بسمه تعالی

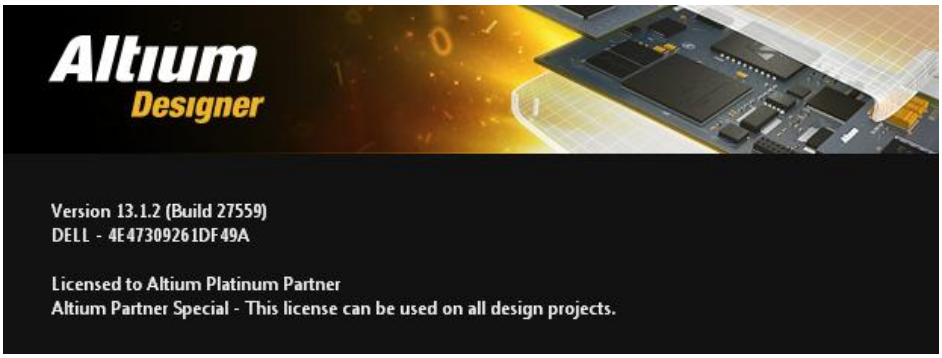
آموزش گام به گام (مبتدی تا پیشرفته)

چند مثال از پروژه های عملی Verilog / VHDL

فهرست مطالب

جزئیات نرم افزارهای مورد استفاده:

Xilinx ISE Design Suite 12.
www.xilinx.com



Altium Designer 13
www.altium.com

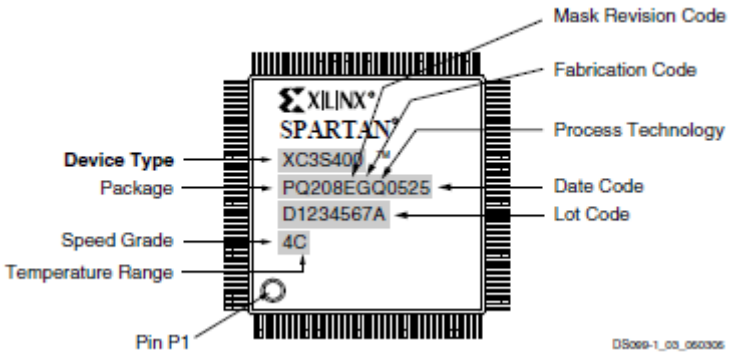
آزمایش ۱

معرفی برد آموزشی و نحوه ی برنامه ریزی آن با یک مثال ساده (چشمک زن)

معرفی قطعه

این برد آموزشی دارای آی سی SPARTAN III XC3S400-4PQ208C با مشخصات زیر می باشد:

- ۴۰۰۰۰۰ سلول منطقی- Logic Cells
- ۵۶ کیلوبیت حافظه توزیع شده
- ۲۸۸ کیلوبیت حافظه بلوکی - Embedded RAM (K)
- ۱۶ ضرب کننده اختصاصی - DSP Multipliers (18x18)
- ۴ عدد Digital Clock Managers – DCM
- دارای ۲۰۸ پین پایه - PQ208 -208 Pin Plastic Quad Flat Pack
- ۱۴۱ پین ورودی/خروجی همه منظوره - I/O Pin Count
- ۶۲ جفت پین ورودی/خروجی تفاضلی - Differential Pair Count



PQ208 Footprint

Left Half of Package (Top View)

XC3S50
(124 max. user I/O)

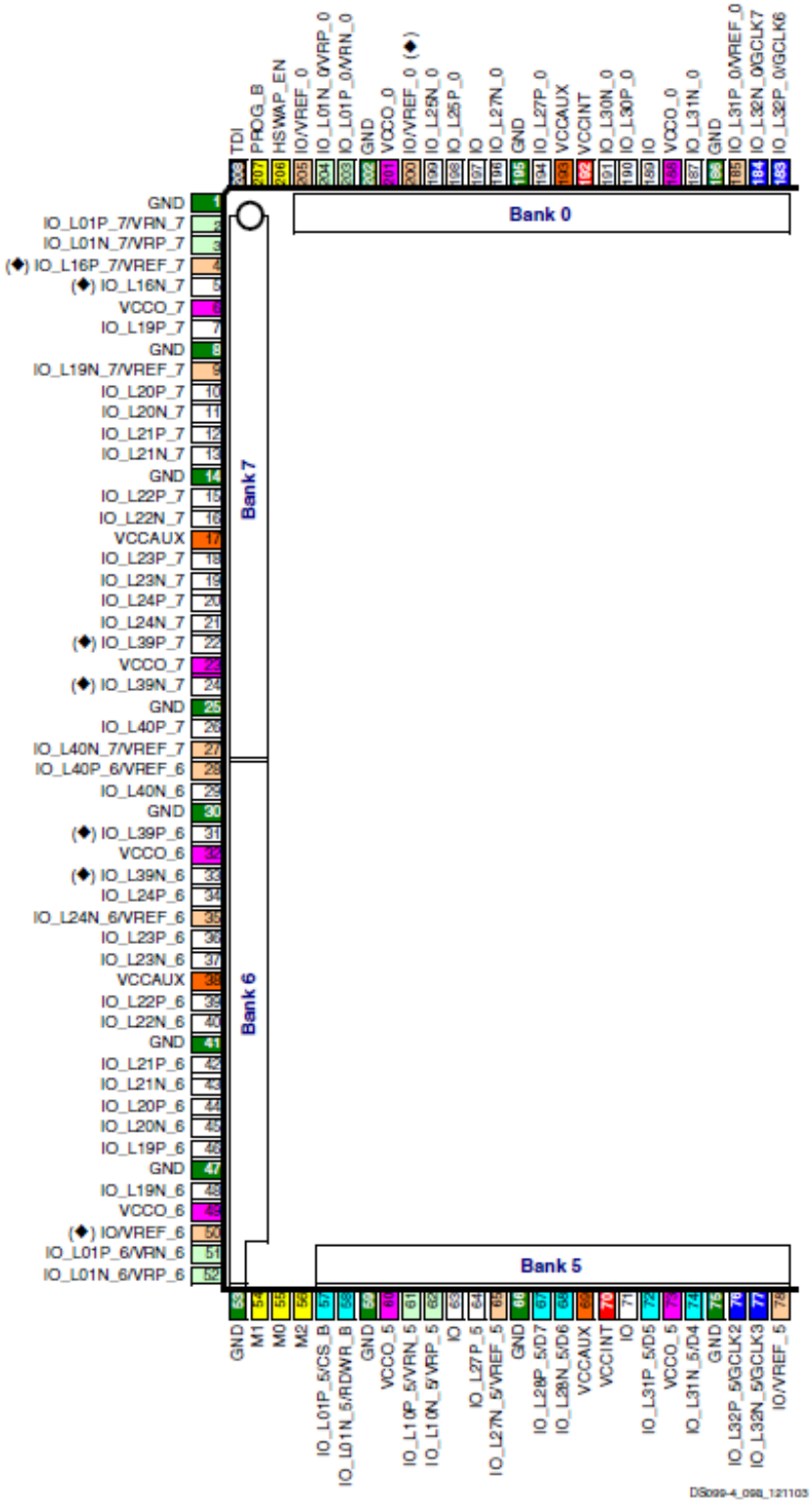
72	**I/O:** Unrestricted, general-purpose user I/O
16	**VREF:** User I/O or input voltage reference for bank
17	**N.C.:** Unconnected pins for XC3S50 (◆)

