گزارش کار آزمایشگاه معماری

استاد ارجمند:

سرکار خانم مهندس اسدی

تهیه کنندگان:

آرزو گردانی و ساراسکوت

و محبوبه صالحی

اردیبهشت ۱۳۹۶

مدارهای منطقی با دیجتال:

ازساده ترین گیت های منطقی مثل ..... And , or,xor ساخته میشوند تا به مدارهای منطقی مجتمع یاهمان IC ها برسد که متشکل ازچندین گیت های ساده میباشند. با پچیده ترشدن مدارهای منطقی مجتمع یا همان IC ها برسند که متشکل ازچندین بیت های ساده می باشند. با پچیده ترشدن مدارهای منطقی دیجتال قابل برنامه ریزی به نام PLA ارائه شده اند وبعد ازآن مدارهای قابل برنامه ریزی PLA ,PLD نیزبه بازار عرضه شدند وبلاخره مدارهای پچیده تر قابل برنامه ریزی FPGA,CPLD ساخته شده اند.

FPGA, CPLD براساس سلول های منطقی قابل برنامه ریزی طراحی شده اند که ارتباط بین این سلول ها نیز قابل برنامه ریزی میباشد. چون FPGA,CPLD ازنظر برنامه ریزی و کاربرد مشابه به هم هستند. دربرخی نوشته هاCPLD را نوعی FPGA می نامند.

اولین FPGA به وسیله ی شرکت Xilinx ارائه شده ازآن زمان به بعد FPGA های متفاوتی توسط شرکت های مختلف ارائه میشوند.

نکته مهم :

این است که FPGA ها را نباید به Cpu ها ومیکروکنترل هایی مثلARM, AVR اشتباه گرفت زیرا آنها خودشان ساختار سخت افزار ازپیش آماده دارند و فقط کافی است بازبان مورد نظرشان به آنها دستوراتی داد تا اجرا کنند ولی FPGA ها درابتدا داخلشان هیچ چیزنیست و این شمایید که با زبان VHDL سخت افزار موردنظرتان را پیاده سازی کنید. مثلا میتوانید همان میکروکنترلر را داخل FPGA پیاده سازی کنید.درواقع ساختار سخت افزار آنها کاملا متغییراست.

نرم افزار ISE :

قسمت بزرگی از کارکردن با FPGA مربوط به زبان VHDL میباشد.باتوجه به این که این زبان وابستگی به هیچ نرم افزاری ندارد میتوان برنامه VHDL نوشته شده برای یک FPGA شرکت Altera را برروی یک FPGA شرکت Xilinx ریخت با این تفاوت که باید مجددا توسط کامپایلر ساخته شود.

برنامه نویسی وطراحی FPGA به طورکلی به دوصورت انجام می پذیرد:

- الف) با استفاده از زبان های توصیف سخت افزارمی باشد.

نظیر: VHDL,Verilog,HDL

- ب) با استفاده از طراحی مدار

VHDL :یکی از زبانهای توصیف سخت افزار می باشد.ازگیت های ساده AND, OR گرفته تا CPU یعنی هرمدل سخت افزار دیجتال VHDL توصیف کرد ورفتار آن را روی کاغذ آورد بدون نیاز به این که بخواهیم سخت افزار مورد نظر را به صورت کیفی توصیف کنیم. این زبان برای برنامه ریزی در Program ، Ic های نظیر cpld,pld,pla ها والبته FPGA استفاده میشود.

نرم افزار ISE سه پنجره اصلی دارد:

۱. Source :

می شود به snaption سورس ها وفایل هایی که ساخته می شود دسترسی پیداکرد.

۲. Processes :

وقتی فایلی ازقسمت سورس انتخاب می کنیم داخل این بخش عملیاتی که باید به روی فایل انجام شود نشان داده می شود.

۳. Transcript :

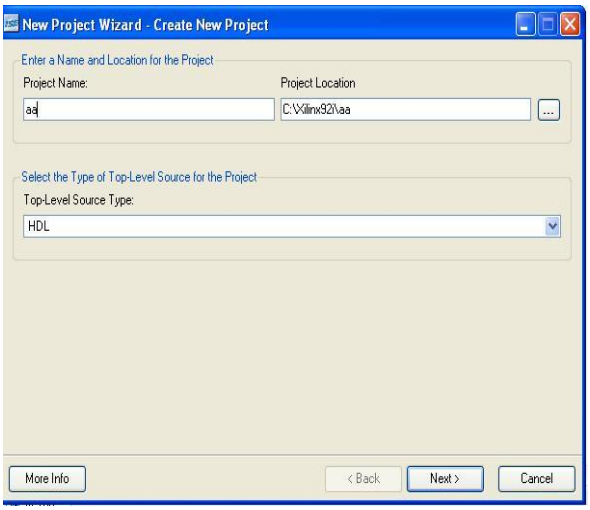
وقتی عملیات را درProcesses روی فایل انجام می دهد دراین قسمت گزارش روند عملیات داده می شود.

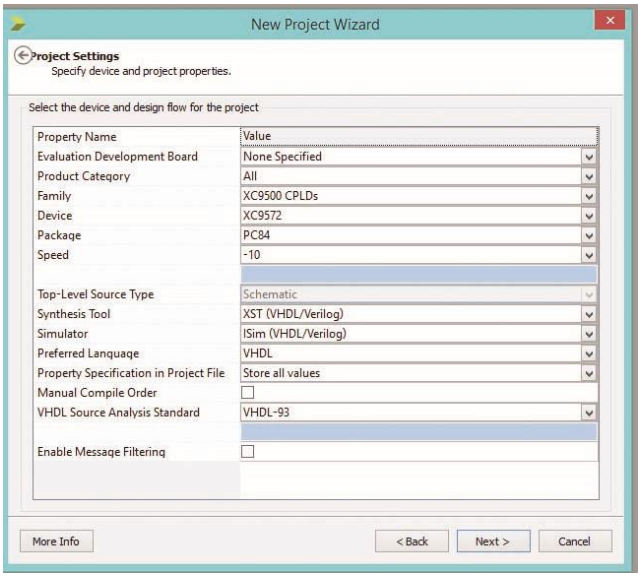
ابتدازمانی که نرم افزار رابازکردیم یک پروژه جدید ایجاد می کنیم.مراحل به صورت زیراست:

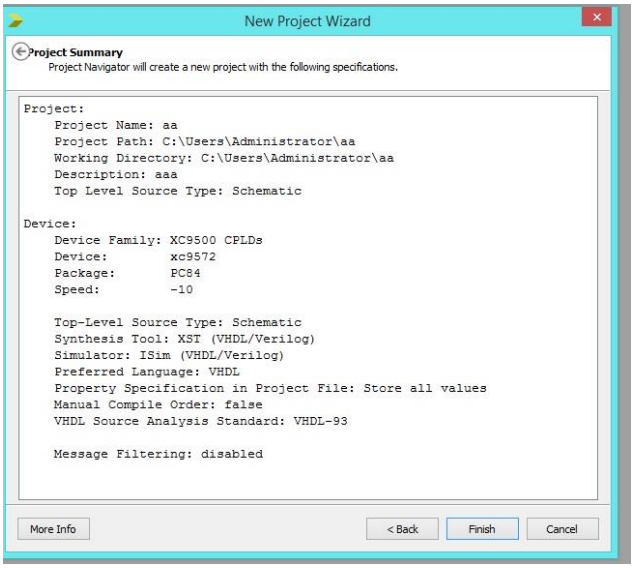
ازمنوی file گزینه ی new project وسپس نام برای پروژه انتخاب کرده وسپس قسمت Top Level Source که حالت توصیف سخت افزار برای برنامه نویسی به زبان VHDL انتخاب می کنیم و سپس گزینه ی Next را انتخاب میکنیم.که این قسمت گزینه ی Schematic را انتخاب می کنیم وسپس گزینه ی Next را انتخاب میکنیم.

در صفحه بعد باید تنظیمات به صورت زیرباشد:

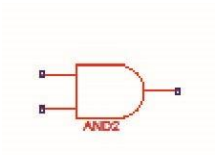
Product Category	All
Family	Mc۹۵۰۰ CpldS
Device	Xc۹۵۷۲
Package	Pc۸۴
Speed	-۱۰
Top-Level Source Type	schematic
Synthesis Tool	XST (VHDL/Verilog)
Simulator	LSE Simulator (VHDL/Verilog)
Preferred Language	VHDL



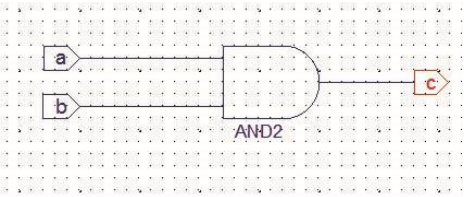




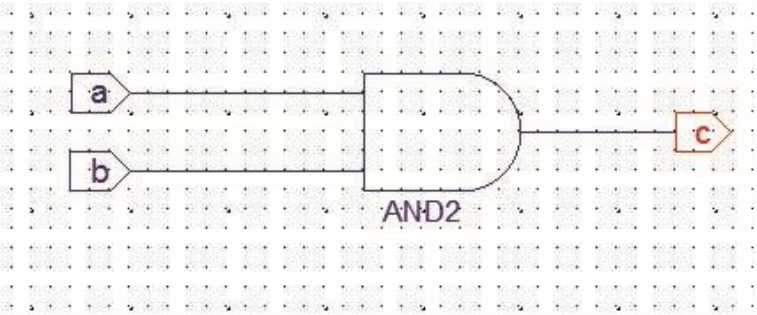
سپس Next و درآخرین مرحله finish را انتخاب می کنیم. سپس درصفحه ای که باز می شود درپنجره source قسمت categories گزینهAll symbols را انتخاب می کنیم وسپس از قسمت symbols گیت ۲ And را انتخاب می کنیم که به شکل زیر می باشد.

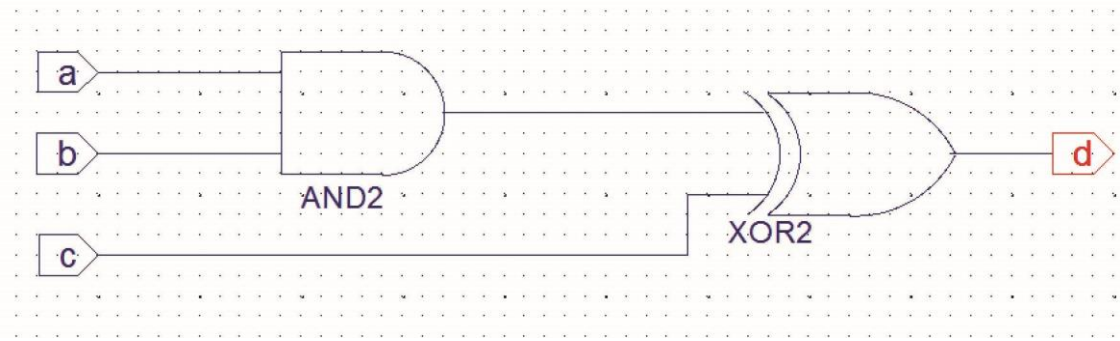


ابزار Add wire را به ورودی وخروجی ۲ And وصل می کنیم ابزار Add Io / marker wire را باید به ورودی وخروجی وصل کنیم به آن یک اسم مثل A می دهیم.



