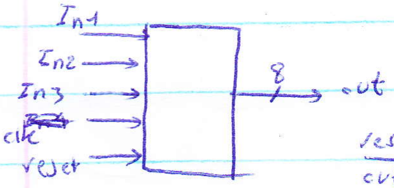
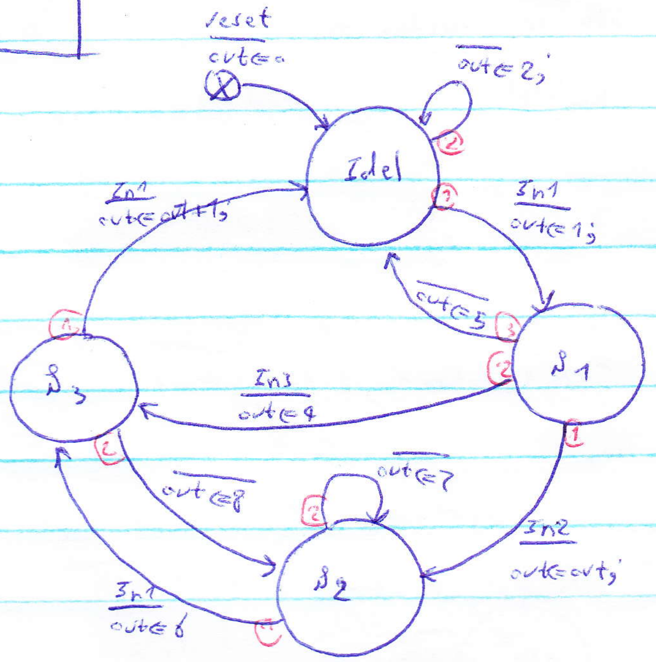
کد HDL ← سه always ← 1. از دو state الان state های بعدی.

2. تو هر state خروجی چی باشه.

3. next state و current state رو بریزه.

کد وریلاگ معادل FSM را بنویسید.





بهتر از با define کلمه چون تعریف کند.

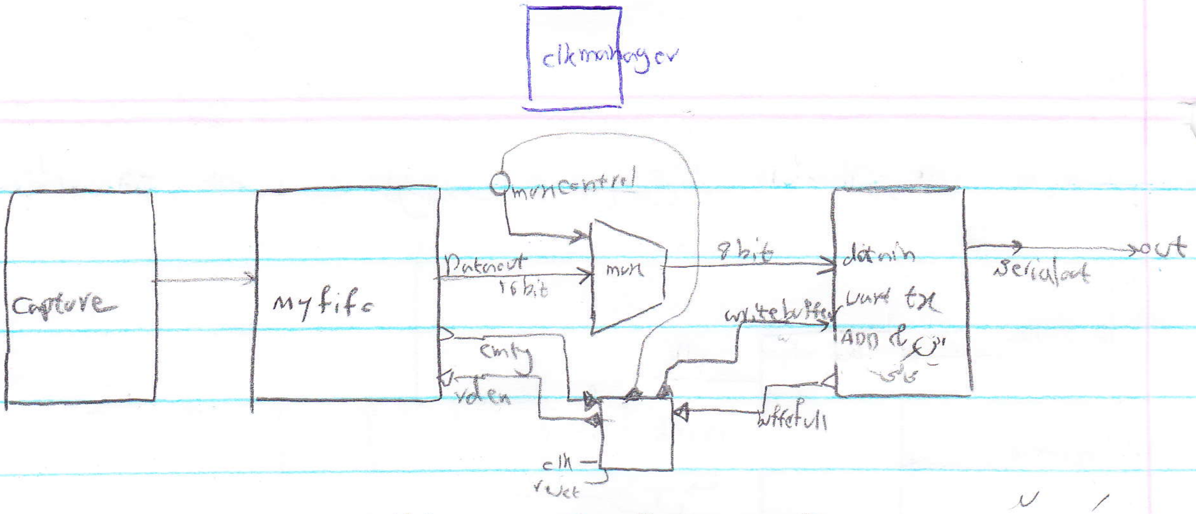
```
`define idle 2'b00;
`define s1   2'b01;
`define s2   2'b10;
`define s3   2'b11;
module
  :
  :
wire [1:0] ns;   reg [1:0] s;
//  nextstate          current state
assign ns = ( (s=='idele) && In1) ? 's1 :
            ((s=='idele)) ? Idel :
            ((s=='s1) && In2) ? s2 :
            ((s=='s1) && In3) ? s3 :
```

```
((s=='s1)) ? 'Idel :
((s=='s2) && In1) ? 's3 :
(s=='s2)
        :
        :
        :
```

