به نام خدا

تکلیف سری ششم fpga

محدثه غفوری(9632133)

قسمت الف) ماژول calc : این ماژول بسته به ورودی op هشت عملیات مختلف را باید انجام دهد که برای مقادیر مختلف ان و خروجی های متناظر با ان از case برای پیاده سازی ان استفاده میکنیم . در این ماژول برای اینکه از داخل ماژول بتوانیم عملیات روی ماتریس ها و مقادیر داخل ماتریس ها براحتی کنترل کنیم سه خروجی addra_r, addrb_r, addrc_w را بصورت خروجی تعریف میکنیم تا هم خواندن از حافظه های A , B و هم نوشتن روی حافظه C را براحتی کنترل کنیم . همچنین برای کنترل زمان نوشتن روی حافظه C بوسیله ی خروجی wec که پایه فعال ساز نوشتن این حافظه است ، از داخل این ماژول و همزمان با ریختن حاصل عملیات روی حافظه این پایه را فعال میکنیم.لازم به ذکر است دو ورودی din_A , din_B خانه های حافظه های A , B هستند که در زمان تعیینی توسط همین ماژول calc وارد میشوند تا عملیات مورد نظر روی انها صورت گیرد (اینکه کدام عنصر حافظه ها وارد این ماژول شود ، توسط پایه های ادرسی که ماژولcalc مقدار دهی و تعیین میکند مشخص میشود)

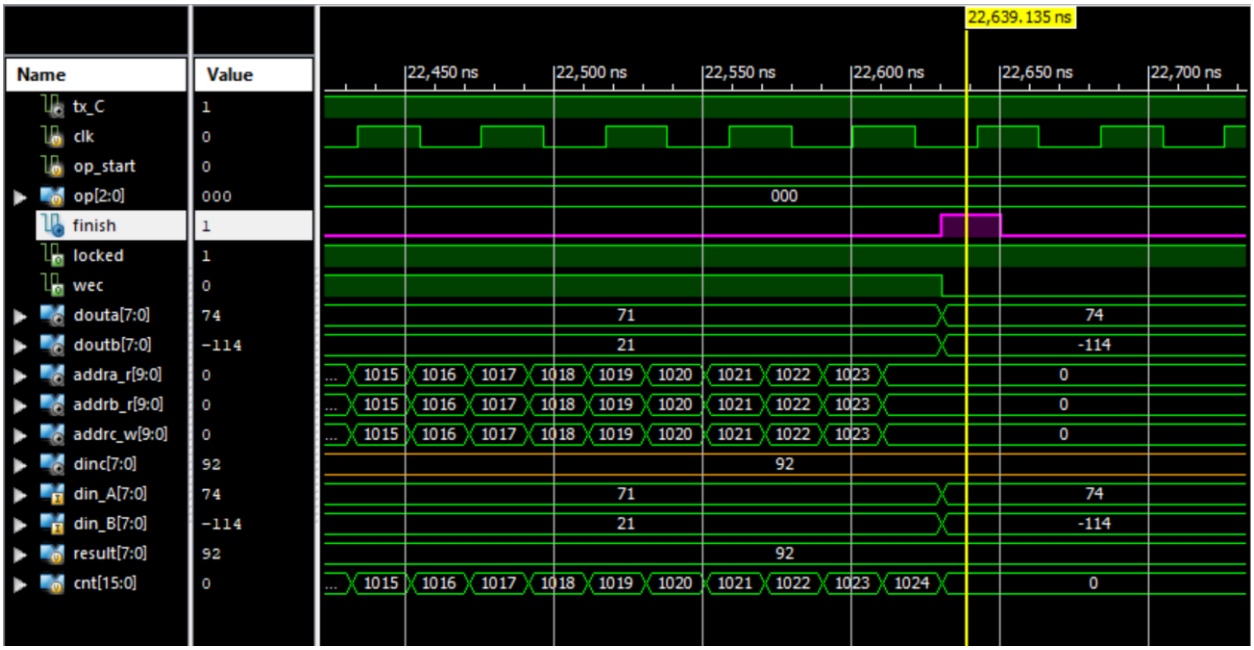
در حالت های op=000, 001 , 010 دو ماتریس A , B باهم جمع یا کم میشوند که در این حالات عملیات مورد نظر درایه به درایه متناظرا روی عناصر این دو ماتریس ، اعمال میشود بنابراین کافی است که در هر کلاک درایه های متناظر را با هم جمع یا کم کرد و در کلاک بعدی روی خانه های این دو ماتریس جلو رفت . بنابراین ادرس های این دو ماتریس هم زمان باهم و با ماتریس حاصل جواب C یک واحد رو به جلو حرکت میکنند . حاصل جمع دو عدد هشت بیتی علامت دار حداکثر 254 میشود که این مقدار در هشت بیت میگنجد بنابراین کافی است حاصل را در متغیر هشت بیتی result قرار دهیم که همان هشت بیت پر ارزش را در خود دارد .

