به نام خدا

تکلیف سری پنجم fpga

محدثه غفوری(9632133)

سوال 1) در قسمت فرستنده پروتکل uart میدانیم فرستنده با امدن دستور شروع ارسال از ورودی شروع به گرفتن داده ی کاربر و ساخت فریم متناسب با این پروتکل میکند بنابراین با یک شدن ورودی TX_send تسک مربوط به تولید پریتی ها calculating_parity(data_frame,in) را صدا میکنیم و خروجی انرا که حاصل پریتی ها و داده ی کاربر به ترتیب گفته شده در صورت سوال است را در یک قالب بین استارت بیت و استاپ بیت قرار میدهیم و شروع به ارسال میکنیم . چون ورودی های مختلف در زمان های مختلف به تسک مورد نظر وارد میشوند برای اینکه برای هر ورودی یک حافظه مجزا در نظر گرفته شود و خروجی های مجزا بدون اثر گرفتن از سایر ورودی ها ساخته شود تسک را از نوع automatic تعریف میکنیم و چون میخواهیم بصورت سریال داده ها را ارسال کنیم انها را در یک رجیستر بنام shift_reg قرار میدهیم و با شیفت دادن ان در هر لبه کلاک shift_reg[0] را به خروجی منتقل میکنیم

در قسمت گیرنده در هر لبه کلاک ورودی های مختلف را دریافت کرده و در یک بافر بنام register ذخیره میکنیم و هر 9 لبه کلاک این بافر را چک میکنیم تا اگر داده ای بین استارت بیت (صفر) و استاپ بیت (یک) بطول هفت بیت است برداشته و عملیات تصحیح خطا را روی ان انجام دهیم . پس از استراخ فریم داده ها و پریتی ها ، انها را در یک متغیر بنام data_false قرار میدهیم و با فراخوانی c(data_false) شماره ی بیت دارای خطا در این فریم را یافته و ان بیت را معکوس میکنیم تا صحیح شود ، حال با توجه به اینکه میدانیم در کدام خانه های این متغیر داده ی کاربر قرار داشته است پس از اصلاح خطا {data_false[6],data_false[5],data_false[4],data_false[2]} را بعنوان خروجی معتبر و اصلاح شده خواهیم داشت

در تست بنچ این سوال باید برای یکی از ماژول های فرستنده و گیرنده تست بنچ بنویسیم و سپس ماژول دیگر را اضافه کنیم و پس از معرفی پورت ها یک متغیر بنام ctrl تعریف کرده و در صورت فعال شدن این متغیر خروجی فرستنده را معکوس کرده (خطا ایجاد میکنیم) و در غیر این صورت خود خروجی فرستنده به گیرنده منتقل میشود بنابراین باید از دستور assign با شرط روی متغیر ctrl استفاده کنیم اما هنگام instance گرفتن از دو ماژول به اروری برخوردم که موفق به رفع ان و ران کردن شبیه سازی نشدم

سوال 2) توجه : در این سوال فرض را بر این گذاشته ام که در دو حالت دمای بالا و خطر مرگبار بعلت نبود push button برای این قسمت ها ، بعد از پایین امدن دما یا رفع خطر سنسور ها بطور خودکار غیر فعال شده و اتومات ریست

