مقدمه

امروزه نیاز به پردازش تصویر در مصارفی مانند پزشکی، امنیتی، نظامی، فضایی و... به سرعت در حال افزایش می باشد. حجم پردازش، پیچیدگی پردازش بویژه نیاز به پردازش های بلادرنگ طراحان را بسوی پیاده سازی های خاص منظوره به صورت سخت افزاری و یا نرم افزاری به همراه الگوریتم های پردازش موازی هدایت کرده است. در این پرژه هدف پیاده سازی سخت افزاری فیلتر میانگین همسایه ها بر روی FPGA به از طریق الگوریتم های محاسباتی و همچنین الگوریتم های پردازش موازی می باشد.

در این پروژه الگوریتم میانگین همسایه ها بر روی FPGA پیاده سازی شده است، حداکثر تاخیر در نظر گرفته شده معادل یک جمع کننده پانزده بیتی می باشد برای رسیدن به این تاخیر الگوریتم های محاسباتی مانند تقسیم بر عدد ثابت، جمع کننده CSA با چندین عملوند مورد استفاده قرار گرفته شده است و همچنین در قسمت حافظه از ماژول های موازی جهت پردازش اطلاعات استفاده شده است. زبان توصیف سخت افزاری که در این پروژه مورد استفاده شده است VHDL می باشد.

در ادامه به بررسی الگوریتم های پردازش موازی خواهیم پرداخت سپس به بررسی نحوه ای پیاده سازی الگوریتم میانگین همسایه ها خواهیم پرداخت که به دو بخش واحد حافظه و واحد محاسباتی تقسیم می شود و در انتها نتایج حاصل از این پروژه مورد بررسی قرار می گیرد.

فصل اول
الگوریتم های پردازش تصویر

۱-۱ مقدمه

برای حذف نویز از تصاویر می توانیم از الگوریتم های فراوانی استفاده کنیم که بعضی از آنها به آسانی پیاده سازی و بعضی دیگر دارای پیاده سازی پیچیده تری هستند این الگوریتم ها بنا به کاربردهایی که مورد استفاده قرار می گیرند، دارای نتایج و عملکردهای متفاوتی هستند. در این بخش سعی می کنیم چند الگوریتم متداول برای حذف نویز از تصاویر را به اجمال توضیح دهیم.

۱-۲ الگوریتم میانگین همسایه ها[۱]

در این الگوریتم اساس کار بر مبنای ساختار فایلهای تصویری و پیکسل ها می باشد و به این صورت عمل می کند که طبق عدد گرفته شده در همسایگی پیکسل مرکزی مربعی را فرض می کند که این عدد می تواند ۳،۵ ویا ۷ باشد(به طور مثال اگر عدد گرفته شده ۳ باشد مربعی ۳*۳ که ۸ پیکسل اطراف مرکزی میشود را در نظر می گیرد) و سپس مقادیر این پیکسل ها را باید با یکدیگر جمع نموده و در نهایت این مجمع را بر تعداد پیکسلهای جمع شده تقسیم می کند. در واقع اگر $G(X,Y)$ تابع حاصل از نتایج باشد که جایگزین خانه مرکزی می شود و $F(X,Y)$ نقاط اطراف نقطهٔ مرکزی باشند. M تعداد نقاط باشد آنگاه الگوریتم را به صورت زیر خواهیم داشت: $G(X,Y)= \sum_{n,m \, £ \, S} F(n,m)$ این الگوریتم باعث از بین رفتن نویز نمی شود بلکه باعث می شود تا نویز به طور یکنواخت در کل تصویر پخش شود.

۱-۳ الگوریتم فیلتر میانه[۲]

این الگوریتم نیز بر اساس ساختار پیکسل های فایل کار می کند و دقیقا مانند الگوریتم میانگین همسایه ها یک مربع به دور یک پیکسل مرکزی در نظر می گیرد ولی با این تفاوت که این بار عملیات میانگین گیری صورت نمی گیرد بلکه مقادیر خانه ها را در یک آرایه ریخته و آنها را مرتب می کند خانه ای که دارای مقدار میانه آرایه است را به عنوان خانه مرکزی قرار می دهد و بقیه نقاط نیز به همین تر تیب اعمال می شوند. این الگوریتم بنا به نوع الگوریتم مرتب سازی دارای زمان مصرفی $O(n^2)$ یا $O(n \, Log(n))$ می باشد در نتیجه از الگوریتم قبلی زمان بیشتری را مصرف می کند که دارای زمان مصرفی $O(n)$ بود ولی در اینجا به دلیل ساختار و نوع الگوریتم و اینکه نقاط به ترتیب مرتب سازی می شوند به مرور نویز از تصاویر حذف می شوند در صورتی که درالگوریتم قبلی نویز ها از تصاویر خذف نمی شدند. برای درک بهتر الگوریتم Median در شکل ۱-۱ یک مثال آورده شده است.

۱۲۷	۱۲۵	۱۲۲
۱۲۵	۲۴۰	۱۲۴
۶	۱۲۱	۱۱۹

Median 3*3 →

۱۲۷	۱۲۵	۱۲۲
۱۲۵	124	۱۲۴
۶	۱۲۱	۱۱۹

شکل ۱-۱ عملکرد الگوریتم Median

[۱] Neighborhood Averaging .
[۲] Median Filter .

پس از مرتب سازی عدد ۱۲۴ در میانه آرایه قرار گرفته و جایگزین خانه مرکزی می شود.

۱-۴ الگوریتم فیلتر پایین گذر[۳]

در این الگوریتم یک دایره به مرکز پیکسل مورد نظر می گیرد و بر اساس یک تابع محاسبه می کند که کدام یک از نقاط داخل دایره قرار می گیرند و کدامیک خارج ایره نقاطی که داخل دایره هستند را در نقطه مرکزی ضرب می کند و نتیجه را باز می گرداند، این الگوریتم بیشتر برای تقویت نقاط تضعیف شده تصویر کاربرد دارد کارایی که ما از آن می خواهیم را نخواهد داشت.

۱-۵ الگوریتم میانگینی از چندین تصویر[۴]

این الگوریتم زمانی به کار می رود که چندین نمونه از یک تصویر داشته باشیم و به این صورت عمل می کند که با مقایسه این تصاویر و یافتن نقاط متناظر در تصاویر آنها را به عنوان نتیجه قرار می دهد و بهترین گزینه را از بین چندین تصویر انتخاب می کند و مهمترین مشکل این الگوریتم نیز در دسترس نبودن چندین نمنه از یک تصویر می باشد.
بنا به عملکرد الگوریتم Median و نوع استفادهای که مورد نیاز ما که بیشتر مقصود حذف نویز از تصاویر و مقایسه الگوریتمهای سریال و موازی و تاثیر حجم پردازش و حجم داده در روند موازی سازی بود الگوریتم مناسبی به نظر می رسد .

۱-۶ الگوریتم نقطه میانی[۵]

این الگوریتم مانند الگوریتم Median ابتدا مقادیر پیکسل های داخل پنجره را مرتب کرده و سپس کوچکترین عنصر را با بزرگترین عنصر جمع کرده و میانگین آن دو را بر می گرداند.