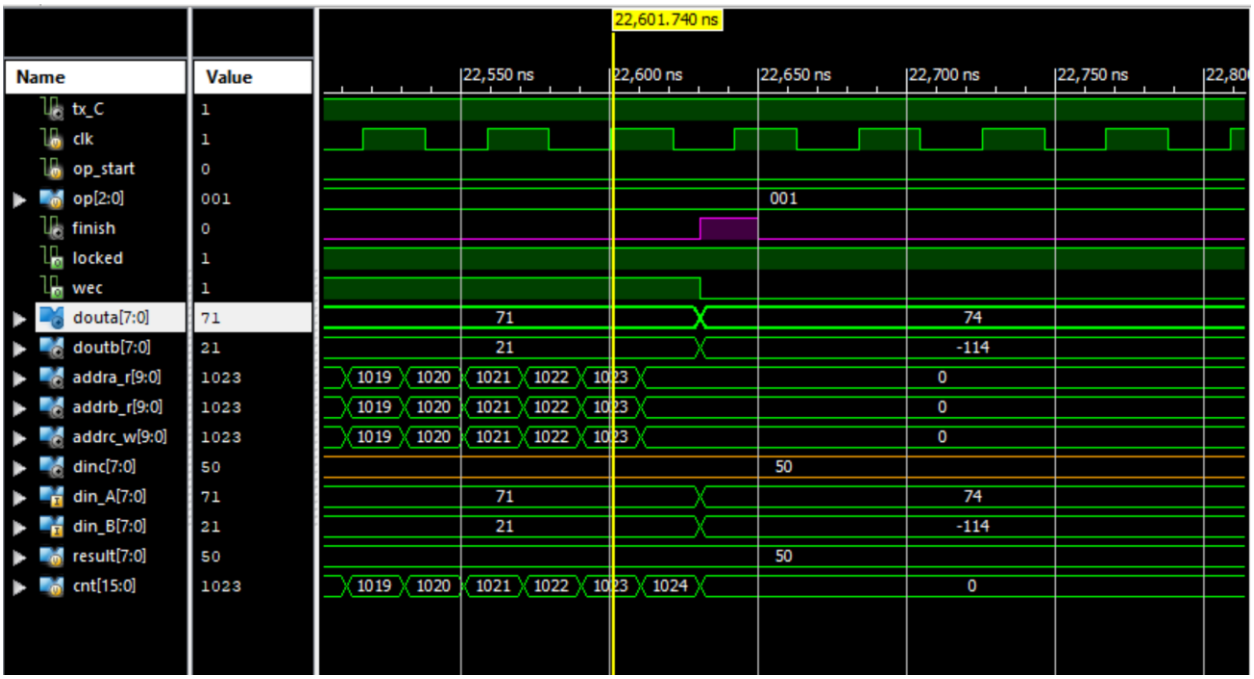
در حالت op=011 دو ماتریس A , BT در هم ضرب میشوند. از انجایی که سطر و ستون های ماتریس B و ماتریس B ترانهاده برعکس هستند برای بدست اوردن هر عنصر ماتریس C بجای ضرب کردن سطر های A در ستون های BT، میتوان سطر های A را در سطر های B ضرب کرد و تمامی این ضرب ها را با هم جمع کرد . از انجایی که درایه های سطر ها نظیر به نظیر در هم ضرب میشوند در هر مرحله شماره ستون درایه ها یکسان و شماره سطر ها متفاوت خواهد بود . متغیر m روی ستون های مختلف A,B حرکت میکند و متغیر iسطر های ماتریس A و متغیر j سطر های ماتریس Bرا نمایش میدهد . هنگامی که تمامی عناصر یک سطر نظیر به نظیر ضرب شدند ، یعنی به اخرین ستون هر سطر رسیدیم m=31 میشود و flag=1 میشود بنابراین flag نشان دهنده اتمام عملیات و اماده شدن پاسخ یک درایه از حافظه c است . برای بدست اوردن عناصر مختلف c باید با ثابت نگه داشتن سطر A روی سطر های مختلف B حرکت کنیم تا یک سطر مشخص ماتریسC بدست اید و برای سطر بعدی ان باید یک واحد شماره سطر A را جلو ببریم و دوباره از سطر اول تا اخر B عناصر را در هم ضرب کرده و با هم جمع کنیم . بنابراین هرگاه به انتهای سطر رسیدیم و flag فعال شد یعنی یک خانه از c اماده شده است و میتوانیم به خانه بعدی برویم و یک واحد پایه ادرس addrc_w را افزایش دهیم و پس از پر شدن تمامی خانه های ماتریس c متغیر ها پاک و صفر کنیم و wec را غیر فعال کنیم . برای نگه داشتن هشت بیت پر ارزش حاصل عملیات و ریختن این هشت بیت در ماتریس c از متغیر هشت بیتی result و

