



آزمایشگاه معماری کامپیوتر

COMPUTER ARCHITECTURE LAB

به همراه زبان توصیف سخت افزاری **VHDL**

مهندسی کامپیوتر

مدرس : دلدار

ویرایش اول بهار ۱۳۹۴

فهرست

.....	مقدمه مدار منطقی
.....	آزمایش شماره ۱
.....	آزمایش شماره ۲
.....	آزمایش شماره ۳
.....	آزمایش شماره ۴
.....	آزمایش شماره ۵
.....	آزمایش شماره ۶
.....	آزمایش شماره ۷
.....	آزمایش شماره ۸
.....	آزمایش شماره ۹
.....	آزمایش شماره ۱۰
.....	آزمایش شماره ۱۱
.....	آزمایش شماره ۱۲
.....	آزمایش شماره ۱۳
.....	آزمایش شماره ۱۴
.....	مطلوب تكميلي

■ آلبرت انیشتین : به خاطر داشته باشید هر چه در مدارس خود یاد می گیرید نتیجه کار نسل های بیشماری است که در اثر کوشش آرزومندانه همه مردم جهان به ثمر رسیده است و سپرده ای است در دست شما ، از آن استفاده ببرید و به آن بیفزایید تا روزی که آنرا با کمال امانت و وفاداری به فرزندانتان بسپارید .

■ خواننده گرامی این جزو در دست ویرایش می باشد در صورت مشاهده هرگونه ایجاد یا اشکال تایپی و یا هرگونه انتقاد و پیشنهاد با ایمیل اینجانب تماس بگیرید deldar_edu@yahoo.com

پیش از انجام آزمایش لازم است که با مفاهیم زیر به خوبی آشنا باشید :

۱ - مبنای ها

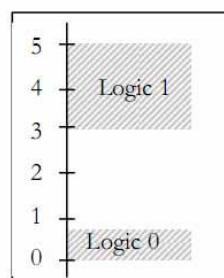
دده	دو دویس	هشتگانی	شانزده تالی
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E
31	11111	37	1F

۲ - محاسبات دودویی

A+B	نقلی	جمع
0+0	0	0
0+1	0	1
1+0	0	1
1+1	1	0

۳ - کدهای دیجیتال

۴ - سطوح دیجیتال صفر و یک منطقی



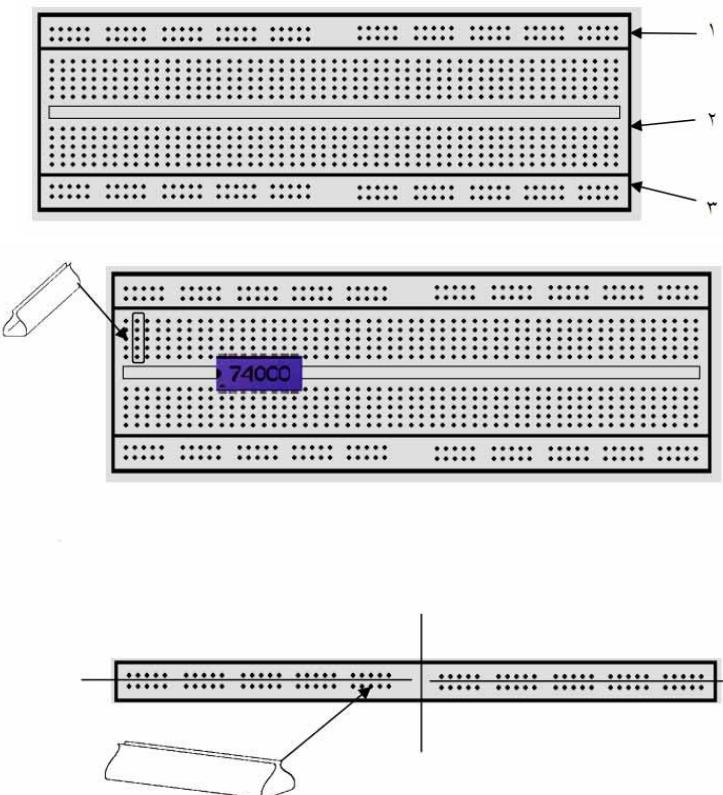
۵ - شکل موج های دیجیتال

موج مربعی و پالس

تفاوت سطح و لبه

۶ - آشنایی با قطعات الکترونیک

مقاومت ، خازن ، سلف ، دیود ، ترانزیستور ، LED ، 7-SEGMENT ، برد بورد



۷ - آشنایی با گیت های منطقی

نمودار استفاده در جریان	جدول صحت	عمل منطقی															
$Y = A$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>Y</td></tr> <tr> <td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td></tr> </table>	A	Y	0	0	1	1	$A \rightarrow Y$									
A	Y																
0	0																
1	1																
$Y = \bar{A}$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>Y</td></tr> <tr> <td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td></tr> </table>	A	Y	0	1	1	0	$\bar{A} \rightarrow Y$									
A	Y																
0	1																
1	0																
$Y = A \cdot B$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>B</td><td>Y</td></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	$A \cdot B \rightarrow Y$
A	B	Y															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
$Y = \bar{A} \cdot \bar{B}$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>B</td><td>Y</td></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	$\bar{A} \cdot \bar{B} \rightarrow Y$
A	B	Y															
0	0	1															
0	1	1															
1	0	1															
1	1	0															
$Y = A + B$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>B</td><td>Y</td></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	$A + B \rightarrow Y$
A	B	Y															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
$Y = \bar{A} + B$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>B</td><td>Y</td></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	$\bar{A} + B \rightarrow Y$
A	B	Y															
0	0	1															
0	1	0															
1	0	0															
1	1	0															
$Y = A \oplus B$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>B</td><td>Y</td></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>1</td></tr> <tr> <td>1</td><td>1</td><td>0</td></tr> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	$A \oplus B \rightarrow Y$
A	B	Y															
0	0	0															
0	1	1															
1	0	1															
1	1	0															
$Y = \bar{A} \oplus B$	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>A</td><td>B</td><td>Y</td></tr> <tr> <td>0</td><td>0</td><td>1</td></tr> <tr> <td>0</td><td>1</td><td>0</td></tr> <tr> <td>1</td><td>0</td><td>0</td></tr> <tr> <td>1</td><td>1</td><td>1</td></tr> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	$\bar{A} \oplus B \rightarrow Y$
A	B	Y															
0	0	1															
0	1	0															
1	0	0															
1	1	1															

۸ - تعریف آی سی



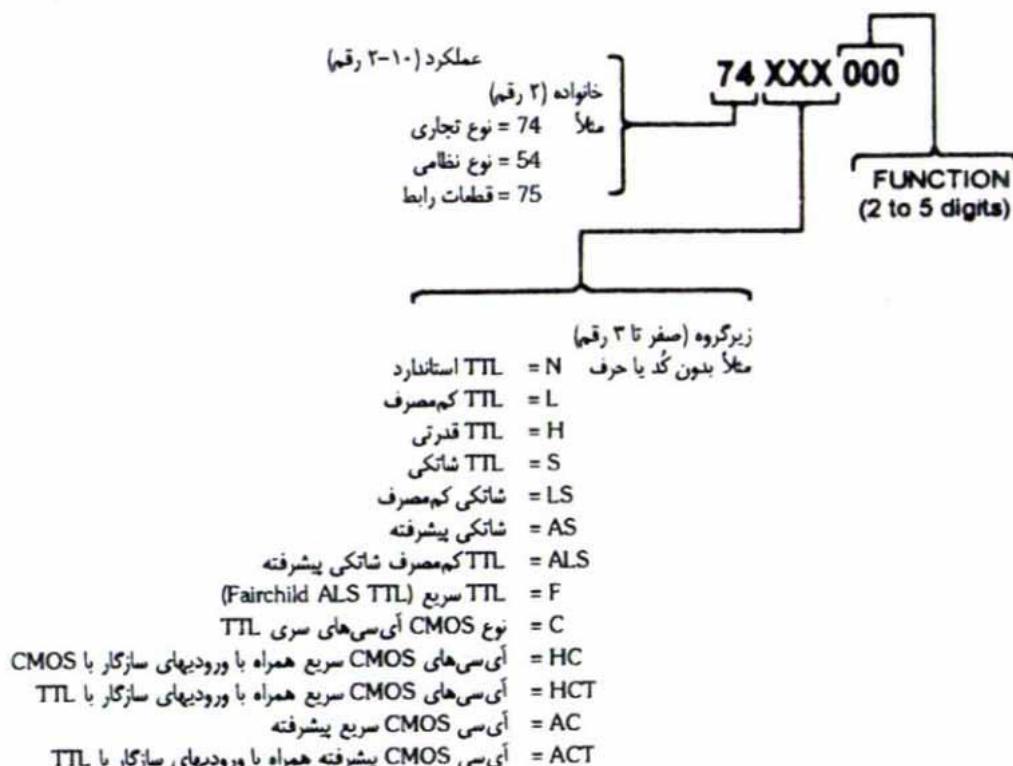
۹ - انواع آی سی های دیجیتال از نظر تراکم

SSI	Small Scale Integration
MSI	Medium Scale Integration
LSI	Large Scale Integration
VLSI	Very Large Scale Integration
SLSI	Super Large Scale Integration
ULSI	Ultra Large Scale Integration

۱۰ - منابع تغذیه و ولتاژ تغذیه آی سی ها

ACTIVE LOW & ACTIVE HIGH - ۱۱

۱۲ - شناسایی پایه های آی سی های دیجیتال



دانشجوی گرامی جدول زیر را در هر جلسه به همراه داشته باشید این جدول ارزیابی شما در تمرینات می باشد . 

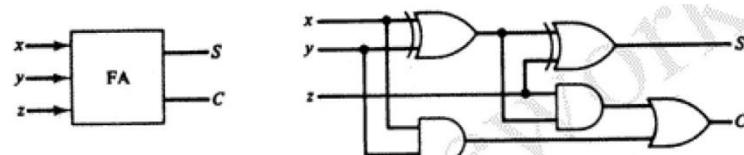
ردیف	موضوع تمرین	شماره صفحه	ملاحظه
۱			
۲			
۳			
۴			
۵			
۶			
۷			
۸			
۹			
۱۰			
۱۱			
۱۲			
۱۳			
۱۴			
۱۵			
۱۶			
۱۷			
۱۸			
۱۹			
۲۰			
۲۱			
۲۲			
۲۳			
۲۴			
۲۵			
۲۶			
۲۷			
۲۸			
۲۹			
۳۰			
۳۱			
۳۲			
۳۳			
۳۴			

آزمایش شماره ۱ : تست چند گیت ساده

ابتدا با کمک کدهای انتهای جزوه سعی کنید کدهای مربوط به گیت های AND و OR و XOR را نوشته و آنها را تست کنید .

تمرین : سعی کنید کدهای مربوط به گیتهای NOT و NAND و NOR و XNOR و BUFFER را نوشته و آنها را تست نمایید . 

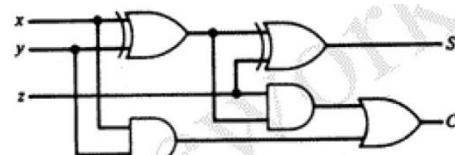
آزمایش شماره ۲ : مدار تمام جمع کننده FULL ADDER



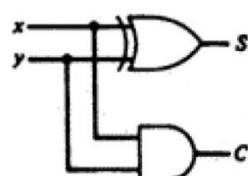
(ب) بلاک دیاگرام

مدار تمام جمع کننده

(الف) دیاگرام منطقی



تمرین ۱ : مدار نیم جمع کننده



(ب) دیاگرام منطقی

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

(الف) جدول درستی

نیم جمع کننده

تمرین ۲ : مدار تابع رسم شده



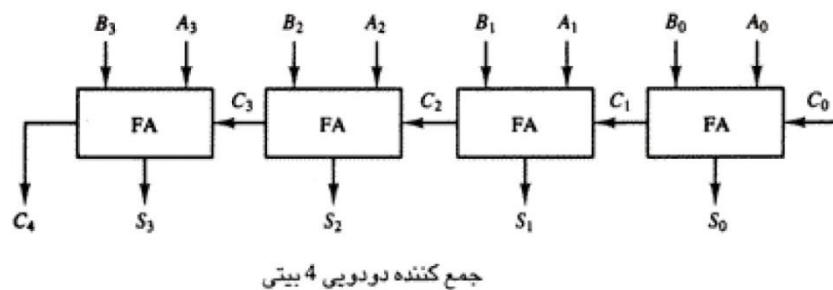
(ب) دیاگرام منطقی

z	y	x	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

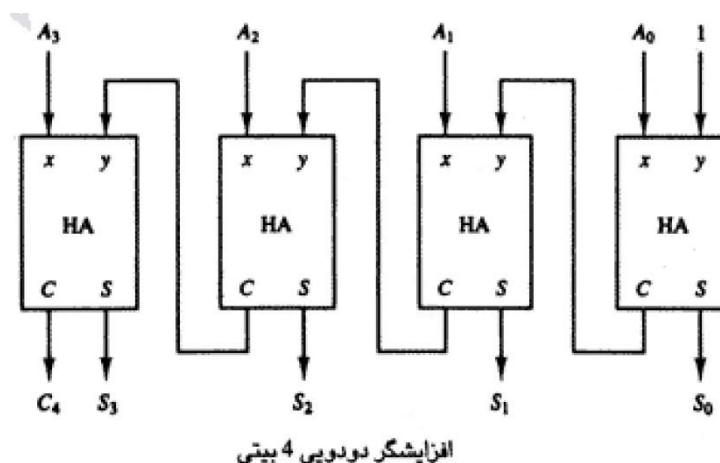
(الف) جدول درستی

$$F = x + y'z + y'z'$$

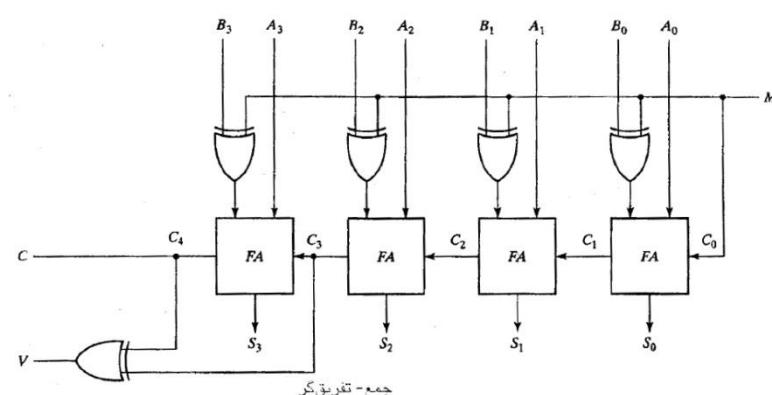
آزمایش شماره ۳ : مدار جمع کننده چهاربیتی



تمرین ۱ : مدار افزایشگر دودویی با کمک نیم جمع کننده ها

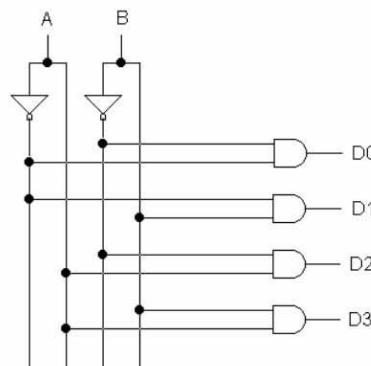


تمرین ۲ : مدار جمع کننده - تفریق گر به همراه پرچم های CF, OF, SF, ZF



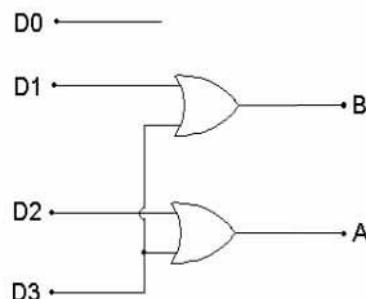
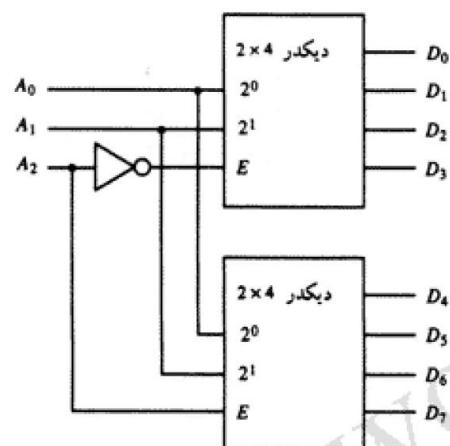
آزمایش شماره ۴ : دیکدر و انکدر

❖ آزمایش های زیر را انجام و نتایج را ثبت کنید :

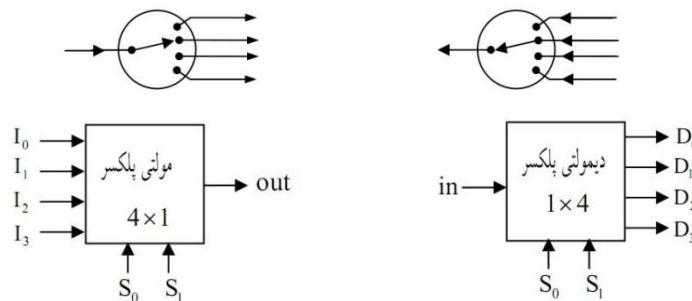


A	B	D0	D1	D2	D3
0	0				
0	1				
1	0				
1	1				

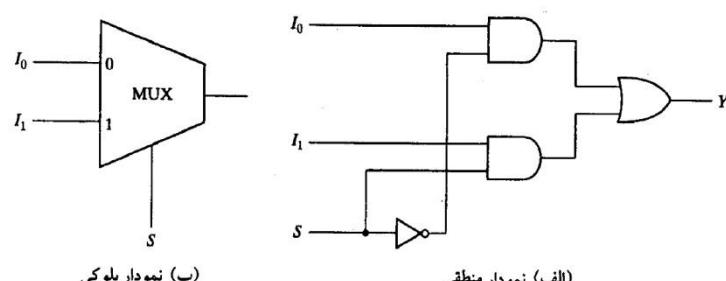
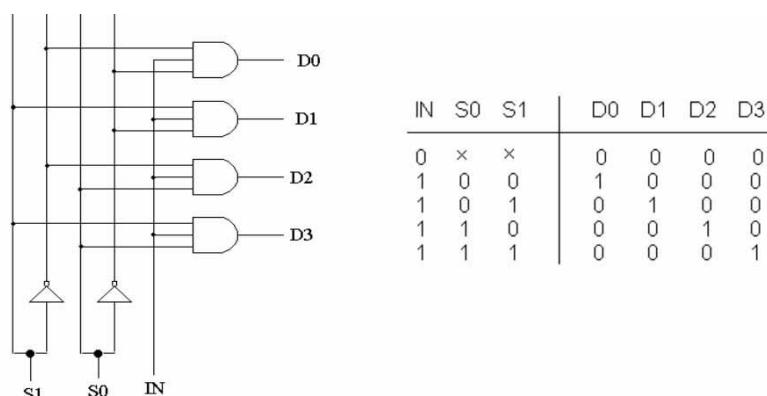
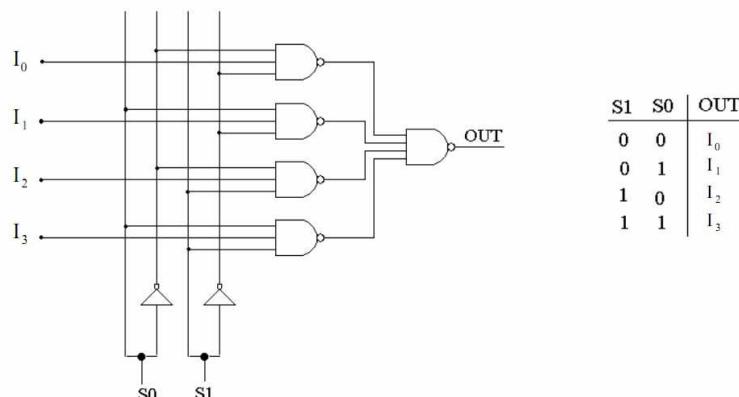
D0	D1	D2	D3	A	B
1	0	0	0		
0	1	0	0		
0	0	1	0		
0	0	0	1		

❖ به کمک دو دیکدر 2×4 یک دیکدر 3×8 طراحی کنیدیک دیکدر 8×3 (یا 3×8) ساخته شده با دو دیکدر 4تمرين 1 : ساخت دیکدر ۵ به ۳۲ با دیکدر های ۳ به ۸ و ورودی فعالساز و یک دیکدر ۲ به ۴ (دیکدر از عبارت 24_{dec} استفاده کنید .)

