به نام خدا

تکلیف سری سوم fpga

محدثه غفوری(9632133)

سوال 1) الف: برای حل این سوال دو رویکرد وجود دارد

1) به مدت یک ثانیه صبر کنیم و تعداد پالس هایی که در این مدت توسط سنسور دریافت میشود را شمارش کنیم و آنرا تقسیم بر دو کنیم زیرا با توجه به سوال هر دو پالس امده نماینده یک دور موتور است . اما این رویکرد مناسبی نیست زیرا برای اعلام دور بر ثانیه موتور هر بار باید یک ثانیه صبر کنیم و نمیتوان تغییرات دور موتور را که در زمان هایی زیر یک ثانیه اتفاق می افتد را گزارش کنیم
2) در این رویکرد با دریافت هر پالس بوسیله ی یک کانتر مدت زمان یک بودن پالس را بر اساس کلاک مرجع ورودی یعنی **MHz40** اندازه گیری میکنیم . باتوجه به اینکه **duty cycle** پالس های ورودی **50** درصد است همین مدت زمان هم پالس صفر است اما با توجه به آنکه باید دو پالس برای یک دور موتور در نظر گرفته شود داریم

T=25ns , cnt= the number of clock edge when pulse is 1
t1=the duration of being 1 = T*cnt
t0= the duration of being 0 = T*cnt
duration of one pulse = 2*T*cnt
duration of two pulse which shows 1 cycle of motor =2*2*T*cnt
1 cycle ➔ 4*T*cnt (s)
? ➔ 1 (s)
?= 1 / (4*T*cnt) cycle/second
Out = 1 / (4*25ns *counter) = 10^ 7 / counter

با توجه به توضیحات بالا برای انکه بتوانیم سریع تر دور بر ثانیه موتور را گزارش کنیم تنها به مشاهده و اندازه گیری زمان یک بودن یک پالس نیاز داریم زیرا در زمان صفر شدن پالس کانتری که میزان زمان یک بودن را محاسبه کرده است تحت عملیات قرار میگیرد تا خروجی و تعداد دور بر ثانیه را به ما بدهد
در اینجا علاوه بر کانتر **cnt** از یک کانتر کمکی **counter** نیز استفاده کردیم تا محاسبات را بوسیله ی ان انجام دهیم زیرا **cnt** با صفر شدن پالس باید صفر و پاک شود تا بتواند هنگام دیدن پالس بعدی مدت زمان یک بودن (تعداد لبه ی بالارونده ی کلاک مرجع) را بدرستی و مستقل از پالس قبلی اندازه گیری کند
توجه شود که اگر **reset** ماژول فعال شود خروجی و تمامی رجیستر های درگیر برای نمایش خروجی باید صفر و پاک شوند . در این حالت اگر بعد از فعال شدن ریست زمانی ریست دوباره **0** شود که پالس صفر باشد برای انکه **counter** صفر شده است و خروجی بی نهایت تولید میشود با گذاشتن یک شرط به برنامه میگوییم تا امدن لبه ی بالارونده ی پالس و یک شدن ان خروجی را صفر نمایش دهد و مانع از بی نهایت شدن خروجی میشویم

ب : دور موتور در حالت اول **1000 rpm** و یا **16 ~ 50/3 = 1000/60** است (چون محاسبات ماژول بصورت صحیح انجام میشود از قسمت اعشار دور بر ثانیه موتور صرف نظر میشود و انرا دور میریزد)

50 / 3 = out = 10 ^ 7 / counter ➔ counter = 6* 10^5

t= 6 * 100000 * 25ns = 15 ms مدت زمان یک یا صفر بودن یک پالس

1 (s) / 15 ms ~ 66

برای تولید پالس مربوط به این دور خاص موتور (1000 rpm) به اندازه یک ثانیه باید 66 تا 15 میلی ثانیه صفر و یک (پالس را تغییر سطح بدهیم) تولید کنیم یا به عبارتی 33 تا پالس با دوره تناوب 30 ms تولید کنیم

دور موتور در حالت دوم 2000 rpm یا 2000/60 ~ 100/3 است

100 / 3 = out = 10 ^ 7 / counter ➔ counter = 3* 10^5

t= 3 * 100000 * 25ns = 7.5 ms مدت زمان یک یا صفر بودن یک پالس

1 (s) / 7.5 ms ~133

برای تولید پالس مربوط به این دور خاص موتور (2000 rpm) به اندازه یک ثانیه باید133 تا 7.5 میلی ثانیه صفر و یک (پالس را تغییر سطح بدهیم) تولید کنیم یا به عبارتی 66 تا پالس با دوره تناوب15 ms تولید کنیم بنابراین مقدار 133 به 66 تای قبلی در حلقه ی for تست بنچ اضافه میشود تا این پالس را نیز تولید کند

دور موتور در حالت سوم 3000 rpm یا 3000/60=50 است

50 = out = 10 ^ 7 / counter ➔ counter = 2* 10^5

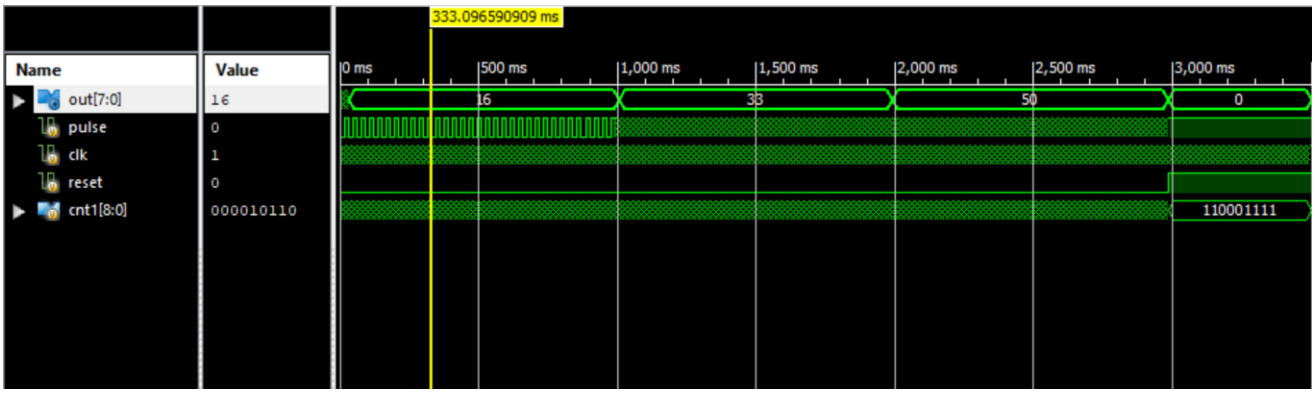
t= 2 * 100000 * 25ns = 5ms مدت زمان یک یا صفر بودن یک پالس

