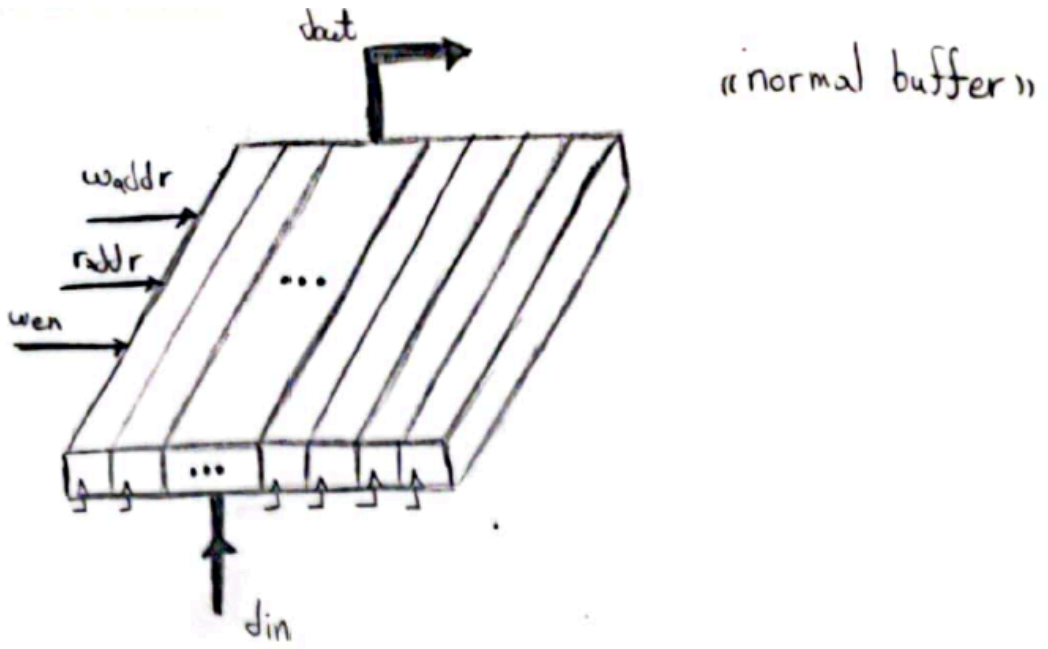
CAD (Computer Aided Design)
CA2

Sobhan Kooshki Jahromi 810101496
Seyed Navid Hashemi 810101549

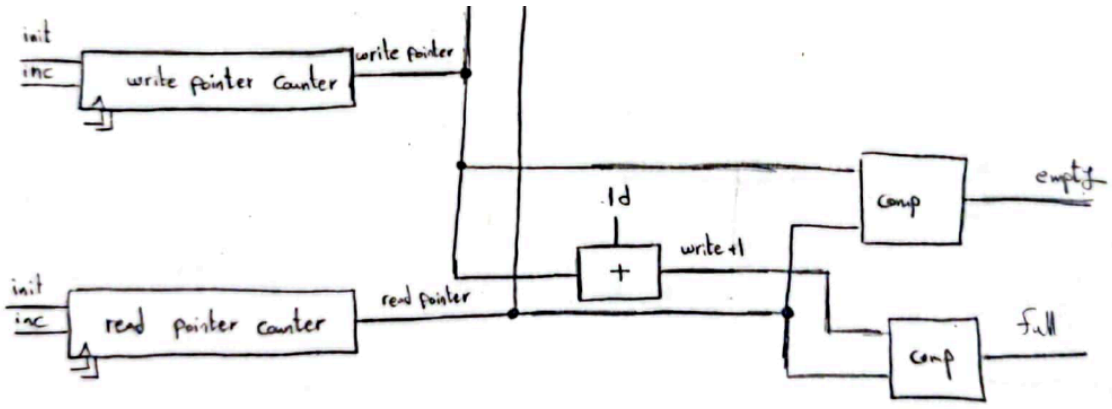
در این پروژه ما به دنبال پیاده سازی بافر FIFO هستیم که به صورت چرخشی عمل میکند برای پیاده سازی این بافر ابتدا ی بافر معمولی را پیاده سازی میکنیم که میتوانیم با دادن ادرس به ان داده ای را در ان بنویسیم یا اینکه با دادن یک ادرس به ان دیتایی از ان بخوانیم.



