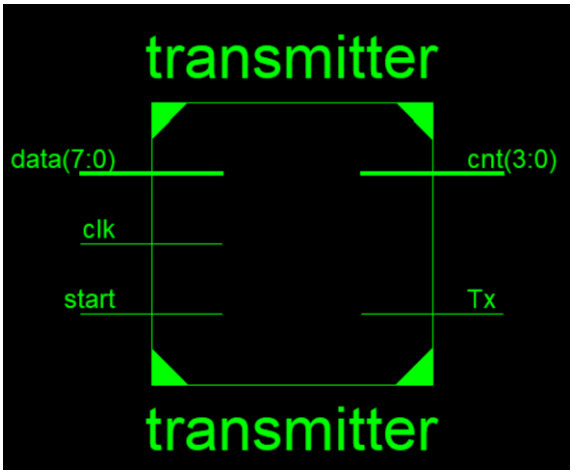
به نام خدا

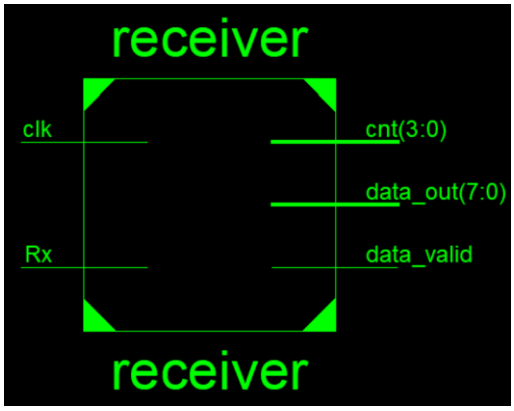
گزارش آزمایش شماره 3

محدثه غفوری(9632133)

گروه چهارشنبه عصر



ماژول فرستنده مطابق شکل بالا داده ای که کاربر میخواهد ارسال کند را از پورت data دریافت میکند و هنگامی که start توسط کاربر فعال شد برای ساخت فریم uart داده را به همراه استارت و استاپ بیت بصورت {1'b1,data,1'b0} در متغیر shift_register قرار میدهیم و چون clk / baud rate = 2500 است باید هر بیت از shift_register را طی 2500 کلاک مدام ارسال کنیم بنابراین برای شمارش 2500 کلاک از متغیر cnt_transmit استفاده میکنیم ، بدین صورت که پس از اینکه هر بیت از shift_register طی 2500 کلاک روی پورت خروجی TX قرار گرفت ، cnt_transmit صفر میشود و یک واحد به cnt اضافه میکند . cnt نشان دهنده تعداد بیت های shift_register است که طی 2500 کلاک ارسال شده اند. تا زمانی که تمامی بیت های shift_register ارسال نشده اند ، یعنی cnt به 11 نرسیده است ، بعد از ارسال هر بیت از سمت چپ 1’b1 به shift_register شیفت میدهیم و shift_register[0] را به پورت خروجی Tx متصل میکنیم . هنگامی که تمامی فریم uart ارسال شد برای ریست کردن shift_register و اینکه هنگامی که فرستنده داده ای ارسال نمیکند باید روی خروجی همواره عدد یک را ارسال کند ، کل shift_register را با 10’b1111_1111_11 پر میکنیم و برای اینکه برای دیتای بعدی اماده شویم باید cntو فلگ فعال شدن ماژول یعنی start_flag را 0 میکنیم



