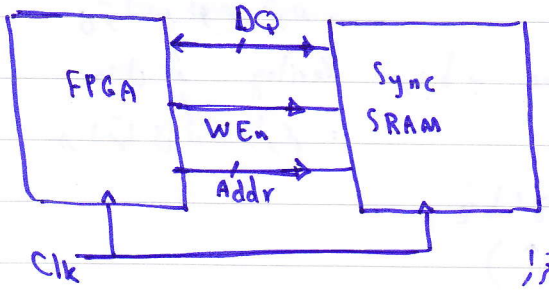
صفحه ۱

تمرین های ۸ و ۹ : کاملا در حد تمرین های امتحان

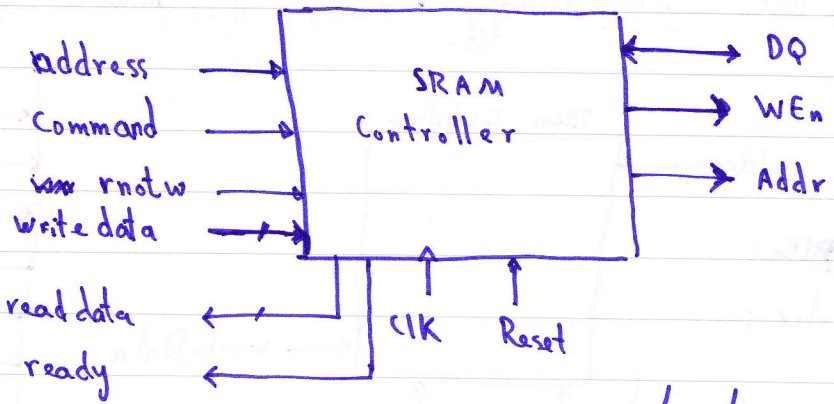
حافظه SRAM فازی که Synchronous هم هست مطابق شکل به FPGA متصل می شود :



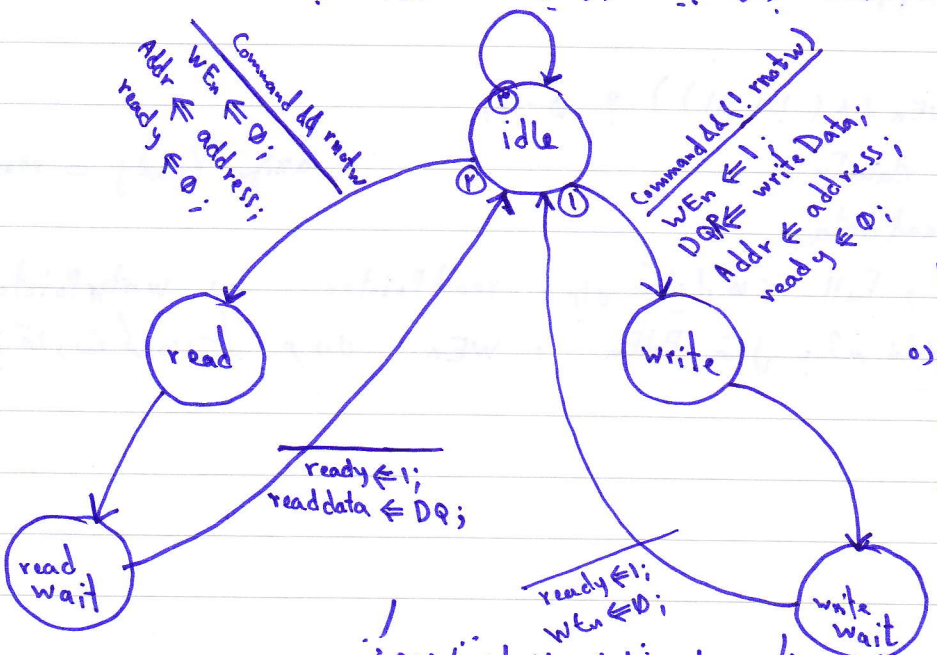
نوع ساده این حافظه SRAM سیگنال های دیگر را ندارد. مثلا سیگنال OE که به همراه آن را فعال نگاه می داریم چون که SRAM به نقطه به FPGA وصل است و روی یک پایه نیست.

فرض شده که تاخیرها صفر است. این فرضی است که در عمل هرگز برقرار نیست و محدودیت های خاصی را به طرح ما اعمال می کند.