زمانی که مدار راطراحی کردیم بایدآن راکامپایل کنیم که برای کامپایل کردن ابتدا فایل را save می کنیم وسپس درپنجره source موردنظر را انتخاب ودرقسمت process را انتخاب وروی گزینه Impalement Design دابل کلیک می کنیم تاشروع به کامپایل شدن کند.زمانی پیغام مناسب نمایش داده شده پیغام مبنی برکامل شدن کامپایل درپنجره source گزینه new source ایجادکرده و درپنجره ای که بازمی شود گزینه ی Test bench waveform راانتخاب وسپس وسپس گزینه ی وfinish را انتخاب کرد. پنجره ای بازمی شود روی قسمت آبی رنگ ها کلیک کرده وسپس درپنجره ی source قسمت source to گزینه ی Behavioral Simulation را انتخاب کرده وسپس فایل source مورد نظر را انتخاب کرده واز قسمت Process گزینه ی Xilinx Ise simulator وسپس گزینه ی Behavioral mode را انتخاب می کنیم که ببینیم مدار درست کار میکند یا خیر؟

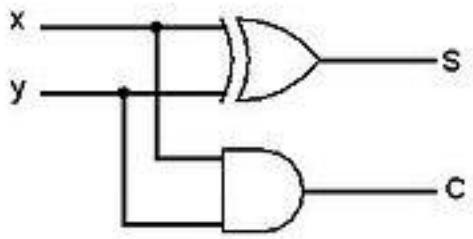




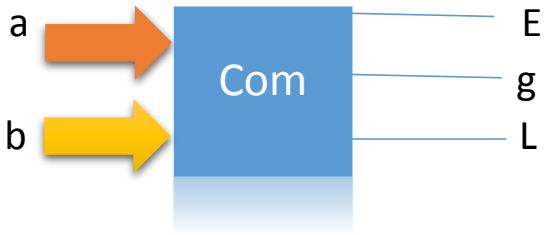
مدارنیم جمع کننده:

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

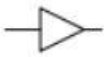
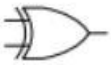
جدول درستی



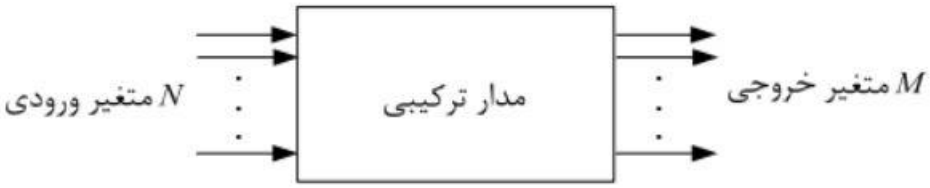
دیاگرام مدار منطقی نیم جمع کننده



انواع گیت های منطقی :

نام گیت	نماد	تابع جبری	جدول درستی
AND		$f = xy$	x y f 0 0 0 0 1 0 1 0 0 1 1 1
OR		$f = x + y$	x y f 0 0 0 0 1 1 1 0 1 1 1 1
NAND		$f = \overline{xy}$	x y f 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$f = \overline{x + y}$	x y f 0 0 1 0 1 0 1 0 0 1 1 0
NOT		$f = \bar{x}$	x f 0 1 1 0
Buffer		$f = x$	x f 0 0 1 1
XOR		$f = x \oplus y$	x y f 0 0 0 0 1 1 1 0 1 1 1 0
XNOR		$\overline{f = x \oplus y}$	x y f 0 0 1 0 1 0 1 0 0 1 1 1

مدارهای منطقی ترکیبی:

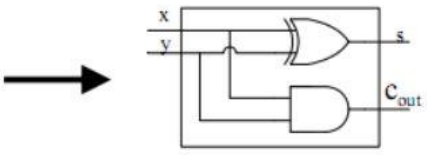


۱. نیم جمع کننده:

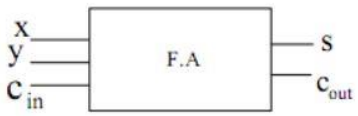
$$\begin{array}{r} x \\ +\ y \\ \hline c_{out}\ s \end{array}$$

x	y	s	c_{out}
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

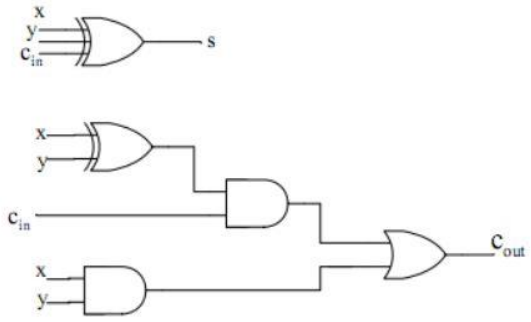
$$\begin{cases} s = x \oplus y \\ c_{out} = xy \end{cases}$$



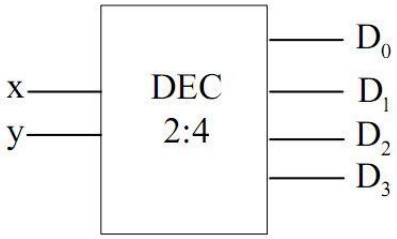
۲. تمام جمع کننده:



$$\begin{cases} s = x \oplus y \oplus c_{in} \\ c_{out} = xy + c_{in}(x \oplus y) \end{cases}$$

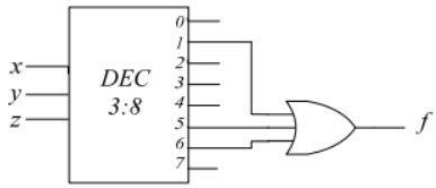


۳. دیکدر:

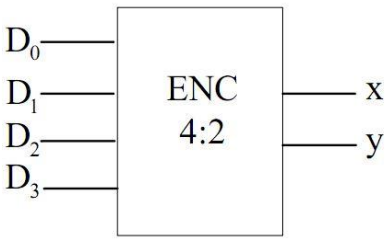


y	x	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

پیاده سازی تابع $f(z,y,x)=\sum m(1,5,6)$ با استفاده از دیکدر:

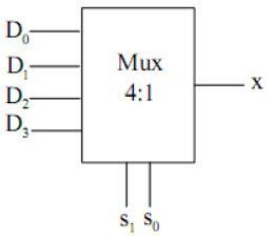


۴. انکدر:



D_3	D_2	D_1	D_0	y	x
0	0	0	1	0	0
0	0	1	0	0	1
1	1	0	0	1	0
1	0	0	0	1	1

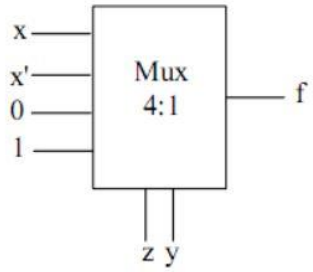
۵. مالتی پلکسر:



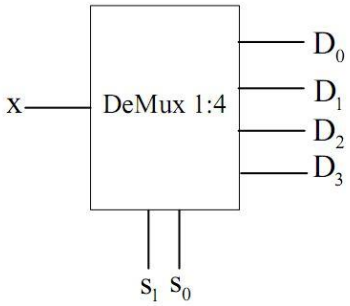
s_1	s_0	x
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

پیاده سازی تابع $f(z, y, x) = \sum m(1,2,6,7)$ **با استفاده از مالتی پلکسر ۴ به ۱** (برای پیاده سازی یک تابع n متغیره به یک ماکس 2^{n-1} بیتی نیاز است)

z	y	x	f	
0	0	0	0	f=x
0	0	1	1	
0	1	0	1	f=x'
0	1	1	0	
1	0	0	0	f=0
1	0	1	0	
1	1	0	1	f=1
1	1	1	1	

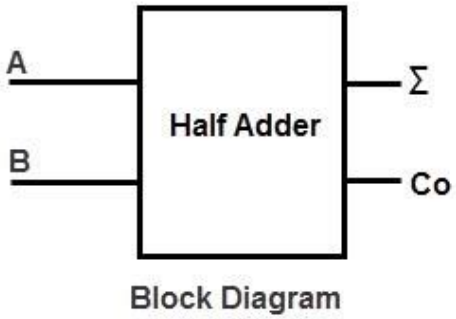


۶. دی مالتی پلکسر:



s_1	s_0	D_3	D_2	D_1	D_0
0	0	0	0	0	x
0	1	0	0	x	0
1	0	0	x	0	0
1	1	x	0	0	0

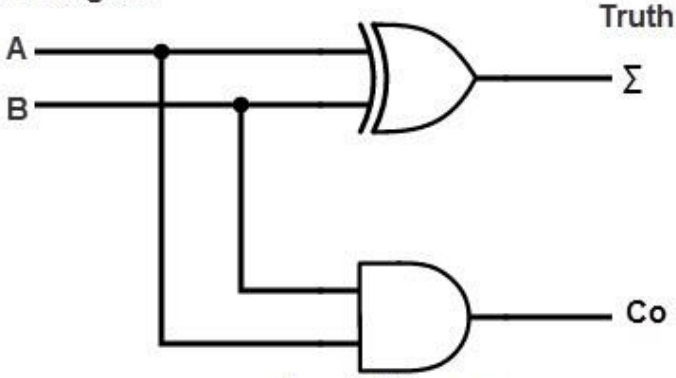
:HAFF adder



Block Diagram

A	B	Sum (∑)	Carry Out (Co)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

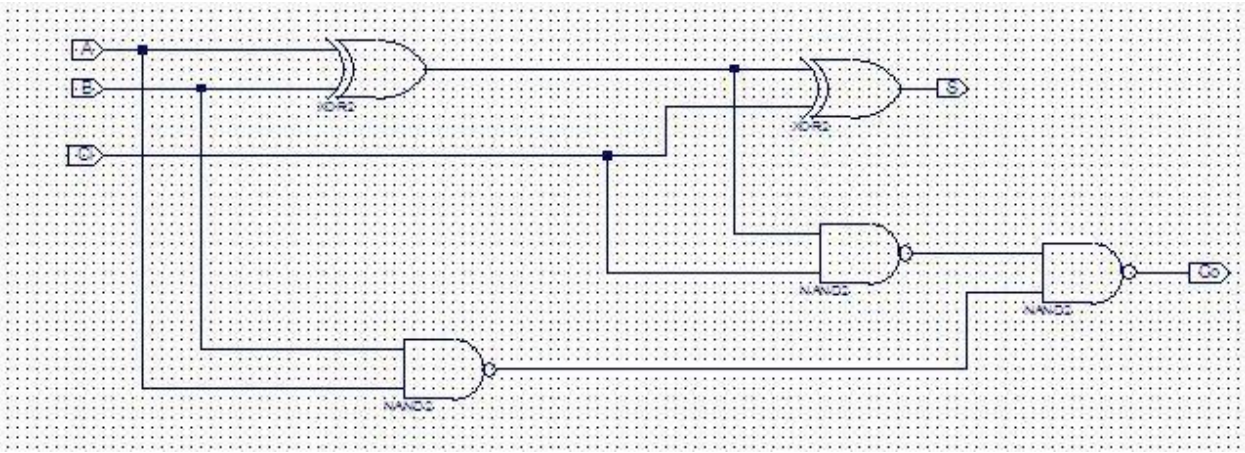
Truth Table



Logic Diagram

: Full adder

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



```
S<= (a xor b) xor ci;
Co<= (a nand b) nand (ci nand (a xor b));
```

معرفی پورت های ورودی وخروجی به وسیله جمله ;( )port انجام می شود. داخل پرانتز ابتدا نام ورودی ها وخروجی ها نوشته می شود. اگر برنامه چندین port ورودی وخروجی داشته باشد می توان همه ورودی ها ویا همه خروجی ها را کنارهم با علامت ، بینشان وارد کنیم . بعد ازنام پورت علامت : گذاشته می شود و بعد نوع port که ورودی یا خروجی است مشخص شود.

سپس با وجود یک space (فاصله) جنس پورت مشخص می شود که تا این جا به صورت پیش فرض std-logic انتخاب می شود.

درقسمت پورت برای معرفی ورودی ها وخروجی ها باید بینشان علامت ; قرار داده شود.

قسمت اصلی برنامه قسمت سوم آن می باشد که بدنه اصلی برنامه است.