XC3S200, XC3S400
(141 max user I/O)

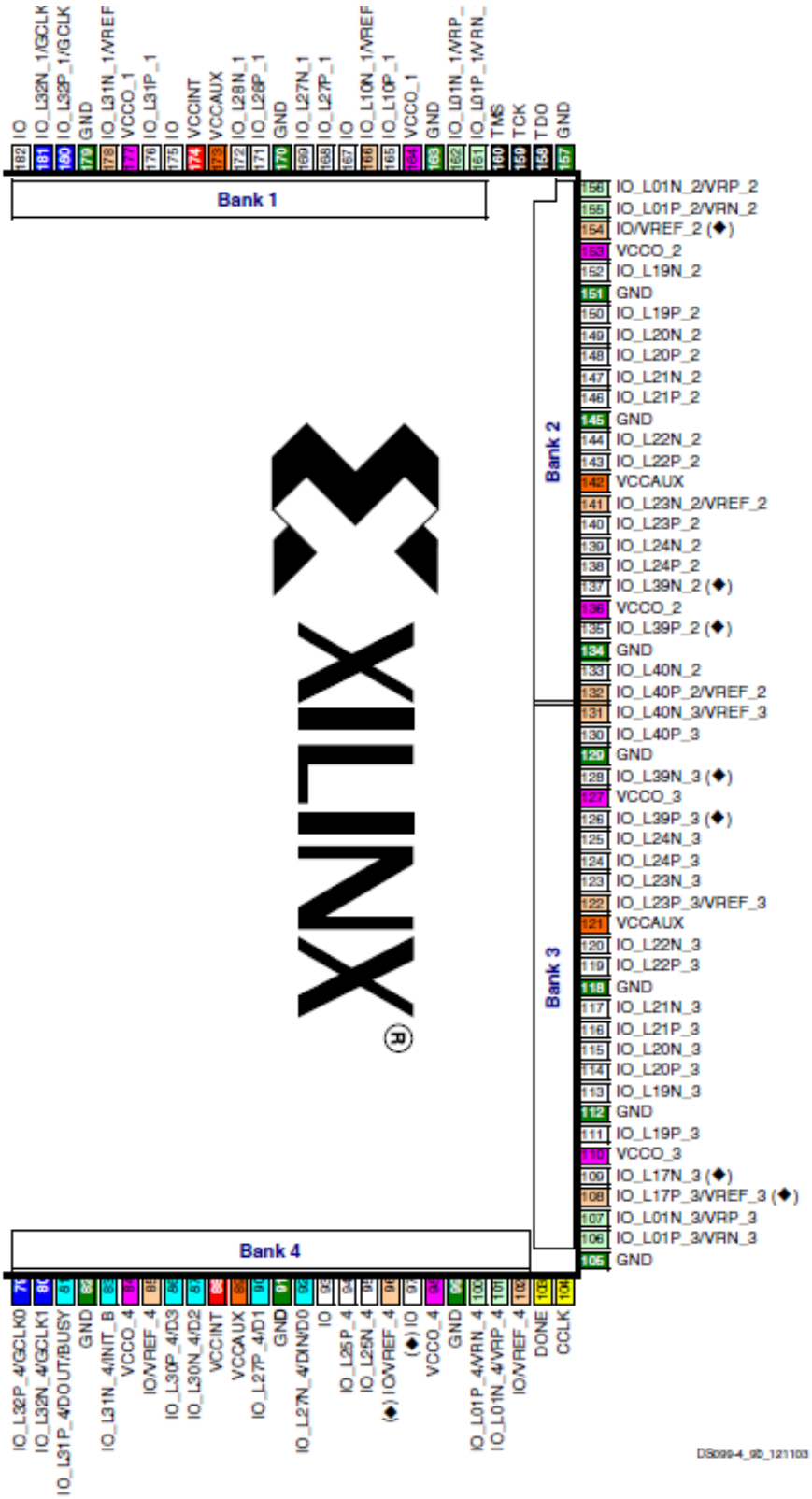
83	**I/O:** Unrestricted, general-purpose user I/O
22	**VREF:** User I/O or input voltage reference for bank
0	**N.C.:** No unconnected pins in this package

All devices

12	**DUAL:** Configuration pin, then possible user I/O
8	**GCLK:** User I/O or global clock buffer input
16	**DCI:** User I/O or reference resistor input for bank
7	**CONFIG:** Dedicated configuration pins
4	**JTAG:** Dedicated JTAG port pins
4	**VCCINT:** Internal core voltage supply (+1.2V)
12	**VCCO:** Output voltage supply for bank
8	**VCCAUX:** Auxiliary voltage supply (+2.5V)
28	**GND:** Ground