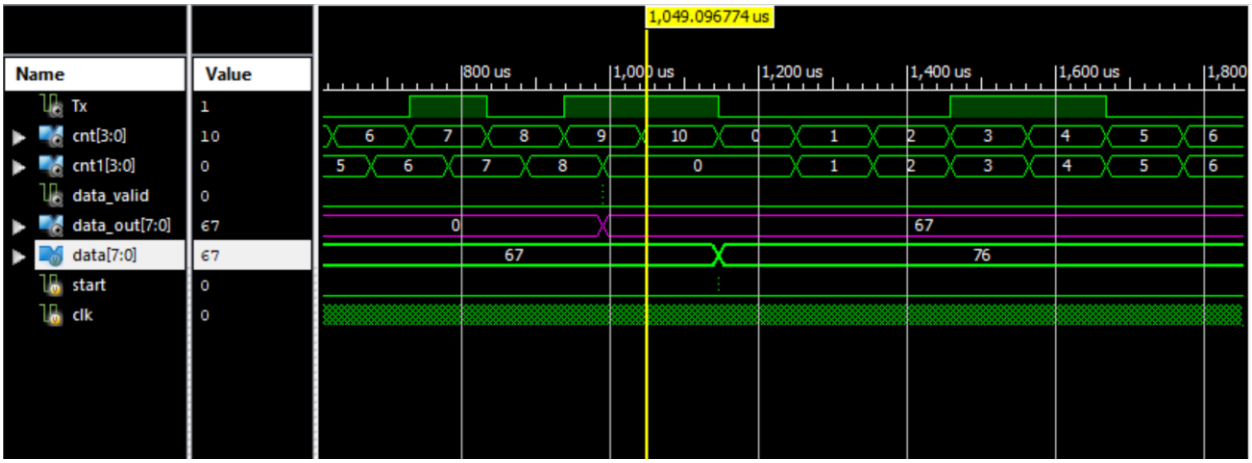
ماژول گیرنده مطابق شکل بالا بصورت سریال داده ارسالی توسط فرستنده را از پورت ورودی Rx دریافت میکند و هنگامی که Rx برابر استارت بیت یعنی 0 بود شمارنده cnt_start فعال میشود و برای اینکه مطمئن شویم ورودی واقعا استارت بیت است و در اثر نویز 0 نشده است باید Rx طی حداقل 1250 کلاک یعنی نصف طول هر بیت ارسالی توسط فرستنده (مطابق قسمت قبل در توضیحات فرستنده ، دیدیم که طول هر بیت با توجه به کلاک و باد ریت برابر 2500 است) برابر 0 صفر باشد . پس اگر ورودی صفر بماند یعنی استارت بیت امده است و هنگامی که cnt_start به 1250 رسید باید ماژول گیرنده فعال شود و به فاصله هر 2500 کلاک از ورودی نمونه برداری کند پس متغیر cnt_sample را برای اینکه بعد از 2500 کلاک ورودی را دریافت کنیم ، تعریف میکنیم و تعداد بیت های دریافتی از فریم uart را با شمارنده cnt نمایش میدهیم تا در خروجی به کاربر نشان بدهیم چند بیت از 10بیت فریم دریافت شده است . برای اینکه ورودی Rx را مرتب کنیم و با رعایت ارزش های هر بیت به کاربر تحویل دهیم از یک بافر بنام register استفاده میکنیم و چون طبق پروتکل uart داده ها از استارت بیت تا استاپ بیت از کم ارزش به پر ارزش دریافت میشوند باید برای این بافر Rx از سمت چپ به این بافر وارد شود و با شیفت یافتن این بافر را پر کند، بصورت زیر :

```
register <= {Rx,register[7:1]}
```

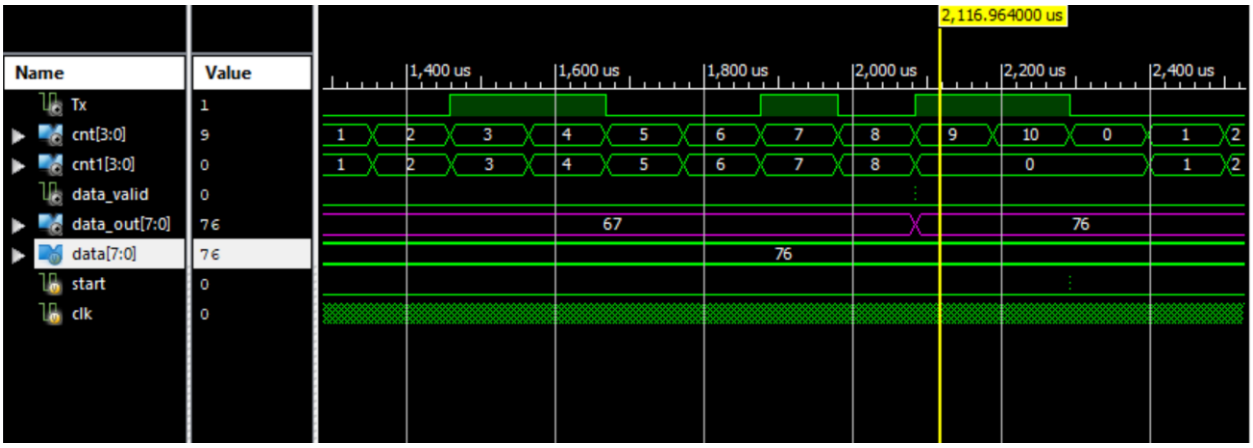
هنگامی که هر 8 بیت داده دریافت شد و استاپ بیت که یک است طی 2500 کلاک دریافت شد ، ماژول گیرنده بصورت کامل دیتا را دریافت کرده است و محتویات register را میتوانیم بعنوان خروجی در پورت data_out به کاربر تحویل دهیم و برای اینکه به کاربر نشان دهیم چه زمانی خروجی معتبر است و باید دیتا را از پورت خروجی بردارد از data_valid استفاده میکنیم و برای متوقف کردن ماژول گیرنده start_flag را 0 میکنیم و بافر register را پاک میکنیم .