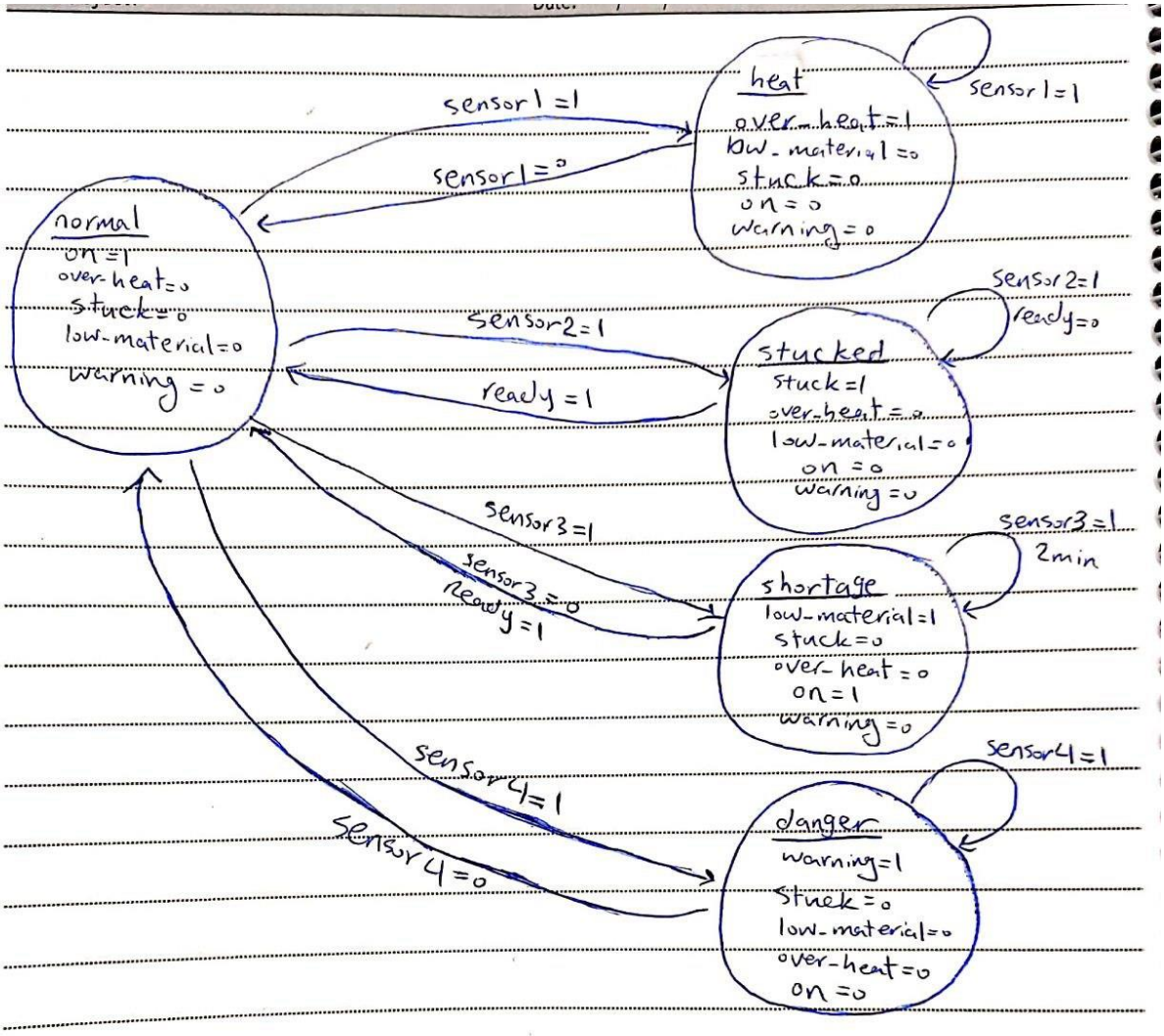
شده و دستگاه به حالت عادی خود باز میگردد بنابراین در اصل با غیر فعال شدن سنسور ها به حالت عادی و نرمال بازمیگردیم و در دو مورد دیگر کافی است با فشار دادن دکمه مورد نظر از ان حالت خارج شده و به وضعیت عادی بازمیگردیم

بنابر فرض سوال در هر زمان فقد یک سنسور میتواند فعال باشد و بعد از رفع مشکل ان سنسور و بازگشت به حالت عادی سنسور های دیگر میتوانند فعال شوند همچنین هنگام کمبود مواد اولیه و خطای مرگبار با توجه به خاموش شدن خط تولید خروجی on برابر صفر میشود همچنین برای دو حالت دیگر بعلت خاموش شدن دستگاه یا راه اندازی مجدد ان که در سوال اشاره شده است ، در این حالات نیز on برابر صفر میشود تا اثر خاموش شدن یا راه اندازی مجدد دستگاه مشخص شود بنابراین با فعال شدن هر یک از سنسور ها on= 0 میشود ، بجز هنگام کمبود مواد اولیه که به مدت 2 دقیقه دستگاه صبر میکند و بعد on برابر 0 میشود اما درمورد بقیه خروجی ها هر کدام از انها فقط هنگامی که state مربوط به انها فعال شود ، فعال میشوند و بقیه ی خروجی های غیر مربوط به ان state صفر هستند

برای صبر کردن به مدت دو دقیقه باید این مدت زمان را نسبت به پریود کلاک در نظر گرفت یعنی 2min / T یا 48*(10 ^7) =1000000 * 4 * 60 * 2 = 2min * F = 2min * 4MHz بنابراین به تعداد 48*(10 ^7)

