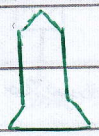
در این موشک سه مرحله ۱- سامانه تعادل ۲- navigation : جهت میدان مغناطیسی، شتاب لحظه‌ای (AMU) ۳- پردازش تصویر برای تشخیص هدف

در هنگام حرکت موشک سه مرحله داریم:



۱- هنگام برخاستن از زمین: سامانه تعادل

۲- هنگام حرکت در هوا: سامانه navigation

۳- هنگام برخورد به هدف: پردازش تصویر برای تشخیص هدف.

حال اگر یک chip داشته باشیم که ساختارش عوض شود (کنترل)... پردازشی را که نیاز به هزینه زیاد دارد خواهیم داشت.

IMU:

Intertial Measurement Unit

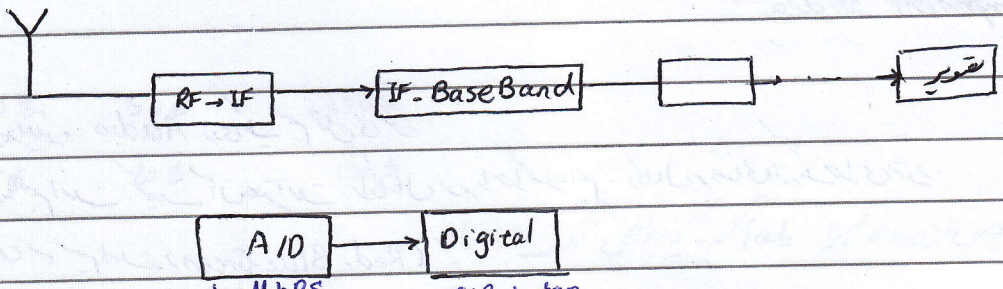
جهت و گرانش زمین و شتاب لحظه‌ای را در هر لحظه به ما نشان می‌دهد.

در انیمیشن سازی به کار می‌رود. یک تیرک لباس حاوی سنسورهای IMU می‌پوشد و هر قدر حرکت می‌کند در کامپیوتر می‌توان ثبت کرد.

* کمپانی تولید magnetometer و Honey wall می‌باشد.

مثال: گیرنده تلویزیون.

در گیرنده تلویزیون تعداد زیادی سوئیچ آنالوگ داریم. حال اگر یک A/D سریع داشته باشیم می‌توان سراسر سیگنال را دیجیتال کرد و پردازش‌ها را به صورت دیجیتال انجام داد. و سوئیچ زیاد گیرنده تبدیل به یک چیپ می‌شود.



باید با نرخ بالایی نمونه برداری کنیم تا اطلاعات را از دست ندهیم.

۴۰۰MHZ - ۴۰۰MHZ

DVBS 9-11 GHZ . 1.0 - 2.0 GHZ بعد از دمدولاتور

در عمل به کار برده شده برای هر نمونه باید ۱۰ ضرب و ۱۰ جمع انجام شود.

ضرب و جمع در یک ثانیه = ۱۰۰۰M تا سه ثانیه 10x100M

Pentium 4 ⟶ max f: 1.5 GHZ

اگر پنتیوم ۴ ، استفاده کنیم بخش زیادی از توان پردازشی آن صرف محاسبات FIR می شود.

باید یک چیپ سازم که توی آن پر ضرب شده باشد نه اینکه از یک چیپ آماده استفاده کنیم.

حتی ضرب هم نشد اگر ما می توانستیم یک چیپ قابل برنامه ریزی داشته باشیم که بتوانیم FIR را سخت افزاری پیاده کنیم.

مثال. قرارداد سیستمی داشته باشیم که در آن یک سیگنال ویدئویی بیاید و آنرا میزبستیم. وقتی می خواهیم در طول یک کانال مخابراتی، ویدئو را کسی نبیند و آنرا رمز کنیم)

انواع Vrideo آنالوگ:

Video - Composit

Super - Video : S-video

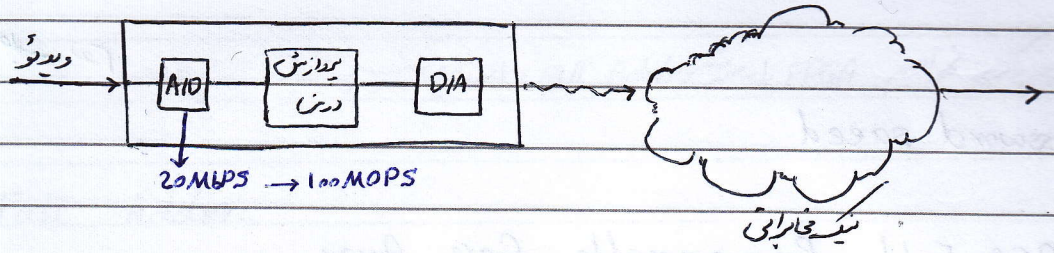
Component Video

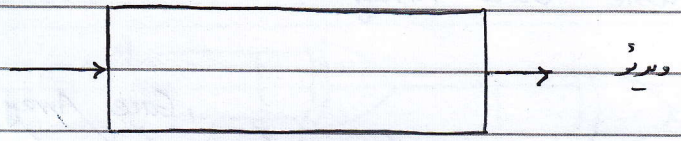
کل ویدئو و Audio روی یک سیم حمل می شود

دوتا سیم است. کیفیت آن بهتر است زیرا کانال نور را جدا کرده ایم. و کانال روشنایی از رنگ جدا می شود.

روی سه سیم است (Red - Blue - Green)

وزارت بازرگانی شرکت بازرگانی دولتی ایران





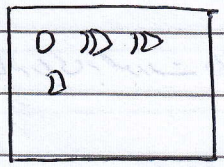
یک چیپ خام با ویژگی های زیری خواهیم :

1- قدرت پردازش خوب

2- از تمام چیزهایی که درون آن اتفاق می افتد خبر داشته باشیم .

3- قیمت معقول

4- بتوانیم برد مربوطه را خودمان بزنیم .



FPGA یک دریا یا یک عالمه گیت خام می باشد که این گیتها را می توانیم به هر صورت دلخواه درونی آن قرار دهیم . مثلاً می توانیم گیتها را طوری تنظیم بدهیم که تبدیل به MP ، پردازش کننده تصویر و ... تصویر شود . و حتی می توانیم برنامه آنرا عوض کنیم .

اگر بخواهیم یک چیپ بسازیم دیگر به fab دسترسی نداریم از FPGA استفاده می کنیم .

سنگاپور ، چین ، آمریکا ، کانادا تکنولوژی ساخت سیلیکون را دارند . (کره جنوبی ، سامسونگ)

قطب ساخت سیلیکون : تایوان

30 $	50 $	150 $
4 $	20 $	150 $

بنابراین از دو راه آنرا fab کنیم دگرنه در واقعی تکنیک می خوریم .

وزارت بازرگانی شرکت بازرگانی دولتی ایران

وزارت بازرگانی شرکت بازرگانی دولتی ایران

جلسه دوم:

Password: saeed

FPGA: Field Programmable Gate Array

Gate Array:
یک مدار که کلیت به صورت یک آرایه قرار گرفته اند.

PGA:
قابل برنامه ریزی به صورت دلخواه می باشد ی توان connection ها را به صورت دلخواه برنامه ریزی کرد.

Field:
محلی که مدار الت دستگاه در آن محل کار کند را فیلدی گویند.

در خود محل قابل برنامه ریزی الت پس به هیچ تکنولوژی خاصی برای پیاده سازی آن نیاز نداریم و نیاز به تجهیزات خاصی نداریم.

Lab On a Chip:
از سیستمی وارد بررسی شده الت. وسایل آزمایشگاهی با مقیاس بزرگ در فضای محدود کوچک شده Integrate شده اند.

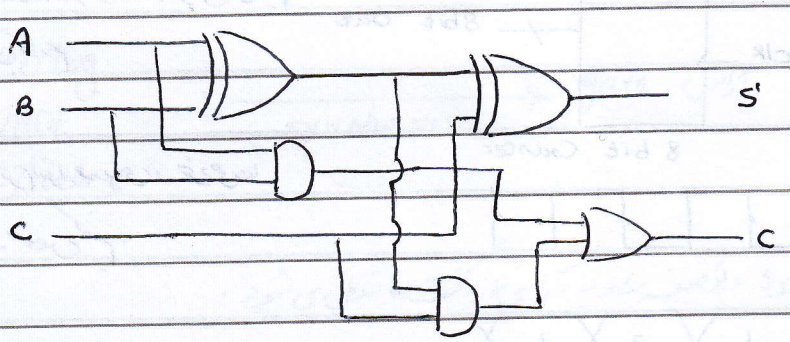
FPLOAC:
همیک دستگاهی انجام دانش های شیمیایی را داشته باشد در محل قابل برنامه ریزی باشد.