1 (s) / 5 ms = 200

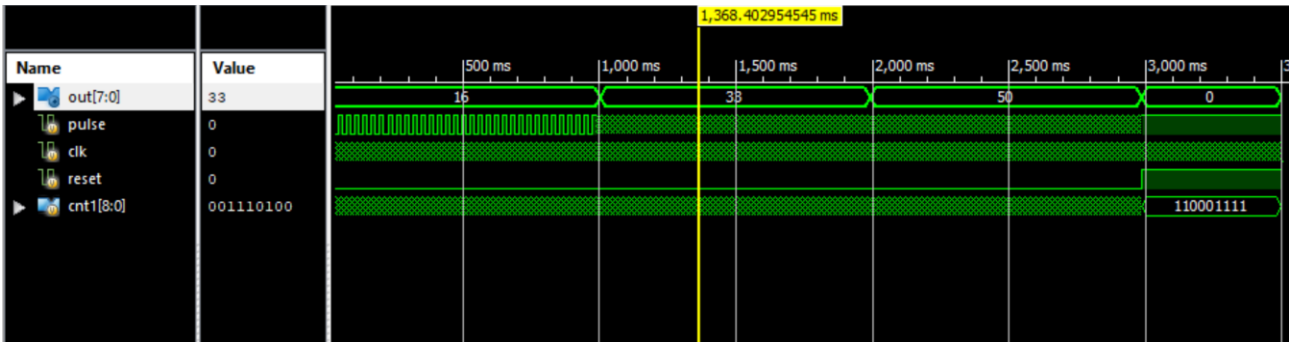
برای تولید پالس مربوط به این دور خاص موتور (3000 rpm) به اندازه یک ثانیه باید200 تا 5 میلی ثانیه صفر و یک (پالس را تغییر سطح بدهیم) تولید کنیم یا به عبارتی 100 تا پالس با دوره تناوب10 ms تولید کنیم بنابراین مقدار 200 به133+ 66 تای قبلی در حلقه ی for تست بنچ اضافه میشود تا این پالس را نیز تولید کند