قسمت سوم برنامه architecture می باشد.که معماری اصلی برنامه دراین قسمت انجام می شود.

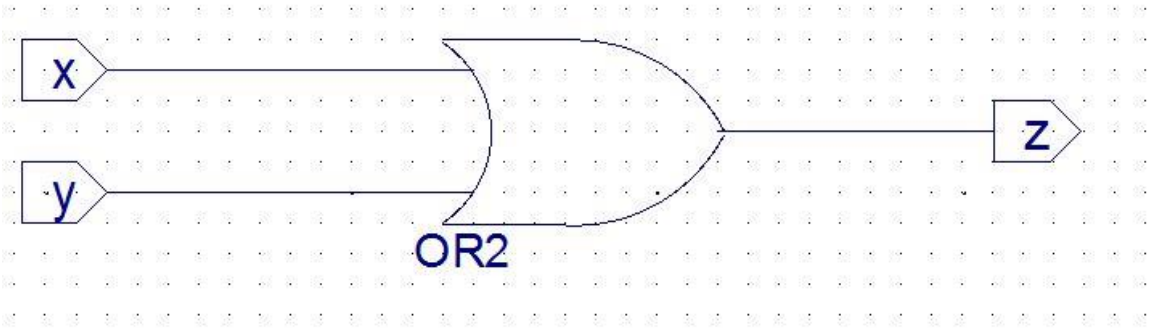
قسمت سوم با جمله architecture Behavioral of aa is شروع می شود وبا جمله end Behavioral تمام می شود.

برای نوشتن دستورات برنامه ابتدا جمله begin درقسمت architecture وارد میشود وبعد دستور برنامه نوشته می شود.

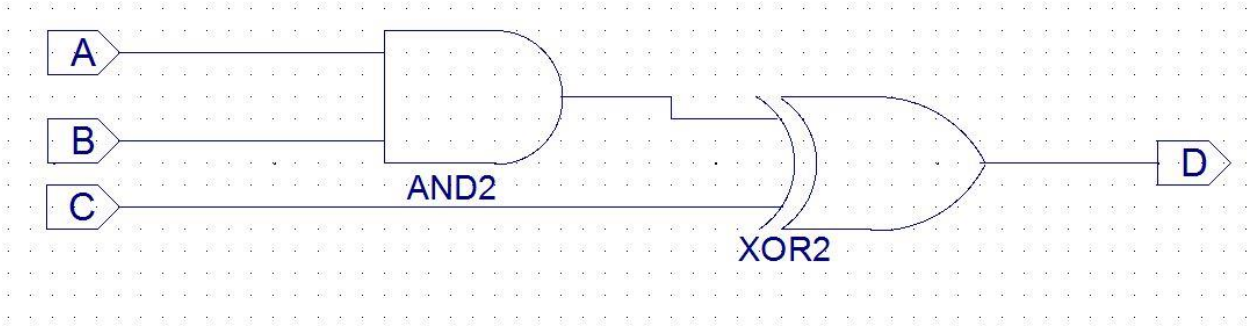
نکته: تمامی گیت های منطقی برای زبان VHDL تعریف شده است وزمانی که تایپ شوند با رنگ آبی مشخص می شوند.

نکته : برای مقداردهی به خروجی ازعلامت کوچکتر مساوی استفاده می شود.

نکته: برای نوشتن اعداد منطقی تک بیتی داخل برنامه آن ها را داخل' ' قرارمی دهیم وبرای نوشتن اعداد منطقی چندبیتی داخل برنامه " " قرار می دهیم.



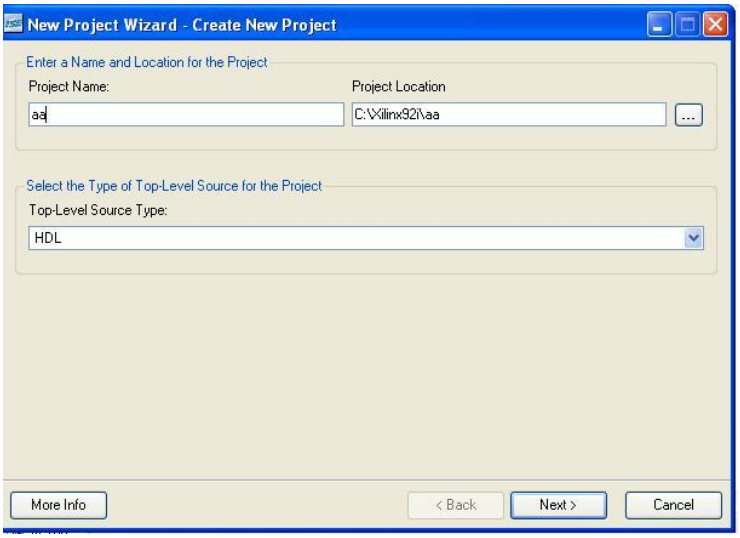

```
Z<=x or y;
```

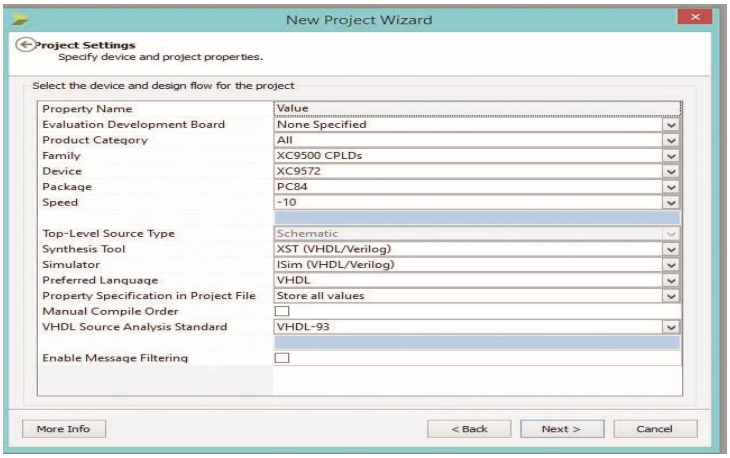



```
d<=(a and b) xor c
```

برنامه نویسی به زبان VHDL :

از فایل سمت چپ Projeh را انتخاب می کنیم بعد new project اسم جدید می دهیم و گزینه Top level source و گزینه HDL انتخاب می کنیم وسپس درپنجره بازشده (device properties) تنظیمات اول برنامه را انتخاب می کنیم.





بعد تنظیمات گزینه next و finish رامی زنیم.

روی source کلیک راست کرده و گزینه new source و یک اسم به فایل می دهیم VHDL MODULE را انتخاب میکنیم.

نکته : توجه داشته باشید که اسم ماژولها نباید گیت یا اسم های خاص باشد.

سپس گزینه ی NEXT را می زنیم.سپس یک پنجره جدید بازمی شود که دراین جا باید ورودی ها وخروجی ها را X,Y وخروجی را Z انتخاب می کنیم.

در قسمت Direction اسم پورت های ورودی وخروجی وارد می شود. سه پورت z,x,y راوارد می کنیم.

X in (ورودی)

Y in(ورودی)

Z out(خروجی)

درقسمت port name اسم پورت های ورودی وخروجی راوارد کرده ودرقسمت

Direction باید اسم پورت های ورودی وخروجی وارد شود.

اگرورودی وخروجی ها تک بیتی نبود Bus را انتخاب می کنیم وسپس گزینه next را انتخاب می کنیم.

Summary قسمت زیادی ازبرنامه راخودش می نویسد.

Library کتابخانه VHDL است. بخش دوم Entity a اطلاعات مدار وارد می شود با فایل های اطلاعات پورت وارد می شود ودر این بخش شیوه وارد کردن اطلاعات پورتها به این صورت است . پورت ();

داخل پرانتز ابتدا اسم پورت نوشته می شود وبعد علامت دونقطه (:) گذاشته می شود بعد ازآن اسم پورت که ورودی وخروجی مشخص می شود وسپس space که نوع پورت است را انتخاب می کنیم. نوع پوت sw کاملترین نوع پورت است که انتخاب می کنیم.

داخل VHDL باید ; گذاشته شود. درزبان VHDL یک سری کلمات کلیدی وجود دارد که هرکدام معنای خاصی دارد برای مثال دوخط تیره یا – برای ارائه توضیحات دربرنامه استفاده می شود وبرای برنامه ندارد پس هرکجای بخواهیم بنویسیم دوخط تیره می گذاریم وتا پایان خط می توانیم گزارش را بنویسیم اگر بیشتر از یک خط بود علامت خط تیره را قرار می دهیم.

archtive behavioral offaais: بعد ازاجرای begin شروع می شود وبا دستور end behavioral شروع می شود بعد begin واردمیشود بعد دستور نمی شودوbehavioral وارد می شود.

درزبان VHDL برای مقداردهی به پارامتر ها کوچکتر بزرگتر نوشته می شود . انتهای دستور ; هست . Or آبی می شود ودستور را در archtive وارد می کنیم.

Z<x or y

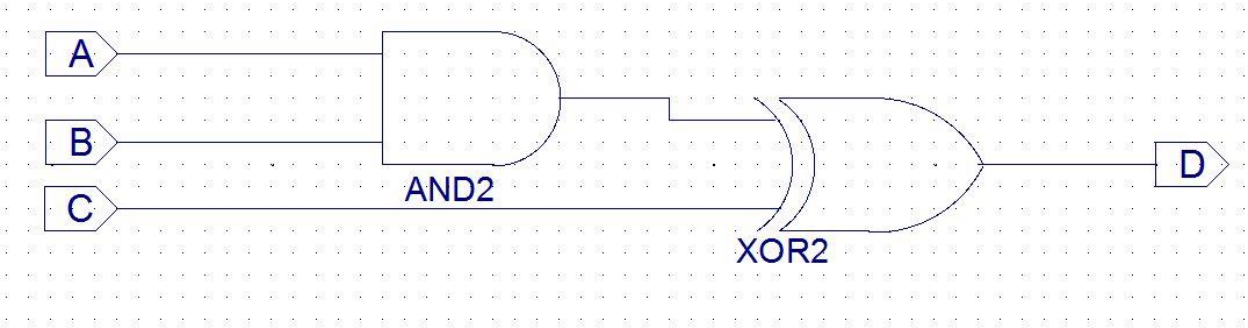
وقتی کامل شد برنامه را save را انتخاب می کنیم . هربرنامه ایی که شما بنویسید ابتدا باید کامپایل بشود بعد باید به خروجی برویم. درقسمت source پسوند VHDL را انتخاب کرده سپس روی آن کلیک کرده بعد قسمت procees ، Increment design را دابل کلیک می کنیم بعدشروع به کامپایل می کند . پنجره جدید بازمی کند که پیغام Succefully می دهد.

دستورورود مدارمنطقی:

دربرنامه نویسی به زبان VHDL برای مقادیر تک بیتی ازعلامت سینگل کوتشین استفاده می شود وبرای دوبیتی " " استفاده می شود.

درقسمت archtive

d<=a(a and b) or c



لیست حساسیت process :

یک ساختارطراحی VHDL است که برای شکل دهی به الگوریتم می باشد. اگر process دربرنامه اضافه شود تنها زمانی که لیست حساسیت تغییرکند خروجی ها محاسبه می گردند ودرتوان مصرفی cpu وزمان مصرفی محاسبات صرفه جویی می شود.

لیست Process میتوان شامل یک نام اختیاری باشد.برای مثال این نام : مجزا می شود.

Ha: نام لیست process اختیاری است که مخفف (Haff adder) است.

```
HA: process(x, y) is
```

لیست حسایست که درجلوی کلمه process آورده می شود نشان دهنده این است که process براساس این سیگنال ها اجرا می گردد یا اصطالحا حساس می باشد که دراینجا سیگنالهای x,y لیست حساسیت process هستند.