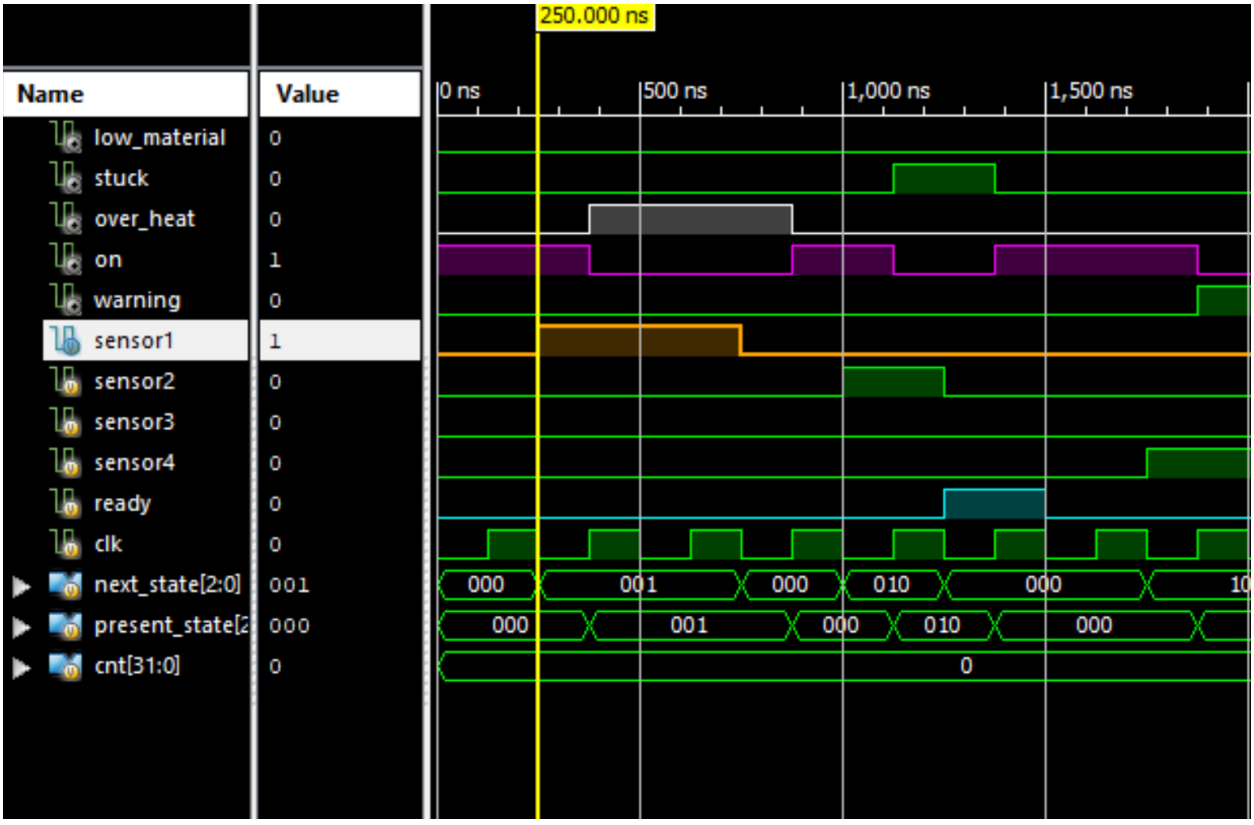
لبه بالارونده کلاک باید صبرکنیم و خط تولید را روشن نگه داریم و اگر از این دو دقیقه بیشتر شد انگاه خط تولید و دستگاه خاموش شده و بدنبال ان تا اضافه کردن مواد اولیه (خاموش شدن سنسور مربوطه) و فشار دادن دکمه ready خط تولید خاموش بماند و در state کمبود مواد اولیه باقی بماند پس متغیر cnt با فعال شدن سنسور 3 شروع به شمارش این دو دقیقه میکند و به محض تمام شدن این دو دقیقه on=0 میشود و با خاموش شدن دستگاه ها سنسور نیز غیر فعال میشود (در ورودی و تست بنچ) و هنگامی که مواد تامین شدند (ready=1) شد به حالت نرمال و روشن شدن خط میرویم

در تست بنچ از انجایی که باید به مدت 2 دقیقه صبر کنیم و درصورت نوشتن #120,000,000,000 برای تاخیر بر حسب نانو ثانیه ای بعلت بزرگ بودن عدد به ارور بر میخوریم بنابراین از حلقه repeat به تعداد 40000 و به مدت زمان 3000,000 تاخیر استفاده میکنیم ;#3000000 (40000) repeat