System On a Chip:

وزارت بازرگانی شرکت بازرگانی دولتی ایران

فرض کنید نخواهیم FPGA را به صورت Full Adder برنامه ریزی کنیم.

Full Adder:



یک روش طراحی این است که تمام یک مدار را بکشیم و آنرا به نرم افزار موجود بدهیم و آن نرم افزار برای ما کد FPGA مناظر را تولید می کند. اشکال این روش سخت و وقت گیر است و رفع اشکال از آن زمان می برد.

حال اگر یک نرم افزار داشته باشیم که با هوش باشد و وقتی ما می گوییم Full Adder می خواهیم بسازد و آن کار این گفتن را انجام دهد

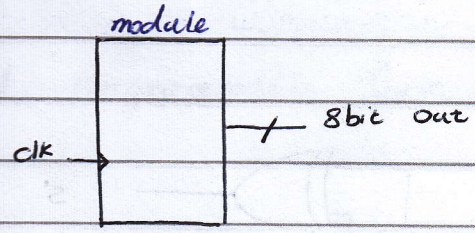
- نرم افزاری که
باهوش باشد
جمله ای که به آن می گوییم را بسازد
یا آرایش گفتی مدارها آشنا باشد.

اگر این نرم افزار را داشته باشیم، کافی است کار مدار را توصیف کنیم و نرم افزار خودش مدار را ایجاد می کند در این حالت دیگر لازم نیست تمام یک مدارها را خودمان بکشیم.

وزارت بازرگانی شرکت بازرگانی دولتی ایران
وزارت بازرگانی شرکت بازرگانی دولتی ایران

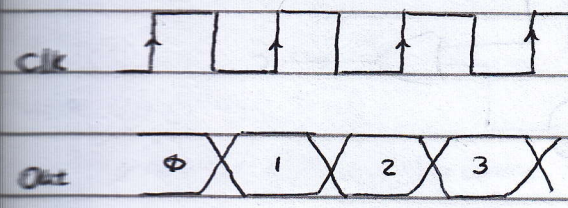
فرض کنید بخواهیم FPGA را به یک شمارنده ۸ بیتی تبدیل کنیم.

باید ببینیم یک شمارنده ۸ بیتی چه کاری انجام می دهد و چه عملکردی دارد؟



اسم هر module و بلوک که طراحی می کنیم ماژول می گذاریم.

باید برای نرم افزار هدردی و خروجی ها ماژول را مشخص کنیم.



مداری می خواهیم که: خروجی اش در هر لبه بالارونده یکی زیاد شود.

نرم افزاری که داریم فقط Syntax خاصی را می فهمد.

مداری می خواهیم که همیشه در هر لبه بالارونده خروجی یکی زیاد شود.

```
module  8bit Counter (input  clk, output [7:0] out);

always  @(Posedge  clk)
        out <= out + 1 ;

endmodule
```

(لبه بالارونده)

هر گزینه که مدار را توصیف کنیم ok است. یعنی برای توصیف یک مدار، syntax های متفاوتی وجود دارد.

زبان فوق، توصیف به زبان Verilog می باشد. البته زبانهای دیگری نیز برای توصیف سخت افزار وجود دارد: مانند VHDL ،C ،System c و برای همه این زبانها نرم افزاری وجود دارد که محیط این

زبانهای توصیفات سخت‌افزاری هستند.

اسم این نرم افزارها هوشمند که توصیفات را به آرایش گیتی تبدیل می کند Synthesizer می باشد.

توصیف مدار: Verilog, VHDL, System C

synthesis → آرایش gate

در مرحله synthesis، توصیف مدار به آرایشی از گیت‌ها تبدیل می شود.

می توان سازه‌های عملکرد مدار را توصیف کرد و نرم افزاری هوشمند وجود دارد که آن را به صورت بهینه به آرایشی از گیت‌ها تبدیل می کند. که حتما این آرایش از حالتی که ما خودمان آنرا طراحی می کنیم (به صورت تکنیک) بهینه تر است.

توصیف شمارنده ۴ بیتی با زبان C:

این یک کلاس است

```
SC_MODULE(counter) {
SC_in_clk     clock;
SC_out < SC_uint<4>   out;          uint: unsigned int
SC_uint<4>    count;
void  inc_counter ()  {count= count+1;}
SC_CTOR (counter) {                 class Constructor
  SC_METHOD(inc_counter);
  Sensitive << clock.pos( );
  }
```

وزارت بازرگانی شرکت بازرگانی دولتی ایران

clk

out

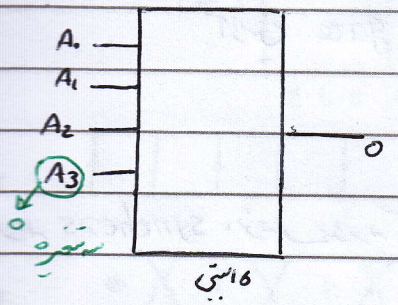
module 8bit

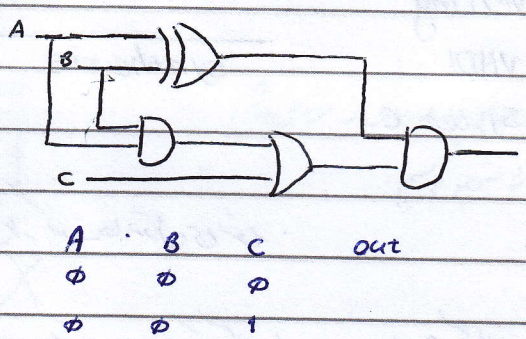
always

endmodule

ساختار اصلی داخلی FPGA :

اگر در مدار منطقی یک حافظه 16 بیتی با 4 ورودی آدرس و یک خروجی داشته باشیم می توانیم یک تابع منطقی 4 متغیره را می توان با آن پیاده کرد ؟



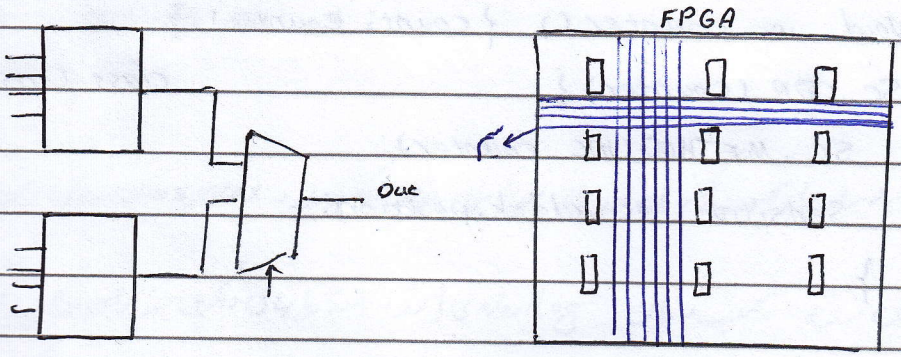


$F(A_0, A_1, A_2, A_3)$

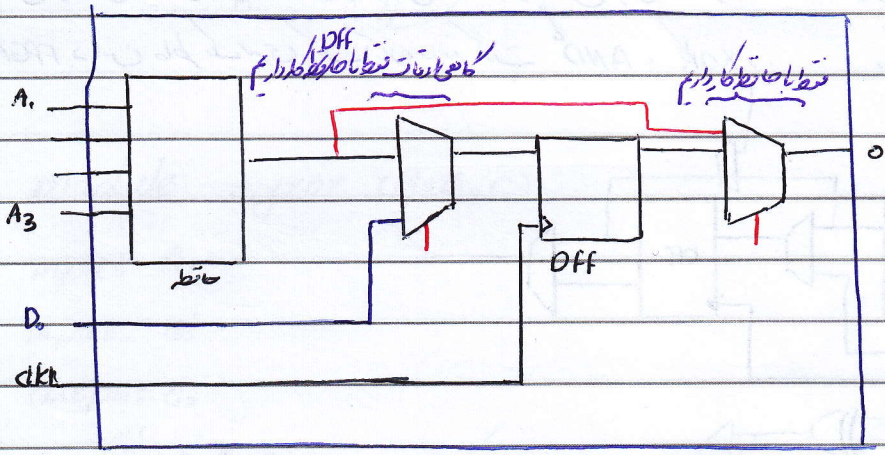
چه A و B و C را به مدار فوق یا مدار حافظه ابدهیم هیچ خروجی نمی گیرند . خروجی در هر دو حالت یکی است .