نتایج شبیه سازی و waveform ها

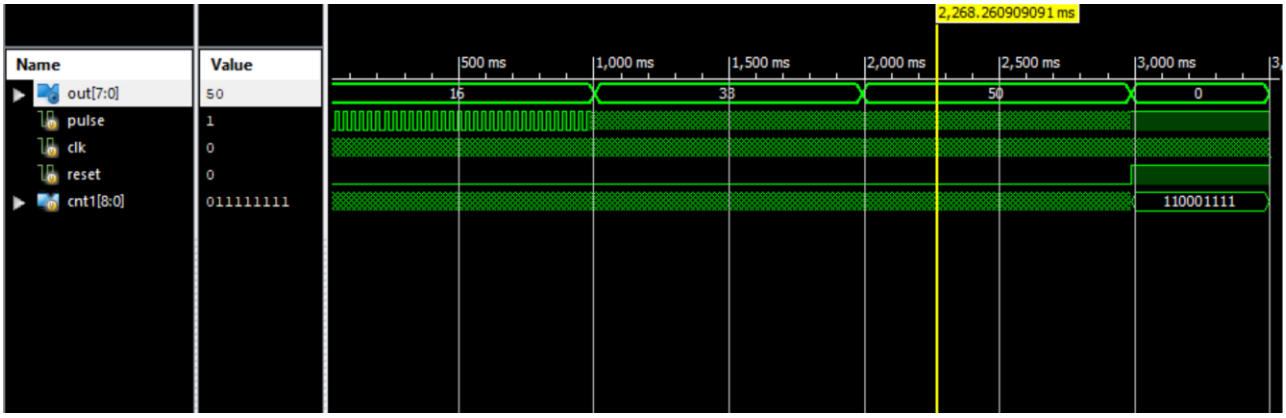
حالت 1000 دور بر دقیقه



حالت 2000 دور بر دقیقه

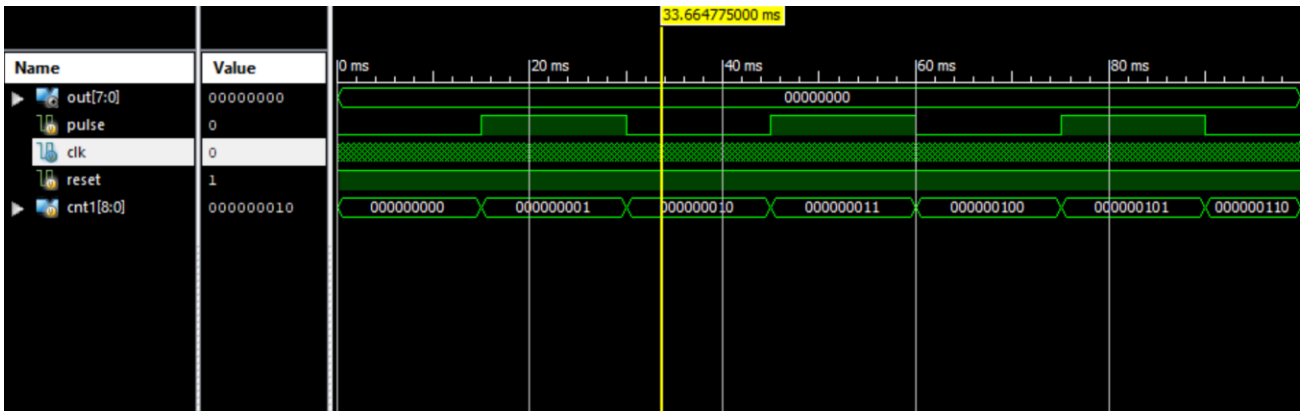


حالت 3000 دور بر دقیقه

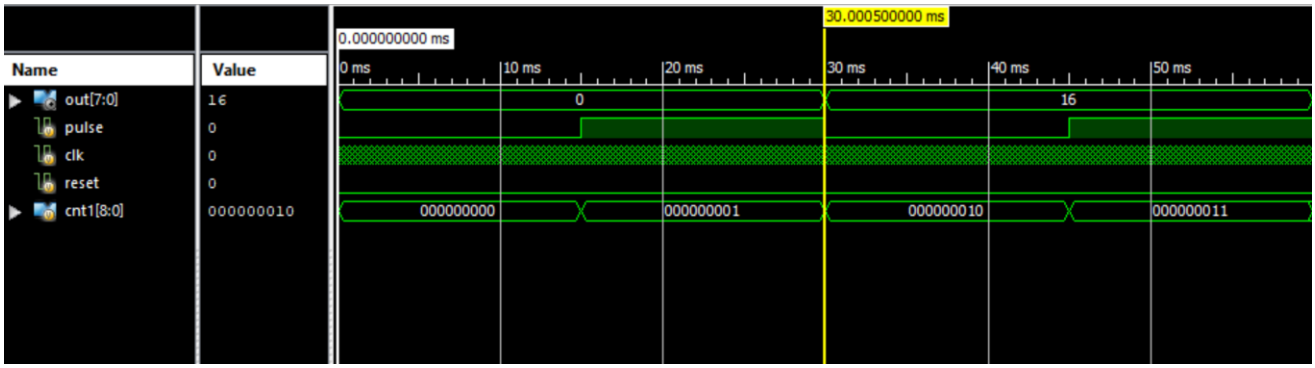


حالت reset فعال

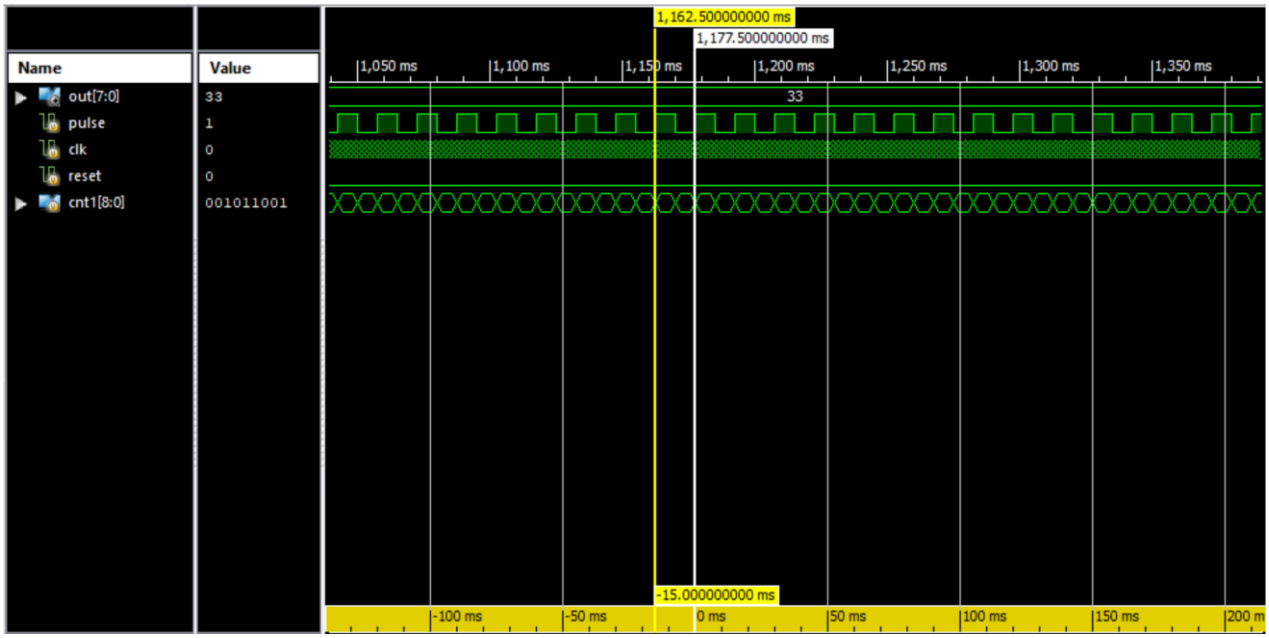
با وجود تولید پالس خروجی صفر میشود



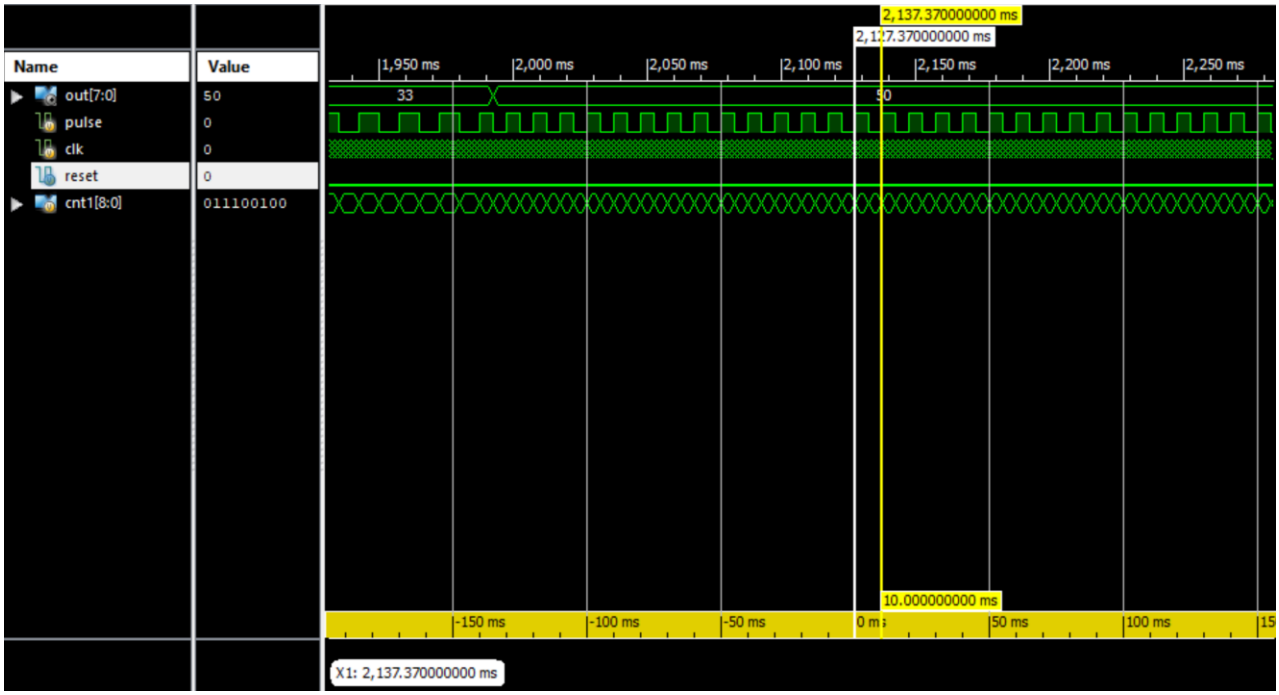
تولید پالس اول با دوره تناوب 30 میلی ثانیه