Process بعداز begin archtive آورده می شود وبعد ازآن دومرتبه کلمه begin آورده شود وبعد ازآن دومرتبه کلمه begin آورده می شود. archtive میتواند شامل چندین process باشد که موازی باهم اجرا می شود.

```
Process(x,y)is
```

Process با این جمله آغاز می شود ودرانتهای آن لزومی به علامت ; نمی باشد.

لیست process باجمله end process; به پایان می رسد.

نکته: در process میتوان انواع دستوراتی که در VHDL وجود دارد استفاده کرد. به کمک این دستورات می توانیم نحوه ی ارتباط بین ورودی ها وخروجی ها رابرقرار کنیم.

```
Process(x,y) is
Begin
C<=x and y;
S<=x oxr y;
End process;
```

متغییرواسط:

نکته : از متغییرها درprocess وزیر برنامه ها می توان استقاده کرد ودرمحدوده process یازیربرنامه باید معرفی شود.

نکته: از یک متغییر دردو process نمی توان استفاده شود.

نکته: متغییرها برخلاف سیگنال ها درای اجزای حافظه ای نیستند بنابراین تاخیر ندارند و معمولا برای اهداف محاسباتی استفاده می شود، متغییرها فقط مقداری را دریک زمان نگه می دارند و نمی توان شکل موج آنها رامشاهده کرد.

نکته: درطول برنامه برای مقداردهی به متغییرها ازنماد =: استفاده می شود.

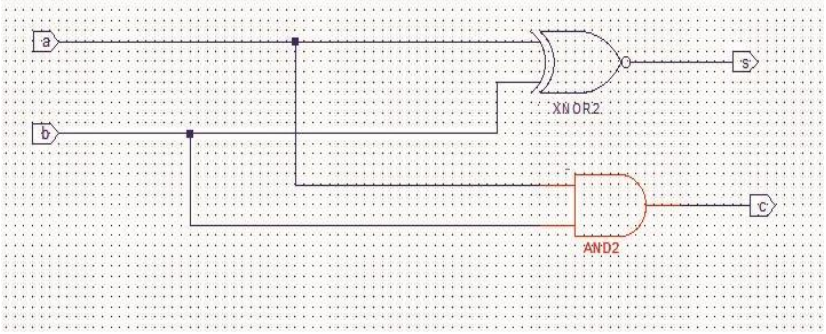
ممکن است در یک process ازچندین متغییرواسط استفاده شود که همه آنها را درکنار هم معرفی می کنیم وبا علامت ; آنها را مجزا می کنیم برای مثال:

```
Variable a,b : STD_LOGIC;
```

a : اسم متغییرواسط

STD_LOGIC: جنس متغییر واسط که اکثرا موارد STD_LOGICانتخاب می شود.

مدارHAFF ADDER:



برنامه Haff adder:

```
Process(x,y) is
Begin
C<=x and y;
S<=x oxr y;
End process;
```

……

```
S<=(a oxr b) xor c;
Co<=(a nand b) nand (ci nand(a xor b));
Msb=۳
Lsb=o
```

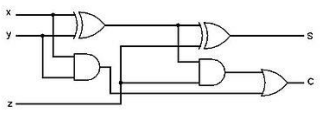
دیکدر(Decoder):

```
C<=a or b ;
```

مدار full adder:

z	x	y	s	c
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

جدول درستی



دیاگرام مدار منطقی تمام جمع کننده

```
Process (a,b, ci)is
Variable w: std_logic;
Variable d: std_logic;
Variable x: std_logic;
Begin
W:= a xor b;
D:= (a nand b);
x:=ci nand w;
s<= ci xor w;
End process;
```

Schematicمالتی پلکسر:

برنامه نویسی برای mux مالتی پلکسر:

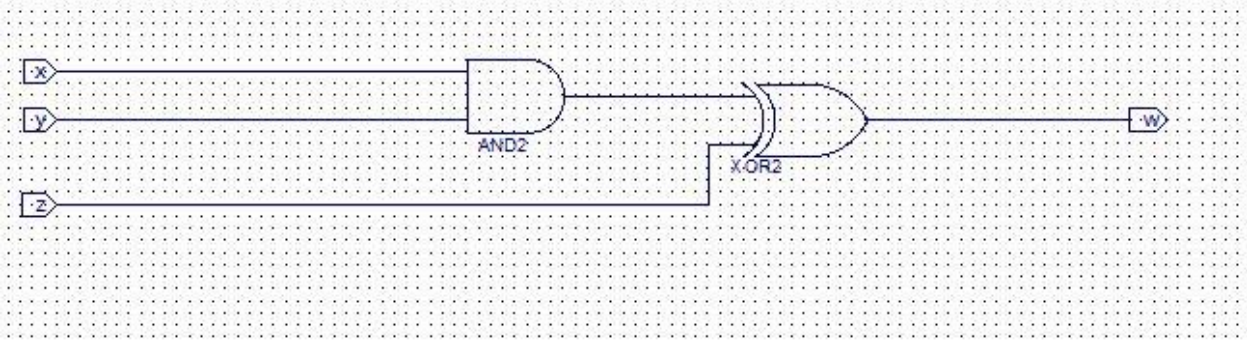
```
Process (i۰, i۱,s)
Variable a: std_logic;
Variable b: std_logic;
Begin
a:=i۰ and (not s);
b:= s and b;
۰<= a and b;
End process;
```

: Mux ٤ *١

```
IF- the-else
if(condition)then
do something;
else
do something different
end if;
```

……

```
Process (i۰, i۱, s)
Begin
If(s='۰') then
۰<=io;
Else if(s='۱') then
o<=i۱;
End if;
End process;
```



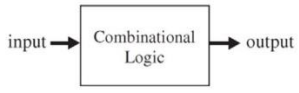
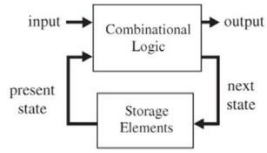

```
Architecture Behavioral of A is
Process(x,y,z) is
Variable a: STD_LOGIC
Begin
Process(x,y,z) is
Variable a: STD_LOGIC
A: =x and y;
w <= a or z;
End process;
End behavioral;
```

……

```
process (x,y,z)is
variable a:std_logic;
variable b:std_logic;
begin
a:=x and y;
b: a xor z;
w<= a nand b;
end if;
end process;
```

شکل مداربالا

ساختارمدارترتیبی وترکیبی:

مدارهای منطقی ترکیبی	مدارهای منطقی ترتیبی
	
ساختارها و دستورات ترکیبی در زبان VHDL	ساختارها و دستورات ترتیبی در زبان VHDL
When/Else With/Select For/Generate Block	IF Case Loop Variable Process Function Procedure

دستورات ترکیبی

۱- دستور When- else

مثال	قالب دستور
outp <= "000" WHEN (inp='0' OR reset='1') ELSE "001" WHEN ctl='1' ELSE "010";	نام سیگنال <= مقدار ۱ when شرط ۱ else مقدار ۲ when شرط ۲ else ..... مقدار آخر;

۲- دستورWhich - select

مثال	قالب دستور
With a select b <= d0 when “000” , d1 when “001” , d2 when “002” , Unaffected when others;	With سیگنال شرط Select نام سیگنال <= مقدار ۱ when مقدار سیگنال شرط , مقدار ۲ when مقدار سیگنال شرط , ... مقدار آخر when others;

۳- دستور for/generate

مثال	قالب دستور
OK: for i in 0 to 7 generate output(i) <= '1'1 when (a(i) and b(i))='1' else '0'; end generate;	برچسب (اجباری): for نام سیگنال in ابتدای حلقه to انتهای حلقه generate begin عبارت; .... عبارت; end generate;

Check:
If (Reset = '1') generate
begin
Output <= '0';
End generate;
	برچسب (اجباری):
	if شرط generate
	begin
	عبارت;
	....
	عبارت;
	end generate;

۴- ساختار Block:

برچسب (اجباری): Block معرفی سیگنال های مورد نظر; Begin عبارت; .... عبارت; End Block برچسب;	b1: BLOCK SIGNAL a: STD_LOGIC; BEGIN a <= input_sig WHEN ena='1' ELSE 'Z'; END BLOCK b1;

برچسب (اجباری): Block (عبارت شرط)
معرفی سیگنال های مورد نظر;
Begin
عبارت (مشروط یا غیر مشروط);
....
عبارت (مشروط یا غیر مشروط);
End Block برچسب;
	b1: BLOCK (clk='1')
	BEGIN
	q <= GUARDED d;
	END BLOCK b1;

دستورات ترتیبی عبارتند از:

۱) If
۲) Wait
۳) Case
۴) Loop
۵) Variable

این دستورات تنها در بلوک های زیر می نوانند نوشته شوند:

Process

Function

Procedure

بلوک های ترتیبی

۱) Process :

قالب بلوک Process	مثال

(برچسب اختیاری) :
Process (لیست سیگنال هایی که پروسس با تغییر آن ها اجرا می شود)
Begin
دستورات ترتیبی;
END Process (برچسب اختیاری) ;
	Process(clk,rst)
	Begin
	If (rst = '1') then
	q <= '0';
	elsif (clk'event and clk = '1') then
	q <= d;
	end if;
	End process;

برای محدود کردن اجرای برنامه به یکی از لبه های کلاک باید ازدستورات زیر در برنامه استفاده کنیم:

- If (clk'event and clk = '1') then ..... مدار به لبه بالا رونده کلاک حساس است
- If (rising_edge(clk)) then ..... مدار به لبه بالا رونده کلاک حساس است
- If (clk'event and clk = '0') then ..... مدار به لبه پایین رونده کلاک حساس است
- If (falling_edge(clk)) then ..... مدار به لبه پایین رونده کلاک حساس است

دستورات ترتیبی :

۱) Variable , single :

در VHDL برای تعریف سیگنال های میانی ازدروش زیر استفاده می شود.

Single:

Variable:

قالب دستور	مثال
signal نام سیگنال : نوع سیگنال := مقدار اولیه;	signal control: bit := '0' signal count: integer range 0 to 100; signal y: std_logic_vector(7 downto 0);
variable نام متغیر : نوع متغیر := مقدار اولیه ;	variable control: bit := '0' variable count: integer range 0 to 100; variable y: std_logic_vector(7 downto 0);

مقایسه

	Variable	Signal
محل استفاده	بلوک های ترتیبی	Architecture و Package
محلی/سراسری	مقدار variable محلی است و تنها داخل بلوک ترتیبی می تواند مقدار دهی شود و یا مقدار آن خوانده شود.	مقدار Signal سراسری است و در کل برنامه می توان مقدار آن را خواند یا آن را مقدار دهی کرد.
زمان مقداردهی	مقدار آن بلافاصله آپدیت می شود.	وقتی داخل بلوک های ترتیبی مقداردهی شود مقدار آن پس از پایان اجرای بلوک آپدیت می شود.
علامت تخصیص مقدار	:=	<=

۲) if

قالب دستور	مثال

If شرط then
دستورات;
Elsif شرط then
دستورات;
End if;
	`if (rst='1') then`
	`  output <= "00000000";`
	`elsif (clk'event and clk='1') then`
	`  output <= input;`
	`end if;`

۳) Wait

قالب دستور	مثال

Wait Until شرط;	`Process  --no sensitivity list`
	`Begin`
	`  wait until (clk'event and clk='1');`
	`  if (rst='1') then`
	`  output <= "00000000";`
	`  elsif (clk'event and clk='1') then`
	`  output <= input;`
	`  end if;`
	`End Process;`

Wait ON ... و سیگنال ۲ سیگنال۱;	`Process  --no sensitivity list`
	`Begin`
	`  wait on rst, clk;`
	`  if (rst='1') then`
	`  q <= '0';`
	`  elsif (clk'event and clk='1') then`
	`  q <= d;`
	`  end if;`
	`End Process;`

Wait For زمان;	`wait for 10 ns;` `output <= '1';`

۴) Case :