در صورتی که متغیرهای زیادی داشته باشیم می توانیم از حافظه بزرگتر استفاده کنیم .

در واقع در FPGA هیچ گیتی وجود ندارد تا آنکه وجود دارد بلوکهای 4 بیتی حافظه ای باشد و یک سیم بندی خاصی بین آنها وجود دارد به طوری که هر دو یا چند حافظه دلخواه را بتوان به هم مرتبط کرد .



با کمبینه‌ی ، می‌توان هر مدار ترکیبی (بدون حافظه) را پیاده‌سازی کرد. برای این که بتوان قابلیت پیاده‌سازی یک مدار ترتیبی را داشته باشیم به آن یک DFF اضافه می‌کنیم.

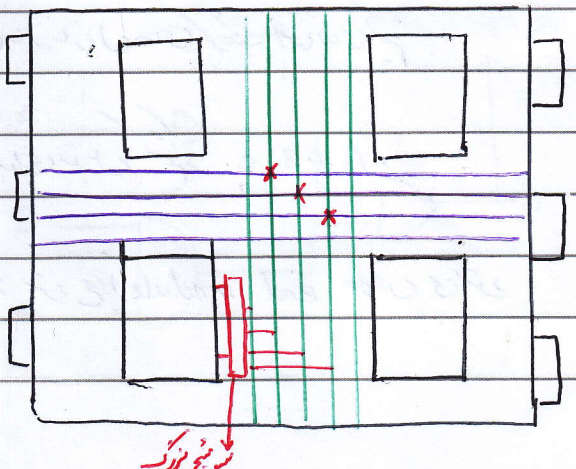


1- باید حافظه را پر کنیم
2- سیگنال کنترلی مالتی پلکسرها را تعیین کنیم
3- پین بندی داخل FPGA

اطلاعاتی که باید به FPGA بدهیم تا مدار دلخواه خود را داشته باشیم می‌باشد.

حافظه یک SRAM می‌باشد و با قطع برق، اطلاعات آن از بین می‌رود.

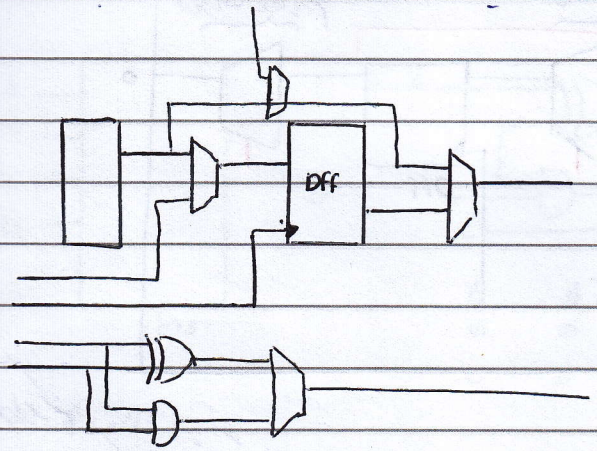
داخل FPGA را از مدار فوق می‌کنیم.



X: سوئیچ ساده

باید سوییچ ها را نیز برنامه ریزی کرد.

بعضی از اعمال هستند که زیاد به کار می روند، مثل ضرب کردن، جمع کردن، look up (جدول sin). طراحان FPGA به این خاطر یک سری گیت آماده مانند گیت AND، XOR و ... را به آن اضافه می کنند.



می توان اصلاً بلوک ضرب کننده و چند بلوک دیگر را به صورت آماده در FPGA گذاشت که قابلیت ها آن بالا رود. در این حالت سرعت نیز بالا می رود.

جلسه سوم

آنچه برای ما اهمیت دارد، رابط سریال است.

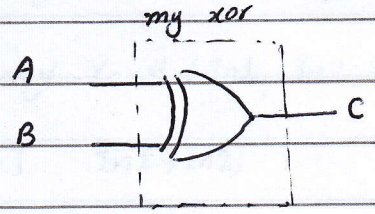
در FPGA عمای طراحی های ساز ساژول های کوچک استفاده می کنیم.

فرض کنید بخواهیم FPGA را به برای تبدیل کنیم که A⊕B=C.

کد همواره با module شروع با end module خلاص می شود.

وزارت نیرو، عالی شرکت برق عالی دولتی ایران

آخر هر دستور یک ; داریم غیر از end module .



پورتها

```
module myxor (A,B,C);
input A;
input B;
output C;
xor X(A,B,C)
endmodule
```

ایم

اعلان از نوع دروازه

حال باید XOR را تعریف کنیم.

باید به هر اعلان یک اسم بدهیم برای اینکه بتوانیم براحتی به آنها اشاره کنیم.

A و B را وصل می کند به پورتهای ورودی xor و C را وصل می کند به پورتهای خروجی xor .

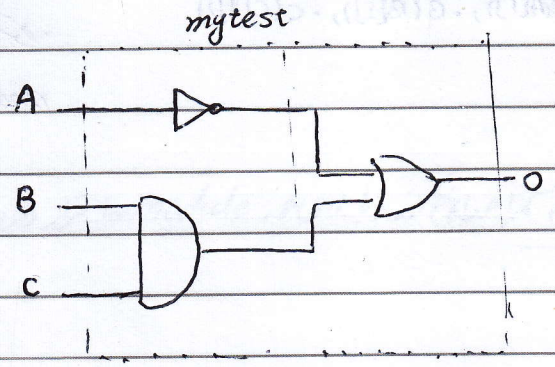
البته می توان از عبارتهای توصیفی نیز استفاده کرد و به جای عبارت xor X(A,B,C) عبارت توصیفی زیر را نوشت.

```
assign C = A ^ B;
```

اعلان

هر وقت خواستیم یک عبارت ترکیبی را بیان کنیم از کلمه "assign" استفاده می کنیم.

Example:

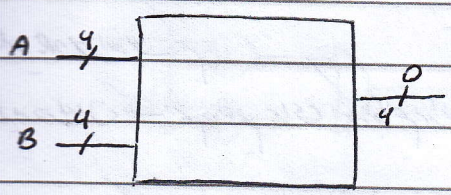


وزارت بازرگانی شرکت بازرگانی دولتی ایران

```
module mytest (O, A, B, C);
output O;
input A, B, C;
assign O = (B&C)|(~A);
end module
```

ماژول هایی توانند همدیگر را صدا بزنند یعنی می توان چند ماژول کوچک طراحی کرد و بعد آنها را ترکیب کرد و یک ماژول بزرگتر ساخت.

Example:



اول یک XOR دوبیتی و سپس یک XOR چهاربیتی می سازیم.

```
module myXor2 (A, B, C);
output [1:0] C;
input [1:0] A, B;
myXor Ins1 (.A(A[0]), .B(B[0]), .C(C[0]));
myXor Ins2 (.A(A[1]), .B(B[1]), .C(C[1]));
end module
```

Don't forget

* اول عدد بزرگتر
* وقتی بخواهیم یک ماژول را در یک ماژول دیگر صدا بزنیم به آن یک اسم می دهیم.

اسم ماژول قبلی — اسم ماژول جدید

باید مشخص کنیم که سیگنال های کدام (کدجدید) طرف کدام سیگنال کدقبلی متصل می شود.

```
module myXor (A, B, C)
.A ↵ .B .C
```

Example:

```
module  my Xor4 (In1 , In2 ,Out);
input [3:0]   In1 , In2;
output [3:0]  out;

my Xor 2    Ins1 ( .A(In1[1:0]),
                   .B(In2[1:0]),
                   .C(Out[1:0]));
myXor2      Ins2 ( .A(In1[3:2]) , .B(In2[3:2]),
                   .C(out[3:2]));
end module
```

در این مثال ما از ۴ بیت برای ورودی و خروجی استفاده کرده‌ایم که خواستیم ساده‌تر توضیح دهیم.

Example:

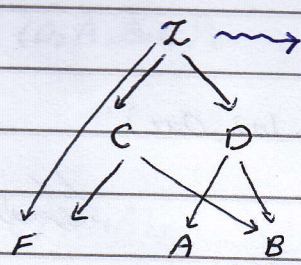
در myXOR2 می‌توان به شکل زیر عمل کرد:

```
module   myXor2(A,B,C);
 output    [1:0]  C;
 input     [1:0]  A,B;
 assign    C = A^B;
 endmodule
```

Top module:

module‌ی که هیچ کس را صدا نمی‌کند و با آن همه module‌ها تراز دارد.