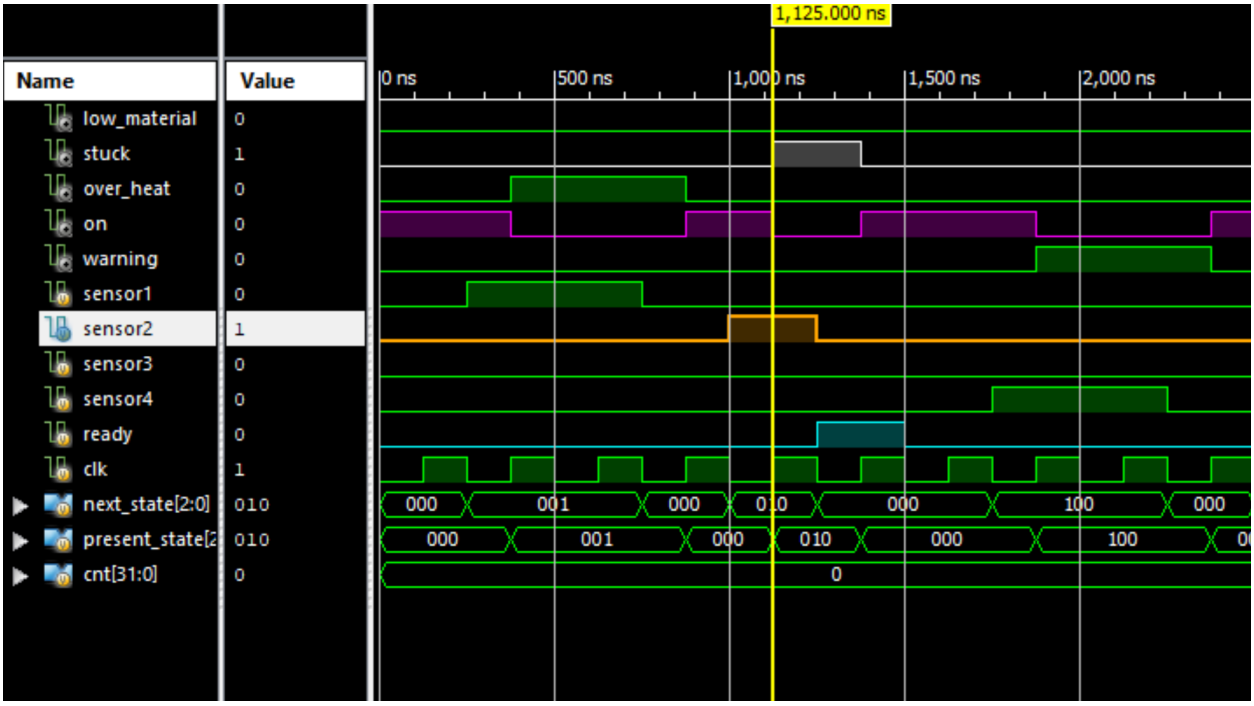


نتایج شبیه سازی و waveform ها

فعال شدن سنسور اول و ورود به حالت heat

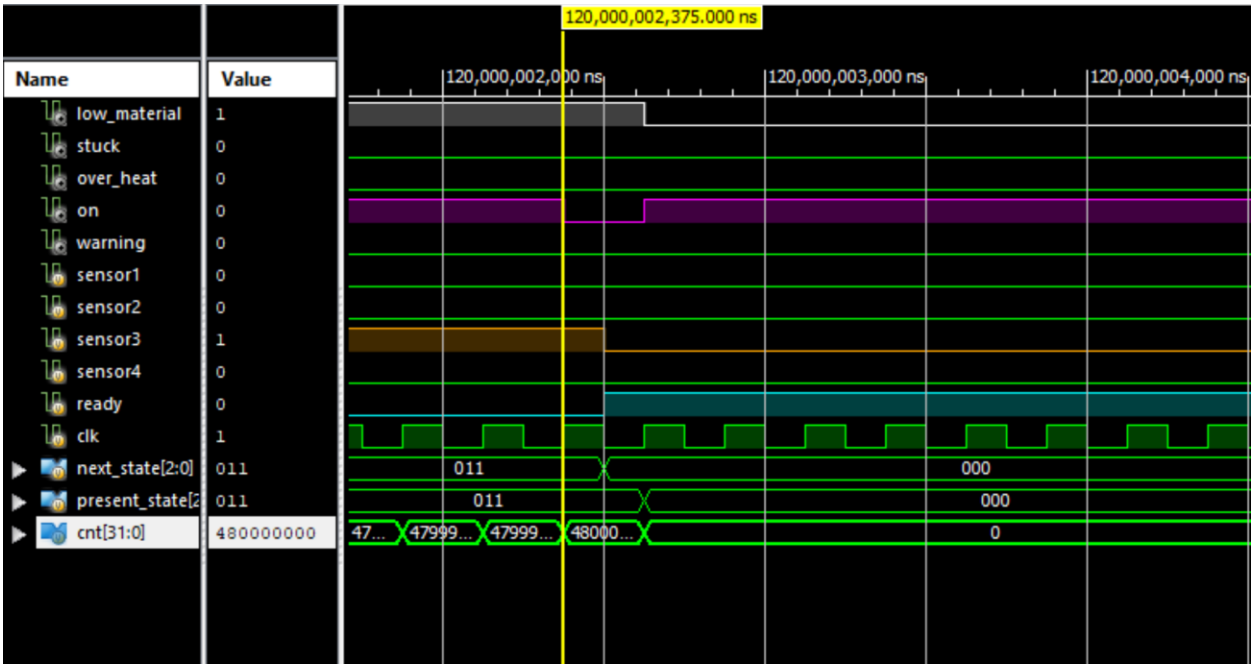


فعال شدن سنسور دوم و ورود به حالت stucked . در پی این حالت خط خاموش میشود و تا یک شدن کلید ready منتظر میماند