} state ها
ادامه بدی

```
always @(posedge clk or posedge reset)
    if (reset) begin s<='idel; out<=0; end
    else begin
        s<=ns;
        case (s)
            'idle : if (In1) out<=1; else out<=2;
            's1 : begin
                    if (In2) out<=1;
                    else if (In3) out<=4;
                    else out<=5;
                  end
            's2 :
               :
               :
        endcase
    end
```

بقیه state ها
ادامه بدی



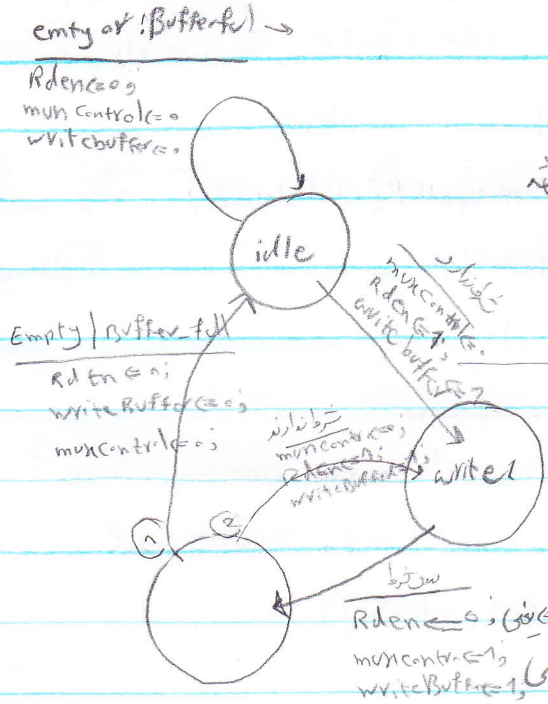
assign datain = (muxcontrol)? Dataout[15:8] : Dat[7:0] ⇐ mux کد وریلاگ

حال باید ماجولی تولید کنی که Rden fifo را بسازیم و muxcontrol و writebuffer را باید پیاده کنی که fifo emty نباشد و باید دقت کنی که uart full نباشد

باید یک ترتیب روی ماجول بکشیم statediagram :

idle: یعنی fifo emty یا uart full باشد

اینجا کاری نداریم.

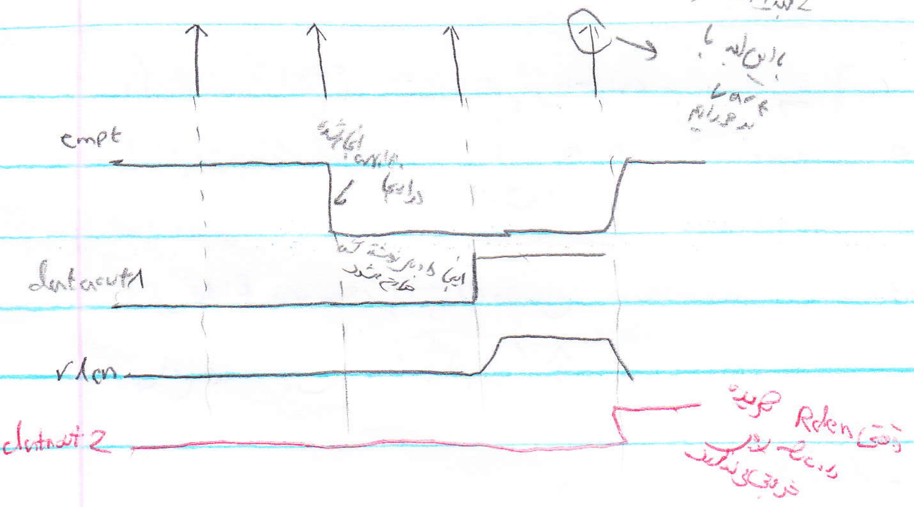


یعنی هنوز ننوشته‌ایم باید دقت کنی که اینجا نباید Rden را 1 کنی.

