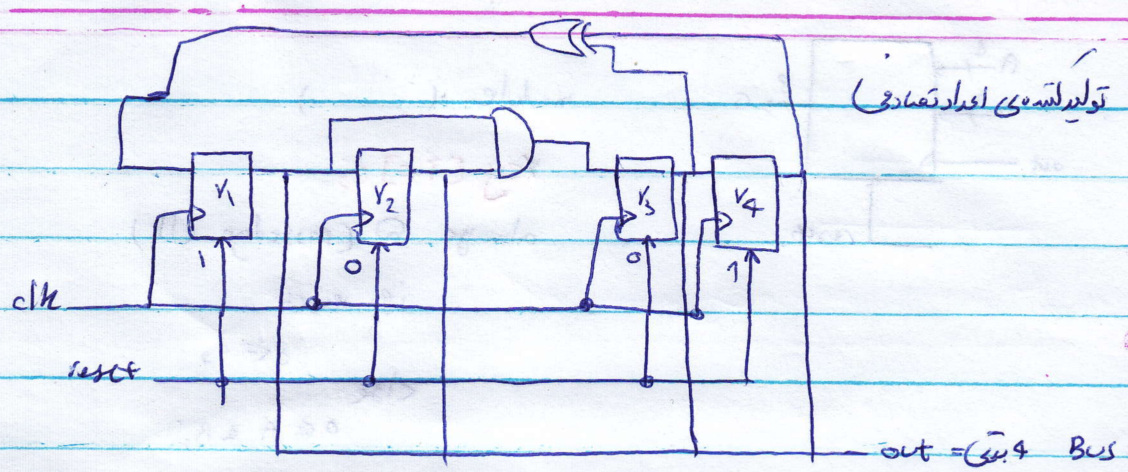
تا اینجا scan شد

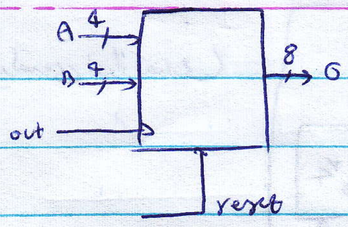
۸۸/۱۲/۴

[illegible]

تولیدکننده‌ی اعداد تصادفی




```
module random (clk, reset, out);
  input clk, rest;
  output [3:0] o;
  reg r1, r2, r3, r4;
  always @ (possedge clk)
     if (reset)
        begin     r1 <= 1;  r2 <= 0;  r3 <= 0;  r4 <= 1;  end
     else
        begin
           r1 <= r3 ^ r4;
           r2 <= r1;
           r3 <= r2 & R1;
           r4 <= r3;
        end
        assign  o = {r4, r3, r2, r1};
endmodule
```




```
module (          )
  reg [7:0] o;
  always @ (posedge clk)
      if reset
          o <= 0;
      else
          o <= A * B;
```

می‌خواهم که با هر لبه بالارونده 0،1،2،4،8 شود.

اگر [illegible]

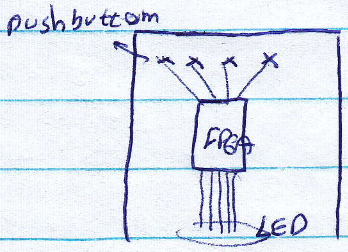
```
if (Reset)    o <= 0;
else
    begin
     if (o == 0)
         o <= 1;
     else
         o <= o*2;
end
```

```
o = { o[2:0], 0 }
```

هر lookuptable و هر DFF یک بلوک را می‌سازند. این بلوک‌ها با هم ۴ تا slice را درست می‌کنند.

sram که ۱۶ بیتی. یک logiccell را هم دوتا

که هر ۴ تا CLB را می‌سازند. که همه slice ها به یک سوییچ وصل می‌شود.



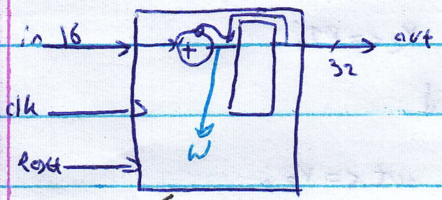
پس هر ورودی و خروجی باید مشخص باشد که کدام پایه FPGA است.

↳ user constraint

که با planAhead جای پایه‌ها را مشخص می‌کنیم.

کوئیز

- implematation = مرحله‌ای که روی برد می‌ریزد. الان مثلاً مدار رو روی fpga بچینیم placement
مثل مرحله PCB. بعد سیم‌کشی است داخل fpga که routing است. و این دو مرحله
باهم place and rout است.



- module Acc (out, In, clk, Reset)

```
input [15:0] In;
input clk, Reset;
output [31:0] out;
reg [31:0] out;
always @ (possedge clk)
    if (!Reset)
        out <= 0;
    else
        out <= out + In;
endmodule
```

← هر گاه با لبه ساعت می‌خواهی در متغیر تغییر ایجاد کنی باید reg تعریف کنی.

← چون مدار ترتیبی داریم.