---

۳. Low pass Filter
۴. Averaging OF Multiple Image
۵. Midpoint

فصل دوم

پیاده سازی الگوریتم میانگین همسایه ها

در این قسمت می توان مسئله را به دو قسمت واحد حافظه و واحد محاسبات تقسیم کرد.

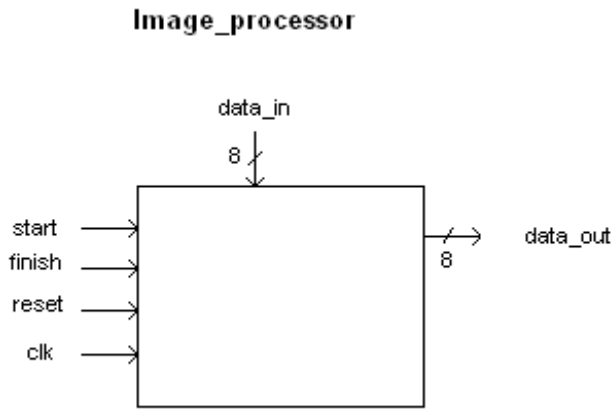
اجزای تشکیل دهنده ی این پروژه :

۱. Memory

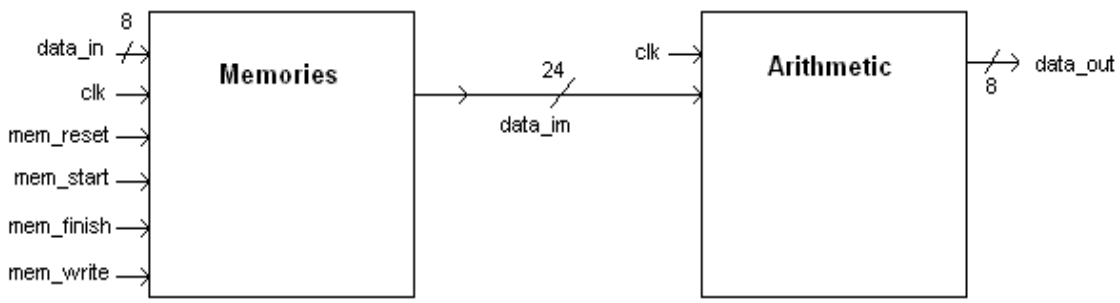
Write in Memory.1.1

Read of Memory.2.1

Arithmetic.2



شکل ۱-۲: black_box(top_level_module)

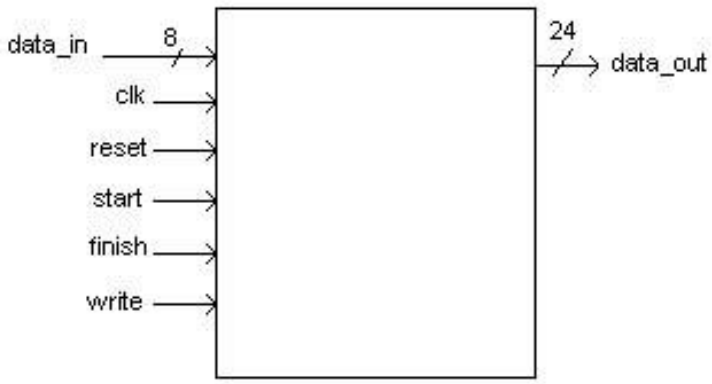


شکل ۲-۲: مسیر داده ماجول اصلی

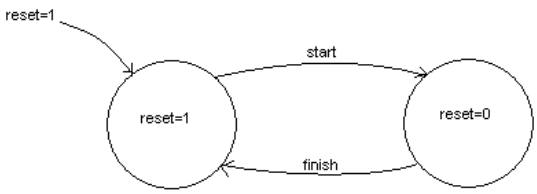
۱.حافظه :

در ابتدا به بررسی ماژول حافظه خواهیم پرداخت. در حالت کلی *Memory* با ۲ سیگنال *Start* و *Finish* در یک *State Machine* کنترل می شود. که در ابتدا با *Reset* کلی سیستم وارد *State Machine* شده که در این *State* , *reset=1* است. پس تمام سیگنالها *Reset* می شوند و با سیگنال *Start = 1* , به *State* دیگر رفته که *reset=* ۰ بوده و *Write* روی *Memory* صورت می گیرد و و با سیگنال *Finish* = ۱ دوباره *reset=1* شده و *Write* پایان یافته و *Read* داریم.

RAM Module:



Memory Unit Controller

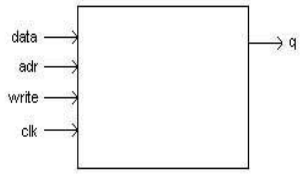


شکل ۳-۲: *state machine* جهت کنترل عملیات در *Memory*

1.1 نوشتن در حافظه

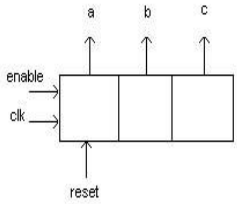
در این حالت دو سیگنال start و we(write enable) وجود دارد. در واقع وقتی که این دو سیگنال همزمان یک می شوند نوشتن روی Memory آغاز میشود که این کار به وسیله کنترلر ماجول اصلی انجام می شود. در ابتدا با کامپوننت های موجود در Memory آشنا می شویم:

Memory Unit:
Port map (data, adr, write, clk)



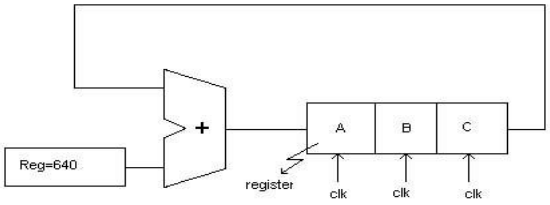
write	clk	Comment
0	↑	g = data(adr)
1	↑	data(adr) = q

3Bit Shift Register:
Port map(reset, enable, clk, out(2 downto 0))



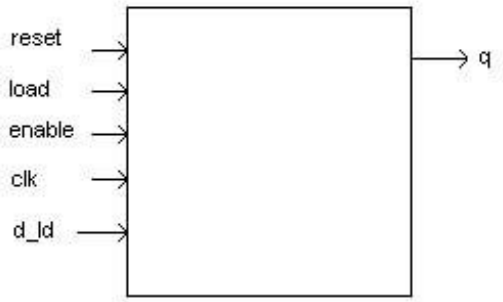
reset	en	clk	
1	x	x	out = 001
0	0	x	out = out
0	1	↑	out=(cout(1 downto 0),cout2)

Offset:
Port map(reset, clk, a, b, c)



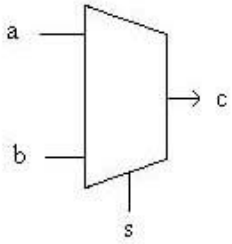
reset	clk	a	b	c
1	x	0	0	0
0	↑	c + 640	a	b