وزارت بازرگانی شرکت بازرگانی دولتی ایران



assign :

ابتدایی که با آن می‌توان خیلی راحت مدارهای ترکیبی را توصیف کرد.

Example:

ضرب کننده 4 بیتی :

```
module  Mult4 ( In1, In2,out);
input  [3:0]  In1, In2;
output [7:0]  out;
assign  out = In1* In2;
end module
```

Example:

جمع کننده 4 بیتی :

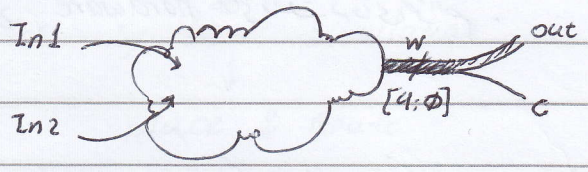
```
module  Adder4 (In1, In2, Out, C);
output  [3:0]  out;
output         C;
input   [3:0]  In1, In2;
Wire    [4:0]  w;
assign  w= In1+In2;
```

هر چیزی که درون خط باید 4 بیتی باشد.

یک بیس 5 بیتی در نظر گرفتیم.

حالا کافی است w را در C و out بریزیم.

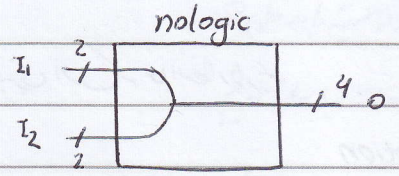
```
assign C = w[4];
assign out = w[3:0];
end module
```



فرض کنید در ماسیم داشته باشیم و بخواهیم آنها را با هم ترکیب کنیم و یک بیس با بعرض بیشتر بسازیم.

اگر ضد assign بیت هم داشته باشیم می‌توانیم یک assign بنویسیم و به جای ۱، ۲، ۳ و ۴ بگذاریم

Example:




```
module nologic (I1, I2, O)
input [1:0] I1, I2;
output [3:0] O;
assign O = {I2, I1};
end module
```

چسباندن دو سیم به هم

کوتاه شدن پرانتز

Example:

در همان جمع کننده ۴ بیتی می‌توان wire را نوشت و داریم:

```
assign {C, out} = In1 + In2;
```

Example:

```
module
input [3
output [
assign o
end mod
```

Example:

```
module
output
output
input
wire
assign
```

وزارت بازرگانی شرکت بازرگانی دولتی ایران

: Concatenation

چسباندن چند سیم کوچک را یا یک باس بزرگتر.

باید مطمئن شویم که عملکرد hardware هایی است که ما می خواهیم .

: Simulator

نرم افزارهایی که مدار را برای ما شبیه سازی می کنند و قبل از پیاده کردن روی Hardware ما را مطمئن می کنند که مدار همان کاری را می کند که ما می خواهیم .

نرم افزار زیلنکس برای ما سنتز می کند .

مراحل شبیه سازی :

۱. ما عملکرد این ماژول که آن چیزی است که ما انتظار داریم

عملکرد ماژول Functional Simulation

همه چیز ایده آل است ما هیچ hardware ای نداریم . تأخیر گیتها مهم است setup time hoff ندارند .

۲. Synthesis و تبدیل به گیت

۳. شبیه سازی گیتها

Post Synthesis Simulation:

۴. place & Route در FPGA

۵. Post Route Simulation

تأخیرها واقعی . ابزار مدار را عمل شان داده می شود .

وزارت بازرگانی شرکت بازرگانی دولتی ایران

طراحی

↓

Functional verification

↓

synthesis

↓

Post synthesis simulation

↓

Place & Route

↓

Post Route simulation

جلسه چهارم

assign c, A|B or

طراحی برای FPGA چند مرحله دارد که همیشه باید طی شود:

۱. این دفعه می خواهیم ورودی ها را از دکمه های نیاز داریم.

1. Design Entery

submodule های برای انجام کار خاص طراحی می شود. نوشتن کد verilog نزدیکی در این مرحله وجود دارد.

2. synthesis:

تبدیل به سخت افزار و تطابق با توصیات دیگت

3. Implementation:

مرحله پیاده سازی

قبل از رفتن از مرحله به مرحله ی بعد Design را تست کنیم. این کار با دو روش زیر انجام می شود.

Functional simulation

وزارت بازرگانی شرکت بازرگانی دولتی ایران

وزارت بازرگانی شرکت بازرگانی دولتی ایران

Functional

Post Synth

Post Route

وزارت بازرگانی شرکت بازرگانی دولتی ایران

در این مرحله هیچ تأخیری وجود ندارد. در این حالت توصیف مداری را شبیه سازی می کنیم. ff ها set up & hold time ندارند.

Post synthesis simulation ← در مرحله بعد خود آلایش گیتی را شبیه سازی می کنیم.

در این مرحله نیز تأخیر نداریم.

Post Route simulation در مرحله بعد خود Design پیاده سازی شده روی FPGA شبیه سازی می شود که تأخیرها همان تأخیر واقعی گیت ها هستند.

IC های، FPGA ساخته شده توسط شرکتهای مختلف تأخیرهای متفاوتی دارند.

نرم افزارهای شبیه ساز:

1-Active HDL ⟶ Aldec Company

Interface گرافیکی شکیلی دارد برای یادگیری و شروع خوب است. ساده است و به همین دلیل قدرت آن پایین است. برای کارهای ساده خوب است ولی در پروژه ها از آن استفاده نمی شود.

2. Model Sim ⟶ Mentor Company

Interface آن چندان قوی نیست ولی simulator قوی است. برای کارهای پروژه

3. IUS (Insacive Unified Simulator) ⟶ Cadance Company

منطبق Linux کار می کند. بسیار قدرتمند و گران است. برای پروژه های جدی خوب است.

برای هیچ پروژه جدی از windows استفاده نمی کنیم و از Linux استفاده می کنیم.

4. VCS ⟶ Synopsys

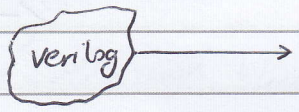
منطبق Linux کار می کند.

5_ System c

علت قدرتمند بودن 3 و 4:

کد در بلاک را تبدیل به opcode های cpu ای که داریم می کند به همین دلیل قوی است.

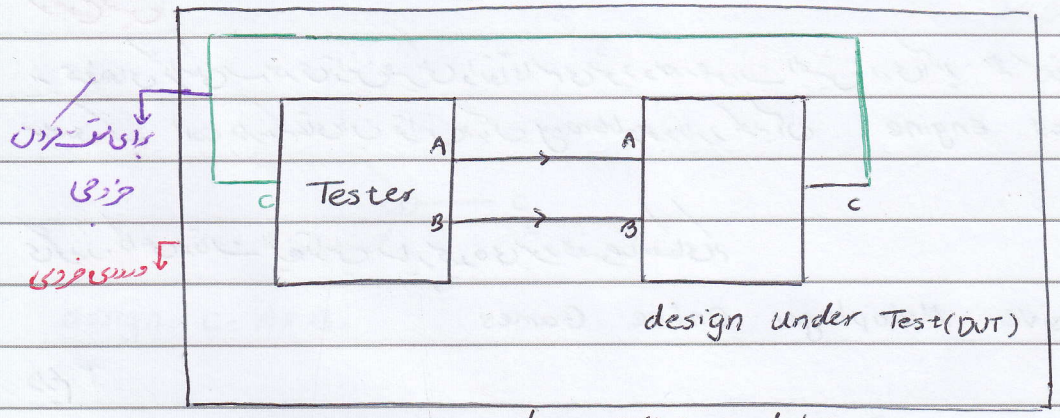
Native Code Simulator



متغیرهای که در اینجا تعریف کنند همان صورت که در FPGA تعریف کند وجود دارد.

وقتی با c برنامه می نویسیم مدار را توصیف می کنیم و وقتی با GCC آن را شبیه سازی می کنیم در واقع simulator است.

برای تست کردن ماژول یک ماژول تستری می نویسیم که خودش آن، در ورودی ماژول بعدی می دهد در ماژول * یک برنامه verilog ی می نویسیم تا شکل موجی که می خواهیم را تولید کند.



tester :

سیگنالی تولید می کند که می توانیم با آن عملکرد مدار را تست کنیم.

Example:

Reg [3:0] rA, rB;

initialize Begin

rA = 10;

rB = 20;

#300.

.

.

End

Assign A=rA, B=rB

با این کلک می‌توان مقادیر دوردی را تعریف کرد.