آزمایش شماره ۵ : مولتی پلکسر و دی مولتی پلکسر



❖ آزمایش های زیر را انجام و نتایج را ثبت کنید :



تمرین ۱ : به کمک ۴ مولتی پلکسر 2×1 یک مولتی پلکسر 2×1 چهارتایی طراحی و تست کنید.



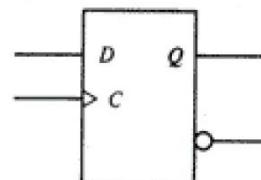
تمرین ۲ :

آزمایش شماره ۶ : آشنایی با فلیپ فلاپها

تمرین ۱ : فلیپ فلاپهای زیر را در نرم افزار تست نمایید و نتایج را ثبت کنید

D	$Q(t+1)$
0	0 پاک شدن، 0
1	1 نشانده شدن، 1

(ب) جدول مشخصه

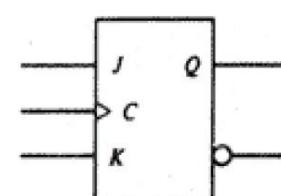


(الف) سیبل گرافیکی

فلیپ فلاپ D

J	K	$Q(t+1)$
0	0	$Q(t)$ پلاتفیر
0	1	0 پاک شدن، 0
1	0	1 نشانه شدن، 1
1	1	$Q'(t)$ منم

(ب) جدول مشخصه

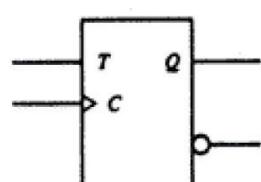


(الف) سیبل گرافیکی

فلیپ فلاپ JK

T	$Q(t+1)$
0	$Q(t)$ پلاتفیر
1	$Q'(t)$ منم

(ب) جدول مشخصه



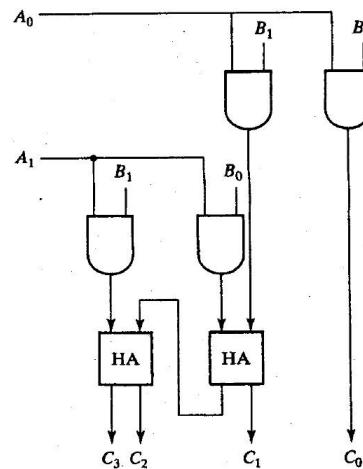
(الف) سیبل گرافیکی

فلیپ فلاپ T

تمرین ۲ : مدار ضرب کننده دو بیتی را تست و نتایج را ثبت کنید.

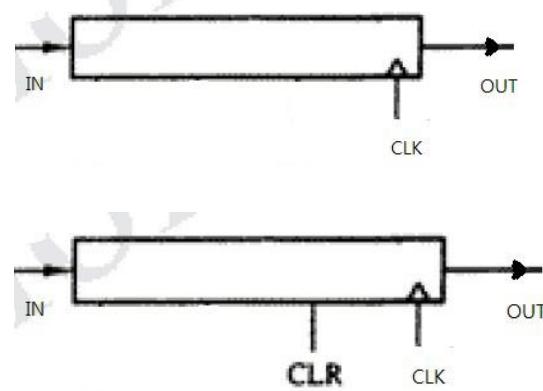
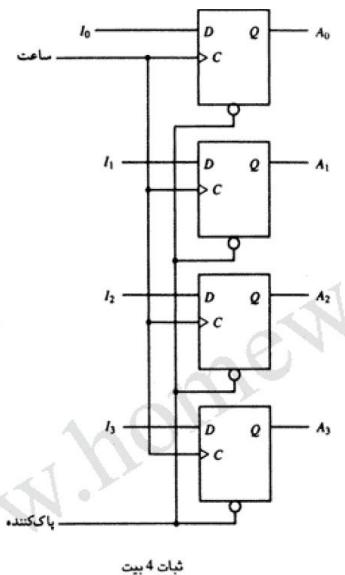
$$\begin{array}{r}
 \text{مضروب} & B_1 & B_0 \\
 \text{مضروب فیه} & A_1 & A_0 \\
 \hline
 & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 \\
 \hline
 C_3 & C_2 & C_1 & C_0
 \end{array}$$

ضرب دو دو بیتی ۲ بیت در ۲ بیت

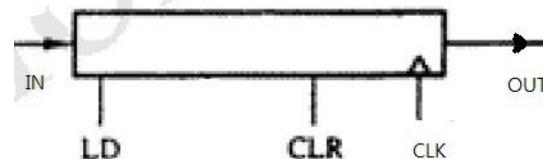
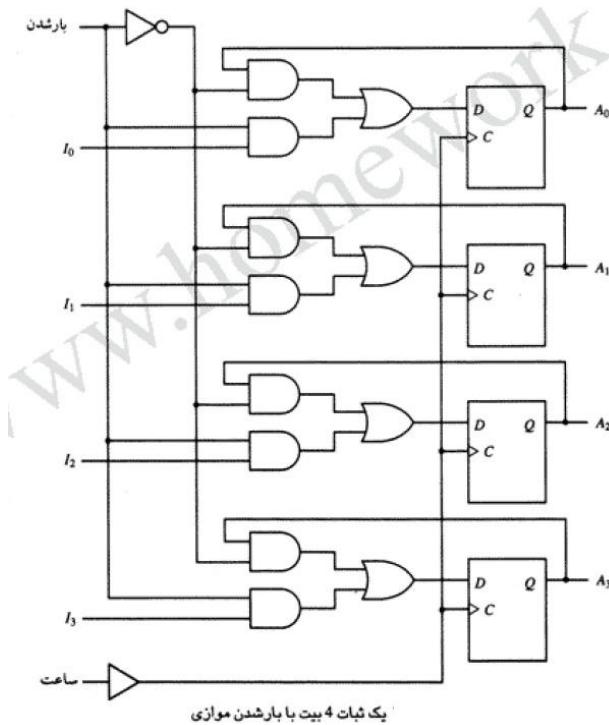


آزمایش شماره ۷ : ثبات register

❖ ثبات ۴ بیتی زیر را به کمک فلیپ فلاپ D طراحی و تست کنید.



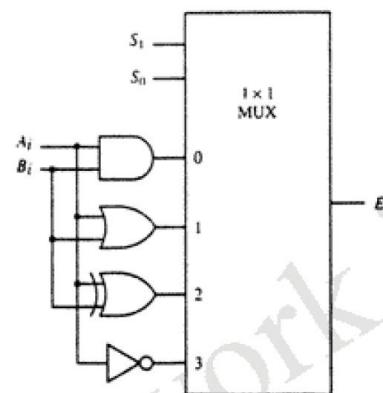
تمرین ۱ : ثبات ۴ بیتی زیر را تست کنید.



آزمایش شماره ۸ : مدار منطقی

S_1	S_0	خروجی	عمل
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \bar{A}$	منverse

(ب) جدول تابع



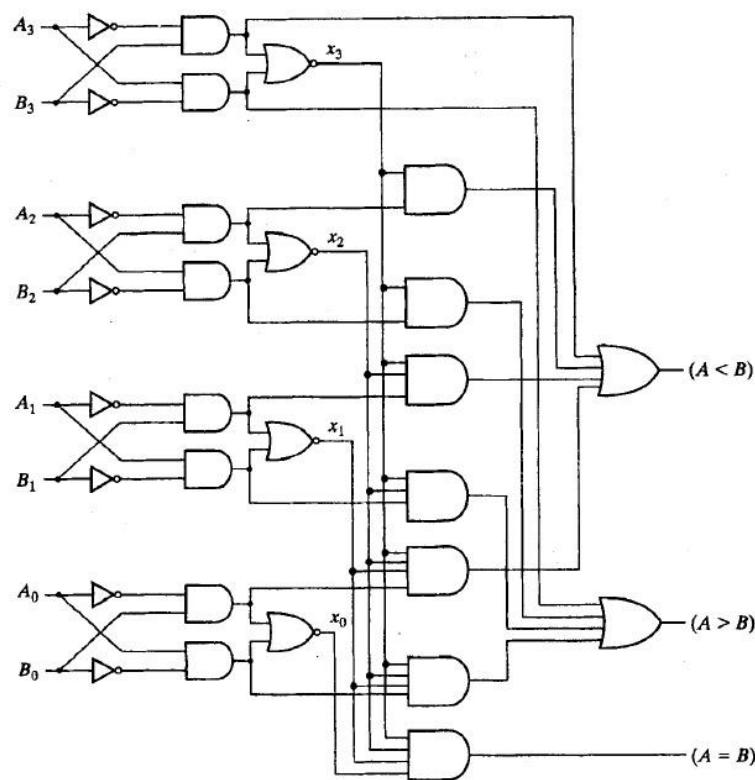
(الف) نمودار منطقی

یک طبقه از مدار منطقی

تمرین ۱ : مدار منطقی طراحی کنید که جدول زیر را تحقق بخشد .

S2	S1	S0	out
0	0	0	AND
0	0	1	OR
0	1	0	XOR
0	1	1	XNOR
1	0	0	NAND
1	0	1	NOR
1	1	0	NOT A
1	1	1	BUFFER

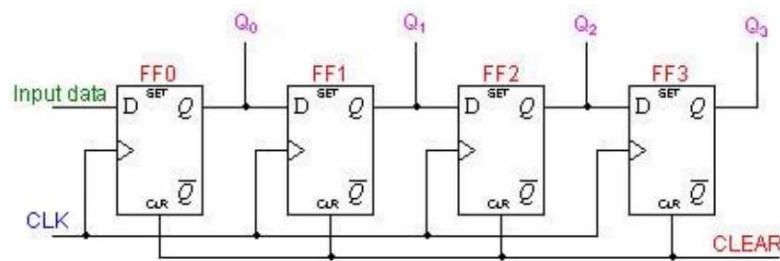
تمرین ۲ : مقایسه کننده ۴ بیتی با ۲ ثبات ۴ بیتی طراحی و تست نمایید . 



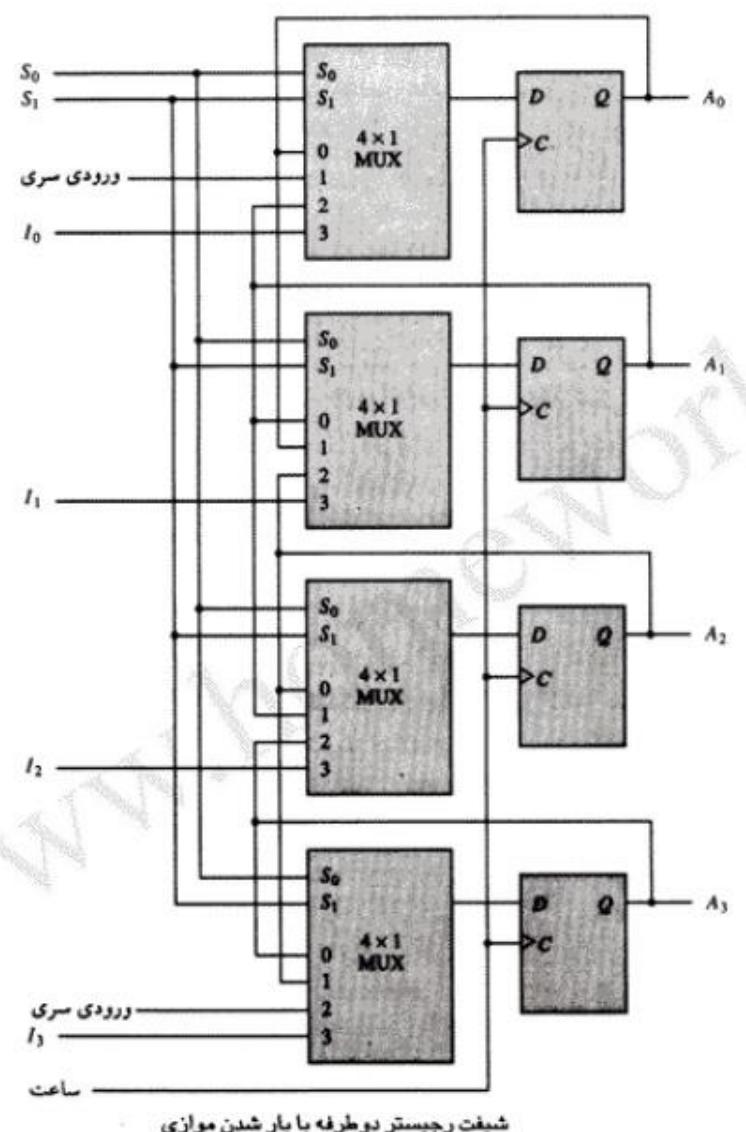
مقایسه گرمقدرا چهار بیتی

آزمایش شماره ۹ : شیفت رجیسترها

❖ با استفاده از فلیپ فلاپ D یک شیفت رجیستر طراحی کنید . SIPO .

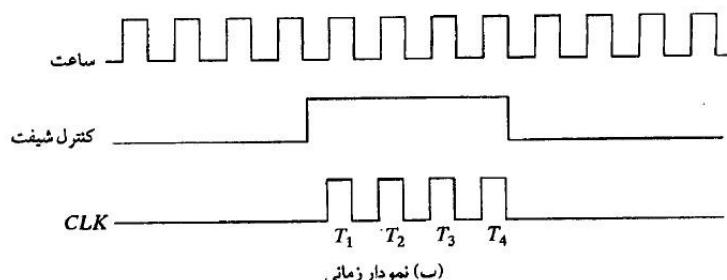
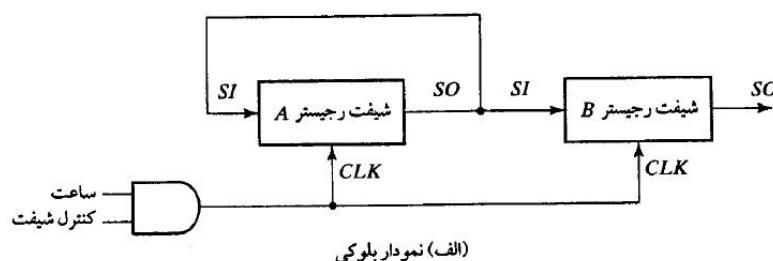


تمرین ۱ : شیفت رجیستر ۴ بیتی زیر را تست کنید . (این مدار هم بصورت رجیستر و هم شیفت رجیستر عمل می کند)



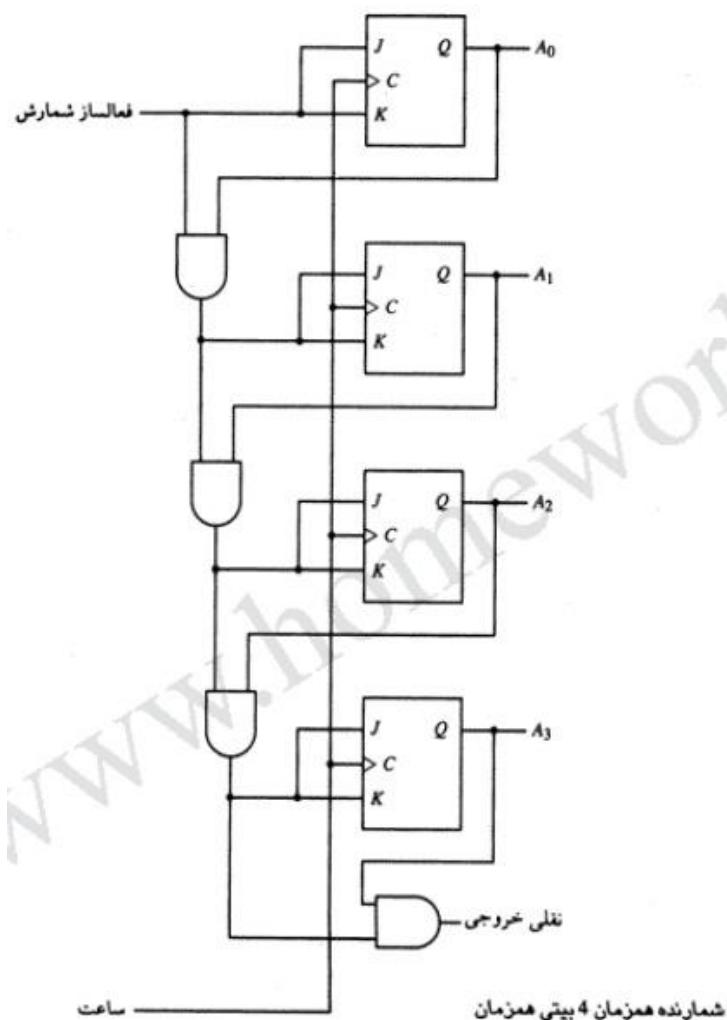
کنترل شیوه		عملکرد ثبات
S_1	S_0	
0	0	پلاتفیر
0	1	شیفت به راست (هاین)
1	0	شیفت به چپ (بالا)
1	1	بارشدن موازی

تمرين ۲ : با کمک مدارهای این آزمایش مداری بصورت دیاگرام زیر طراحی کنید . 



