به نام خدا

توضیحات تمرین دوم درس طراحی سیستم های دیجیتال – دکتر فصحتی

امیرمحمد شربتی ناوان 402106112

1404/2/30

1. در این سوال باید یک گذرگاه مشترک ایجاد کنیم. این کار را با دو روش متفاوت انجام میدهیم. یک بار به کمک بافر سه‌حالته و یک بار هم به کمک تسهیم کننده. ابتدا به بافر سه‌حالته می‌پردازیم:

ابتدا یک tri-state buffer طراحی می‌کنیم. در ابتدا یک parameter تعریف میکنیم که مقدار پیش فرض آن 8 است. این مقدار نشان دهنده سایز بافر ما هست. یعنی قادر به عبور N می‌شود. طبیعتا ورودی N بیتی میخواهیم و یک سیگنال کنترلی به نام oe (output enable). اگر این سیگنال یک باشد، خروجی همان سیگنال ورودی است. در غیر این صورت مقدار z در آن قرار میگیرد.

این سیگنال خروجی از جنس output نیست. بلکه از جنس inout است. در وریلاگ inout port ها به اصطلاح دو طرفه یا همان bidirectional هستند. در اینجا ما میخواهیم هم بتوانیم دیتای ورودی را به گذرگاه منتقل کنیم (یعنی bi\_date باید در نقش خروجی عمل کند) و هم بعضی مواقع میخواهیم روی device چیزی بنویسیم. (یعنی محتویات گذرگاه را به دستگاه منتقل کنیم). پس نیازمند یک wire دوطرفه هستیم. (سیمی که به bus وصل می‌شود دوطرفه است.)



در فایل دوم یک کنترلر (bus\_controller) برای این گذرگاه ساختم. چون شکل سوال شامل دو دستگاه است، پس کنترلر هم فقط برای کنترل دو دستگاه پیاده سازی شد. در ورودی این ماژول، دو سیم برای request دو دستگاه داریم. یعنی دستگاه ها برای بارگذاری اطلاعات خود بر گذرگاه درخواست میدهند. دو wire هم داریم که کلاک و ریست هستند. دو خروجی داریم که اجازه دسترسی به دستگاه را صادر و کنترل میکنند. اگر reset نباشد، اولویت با درخواست دستگاه اول است. grant1 = 1 , grant2 = 0 می‌شود. برای درخواست دستگاه دوم هم همین است. و اگر هیچ کدام درخواستی نداشتند، grant برای هر دو صفر میشود.

به فایل اصلی می‌رسیم. به کمک این دو فایل گذرگاه مشترک با بافر سه‌حالته را می‌سازیم. ورودی این ماژول شامل متغیر های زیر است:

بعد از متغیر های کلاک و ریست، data\_in\_x میگوید دستگاه x میخواهد چه داده ای روی bus قرار دهد. reqX یعنی دستگاه X می‌خواهد روی باس داده را قرار دهد. Data\_out\_x هم داده ای است که دستگاه x از روی گذرگاه می‌خواند.

گذرگاه مشترک همان bi\_data است. g1 و g2 برای کنترل استفاده میشوند.

در خطوط بعد از ماژول هایی که داشتیم استفاده میکنیم. ابتدا کنترلر را instantiate میکنیم. خروجی grant را در g1 , g2 می‌ریزیم. از همین سیگنال ها در دو tri-state buffer ای که بعدا instantiate میکنیم، استفاده میکنیم. حداکثر یکی از این بافر های سه حالته فعال میشود و داده خود را روی باس قرار میدهد. دقیقا چیزی که از باس با بافر سه حالته انتظار داریم. در نهایت هم دیتای باس به عنوان خروجی device ها نشان داده میشود.

ماژول bus\_with\_mux مربوط به طراحی باس با mux است.

برای طراحی به کمک mux میتوان از این دستور ternary condition استفاده کرد:

assign bus = (sel == 1'b0) ? data\_in\_1 : data\_in\_2;

اما چون می‌خواهیم تاخیر ها را به درستی حساب کنیم، شاید روش gate level modeling روش بهتری باشد. پس این خط را به این خط که در واقع پیاده سازی داخلی mux است، تغییر دادم:

assign bus = (~sel & data\_in\_1) | (sel & data\_in\_2);

در این حالت راحت‌تر می‌توان تاخیر گیت ها را محاسبه کرد. بدین ترتیب منطق bus راحت بدست آمد. همین bus را در خطوط بعد به data\_out مربوط به دو دستگاه منتسب می‌کنیم.

درباره استفاده از tri-state buffer میتوان این مدار را طراحی کرد:

module tristate\_buf #(parameter N = 8)(

    input wire [N-1:0] data\_in,

    input wire oe,

    inout wire [N-1:0] bi\_data

);

    assign  bi\_data = oe ? data\_in : {N{1'bz}};

endmodule

و بنده این کد را به پیوست ارسال کردم. می‌توان در خط یکی به آخر مربوط به assign، delay را ست کرد. در ماژول اصلی پیاده سازی باس با tri-state می‌توان بدین شکل از آن instance گرفت:

   tristate\_buf #(.N(N)) ts\_buffer1 (.data\_in (data\_in\_1), .oe (g1), .bi\_data (bi\_data));

افزون بر این روش می‌توان از tri-state آماده وریلاگ استفاده کرد.