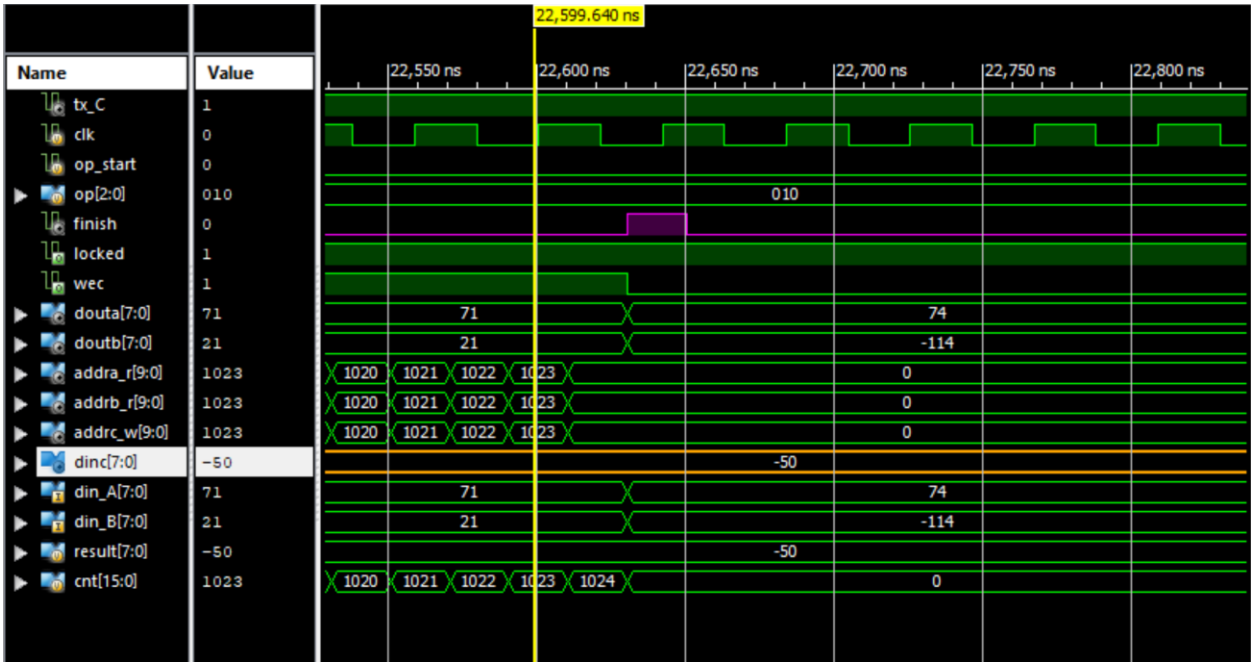
متغیر 11 بیتی temp2 استفاده میکنیم زیرا هر عنصر c در بدترین حالت برابر (127*127)*32 است که این حاصل در یک متغیر 19 بیتی جا میگیرد و همان طور که گفته شد هشت بیت پرارزش یعنی result را به dout_c میدهیم

در حالت op=100,101 ماتریس های A , B ترانهاده میشوند یعنی جای سطر و ستون شان عوض میشود و تغییری در خود مقادیر شان بوجود نمی اید بنابراین تنها کافی است که پایه های ادرس ماتریسی که میخواهیم ترانهاده کنیم و ماتریس حاصل جواب را کنترل کنیم . از انجایی 32 سطر و ستون داریم میتوانیم ادرس هر خانه را با 32 سطر و ستون داریم میتوانیم ادرس هر خانه را با i+j*32 نمایش دهیم که در ان j نمایش دهنده شماره سطر با شروع از صفر ، و i نمایش دهنده شماره ستون با شروع از صفر است . با توجه به اینکه در ماتریس حاصل جواب سطر و ستون های ماتریس اصلی برعکس شده اند میتوان ادرس هر خانه ی حاصل جواب را با i*32+j نمایش داد . با توجه به کد نوشته شده با ثابت نگه داشتن i روی یک ستون خاص از ماتریس اصلی حرکت میکنیم(با زیاد کردن شماره سطرj) و در هر گام عناصر ستون ماتریس اصلی را در یک سطر از ماتریس حاصل میریزیم و هنگامی که عناصر یک ستون منتقل شدند با افزایش i به ستون بعدی میرویم و این عملیات را تا اخرین ستون که i=31 است انجام میدهیم . در این حالت چون هیچ عملیاتی روی مقادیر عناصر حافظه ها صورت نمیگیرد و همان ها با تغییر مکان به ماتریس c منتقل میشوند بنابراین به همان صورت ورودی که هشت بیتی هستند باقی میمانند و کافی است مقادیر انها را در متغیر هشت بیتی result قرار دهیم تا روی حافظه c نوشته شود