و مقادیر خروجی مناظر با دوردهای مختلف را ست کرد.

جلسه پنجم:

موضوع تحقیق:

محیط‌های گرافیکی سه بعدی توانایی فیزیکی واقع‌گرایانه‌ای می‌شود و بعد به صورت انجین در می‌آید. همه سیستم‌های سه بعدی یک لایه در بدنه‌های این قوانین به نام library وجود دارد که به آن physics Engine می‌گویند. گاهی اوقات این قوانین بسیار پیچیده می‌شود مخصوصاً هنگامی که

Massive Multiplayer Online Games

داریم.

با سنگین‌تر شدن محاسبات سرعت و توان CPU کم می‌شود به همین دلیل یک chip طراحی می‌کنیم که تنش physics Engine را دارد.

** چگونه می‌توان یک physics Engine را FPGA طراحی کرد؟

وزارت بازرگانی شرکت بازرگانی دولتی ایران

در قدم اول باید ببینیم در physics Engine چه توابعی وجود دارد؟

* چیپ physics engine و قابلیت هایی که در اختیار ما می گذارد.

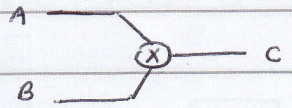
در verilog مدارهای ترکیبی را پیاده سازی کردیم. در واقع تابع خود را با assign تعریف کردیم.

assign out = ~In;

continuous Assignment

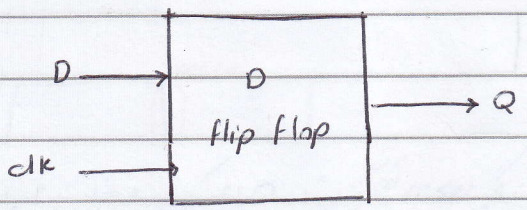
Continuous Assignment:

هر لحظه خروجی از روی ورودی محاسبه می شود اگر در لحظه ای از زمان یکی از ورودی ها را تغییر دادیم در لحظه خروجی نیز تغییری کند.

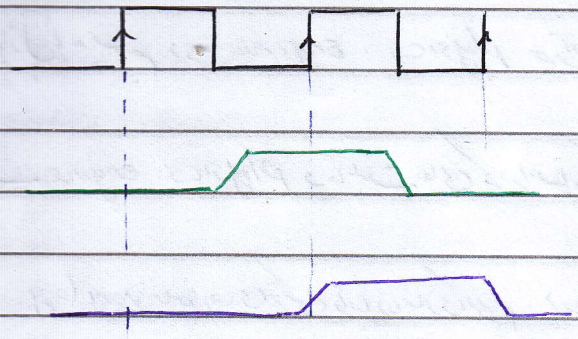


assign C = A * B;

پیاده سازی مدارهای ترکیبی:



وزارت بازرگانی شرکت بازرگانی دولتی ایران
وزارت بازرگانی شرکت بازرگانی دولتی ایران



فرض ff در لبه بالا رونده فعال می‌شود.

همیشه در هر لبه بالا رونده خروجی را برابر با ورودی قرار بده.

اگر سایز سگنال را ننویسیم به طور default آن را سیم در نظر می‌گیرد.

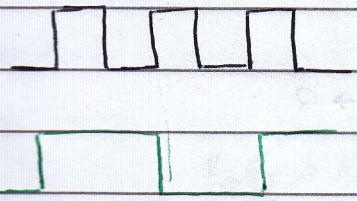
```
module dFF (D, clk, Q);
input D, clk;
output Q;
reg Q;
always @ (posedge clk)
Q <= D;
endmodule;
```

هنگامی که در Always در آن مقداری می‌ریزیم باید reg تعریف شود.

Example:

TFF:

در هر لبه بالا رونده Not می‌شود.



```
module tff (clk,Q);
input clk;
output Q;
reg Q; or negedge
always @ (posedge clk)
Q <= ~Q;
end module
```

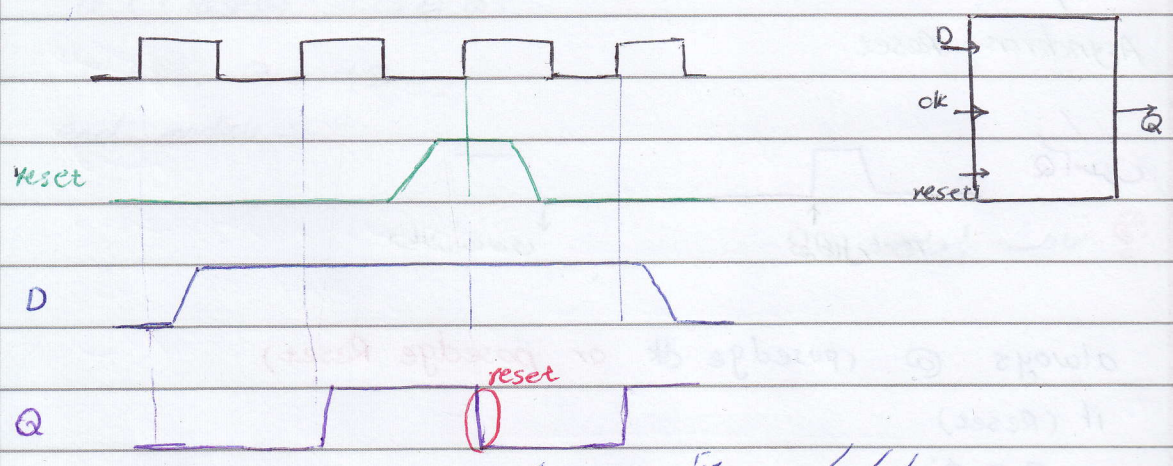
مدارهای فوق ناقص است زیرا در ترتیبی حتماً reset می خواهد و با آن تخصیص می کنیم، مدار اولیه خروجی چه قدر است.

Example:

Dff with Reset:

سنکرون Reset:

در reset سنکرون، reset در لبه های پالس اعمال می شود. بنابراین اعمال نمی شود.



در هر لبه بالارونده ساعت نگاه کن اگر reset همزمان Q را صفر کن (بنابراین صورت D را در Q بریز).

وزارت بازرگانی شرکت بازرگانی دولتی ایران

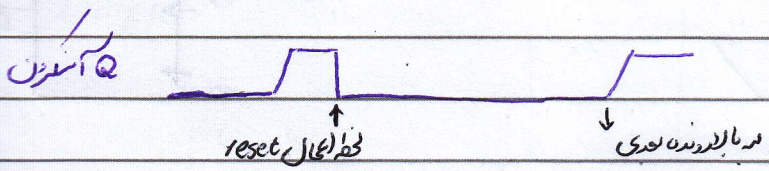
وزارت بازرگانی شرکت بازرگانی دولتی ایران

```
module DffR (D, clk, Reset, Q);
input D, clk, Reset;
output Q;
reg Q;
always @ (Posedge clk)
if (Reset)
    Q <= 0;
else
    Q <= D;
endmodule
```

می توانستیم از reset خود ff استفاده نکنیم و دستور منطقی بنویسیم ولی در این حالت سرعت حتی افت پیدا می کند زیرا قبل از ff یک لایه logic اضافه کرده ایم. حتی اگر ff ما پایه reset نداشته باشد synthesizer آنرا تولید می کند.

Example:

Asynchron Reset:




```
always @ (posedge clk or posedge Reset)
if (Reset)
    Q <= 0;
else
    Q <= D;
```

وزارت بازرگانی شرکت بازرگانی دولتی ایران

در این حالت مدار نسبت به لبه event حساس است.

هر متغیری را قطعاً نمی‌توان در یک Always مقداردهی کرد.

در زبان verilog همه کدها به صورت همزمان اجرا می‌شود زیرا همه این‌ها یک سخت‌افزار است.

در زبان C دستورات خط به خط اجرا می‌شود.

Example:

TFF with Reset:

```
module tffR (ResetL, clk, Q);
input  ResetL, clk;
output Q;
reg Q;
always @ (posedge clk or negedge ResetL)
if (!ResetL)    Q <= 0;
else            Q <= ~Q
end module.
```

تمارت ! ولد :

وزارت بازرگانی شرکت بازرگانی دولتی ایران
وزارت بازرگانی شرکت بازرگانی دولتی ایران

Example:

Counter(8 bit Counter):

```
module Counter (out, clk, Reset, value, updown);
output [7:0] out;
input clk, Reset, updown;
input [3:0] value;
reg [7:0] out;
always @ (posedge clk)
if (Reset)  out <= 0;
else        out <= out+1;
                    value
* else if (updown)  out <= out + value;
  else out <= out - value;
endmodule
```

fpga.googolia.com

حلسه ششم:

کوئیز 1:

1. تعریف واژه‌های زیر به‌صورت بسیار مختصر:

الف) Synthesis

ب) Functional Simulation

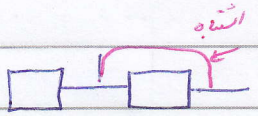
2. طراحی Inc1.