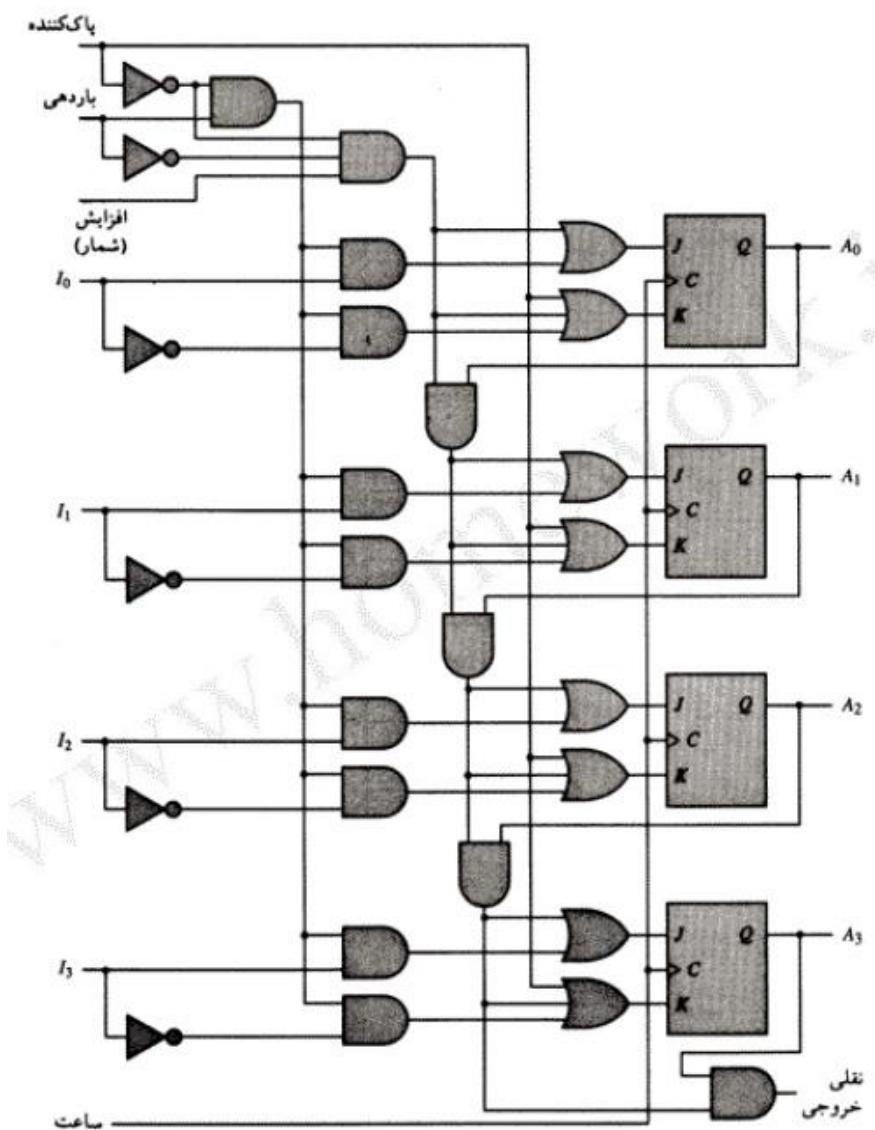
انتقال سریال از ثبات A به ثبات B

آزمایش شماره ۱۰ : شمارنده



تمرين ۱ : يك شمارنده از صفر تا ۹۹ طراحی کنيد . □

تمرین ۲ : شمارنده زیر را تست کنید . این شمارنده قابلیت شمارش از یک عدد مشخص را دارد .

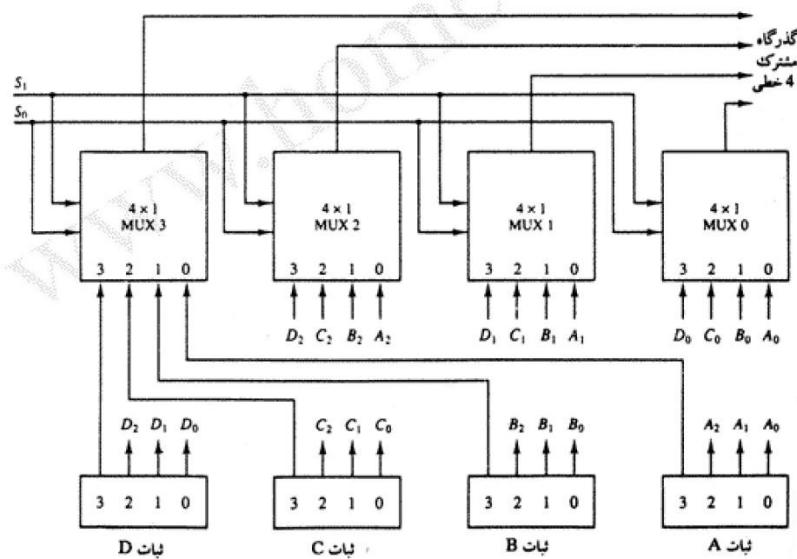


شمارنده بودویس ۴ بیتی با بارشدن موازی و پاک شدن همزمان

عمل	افزایش	ساعت	بارشدن	پاک شدن	ساعت
بلا تغییر	0	↑	0	0	
واحد افزایش شمارش	1	↑	0	0	
بارگذرن ورودی های I_0 تا I_3	x	↑	0	1	
پاک کردن خروجی ها به 0	x	↑	1	x	

آزمایش شماره ۱۱ : طراحی مدار گذرگاه

❖ در مدار شکل زیر به جای ثباتها می توانید از آی سی مربوط به رجیستر استفاده کنید.

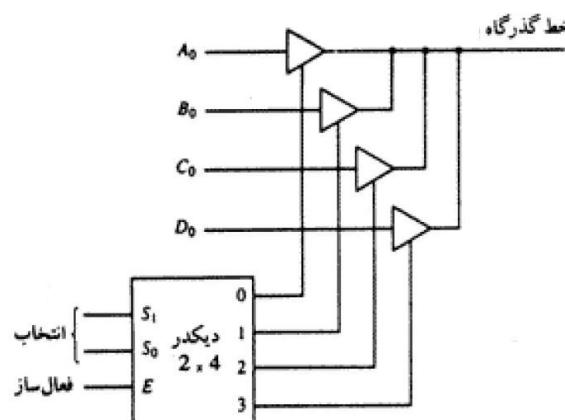


سیستم گذرگاه برای چهار ثبات

جدول تابع برای گذرگاه

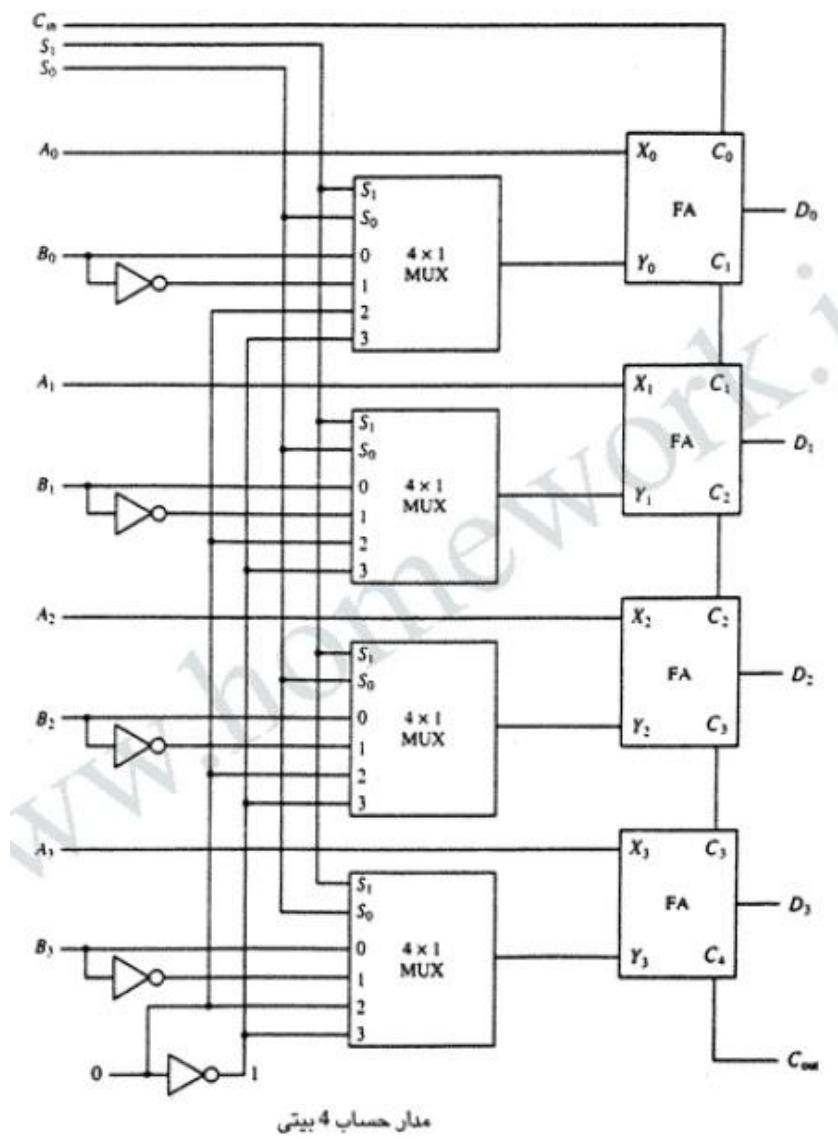
S_1	S_0	ثباتی که انتخاب می شود
0	0	A
0	1	B
1	0	C
1	1	D

❖ ابتدا مدار زیر را تست کنید . (گذر گاه تک بیتی)



❖ اکنون با استفاده از مدار فوق یک گذرگاه چهاربیتی طراحی کنید .

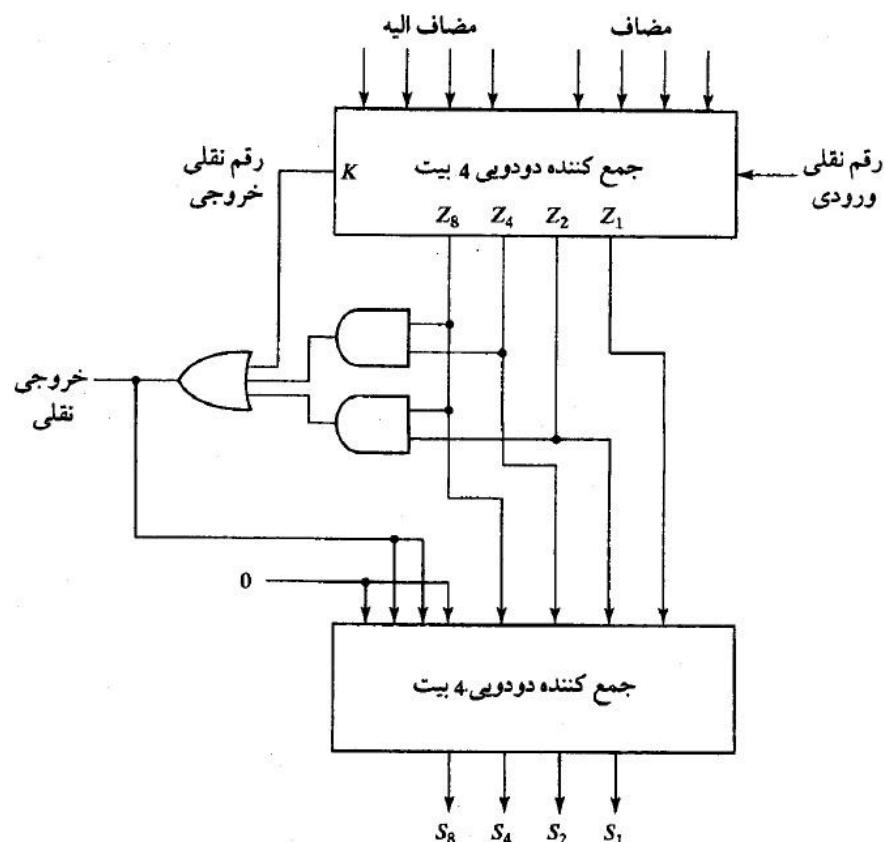
آزمایش شماره ۱۲ : طراحی مدار حساب



جدول تابع مدار حساب

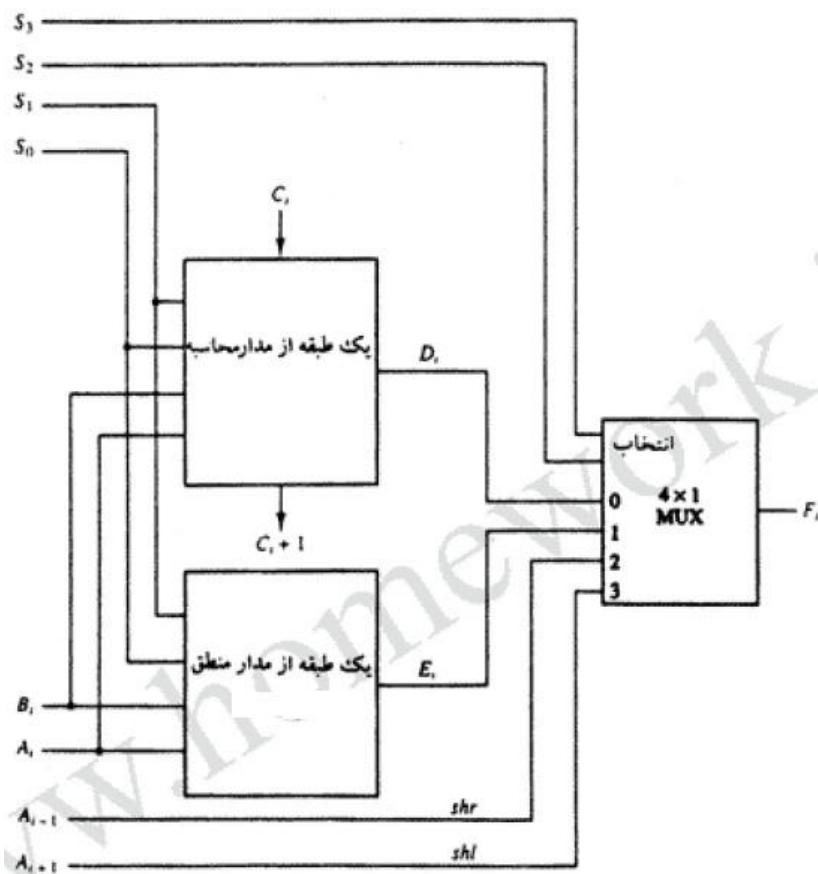
انتخاب			ورودی	خروجی	ریز عمل
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	
0	0	0	B	$D = A + B$	جمع
0	0	1	B	$D = A + B + 1$	جمع با نقلی
0	1	0	\bar{B}	$D = A + \bar{B}$	تفريق با فرض
0	1	1	\bar{B}	$D = A + \bar{B} + 1$	تفريق
1	0	0	0	$D = A$	Aنتقال
1	0	1	0	$D = A + 1$	Aفزایش
1	1	0	1	$D = A - 1$	کاهش
1	1	1	1	$D = A$	Aنتقال

تمرین ۱ : جمع کننده BCD



نمودار بلوکی یک جمع‌کننده BCD

آزمایش شماره ۱۳ : طراحی مدار محاسبه و منطق ALU

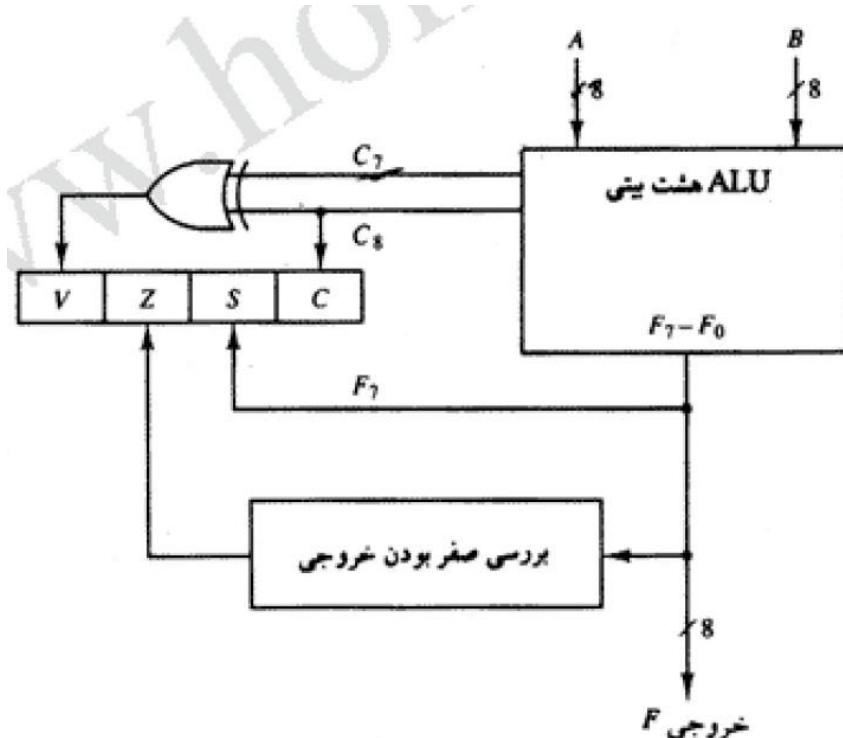


'یک طبقه از واحد حساب، منطق و شیفت'

جدول تابع برای واحد حساب، منطق و شیفت

انتخاب کننده عمل					عمل	تابع
S ₃	S ₂	S ₁	S ₀	C _{in}		
0	0	0	0	0	F = A	انقال
0	0	0	0	1	F = A + 1	افزایش
0	0	0	1	0	F = A + B	جمع
0	0	0	1	1	F = A + B + 1	جمع با رقم نقل
0	0	1	0	0	F = A + B̄	تفریق با فرض
0	0	1	0	1	F = A + B̄ + 1	تفریق
0	0	1	1	0	F = A - 1	A کاهش
0	0	1	1	1	F = A	A انقال
0	1	0	0	x	F = A ∧ B	AND
0	1	0	1	x	F = A ∨ B	OR
0	1	1	0	x	F = A ⊕ B	XOR
0	1	1	1	x	F = Ā	منس کردن A
1	0	x	x	x	F = shr A	شیفت A به راست و به داخل
1	1	x	x	x	F = shl A	شیفت A به چپ و به داخل

تمرین ۱ : یک ALU هشت بیتی به همراه ثبات پرچم طراحی کنید . 