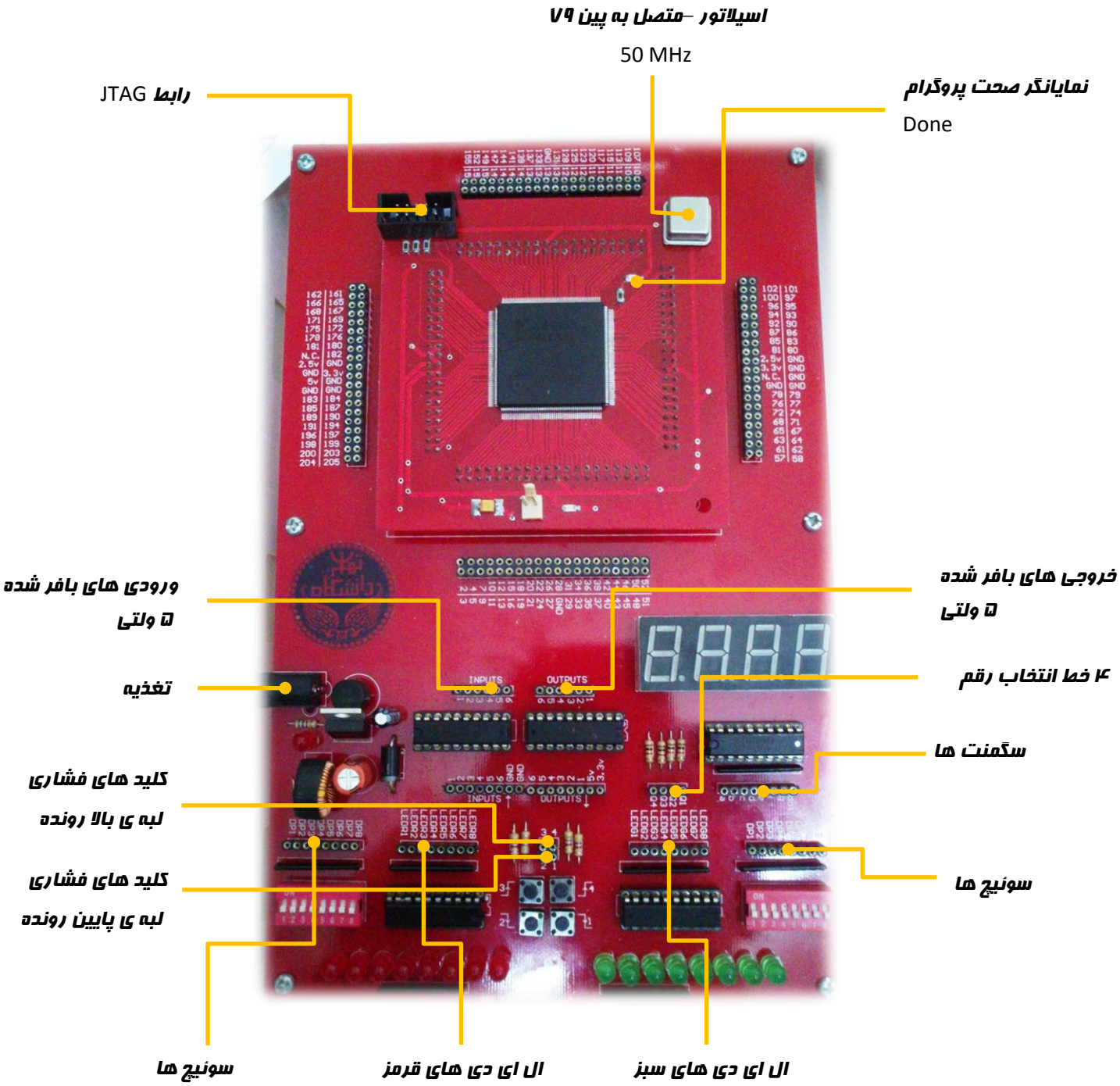


PQ208 Package Footprint (Top View). Note pin 1 indicator in top-left corner and logo orientation.



PQ208 Package Footprint (Top View) Continued

معرفی اجزای برد



شروع به کار

برای آشنایی با مراحل کار، با یک مثال ساده ادامه می دهیم:

در این آزمایش قصد داریم یک چشمک زن ساده داشته باشیم.

مراحل کلی کار بدین صورت است که شما می توانید برنامه ی خود را به هر زبانی (ترجیحا Verilog) که مسلط هستید بنویسید. و پس از ساخت پروژه در نرم افزار Xilinx ISE Design Suite آن را ابتدا سنتز (Synthesize) و سپس امپلیمنت (Implement) و روی قطعه پروگرام می کنیم.

آماده سازی سخت افزار

یکی از ال ای دی های روی برد را (به عنوان مثال) به پایه ی ۲۰ وصل کنید.

آماده سازی نرم افزار

پس از اجرای نرم افزار Xilinx ISE از مسیر زیر، از منوی File، یک پروژه ی جدید می سازیم.

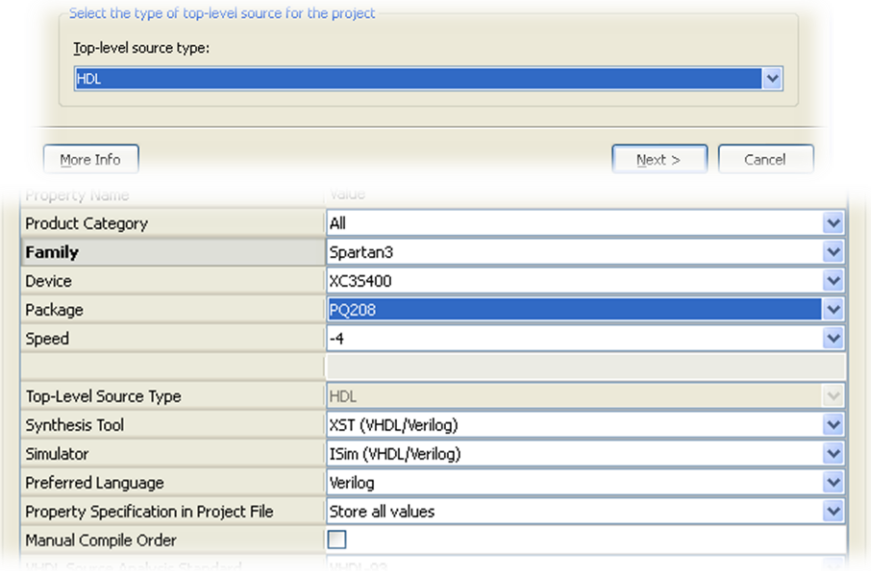
All Programs > Xilinx ISE Design Suite 12.1 > ISE Design Tools > Project Navigator

1.

اجرای برنامه و ساخت پروژه ی جدید

2.

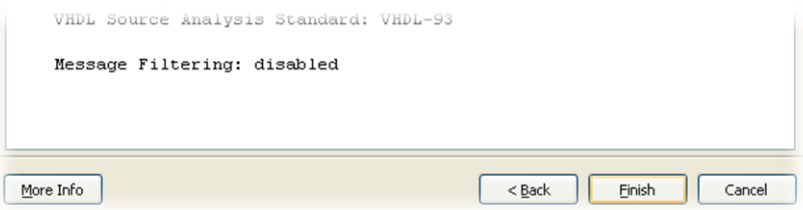
انتخاب نوع Top level



3.

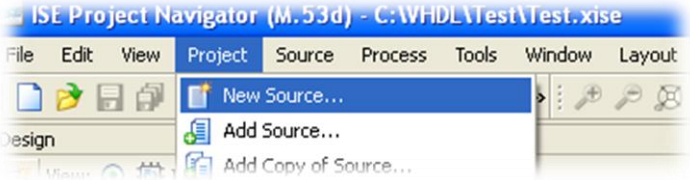
انتخاب دقیق قطعه

4.



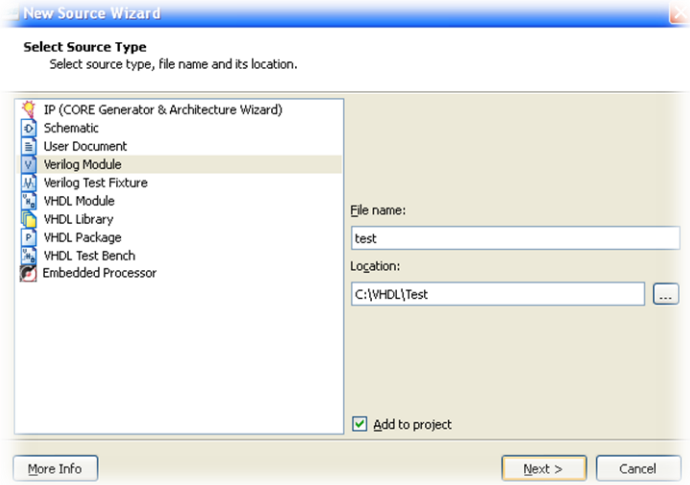
5.