فعال شدن سنسور چهارم و ورود به حالت خطر و ریست شدن اتومات دستگاه و روشن شدن مجدد بعد از رفع خطر (غیر فعال شدن سنسور 4 ام)

با فعال شدن سنسور سوم و انتظار 2 دقیقه ای و پس از ان فشار کلید ready و غیر فعال شدن سنسور



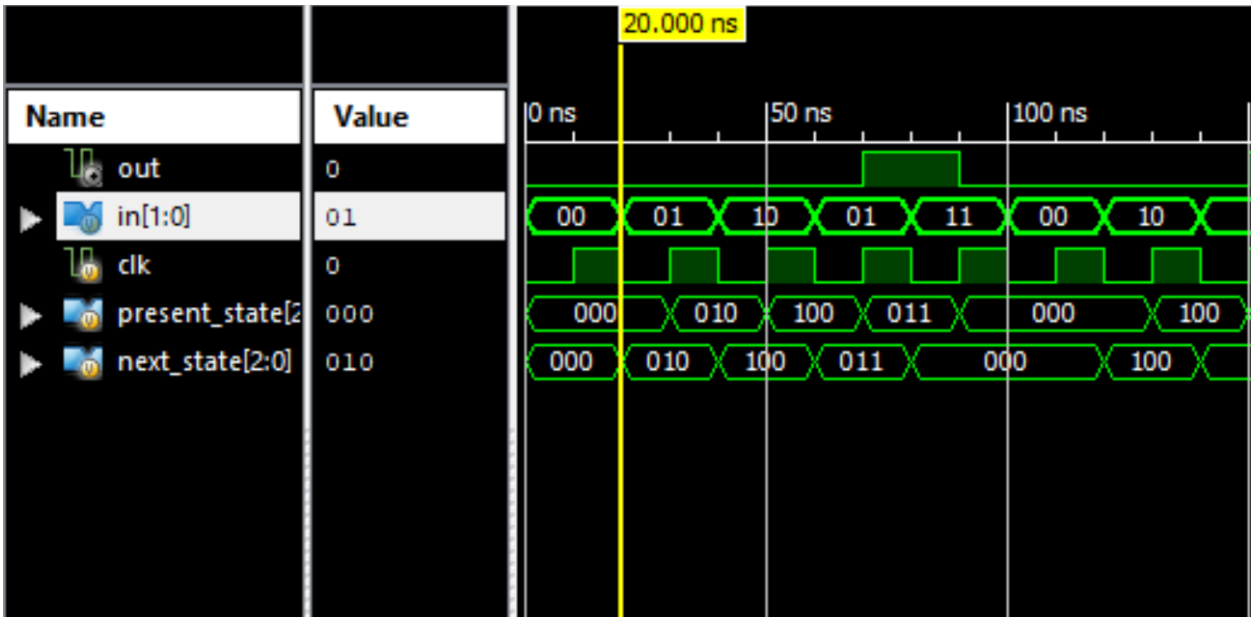
همان گونه که مشاهده میشود بعلت سنکرون سازی طرح و عوض شدن state های طرح در لبه های منفی و اعمال این تغییرات در لبه های مثبت به اندازه نصف پریود کلاک تاخیر در مشاهده ی خروجی و اقدام های مختلف روبات وجود دارد که در صورت combinational کردن طرح میتوان این تاخیر را از بین برد اما در این صورت کنترل کمتری روی طرح و مشاهده ی خروجی ها داریم

سوال 3) با توجه به دیاگرام ماشین حالت سوال ، با ورودی های مختلف state ها تغییر پیدا میکنند بنابراین بعلت وجود 6 تا حالت باید 6 پارامتر به نام های s0 , s1 , s2 , s3 , s4 , s5 تعریف کرده و برای کد گزاری انها از کدگزاری باینری استفاده میکنیم پس برای پوشش 6 حالت باید پارامتر ها سه بیتی باشند همچنین بعلت مقدار گرفتن present_state , next_state از این پارامتر ها ، این دو متغیر نیز 3 بیتی تعریف میشوند .