تولید پالس دوم با دوره تناوب 15 میلی ثانیه



تولید پالس سوم با دوره تناوب 10 میلی ثانیه

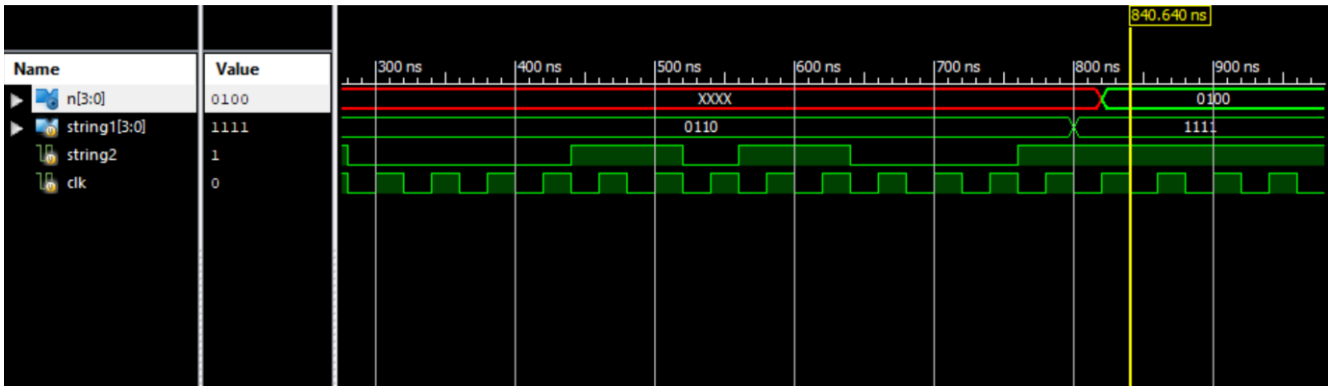


سوال 2) الف : در این حالت با در نظر گرفتن همپوشانی چون string1 چهار بیتی است بنابراین ورودی را تک به تک دریافت کرده و در یک بافر بطول 4 بیت (بنام register) قرار میدهیم تا برای مقایسه با string1 از آن استفاده کنیم . حال با استفاده از یک cnt گرفتن ورودی را در لبه ی بالارونده کلاک کنترل میکنیم و چون هر frame ورودی 20 بیت طول دارد این کانتر باید برای شمارش این مقدار 5 بیتی باشد .

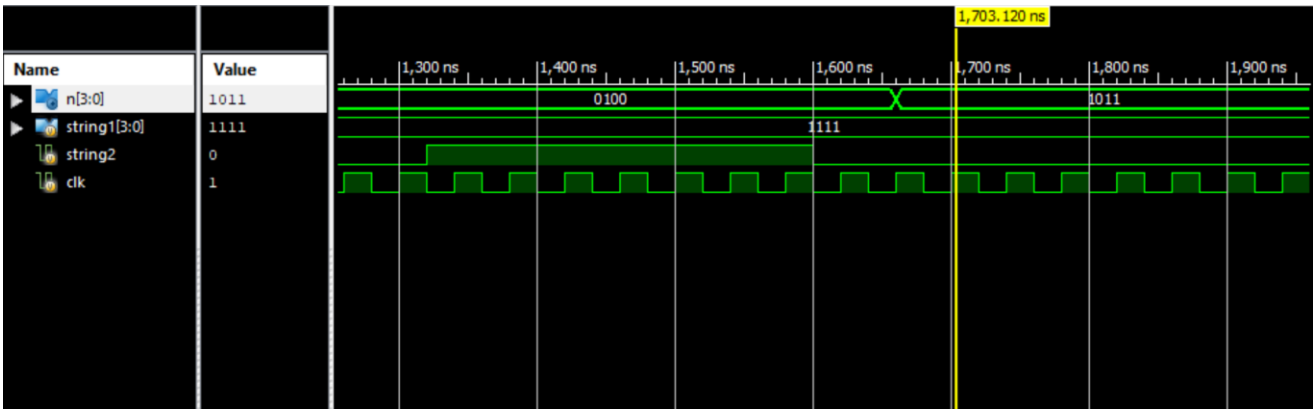
برای انکه بتوانیم همپوشانی ها را هم اثر دهیم باید برای هر ورودی در کنار ورودی های قبلی مقایسه با string1 را انجام دهیم بدین منظور ورودی را با register [3:1] که بافر مورد نظر است concatenate میکنیم و داخل همین بافر قرار میدهیم تا بتوانیم یکی یکی روی frame ورودی حرکت کنیم و درصورتی که با دنباله موردنظر برابر بود a یک عدد افزایش میابد سپس پس از دریافت یک frame بیست بیتی خروجی را نمایش میدهیم و برای اماده شدن برای دریافت بیست بیت دوم بافر و کانتر را پاک میکنیم و خروجی را نمایش میدهیم . از آنجایی که برای شمارش string1 جدید روی ورودی جدید باید مقدار a از صفر شروع شود چون دستورات non blocking هستند میتوان مقدار a را نیز قبل شروع دریافت ورودی بعدی صفر کرد زیرا میدانیم با وجود همزمانی دستورات مقدار قبلی a در خروجی یعنی n قرار داده میشود و پس از لبه ی کلاک مقدار a صفر میشود

نتایج شبیه سازی و waveform ها

Frame ورودی = 10001101100001100110 , string1= 0110



Frame ورودی دوم =01111111001111111111 و string1=1111



توجه : برای مشاهده هر دو ورودی و نتایج ان زمان شبیه سازی باید بیش از 1 میکرو ثانیه باشد در حدود 2 میکرو ثانیه

ب: در این قسمت میخواهیم بدون در نظر گرفتن همپوشانی ها تعداد string1 را در یک frame بیست بیتی از string2 شمارش کنیم بدین منظور یک counter تعریف میکنیم با شروع این کانتر از صفر تا 3 صبر میکنیم تا در طی چهار کلاک متغیر register پر شود در واقع ورودی string2 را تا 4 بیت به این رجیستر میدهیم سپس هنگامی که بیت چهارم دریافت شد counter برابر 4 میشود و حال میتوان register را با string1 مقایسه کرد . درصورتی که برابر باشند به متغیر a که این تعداد را میشمارد و انرا در خود نگه میدارد یک واحد اضافه میشود و از انجایی که بدون هم پوشانی برسی میکنیم این قسمت از frame بیست بیتی باید کنار گذاشته شود و دوباره وارد مقایسه نشود(و از ادامه ی این قسمت دوباره برسی ها انجام شود) متغیر register برابر 0 میشود تا قسمت های

بعدی را بگیرد و ادامه ی مقایسه را انجام دهد اما اگر با 4 شدن counter دو متغیر string1 , register برابر نبودند باید یک بیت ورودی دیگر از string2 وارد شود و دوباره عملیات مقایسه انجام شود یعنی در این حالت تا هنگامی که string1 , register برابر نشدند ما به گرفتن ورودی ادامه میدهیم تا هنگامی که بیست بیت دریافت شود و یا برابری اتفاق بیفتد