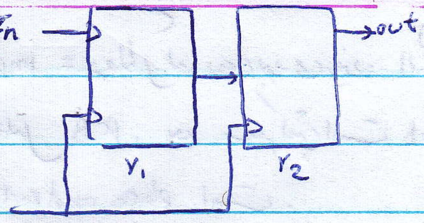
می‌توانی با استفاده از wire مدار را بنویسی ⇐

```
wire [31:0] w;
assign w = out + In;
always @ (possedge clk)
    out <= w;
```

! بلوک begin ، assign معنایی ندارد!!
end ~~assign~~

می‌توان برنامه بالا out را فقط output تعریف کنی و فقط یک reg[31:0] r تعریف کنی که
در بلوک always ، r <= r + In باشد و خارج از بلوک always (چه بالا چه پایین)
assign out=r; .

out هم بنویسی درست است!!




```
reg r1,r2;
always @ (posedge clk)
    begin
        r1 <= In;
        r2 <= r1;
    end
assign out <= r2;
```

<< این دستورها پشت سرهم انجام می‌شود که همزمان رخ می‌دهد.

NUMBERS

تا الان همواره اعداد را دسی‌مال نوشته‌ایم که دلاً صحیح است: assign a=a+123 .

اما گاهی می‌خواهیم مشخص کنیم که تعداد بیت‌ها را مشخص کنیم. مثلاً در نوشته زیر

w = out + In
? = 32 16

ابزار سنتز باهوش است و w را 32bit می‌کند.

علامت کتیش

تعداد بیت عدد ' مبنای عدد خود عدد → 8'd 124 (دسی‌مال)

8'b 01110111 → باینری

8'h 7C → هگزادسیمال

8'b 11

8'b 00000011 ← خودش صفر اضافه می‌کند.

```
wire [7:0] w; wire[15:0] a; output[15:0] Q;
assign Q = a+{8'd0, w}
```

یک صفر هم می‌تواند باشد.

8'b 1111 0000 ≡ 8'hf0 ≡ { 4'b1111 , 4'b0000 } = { 4{1'b1} , 4'b0 }

هم‌ارز است.

* یک سیم 4 حالت برای drive دارد:

- vcc ← 1
- gnd ← 0
- هیچ‌کس drive نکند ← Z
- x ← دو عامل آن را درایو می‌کند یا مثلاً اگر flip flop ای باشد که reset نشود یعنی مقدار اولیه نگرفته باشد در حالت x است.

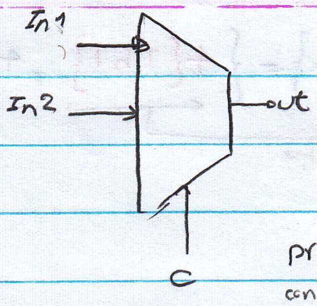
```
module Conter(out,clk);
  output [7:0] out;
  input clk;
  reg [7:0] out;
  always @(posedge clk)
    out <= out + 1;
    out <= ~out
endmodule
```

out <= out + 1; → این برنامه غلط است! زیرا مقدار x با 1 جمع کنی باز هم x است.

out <= ~out ← هم غلط است؟!

← پس حتماً باید در بلوک always ها، Reset داشته باشی!

این مدار یک مدار ترکیبی است :



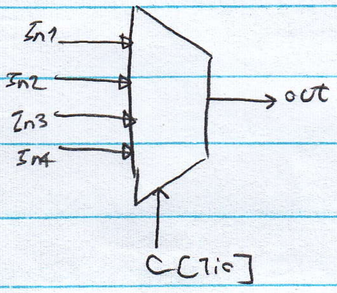

```
module mux2(In1, In2, c, out);
    input In1, In2, c;
    output out;
```

procedural continous assignment → assign out = (c==1)? In1 : In2;

اگر c برقرار باشد out=In1 شود وگرنه out=In2 است.

```
endmodule
```

assign خروجی = (شرط) ? (اگر) گزینه۱ : (وگرنه) گزینه۲ ;




```
assign out = (c==0)? In1 :
             (c==1)? In2 :
             (c==2)? In3 :
             (c==3)? In4;
```

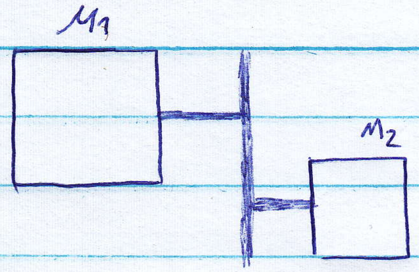
~~(c==3)?~~ ← این را نمی‌خواد. حتماً باید مقداری که هیچ کدام از شرط‌ها برقرار نیست را بنویسیم.

مالتی پلکسر 2 به 1 بالا را می‌توان به نحو پایین هم نوشت :

```
module mux2(out, In1, In2, c);
    output out;
    input In1, In2, c;
    reg out;
    always @(In1 or In2 or c)
        if (c)   out <= In1;
        else     out <= In2;
endmodule
```

یعنی هر تغییری که در سیگنال‌ها رخ داد باید if را انجام دهد!

توجه کن که این مدار هنوز هم ترکیبی است!



وقتی یک Bus مشترک داریم:

در هر لحظه فقط یکی از M_1 یا M_2 ، Bus را

drive می‌کند که احتیاج به بافر سه حالته داریم.

```
assign w=(C)? 1 :
        (B)? 0 : 1'bz;
```

→ یعنی گاهی اوقات که هیچ کدام

از شرط‌ها برقرار نشد مدار به سمت high impedance می‌رود!