حافظه SRAM ارزان و سریع به اندازه بلوک های حافظه داخل FPGA ندارد و ما باید این موضوع را در نظر بگیریم. طرح SRAM Controller داخل FPGA می تواند این شکل باشد :



عملکرد این واحد به این ترتیب است که با دستور read و یا write به آن کار می دهیم. واحد ready را صفر می کند و شروع به انجام task می شود. هرگاه کار خاص شد ready دوباره یک می شود. بدیهی است که یک state machine برای پیاده سازی این واحد بسیار مناسب است.



در این fsm یک ثبات به اندازه عرض حافظه تعریف می کنیم و آن را DQR می نامیم. مقدار داده که قرار است به حافظه نوشته شود را داخل آن می ریزیم.

ممکن است تعداد state ها زیاد شود - برای کاربرد به این حجم سلول حافظه ای داریم استفاده می کنیم.

همانطور که می‌بینید این FSM ساختار بسیار ساده‌ای دارد و بر حسب اینکه حافظه SRAM شما چقدر سریع است و چند سیکل طول می‌کشد تا یک read یا write را کامل انجام دهد، تعداد state ها می‌توانند عوض شوند.

حال کد Verilog مربوط به State Machine را بنویسید (شما سر امتحان حتماً باید کد را بنویسید)

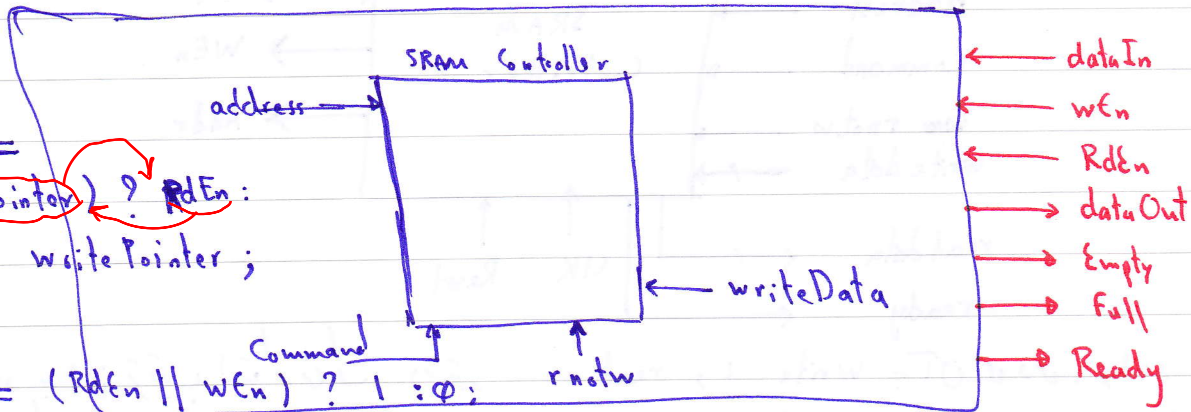
در انتهای کد داریم:

```
assign DQ = (WEn) ? DQR : 8'bz;
```

(مربوط به ↑ نظر دارد)

حال با استفاده از SRAM Controller می‌توانیم یک fifo طراحی کنیم. بدیهی است کسی که از این fifo استفاده می‌کند نمی‌بایست WEn و RdEn را همزمان فعال کند. (بهترین کار این است که کاربر و طراح خودشان واسطشان این موضوع باشد.)

این fifo دارای یک خروجی Ready می‌باشد که هرگاه فعال باشد می‌توان RdEn و WEn را فعال کرد. تا مثلاً سیگنال خروجی Ready فعال نیست کاربر نباید RdEn و یا WEn را فعال کند.




```
assign address =
    (readPointer) ? RdEn :
    writePointer;

assign Command = (RdEn || WEn) ? 1 : 0;
```

البته بهتر بود شرط را اینطور می‌نوشتیم:

```
((RdEn && (!Empty)) || (WEn && (!Full))

assign rnotw = (WEn && (!Full)) ? 0 : 1;
assign writeData = dataIn
assign dataOut = readdata;
assign Ready = ready;
```

برای writePointer و readPointer و برای سیگنال‌های Full و Empty مانند قبل عمل می‌کنیم. با این تفاوت که حالتی که هر دو WEn و RdEn فعال باشند نداریم.