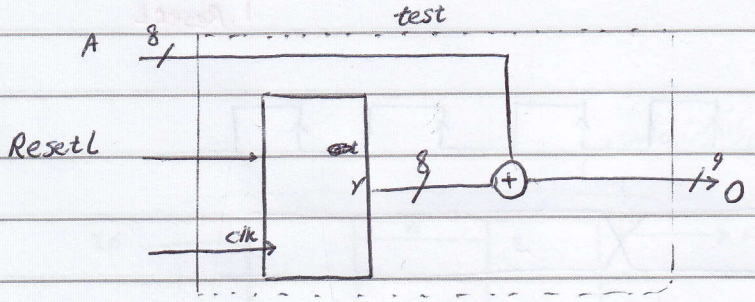
طراحی Inc2 با دیاگرام طراحی Inc1.

وزارت بازرگانی شرکت بازرگانی دولتی ایران

```
module Inc1 (In, out);
input [7:0] In;
output [7:0] out;
assign out= In+1;
endmodule;

module Inc2 (In, out)
input [7:0] In;
output [7:0] out;
* wire [7:0] w;
Inc1 X1( .In(In), .out(w));
Inc2 X2( .In(w), .out(out));
endmodule
```






```
module test (O, A, clk, Reset L);
output [8:0] O;
input [7:0] A;
input clk, Reset L;
* reg [7:0] r;
```

Example:

Counter18

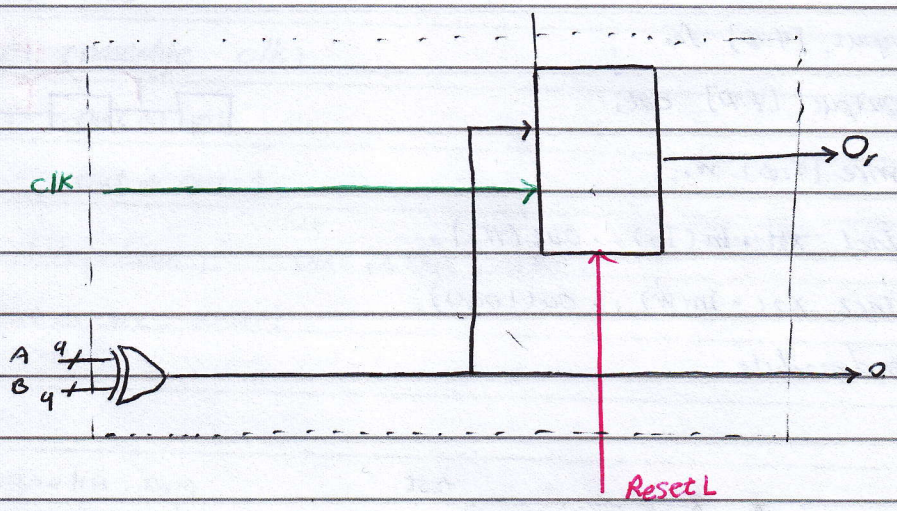
```
module Co
output [7
input clk,
reg [7:0]
always @
if (Reset)
else
  else if
    else out
endmodule
```

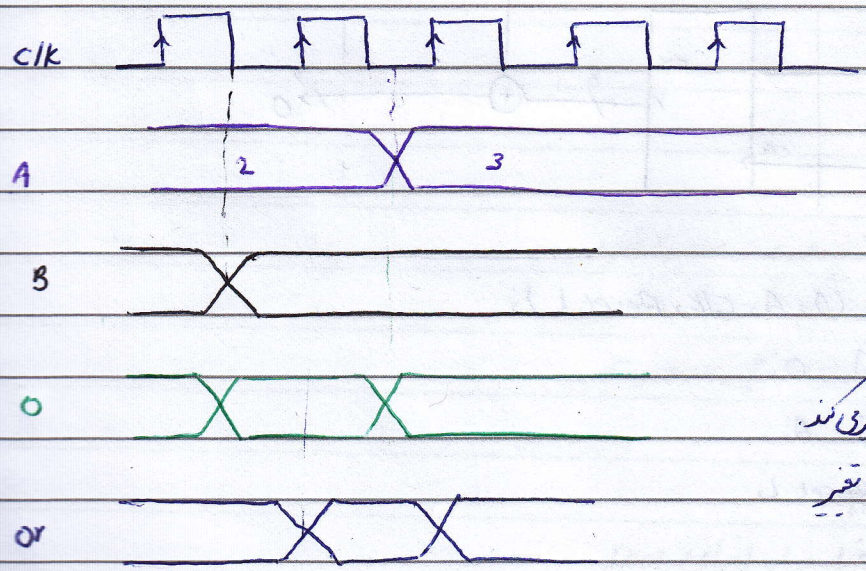
fpga-gaog

وزارت بازرگانی شرکت بازرگانی دولتی ایران
وزارت بازرگانی شرکت بازرگانی دولتی ایران

```
always @ (posedge clk)
  if (!ResetL)   r <= 0;
  else           r <= r+1;
assign O= A+r;
endmodule
```

* چون این قسمت ترکیبی است.





* محض تغییر در ورودی O تغییری کند.
ولی or در هر لبه بالارونده تغییر
می کند.

وزارت بازرگانی شرکت بازرگانی دولتی ایران

reg کردن باعث کاهش آفتری شود.

```
module test2(Or, O, clk, ResetL, A,B);
output [3:0] O,Or;
input  [3:0] A,B;
input  clk  ResetL;
*reg  [3:0] Or;
assign O= A^B;
always @(posedge clk)
if (!ResetL) Or<=0;
else Or<=O;
endmodule
```

reg ← ff خروجی or

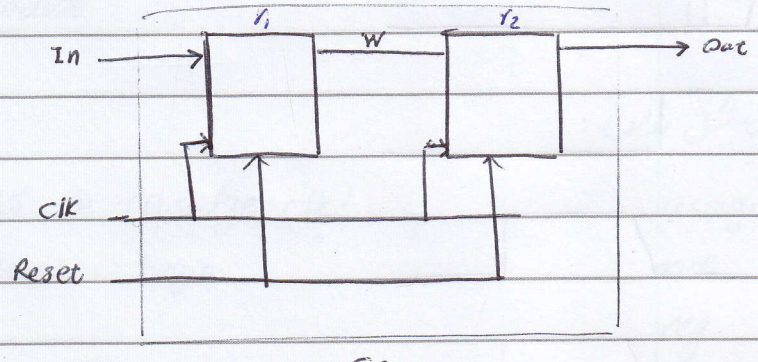
reg کردن یعنی دخیل کردن خروجی ff.

if (A) اگر هرکدام از بیت‌های غیرصفر باشد شرط برقرار است.

می‌توان جای assign و always را عوض کرد زیرا ترتیب مهم نیست.

shift Register:

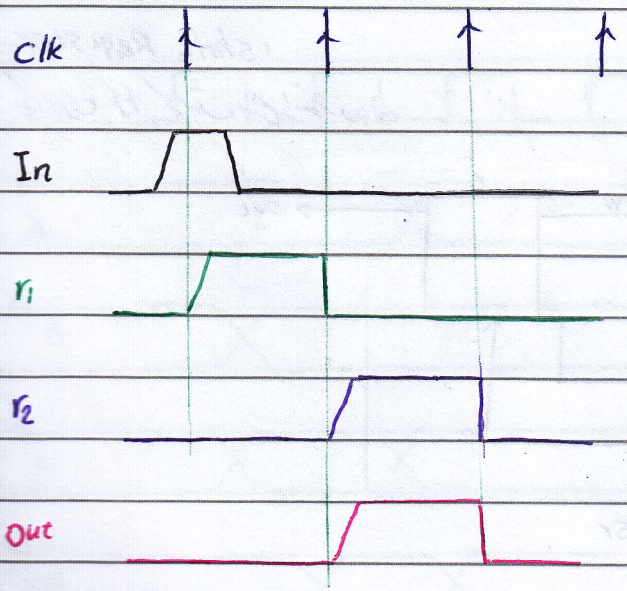
یک سری ff که ریست سنکرون دارند.