بیت‌های ثبات و ضعیفیت

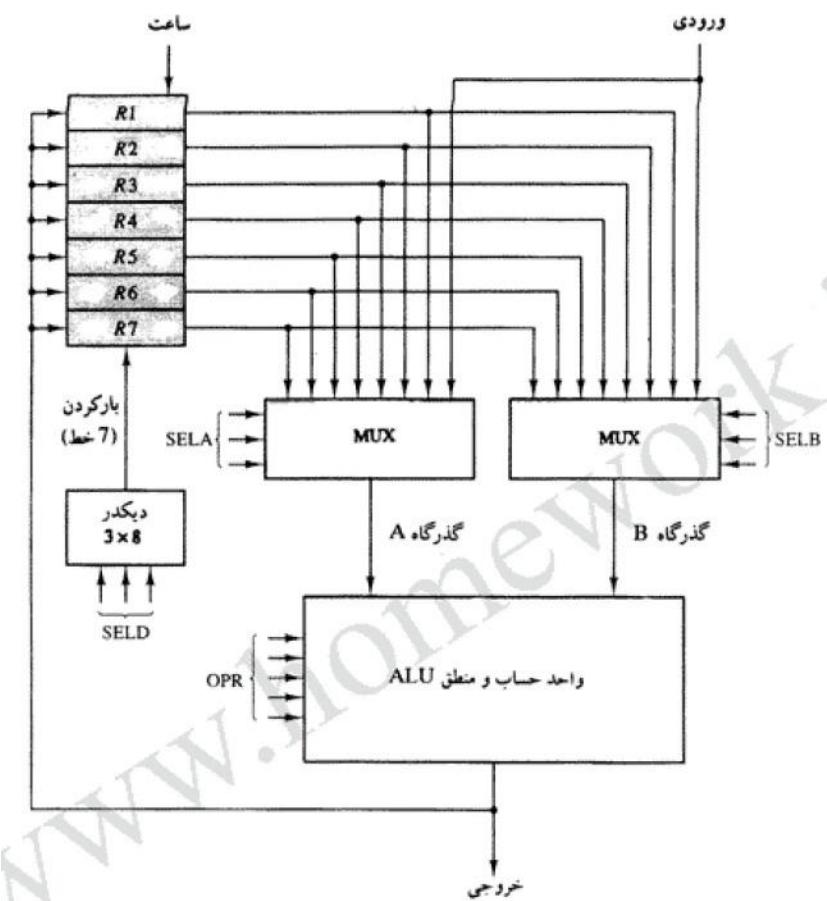
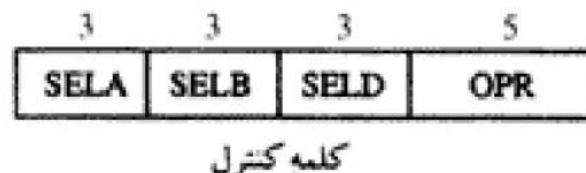
تمرین ۲ : یک ALU تک بیتی با جدول صحت زیر طراحی کنید . 

S1	S0	
0	0	AND
0	1	OR
1	0	NOT B
1	1	A+B

تمرین ۳ : یک ALU تک بیتی با جدول صحت زیر طراحی کنید . 

F	S1	S0	
0	0	0	AND
0	0	1	OR
0	1	0	NOT A
0	1	1	A+B
1	1	1	A-B

آزمایش شماره ۱۴ : طراحی واحد کنترل



کد دودویی	SELA	SELB	SELD
000	ورودی	ورودی	ج
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

opr انتخابگر	عمل	سبل
00000	A انتقال	TSFA
00001	A افزایش	INCA
00010	A+B جمع	ADD
00101	A-B تفریق	SUB
00110	A کاهش	DECA
01000	B و A بات های AND	AND
01010	B و A بات های OR	OR
01100	B و A بات های XOR	XOR
01110	A منم کردن	COMA
10000	شیفت A به راست	SHRA
11000	شیفت A به چپ	SHLA

مشخص کردن سبل					
ریز عمل	SEL A	SEL B	SEL D	OPR	کلمه کنترل
$R1 \leftarrow R2 - R3$	R2	R3	R1	SUB	010 011 001 00101
$R4 \leftarrow R4 \vee R5$	R4	R5	R4	OR	100 101 100 01010
$R6 \leftarrow R6 + 1$	Ri	—	R6	INCA	110 000 110 00001
$R7 \leftarrow R1$	R1	—	R7	TSFA	001 000 111 00000
$Output \leftarrow R2$	R2	—	None	TSFA	010 000 000 00000
$Output \leftarrow Input$	Input	—	None	TSFA	000 000 000 00000
$R4 \leftarrow sh1 R4$	R4	—	R4	SHLA	100 000 100 11000
$R5 \leftarrow 0$	R5	R5	R5	XOR	101 101 101 01100

طراحی ، شبیه سازی ، پیاده سازی و برنامه ریزی چیپ های شرکت XILINX به کمک نرم افزار ISE14.4



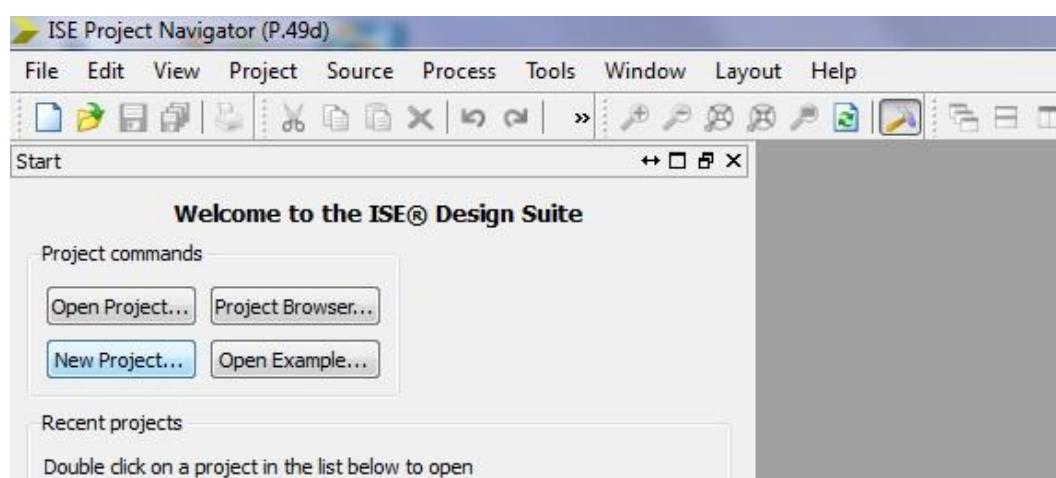
فرایند طراحی :

ابتدا بر روی آیکن زیر کلیک کنید



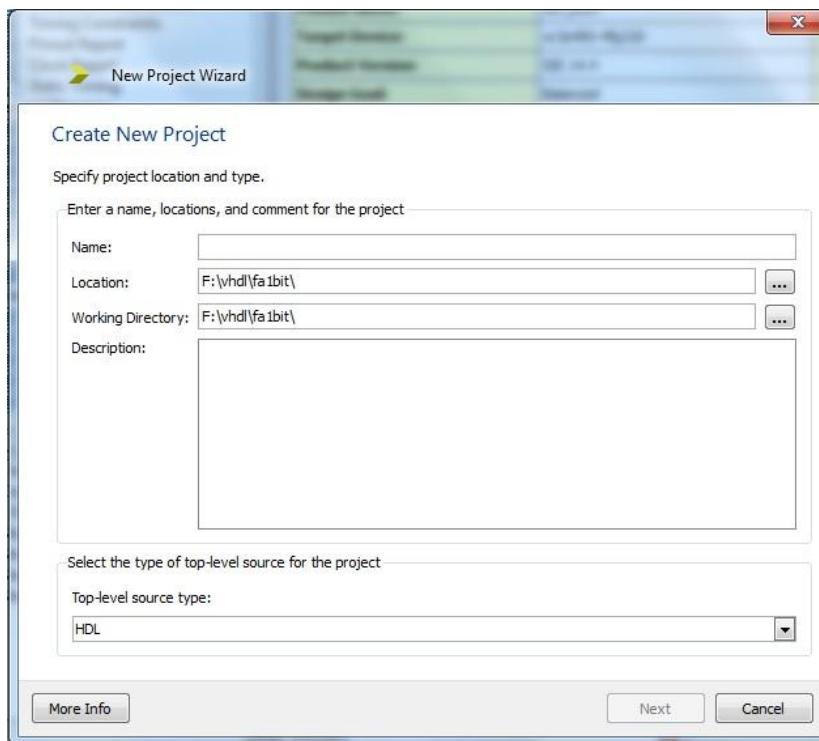
پنجره زیر ظاهر می شود . در صورت عدم ظاهر شدن پنجره به فرم فوق ابتدا از مسیر زیر پروژه در حال انجام را بسته و سپس پروژه جدید را باز کنید .

File > Cloce Project...



سپس از مسیر زیر پروژه جدید را شروع می کنیم .

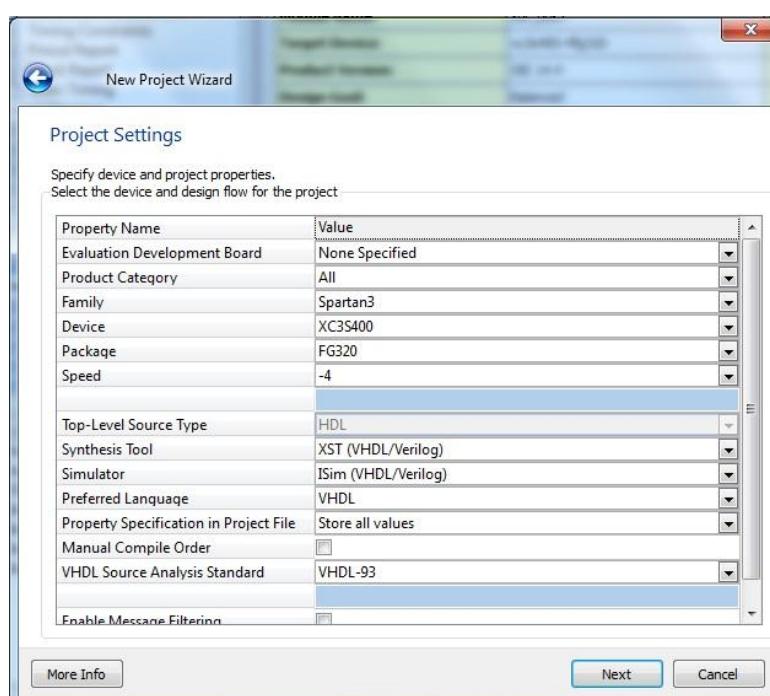
New Project...



در قسمت Name می توان یک اسم دلخواه و در قسمت Location مسیر ذخیره پروژه را انتخاب و مشخص می کنیم.

همچنین در قسمت Top-level source type می توان نوع یا روش توصیف طراحی را مشخص کرد .

با کلیک بر روی Next به مرحله بعد می رویم :



در قسمت Family نوع خانواده چیپ ها و در قسمت Device شماره چیپ و در قسمت Package نوع بسته بندی چیپ و در قسمت Speed فرکانس کاری چیپ را مشخص می کنیم.

پارامترهای دیگری نیز برای تنظیم کردن وجود دارند مثلاً نوع شبیه ساز و یا ابزار سنتز و ...

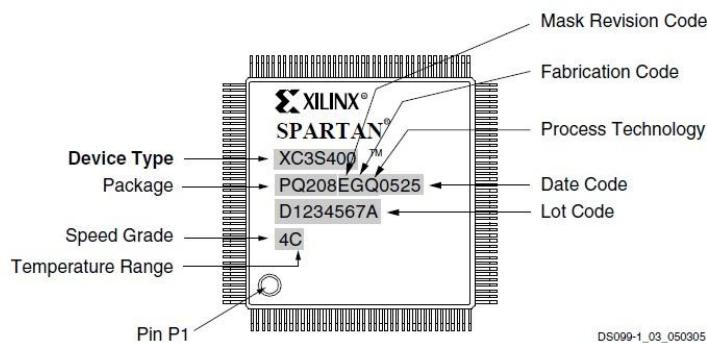
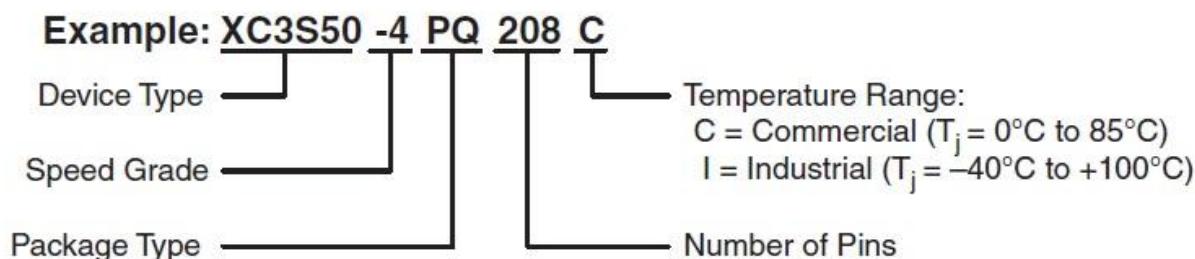


Figure 2: Spartan-3 FPGA QFP Package Marking Example for Part Number XC3S400-4PQ208C

