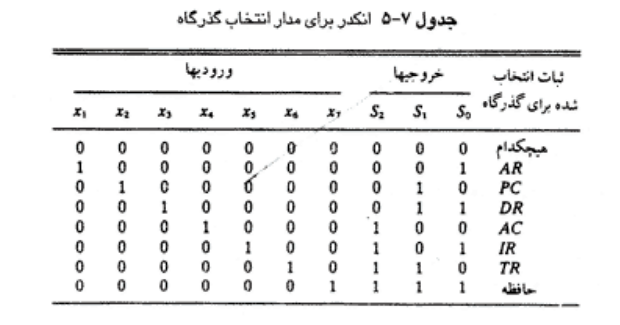
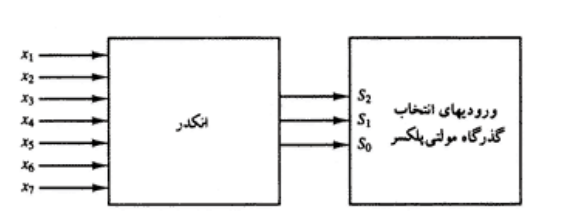
**در این پروژه میخواهیم برای مدار زیر کد ایجاد کننده و توضیحات آن را بگوییم :**





**( در اینجا من ورودی و خروجی هارو چند بیتی در نظر گرفتم و برای نوشتن اعداد هم بجای فُرمت دهدهی ، از فُرمت باینری استفاده کردم تا نحوه نوشتن شو آشنا بشید . ببینید توی تصویر سیگنال ها چند بیتی نبودن ، بلکه تک بیتی بودن با اسم های متفاوت اما من تصمیم گرفتم چند بیتی تعریف شون کنم تا حجم کُد کمتری رو داشته باشم ، بنابراین اینکه سیگنال هارو چند بیتی بگیریم یا نگیریم کاملا دل بخواهی هست.)**

**ویک ورودی اینتر داریم.X ابتدا یک ماژول برای انکدر میسازیم، در این انکدر ما هفت ورودی**

**داریم که هر سه به یک مالتی پلکسر میروند و یک خروجی خاص دارند.sهمچنین برای این دیکدر سه خروجی**

module DeCoder(

input [6:0]X,

input en,

output reg[2:0] s

**);**

**تمامی سیگنال هایی که در always مقدار دهی می شوند باید از نوع reg تعریف شوند . بنابراین هرچیزی که توی این بلاک دیدید که سمت چپ علامت = بود رو باید reg تعریف کنید . مابقی سیگنال ها کاملا سلیقه ای هست ، می تونن reg باشن و یا wire باشند . البته سیگنال هایی که ورودی اصلی ماژول هستن حق ندارند reg تعریف شوند . تمامی خط هایی که قراره توی بلاک always اجرا بشن رو باید بین begin و end قرار بدیم . به نوعی اینجوری مشخص میکنیم ابتدا و انتهای بلاک always کجاست . البته یک تبصره وجود داره ، اگه توی بلاک always فقط یک خط برنامه نوشتید ، می تونید این begin و end رو نذارید ولی بهتره که بذارید ، چرا ؟ چون ممکنه بعدا به برنامه خودتون مراجعه کنید و بخواین یه سری خطوط درون بلاک اضافه کنید.**.

always@(\*)

begin if ( en)

begin

case(x)

8’b00000000:begin S=0;end

8’b00000001:begin S=1;end

8’b00000010:begin S=2;end

8’b00000100:begin S=3;end

8’b00001000:begin S=4;end

8’b00010000:begin S=5;end

8’b00100000:begin S=6;end

8’b01000000:begin S=7;end

End case

End

Else if (~en)

Begin S=3’bz; end

End

Endmodule