وزارت بازرگانی شرکت بازرگانی دولتی ایران
وزارت بازرگانی شرکت بازرگانی دولتی ایران

```
module Sr (Out, In, clk, Reset);
Output Out;
input In, ck, Reset;
wire w;
reg r1, r2;
always @(Posedge ck)
if (Reset)  r1 <= 0;
else  r1 <= In;
assign w = r1;
always @(Posedge clk)
if (Reset)  r2 <= 0;
else r2 <= w;
assign out = r2;
end module
```

در برنامه نویسی بهتر است سرتا input بنویسیم .



وزارت بازرگانی شرکت بازرگانی دولتی ایران

```
module Sr (Out, In, Clk, Reset);
```

=> : اگر آن در این لحظه از کدام عبارت برداری کن و یک ع بعد اضافه بده.

```
module Sr (out , In, clk , Reset);
output out;
input In, clk, Reset;
reg r;
reg out;
always @ (posedge clk)
if (Reset) begin
out <= 0;
r <= 0;
end
else begin
r <= In;
out <= r;
end
end module
```

for { ... } ← begin / end

وقتی از begin و end استفاده می‌کنیم خط به خط اجرای تو در تو رعایت اجرای هر خط تقریباً همزمان است.

شبیه ۲ فلیپ فلاپ سری

ع عبارت برداری

تمام بولیای always بشکل زیر است.

```
always @ (Posedge clk)
if (Reset) begin
...
end
else begin
...
end
```

assign

wire

reg

وزارت بازرگانی شرکت بازرگانی دولتی ایران

وزارت بازرگانی شرکت بازرگانی دولتی ایران

module

Output

input

wire

reg

always @

if (Reset)

else

assign

always @

if (Reset)

else

assign

end module

clk

In

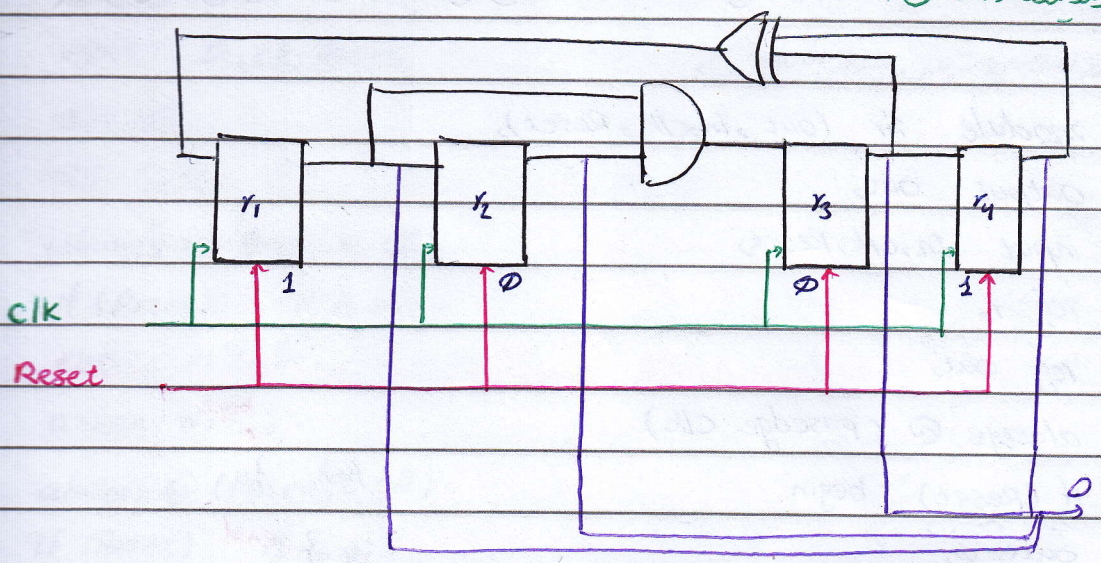
r1

r2

Out

Exp:

تولید کننده عدد تصادفی :




```
module random (clk, Reset, O);
input clk, Reset;
Output [3:0] out;
reg r1, r2, r3, r4;
always @ (posedge clk)
if (Reset) begin
 r1 <= 1;
 r2 <= 0;
 r3 <= 0;
 r4 <= 1;
end
else begin
r1 = r3 ^ r4;
```

وزارت بازرگانی شرکت بازرگانی دولتی ایران

$r_2 <= r_1;$

$r_3 <= r_2 \& r_1;$

$r_4 <= r_3;$

end

assign O = {r_4, r_3, r_2, r_1};

endmodule

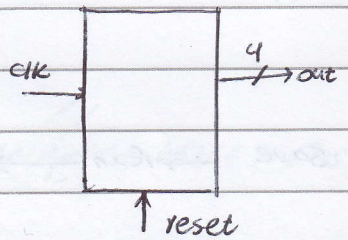
جلسه

مثال:

Counter

if reset O ⇐ Φ

else 0 → 1 → 2 → 4 → 8 → Φ




```
if (reset)
    O <= Φ;
else begin
if (io)
O <= 1
else
O <= O * 2;
end module
```

O <= {O[2:0], Φ}

یا O <= O << 1

روش O <= O * 2 می‌توان نوشت

وزارت بازرگانی شرکت بازرگانی دولتی ایران

وزارت بازرگانی شرکت بازرگانی دولتی ایران

Exp

clk

Reset

module nam

input clk,Re

output [3:0]

reg r_1, r_2, r_3,

always @ (p

if (Reset)

$r_1 <= 1;$

$r_2 <= 0;$

$r_3 <= 0;$

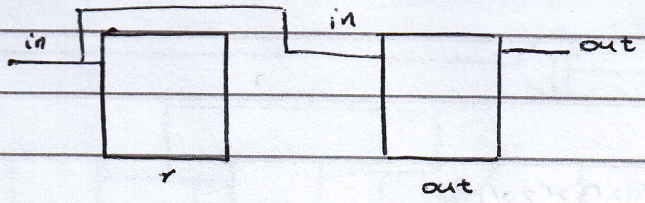
$r_4 <= 1;$

end

else begin

$r_1 = r_3 \wedge r_4;$

در shift reg اگر به جای =< از = استفاده شود، مدار به شکل زیر درمی آید:



نرم افزار:

با یک editor خاص، کدورپلاگ می نویسیم.

نرم افزار syn plify:

می خواهیم تعیین کنیم که FPGA برای چه نوع FPGA سنتز شود. با Add-file برنامه Verilog

را به پروژه اضافه می کنیم.

Save → Run → کل عمل سنتز انجام می شود

سنتز ۲ مرحله دارد:

۱- کدورپلاگ تبدیل به مجموعه ff و گیت می شود (مثلاً کد RTL)

در این مرحله ساختار FPGA مهم نیست.

برمدل می دهد

با کلیک روی ⊕ شکل مدار را به ما می دهد مداری که سنتز شده است.

در مرحله ۲ با توجه به ساختار داخلی FPGA، map کد را طبق optimizing هم انجام می شود

در واقع بر اساس المانهای از پیش FPGA، مدار را پیاده سازی می کند. الان اطلاعاتی در

وزارت بازرگانی شرکت بازرگانی دولتی ایران

خروجی placement اینها روی FPGA و routing بین آنها داریم.

اینجا یک فایل text با پسوند edf ساخته می‌شود که این در روی مرحله بعد کاربرد است.

می‌توانیم بلوک‌ها را FPGA تعیین کنیم و بگوییم که مدار قرار است در کدام قسمت بلوک قرار گیرد.

Xilinx ISE:

در این محیط‌ها توان کد نوشت، کد را سنتز کرد و کیف روی FPGA.

یک پروژه می‌سازیم و نوع FPGA ... را تعیین می‌کنیم.

Right click → Add source

می‌توان ISE را طوری تنظیم کرد که بعد سنتز شده synplify را صدا بزند.

کد HDL تبدیل شده Technology Schematic

implement Design $\xrightarrow{\text{right click}}$ run ⇒ Routing & Placement

اینها را روی برد می‌چیند و routing بین آنها را انجام می‌دهد ← File Generate programming

خانه‌ی با پسوند bit. تحویل می‌دهد که می‌توان آن را مستقیم روی FPGA ریخت

یا FPGA Editor توان داخل FPGA را دید.

* اگر بخواهیم مشخص کنیم که هر کدام از پورت‌ها روی کدام پایه FPGA قرار بگیرند چون شلاً روی برد هر پایه FPGA را به یک device خاص متصل کرده‌ایم ← در بخش User constraint

وزارت بازرگانی شرکت بازرگانی دولتی ایران

وزارت بازرگانی شرکت بازرگانی دولتی ایران

با نرم افزار plan-ahead محل پایه های FPGA را تعیین کنیم.