مثال	قالب دستور

case sel is
when "00" => x<=a;
when "01" => x<=b;
when "10" => x<=c;
when others => x<=d;
end case;
	Case نام سیگنال is
	When مقدار => دستورات;
	……
	When مقدار => دستورات;
	END Case;

۵) Loop:

مثال	قالب دستور
for i in 1 to inp'high loop outp(i) <= inp(i-1); end loop;	(برچسب اختیاری): FOR نام سیگنال حلقه IN رنج LOOP کدهای ترتیبی; END LOOP(برچسب اختیاری);

while (i<10) loop
wait until clk'event and clk='1';
q <= a;
end loop;
	(برچسب اختیاری): While شرط حلقه LOOP
	کدهای ترتیبی;
	END LOOP(برچسب اختیاری);

درحلقه for از دستورات زیر می توان استفاده کرد:

Exit: برای خارج شدن از حلقه:

Exit(when شرط);

مثال)

For I in to ۱۵ loop

Next when i=skip; -- jumps to next iteration

(…)

End loop;

:Next

برای پرش به اندیس بعدی حلقه

Next (when شرط);

```
FOR i IN 0 TO 15 LOOP
   NEXT WHEN i=skip;     -- jumps to next iteration
      (...)
END LOOP;
```

پیاده سازی مدارهای ترکیبی با استفاده از دستورات ترتیبی

نحوه پیاده سازی مالتی پلکسر ۱*۴

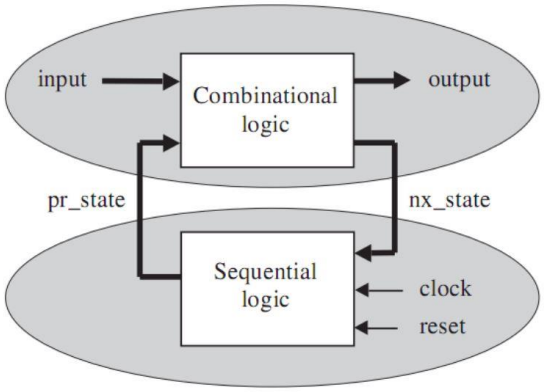
```
architecture archi of mux is
  begin
    process (a, b, c, d, s)
    begin
      if    (s = "00") then o <= a;
      elsif (s = "01") then o <= b;
      elsif (s = "10") then o <= c;
      else o <= d;
      end if;
  end process;
end archi;
```

ماشین حالت

۱- میلی: در ماشین حالت میلی، وضعیت خروجی به حالت فعلی و ورودی های فعلی بستگی دارد.

۲- مور: در ماشین حالت مور، خروجی تنها به وضعیت فعلی بستگی دارد.

شکل زیر بلوک دیاگرام کلی یک ماشین حالت را نشان می دهد:



برای تعریف یک ماشین حالت ابتدا باید متغیری از نوع شمارشی تعریف کرد.

Type نام **is** (مقادیر مورد نظر)

برای مثال برای تعریف ماشین حالتی که چهار حالت دارد باید متغیری مانند زیر تعریف کرد. نام اختصاص داده شده به حالت ها اختیاری است.

Type my_state **is** (s0,s1, s2, s3)**;**

حالت فعلی و حالت بعدی سیگنال های داخلی هستند که نوع آن ها باید مطابق با نوع شمارشی تعریف شده باشد. مثال:

Signal pr_state, nx_state: my_state**;**

۱- میلی: در ماشین حالت میلی، وضعیت خروجی به حالت فعلی و ورودی های فعلی بستگی دارد.

۲- مور: در ماشین حالت مور، خروجی تنها به وضعیت فعلی بستگی دارد.

شکل زیر بلوک دیاگرام کلی یک ماشین حالت را نشان می دهد:

قالب کلی کد به قسمت ترتیبی صورت زیر است:

در این قسمت با لبه کلاک سیستم به حالت بعدی می رود. در صورتی که ریست مدار فعال شود سیستم به حالت اولیه برمی گردد.

```
PROCESS (reset, clock)
BEGIN
   IF (reset='1') THEN
      pr_state <= state0;
   ELSIF (clock'EVENT AND clock='1') THEN
      pr_state <= nx_state;
   END IF;
END PROCESS;
```

قالب کلی قسمت ترتیبی :

دراین قسمت خروجی ها مقداردهی می شود و حالت بعدی نیز مشخص می شود.

```
PROCESS (input, pr_state)
BEGIN
   CASE pr_state IS
      WHEN state0 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state1;
         ELSE ...
         END IF;
      WHEN state1 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state2;
         ELSE ...
         END IF;
      WHEN state2 =>
         IF (input = ...) THEN
            output <= <value>;
            nx_state <= state2;
         ELSE ...
         END IF;
      ...
   END CASE;
END PROCESS;
```

قالب کلی طراحی ماشین:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
---------------------------------------------------
ENTITY <entity_name> IS
   PORT ( input: IN <data_type>;
          reset, clock: IN STD_LOGIC;
          output: OUT <data_type>);
END <entity_name>;
---------------------------------------------------
ARCHITECTURE <arch_name> OF <entity_name> IS
   TYPE state IS (state0, state1, state2, state3, ...);
   SIGNAL pr_state, nx_state: state;
BEGIN
   ---------- Lower section: -----------------------
   PROCESS (reset, clock)
   BEGIN
      IF (reset='1') THEN
         pr_state <= state0;
      ELSIF (clock'EVENT AND clock='1') THEN
         pr_state <= nx_state;
      END IF;
   END PROCESS;
   ---------- Upper section: ----------------------
   PROCESS (input, pr_state)
   BEGIN
      CASE pr_state IS
         WHEN state0 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state1;
            ELSE ...
            END IF;
         WHEN state1 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state2;
            ELSE ...
            END IF;
         WHEN state2 =>
            IF (input = ...) THEN
               output <= <value>;
               nx_state <= state3;
            ELSE ...
            END IF;
         ...
      END CASE;
   END PROCESS;
END <arch name>;
```

مقدمه :

FPGAچیست؟ FPGAاز تعداد بسیار زیادی گیت منطقی در داخل خودش ساخته می شود. این آی سی نسبت به عدم نویز پذیری بسیار مقاوم تر از انواع میکروکنترلر ها است. ویژگی مثبت FPGA ها نسبت به تراشه های میکروپروسسورها و میکروکنترلر ها و DSP ها این است که می تواند به صورت موازی بستورات را اجرا کند. همچنین توسط این آی سی می توان هر نوع میکروکنترلر و میکروپروسسوری را پیاده سازی نمود. به دلیل پردازش موازی این آی سی در صنایع مخابرات جهت سوییچ های مخابراتی و لایه های شبکه استفاده می شود. همچنین در صنایع نظامی جهت کنترل موشک استفاده می شود. این آی سی می تواند به عنوان پردازش سیگنال و تصویر نیز استفاده شود که این ویژگی توسط نرم افزار MATLAB پشتیبانی می شود. با استفاده از این آی سی می توان تمامی مدارات منطقی را پیاده نمود. همچنین این آی سی در PLC هایی مانند Sv-۴۰۰ استفاده می شود. ویژگی منفی این آی سی ها این است که قیمت آنها نسبت به میکروکنترلر ها بالاتر است.

زبان برنامه نویسی FPGA چیست؟ زبان های برنامه نویسی مانندC ، Basicو اسمبلی به صورت خط به خط دستورات را اجرا می کنند. این زبان ها قدرت کافی جهت پردازش موازی و حفظ حالات قبلی مدار را ندارند. در نتیجه زبان های توصیف سخت افزار HDL ساخته شد. مشهور ترین این زبان ها عبارتند از:

۱) System C : این زبان شباهت بسیار زیادی به زبان C دارد.
۲) System Verilog : این زبان نیز شباهت هایی به زبان C دارد. این زبان توسط شرکت Cadence گسترش پیدا کرد و کمی بعد از زبان VHDL استاندارد IEEE شد.
۳) VHDL : این زبان یکی از مشهور ترین زبان های توصیف سخت افزار می باشد که در وزارت دفاع آمریکا با همراری شرکت هایTexas Instrument ، IBMو Intermetrics ساخته شد و اولین زبان توصیف سخت افزار بسیار قوی می باشد.

امروزه مشهور ترین زبان های توصیف سخت افزار زبان VHDL و Verilog می باشد.

چگونه از نرم افزار ISE استفاده کنیم؟ راه اندازی سون سگمنت کاتد مشترک با زبان VHDL و FPGA های (Xilinx) قسمت بزرگی از کارکردن با FPGA مربوط به برنامه نویسی به زبان VHDL می باشد. با توجه به اینکه این زبان وابستگی به هیچ نرم افزاری ندارد، یعنی می توان برنامه ی VHDL نوشته شده برای یک FPGA Altera ر بر روی یک FPGA Xilinx ریخت با این فرق که باید مجددا توسط کامپایلر Xilinx سنتز شود. البته ممکن است بعضی از FPGA ها بلوک هایی داخل خودشان داشته باشند که دیگر FPGA ها ندارند) مانند پردازنده (DSPکه این یک حالت استثنا می باشد.

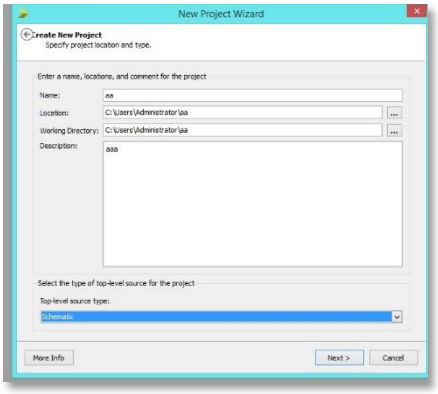
در این قسمت از نرم افزار ISE ورژن ۱۴ استفاده شده است که با Windows ۷ نیز کار می کند Crack .این نرم افزار را می توانید از قسمت دانلود دریافت نمایید.

نخست نرم افزار ISE را باز نمایید.

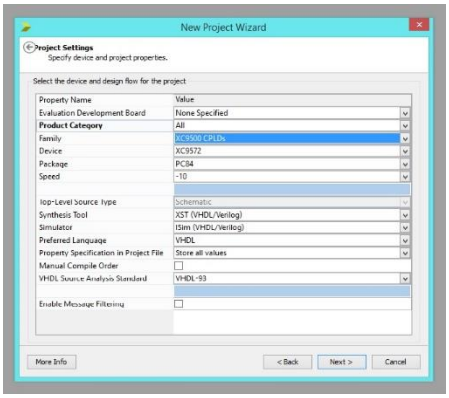
مراحل استفاده از برنامه

برای شروع برنامه از قسمت start/ ALL program/Xilinx Ise/ISE Design tools/ project

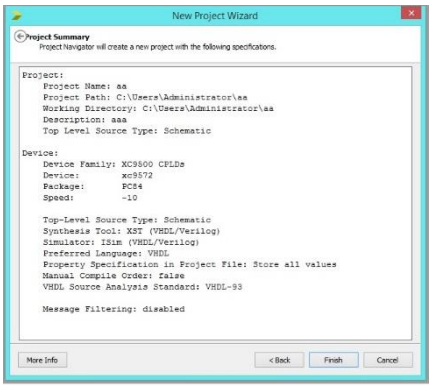
اجراکنید. از منوی file گزینه New Project را انتخاب کنید. پنجره new project را انتخاب کنید.

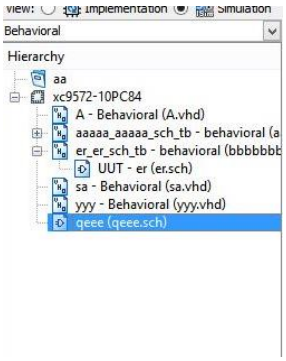


درقسمت new نام پروژه را تایپ می کنیم. مسیر مورد نظر را برای ذخیره پروژه جدید انتخاب کنید ودرقسمت Top level source type گزینه HDL را انتخاب می کنیم.وسپس next را انتخاب می کنیم تا وارد پنجره project settings شوید.

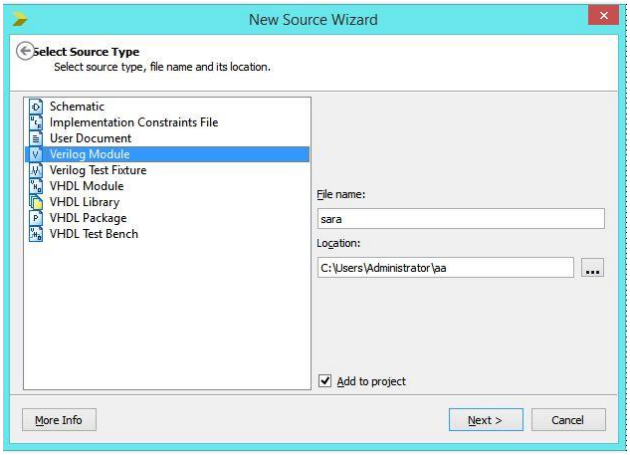


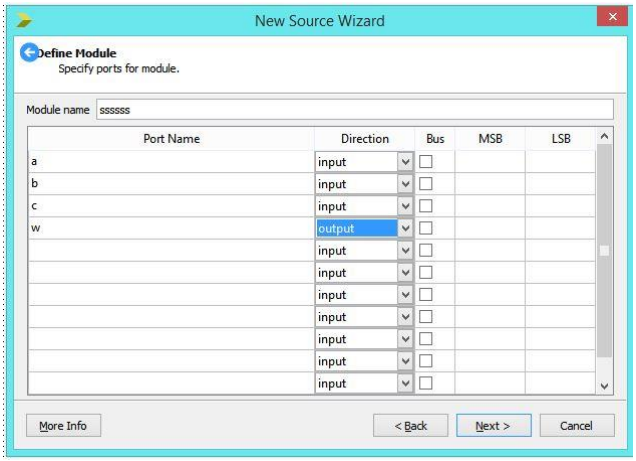
با کلیک کردن روی گزینه next خلاصه مشخصات پروژه نمایش داده می شود . با کلیک روی گزینه یfinsh پروژه ایجاد می شود ودرپنجره روبرونمایش داده خواهد شد.

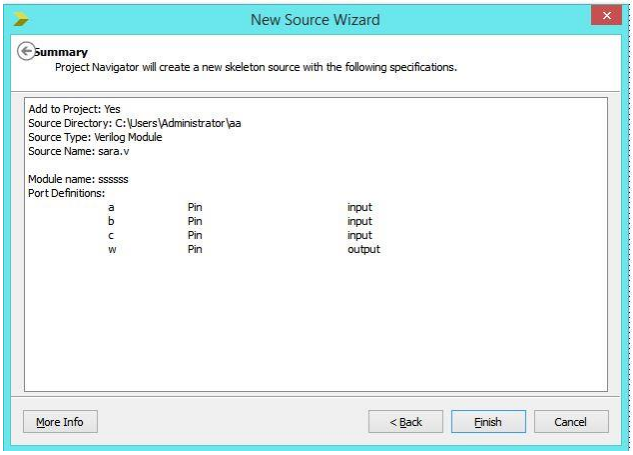




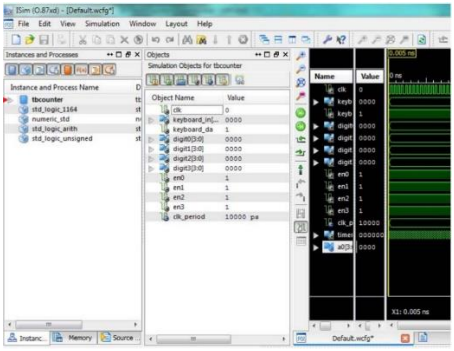
ابتدا در source کلیک راست کرده وگزینه Verilog Module را انتخاب کرده و next را انتخاب کرده و سپس پنجره بعد برای وارد کردن نام ورودی وخروجی های مدار است و گزینه finish را کلیک کرده.







برای نوشتن فایل test bench کلیه دستوراتی را که برای نوشتن برنامه اصلی استفاده می کنیم .



می توان به کاربرد علاوه براین سه دستورخاص نیز وجود دارند که فقط در محیط شبیه سازی وبرای نوشتن فایل test bench

۱) Wait:

این دستور به سه شکل زیر نوشته می شود

Wait for پرکاربردترین شکل این دستورات است.

۲) Report :

درقسمت پایین پنجره پنلی به نام console وجود دارد که گزارشی ازروند شبیه سازی درآن جا نمایش داده می شود با استفاده دستور Report می توان پیغام خودرانیز مشخص کنید.

- **Report** string [**severity** type] → مثال: **Report** "Test Completed" **severity note**

۳) Assert :

با استفاده ازاین دستور می توان وضعیت یک سیگنال را با وضعیت مطلوب مقایسه کرد ودر صورتی که مقدارسیگنال خلاف مقدار مورد نظر باشد درپنجره console نمایش داد.

- **Assert** condition [**report** string] [**severity** type] → مثال:

Assert s='1' **report** "Error, Expected 1 for S" **severity error**;

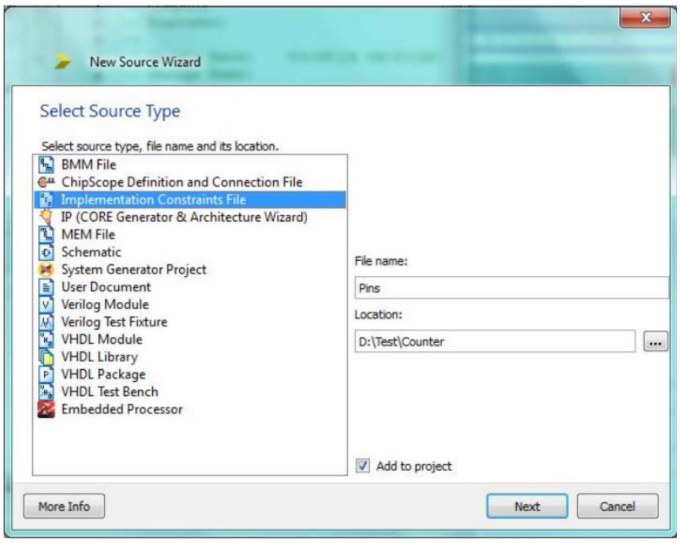
Test bench برای یک جمع کننده چهاربیتی:

```
-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
   -- Initialize input signals
   A <= "0000";
   B <= "0000";
   Ci <= '0';

   -- Loop over all values of A
   for I in 0 to 15 loop
      -- Loop over all values of B
      for J in 0 to 15 loop
         -- Wait for output to update
         wait for 10ns;
         -- Check value of Sum
         assert (S = A + B) report "Expected sum of " &
            integer'image(to_integer(unsigned((A + B)))) & " for A = " &
            integer'image(to_integer(unsigned((A)))) & " and B = " &
            integer'image(to_integer(unsigned((B)))) & ", but was " &
            integer'image(to_integer(unsigned((S)))) severity ERROR;
         -- Increment to the next value of B
         B <= B + "0001";
      end loop;
      -- Increment to next value of A
      A <= A + "0001";
   end loop;

   -- Echo to user that test is finished
   report "Test completed";
   wait;
END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

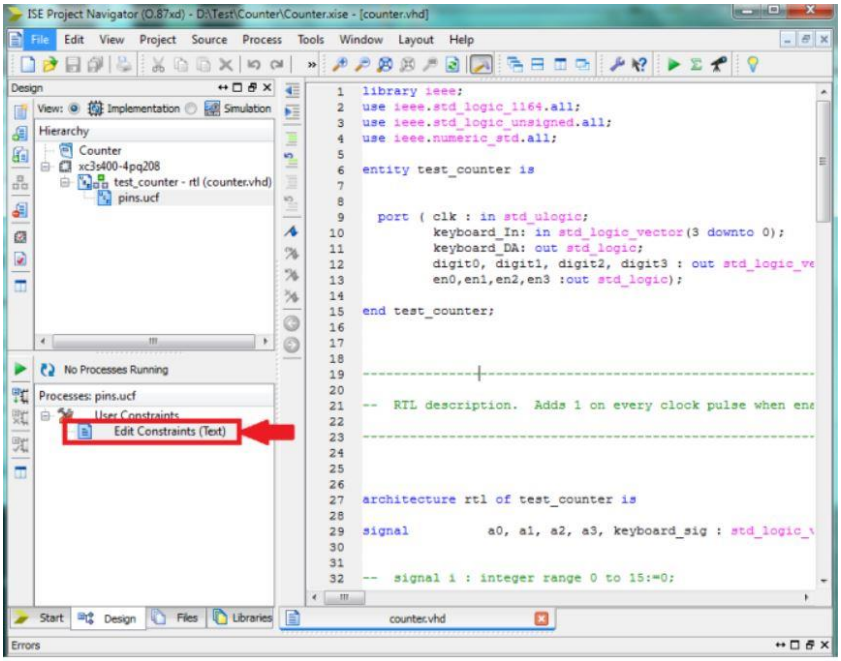
اختصاص شماره پایه های موردنظر تراشه به ورودی ها وخروجی ها

بعد از رفع خطاهای احتمالی و شبیه سازی برنامه، باید فایل با پسوند ucf. جهت اختصاص پایه های مورد نظر از FPGA به ورودی خروجی های برنامه را ایجاد کرد. ابتدا بر روی پروژه کلیک راست کـرده و گزینـه New Source را انتخاب می نماییم. سـپس طبـق شـکل ۴-۲۴ گزینـه Implementation Constrains File را انتخاب و در قسمت File name اسم فایل دلخواه را وارد می کنیم و بر روی Next کلیک و سـپس Finish را می زنیم. در این حالت فایل مورد نظر با پسوند ucf. ایجاد می گردد (مثلا Pins.ucf).

ایجاد فایل usf برای اختصاص پایه های Fpg



حال مطابق شکل زیر به قسمت User Constraints رفته و زیر شاخه Edit Constraints را کلیک می کنیم تا باز شود (شکل ۴-۲۵).



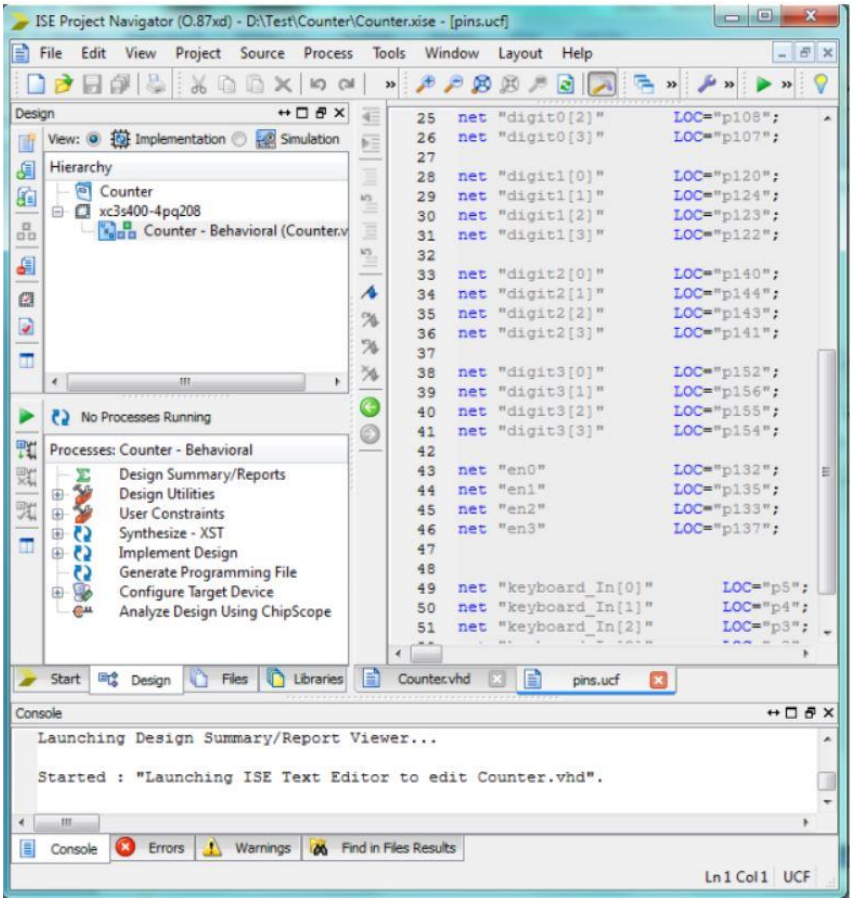
برای اختصاص هر پایه به یکی از پورت های ورودی یا خروجی به شکل زیر عمل می شود.