Counter:
Port map(reset, load, en, clk, d_ld, q)



reset	ld	en	clk	q
1	x	x	x	0
0	1	x	↑	d_ld
0	0	0	x	q
0	0	1	↑	q=q+1

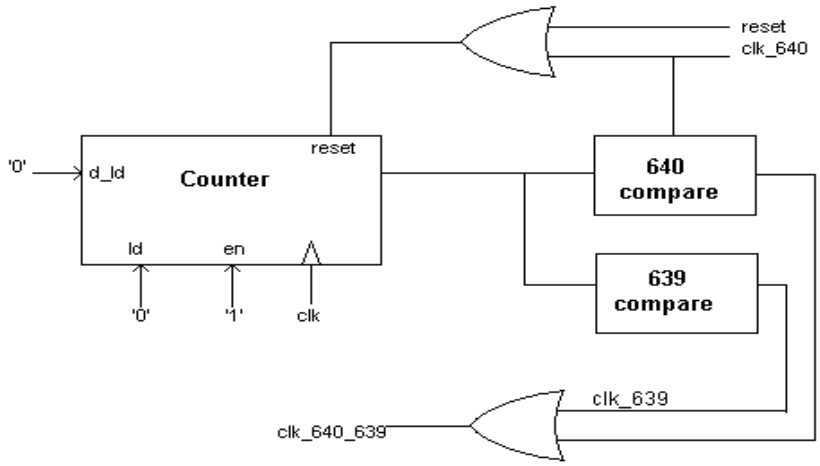
Multiplexer:
Port map(s, a, b, c)



s	c
0	a
1	b

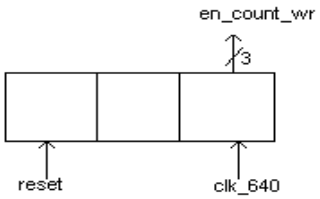
نحوه اتصال و نام گذاری سیگنال ها و کامپوننت در ماجول *Memory* :

Counter_640:

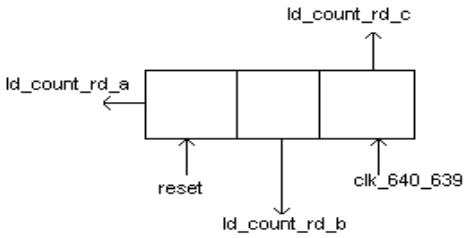


---

shift_reg_3bit:
(for write enable)

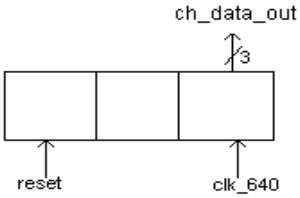


Offset:

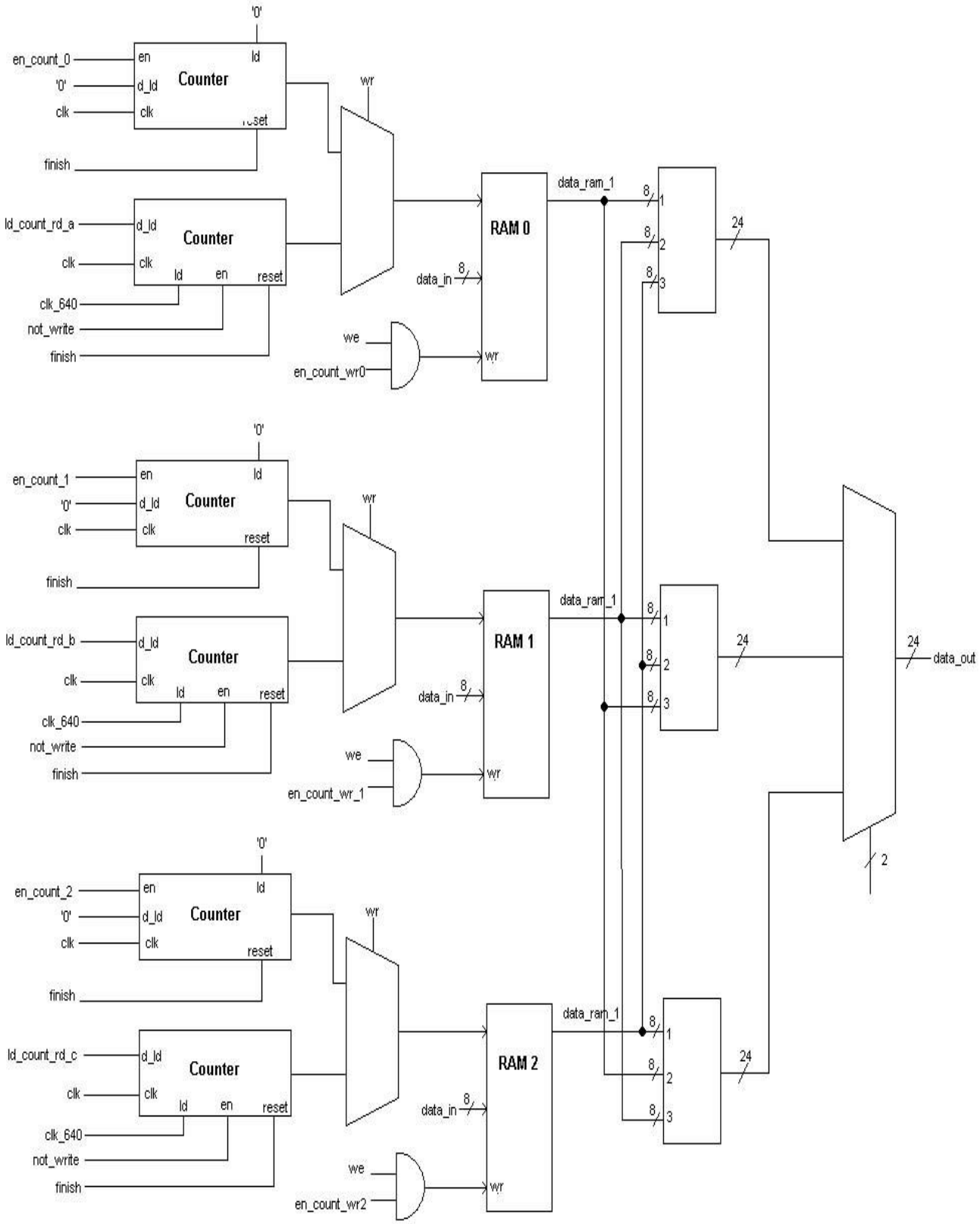


---

shift_reg:
(for mux data_out)



و در این قسمت مسیر داده اصلی *Memory* قرار دارد:



همان طور که گفته شد با یک شدن سیگنال های start و we فرایند نوشتن در ram ها آغاز می شود:

در ابتدا یک شیفت رجیستر سه بیت وجود دارد که برای انتخاب ram جهت نوشتن استفاده می شود که دارای یک خروجی سه بیت می باشد(en_count_wr) که هر کدام از این بیتها جهت مشخص کردن ramکه باید فرایند نوشتن در آن صورت بگیرد می باشد(و به enable ، شمارنده ram1 نیز متصل می باشد) و clk این ماجول به clk_640 که هر ۶۴۰ بار پس از یک شدن start یک با یک می شود وصل می شود.این شیفت رجیستر به مقدار (۰۰۱) ریست می کند یعنی در ابتدا با این مقدار شروع به شیفت دادن می کند که برای اولین شیفت به مدت ۶۴۰ کلاک یا به عبارتی تا یک شدن clk_640 طول می کشد.در طی این مدت شمارنده ram1 میشمارد و data_inرا در حافظه ذخیره می کند تا لحظه ای که clk_640 فعال شود که در این حالت شیفت رجیستر یک شیفت پیدا می کند(در طول این مدت ram2,ram3 در حالت read قرار دارند و شمارنده نوشتن آنها غیر فعال یعنی enable شمارنده نوشتن آنها صفر می باشد).