افزودن یک کد جدید



6.

انتخاب نوع کد و اسم



ترجیحا در اسم از فاصله و خط فاصله (-) استفاده نشود.

جهت خوانایی بهتر بجای فاصله از زیر خط (_) استفاده نمایید.

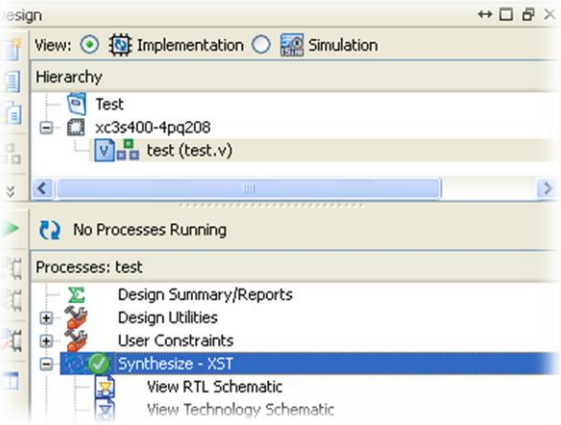
این اسم همان اسم ماژول در ادامه است.

حالا می توانید کد مورد نظر(که در واقع یک مقسم فرکانس است) را بنویسید.

test.v

```
module test(clk ,led);
input clk;
output led;
reg [32:0] counter=32'b0;
reg led;
    always @(posedge clk)
    begin
        counter = counter+1;
        if(counter<50000000)
            if(counter<25000000)
                led=1'b1;
            else
                led=1'b0;
        else
            counter=32'b0;
    end
endmodule
```

همین پروژه را به همراه کد می توانید در پوشه ی 1. Simple LED blinker پیدا کنید.



پس از ذخیره ی کد جهت عیب یابی یک بار سنتز کنید:
در حالی که در پنجره ی Design روی کد کلیک کرده اید، در پنجره ی Process روی Synthesize-XST دبل کلیک کنید تا فرآیند سنتز شروع شود. در صورتی که خطایی در کد وجود نداشته باشد سنتز تا انتها با موفقیت اجرا می شود و علامت تیک سبز رنگ نمایان می شود.

برای انتقال برنامه به روی قطعه باید این برنامه به قطعه مان Map کنیم.
برای این کار کافیست همانند سنتز روی Implement دبل کلیک نمایید.
توجه کنید که سیگنال ها به صورت بهینه (از نظر زمان تاخیر و...) به پین های روی قطعه مسیریابی(Routing) می کند و لزوما به پین مدنظر ما مسیر یابی نمی شود لذا برای تعیین اینکه کدام سیگنال به چه پینی از قطعه می بایست مسیریابی شود باید یک فایل محدودیت (Constrain) ایجاد کنیم. برای اینکار از منوی Project، New Source... را انتخاب کنبد و Implementation Constraint File را با نامی دلخواه ایجاد کنید.

درون این فایل با فرمت خاصی (به صورت زیر) محدودیت های مسیریابی را بیان می کنیم:

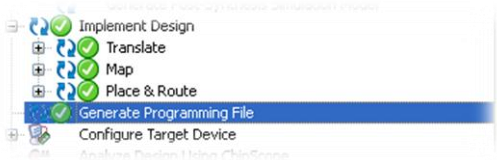
UserConstraintFile.ucf

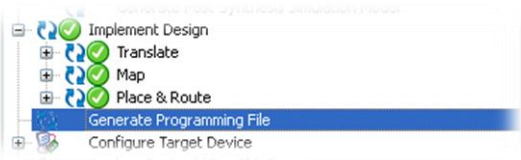
```
NET "clk"  LOC = "p79";
NET "led"  LOC = "p20";
```

همان طور که قبلا اشاره شد، پالس ساعت ورودی (اسیلاتور50 MHz) در برد به پین ۷۹ متصل است.
پایه ی خروجی led نیز (به صورت دلخواه) به پین ۲۰ متصل است.

با وجود این فایل و پس از ذخیره ی آن با دبل کلیک رویImplement سیگنال ها به صورتی که شما خواسته اید مسیریابی می شوند.

پس از گذر با موفقیت از مراحل بالا، کد می تواند به فایل باینری (که به قطعه منتقل می شود.) تبدیل شود.
برای این کار روی Generate Programming File دبل کلیک کنید:



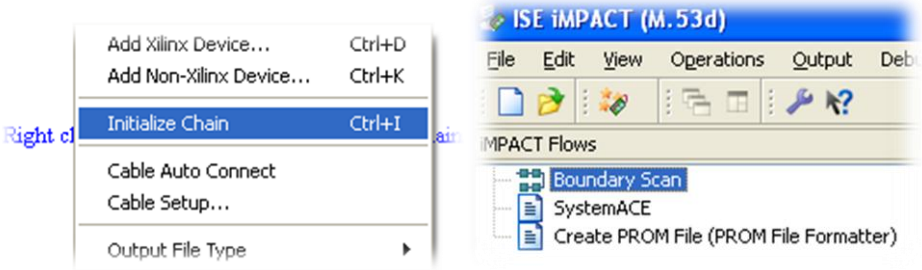


حالا برد آماده ی برنامه ریزی است،

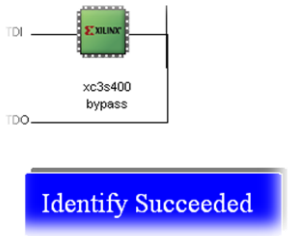
نحوه پراگرام کردن قطعه

برای اینکار ابتدا پروگرامر را به پورت LPT متصل کنید و سوکت JTAG آن را به برد وصل کنبد سپس آداپتور را به برد متصل کنید تا برد روشن شود. اکنون قطعه در حالت ریست قرار دارد.

سپس روی Configure Target Device دبل کلیک کنید تا پنجره ی ISE iMPACT باز شود سپس روی Boundary Scan دبل کلیک کنید. سپس بر روی صفحه کلیک راست کرده وInitialize Chain را برگزینید.