NET "نام پین ورودی یا خروجی" LOC="P شماره پایه مورد نظر از تراشه";

برای مثال اگر بخواهیم پایه ۱۰۶ از تراشه را به پورت a اختصاص دهیم باید بنویسیم:

NET "a" LOC="P106 ";

شکل ۴-۲۶ نمونه ای از یک فایل ucf. نوشته شده را نشان می دهد.



نکته:

ساختار When-else دستوری است که برای مقداردهی سینگل ها به صورت شرطی به کارمی رود به این معنا که سیگنال به یک مقدارتعلق می گیرد درصورتی که شرط آن برقرارباشد.

برای مثال برنامه مالتی پلکسر ۲*۱ بادستور when, else به این شکل می باشد.

```
۰<= t۰ when s=’۰’ else

I۱ when s=’۱’;
```

برای تک بیتی " می گذاریم وبرای بیش از تک بیتی " " می گذاریم.

درجمله اول ; احتیاج ندارد.

جمله دوم باشرط دوم شروع می شود.

حتما انتهای دستور یعنی درخط دوم باید ; قرار داده شود.

Mux ۴*۱ :

ابتدا ورودی ها وخروجی ها رامشخص می کنیم.

ورودی ها :s,i۳,i۲,i۱,i۰

خروجی ها :۰

برای s باید bus فعال شود و msb=۱,lsb=o باید باشد.

```
۰ <= i۰ when s ="۰۰" else

i۱ when s="۰۱" else

i۲ when s="۱۰" else

i۳ when s="۱۲" else;
```

ساختار if- then –else برای مقداردهی و یا انتخاب جملات می باشد واجرای آن براساس ارزیابی صحیح یا خطا از یک شرط می باشد.

اگر شرط داخل پرانتز if برقرار باشد بعد از اجرای جمله ی do something انتخاب می گردد واگر شرط برقرار نباشد بعد از else جمله something deferent انتخاب می گردد . درانتهای ساختار با end if; بسته می شود.

نکته: این شرط ها وجملات به همین شکل می توانند با دستور else if ادامه داشته باشد.

Mux ۴*۱ :

If- the – else

If (condition) then

Do something;

Else

Do something;

Else

Do something different

End if;

……

:Encoder

Yرا به عنوان خروجی

آن را به صورت زیر Msb,Hsb هردو رافعال کرده و قسمت bus را به عنوان ورودی درنظر می گیریم وسپس w تنظیم می کنیم:

۱) Msb=۱ Lsb=۰

۲) Msb=۳ Lsb=۰

```
Y<="۰" when w (۰)='۱' else
"۰۱" when w (۱)='۱' else
"۱۰" when w (۲)='۱' else
"۱۱" when w (۳)='۱' else
```

مدارمقایسه کننده:

دوعدد تک بیت A<B را باهم مقایسه می کند A<B A=B A>B

```
Process (A,B)is
Begin
If(a>b) then
G<='۱';L<='۰';E<='۰'
Else if(a<b)then
G<= '۰';L<='۱'; E<='۰'
Else if(a=b)then
G<= '۰';L<='۰'; E<='۱'
End process;
```

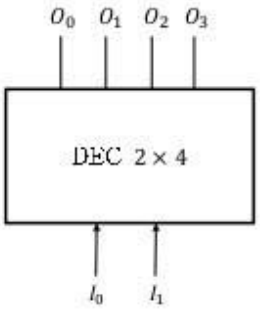
مقایسه کننده ۴ بیتی :

اگر A,B هرکدام چهاربیت باشند.مانند برنامه قبل عمل میکنیم.

:Decoder

دیکدر:

مداری ترکیبی دارای n ورودی و 2^n خروجی



I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

برنامه decoder :

W<= "۱۰۰"

W<= "۰۰۱" when y ="۱۱" else

W<= "۰۰۱۰" when y ="۱۰" else

W<= "۰۱۰۰" when y ="۰۱" else

W<= "۱۰۰۰" when y ="۰۰" else

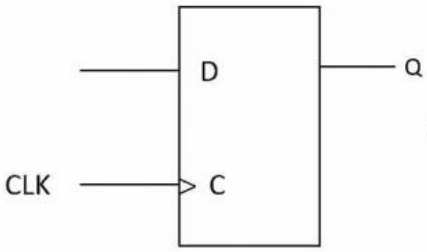
دراینجا باید فقط y را به عنوان ورودی و w را به عنوان خروجی تعریف کنیم و bus آن هم فعال شود نیازی به تعریف w_3 تا w_0 نمی باشد.

انواع مدارات ترتیبی:

- فلیپ فلاپ d
- فلیپ فلاپ t
- فلیپ فلاپ jk

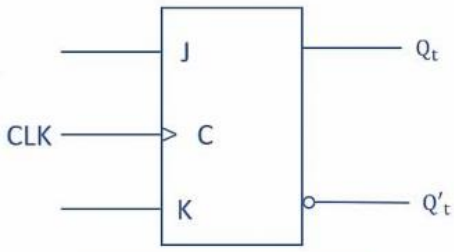
انواع فلیپ فلاپ ها:

فلیپ فلاپ D



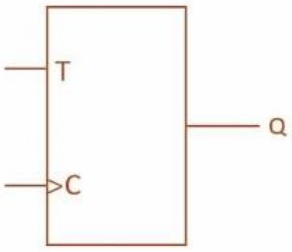
D	Q_{t+1}
0	0
1	1

فلیپ فلاپ JK



J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q'_t

فلیپ فلاپ T



T	Q_{t+1}
0	Q_t
1	Q'_t

مدارات منطقی سنگرون:

If (clk'event and clk='۱') then لبه بالارونده

If (clk'event and clk='۰') لبه پایین رونده

A: out std- Logic: = '۰'

Process (clk, t) is

Begin

If (clk’event ‘and clk='۱') then

If (t='۱۰') then

q<=not (q);

End if;

End if;

End process;

بعدخط if (clk 'event') نوشته میشود ویک end if اضافه میشود.

اگر reset برابر صفرباشد فلیپ فلاپ نوع D کارمی کند به هرمقداری دروردی D باشد به q منتقل می شود ولی زمان rest برابر ۱ باشد خروجی بدون درنظرپرفتن مقداروردی D صفرمی شود.

T	q
۰	Q’(T)
۱	Q (T)

```
Q: input std-logic: =’۰’

Process (clk, T) is

Begin

If (clk’event

 and clk=’۱’) then

If (t=’۰’) then

q<=not (q);

Else

q<=q;

End if;

End if;

End process;
```

نکته :

Entetie: مقداردهی باید شود.

’۰’= : نوشته میشود.

فلیپ فلاپ j-k:

```
q: out std- logic: = '۰'
Process (clk, j, k) is
Begin
If (clk’event ‘and clk='۱') then
If (j='۰' and clk='۱') then q<=q;
Elsif (j='۰' and k='۱') then q<='۰';
Elsif (j='۱' and k='۱') then q<='۱';
Elsif (j='۱' and k='۱') then q<=not (q)';
End if;
End if;
End process;
```

jk با خط rest

```
If (rst='۰') then q<='۱';
Elsif (rst='۱') then q<='۰';
```

عامل لبه باشد

```
If(clk=’۱’)then
Q<=d;
End if;
End process;
Process (clk, d) is
```

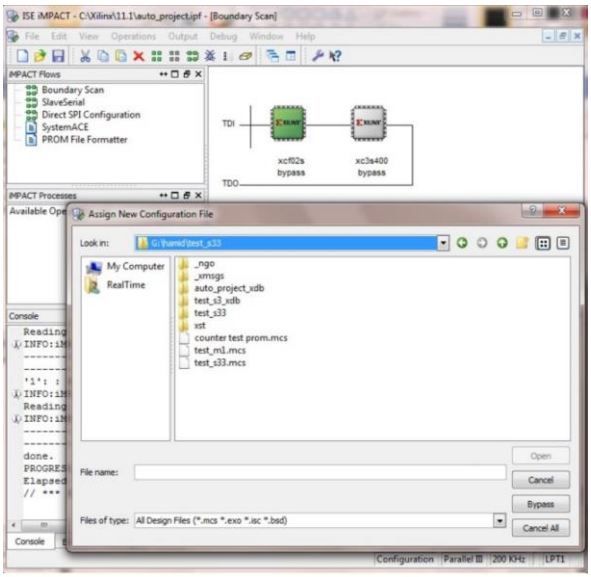
هربرنامه که میخواهیم فایل usf برایش میسازیم برای ساختن usf روی گزینه source بعدگزینه new source را گزینه innmeplaton انتخاب می کنیم یک اسم برایش انتخاب می کنیم بعد next وfinsh را انتخاب می کنیم.

فایل usf کلیک وبعد قسمت process گزینه .....user بازمی کنیم گزینه assign packaging پنجره پایه ic رامشخص می کنیم.

فایل save وok را می زنیم. Source فایل هایلات می شود. Process گزینه Implanted design گزینه grantee programing file وبعد گزینه configure device را دابل کلیک کرده می کنیم.

درپنجره finsh را انتخاب می کنیم (شمای ic) پنجره ایی که بازشده نشان برنامه ایی کابلها راشناسایی شده و اتوماتیک کابلها وصل شده. فایل با پسوند Jed انتخاب می کنیم و open را می زنیم.کنار ic یک خط آمده روش یک بار کلیک راست کرده که ic سبزشود وهنگامی که سبزشد اولین آیکون program بعد پنجره ok را می زنیم. اگه موفقیت آمیزباشد آبی رنگ succefull را می دهد ic به برنامه داده شده.

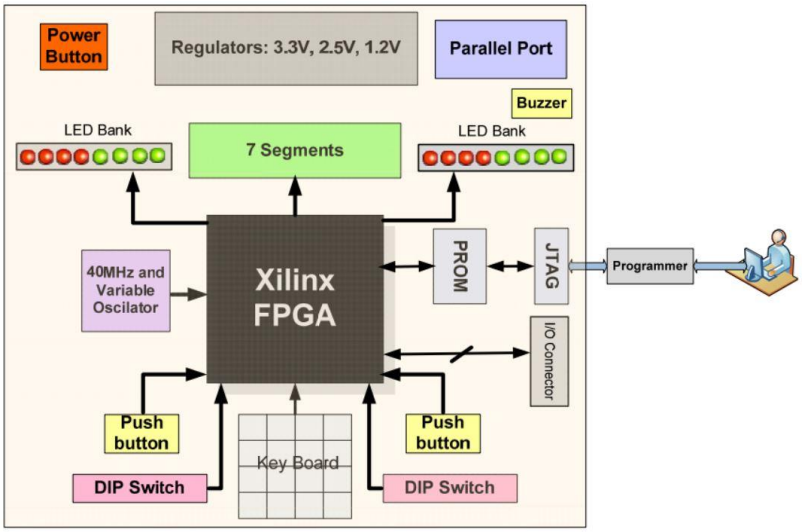
برنامه ریزی FPGA یا حافظه :



پس از انتساب فایل مورد نظر به FPGA یا حافظه، روی تراشه مربوطه کلیک راست کرده و گزینه Program را انتخاب می کنیم. در پایان پیغامی مبنی بر program succeeded ظاهر می شود. تفاوتی در اولویت انتخاب تراشه FPGA یا حافظه جهت برنامه ریزی وجود ندارد. اگر فقط تراشه FPGA را برنامه ریزی کرده باشیم، با قطع تغذیه برد، برنامه ی پروگرام شده بر روی تراشه پاک می شود. در صورتی که حافظه را برنامه ریزی کرده باشیم با قطع تغذیه برنامه پروگرام شده بر روی آن از بین نمی رود.