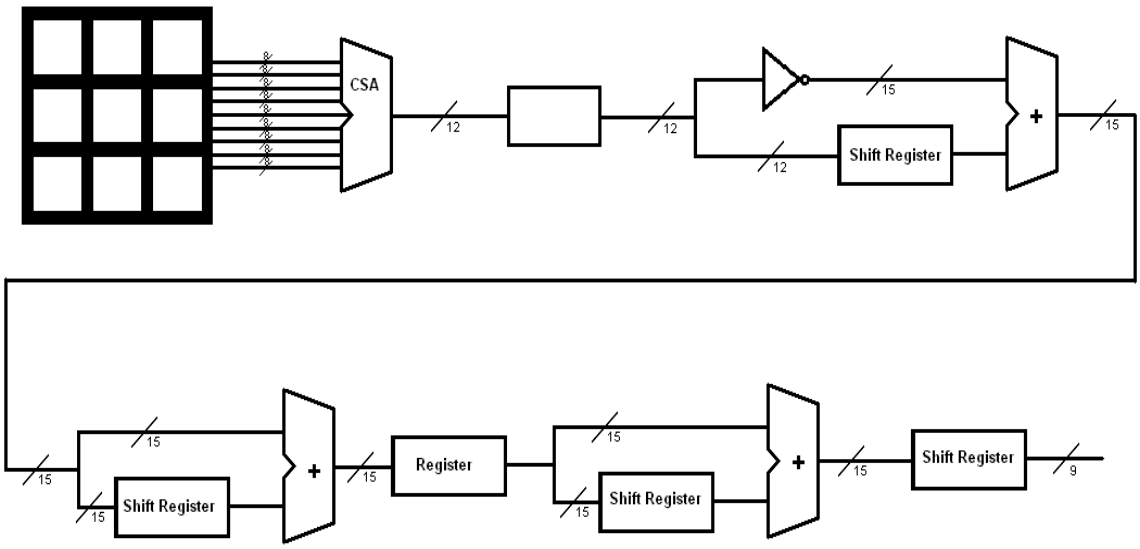
حالا که شمارنده یک شیفت پیدا کرده است ram2 و شمارنده ram2 فعال می باشند و دو ram دیگر در حالت خواندن و شمارنده های آنه غیر فعال می باشد تا لحظه ای دوباره clk_640 فعال شود در این لحظه ram3 فعال ودو ram دیگر غیر فعال هستند تا دوباره clk_640 فعال شود.

البته باید به این نکته توجه داشت که وقتی دوباره clk_640 فعال می شود ram1 در ادامه اطلاعات قبلی (۶۴۰ داده قبلی) از خانه شماره ۶۴۱ فرایند نوشتن را ادامه می دهد و این الگوریتم به همین شکل ادامه پیدا میکند تا زمانی که سیگنال finish فعال شود که پس از فعال شدن آن عملیات خواندن شروع می شود.

1.2 خواندن از حافظه

بعد از اتمام پروسه نوشتن روی RAM ها سیگنال *Finish = 1* و با این کارتمام *Controller*ها *Reset* میشوند و دوباره *Start = 1* و این در حالی است که *write* = ۰ است و به این معنا است که ما در حال حاضر در ***Read*** هستیم. در این قسمت از یک offset برای بدست آوردن محل خواندن استفاده می شود که در هر ۶۴۰ ،clk ویا به عبارتی وقتی clk_640 فعال می شود این افست ها در شمارنده های خواندن لود می شوند و خود شمارنده ها تا انتهای سطر شمارش می کنند و در انتها خروجی ram ها بصورت چرخش در سه حالت در خروجی قرار می گیرد .که به وسیله یک مالتی پلکسر سه ورودی که به صورت چرخشی ورودی های خود را انتخاب می کند و د رخروجی می گذارد.

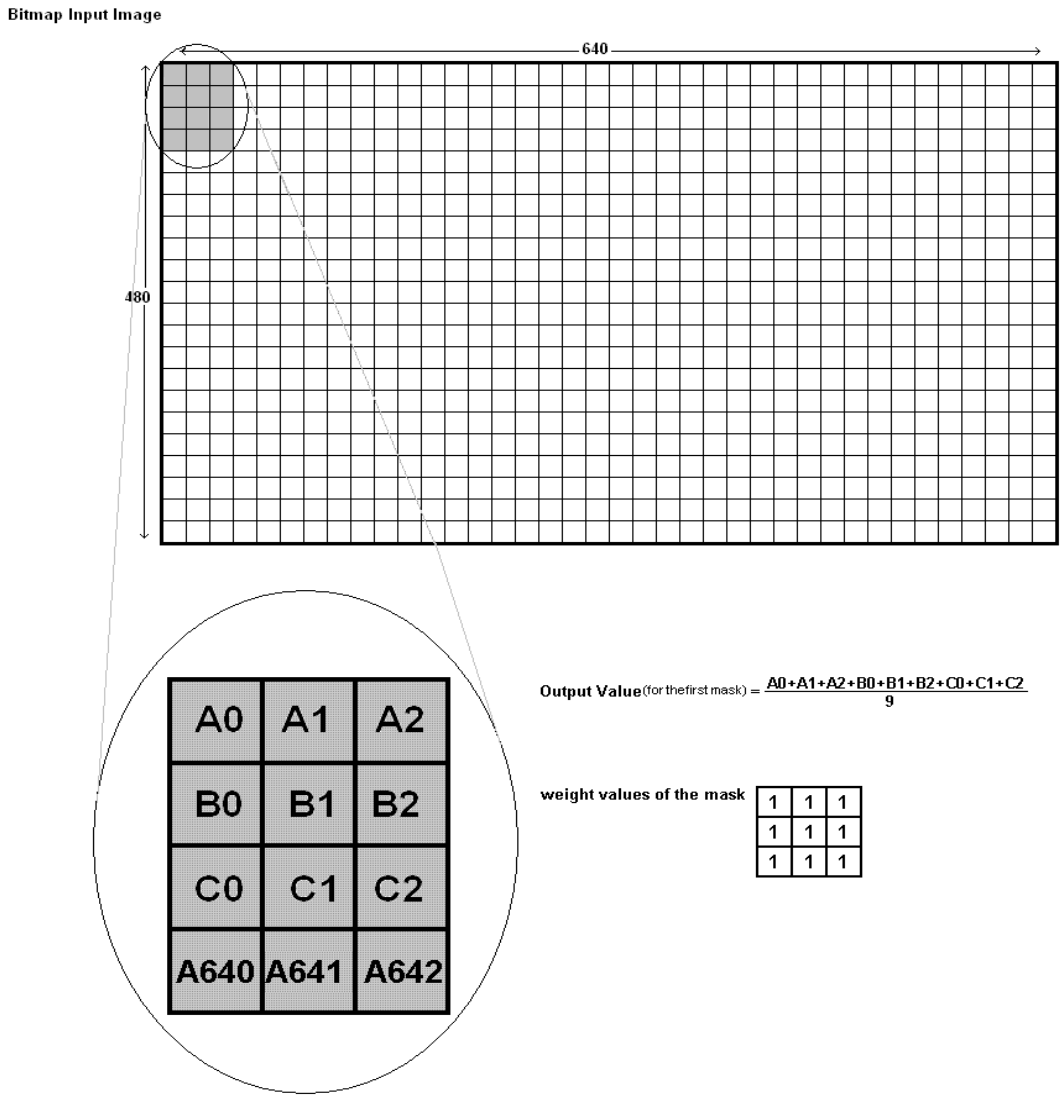
2. واحد محاسبه:



مسیر داده واحد محاسبه

2.1. الگوریتم پردازش تصویر:

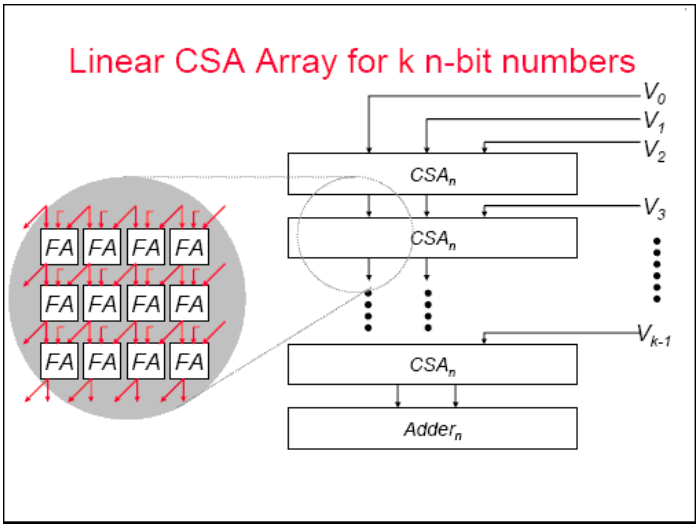
الگوریتمی که ما پیاده سازی کردیم زیر فضای متوسط تصویر را به ما می دهد که دراصطلاح تصویر را نرم میکند این الگوریتم ***Smooth*** نام دارد. که توسط یک Mask که تمام ضرایب آن ۱ هست روی تصویرحرکت می کند. روال کار به این صورت است که ۹ Pixel ی که هر سری خوانده می شوند در یک Mask که تمام ضرایب آن ۱ هست ضرب شده و مجموع این ۹ Pixel بر عدد ۹ تقسیم شده و حاصل این تقسیم در خانه وسط ذخیره می شود و کل تصویر به همین منوال پیمایش می شود و در آخر تصویر نرم شده را ما درخروجی خواهیم داشت.از این به بعد تصویر ما قابلیت پذیرش اغلب الگوریتم پردازش تصویر را دارد. برای پیاده سازی محاسبات این الگوریتم که شامل جمع و تقسیم است از متد های زیر استفاده می کنیم:



2.1.1.الگوریتم Carry-Save Adder (CSA):

این الگوریتم در این مسئله برای جمع ۹ رجیستر که هر کدام شامل bit ۸ داده هست مورد استفاده قرار میگیرد.

نمای از یک csa:



۲,۱,۲. الگوریتم تقسیم:

در این نوع خاص که تقسیم بر عدد ثابت می باشد، ما از یک روش خاص تقسیم بر عدد ثابت استفاده کردیم.
برای پیاده سازی تقسیم بر عدد ۹ از این روش می توان استفاده کرد که یک متغییر s را در نظر می گیریم
که باید بر عدد ۹ تقسیم شود.

s / 9 ==> 7s / 7*9 = 7s / 2^6 -1 = 7s / 2^6 (1- 2^ -6)

9 * x = 2 ^ n -1 ==> x =7 ==> 7*9 = 63 = 2^6 -1

1 / 2^6 -1 = (1+ 2^ -6) (1+ 2^ -12) (1+ 2^ -24) ...

===> s /9 =s1 (1+ 2^ -6) (1+ 2^ -12) (1+ 2^ -24) ...

s1 = 7s = (8s -s) = SLL(s , 3) - s ---> Divider_part1

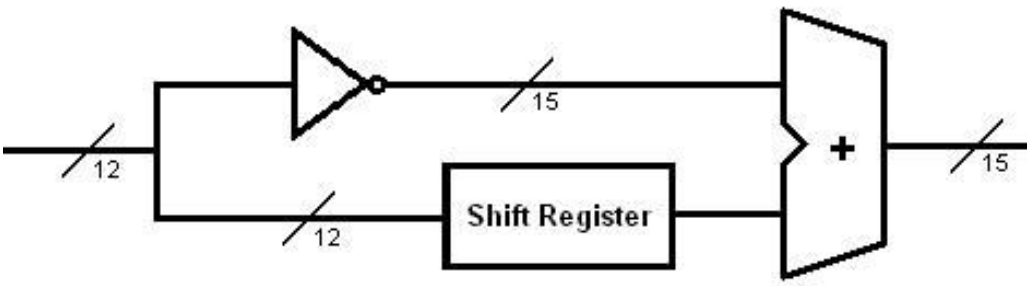
s2 = s1 + SLR(s1 , 6) ---> Divider_part2

s3 = s2 + SLR(s2 , 12) ---> Divider_part3 (contain s4)

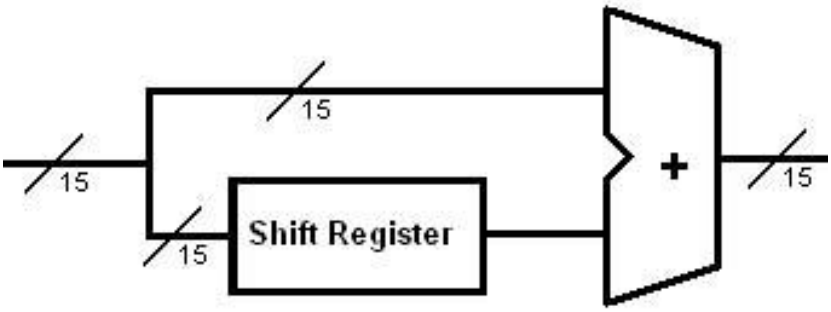
s4 = SLR(s3 , 6)

