diagram



Implementation

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity decimal_to_BCD is
    Port ( i : in    STD_LOGIC_VECTOR (9 downto 0);
           o : out    STD_LOGIC_VECTOR (3 downto 0));
end decimal_to_BCD;

architecture Behavioral of decimal_to_BCD is
begin
    o(3) <= i(9) or i(8);
    o(2) <= i(7) or i(6) or i(5) or i(4);
    o(1) <= i(7) or i(6) or i(3) or i(2);
    o(0) <= i(9) or i(7) or i(5) or i(3) or i(1);
end Behavioral;
```

Design: 1-to-2 Decoder

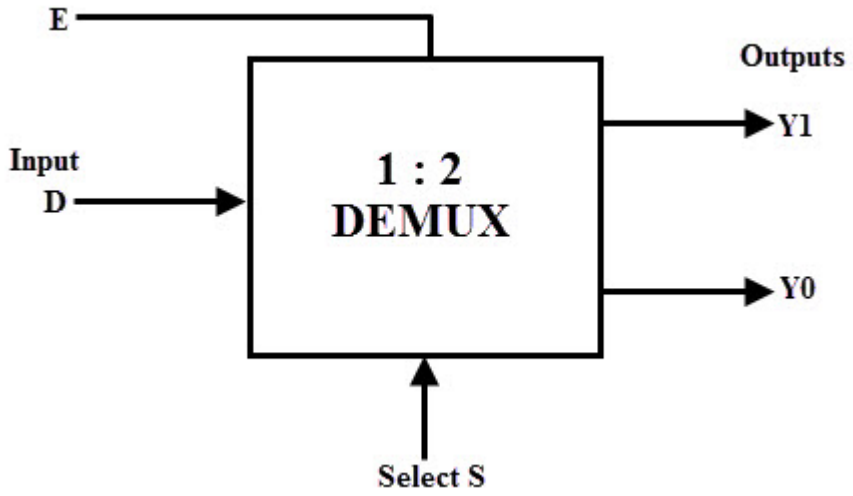
Description

Decoder is a combinational circuit that has ‘n’ input lines and maximum of 2^n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. A 1-to-2 decoder has 1 input lines and 2 output lines. An enable input is provided to switch the decoder on and off.

Truth Table

INPUTS		OUTPUTS	
e	i_0	o_1	o_0
0	x	0	0
1	0	0	1
1	1	1	0

Diagram



Implementation

Using Structural Modelling

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity onetotwo_decoder is
    Port(e: in BIT;
         i: in BIT;
         o: out BIT_VECTOR(1 downto 0));
end onetotwo_decoder;

architecture Behavioral of onetotwo_decoder is
begin
    o(0) <= e and not(i);
    o(1) <= e and i;
end Behavioral;
```

Using Behavioral Modelling using Concurrent Statements

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity onetotwo_decoder_conc is
    Port(e : in BIT;
         i: in BIT;
         o: out BIT_VECTOR(1 downto 0));
end onetotwo_decoder_conc;

architecture Behavioral of onetotwodecoder_conc is
begin
    with (e & i) select o <=
        "01" when "10",
```

```
        "10" when "11",
        "00" when others;
end Behavioral
```

Using Sequential Statements

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity onetotwo_decoder_sequential is
    PORT(e : in STD_LOGIC;
         i : in STD_LOGIC;
         o : out STD_LOGIC_VECTOR(1 downto 0));
end onetotwo_decoder_sequential;

architecture Behavioral of onetotwo_decoder_sequential is
begin
    process(e,i)
    begin
        if (e = '0') then o <= "00";
        elseif (i = '0') then o <= "01";
        elseif (i = '1') then o <= "10";
        end if
    end process
end Behavioral;
```

Design: 2-to-4 decoder

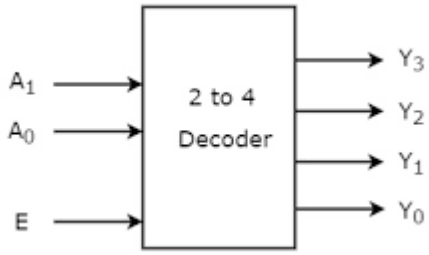
Description

A 2-to-4 decoder has 2 input lines and 4 output lines. An enable input is provided to switch the decoder on and off.