معرفی سیستم آزمایشگاه:

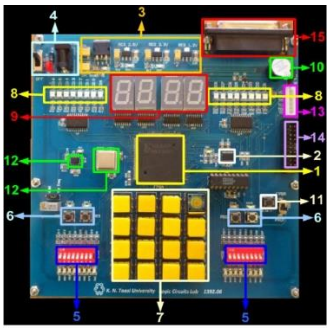
بورد FPGA موجد در آزمایشگاه بر اساس تراشه Spartan3-XC3s400 شرکت Xilinx طراحی شـده اسـت. شکل زیر بلوک دیاگرام کلی بورد طراحی شده را نشان می دهد. نحوه ارتباط بخش های مختلف با FPGA در شکل زیر دیده می شود.



امکانات بورد عبارتند از:

- تراشه FPGA از خانواده SPARTAN3 و نوع XC3S400-PQG208
- حافظه XCF02S به ظرفیت 2MBit
- دو عدد دیپ سوییچ هشت تایی جهت اعمال داده های ورودی به تراشه
- چهار عدد کلید فشاری جهت اعمال ورودی های پالسی به مدارهای طراحی شده
- کیبورد ۱۶ تایی برای اعمال اعداد دهدهی ۰ تا ۱۵به ورودی های مدار
- شانزده عدد دیود نوری مجهز به بافر جهت نمایش داده های خروجی

- ۴ عدد سون سگمنت جهت نمایش داده های خروجی
- یک زنگ برای اعمال خروجی های صوتی
- کلید ریست برای ریست کردن FPGA
- دو نوع مدار کلاک با فرکانس های مختلف: یک کلاک ۴۰ مگاهرتز و یک کلاک متغیر برای فرکانس های پایین
- امکان برنامه ریزی تراشه FPGA با استفاده از پروگرمر USB و یا پارالل



۱- تراشه FPGA: تراشه FPGA هسته اصلی بورد را تشکیل می دهد. این تراشه از سلول های منطقی قابل برنامه ریزی ساخته شده است که ارتباط بین این سلول ها نیز قابل برنامه ریزی است. تراشه های FPGA بسیار سریع هستند و برای پیاده سازی مدارهای دیجیتال استفاده می شوند. پس از برنامه ریزی FPGA می توان مدار طراحی شده را تغییر داد و FPGA را مجددا برنامه ریزی نمود. سرعت بالای این تراشه ها آن ها را مساعد کارهای پردازشی سنگین مثل پردازش تصویر، صدا و ... می کند و سرعت این پردازش نسبت به سیستم های دیگر خیلی بالاتر است. سرعت بالای این تراشه ها از قابلیت پردازش موازی و انجام چندین عملیات به صورت هم زمان ناشی می شود. میکروکنترلرها، تراشه های DSP و میکروپروسسورها دستورات را به ترتیب و پشت سر هم اجرا می کنند و قابلیت پردازش موازی در آن ها وجود ندارد.

در این بورد از تراشه XC3S400-4PQ208I که از خانواده Spartan3 و از تولیدات شرکت Xilinx است استفاده شده است. این تراشه دارای ۴۰۰۰۰۰ هزار گیت منطقی و حداکثر ۲۸۴ پایه I/O است. کلیه کارهای پردازش و برقراری ارتباط با قطعات جانبی و کاربر توسط این تراشه انجام می شود.

۲- حافظه: شرکت Xilinx حافظه هایی با ظرفیت ۱ تا ۳۲ مگابایت را که از نوع Platform Flash PROM هستند عرضه کرده است. این حافظه ها که در دو سری XCFxxS (تغذیه ۳,۳ ولت) و XCFxxP (تغذیه ۱,۸ ولت) معرفی شده اند برای برنامه ریزی FPGA در مد Master/Slave Serial به کار می روند. در این بورد از حافظه XCF02S با ظرفیت 2MBit استفاده شده است.

۳- رگولاتورها: تراشه XC3S400 و سایر آی سی های روی بورد به چهار ولتاژ مختلف برای کار نیاز دارند: ۱,۲ ولت، ۲,۵ ولت، ۳,۳ ولت و ۵ ولت. ولتاژ ۵ ولت توسط آداپتوری که به تغذیه بورد متصل می شود تامین می گردد. سایر ولتاژ ها با استفاده از سه رگولاتور یا تنظیم کننده ولتاژ تامین می شوند. کنار تغذیه ورودی و هر یک از رگولاتورها یک دیود نوری تعبیه شده است که وجود یا عدم وجود ولتاژ مربوطه را نشان می دهد. یک دیود نیز برای حفاظت از تغذیه معکوس نیز روی بورد قرار داده شده است.

۴- تغذیه: تغذیه بورد ولتاژ ۵ ولت است که با اتصال یک آداپتور ۵ ولت به کانکتور تغذیه، وارد بورد می شود. با اتصال کلید کنار کانکتور، بورد روشن می شود.

۵- دیپ سوییچ: این بورد شامل دو عدد دیپ سوییچ هشت تایی است که برای گرفتن ورودی از کاربر استفاده می شوند. یک دیپ سوییچ مجموعه ای از چند کلید جداگانه است که در بسته ای مشابه با یک IC قرار دارند. برای اطمینان از اتصال کلیدهای دیپ سوییچ، هر کلید به یک دیود نوری مجهز شده است که به هنگام اتصال کلید روشن می شود.

۶- کلیدهای فشاری: وجود چهار عدد کلید فشاری امکان اعمال ورودی های پالسی به مدار را فراهم ساخته است. برای هر کلید فشاری یک یک دیود نوری قرار داده شده است که به هنگام فشار دادن کلید روشن می شود.

۷- صفحه کلید: این صفحه کلید از ۱۶ عدد کلید فشاری تشکیل شده است. با فشردن هر کلید یک سطر و یک ستون از صفحه کلید فعال می شود. خروجی سطرها و ستون های صفحه کلید به آی سی 74922 داده می شوند و آی سی کد باینری چهار بیتی مربوطه را استخراج می کند و به عنوان ورودی به FPGA می دهد.

۸- دیودهای نوری: دیودهای نوری در چهار رنگ مختلف و برای نمایش خروجی مورد نظر کاربر روی بورد قرار داده شده اند. این بورد مجهز به ۱۶ دیود نوری است و بنابراین قابلیت نمایش اعداد تا ۱۶ بیت را داراست.

۹- سون سگمنت: چهار عدد سون سگمنت برای نمایش اعداد به صورت دهدهی روی بورد قرار داده شده است. هر سون سگمنت مجهز به یک آی سی مبدل باینری به سون سگمنت است. بنابراین کاربر می تواند هر خروجی ۴ بیتی را مستقیما روی یک سون سگمنت نشان دهد.

۱۰- زنگ: اگر در مدارهای طراحی شده نیاز به یک خروجی صوتی باشد از این زنگ استفاده می شود.

۱۱- کلید ریست: از این دکمه فشاری برای ریست کردن FPGA استفاده می شود.

۱۲- اسیلاتور: یک اسیلاتور با فرکانس 40MHz روی برد قرار گرفته که به یکی از پایه های FPGA متصل می باشد و می تواند به عنوان کلاک اصلی برد مورد استفاده قرار گیرد. اسیلاتور دیگری روی بورد تعبیه شده است که فرکانس آن قابل تنظیم است و می تواند از ۰/۰۰۱ هرتز تا ۳۰۰ کیلوهرتز باشد. این اسیلاتور در مواردی استفاده می شود که به فرکانس های پایین نیاز است و برای راحتی کاربر قرار داده شده است. فرکانس این کلاک با یک مقاومت متغیر تنظیم می شود. با جابجا کردن یک Jumper بین دو حالت ممکن خود، می توان انتخاب کرد کدام یک از کلاک ها به مدار اعمال شوند.

۱۳- پورت JTAG: در صورتی که بخواهیم تراشه FPGA را با استفاده از پروگرمر USB برنامه ریزی کنیم از باید آن را پورت JTAG روی بورد متصل کنیم. FPGA از طریق پورت JTAG با کامپیوتر ارتباط برقرار کرده و تبادل داده می کند. JTAG در واقع استانداردی است که برای برقراری ارتباط بین یک دستگاه و اجزای جانبی استفاده می شود. وظیفه اصلی این پورت برنامه ریزی FPGA و یا خواندن برنامه ذخیره شده

۱۴- کانکتور I/O: برای ارتباط با خارج بورد یک کانکتور ورودی-خروجی ۲۰ پایه در نظر گرفته شده که به پایه های کاربری باقیمانده FPGA متصل شده است و می توان از آن برای ارتباط با اجزای جانبی مانند LCD و یا بورد های دیگر استفاده کرد.

روی آن است. به هنگام برنامه ریزی FPGA، داده از طریق کامپیوتر به یک پروگرمر داده می شود و سپس از طریق پورت JTAG به حافظه یا FPGA منتقل می شود.

۱۴- کانکتور I/O: برای ارتباط با خارج بورد یک کانکتور ورودی-خروجی ۲۰ پایه در نظر گرفته شده که به پایه های کاربری باقیمانده FPGA متصل شده است و می توان از آن برای ارتباط با اجزای جانبی مانند LCD و یا بورد های دیگر استفاده کرد.

۱۵- پورت موازی: برای برنامه ریزی تراشه FPGA از طریق پورت موازی کامپیوتر، باید یک سر کابل پارالل را به کامپیوتر و سر دیگر آن را به کانکتور نشان داده شده روی بورد متصل کرد.

معرفی سیستم آزمایشگاه:

FPGA از طریق پایه های I/O با امکانات جانبی بورد ارتباط برقرار مـی کنـد. ورودی هـا و خروجـی هـای روی بورد مطابق جدول زیر به پایه های I/O تراشه FPGA متصل شده اند.

INPUT PINS			
DIP Switch 2		DIP Switch 1	
D0 8	D4 4	D0 43	D4 37
D1 9	D5 5	D1 42	D5 36
D2 10	D6 12	D2 40	D6 35
D3 11	D7 15	D3 39	D7 34
Key board			
Key – D3	Key – D2	Key – D1	Key – D0
52	51	50	48
Clocks			
183			

OUTPUT PINS							
LEDs							
LED 1	LED 2	LED 3	LED 4	LED 5	LED 6	LED 7	LED 8
93	94	95	96	97	100	101	102
LED 9	LED 10	LED 11	LED 12	LED 13	LED 14	LED 15	LED 16
161	162	165	166	167	168	169	171
7SEG 4		7SEG 3		7SEG 2		7SEG1	
D0 152		D0 140		D0 120		D0 106	
D1 156		D1 144		D1 124		D1 109	
D2 155		D2 143		D2 123		D2 108	
D3 154		D3 141		D3 122		D3 107	
7SEG 4 _En		7SEG 3 _En		7SEG 2 _En		7SEG 1 _En	
137		133		132		132	
BUZZER							
111							