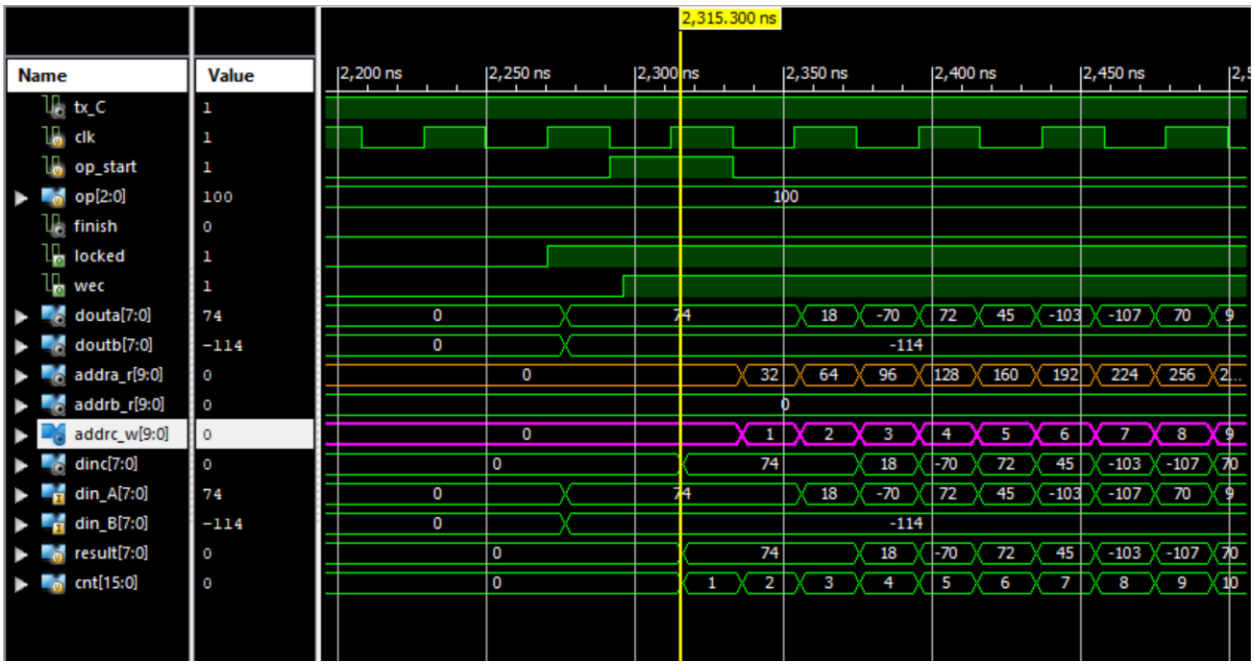
در حالت op=110 , 111 باید trace ماتریس ، یعنی جمع عناصر روی قطر اصلی ماتریس را بدست اوریم و در اولین خانه ی ماتریس c بریزیم بنابراین نیازی به جابجایی ادرس های حافظهc نیست و پایه ادرس c روی همان صفر ، یعنی روی اولین درایه اش باقی میماند . اما در ماتریس اصلی برای حرکت روی قطر اصلی میدانیم درایه های قطر اصلی دارای ادرس های ... , 66 , 33 , 0 دارند . یعنی عناصر روی قطر اصلی ادرس دارند . یعنی عناصر روی قطر اصلی ادرس i*33 دارند که i از صفر (اولین خانه ی ماتریس) شروع شده و تا 31 (اخرین خانه ماتریس با ادرس 1023) جلو میرود . بنابراین کافی است با حرکت روی این ادرس ها ، مقادیر داخل این خانه ها را با هم جمع کنیم و در نهایت 8 بیت پر ارزش این حاصل جمع را در خروجی قرار دهیم که برای این کار از دو متغیر result بطول هشت بیت و temp3 بطول پنج بیت استفاده میکنیم . بدین صورت که حاصل این جمع حداکثر (127+127)*32 میباشد(اعداد بصورت هشت بیتی علامت دارد بین 127 , 128- تغییر میکنند) که این مقدار در 13 بیت میگنجد بنابراین این حاصل را در {result,temp3} ریخته و بوسیله دستور assign هشت بیت پر ارزش ان به خروجی dout_c تخصیص داده میشود