هر یک از پورت ها را بگشیم و سیم دهیم روی پایه های FPGA.

جلسه هشتم:

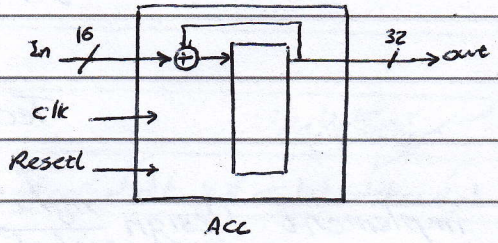
کوئیز ۳:

۱- در طراحی FPGA در مرحله Implementation چه کاری انجام می شود؟

مرحله ای که با آن design را روی FPGA می ریزیم که در این مرحله انجام می شود placement است و پس از آن مرحله بعد routing است که اتصالات داخلی FPGA را روی هم سیم کشی بین آنها انجام می شود. مرحله place & route

۲- یک واحد Accumulator طراحی کنید.

```
module Acc (Out, In, clk, ResetL);
output [31:0] Out;
input [15:0] In;
input ResetL, Clk;
reg [31:0] out;
always @ (posedge clk)
if (! ResetL)
    out <= 0;
else
   out <= out + In;
end module.
```



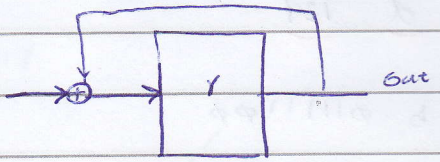
در بلوک begin-end هیچ چیز نمی توان assign کرد.

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

وزارت بازرگانی شرکت بازرگانی دولتی ایران

* از assign می‌توان در بلوک always استفاده کرد!

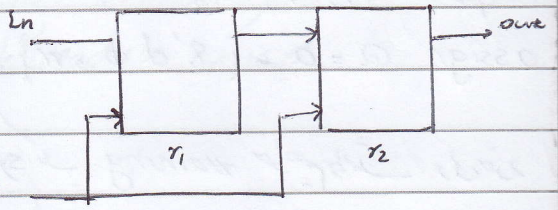
```
always ...
    r <= r + In;
```




```
assign out <= r;
```

example: shiftreg

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



Numbers in Verilog:

از مبنای دهدهی در وریلاگ به راحتی می‌توان استفاده کرد.

```
assign w = A + 1230;
```

گاهی اوقات عکس العمل نخواهیم تعداد بیت‌های یک عدد را مشخص کنیم یا عدد را در مبنای ۲ مشخص کنیم.

```
assign w = out + In;
         32   32    16
```

وزارت بازرگانی شرکت بازرگانی دولتی ایران
وزارت بازرگانی شرکت بازرگانی دولتی ایران

برای مشخص کردن یک عدد به صورت زیر عمل می کنیم:

8 ' d 124

8' b 0111100

8' h 7C

علت مشخص کردن تعداد ارقام در نمایش دسیمال:

wire [7:0] w;

wire [15:0] a;

output [15:0] Q;

assign Q = a + {8'd0 , w};

اگر کاربر این را انجام ندهیم، در مرحله synthesis یک warning خواهیم داشت ولی خود synthesizer این کار را انجام می دهد.

* در پایتون هیچ ضروری در مورد تعداد صفر ندارد و باید خود طراح (مثلاً مثل ۲) را در نظر بگیرد.

8'b 1 → 8'b 00000001

اگر تعداد بیت های نوشته شده بیشتر از تعداد بیت مشخص شده بود؟

8'b 1111 0000

8'h f 0

{4'b 1111 , 4'b 0} = {4{1'b1} , 4'b 0}

وزارت بازرگانی شرکت بازرگانی دولتی ایران

* زمانی که نخواهیم تعداد بیت را به صورت دقیق مشخص کنیم از کلیک فوق استفاده می کنیم.

Octal Representation ⟶ 8'o17

حال بیایم w چه حالت های ممکن است داشته باشد؟

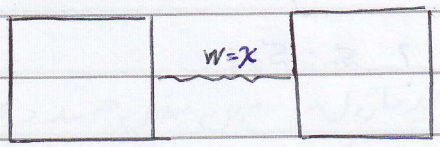
w

1 ⊣

ϕ ⏚

z هیچ module آن را drive نمی کند

x بین زیاد module آن را drive می کند.




```
module counter (out, clk);
output [7:0] out;
input clk;
reg [7:0] out;
always @ (posedge clk)
out <= out + 1;        ← حتما یک داریم
end module
```

این بلوک کاملا غلط است زیرا در ابتدا مقدار اولیه out مشخص نیست و x است و با گذشت زمان هم چنان x باقی می ماند.

وزارت بازرگانی شرکت بازرگانی دولتی ایران
وزارت بازرگانی شرکت بازرگانی دولتی ایران

8'd 129

8'b ...

8'h 7C

wire [7:0]

wire [15:0]

output [15:

assign Q =

8'b 1 ⟶ 8

8'b ...

8'h f ...

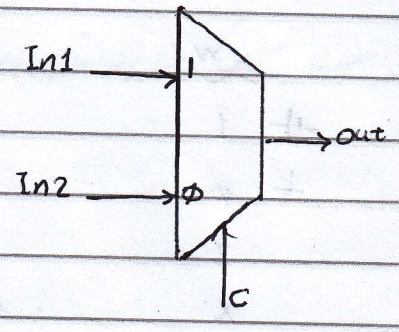
{4'b ..., 4'

همواره باید در مدولهایی که FF داریم از سیگنال Reset استفاده کنیم.

مثالی جهت نشان دادن استفاده جدید از assign :

Example:

```
module mux2 (out, In1, In2, C);
input  In1, In2, C;
output out;
assign out = ( C ) ? In1 : In2;
endmodule:
```

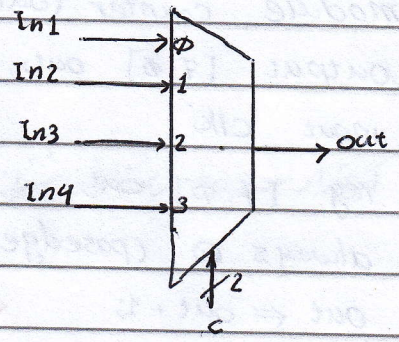


Procedural Continuous Assignment:

assign خروجی = (شرط) ? S_1 : S_2;

Example

```
assign out = ( C == ϕ ) ? In1 :
             ( C == 1 ) ? In2 :
             ( C == 2 ) ? In3 :
                          In4;
```




```
out = (cond1) ? ... :
      (cond2) ? ... :

      : ( ... )
```

* یک شخت ضروری که در صورتی که هیچ کدام از شرطها برقرار نبود این مقداردهی انجام می‌شود.

وزارت بازرگانی شرکت بازرگانی دولتی ایران

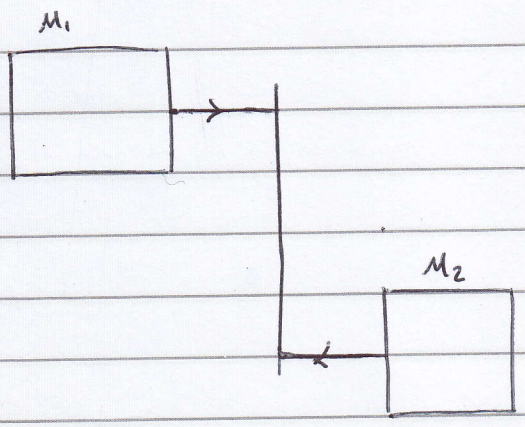
Example:

روشی دیگر برای نوشتن mux2:

```
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
```

* هرکدام از سیگنالها هر تغییری کرد تکه کد بعدی اجرا می‌شود. و با تغییر هر سیگنال بلافاصله خروجی تغییری نمی‌کند و در واقع مانند assign است

فرض در FPGA یک باس مشترک داریم که چند ماژول به آن متصل هستند و چند ماژول آنرا drive می‌کند (در هر لحظه فقط یک ماژول آنرا drive می‌کند) در این حالت در خروجی ماژول باید یک بافر سه حالته داشته باشیم.



می‌توان در این گونه موارد output نوشت. زیرا در این حالت دائماً خروجی را drive می‌کند

وزارت بازرگانی شرکت بازرگانی دولتی ایران

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

وزارت بازرگانی شرکت بازرگانی دولتی ایران