Truth Table

INPUTS			OUTPUTS			
e	i_1	i_0	o_3	o_2	o_1	o_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Diagram



Implementation

Using Structural Modelling

```
library IEEE;
use IEEE.BIT_1164.ALL;

entity twofourdecoder is
    PORT(e: in BIT;
         i: in BIT_VECTOR(1 downto 0);
         o: out BIT_VECTOR(3 downto 0));
end twofourdecoder;

architecture Behavioral of twofourdecoder is
begin
    o(0) <= e and not(i(1)) and not (i(0));
    o(1) <= e and not(i(1)) and (i(0));
    o(2) <= e and(i(1)) and not (i(0));
    o(3) <= e and (i(1)) and (i(0));
end Behavioral;
```

Using Behavioral Modelling Sequential Statements

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity two_four_decoder_sequential is
    Port(e : in STD_LOGIC;
         i : in STD_LOGIC_VECTOR(1 downto 0);
         o : out STD_LOGIC_VECTOR(3 downto 0));
end two_four_decoder_sequential;

architecture Behavioral of two_four_decoder_sequential is
begin
    process(e, i)
    begin
        if(e = '0') then o <= '0000';
        elseif(i = '00') then o <= '0001';
        elseif(i = '01') then o <= '0010';
        elseif(i = '10') then o <= '0100';
```

```
        elseif(i = '11') then o <= '1000';
        end if;
    end process;
end Behavioral;
```

Using Behavioral Modelling Concurrent Statements

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity two_four_decoder_conc is
    Port ( e : in    STD_LOGIC;
           i : in    STD_LOGIC_VECTOR (1 downto 0);
           o : out    STD_LOGIC_VECTOR (3 downto 0));
end two_four_decoder_conc;

architecture Behavioral of two_four_decoder_conc is
begin
    with (e & i) select o<=
        "0001" when "100",
        "0010" when "101",
        "0100" when "110",
        "1000" when "111",
        "0000" when others;
end Behavioral;
```