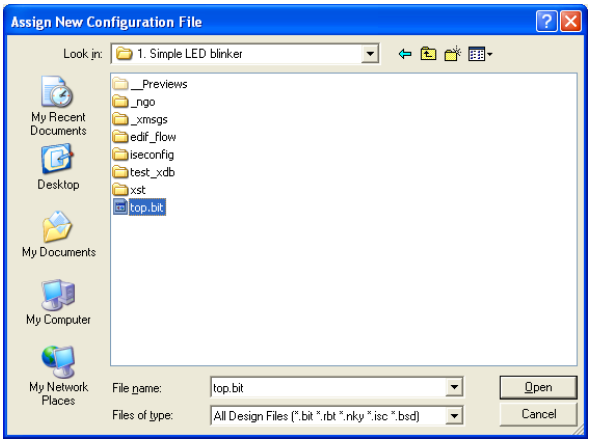


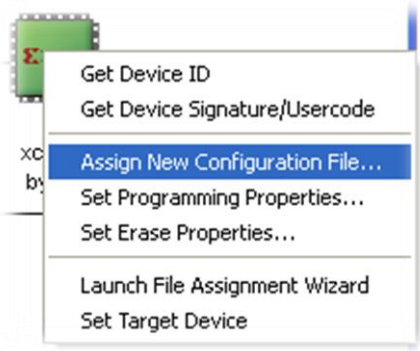
در این حالت پورت ها اسکن می شوند و تمام قطعه های متصل به آن شناسایی می شوند و لیستی از آنها نمایش داده می شود.



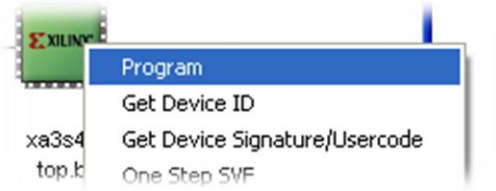
توجه داشته باشید که در اینجا چون تنها یک قطعه متصل است یک مورد آورده شده است.

حالا روی قطعه ی مورد نظر کلیک راست کنید و Assign New Configuration File... کلیک کنید و فایل bit. (که در مرحله ی قبل تولید شده بود) را انتخاب کنید:

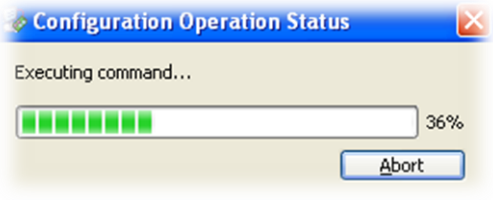




مجدد روی قطعه ی مورد نظر کلیک راست کنید و Program را انتخاب کنید:



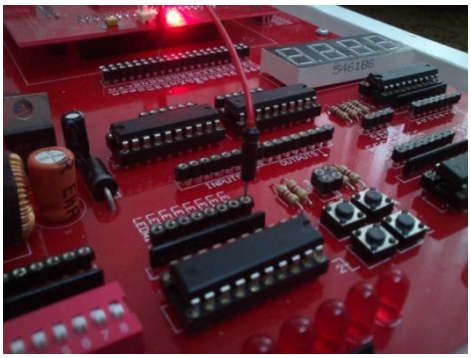
در حال پروگرام:

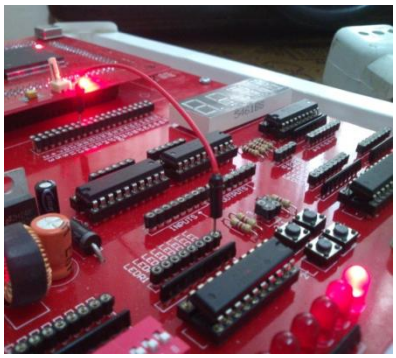


ترجیحا در این حالت برد را ناگهانی خاموش نکنید!

در صورتی که خطایی پیش نیاید پیام موفقیت ظاهر می شود:

تصویری از نتیجه نهایی:





آزمایش ۲

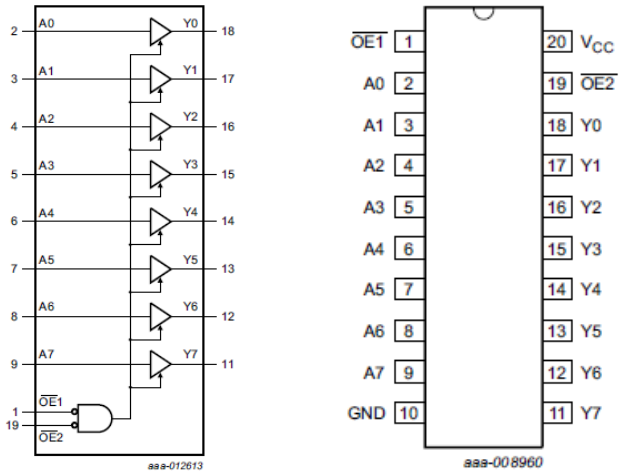
استفاده از بافرهای برد و درایو استپ موتور

معرفی قطعات

بافر:

از آنجا که استپ موتور (یا هر موتور دیگری) نیاز به جریان کشی بالاتری نسبت به میزان حداکثر جریان دهی پایه های FPGA است نیاز به بافرها احساس می شود.

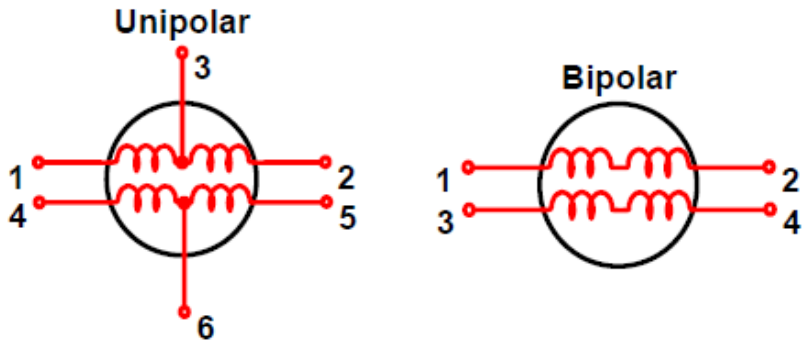
بافر های برد از نوع 74HC541می باشد (که در زیر نشان داده شده است.)



بر روی برد دو پین هدر برای ورودی و خروجی به بافر ها در دسترس است که با اتصال آنها به پین های FPGA می توان با آنها تبادل داده کرد.

استپ موتور:

استپر موتورها به اقسام مختلفی دسته بندی می شوند، که یکی ازآن اقسام، دسته بندی به دو نوع Bipolar و Unipolar است که به صورت ساده ساختمان داخلی آنها در زیر نمایش داده شده است:



انجام آزمایش

- ابتدا نوع موتوری را که در اختیار دارید را (به هر روشی که فراگرفته اید مانند روش اهمتری) تشخیص دهید و سپس کدی برای درایو صحیح آن با نرم افزار مورد نظر بنویسید و مقدمات پیاده سازی برروی برد را مهیا نمایید.

آماده سازی نرم افزار

کد نمونه برای استپرهای Unipolar

top.v

```
module top(clk ,dout,dir);
input clk,dir;
output [3:0] dout ;
reg [3:0] dout ;
reg [32:0] counter=32'b0;
reg pulse=1'b0;
reg [1:0] m=2'b00;
    always @(posedge clk)
    begin
        counter = counter+1;
        if(counter<50000000)
            if(counter<25000000)
                pulse=1'b1;
            else
                pulse=1'b0;
        else
            counter=32'b0;
    end

    always @ (posedge pulse) begin
        if(dir)
            m <= m + 1;
        else
            m <= m - 1;
    end
    always @ (m) begin
     case (m)
      0 : dout = 8;
      1 : dout = 4;
      2 : dout = 2;
      default : dout = 1;
     endcase
    end
endmodule
```

همین پروژه را به همراه کد می توانید در پوشه ی 2. Stepper Motor پیدا کنید.

