PART A :

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity decoder\_2to4\_whenelse is

Port (

A : in STD\_LOGIC\_VECTOR(1 downto 0);

EN : in STD\_LOGIC;

Y : out STD\_LOGIC\_VECTOR(3 downto 0)

);

end decoder\_2to4\_whenelse;

architecture Behavioral of decoder\_2to4\_whenelse is

begin

Y <= "0001" when (EN = '1' and A = "00") else

"0010" when (EN = '1' and A = "01") else

"0100" when (EN = '1' and A = "10") else

"1000" when (EN = '1' and A = "11") else

"0000";

end Behavioral;

PART B:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity decoder\_2to4\_select is

Port (

A : in STD\_LOGIC\_VECTOR(1 downto 0);

EN : in STD\_LOGIC;

Y : out STD\_LOGIC\_VECTOR(3 downto 0)

);

end decoder\_2to4\_select;

architecture Behavioral of decoder\_2to4\_select is

begin

with A select

Y <= "0001" when "00",

"0010" when "01",

"0100" when "10",

"1000" when "11",

"0000" when others when EN = '1';

Y <= "0000" when EN = '0';

end Behavioral;

Discussion

**روش when-else:**

* **ساده و خوانا برای شرایط ترکیبی:**  
  می‌توان ترکیبی از چند شرط مثل EN و A را در هر خط بررسی کرد.
* **اولویت‌بندی صریح:**  
  چون شرط‌ها به صورت زنجیره‌ای بررسی می‌شوند، ترتیب خطوط اهمیت دارد.
* **قابل فهم برای شرایط کنترل‌شده:**  
  برای مثال، اگر EN فعال نباشد، به راحتی می‌توان خروجی‌ها را صفر کرد.

**مزیت‌ها:**

* مناسب برای شرایط وابسته به چند شرط مثل EN + ورودی
* راحت برای گسترش به شرط‌های بیشتر

**روش with-select-when:**

* **تمیز و سازمان‌یافته:**  
  برای مقادیر مشخص ورودی مثل A، به سادگی می‌توان مقدار خروجی تعیین کرد.
* **بدون ترتیب اولویت (همه همزمان):**  
  برخلاف when-else که ترتیبی است، with-select همه‌ی مقادیر را به‌صورت مستقل بررسی می‌کند.
* **برای مقادیر قطعی بسیار مناسب است:**  
  مثل دیکودرها، مالتی‌پلکسرها، جداول نگاشت و ...

**مزیت‌ها:**

* کد مرتب‌تر و قابل‌فهم‌تر برای حالاتی با مقدارهای مشخص مثلاً A = "00", "01", ...
* شبیه‌تر به جدول‌های منطقی

**نکته:**

* باید کنترل EN را جداگانه اضافه کنید، چون شرط EN داخل with-select قابل مدیریت مستقیم نیست.