Note : s = sum of 9 pixle

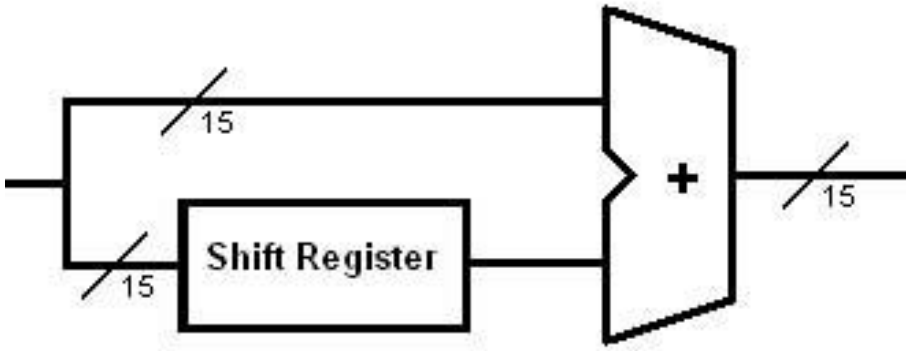
Divider_part1



Divider_part2



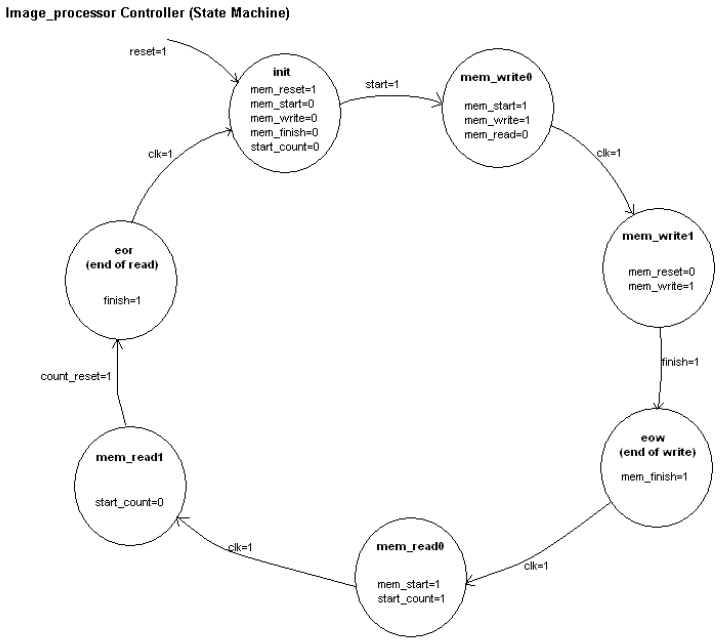
Divider_part3

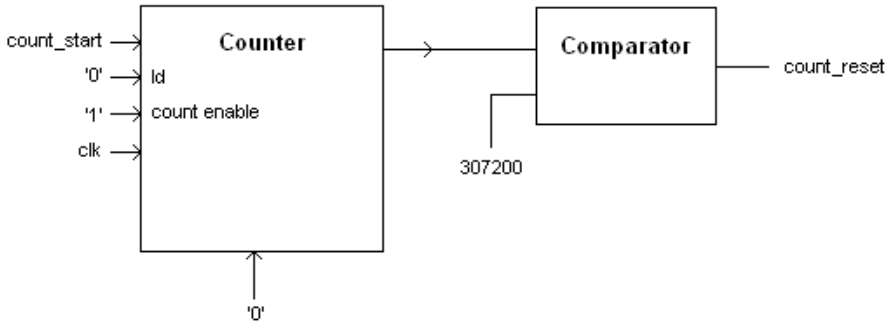


واحد کنترل:

برای کنترل کردن روند کل پروژه و برقراری ارتباط بین Memory و Arithmetic به یک کنترلر احتیاج داریم که توسط یک *State Machine* طراحی شده.

روند کار به این صورت است که با سیگنال Reset = 1 این *State Machine* از *Current = init* کار خود را آغاز می کند.وقتی *Start = 1* و *Current = mem_write0* حال با *clk* به *State* بعد می رود.حال *Current = mem_write1* است که با سیگنال *Finish*= 1 به *Current =End Of Write* که در این وضعیت ***Mode Write*** به پایان رسیده و وارد ***Mode Read*** می شویم پس با *clk* به *State* بعد می رویم حال *Current = mem_read0* وبا یک *clk* به *State* , *mem_read1* می رود و در این *State* ۳۰۷۲۰۰ *Pixle* (۶۴۰*۴۸۰) را می خواند و پردازش روی تصویر صورت می گیرد سپس با *clk* به *State* بعد می رویم حال *Current =End Of Read* که در این وضعیت ***Mode Read*** به پایان رسیده و با *clk* به *State* بعد *Current = init* می رویم.





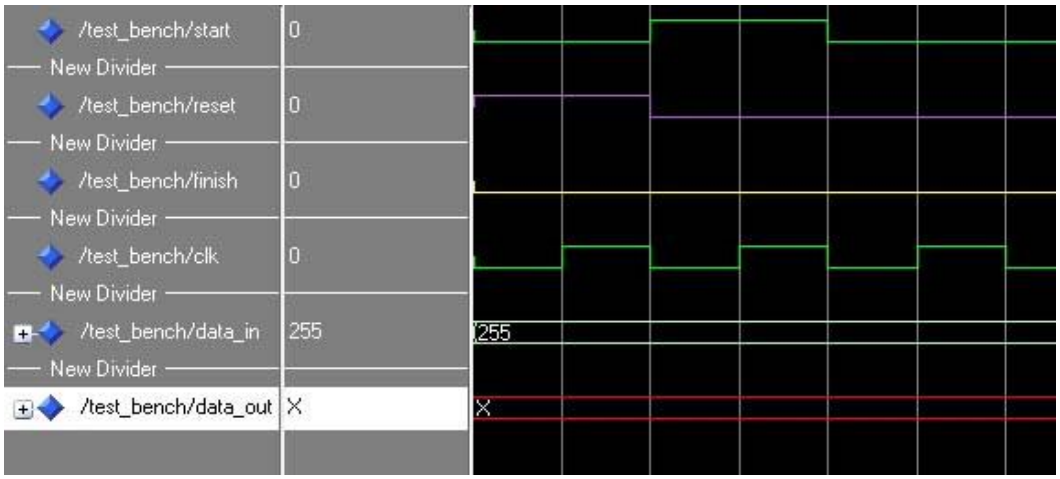
-آزمایش کردن ماژول

همان طور قبلا ذکر شد، جهت آزمایش کردن ماجول که در طراحی کرده ایم. سیگنال های کنتری وجود دارد(*reset,start,finish*) که از این سیگنال ها برای ریست کردن ماجول و آماده کردن برای دریافت و پردازش استفاده می شود. برای تست این ماجول از یک *test_bench* استفاده شده است که در زیر به نحوه ای عملکرد *test_bench* می پردازیم(به دلیل لود بالای *quartus* و وضوح و سادگی نرم افزار ***model sim*** در آزمایش ماجول از نرم افزار *model sim* استفاده شده است، ولی این ماجول قابل سنتز و آزمایش در *quartus* می باشد.):