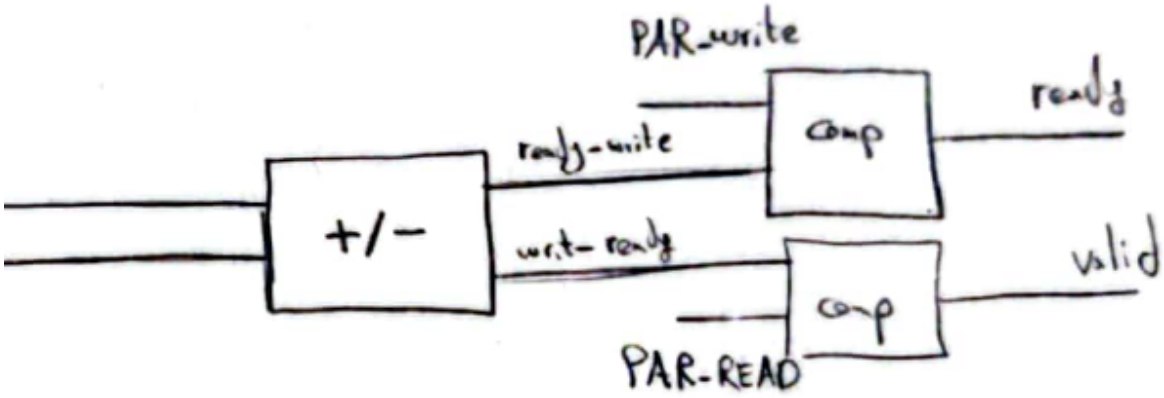
این هسته بافر ما است که بتوانیم همین پیاده سازی را به یک بافر FIFO چرخشی تبدیل کنیم.
برای تبدیل این طرح به یک بافر FIFO چرخشی ما نیاز به چند کامپوننت دیگر و چند تغییر کوچک در این بافر داریم.

DATAPATH

برای اینکه ما پوینترهای write و read را نگه داریم برای ان ها یک counter در نظر گرفته ایم که وقتی بخواهیم مقدار این پوینتر ها را به اندازه PAR_WRITE یا PAR_READ زیاد کنیم دستور inc را به این counter ها میدهیم و ان ها میتوانند به صورت درستی این مقدار را اضافه کنند و اگر این مقدار جدید از تعداد خانه های حافظه بیشتر بود mod ان ها را بگیرند و به ابتدای ادرس ها بیایند
برای هندل کردن empty , full در این طرح ما میاییم و یک خانه حافظه را در طرهمان اضافه میگیریم حالا با تعاریف های زیر میتوانیم راحت این دو سیگنال را خروجی دهیم
Full : اگر write_pointer + 1 = read_pointer ما این سیگنال را یک میکنیم
Empty: اگر write_pointer = read_pointer ما این سیگنال را یک میکنیم
با این تعریف هایی که کردیم datapath ما به شکل زیر میشود.



حال به سراغ هندل کردن نوشتن و خواندن از این بافر میرویم.
طرح ما به صورت زیر است.