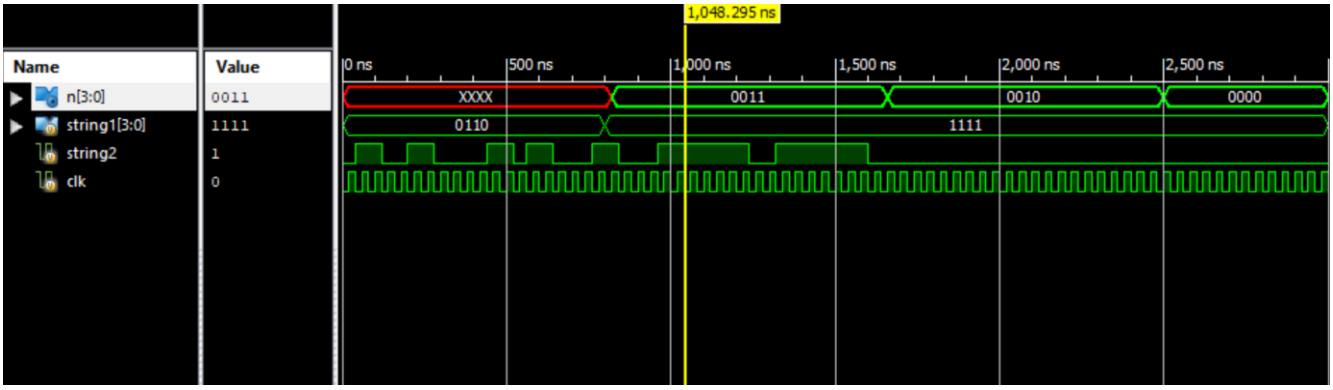
هنگامی که هر بیست بیت frame دریافت شد متغیر a را که تعداد برابری ها را شمارش میکرد در خروجی قرار میدهیم و register را high impedance میکنیم تا تداخلی با شمارش روی frame بعدی رخ ندهد

متغیر cnt کنترل میکند که بیست بیت در بیست کلاک وارد شوند و بعد از دریافت کل بیت ها طی یک کلاک اضافی خروجی را دریافت و متغیر ها برای گرفتن بیست بیت بعدی پاک شوند

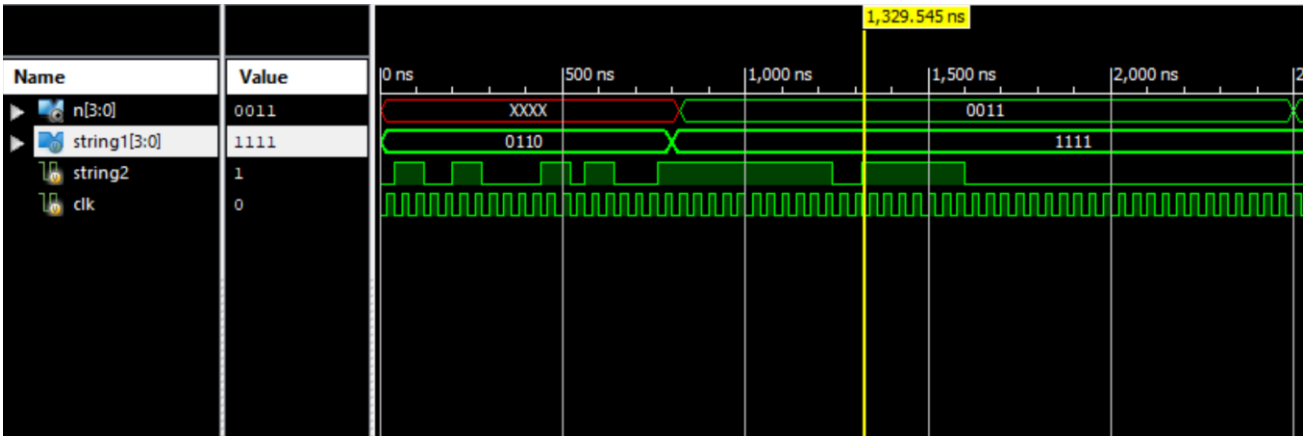
از آنجایی که دستورات این الگوریتم از نوع blocking هستند و در همان لبه متغیر ها update میشود هنگامی که counter برابر 3 است وارد شرط که میشویم یک ورودی دریافت میکنیم و در همان لبه counter برابر 4 میشود و شرط counter ==4 برقرار میشود و وارد عملیات مقایسه میشود و اگر string1 , register برابر نباشند در همان لبه که یک ورودی دریافت کرده بودیم در statement این شرط نیز یک ورودی دریافت میکنیم در واقع بجای دریافت هر ورودی در هر لبه در واقع در یک لبه کلاک دو ورودی دریافت میشود و عملیات مقایسه را مختل میکند زیرا باید با دریافت هر ورودی مقایسه انجام گیرد و درصورت برابری کل ان قسمت از بیست بیت دور ریخته شوند یا ورودی بعدی گرفته شود با دریافت دو ورودی در یک لبه نمیتوان بصورت صحیح روی این بیست بیت حرکت کرد و شمارش را انجام داد بنابراین یک متغیر دیگر cnt2 تعریف میکنیم تا چک کند اگر در یک کلاک ورودی دریافت شد دیگر در ان کلاک ورودی دریافت نشود و بعد از انجام عمل مقایسه در کلاک های بعدی ورودی دریافت شود برای این منظور cnt2 را در هر کلاک با cnt برابر قرار میدهیم و اگر ورودی دریافت شد cnt2 را یک واحد افزایش میدهیم که با cnt برابر نباشد و در ان کلاک ورودی دیگری دریافت نشود

نتایج شبیه سازی و waveform ها

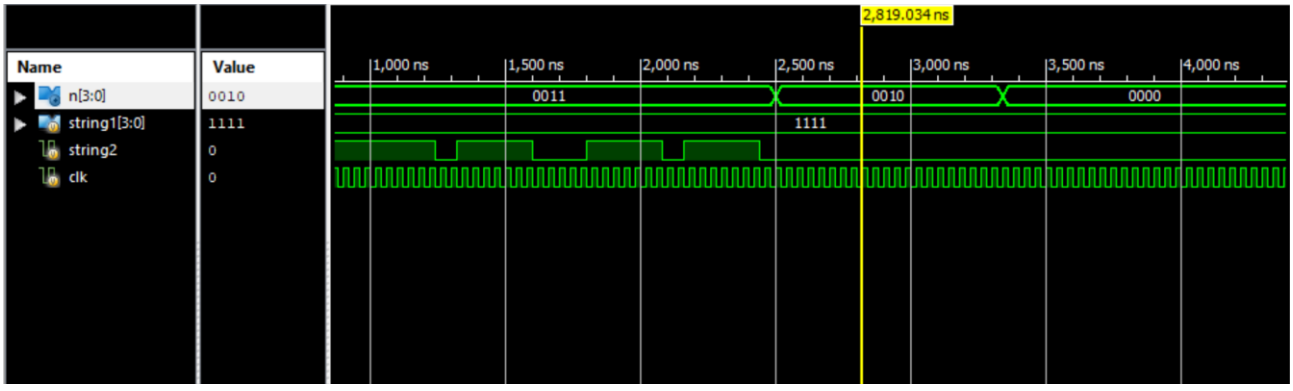
Frame ورودی = 10001101100001100110 , string1= 0110