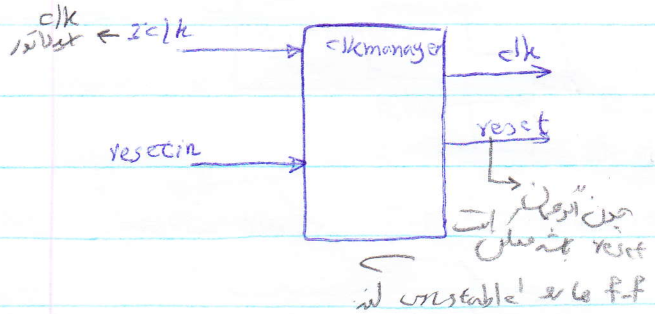
منظور Rden ⇐ اولین داده‌ای که آمده است روی خروجی است تا وقتی Rden برقرار است داده بعدی نمی‌آید ⇐ یعنی این داده را برداشتیم، داده بعدی را می‌گذارد.

2 مثل 1 است حالا

می‌توان به جای مطمئن شدن به حال buffer full از half full استفاده شود.



* کد مربوط به ساختن clk مناسب برای ورودی با فرکانس بیشتر از clk اسیلاتور:




```
wire clko;
wire clkotoBufG;
wire clktoDCM, clkre; wire clkfxtoBufG; reg resetr, resetrr, dummyload;
IBufG IBufGIns (.I (Iclk), .O(clktoDCM))
DCM DCMIns (.clkIn(clktoDcm), .clkfx(clkfxtoBufG)
            .clk0(clk0toBufG)
            .clkFB(clk0));
            .locked());
defparam DCMIns.clkfx_DIVIDE = 5;
defparam DCMIns.clkfx_multiply = 18;
BufG BufGIns (.I (clkfxtoBufG), .O(clk));
always @(posedge clk)
  begin
    resetR <= reset;
    resetRR <= resetR;
  end
assign reset = resetrr;

always @ (posedge clk0)
  dummyload <= ~dummyload;
```

IDL ⇒ Sportatwe

از تو simprims می‌توانی port ها ببینی و بیاری

clk FeedBack

سیگنالی که نشان می‌دهد clk ها valid شده است

$\frac{25 \times 18}{5} = 90$

بخش بازنشانی کد reset

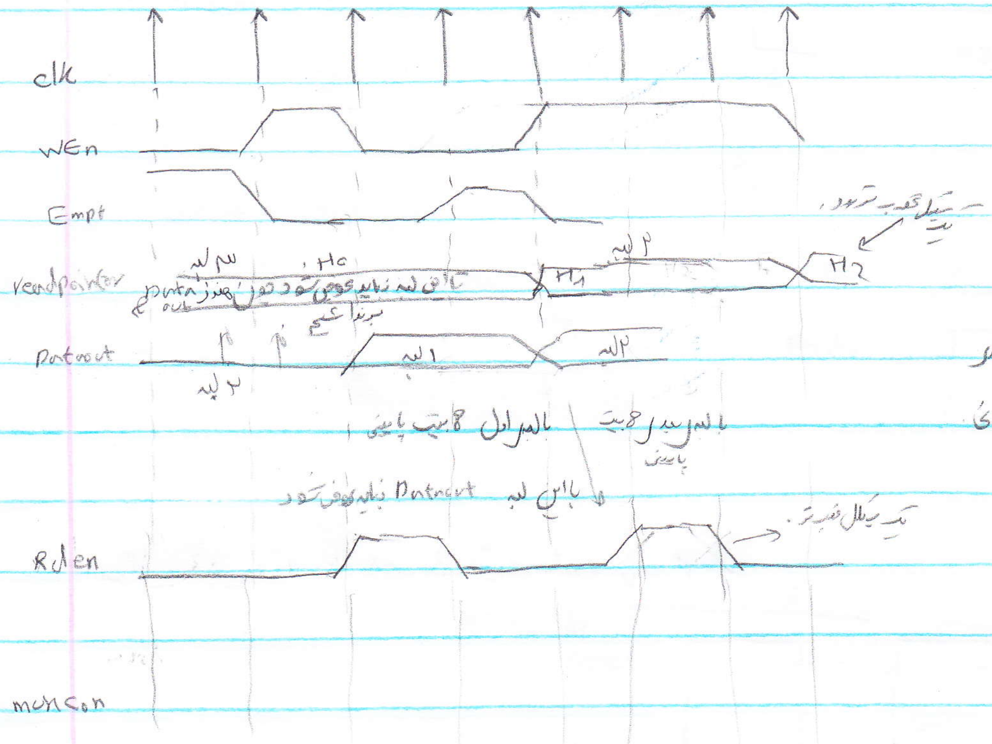
این دو resetin را با synchron 90MHz clk است!