DS099 (v3.1) June 27, 2013
Product Specification

www.xilinx.com

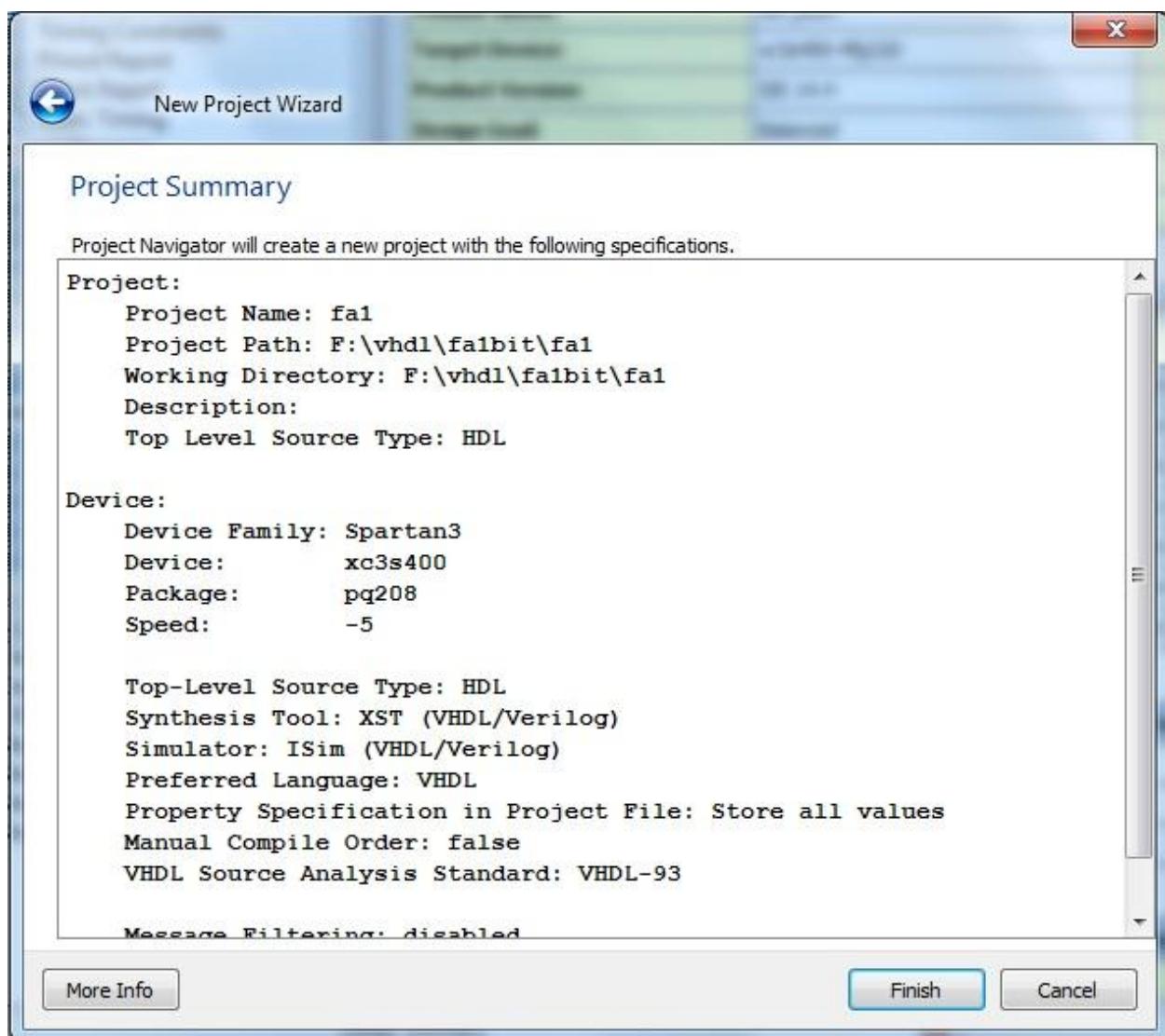
5



-4 > 630MHz

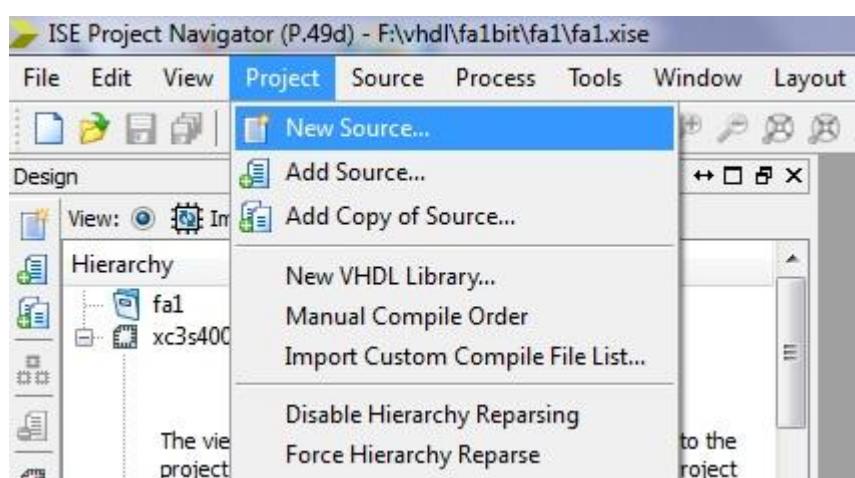
-5 > 725MHz

با کلیک بر روی Next به مرحله بعد می رویم :



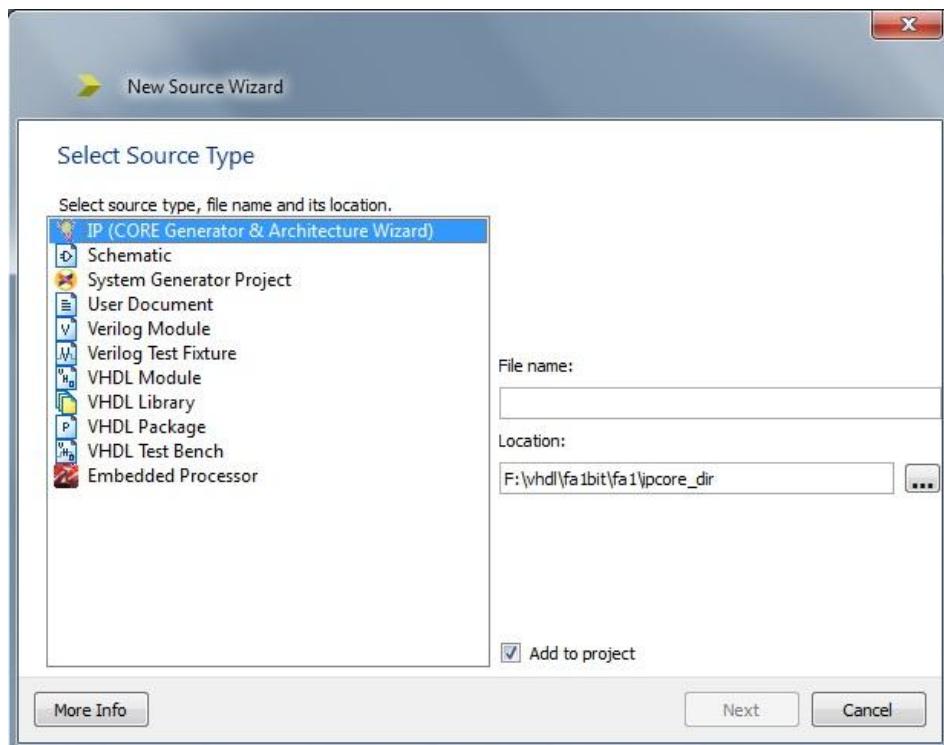
در شکل فوق یک گزارش و خلاصه‌ای از تنظیمات انجام شده مشاهده می‌گردد. بر روی Finish کلیک می‌کنیم.

از مسیر زیر یک source جدید ایجاد می‌کنیم:

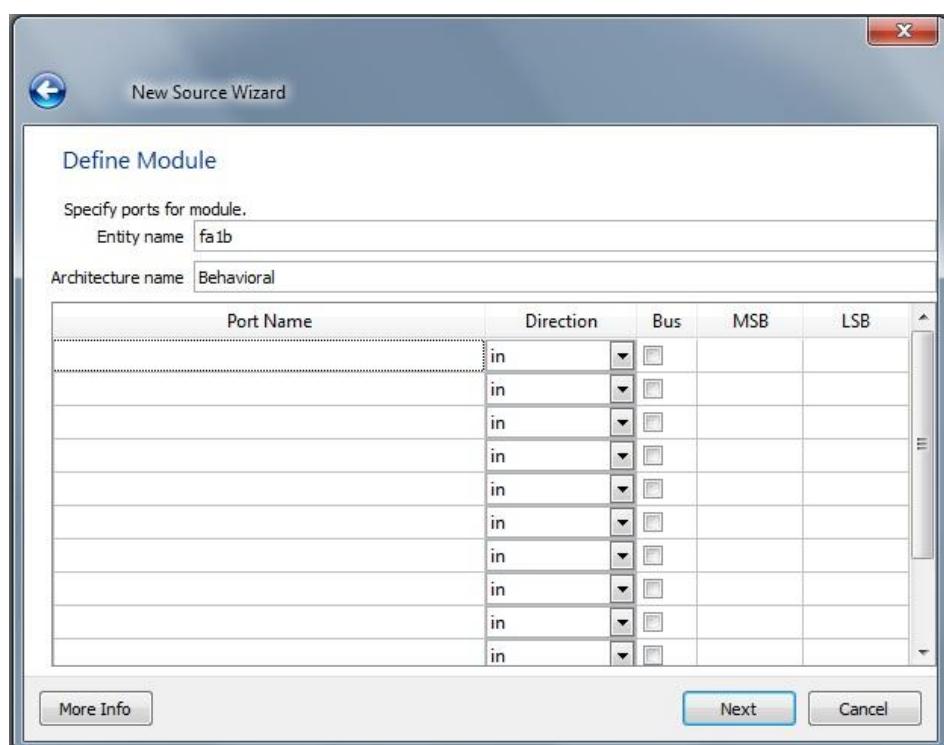


در پنجره ظاهر شده نوع عملیات را انتخاب می کنیم مثلا VHDL Module

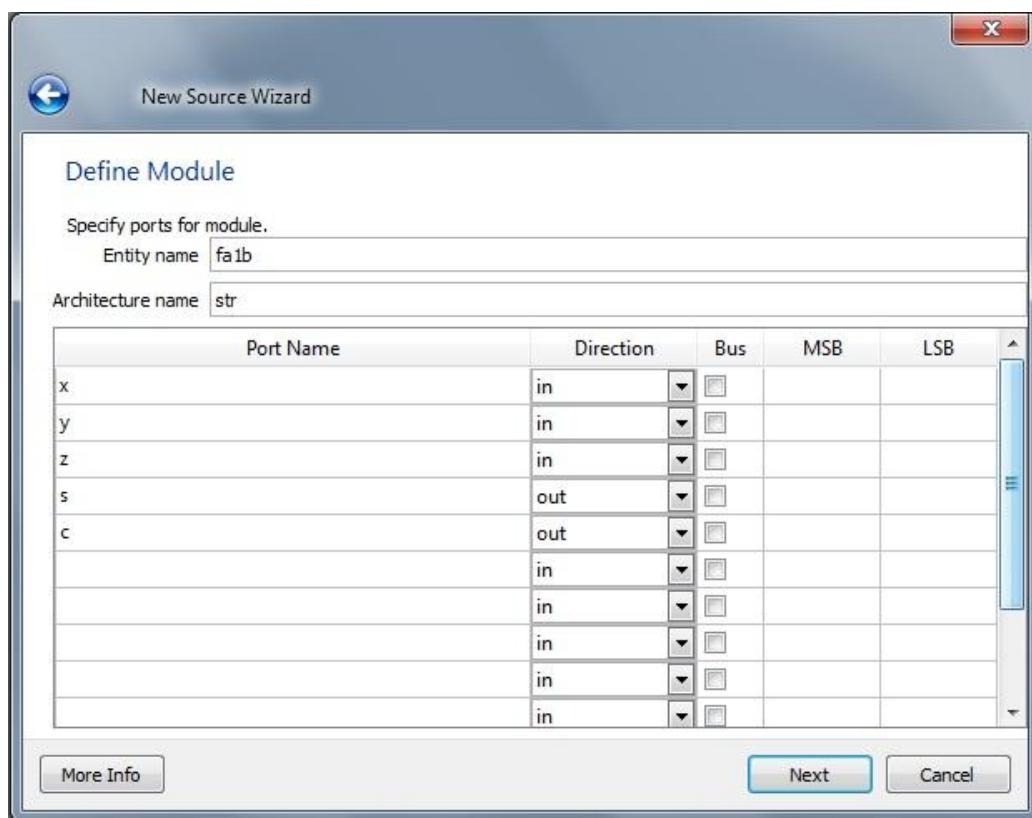
همچنین یک اسم برای این نوع عملیات انتخاب و مسیر ذخیره نیز مشخص می شود ، دقت شود نام فایل بایستی با نام Yeksan باشد :



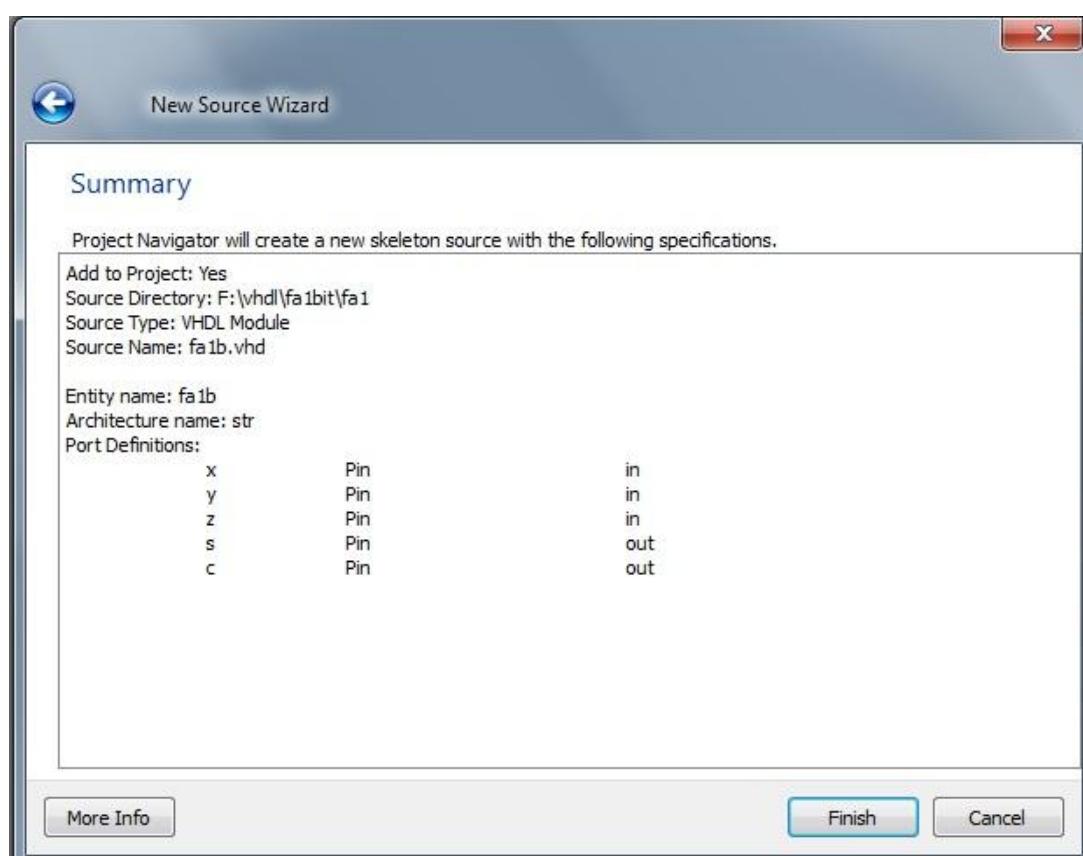
با کلیک بر روی Next به مرحله بعد می رویم :



در این مرحله بایستی پورت های ورودی و خروجی یا دو طرفه را مشخص کنیم مثلًا به صورت زیر :



با کلیک بر روی Next به مرحله بعد می رویم :



در شکل فوق یک گزارش و خلاصه ای از تنظیمات پورتها مشاهده می گردد . بر روی Finish کلیک می کنیم .

در پنجره ظاهر شده می توان کدهای VHDL را تایپ نمود مثلا برای یک گیت AND بصورت زیر :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity and1 is
    Port ( a : in STD_LOGIC;
            b : in STD_LOGIC;
            f : out STD_LOGIC);
end and1;

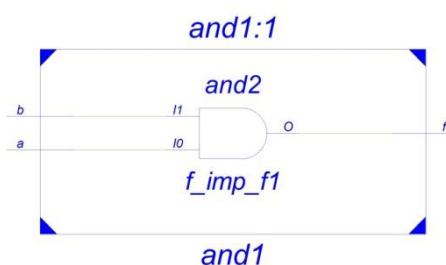
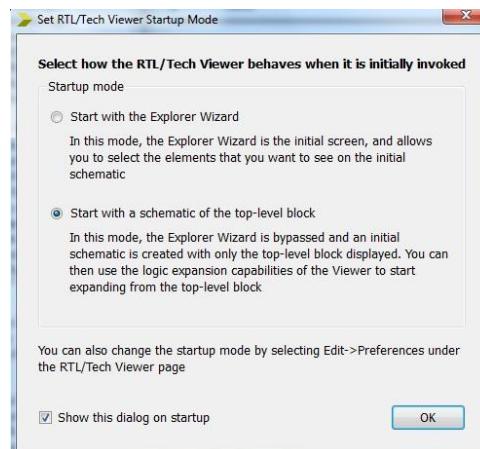
architecture Behavioral of and1 is