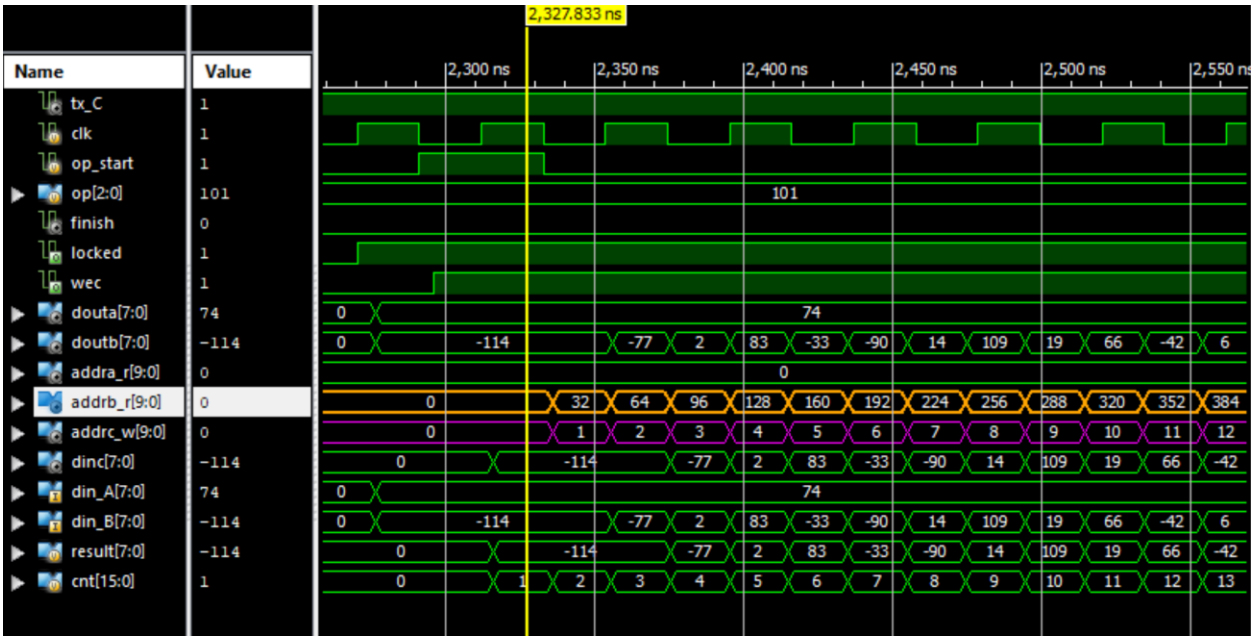

22,639.135 ns
Name
Value
22,450 ns
22,500 ns
22,550 ns
22,600 ns
22,650 ns
22,700 ns
tx_C 1
clk 0
op_start 0
op[2:0] 000
finish 1
locked 1
wec 0
douta[7:0] 74
doutb[7:0] -114
addra_r[9:0] 0
addrb_r[9:0] 0
addrc_w[9:0] 0
dinc[7:0] 92
din_A[7:0] 74
din_B[7:0] -114
result[7:0] 92
cnt[15:0] 0
000
71
74
21
-114
1015 1016 1017 1018 1019 1020 1021 1022 1023
0
92
1024

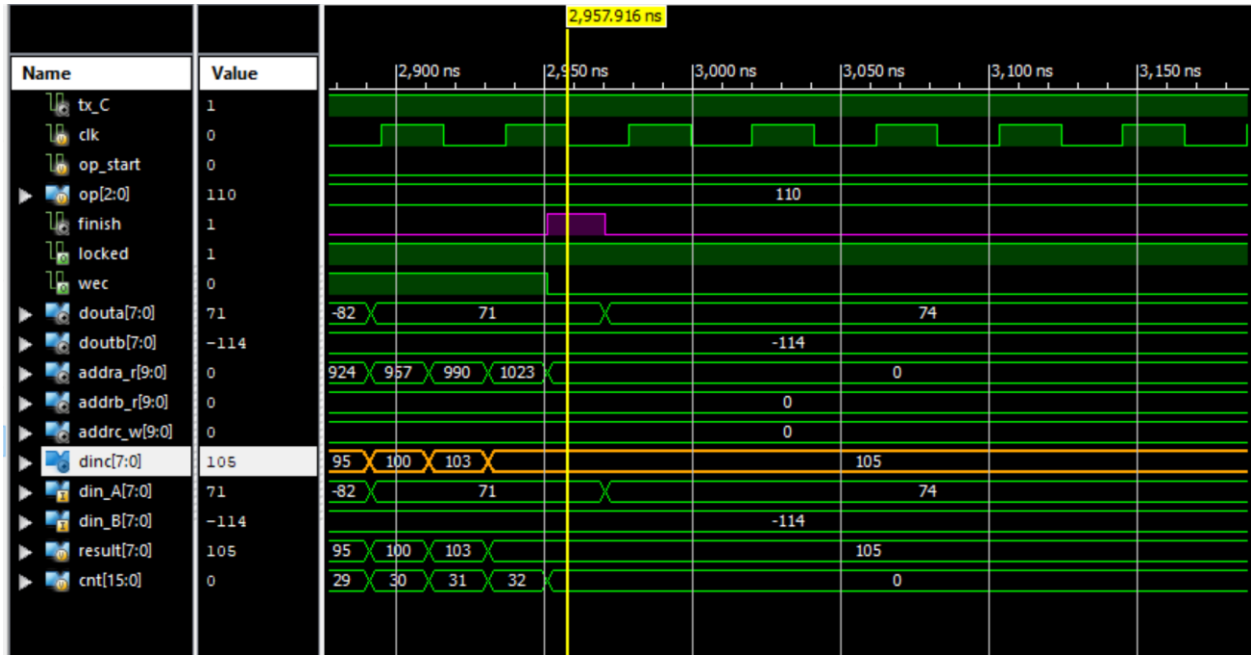

22,601.740 ns
Name
Value
22,550 ns
22,600 ns
22,650 ns
22,700 ns
22,750 ns
tx_C 1
clk 1
op_start 0
op[2:0] 001
finish 0
locked 1
wec 1
douta[7:0] 71
doutb[7:0] 21
addra_r[9:0] 1023
addrb_r[9:0] 1023
addrc_w[9:0] 1023
dinc[7:0] 50
din_A[7:0] 71
din_B[7:0] 21
result[7:0] 50
cnt[15:0] 1023
001
71
74
21
-114
1019 1020 1021 1022 1023
0
50
1024