Frame ورودی دوم =01111111001111111111 و string1=1111



Frame ورودی سوم =01111111001111111000 و string1=1111



سوال 3) مطابق با بلوک دیاگرام سوال ابتدا ورودی اول frame چهار بیتی با shiftreg[4] عملیات xor میشود سپس این حاصل وارد shiftreg[0] و وارد shiftreg[1] میشود و مقدار shiftreg[1] با حاصل اولین xor عملیات xorمیشود سپس وارد shiftreg[2] و shiftreg[3] و shiftreg[4] میشود

این عملیات تا جایی که تمامی بیت های هر frame چهار بیتی وارد بلوک سوال شوند و عملیات های مورد نظر روی انها انجام شود ادامه میابد

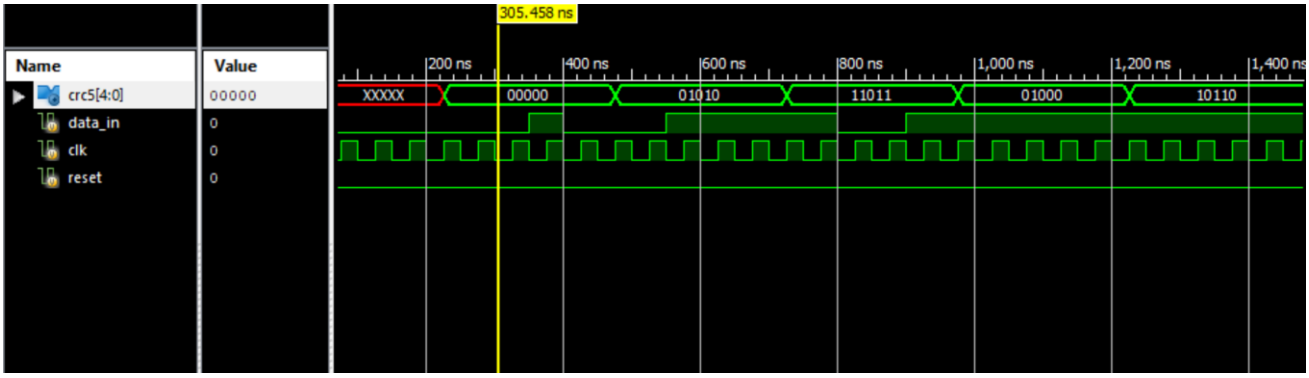
در نهایت هنگامی که هر چهار ورودی اعمال شد(طی 4 کلاک چهار بیت ورودی دریافت شده و در هر کلاک این بلوک کاملا اجرا شده و مقدار shiftreg[4] بروزرسانی میشود) و این عملیات فیدبک دار روی همه بیت ها انجام شد خروجی اماده ی نمایش است حال با نمایش خروجی باید اماده ی دریافت چهار بیت بعدی شویم بنابراین باید تمامی

شیفت رجیستر های بلوک دیاگرام سوال پاک و صفر شوند برای این عملیات دو روش وجود دارد :

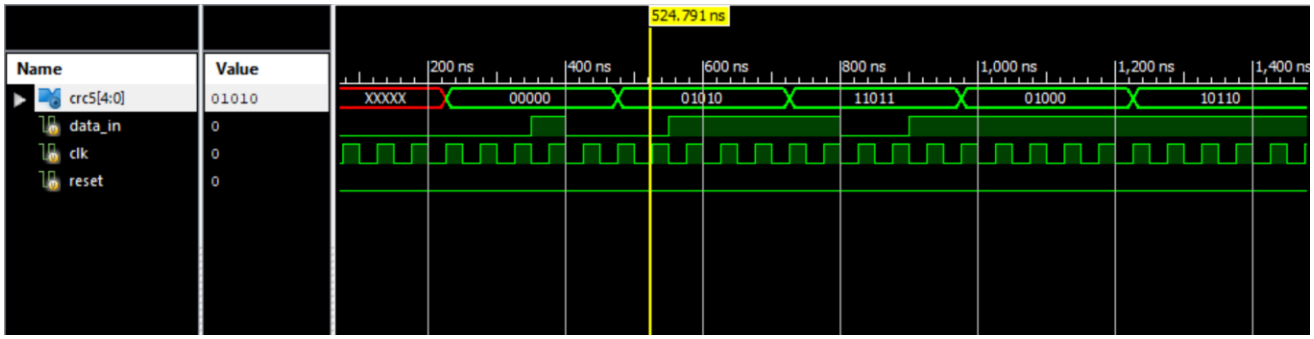
1) بعد از ورود بیت چهارم هر frame به طول shiftreg که 5 بیت طول دارد 0 وارد بلوک دیاگرام سوال کنیم تا کاملا پاک شوند و اماده ی دریافت frame بعدی شود
2) در لبه ی پایین رونده ای که بیت چهارم یعنی بیت اخر هر frame را وارد بلوک دیاگرام میکنیم در لبه ی بالا رونده ی بعدی عملیات تولید کد crc با اخرین بیت انجام میشود و در لبه ی بالا رونده ی بعد از آن خروجی را از shiftreg میخوانیم و در این هنگام کل shiftreg را صفر میکنیم و تک تک بیت های انرا پاک و صفر میکنیم تا اماده ی دریافت frame بعدی شود ! از آنجایی که هنگامی که اخرین ورودی هر فریم را اعمال میکنیم (لبه ی پایین رونده کلاک) تا دریافت خروجی و پاک شدن رجیستر بلوک دیاگرام برای اماده شدن گرفتن frame بعدی به اندازه ی دو لبه ی بالا رونده (یکی انجام عملیات روی بیت اخر هر frame و یکی دریافت خروجی و پاک کردن shiftreg) باید صبر کنیم یعنی دو کلاک باید زمان بدهیم تا frame بعدی برا بتوانیم اعمال کنیم از آنجایی که کلاک در نظر گرفته شده در تست بنچ 20MHz (دوره تناوب 50ns) است زمان دو کلاک برابر 100ns است بنابراین بین اعمال هر frame ورودی خواسته شده #100 تاخیر اعمال میشود تا خروجی ها صحیح و معتبر باشند و با قبلی ها تداخل پیدا نکنند . و از این طریق از صرف تعداد کلاک و زمان بیشتر با دادن 5 کلاک تاخیر به بلوک یا وارد کردن 5 تا 0 به بلوک برای پاک شدن shiftreg خودداری کرده ایم