begin
    f<=a and b;
end Behavioral;
```

پس از تایپ کدها ابتدا بایستی کدها از نظر خطای نگارش بررسی و سپس عملیات سنتز انجام شود بدین منظور ابتدا بر روی Synthesize - XST کلیک و سپس بر روی Check Syntax کلیک می کنیم . خطاهای احتمالی را نیز برطرف کرده و مجدداً عملیات فوق را تکرار می کنیم .



با کلیک بر روی View RTL Schematic می توان نتیجه عملیات سنتز را مشاهده کرد مثلا برای یک گیت AND زیر ظاهر شده است :



از قسمت زیر می توان تعداد قطعات استفاده شده از چیپ و قطعات باقیمانده را مشاهده کرد :

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices		1	4656
Number of 4 input LUTs		1	9312
Number of bonded IOBs		3	232

از این قسمت می توان جهت گزارش گیری از عملیات سنتز استفاده نمود :

Screenshot of the Synthesis Report window showing resource utilization and timing analysis.

- Left Panel:** Navigation tree showing:
 - Place and Route Messages:
 - Timing Messages
 - Bitgen Messages
 - All Implementation Messages
 - Detailed Reports:
 - Synthesis Report
 - Translation Report
 - Map Report
 - Place and Route Report
 - Post-PAR Static Timing Report
 - Power Report
 - Bitgen Report
 - Secondary Reports:
 - ISIM Simulator Log
 - WebTalk Report
- Right Panel:**
 - Device Utilization Summary:**

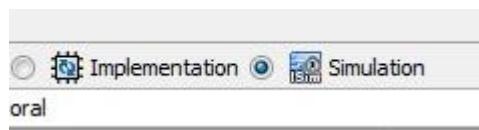
Selected Device : 3s500efg320-5

Number of Slices:	1	out of	4656	0%
Number of 4 input LUTs:	1	out of	9312	0%
Number of IOs:	3			
Number of bonded IOBs:	3	out of	232	1%
 - Partition Resource Summary:**

No Partitions were found in this design.
 - TIMING REPORT**

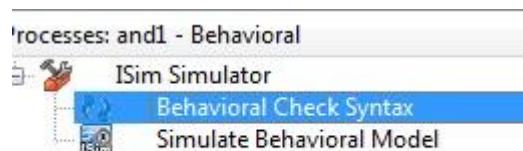
فرایند شبیه سازی :

ابتدا بر روی ایکن زیر کلیک می کنیم

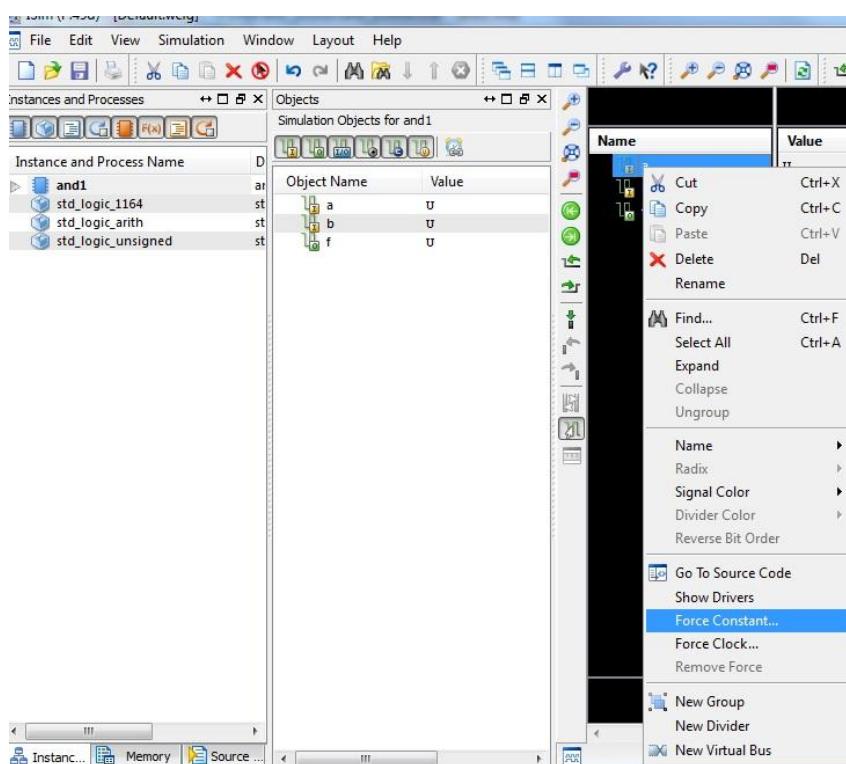


به دو روش می توان شبیه سازی را انجام داد البته از طریق نرم افزار Modelsim نیز شبیه سازی انجام می شود که بعداً به آن خواهیم پرداخت.

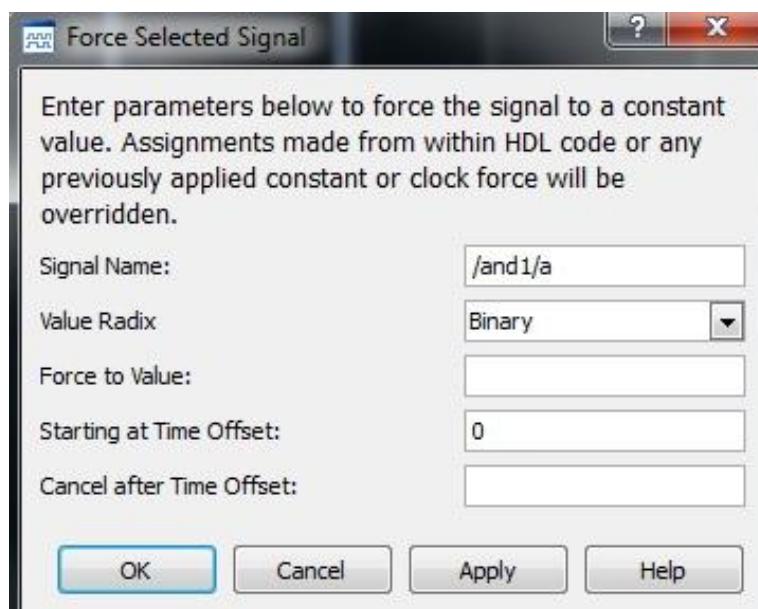
روش اول : در این روش ابتدا بر روی طرح کلیک کرده و سپس از پنجره زیر با کلیک بر روی **Model** وارد نرم افزار شبیه ساز می شویم



همانطور که دیده میشود در نرم افزار شبیه ساز تمامی گره های ورودی و خروجی دیده می شوند . با کلیک راست بر روی گره های ورودی و انتخاب گزینه ... Force Constant... می توان گره های ورودی را مقداردهی کرد



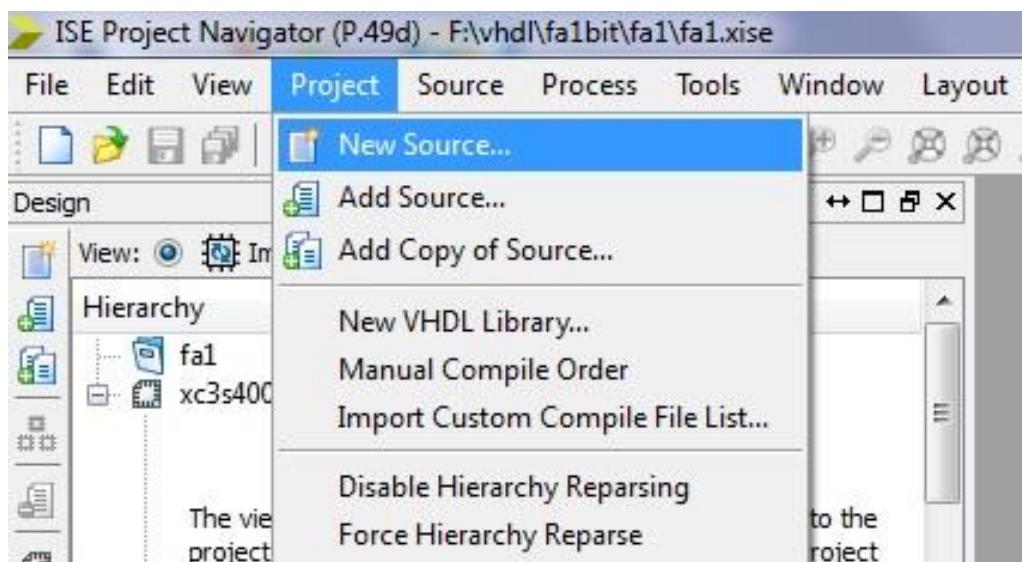
در پنجره ظاهر شده می توان مقدار و بناء مورد نظر را وارد کرد



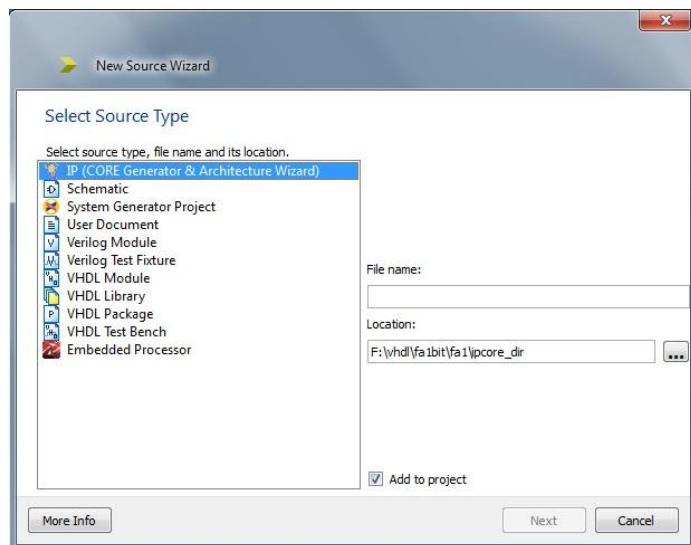
پس از مقدار دهی ورودی ها از قسمت های زیر می توان زمان پایان شبیه سازی و یا اجرای آن و یا بازنشانی آن را انجام داد



روش دوم : در این روش که روش حرفه ای تری می باشد از یک برنامه به نام Test Bench برای شبیه سازی استفاده می شود بدین صورت که ابتدا بایستی از پنجره زیر یک Source جدید ایجاد کرد



سپس از پنجره زیر گزینه VHDL Test Bench را انتخاب کرد



با انتخاب این گزینه و انتخاب اسم و مکان یک پنجره ظاهر شده که در آن باستی کد VHDL مربوط به برنامه شبیه سازی نوشته شود . مثلا یک نمونه برنامه تست جهت تست گیت AND به صورت زیر است :

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test1 IS
END test1;

ARCHITECTURE behavior OF test1 IS
COMPONENT and_2input
PORT (
    a : IN std_logic;
    b : IN std_logic;
    f : OUT std_logic );
END COMPONENT;
signal a : std_logic := '0';
signal b : std_logic := '0';
signal f : std_logic;
BEGIN
uut: and_2input PORT MAP (

```

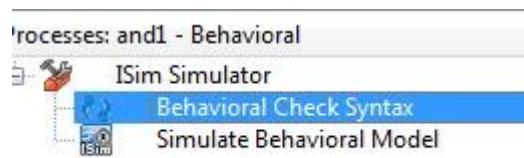
```
a => a,  
b => b,  
f => f );
```

```
a <= '1' after 100ns, '0' after 200ns, '1' after 300ns, '0' after  
400ns, '1' after 500ns, '0' after 600ns;
```

```
b <= '1' after 200ns, '0' after 400ns, '1' after 600ns;
```

```
END;
```

پس از پایان نوشتن ابتدا بایستی از طریق ایکن Behavioral Check Syntax خطای نگارشی بررسی و سپس از گزینه Simulate Behavioral Model عملیات شبیه سازی انجام می شود متنها این بار بطور اتوماتیک شبیه سازی انجام خواهد شد

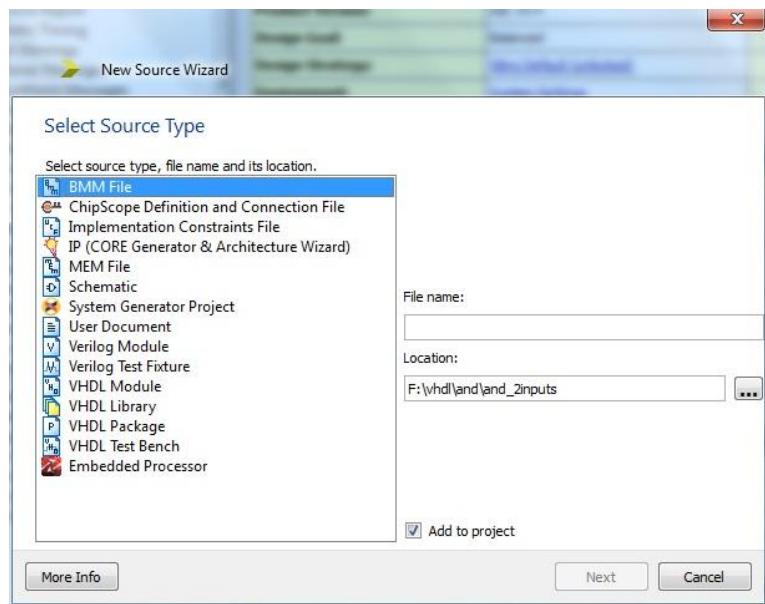


با کلیک بر روی ایکن زیر کل فضای شبیه سازی قابل رویت می باشد



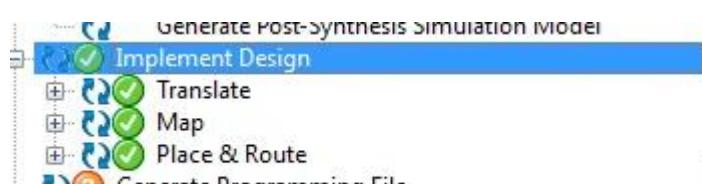
فرایند پیاده سازی :

در این مرحله طرح بایستی در چیپ جاسازی شده و پایه های ورودی و خروجی طرح در چیپ مشخص شوند بدین منظور ابتدا بر روی طرح کلیک راست کرده و گزینه New Source را انتخاب می کنیم

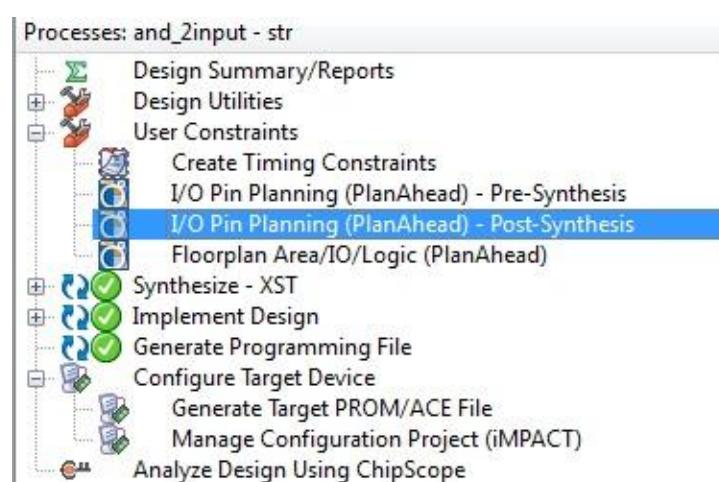


از پنجره ظاهر شده Implementation Constraints File را انتخاب می کنیم.

سپس گزینه Implement Design را برای شروع فرایند پیاده سازی فعال می کنیم.



سپس به قسمت Plan Ahead I/O Pin Planning (lanAhead) – Post-Synthesis رفته و نرم افزار Plan Ahead را اجرا می کنیم



با اجرای این نرم افزار می توان پایه های طرح و نوع پایه و ولتاژ کار پایه را برای چیپ تعريف کرد:

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Stre...	Slew Type	Pull Type
All ports (3)											
Scalar ports (3)											
a	Input		P2	<input checked="" type="checkbox"/>		7 LVCMS25	2.500			NONE	
b	Input		P3	<input checked="" type="checkbox"/>		7 LVCMS25	2.500			NONE	
f	Output		P4	<input checked="" type="checkbox"/>		7 default (LVCMS25)	2.500		12 SLOW	NONE	

در شکل زیر نمایی از پایه های چیپ دیده میشود :



پس از ذخیره و خارج شدن از این نرم افزار با کلیک کردن بر روی طرح ایجاد شده پنجره جدیدی ظاهر می شود که در این پنجره نیز می توان وضعیت پایه ها را تغییر داد

Implementation View: Implementation Simulation

Hierarchy

- and_2inputs
- xc3s400-5pq208
 - and_2input - str (and_2input.vhd)
 - and_2input.ucf

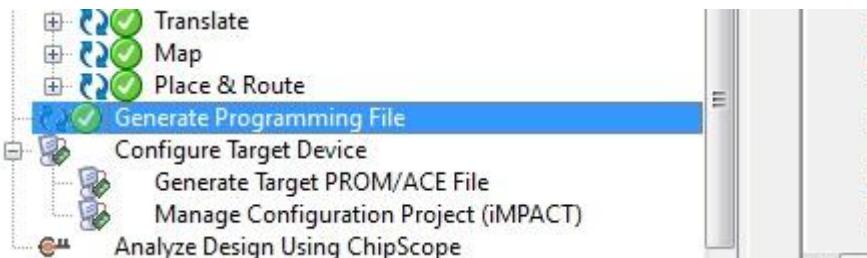
```

1 | # PlanAhead Generated IO constraints
2 | NET "a" IOSTANDARD = LVCMS25;
3 | NET "b" IOSTANDARD = LVCMS25;
4 |
5 |
6 | # PlanAhead Generated physical constraints
7 | NET "a" LOC = P2;
8 | NET "b" LOC = P3;
9 | NET "f" LOC = P4;
10

```

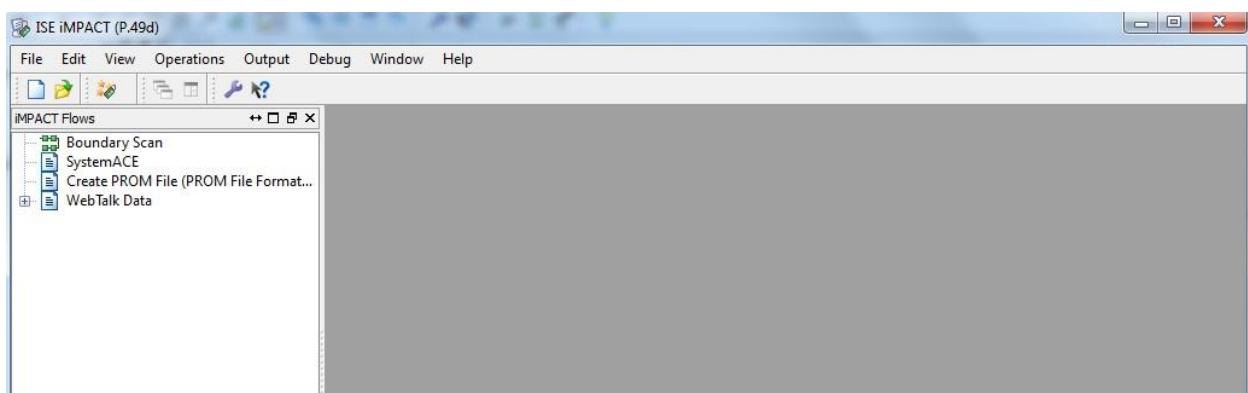
فرایند برنامه ریزی :

در این مرحله می توان طرح ایجاد شده را بر روی چیپ مورد نظر پروگرام کرده و بطور عملی تست کرد برای این کار ابتدا فایل مورد نیاز دستگاه پروگرام بصورت زیر با کلیک بر روی Generate Programming File ایجاد می شود .