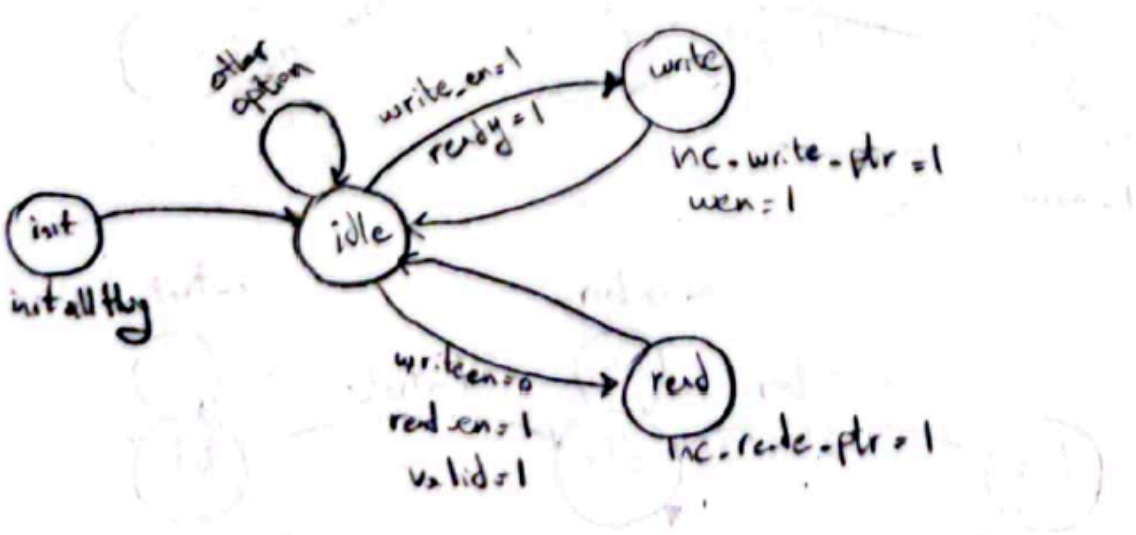


ان دو مقدار که به +/- امده اند همان write_pointer و read_pointer است
ما زمانی میتوانیم میتوانیم در این بافر بنویسیم که بافر ما به اندازه PAR_WRITE ظرفیت داشته باشد و زمانی میتوانیم از ان بخوانیم که به اندازه PAR_READ داده در ان داشته باشیم این دو شرط را ما با استفاده از سیگنال های valid , raedy درست میکنیم.
حالا روش کار این است که ما می ابیم با منها کردن read_pointer از write_pointer فاصله این دو پونتر را میسنجیم ببینیم چقدر جای خالی داریم اگر این مقدار منفی شد ب اندازه تعداد خانه های حافظه به ان اضافه میکنیم. سپس تعداد این فضاهای خالی را با PAR_WRITE مقایسه میکنیم اگر بزرگتر مساوی با ان بود دستور ready را یک میکنیم در غیر این صورت صفر است.
برای valid هم دقیقا این موضوع وجود دارد. ما می آییم و write_pointer را ازread_pointer کم میکنیم اگر منفی هم شد با تعداد خانه های حافظه اضافه میکنیم. این به ما تعداد خانه های دارای دیتای معتبر را نشان میدهد و این مقدار را با PAR_READ مقایسه میکنیم اگر بزرگتر یا مساوی بود valid را یک میکنیم در غیر اینصورت صفر نگه میداریم.
کار ما در datapath تمام شده است.

CONTROLLER

در بخش کنترلر ما در فاز اول اشتباهاتی داشتیم که با هدف پروژه در تناقض بود و مجبوربه اصلاح این بخش شدیم
در چیزی که از ما خواسته شده در این پروژه این بوده است که ما بتوانیم در دو clk پشت سر هم مقادیر را در بافر بنویسیم و فاصله ای بین این دو عمل نباشد و دقیقا همین موضوع

برای خواندن از بافرهم صادق است. در controller قبلی که در شکل پایین میبینیم این موضوع درست نبود چون ما با استفاده از یک clk باید از دوباره به استیت ابتدایی برویم تا بتوانیم تصمیم دیگر بگیریم و این موضوع تناقض با نوشتن در دو clk متوالی بود.
یک اشکال دیگر که طرح زیر داشت این بود که ما اولویت نوشتن را بالاتر از خواندن در نظر میگرفتیم این باعث میشود که نتوانیم همزمان هم بخوانیم و هم بنویسیم که این هم با اهداف پروژه ناسازگار بود.



ما با استفاده از تغییر استیت ماشین moore به استیت ماشین combinational توانستیم هر دو مشکلی که در بالا گفته شده بود را تصحیح کنیم و داده های درستی از تستی که کردیم بدست اوریم کد کنترلر combinational ما به این صورت است

```
assign inc_w = (write_en && ready) ? 1'b1 : 1'b0;
assign wen = (write_en && ready) ? 1'b1 : 1'b0;
assign inc_r = (read_en && valid) ? 1'b1 : 1'b0;
```

در اینجا ما به صورت combinational سیگنال های کنترلری inc_w , inc_r , wen را درست میکنیم و وقتی که کلاک خورده شد حالا میتوانیم مقدار اپدیت شده پونتر ها را داشته باشیم و دیگر نیاز به صبر برای مقدار جدید پوینتر ها در طرح نیست با ایتفاده از این کنترلر ما میتوانیم همزمان هم بنویسیم و هم بخوانیم.

TESTBENCH

با صحبت هایی که کردیم حالا میتوانیم مدار را تست بگیریم و نتیجه در عکس زیر قابل مشاهده است.