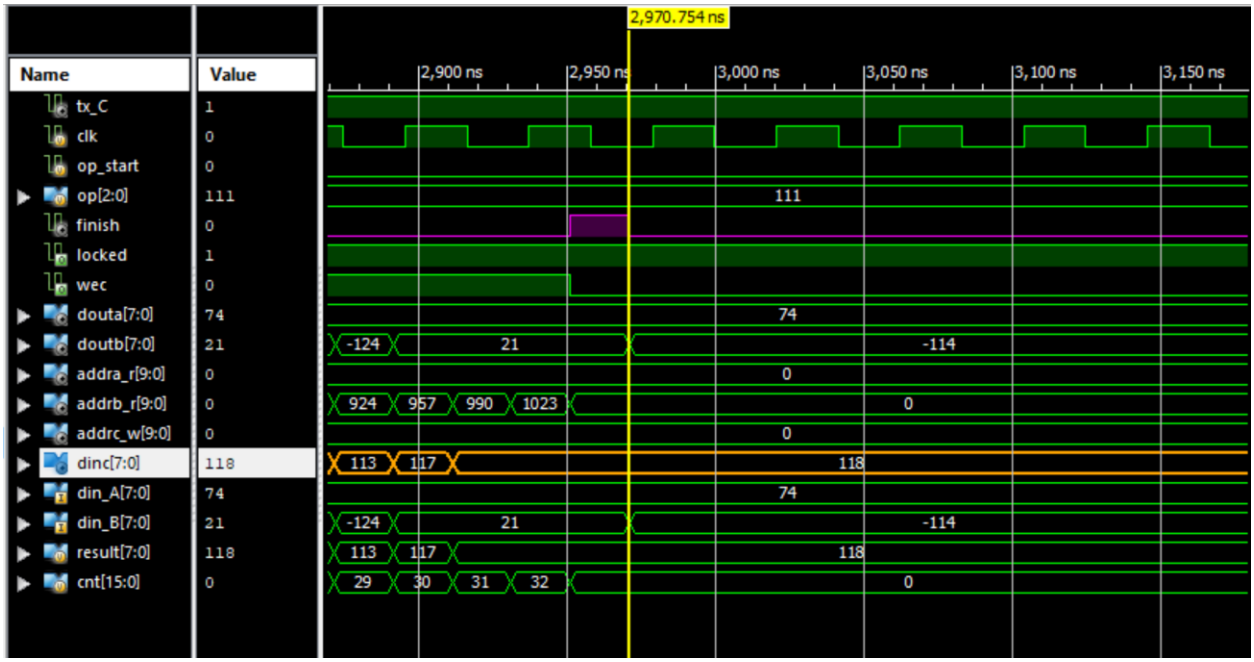

22,599.640 ns
Name
Value
tx_C 1
clk 0
op_start 0
op[2:0] 010
finish 0
locked 1
wec 1
douta[7:0] 71
doutb[7:0] 21
addra_r[9:0] 1023
addrb_r[9:0] 1023
addrc_w[9:0] 1023
dinc[7:0] -50
din_A[7:0] 71
din_B[7:0] 21
result[7:0] -50
cnt[15:0] 1023
22,550 ns
22,600 ns
22,650 ns
22,700 ns
22,750 ns
22,800 ns


2,955.935 ns
Name
Value
tx_C 1
clk 1
op_start 0
op[2:0] 011
finish 0
locked 1
wec 1
douta[7:0] -44
doutb[7:0] -65
addra_r[9:0] 32
addrb_r[9:0] 32
addrc_w[9:0] 0
dinc[7:0] 48
din_A[7:0] -44
din_B[7:0] -65
result[7:0] 48
cnt[15:0] 33
2,850 ns
2,900 ns
2,950 ns
3,000 ns
3,050 ns


2,315.300 ns
Name
Value
2,200 ns
2,250 ns
2,300 ns
2,350 ns
2,400 ns
2,450 ns
tx_C 1
clk 1
op_start 1
op[2:0] 100
finish 0
locked 1
wec 1
douta[7:0] 74
doutb[7:0] −114
addra_r[9:0] 0
addrb_r[9:0] 0
addrc_w[9:0] 0
dinc[7:0] 0
din_A[7:0] 74
din_B[7:0] −114
result[7:0] 0
cnt[15:0] 0


2,327.833 ns
Name
Value
2,300 ns
2,350 ns
2,400 ns
2,450 ns
2,500 ns
tx_C 1
clk 1
op_start 1
op[2:0] 101
finish 0
locked 1
wec 1
douta[7:0] 74
doutb[7:0] −114
addra_r[9:0] 0
addrb_r[9:0] 0
addrc_w[9:0] 0
dinc[7:0] −114
din_A[7:0] 74
din_B[7:0] −114
result[7:0] −114
cnt[15:0] 1


2,957.916 ns
Name
Value
2,900 ns
2,950 ns
3,000 ns
3,050 ns
3,100 ns
3,150 ns
tx_C 1
clk 0
op_start 0
op[2:0] 110
finish 1
locked 1
wec 0
douta[7:0] 71
doutb[7:0] -114
addra_r[9:0] 0
addrb_r[9:0] 0
addrc_w[9:0] 0
dinc[7:0] 105
din_A[7:0] 71
din_B[7:0] -114
result[7:0] 105
cnt[15:0] 0


2,970.754 ns
Name
Value
2,900 ns
2,950 ns
3,000 ns
3,050 ns
3,100 ns
3,150 ns
tx_C 1
clk 0
op_start 0
op[2:0] 111
finish 0
locked 1
wec 0
douta[7:0] 74
doutb[7:0] 21
addra_r[9:0] 0
addrb_r[9:0] 0
addrc_w[9:0] 0
dinc[7:0] 118
din_A[7:0] 74
din_B[7:0] 21
result[7:0] 118
cnt[15:0] 0

قسمت ب) ماژول clk_gen : این ماژول کلاک ورودی سیستم را که 24MHz است را دریافت میکند و بوسیله ی یک DCM و عملیاتی دیگر کلاک ماژول calc که صد مگاهرتز و کلاک ماژولهای uart که 115200 هرتز هستند را تولید میکند . برای تولید clk_calc کافی است با استفاده از همان DCM کلاک ورودی بیست و چهار مگاهرتز را به 100 مگاهرتز تبدیل کنیم و در لبه های بالا رونده این کلاک صد مگاهرتزی اگر LOCKED فعال شده بود و کلاک ها اماده ی استفاده بودند ، clk_calc را نات کنیم تا سنکرون با لبه های کلاک صد مگاهرتزی تغییر کند و در واقع این خروجی هم یک کلاک صد مگاهرتزی شود . برای تولید کلاک 115200 هرتزی با یک تقسیم ساده میبینیم