نتایج شبیه سازی و waveform ها

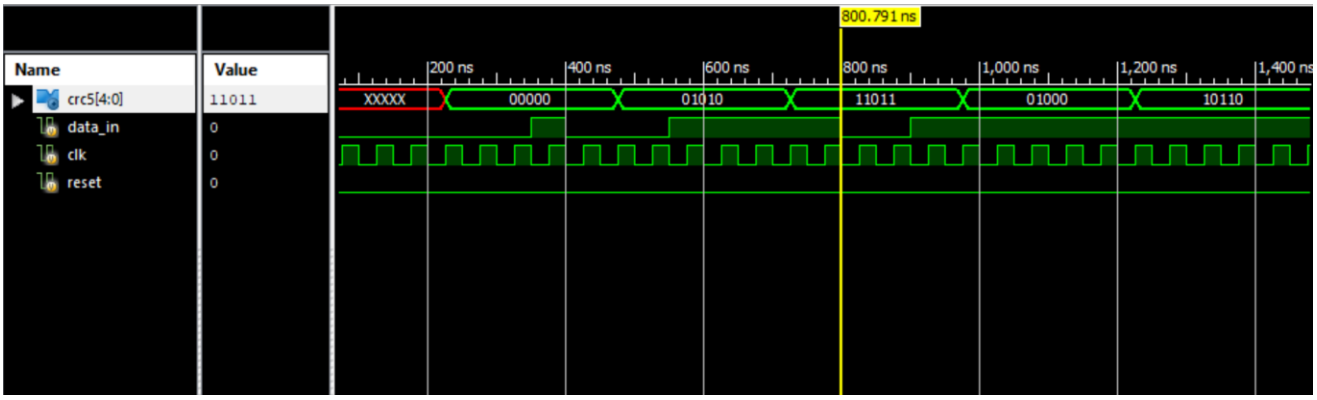
''0000''



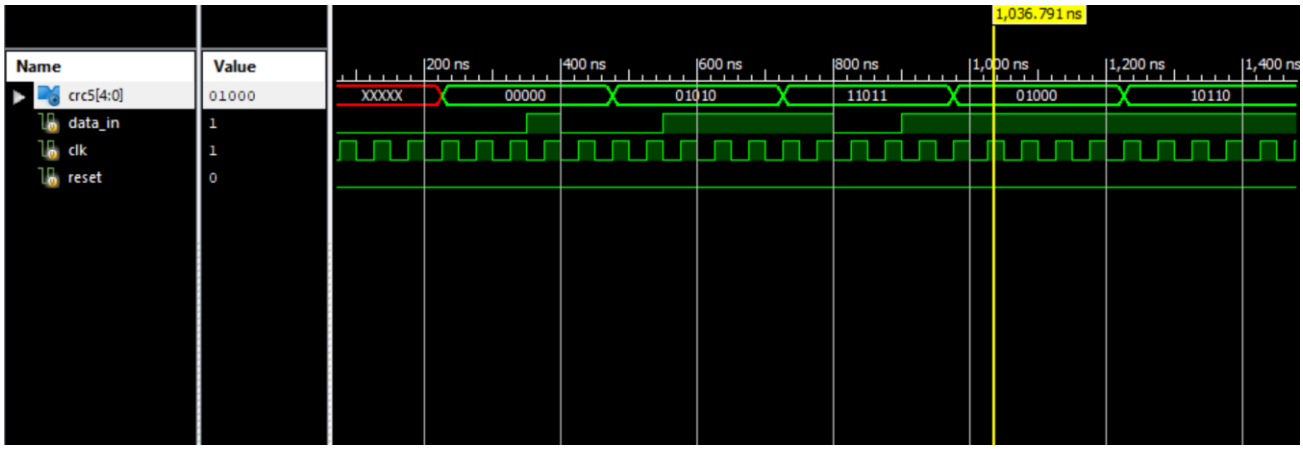
''0010''



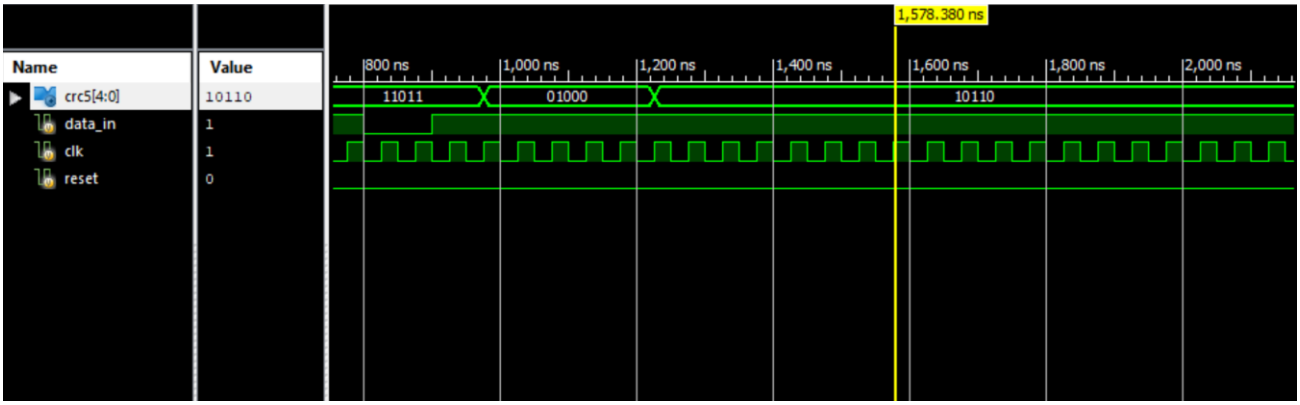
''0111''



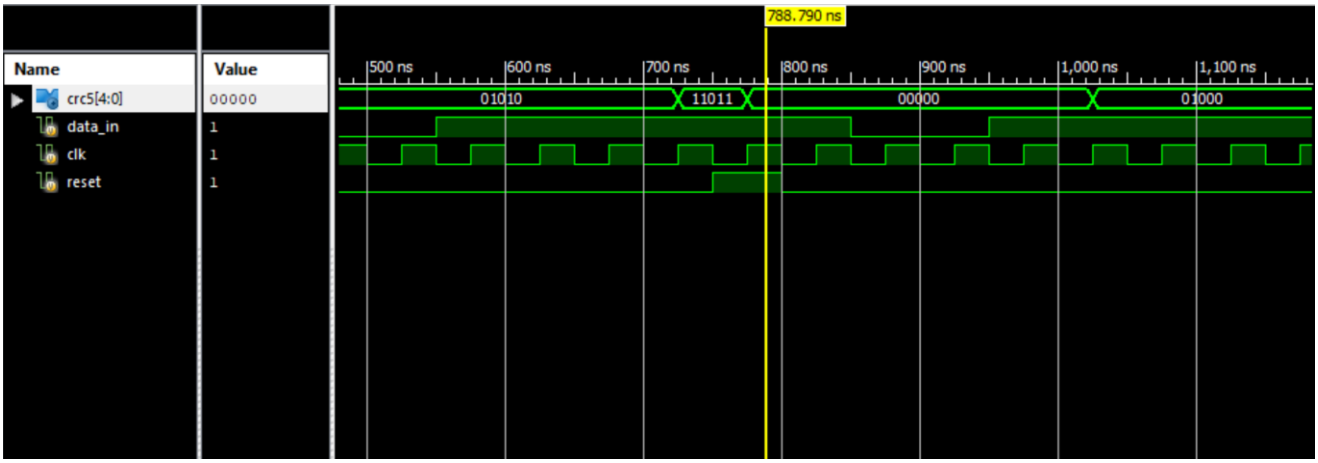
''1001''