Design: 3-to-8 decoder

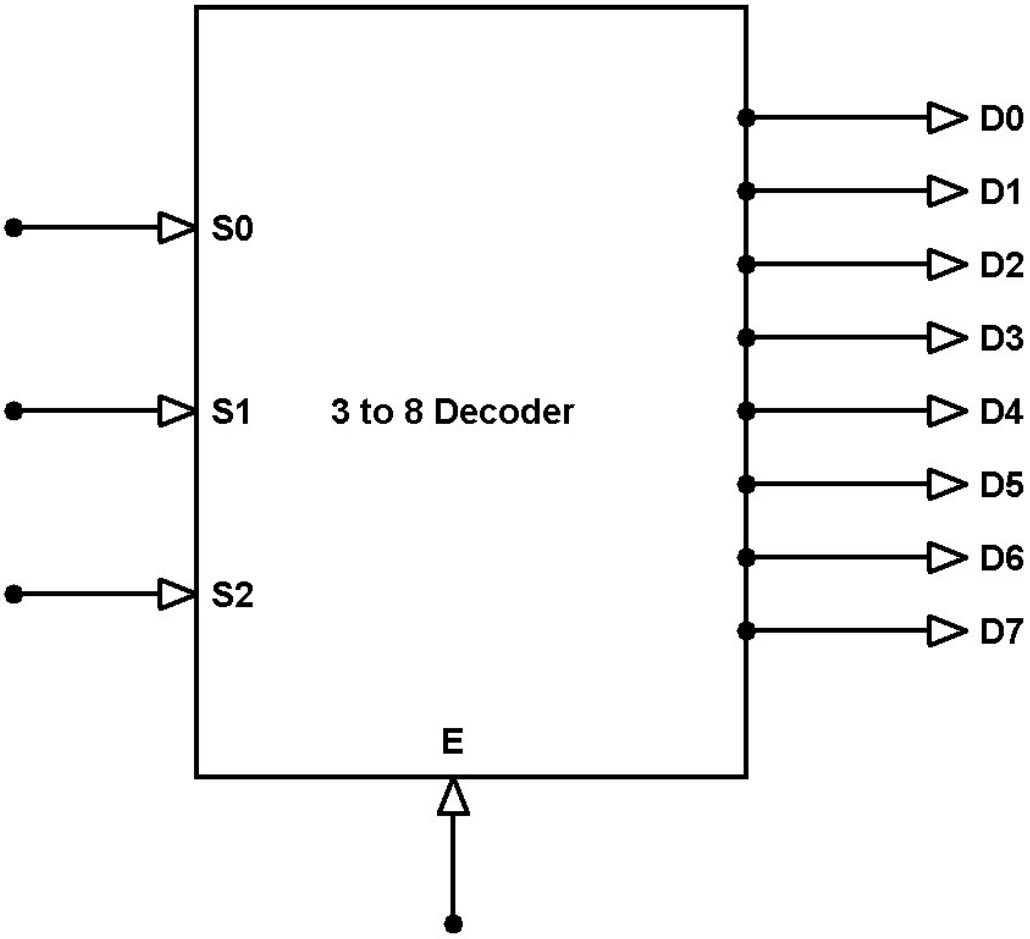
Description

A 3-to-8 decoder has 3 input lines and 8 output lines. An enable input is provided to switch the decoder on and off.

Truth Table

INPUTS				OUTPUTS							
e	i_2	i_1	i_0	o_7	o_6	o_5	o_4	o_3	o_2	o_1	o_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Diagram



Implementation

Using Component Instantiate

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity three_to_eight_comp is
    Port ( inp : in STD_LOGIC_VECTOR (2 downto 0);
           en : in  STD_LOGIC;
           op : out STD_LOGIC_VECTOR (7 downto 0));
end three_to_eight_comp;

architecture Behavioral of three_to_eight_comp is
    component two_to_four_decoder is
        Port( e : in STD_LOGIC;
              i : in STD_LOGIC_VECTOR(1 downto 0)
              o : out STD_LOGIC_VECTOR(3 downto 0));
    end component
```

```
    signal notinp: STD_LOGIC;
    begin
        notinp <= not inp(2);
        dec1: two_to_four_decoder port map(inp(2),inp(1 downto 0),op(7 downto 4));
dec2: two_to_four_decoder port map(notinp,inp(1 downto 0),op(3 downto 0));
end Behavioral;
```

Using Procedural Statement

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity three_to_eight_proc is
    Port ( inp : in STD_LOGIC_VECTOR (2 downto 0);
           op : out STD_LOGIC_VECTOR (7 downto 0));
end three_to_eight_proc;

architecture Behavioral of three_to_eight_proc is
procedure two_to_four_decoder
            (e : in STD_LOGIC (1 downto 0);
             o : out STD_LOGIC_VECTOR(3 downto 0)) is
begin
    with (e&i) select o <=
        "0001" when "100",
        "0010" when "101",
        "0100" when "110",
        "1000" when "111",
        "0000" when others;
end procedure;

begin
    process(inp)
    variable temp_var: STD_LOGIC_VECTOR(7 downto 0);
    begin
        dec1: two_to_four_decoder(inp(2), inp(1 downto 0), temp_var(7 downto 4));
        dec2: two_to_four_decoder(not inp(2), inp(1 downto 0), temp_var(3 downto 0));
        op <= temp_var;
    end process;
end Behavioral;
```