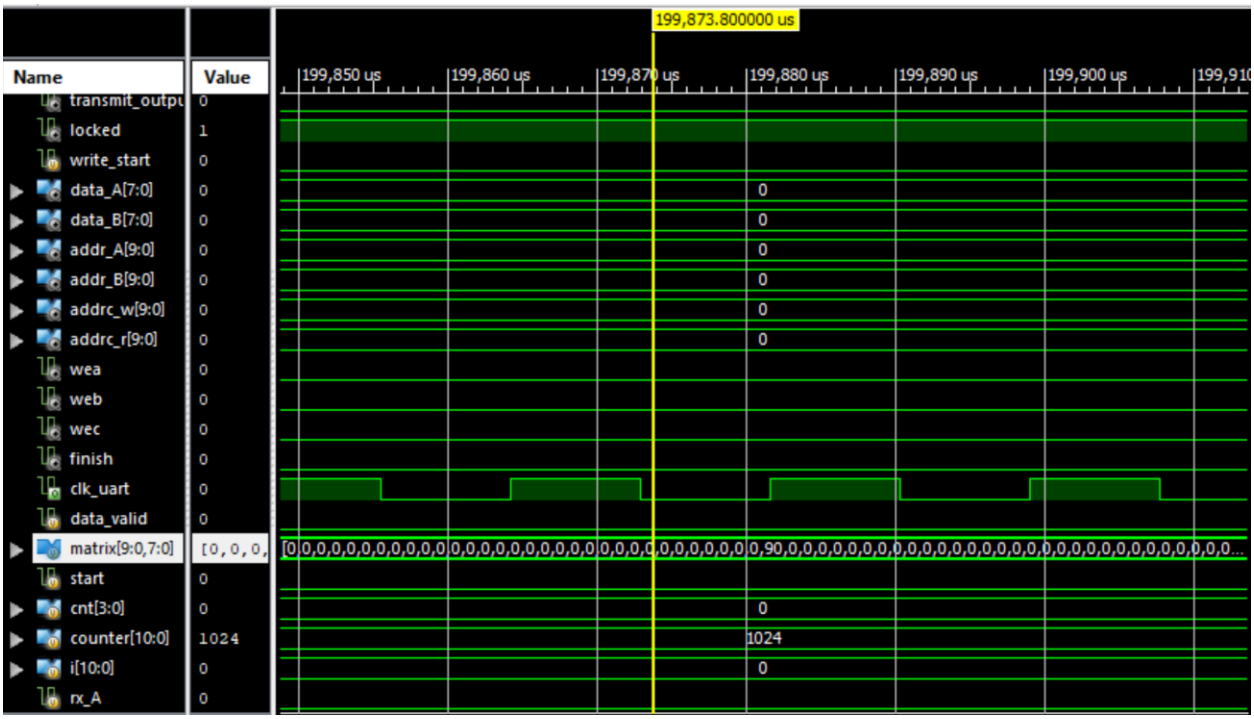
72MHz / 115200 = 625 میشود بنابراین برای این کلاک بصورت دقیق ، چون DCM نمیتواند کلاک های کوچک را درست کند یک کلاک 72 مگاهرتزی تولید میکنیم و هر 625 بار یکبار clk_uart را نات میکنیم تا این کلاک کوچک تولید شود

ماژول UART_rec : این ماژول طبق پروتکل یورات باکلاک 115200 داده های سریال را از ورودی دریافت میکند و خانه های حافظه های A یا B را مقدار دهی میکند . برای این منظور کافی است از یک متغیر ده بیتی بنام register استفاده کنیم تا داده های ورودی داخل ان قرار گیرند و در هر کلاک یک واحد به سمت چپ شیفت پیدا کند و پس از گرفتن هر ده تا ورودی ، مقدار خود را چک کند اگر طبق پروتکل یورات استارت بیت (صفر) و استاپ بیت (یک) در ابتدا و انتهای این فریم ده بیتی باشند هشت بیت وسطی خود را بعنوان یک عدد هشت بیتی بردارد تا وارد خانه های ماتریس مورد نظر کند . پس از دریافت هر عدد هشت بیتی انرا در خانه های یک ارایه دو بعدی بنام matrix قرار میدهیم که 1024 تا خانه بطول هشت بیت دارد بنابراین با دریافت هر عدد هشت بیتی counter یک واحد زیاد میشود تا هر 1024 تا خانه ی این ارایه دو بعدی مقدار دهی شود و هنگامی که counterبرابر 1024 شد ، یعنی تمامی عدد های هشت بیتی برای مقدار دهی به خانه های ماتریس دریافت شدند، منتظر میمانیم تا write_start یک شود و ما بتوانیم این اعداد را به بیرون ماژول هدایت کنیم تا روی حافظه ها نوشته شوند . برای اینکه بتوانیم نوشتن روی حافظه ها را کنترل کنیم از یک خروجی data_valid استفاده میکنیم که به wea یا web متصل میشوند زیرا با فعال شدن write_start باید بتوانیم مقادیر دریافتی در ماژول گیرنده را روی حافظه ها بنویسیم و برای این کار نیاز به کنترل پایه فعال ساز نوشتن حافظه ها داریم که از طریق فعال کردنdata_valid در زمانی که روی حافظه ها مینویسیم ، به این هدف دست پیدا میکنیم