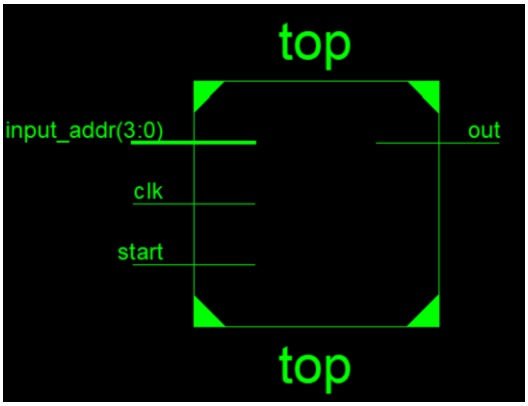
نتایج شبیه سازی و waveform ها

اولین دیتا ورودی به ماژول فرستنده 67 است که میبینیم در گیرنده به درستی دریافت شده است و گیرنده 67 را روی پورت خروجی اش در data_out قرار داده است



دومین دیتا ورودی به ماژول فرستنده 76 است که میبینیم در گیرنده به درستی دریافت شده است و گیرنده 76را روی پورت خروجی اش در data_out قرار داده است





در قسمت دوم آزمایش میخواهیم داده های موجود در خانه های یک رام را بوسیله ماژول transmitter ارسال کنیم و با دریافت شان توسط ماژول receiver انها را در یک رم بنویسیم . و در پایان با دریافت یک ادرس از کاربر خانه های رام و رم با این ادرس را مقایسه کنیم و در صورت برابر بودن خروجی out را یک کنیم

در ابتدا خانه های رام را بصورت زیر مقداردهی اولیه میکنیم

memory_initialization_radix=10;

memory_initialization_vector=63,48,91,79,102,109,125,7;

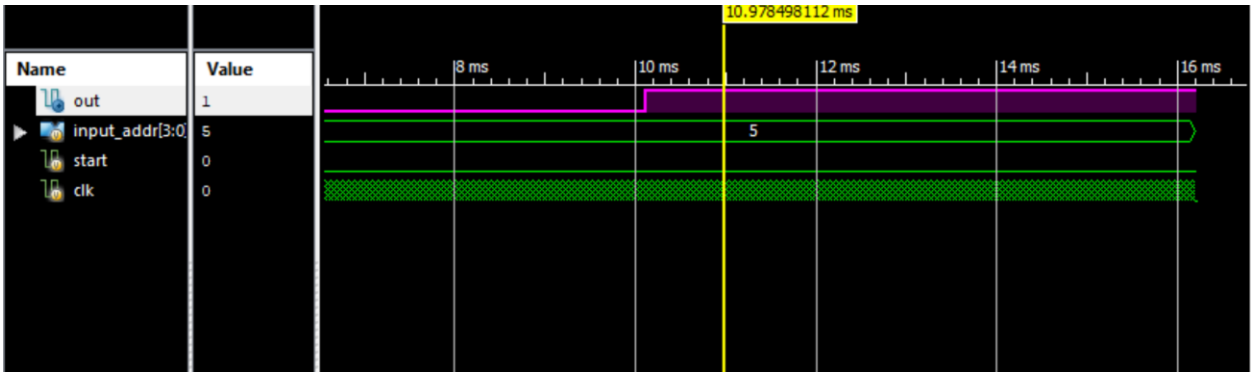
سپس در ماژول top اتصالات لازم بین دو ماژول transmitter و receiver را برقرار میکنیم . برای اینکه در زمان های مناسب ماژول فرستنده را فعال کنیم تا از رام اطلاعات را دریافت و شروع به ارسال کند، از متغیر start_trans استفاده میکنیم که این متغیر هر بار که ارسال یک داده از خانه های رام را انجام داد و به خانه ی بعدی رفت ، فعال میشود . از انجایی که طبق توضیحات ماژول فرستنده ، پایان ارسال با 11 شدن شمارنده خروجی فرستنده مشخص میشود ، هر بار که cnt_trans برابر 11 شد و هنوز به اخرین خانه رام نرسیده بودیم یعنی (addrm1<8) بود باید start_trans فعال شود و یک واحد به پایه ادرس رام یعنی addrm1 اضافه شود تا ارسال داده ی موجود در خانه ی بعدی اغاز شود