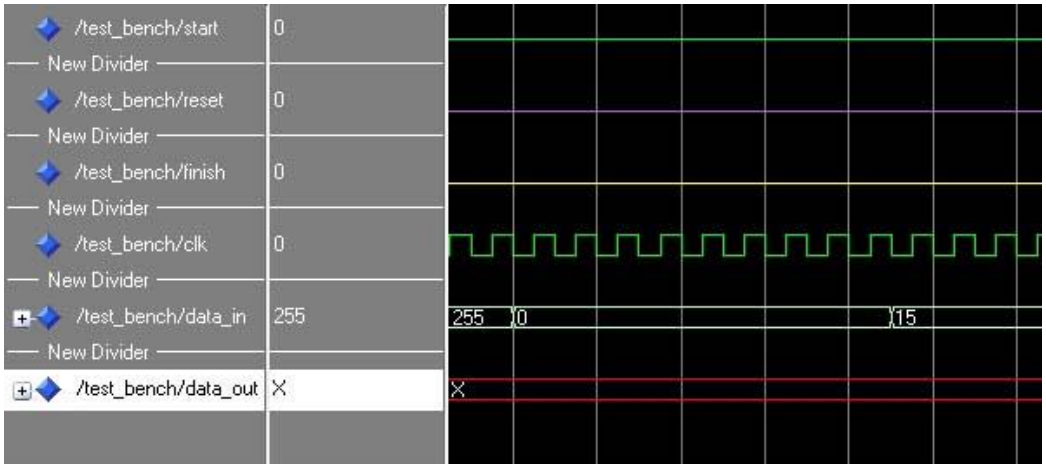
مراحل تست:

۱. در ابتدا سیگنال *reset* را یک می کنیم.

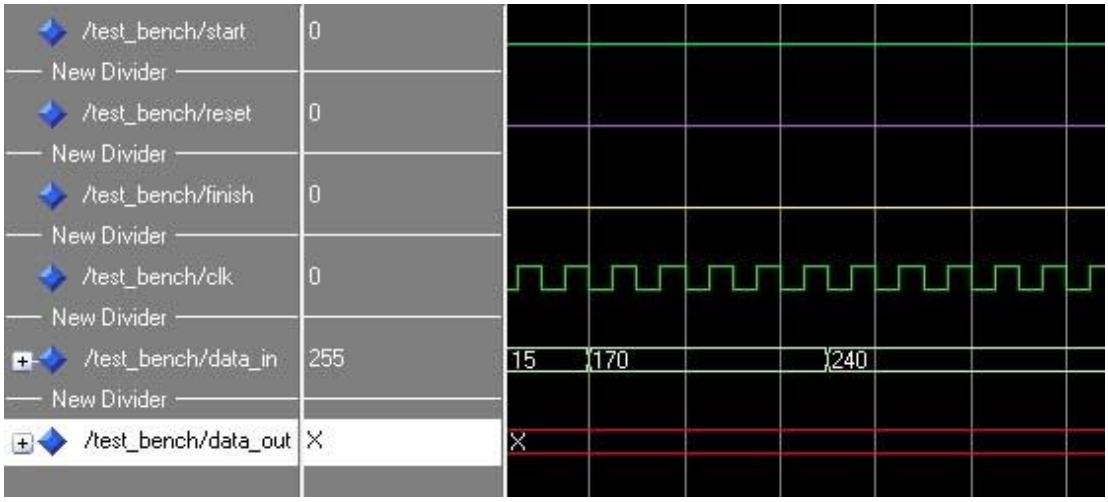
۲. سپس اطلاعات ورودی را آماده کرده وبا یک کردن سیگنال *start* ماجول با هر *clk* یک خانه از خانه های حافظه خود را پر می کند.



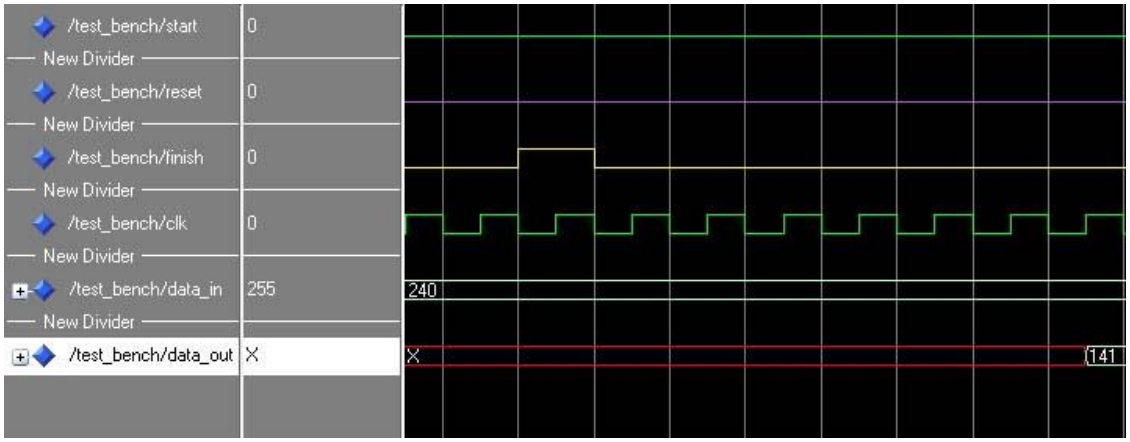
۳. در ای مرحله اطلاعات سطر اول که همه مقدار ۲۵۵ را دارا میباشند وارد می شوند بعد از ۶۴۰، *clk* اطلاعات سطر دوم وارد میشوند.



۴. در این مرحله اطلاعات مربوط به سطر سوم و در ادمه سطر چهارم و پنجم وارد میشود.



۵. در آزمایش این ماجول تست پنج سطر جهت آزمایش صحت عملکرد ماجول کافی می باشد بدین جهت تنها از پنج سطر جهت آزمایش استفاده شده است و در سطر پنج سیگنال کنترلی *finish* فعال شده است، بدین معنی که اطلاعات ورودی به پایان خود رسیده است. ملاحظه می شود که به دلیل سختار حافظه و پایپ لاین(*pipeline*) بودن حافظه خروجی ماجول بهد از هفت *clk* آماده می شود.



حال به بررسی اطلاعات ورودی و اطلاعاتی که باید در خروجی می پردازیم.

data_in	0	1	2	3	4	5	6	7	8	9	10	11	12	13	….	638	639
0-639	255	255	255	255	255	255	255	255	255	255	255	255	255	255	….	255	255
640-1279	0	0	0	0	0	0	0	0	0	0	15	15	15	15	….	15	15
1280-1919	170	170	170	170	170	170	240	240	240	240	240	240	240	240	….	240	240
1920-2559	240	240	240	240	240	240	240	240	240	240	240	240	240	240	….	240	240
2560-3199	240	240	240	240	240	240	240	240	240	240	240	240	240	240	….	240	240
…	U	U	U	U	U	U	U	U	U	U	U	U	U	U	….	U	U