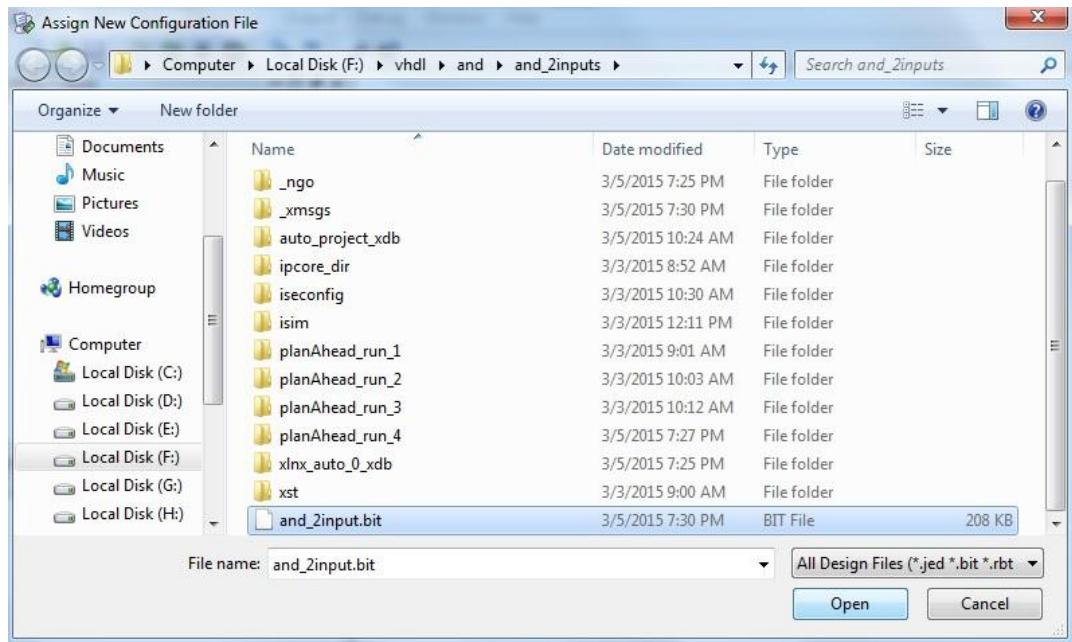
بدین منظور فایلی با پسوند bit ایجاد می شود .

اگر فایل مورد بایستی بر روی Flash پروگرام شود بایستی بر روی Generate Target PROM/ACE File کلیک کرد . در انتها با کلیک بر روی Manage Configuration Project(iMPACT) نرم افزار مربوط به پروگرامر اجرا خواهد شد .

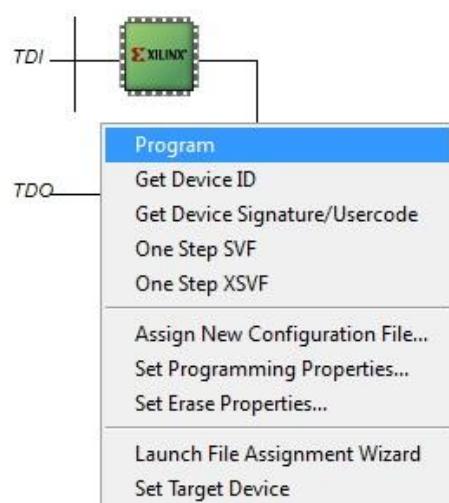
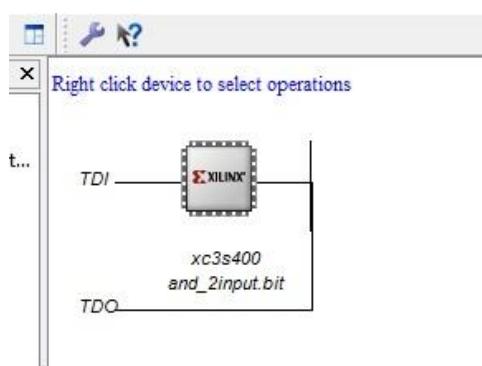


ابتدا بایستی بر روی Boundary Scan کلیک کنیم و سپس در پنجره ایجاد شده کلیک راست کرده و با انتخاب گزینه فایل bit ایجاد شده در مرحله قبل را انتخاب کنیم :





اکنون بر روی چیپ کلیک راست کرده گزینه Program را انتخاب می کرم



مقدمه ای بر FPGA و زبان توصیف سخت افزاری VHDL

Field Programmable Gate Array **FPGA**



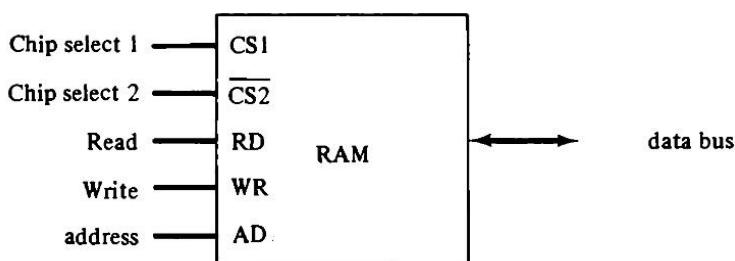
انواع طراحی سخت افزار به سبک دیجیتال :

- ترانزیستور
- مدار و گیت
- بلوک و سیستم
- برنامه نویسی

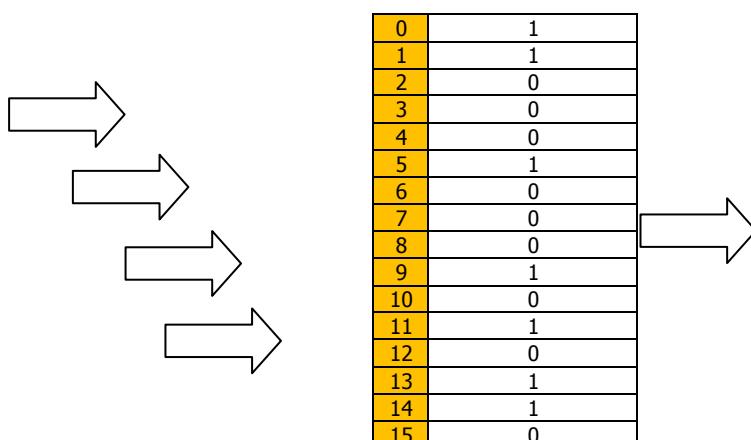
مثال : فرض کنید می خواهیمتابع زیر را تحقق بخشیم

$$F(A,B,C,D) = \sum(0,1,5,9,11,13,14)$$

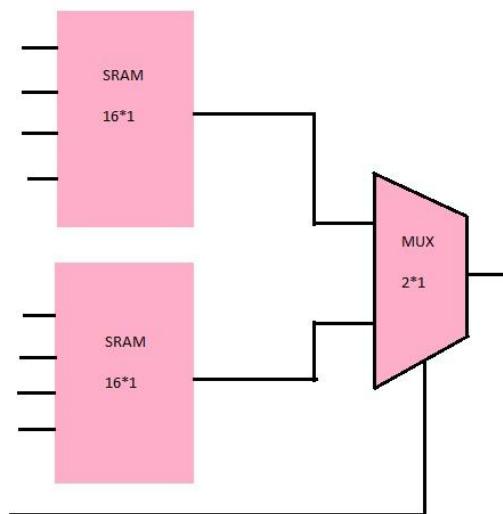
برای این منظور میتوان از جدول کارنو و سپس گیت های منطقی کمک گرفت و یا می توان به کمک مولتی پلکسر و یا دیکدر این طراحی را انجام داد . اما می خواهیم تابع فوق را به کمک چیپ RAM طراحی کنیم . این چیپ در ساده ترین حالت به شکل زیر می باشد :



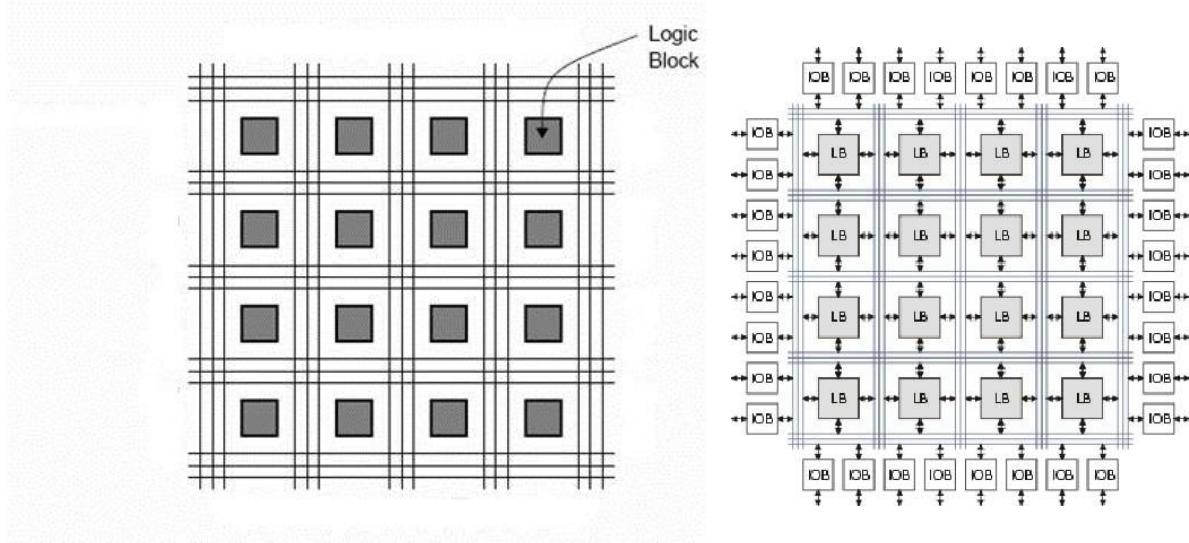
با توجه به تابع داده شده نیازمند یک SRAM با ۴ خط آدرس و یک خط دیتا هستیم بنابراین اندازه این حافظه 16×1 می شود .



اگر متغیرهای ورودی کمتر از ۴ بیت باشند می توان پایه های آدرس با ارزش بالاتر را زمین کرد اما اگر متغیرهای ورودی بیشتر از ۴ بیت باشند از شکل زیر استفاده می شود :

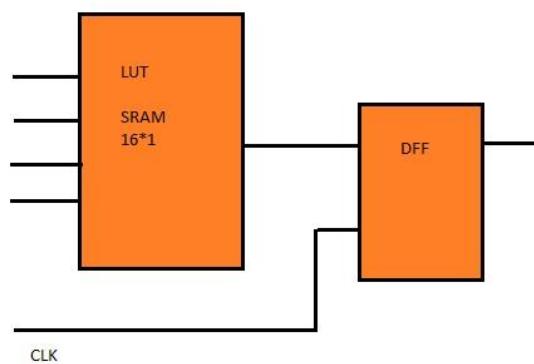


در تراشه های FPGA به هر کدام از این 16×1 یک lookup table یا به اختصار LUT گفته می شود .
شکل های زیر نحوه قرار گیری SRAM ها در تراشه FPGA نشان می دهد .

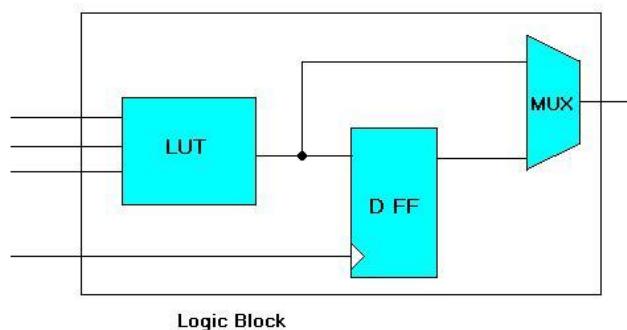


در شکل فوق SRAM ها بخشی از logic block ها هستند فقط تنها ایراد این نوع طراحی آن است که با قطع برق اطلاعات SRAM ها پاک خواهد شد بدین منظور در کنار هر تراشه FPGA یک تراشه ROM نیز قرار می دهند تا به محض وصل شدن برق اطلاعات از ROM وارد بلوک های SRAM شود .

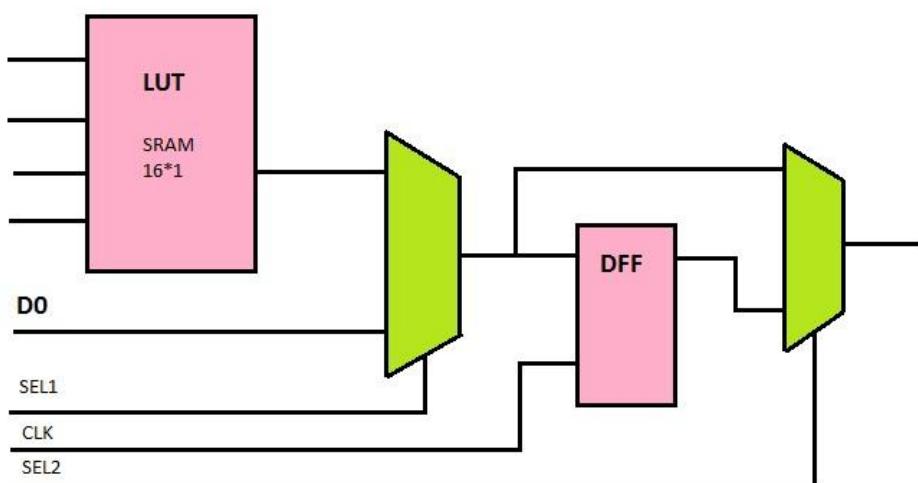
همانطور که در بلوک های SRAM دیده می شود این بلوک ها فقط قادر به پیاده سازی توابع منطقی ترکیبی هستند . برای پیاده سازی توابع منطقی ترتیبی از بلوک منطقی زیر استفاده می شود :



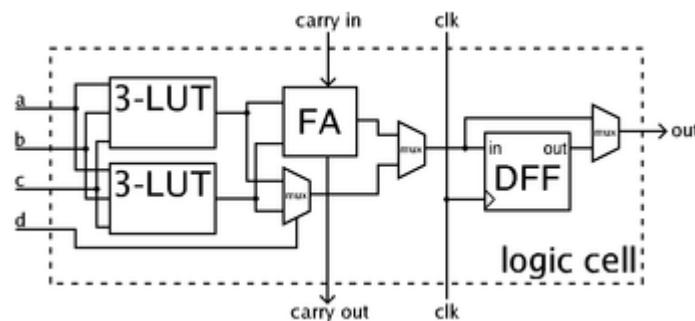
البته که بلوک های انعطاف پذیر بایستی بقلمارا دارای پیاده سازی ترکیبی و ترتیبی بصورت زیر باشند :



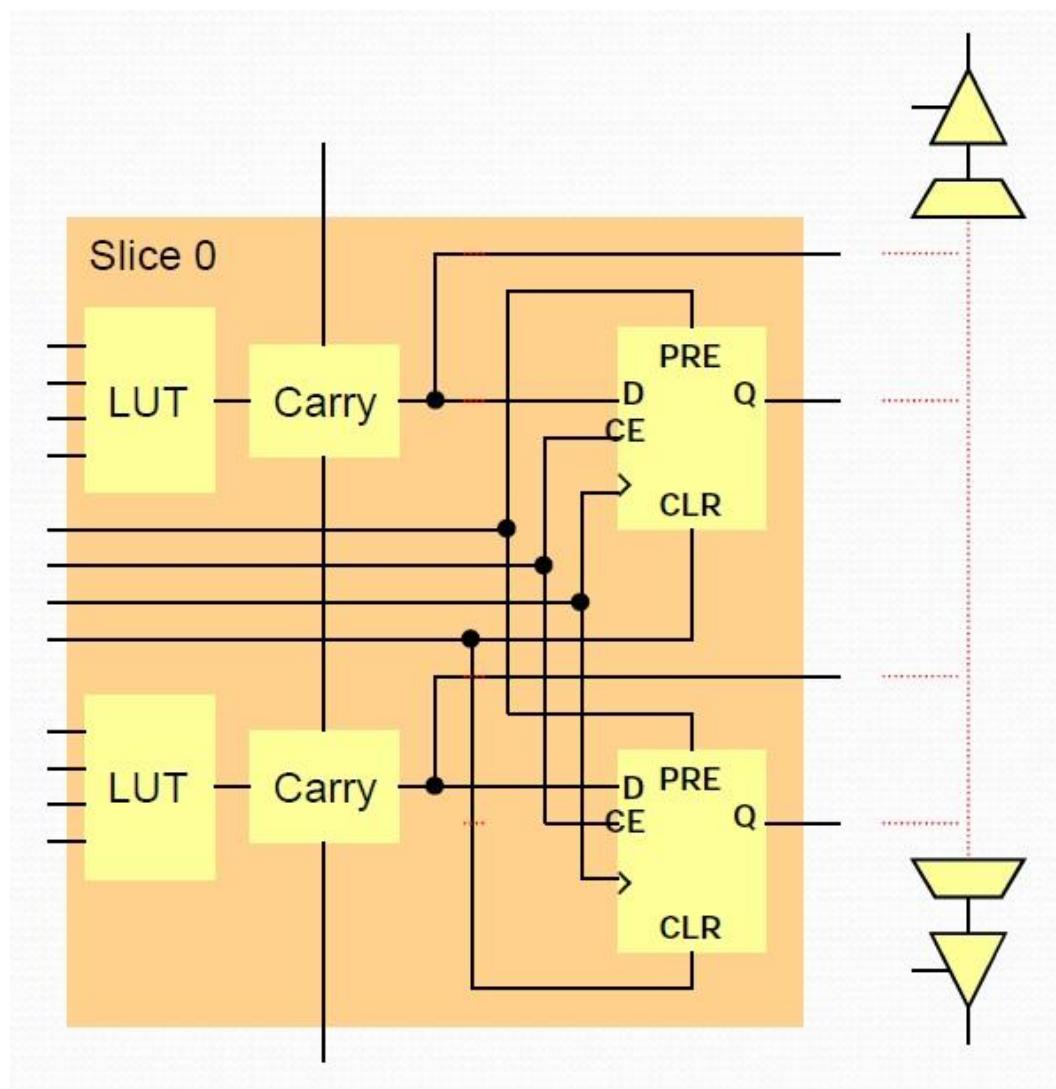
در شکل زیر یک بلوک کاملتری از بلوک های منطقی دیده می شود که به سلول منطقی LC Logic Cell یا به اختصار معروف هستند .