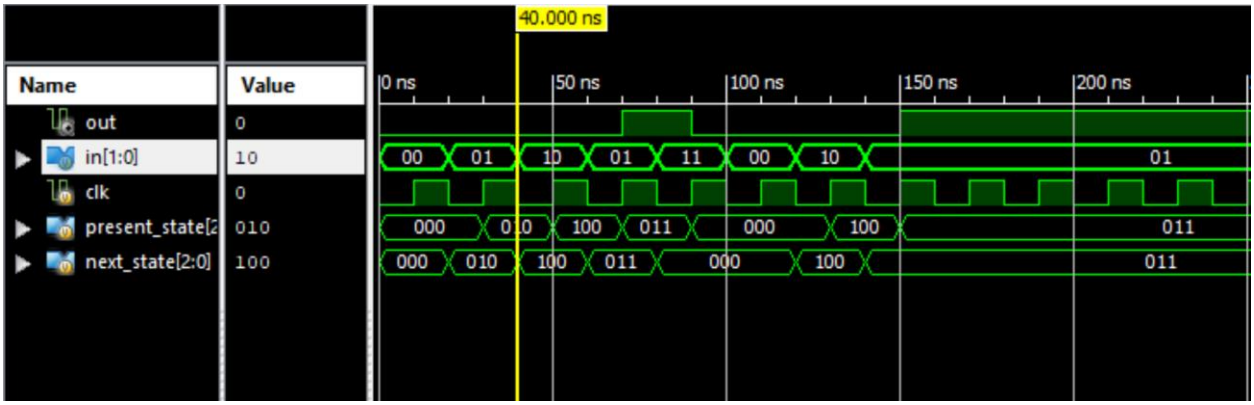
با شروع از حالت s0 (هنگام اولین بار کار کردن fpga درصورتی که مقدار اولیه متغیر present_state را متفاوت بدهیم از حالت دیگری شروع میشود) با توجه به ورودی های مختلف 00و01و10و11 به حالتهای مشخص شده در سوال میرویم و برای سنکرون سازی در هرلبه بالارونده کلاک state ها را تغییر شان را اعمال میکنیم و برای مشاهده ی خروجی ها ، بلافاصله با رفتن به state های بعدی ، خروجی ها را بصورت ترکیبی و combinational تغییر میدهیم که با توجه به دیاگرام سوال خروجی در state های s1 , s3 , s5 برابر یک و در غیر این صورت برابر صفر است

نتایج شبیه سازی و waveform ها

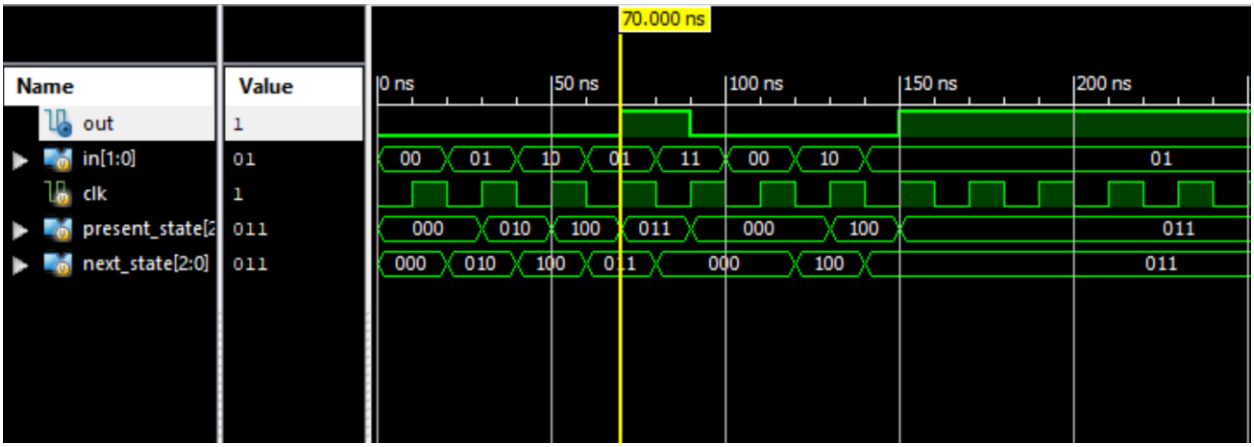
با شروع از s0 و ورودی 01 باید به s2 با کد 10 میرویم