اینکه دو تا پشت سر گذاشتم تا مطمئن شوم همه سیگنال‌ها ثابت شوند تا کار خراب نشود.

در خود folder simprims پارامترها را مشخص می‌کنم ← سپس منتقل می‌کنم در کد

ماکزیمم clk که می‌توان تولید کرد. در III ، 300MHz ، Virtex 600MH

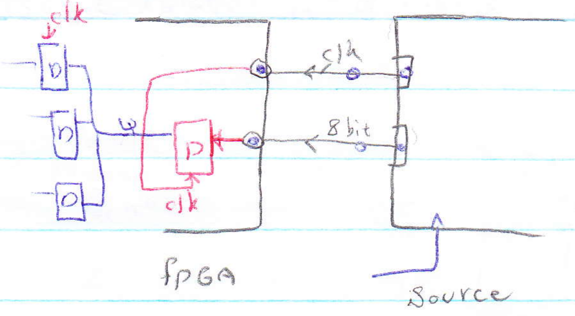
* تکمیل state machine:



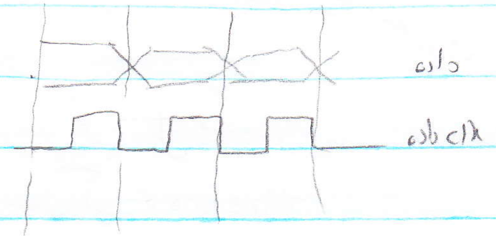
rdEn یک سیکل ۱ نگه دار

و بعد صفر می‌کنی تا ۱۵ بیت جابجا

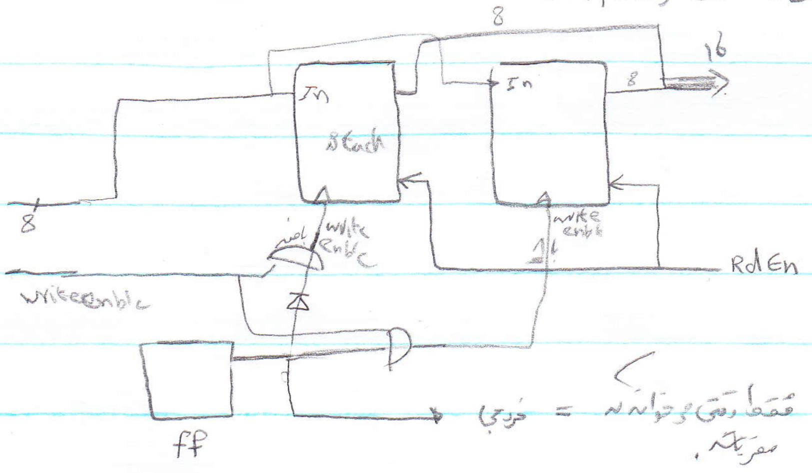
سوال 2: تو امتحان می آید!



تماماً با تأخیر عمل از fpGA است.



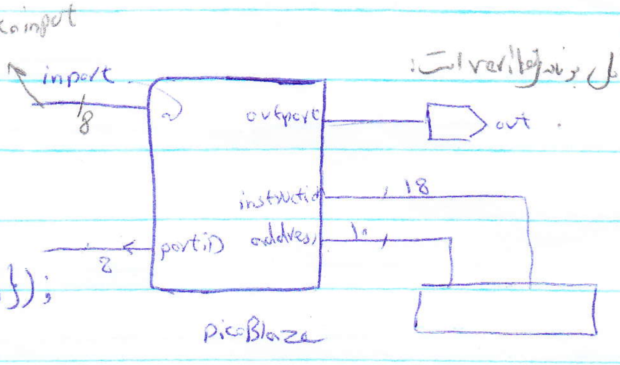
سوال 3: دو تا stack 8 بیتی می سازی و کنارهم بگذار.



سوال 4: این کل برنامه verilog است:

```
assign w1 = B > A;
assign w2 = C > B;
wire: picoinput
[6:7]

assign picoinput = ({w2, w1});
```



برنامه picoBlaze :

Start :

Input S_0 , 00 ; از آدرس ۰۰ بخوان و توی S_0 بریز

Compare S_0 , 03 ; $\overset{1}{w_1}\overset{1}{w_2} = 3$ اگر هردو 1 بود .

JMP Z

Z output 255 , 00