ماژول UART_trans : با فعال شدن finish و اتمام عملیات میتوانیم خانه های ماتریس c را اماده ی ارسال کنیم.برای ارسال 1024 تا خانه ی c از یک متغیر بنام counter که از صفر تا 1024 تغییر میکند استفاده میکنیم و هر بار که یک خانه ی این ماتریس ارسال شد ، یک واحد به این متغیر افزوده میشود . برای فرستادن بیت های هر خانه هنگامی که data_valid فعال بود ، در صورتی که مشغول ارسال دیگری نباشیم (busy=0) باشد میتوانیم ارسال

هشت بیت یک خانه ی خاص از ماتریس c را انجام دهیم که برای این کار باید ابتدا فریم متناسب با پروتکل یوارت یعنی {1,data,0} را تولید کنیم و در ده کلاک متوالی که بوسیله ی متغیر cnt کنترل میشود این فریم را ارسال میکنیم و پس از ارسال کل فریم طی یک کلاک متغیری که فریم در ان قرار میگیرد (shift_reg) را پاک میکنیم (shift_reg<=10'b111111111;) تا اماده ی ارسال خانه ی بعدی حافظه شویم و بدنبال ان ادرس حافظه ی c را یک واحد افزایش میدهیم تا ورودی ماژول خانه ی بعدی این حافظه شود . در واقع در این ماژول خواندن از حافظه ی c بوسیله ی خروجی addr کنترل میشود که به پایه ادرس خواندن c متصل است . این ماژول نیز همانند ماژول گیرنده یورات با کلاک 115200 کار میکند

قسمت ج) برای ساخت ماتریس A باید عناصر روی قطر اصلی باید 90 یا 01011010 باشد و سایر عناصر باید صفر باشند. بدین منظور با تعریف یک متغیر بنام i که از 0 تا 1023 تغییر میکند میتوانیم عناصر روی قطر اصلی که همان مضارب 33 هستند را با 90 مقدار دهی کنیم (اگر باقی مانده i به 33 برابر صفر باشند یعنی i مضربی از 33 بوده است) و ما بقی را با صفر. توجه شود که طبق استاندارد پروتکل یوارت باید استارت بیت و استاپ بیت در ابتدا و انتهای داده ی اصلی قرار گیرند



قسمت د)برای اینکه پورت ها را به پین ها طبق جدول متصل کنیم از **plan ahead** استفاده میکنیم

```
# PlanAhead Generated physical constraints
NET "clk" LOC = P50;
NET "rx_A" LOC = P81;
NET "rx_B" LOC = P83;
NET "tx_C" LOC = P85;
NET "Op_start" LOC = P111;
NET "Op[2]" LOC = P114;
NET "Op[1]" LOC = P115;
NET "Op[0]" LOC = P116;
NET "write_start" LOC = P119;
NET "transmit_output" LOC = P120;
```

با اعمال شرط گذاری زمانی میبینیم حداکثر فرکانس ورودی 50 و حداکثر کلاک یوارت 35 و حداکثر کلاک محاسبات 251 مگاهرتز میباشد

	Met	Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors
1	Yes	TS_clk_gen_clk_calc = PERIOD TIMEGRP "clk_gen/clk_calc" 251 MHz HIGH 50%	SETUP HOLD	0.016ns 0.450ns	3.968ns	0 0
2	Yes	TS_clk_gen_clk_generator_clkout0 = PERIOD TIMEGRP "clk_gen_clk_generator_clkout0" TS_clk * 3 HIGH 50%	SETUP HOLD	2.659ns 0.469ns	4.007ns	0 0
3	Yes	TS_clk_gen_clk_generator_clkout1 = PERIOD TIMEGRP "clk_gen_clk_generator_clkout1" TS_clk * 4.2 HIGH 50%	SETUP HOLD MINPERIOD	3.832ns 0.438ns 3.031ns	0.929ns 1.730ns	0 0 0
4	Yes	TS_clk = PERIOD TIMEGRP "clk" 50 MHz HIGH 50%	MINLOWPULSE	15.000ns	5.000ns	0
5	Yes	TS_clk_gen_clk_uart = PERIOD TIMEGRP "clk_gen/clk_uart" 35 MHz HIGH 50%	SETUP HOLD	17.997ns 0.363ns	10.574ns	0 0

```
12  #Created by Constraints Editor (xc6slx9-tqg144-3) - 2020/07/31
13  NET "clk" TNM_NET = clk;
14  TIMESPEC TS_clk = PERIOD "clk" 50 MHz HIGH 50%;
15  NET "clk_gen/clk_calc" TNM_NET = clk_gen/clk_calc;
16  TIMESPEC TS_clk_gen_clk_calc = PERIOD "clk_gen/clk_calc" 251 MHz HIGH 50%;
17  NET "clk_gen/clk_uart" TNM_NET = clk_gen/clk_uart;
18  TIMESPEC TS_clk_gen_clk_uart = PERIOD "clk_gen/clk_uart" 35 MHz HIGH 50%;
19
```