ورودی 10 از s2 به s4 میرویم

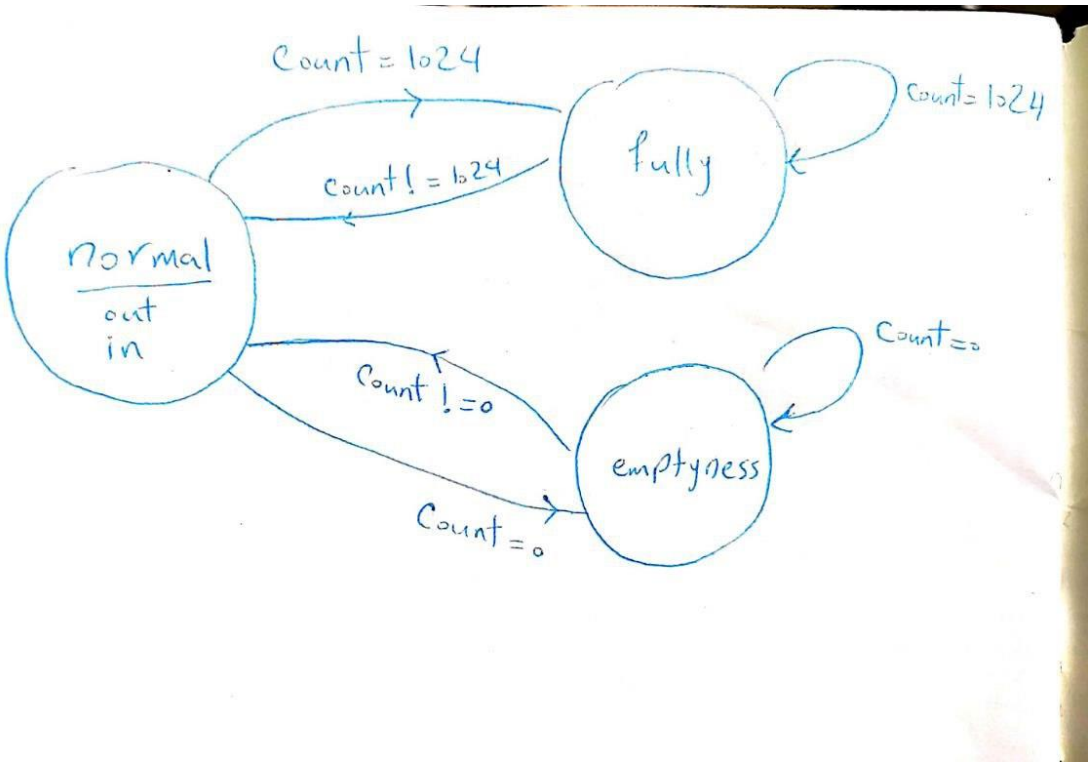


ورودی 01 از s4 به s3 میرویم



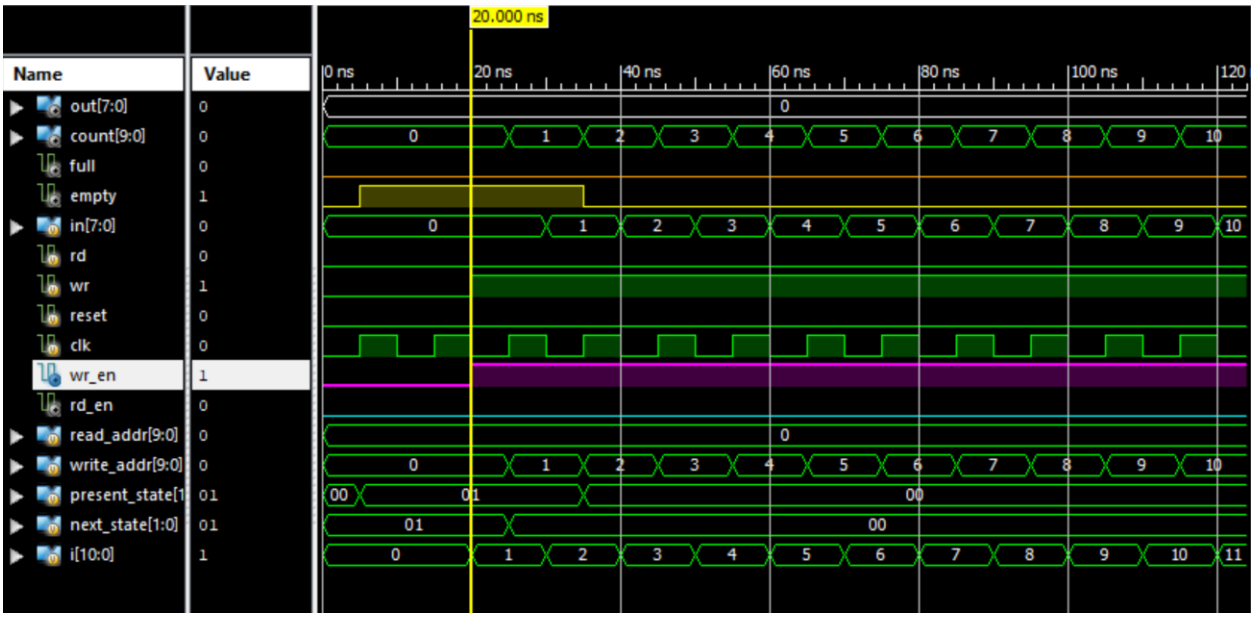
سوال 4) در این سوال یک حالت wait در ابتدا تعریف میکنیم برای هنگامی که کاربر نه از فیفو میخواند و نه مینویسد اما با فعال شدن wrیا rd به یکی از حالتهای write یا read میرویم . اما برای تشخیص حالت های پر یا خالی بودن فیفو باید از یک متفیر بنام countاستفاده کنیم تا فاصله ی بین ادرس های خواندن و نوشتن را محاسبه کند و در صورتی که این فاصله به اندازه ی عمق فیفو منهای یک شود یعنی فیفو پر شده است (ادرس نوشتن به انتهای فیفو رسیده است اما ادرس خواندن در همان ابتدای فیفو مانده است) و اگر فاصله بین دو ادرس به صفر برسد یعنی فیفو خالی شده است (ادرس نوشتن به خواندن رسیده است و تمامی اطلاعات نوشته شده خوانده شده است) و کافی است طبق ماشین حالت زیر عمل کنیم . توجه داشته باشید زمانی میتوانیم از فیفو بخوانیم که کاربر دستور خواندن داده باشد و فیفو نیز خالی نباشد بنابراین صرف فعال شدن پایه rd نمیتوان از فیفو خواند و همچنین درصورتی میتوانیم داخل فیفو داده ی جدید بریزیم که پر نباشد بنابراین درصورت فعال شدن wr باید مراقب پر بودن فیفو نیز باشیم ! در هر لحظه از زمان اگر فقط روی فیف. داده ی جدیدی نوشته شود count بعلت زیاد شدن داده های فیفو یک واحد زیاد میشود و برعکس اگر فقط خوانده شود یک واحد از تعداد داده های داخل فیفو کم میشود امام اگر همزمان هم یک داده نوشته شود و هم خوانده شود در تعداد داده های داخل فیفو تغییری ایجاد نمیشود