از آنجایی که عملیات های جمع و ضرب بسیار پر کاربرد هستند در داخل LC ها این بلوک ها برای انعطاف بیشتر قرار داده شده است که یک CLB و یا Configurable Logic Block را تشکیل خواهد داد :



: Slice

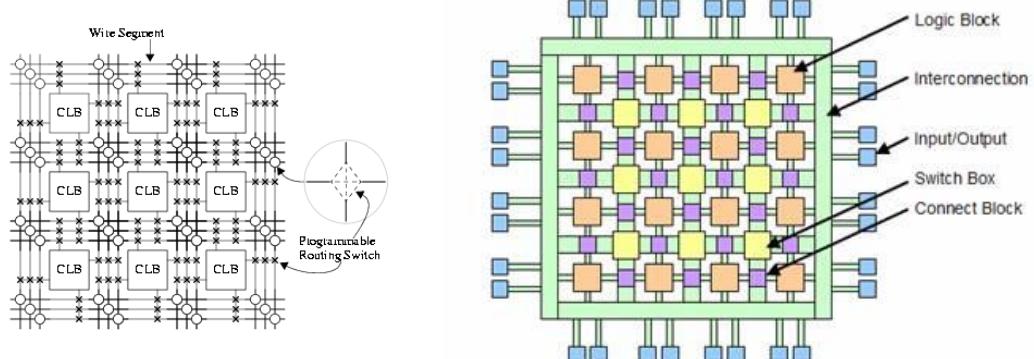
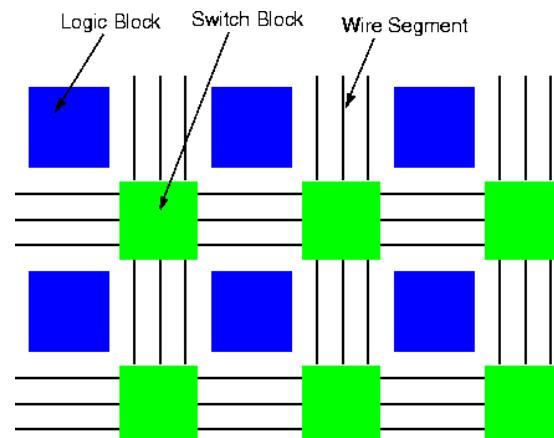
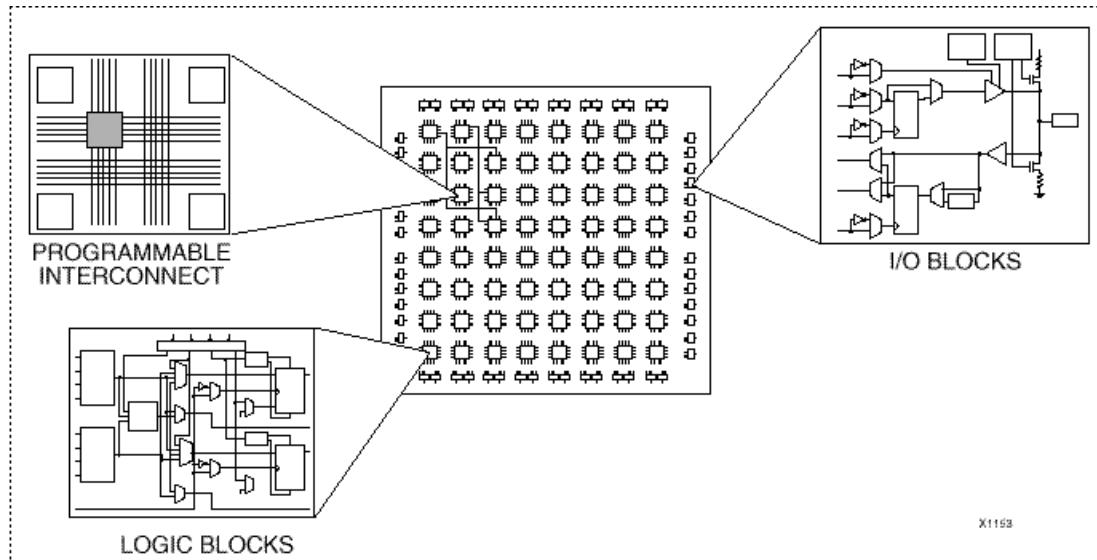


معمولاً یک FPGA بطور کلی از سه قسمت اصلی تشکیل شده است :

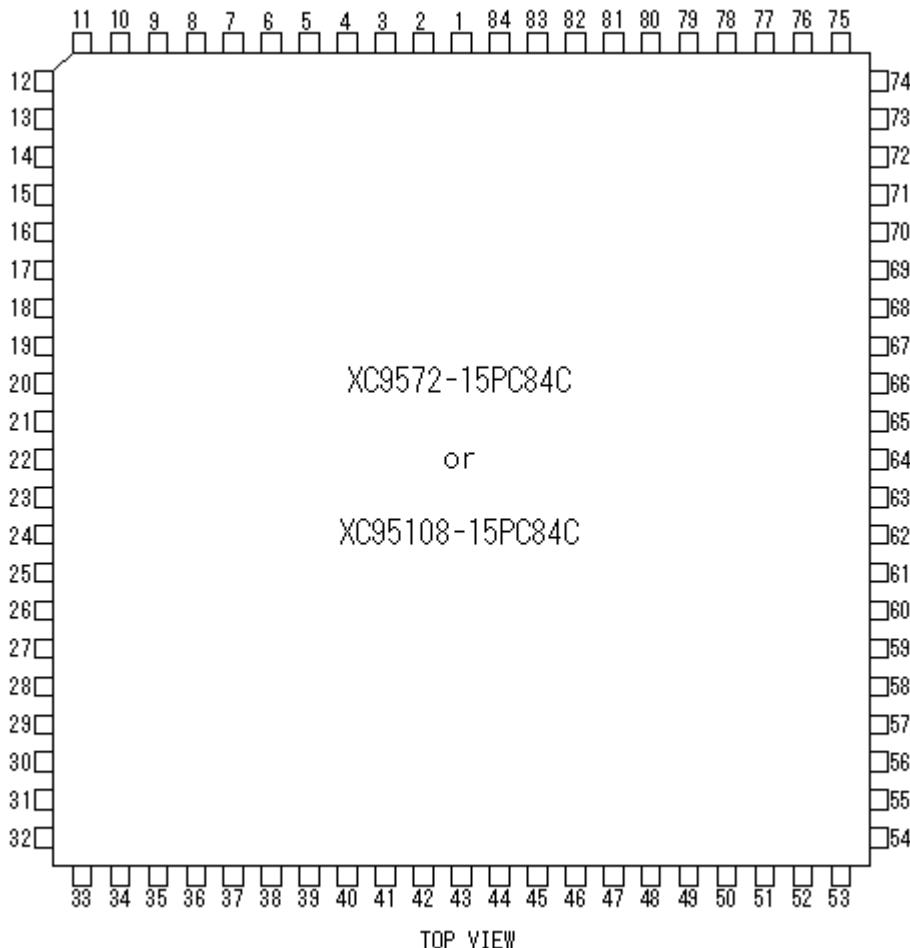
LC ➤

SM ➤

IOB ➤



Switch matrix ها برای اتصال بین CLB ها استفاده می شوند .
در واقع همان عملی است که تمامی Configuration SM ها و LUT ها و mux ها و تمامی قطعات FPGA مناسب طرح برنامه ریزی می شوند و نهایتاً نشان می دهند که FPGA قرار است چه عملی را انجام دهد .



محاسن استفاده از FPGA

FPGA ها در پیاده سازی توابع نسبتاً پیچیده و پیچیده دیجیتال به کار می روند که نیاز به سرعت پردازش بالای دارند . علاوه بر این کاهش سخت افزار مورد نیاز و همچنین برنامه نویسی ساده و استاندارد نیز از دیگر مزیت های استفاده از FPGA است . آنچه که قابلیت و توانایی FPGA ها را بالا برده است توانایی هایی است که پاره ای از آنها در زیر آمده است :

- کاربرد اصلی FPGA در ایجاد هسته های پردازشی می باشد .
- مدارهای دیجیتال پیچیده به آسانی در آنها پیاده سازی می شوند .
- تست مدار سریع است .

- برای تولیدات با تیراژ پایین ارزان تمام می شود.
- متناسب با نیاز، تغییرات لازم را در طراحی می توان انجام داد و مجدا FPGA را با ساختار جدید برنامه ریزی نمود.
- قابل برنامه ریزی توسط کاربر است .
- می توان چند تا هسته پردازشی داخل یک FPGA تعریف کرد تا در یک زمان واحد چندتا کار را باهم انجام بدهد مثلًا شما می توانید با یک FPGA معمولی حدود ۲۰۰ تا هسته atmega32 تعریف کنید و ۲۰۰ تا کار را همزمان انجام بدهید .
- میکروکنترلرها و DSP ها و میکرопرոسسورها به صورت سریال دستورالعمل اجرا می کنند و قابلیت پردازش بصورت موازی در آنها وجود ندارد، اما در FPGA قابلیت پردازش موازی وجود دارد و قابلیت انجام عملیات بصورت همزمان را دارد.
- کاربردهای FPGA خیلی تخصصی می باشد و در اکثر موارد به عنوان پردازشگر در مدارات پردازشی استفاده می شود. سرعت بالای FPGA ها آنها را مساعد کارهای پردازشی سنگین مثل پردازش تصویر و پردازش صدا می کند و سرعت این پردازش نسبت به سیستم های دیگر خیلی بالاتر است.
- FPGA ها در پیاده سازی توابع پیجیده دیجیتالی به کار می روند که نیاز به سرعت پردازش بالایی دارد.
- کاهش سخت افزار مورد نیاز و همچنین برنامه نویسی ساده و استاندارد نیز از دیگر مزیت های استفاده از FPGA است.
- سرعت اجرای توابع منطقی در FPGA ها بسیار بالا و در حد نانو ثانیه است.
- امکان تعریف هر یک از پایه های IC به صورت ورودی یا خروجی یا هر دو
- امکان تعریف وضعیت عملکرد هر پایه در هنگام استفاده یا عدم استفاده به عنوان مثال عملکرد HIGH امپدانس(Z) در هنگام عدم استفاده و یا قرار گرفتن در یک وضعیت منطقی صفر یا یک در هنگام عدم استفاده.
- امکان تشخیص تغییرات سطوح یا لبه های پایین رونده یا بالا رونده منطقی اعمال شده به هر پایه.
- امکان تغییر متنابع معماری داخلی با استفاده از سری های Bootable که نقشه معماری آنها در یک حافظه خارجی نگهداری شده و با تغییر ر آدرس برنامه ریزی می توان IC را با معماری جدید Boot کرده و از آن استفاده کرد.
- امکان برنامه ریزی در مدار (ISP) که این قابلیت را به وجود می آورد تا بدون اعمال تغییراتی که سخت افزاری هستند و تنها از طریق پورت برنامه ریزی JTAG، معماری داخلی IC را تغییر داد.
- کاهش حیرت انگیز حجم مدار و مجتمع سازی در ابعادی تنها به مساحت چند سانتی متر مربع.
- کاهش یکسان سازی عناصر طراحی و از میان بردن تمامی مشکلات ناشی از عدم تطابق استاندارد های مختلف.(LS,HC,S,AS,...).
- از میان بردن تمامی نویز های ناشی از وجود قطعات مختلف و مجزا در مدار.

- کاهش چشمگیر توان مصرفی و اتلاف توان.
 - افزایش سرعت پردازش و خطاهای انتشار به دلیل استفاده از فناوری پیشرفته و دستیابی به خطاهای انتشار تا ۴ ns و فرکانس کلاک فرادر از ۱۷۸ مگاهرتز.
 - کار با دو سطح ولتاژ ۵ و ۳.۳ V جهت استفاده از آنها در دستگاه های قابل حمل مانند گوشی های موبایل
 - ضریب اینمی صد درصد به دلیل عدم امکان دستیابی به محتوای داخلی و عدم توان توصیف محتوای داخلی به دلیل انجام ساده سازی و فشرده سازی بسیار پیچیده.
- و بسیاری از قابلیتهای حیرت انگیز دیگر که امکان انجام یک طراحی مجتمع، کم حجم، بهینه و سریع را فراهم می آورد.

[برگرفته از سایت شرکت رهپویان علم و صنعت آوا]

FPGA

شامل شرکت های بزرگی همچون Xilinx، Altera و Actel می باشد که سری های مختلفی از جمله Xilinx شامل Spartan و Altra و Cyclone و Virtex و Flex10k و Stratix و را به بازار عرضه کرده اند. در کشور ما معمولاً با FPGA های شرکت های مذکور کار می شود که از نظر محبوبیت به صورت زیر هستند :

XILINX
Altera
Actel

نگاهی کوتاه به زبان توصیف سخت افزاری VHDL :

نحوه فراخوانی و بکارگیری یک کتابخانه:

LIBRARY ; نام کتابخانه

نحوه فراخوانی و بکارگیری یک package از داخل کتابخانه اش:

USE قسمت مورد نظر. نام پکیج کتابخانه. نام کتابخانه ;

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

کتابخانه های رایج :

IEEE
 STD_LOGIC_1164
 STD_LOGIC_SIGNED
 STD_LOGIC_UNSIGNED
 STD_LOGIC_ARITH
 NUMERIC_STD
STD
 STAN DARD
UNISIM
 VComponents

تعريف موجودیت :

Entity نام مدار **is**

Port ; (معرفی ورودی و خروجی ها)

End ; نام مدار

تعريف معماری طرح :

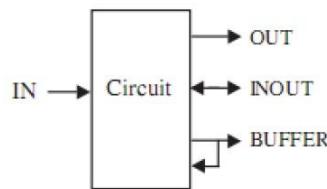
Architecture نام مدار **of** نام معماری **is**

Begin

VHDL کد

End ; نام معماری

: buffer و inout فرق



تعريف سیگنال : پایه یک مژول یا قطعه و یا یک سیم

Signal ; نام و نوع اتصالات

روش نمایش اعداد :

56 , 18 , 4

"10010"

X"5e"

O"47"

جدول زیر انواع داده را نشان می دهد :

Bit	'0', '1'
Bit_Vector	Array of bit
STD_LOGIC	'0', '1', 'X', 'W', 'Z', 'L', 'H', '_'
STD_LOGIC_VECTOR	Array of STD_LOGIC
STD_ULOGIC	'0', '1', 'X', 'W', 'Z', 'L', 'H', 'U', '_'
STD_ULOGIC_VECTOR	Array of STD_ULOGIC
SIGNED	Array of STD_LOGIC
UNSIGNED	Array of STD_LOGIC
Boolean	True and False
Character	7-bit ASCII
integer	-2,147,483,647 to +2,147,483,647
NATURAL	0 to +2,147,483,647
POSITIVE	1 to 2,147,483,647
real	Floating point, min:+1e38 to -1e38
string	Array of characters
time	Hr, min, sec, ms, ns, ps, fs

روش تخصیص مقدار اولیه :

روش اول :

Signal a:bit:='1';

روش دوم :

$A \leq 1'$

▪ پروسس (VHDL ترتیبی) :

Process (لیست سیگنالهایی که با تغییر هر کدام از آنها عبارات داخل پروسس اجرا می شوند)

Begin

عبارات با فرایند اجرای ترتیبی

End process;

▪ برخی عبارات مهم مورد استفاده در پروسس :

▪ : IF ♦

▪ نوع ساده :

If عبارت ۱ **then** عبارت ۲ ;

End if;

اگر عبارت ۱ درست باشد عبارت ۲ انجام می شود در غیر این صورت عملی انجام نشده و از این دستور خارج و به خط

بعد از **end if** می رود.

▪ نوع کلی :

If عبارت ۱ **then** عبارت ۲ ;

Else عبارت ۳ ;

End if;

▪ فرم کلی :

If عبارت ۱ **then** شرط ۱ ;

Elsif عبارت ۲ شرط ۲ ;

.

.

Else عبارت ;

End if ;

▪ : Case ♦

Case شرط **is**

When مقدار شرط ۱ => عبارت ۱ ;

When مقدار شرط ۲ => عبارت ۲ ;

When others => عبارت;

End case;

در دستورات فوق اگر شرط با مقدار هر شرطی برابر باشد عبارت روبروی آن اجرا می شود .

مثال های نمونه : 

❖ نیم جمع کننده :

```
Entity ha is
    Port( a :in std_logic;
            B :in std_logic;
            S:out std_logic;
            C:out std_logic);
End ha;
Architecture half_adder of ha is
begin
    S<=a xor b;
    C<=a and b;
End half_adder;
```

❖ تمام جمع کننده :

➤ روشن ۱

```
Entity Fa is
    port(a,b,Cin:in bit;
          s,Cout:out bit);
End Fa;
Architecture behavioral of Fa is
    Begin
        s<=a xor b xor Cin;
        cout<=(a and b)or(a and Cin)or(b and Cin);
End behavioral;
```

➤ روشن ۲

```
entity fa_1bit is
    port (
        a,b,cin: in STD_LOGIC;
        sum,cout: out STD_LOGIC);
end fa_1bit;

architecture fa_1bit_arch of fa_1bit is
    component ha_1bit
        port(x,y :in std_logic;
              s,c :out std_logic);
    end component;
    signal im1, im2, im3 :std_logic;
begin
    ha1: ha_1bit port map (x=>a, y=>b, s=>im1, c=>im2);
    ha2: ha_1bit port map (x=>im1, y=>cin, s=>sum, c=>im3);
    cout <= im2 or im3;
end fa_1bit_arch;
```

❖ جمع کننده ۴ بیت

```

Entity Fa4bit is
    port(a,b:in bit_vector(3 downto 0);
        Cin:in bit;
        Cout:out bit;
        S:out bit_vector(3 downto 0));
End Fa4bit;
Architecture Structural of Fa4bit is
component Fa is
    port(a,b,Cin:in bit;
        s,Cout:out bit);
End component;
Signal c:bit_vector (3 downto 0);
Begin
    fa0:Fa port map(a(0),b(0),Cin,S(0),c(1));
    fa1:Fa port map(a(1),b(1),c(1),S(1),c(2));
    fa2:Fa port map(a(2),b(2),c(2),S(2),c(3));
    fa3:Fa port map(a(3),b(3),c(3),S(3),Cout);
End Structural;
-----
```

❖ دیکدر



```

Entity decoder is
    Port( a,b :in std_logic;
        M1,m2,m3,m4 :out std_logic;
    );
End decoder;

Architecture behavioral of decoder is
    M1<=(not a) and (not b);
    M2<= a and (not b);
    M3<=(not a) and b;
    M4<=a and b;
End behavioral;
```