- تمام اجزای این کد را به طور کامل تحلیل و بررسی نمایید.
- کد مورد نیاز برای درایور استپرهای Bipolar را هم بنویسید.
- کد را طوری تغییر دهید که بتوان سرعت موتور را با سوئیچ ها کنترل کرد.
- کد را طوری تغییر دهید که بتوان در یک زاویه مطلوب با گشتاور مناسب موتور را نگه داشت.

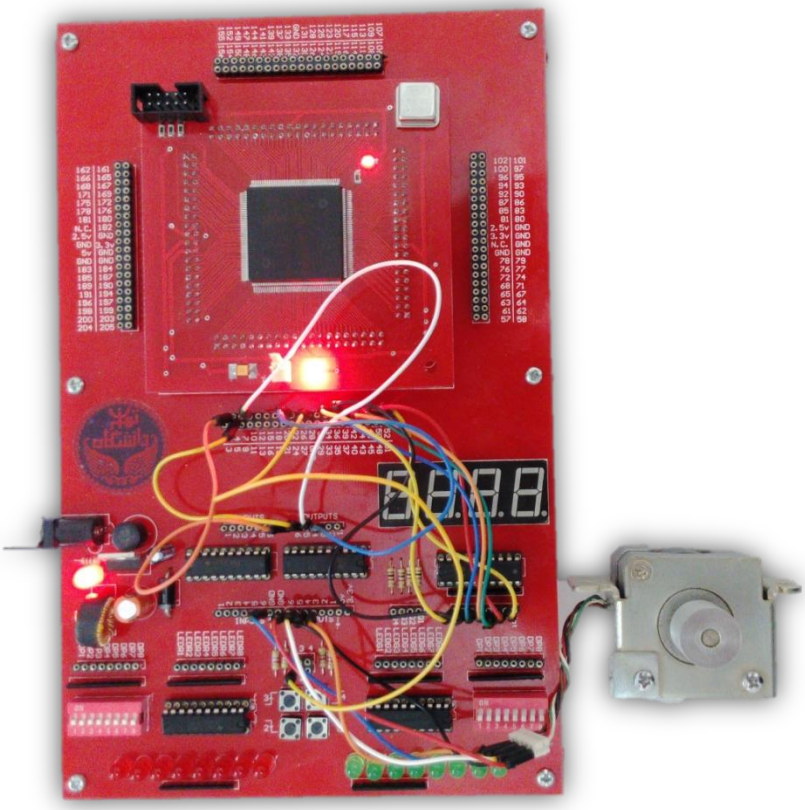
PinAssign.ucf

```
NET "clk"  LOC = "p79";
NET "dir"  LOC = "p28";
NET "dout<0>"  LOC = "p22";
NET "dout<1>"  LOC = "p4";
NET "dout<2>"  LOC = "p7";
NET "dout<3>"  LOC = "p10";
```

آماده سازی سخت افزار

کانکتورهای موتور را به خروجی بافرها وصل کنید و ورودی های متناظر آنها را به پین هایی که در فایل ucf. نوشته اید متصل کنید.

تصویری از نتیجه نهایی:

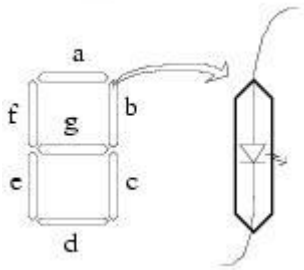


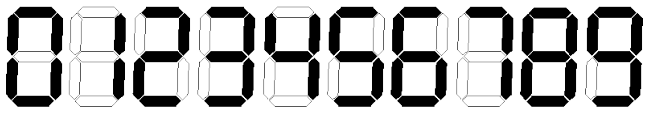
آزمایش ۳

استفاده از نمایشگرهای ۷ قسمتی (7-Segment)

معرفی قطعه

نمایشگر هفت قسمتی یکی از قطعات بسیار مهم در آزمایشگاه است که مشاهده نتایج آزمایش های صورت گرفته را ممکن می سازد. نمایشگر هفت قسمتی در واقع وسیله ای است که از هفت قطعه فتوالکتریک مستقل، مانندLED یا کریستال مایع، تشکیل شده است . قطعات مزبور به فرمی که در شکل زیر نمایش داده شده است در کنار هم قرار گرفته و امکان نمایش اعد اد را مهیا می نمایند. همان طور که در شکل نیز مشاهده می نمایید با فعال نمودن گروه خاصی از اینLED ها اعداد یا حروف خاص و مورد نظر نمایش داده می شود.





به طور کلی نمایشگرهای هفت قسمتی به دو نوع کاتد مشترک و آند مشترک تقسیم می شوند.

در نمایشگر هفت قسمتی کاتد مشترک، کاتد تمامی LED ها به هم متصل شده و به منظور روشن نمودن LED های مورد نظر می بایست ولتاژ آند آنها را نسبت به کاتد افزایش داد. در نمایشگر هفت قسمتی آند مشترک، آند تمامی LED ها به هم متصل شده و به منظور روشن نمودن LED های مورد نظر می بایست ولتاژ کاتد آنها را نسبت به آند کاهش داد.

انجام آزمایش

- برنامه مدار مبدل BCD تک رقمی به نمایشگر هفت قسمتی را توسط نرم افزار مورد نظر اجرا و مقدمات پیاده سازی برروی برد را مهیا نمایید.
- برنامه طوری تغییر دهید که بتوان تا اعداد ۴ رقمی را به صورت مالتی پلکس شده نشان داد.
- برنامه یک تایمر (با دقت ۱۰ میلی ثانیه) را که قابلیت های Stop/Start/Pause/Resume را داشته باشد بنویسید و روی برد پیاده سازی نمایید.

کد نمونه نمایشگر تک رقمی (شمارنده)

top.v

```
module top (clk,up,rst,segment_output);
input up,clk,rst;
output [6:0] segment_output;
reg [6:0] segment_output;
reg [3:0] digit=0;
reg [7:0] debounce=8'b0;

   always @ (posedge clk)
         debounce <= {debounce[6:0], up};

   always @(posedge clk or posedge rst)
    begin
        if(rst)
            digit=0;
        else
            if(&debounce[7:1] & !debounce[0] ==1'b1)
                digit=digit+1;
    end

   always @(digit)
      case (digit)
          4'b0001 : segment_output = 7'b1111001;   // 1
          4'b0010 : segment_output = 7'b0100100;   // 2
          4'b0011 : segment_output = 7'b0110000;   // 3
          4'b0100 : segment_output = 7'b0011001;   // 4
          4'b0101 : segment_output = 7'b0010010;   // 5
          4'b0110 : segment_output = 7'b0000010;   // 6
          4'b0111 : segment_output = 7'b1111000;   // 7
          4'b1000 : segment_output = 7'b0000000;   // 8
          4'b1001 : segment_output = 7'b0010000;   // 9
          4'b1010 : segment_output = 7'b0001000;   // A
          4'b1011 : segment_output = 7'b0000011;   // b
          4'b1100 : segment_output = 7'b1000110;   // C
          4'b1101 : segment_output = 7'b0100001;   // d
          4'b1110 : segment_output = 7'b0000110;   // E
          4'b1111 : segment_output = 7'b0001110;   // F
          default : segment_output = 7'b1000000;   // 0
      endcase

endmodule
```

همین پروژه را به همراه کد می توانید در پوشه ی 3. 7Segment پیدا کنید.