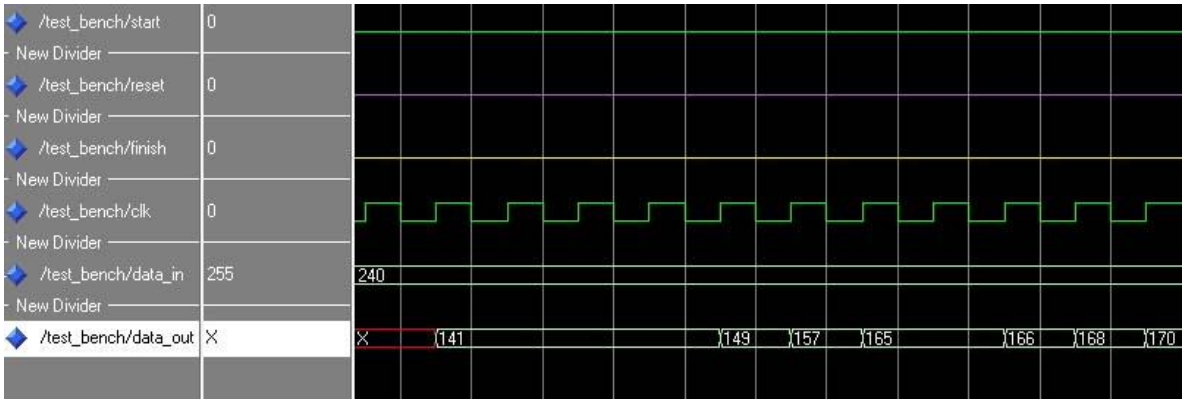
همین طور که در جدول بالا ملاحظه می شود اطلاعات به صورت سری وارد ماجول می شود و ۰-۶۳۹ دیتا اولیه در حافظه **A** و ۶۴۰-۱۲۷۹ در حافظه **B** و ۱۲۸۰ -۱۹۱۹در حافظه **C** قرار می گیرد و دیتا ۱۹۲۰-۲۵۵۹ در حافظه **A** و ۲۵۶۰-۳۱۹۹ در حافظه **B** قرار می گیرد و اطلاعات به همین شکل تا انتها در حفظه ها قرار می گیرند.

ما می توانیم پیش بینی کنیم چه اطلاعاتی را باید در خروجی ظاهر شود:

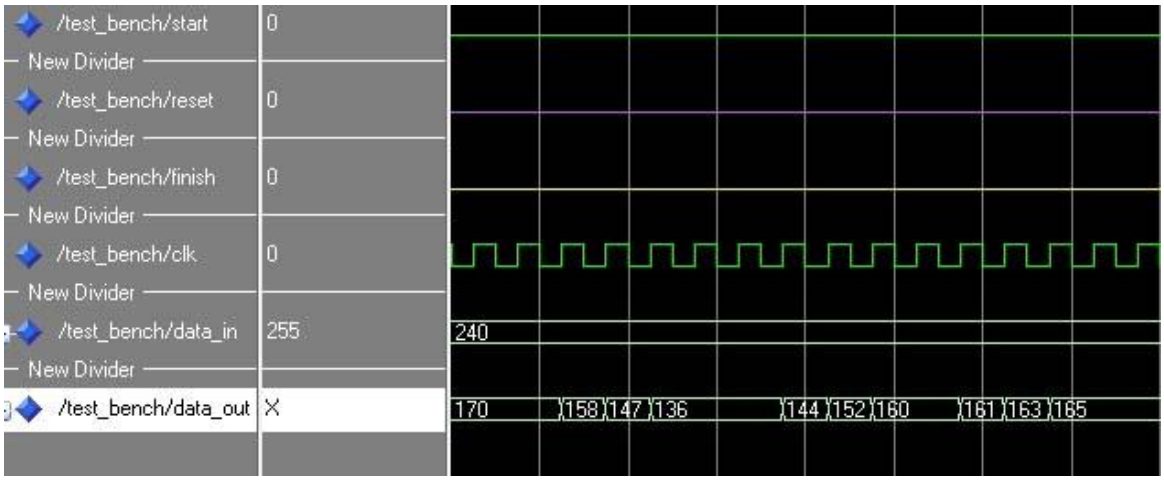
data_out	0	1	2	3	4	5	6	7	8	9	10	11	12	13	…	638	639
0-639			141	141	141	141	149	157	165	165	166	168	170	170	…	170	170
640-1279	158	147	137	137	137	137	144	152	160	160	161	163	165	165	….	165	165
1280-1919	182	199	216	216	216	216	224	232	240	240	240	240	240	240		240	240
…	U	U	U	U	U	U	U	U	U	U	U	U	U	U	…	U	U

اطلاعات باید به ترتیب بالا ظاهر شوند.

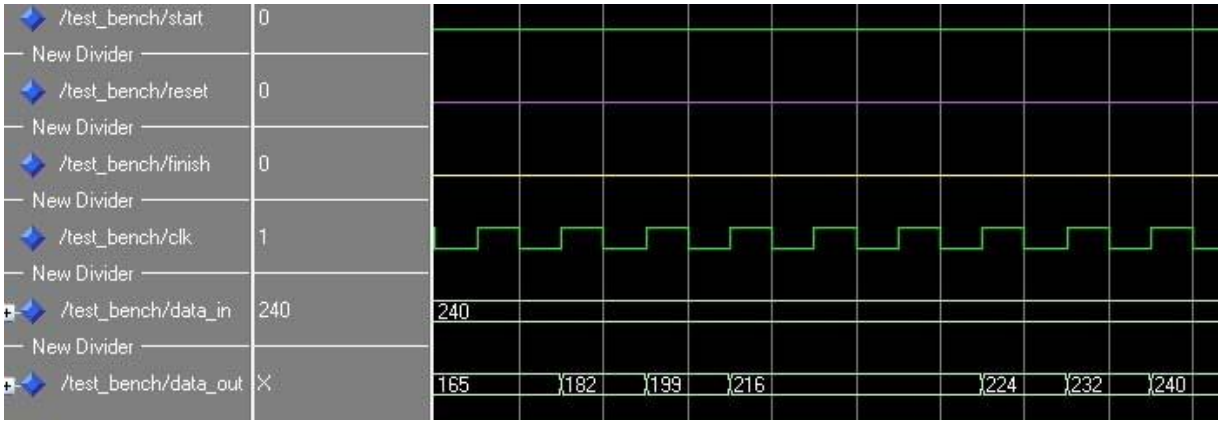
۶. اطلاعات خروجی سطر اول:



۷. اطلاعات خروجی سطر دوم:



۸. اطلاعات خروجی سطر سوم :



فصل چهارم: نتیجه گیری

نتیجه گیری:

با توجه مطالب گفته شده هدف رسیدن به قدرت پردازش بیشتر جهت پردازش های بلادرنگ می باشد. حداکثر تاخیر در نظر گرفته شده برای این مدار تاخیر یک جمع کننده بوده است که بوسیله مجموعه الگوریتم های محاسباتی مانند جمع کننده های چند عملوندی، الگوریتم تقسیم بر عدد ثابت و همچنین تکنیک های پایین موفق به پیاده سازی شدیم.

منابع

- Rafael C. Gonzalez, Richard E. Woods, Digital Signal and Image Processing 2nd, Prentice Hall; (January 15, 2002)
- G?rard Blanchet, Maurice Charbit, Digital Signal and Image Processing Using MATLAB (Digital Signal & Image Processing Series, Wiley-ISTE (2006)
- http://www.imageprocessingplace.com