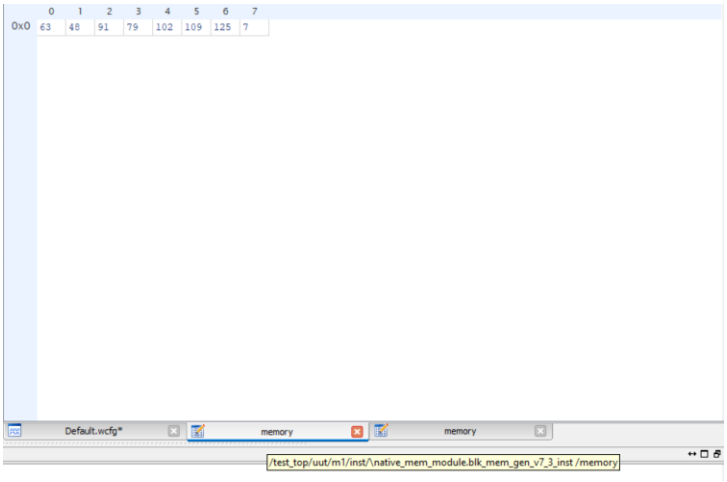
در گیرنده دریافت صحیح بصورت خودکار با اتصال Tx به Rx انجام میشود اما برای تنظیم ادرس رم باید پس از پایان پروسه دریافت در ماژول گیرنده که با 10 شدن cnt_rec مشخص میشود یک واحد به پایه ادرس رم یعنی addrm2 اضافه میکنیم و هنگامی که به اخرین خانه رسیدیم پایه نوشتن رم را wem2 صفر کنیم

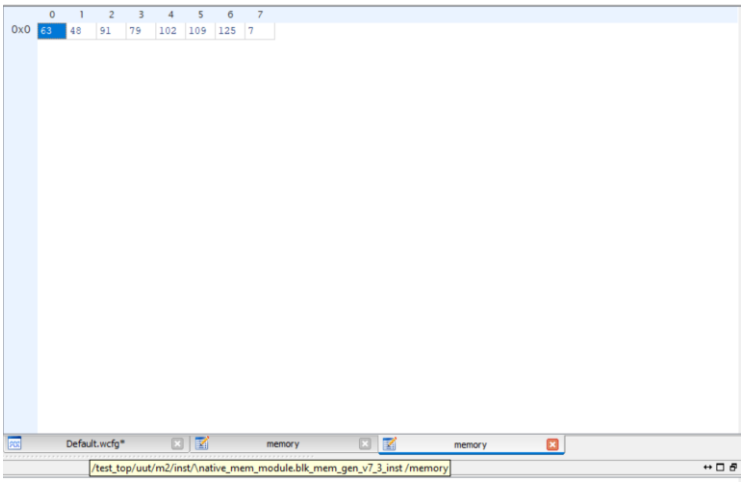
زمانی که wem2 برابر 0 شد یعنی تمامی خانه های رام به رم ارسال شده و مقدار دهی رم خاتمه یافته است ، پس میتوانیم ادرس کاربر را دریافت کنیم با دادن این ادرس به خانه های رم و رام خروجی را مقایسه و out را مقدار دهی کنیم

توجه شود که برای درست بودن کد نوشته شده برای اولین خانه رم بجای اینکه پایه ادرس رم را برابر addrm2 قرار دهیم برابر addrm2 – 1 نوشتیم بنابراین باید ادرس ریافتی از کاربر را یک واحد زیاده کنیم و به پایه ادرس رم بدهیم ، همچنین شرط های مربوط به رم بجای 8> بودن 8=> قرار گرفتند

نتایج شبیه سازی و waveform ها







مطابق تصویر بالا مشاهده میشود محتوای رام و رم یکسان شده است و خروجی out برای خانه 6 ام یعنی ادرس 5 برابر 1 است که یعنی این دو خانه از دو حافظه با هم محتویات یکسانی دارند

در قسمت اخر ازمایش برای استفاده از بلوک های تفاضلی اتصالات را به حالت زیردر تست بنچی که فرستنده و گیرنده بهم مرتبط شده اند یعنی تست بنچ test_transmitterandreceiver، برقرار میکنیم

```
OBUFDS o1(
  .I(Tx),
  .O(oTx),
  .OB(ob)
      );
IBUFDS o2(
  .I(oTx),
  .IB(ob),
  .O(Rx)
      );
```

از انجایی که Tx از برد fpga خارج میشود تا به گیرنده برود ورودی بلوک OBUFDS است و خروجی این بلوک که tx و نات ان است را به بلوک IBUFDS که گیرنده است میدهیم و خروجی این بلوک را به عنوان Rx به گیرنده میدهیم و مشاهده میکنیم که عملیات ارسال و دریافت بدرستی انجام میشود