در تست بنچ ان برای ایجاد 1024 تا داده به ناچار از حلقه ی تکرار شونده استفاده میشود و تا پر شدن فیفو این داده ها با فعال بودن wr روی فیفو نوشته شده و در نهایت از فیفو طی 1024 کلاک بعدی خوانده میشود و فیفو خالی میگردد

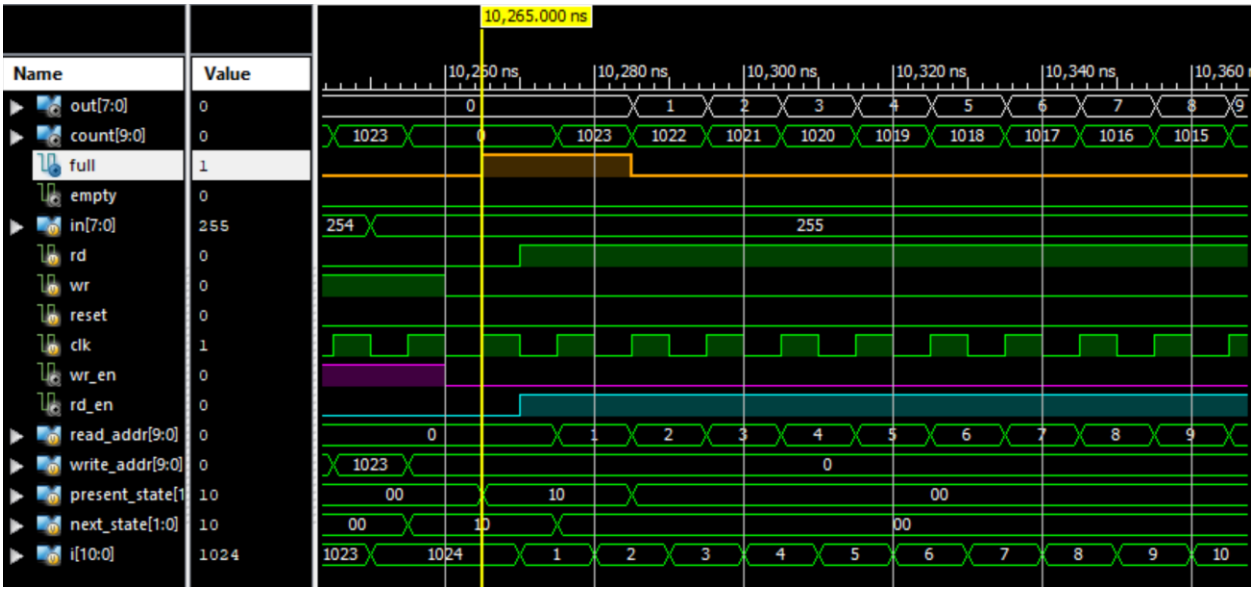


نتایج شبیه سازی و waveform ها

در ابتدا خالی بودن فیفو و بعد از نوشتن یک داده روی ان و روند پر شدن



پر شدن فیفو بعد از نوشتن 1024 داده روی ان



شروع به خواندن کردن مجموعه ی 0 و 1 و ... داده های ورودی