" 1111 "



Reset=1



سوال 4) پروتکل UART مخفف Universal Asynchronous Receiver And Transmitter هنگام ارسال دیتا علاوه بر دیتا تعدادی بیت کنترلی نیز با آن ارسال می‌شود که به این مجموعه اصطلاحاً یک فریم frame گفته می‌شود

بیت شروع START : در وضعیتی که ارسال و دریافت صورت نمی‌گیرد خط انتقال در حالت یک منطقی است. با ایجاد یک لبه پایین رونده توسط فرستنده ، گیرنده از فرستاده شدن اطلاعات آگاه شده و آماده دریافت می‌شود بنابراین بیت شروع صفر منطقی است

بیت های داده DATA : بیت‌های داده اطلاعات اصلی را منتقل می‌کند و می‌تواند متغیر باشد. تعداد این بیت‌ها بباید در فرستنده و گیرنده به صورت یکسان تنظیم شود که در این سوال 8 بیت است

بیت پایان STOP : در انتهای داده برای اعلام اتمام دریافت داده قرار میشود و چون خط انتقال در حالتی که ارسال و دریافت ندارد یک منطقی است بیت پایان نیز یک منطقی خواهد بود

در این سوال بعلت مشخص نکردن BAUD RATE نمونه برداری از داده ها همزمان با لبه ی بالا رونده ی کلاک وارد شده به ماژول انجام میگیرد و بیت توازن هم نداریم

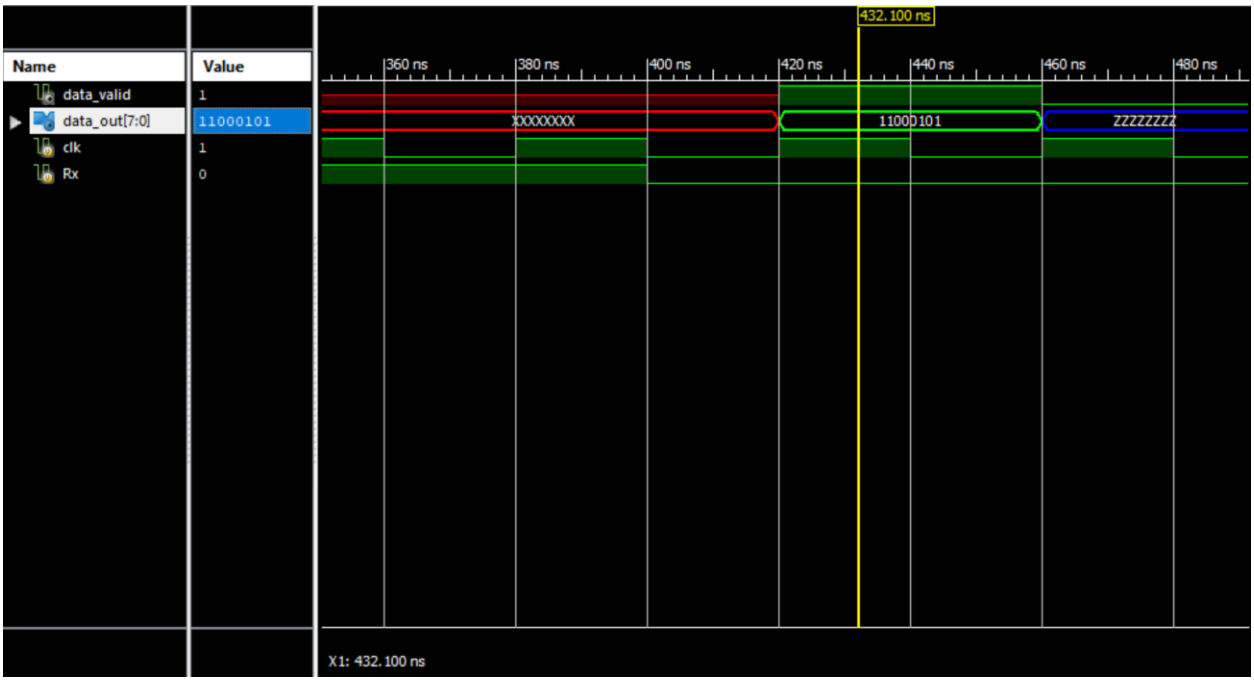
بنابر توضیحات بالا frame این سوال ده بیت طول دارد بیت msb آن یک منطقی و lsb آن صفر منطقی است و همچنین داده ها به ترتیب از کم ارزش به پر ارزش دریافت میشوند بنابراین در یک مجموعه دیتا ده بیتی هر گاه پرارزش ترین بیت یک باشد و کم ارزش ترین بیت صفر باشد 8 بیت وسط دیتای ارسال شده تحت پروتکل uart خواهد بود

بر این اساس یک حافظه به طول ده بیت بنام register در نظر میگیریم که بعلت انکه وضعیت عادی خط که ارسال و دریافتی نباشد یک منطقی است مقدار اولیه این حافظه ده بیتی 10'b1111111111 خواهد بود و با concatenate کردن حافظه با ورودی این حافظه update میشود و زمانی برسی برای گرفتن داده از خط انتقال صورت میگیرد که ورودی اولین بار صفر شود سپس بعد از گرفتن دیتا و یک شدن اخرین بیت frame میتوان هشت بیت وسط را روی خروجی قرار داد و در این هنگام است که خروجی معتبر بوده و data_valid برابر یک میشود

از انجایی که پس از دریافت باید خط انتقال به وضعیت یک منطقی برگردد و گیرنده دوباره اماده ی دریافت frame بعدی شود register نیز باید به مقدار اولیه خود یعنی 10'b1111111111 باز گردد بنابراین بوسیله ی کانتر cnt که با لبه ی بالا رونده ی کلاک ماژول تغییر میکند پس از ده لبه کلاک و گرفتن یک فریم کامل کانتر صفر شده و اماده ی دریافت frame بعدی میشویم و برای مشخص کردن اینکه تا زمان اتمام دریافت و یک شدن data_valid خروجی معتبر نیست انرا high impedance میکنیم و با صفر شدن کانتر منتظر دریافت ورودی بعدی میشویم

نتایج شبیه سازی و waveform ها

دیتا فریم 1110001010



دیتا فریم 1110101110