- تمام اجزای این کد و عملکرد آن را به طور کامل تحلیل و بررسی نمایید.
- علت وجود بلوک always اول را توضیح دهید.

PinAssign.ucf

```
NET "clk"  LOC = "p79"  ;
NET "rst"  LOC = "p26"  ;
NET "up"  LOC = "p10"  ;
NET "segment_output<0>"  LOC = "p36"  ;
NET "segment_output<1>"  LOC = "p39"  ;
NET "segment_output<2>"  LOC = "p42"  ;
NET "segment_output<3>"  LOC = "p44"  ;
NET "segment_output<4>"  LOC = "p46"  ;
NET "segment_output<5>"  LOC = "p50"  ;
NET "segment_output<6>"  LOC = "p52"  ;
```

آماده سازی سخت افزار

هر یک از سگمنت های نمایشگر را به ترتیب به پایه هایی که در فایل ucf. بیان شده متصل کنید. و همچنین برای سیگنال های reset و up -با توجه به کاربرد خود- به دکمه های فشاری حساس به لبه ی بالا رونده یا پایین رونده متصل نمایید.

آزمایش ۴

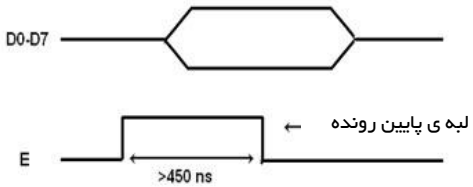
ارتباط با نمایشگرهای کریستال مایع، کاراکتری (LCD)

معرفی قطعه

LCD ۲×۱۶ یک LCD متنی است و دارای دو سطراست که هرسطر دارای ۱۶ مکان برای نمایش کاراکتر می باشد.

- VSS و VDD تغذیه ی LCD را مهیا می کند
- VE ولتاژی بین زمین و تغذیه است که کنتراست تعیین می کند. می‌توان با اتصال یک پتانسیومتر (حدود ۱۰K) به این پایه شدت روشنایی آن را کنترل نمود.
- پایه‌های Data 0 تا Data 7 برای ورود اطلاعات می باشند که می‌تواند داده (کد اسکی کاراکتری که می‌خواهیم نمایش داده شود) یا دستور (عملی که باید انجام شود) باشد.
- برای این که به LCD بفهمانیم که می‌خواهیم داده بفرستیم خط RS را برابر'1' و برای فرستادن دستور این خط را برابر'0' قرار می‌دهیم.

از پایه Data7 می‌توان برای خواندن وضعیت کنونی LCD، که این که آیا مشغول انجام دادن کاری می‌باشد یا خیر استفاده نمود. برای این کار باید خط R/W که در حالت نوشتن (در حالت فرستادن داده ودستور) باید آن را برابر '0' قرار داد، در این حالت (حالت خواندن) آن را برابر '1' قرار دهیم. پس از آن باید Data7 را بخوانیم. اگر Data7 برابر '1'بود، LCD مشغول انجام کار قبلی می‌باشد (Busy می‌باشد) و اطلاعات (دستور یا داده) تازه را نمی‌پذیرد. ولی اگر Data7= '0' بود می‌توان اطلاعات جدید را فرستاد.



در هر حالت برای فرستادن داده یا دستور پس از اعمال اطلاعات به Data 0 تا Data 7 باید مطابق شکل روبرو یک لبه پایین رونده به پایه Enable اعمال کرد تا اطلاعات ترتیب اثر داده شوند. همچنین برای خواندن وضعیت نیز باید یک لبه پایین رونده به Enable اعمال نمود. حداقل مدت زمان '1' بودن پالس اعمال شده به پایه Enable باید 450 ns باشد.

برای تعریف نوع LCD باید دستور0x30 برای LCD های یک سطری، یا دستور0x38 برای LCD های دو سطری فرستاد. پس از آن دستور 0x0E که LCD و مکان‌نما را روشن می‌کند باید فرستاده شود و همچنین باید با فرستادن دستور 0x01 صفحه LCD را پاک کرد.

اگر خواستیم مکان‌نما را یک خانه به سمت راست حرکت دهیم از دستور 0x06 استفاده می‌کنیم و برای حرکت به چپ از دستور0x04 بهره می‌گیریم.

در زیر جدولی از کل دستورات موجود آورده شده است:

Code (Hex)	Command to LCD Instruction Register
1	Clear Display screen
2	Return home
4	Decrement cursor (Shift cursor to left)
6	Increment cursor (Shift cursor to Right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display on, cursor off
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
0C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 Matrix

لیستی از کاراکترهای قابل نمایش بر روی LCD به همراه کد اسکی:

Dec	Hx		Dec	Hx		Dec	Hx		Dec	Hx		Dec	Hx		Dec	Hx		Dec	Hx		Dec	Hx	
16	10		48	30	0	80	50	P	112	70	p	144	90	É	176	B0	°	208	D0		240	F0	
17	11		49	31	1	81	51	Q	113	71	q	145	91	æ	177	B1	±	209	D1		241	F1	
18	12		50	32	2	82	52	R	114	72	r	146	92	Æ	178	B2	²	210	D2		242	F2	
19	13		51	33	3	83	53	S	115	73	s	147	93	ô	179	B3	³	211	D3		243	F3	
20	14		52	34	4	84	54	T	116	74	t	148	94	ö	180	B4	´	212	D4		244	F4	
21	15		53	35	5	85	55	U	117	75	u	149	95	ò	181	B5	½	213	D5		245	F5	
22	16		54	36	6	86	56	V	118	76	v	150	96	û	182	B6	¼	214	D6		246	F6	
23	17		55	37	7	87	57	W	119	77	w	151	97	ù	183	B7	×	215	D7		247	F7	
24	18		56	38	8	88	58	X	120	78	x	152	98	ÿ	184	B8	÷	216	D8		248	F8	
25	19		57	39	9	89	59	Y	121	79	y	153	99	Ö	185	B9	≤	217	D9		249	F9	
26	1A		58	3A	:	90	5A	Z	122	7A	z	154	9A	Ü	186	BA	≥	218	DA		250	FA	
27	1B		59	3B	;	91	5B	[	123	7B	{	155	9B	¢	187	BB	«	219	DB		251	FB	
28	1C		60	3C	<	92	5C	\	124	7C	|	156	9C	£	188	BC	»	220	DC		252	FC	
29	1D		61	3D	=	93	5D	]	125	7D	}	157	9D	¥	189	BD	≠	221	DD		253	FD	
30	1E		62	3E	>	94	5E	^	126	7E	~	158	9E	₧	190	BE	√	222	DE		254	FE	
31	1F		63	3F	?	95	5F	_	127	7F	⌂	159	9F	¿	191	BF	⌐	223	DF		255	FF	
32	20		64	40	@	96	60	`	128	80	Ç	160	A0	á	192	C0	∟	224	E0				
33	21	!	65	41	A	97	61	a	129	81	ü	161	A1	í	193	C1	⌡	225	E1				
34	22	"	66	42	B	98	62	b	130	82	é	162	A2	ó	194	C2	∞	226	E2				
35	23	#	67	43	C	99	63	c	131	83	â	163	A3	ú	195	C3	◸	227	E3				
36	24	$	68	44	D	100	64	d	132	84	ä	164	A4	¢	196	C4	↲	228	E4				
37	25	%	69	45	E	101	65	e	133	85	à	165	A5	£	197	C5	↑	229	E5				
38	26	&	70	46	F	102	66	f	134	86	å	166	A6	¥	198	C6	↓	230	E6				
39	27	'	71	47	G	103	67	g	135	87	ç	167	A7	₧	199	C7	→	231	E7				
40	28	(	72	48	H	104	68	h	136	88	ê	168	A8	ƒ	200	C8	←	232	E8				
41	29	)	73	49	I	105	69	i	137	89	ë	169	A9	⌐	201	C9	⌈	233	E9				
42	2A	*	74	4A	J	106	6A	j	138	8A	è	170	AA	ª	202	CA	¬	234	EA				
43	2B	+	75	4B	K	107	6B	k	139	8B	ï	171	AB	½	203	CB	⌊	235	EB				
44	2C	,	76	4C	L	108	6C	l	140	8C	î	172	AC	¼	204	CC	⌋	236	EC				
45	2D	-	77	4D	M	109	6D	m	141	8D	ì	173	AD	¡	205	CD	▪	237	ED				
46	2E	.	78	4E	N	110	6E	n	142	8E	Ä	174	AE	«	206	CE	▒	238	EE				
47	2F	/	79	4F	O	111	6F	o	143	8F	Å	175	AF	»	207	CF	▒	239	EF				

ارتباط ساده با LCD کاراکتری به زبان VHDL

lcd.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity lcd is
Port (data : out std_logic_vector(7 downto 0);
      rs : out std_logic;
      rw : out std_logic;
      e : out std_logic;
      bl : out std_logic;
      clk : in std_logic;
      reset : in std_logic);
end lcd;

architecture Behavioral of lcd is
constant number_of_data : integer := 27;
constant delay_for_e : integer := 22
constant delay_between_data : integer := 100000;
type two_dimen_array is array (0 to number_of_data - 1) of std_logic_vector (8
downto 0);

constant table : two_dimen_array :=("000111000", "000001100", "000000001",
"101001000", "101100101", "101101100","101101100", "101101111", "110100000",
"101010110", "101001000", "101000100","101001100", "100100001", "011000000",
"101000001", "101101100", "101101001","110100000", "101000100", "101100001",
"101101101", "101101001", "101110010","101100011", "101101000", "101111001");

    type state is (start, write, delay, stop);
    signal current : state;

    signal part : integer;
    begin

    process (clk, reset)
    variable i, count : integer;
    begin
        if reset = '1' then
            current <= start;
            rs <= '0';
            rw <= '0';
            e <= '0';
            bl <= '1';
            count := 0;
            i := 0;
            part <= 0;
        elsif(rising_edge(clk))then
            case current is
            when start =>
                data <= table(i)(7 downto 0);
                rs <= table (i)(8);
                i := i + 1;
                if i < (number_of_data + 1) then
                    current <= write;
                else
                    current <= stop;
                end if;
            when write =>
                if part = 0 then
```

```
                e <= '1';
                count := delay_for_e;
                part <= part + 1;
                current <= delay;
            elsif part = 1 then
                e <= '0';
                count := delay_between_data;
                current <= delay;
                part <= part + 1;
            else
                current <= start;
                part <= 0;
            end if;
        when delay =>
            count := count - 1;
        if count = 0 then
            current <= write;
        end if;
        when stop =>
        end case;
    end if;
end process;

end Behavioral;
```

همین پروژه را به همراه کد می توانید در پوشه ی 4. LCD 2x16_VHDL پیدا کنید.

- تمام اجزای این کد و عملکرد آن را به طور کامل تحلیل و بررسی نمایید.
- نحوه ی محاسبه ی مقدار ثوابت موجود در برنامه را توضیح دهید.
- متن دلخواه دیگری را بر روی LCD نمایش دهید.
- برای متنی دلخواه با دستوراتی که در جدول آورده شده است افکت های حرکتی جالبی درست کنید.
- کد یک ساعت دیجیتال که شامل نمایش ساعت/دقیقه/ثانیه باشد را بنویسید.

ucf.ucf

```
NET "clk"  LOC = "p79";
NET "reset"  LOC = "p3";
NET "e"  LOC = "p51";
NET "rw"  LOC = "p50";
NET "rs"  LOC = "p52";
NET "data<0>"  LOC = "p45";
NET "data<1>"  LOC = "p37";
NET "data<2>"  LOC = "p33";
NET "data<3>"  LOC = "p27";
NET "data<4>"  LOC = "p21";
NET "data<5>"  LOC = "p16";
NET "data<6>"  LOC = "p11";
NET "data<7>"  LOC = "p5";
```

آماده سازی سخت افزار

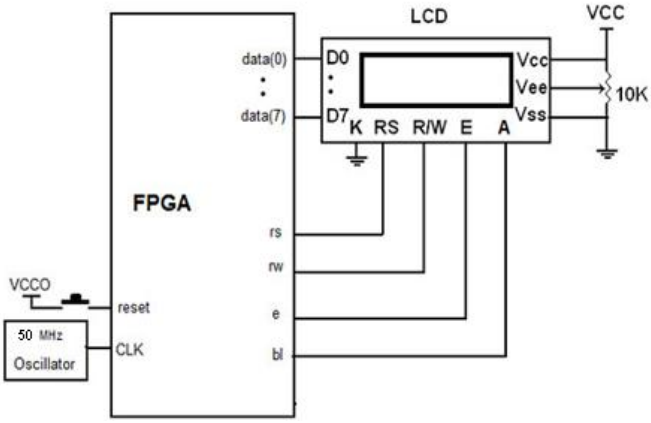
توصیه می شود برای ارتباط راحت با ال سی دی آن را روی بردبورد متصل کنید.

پین های دیتای ال سی دی را به پایه هایی که در فایل ucf. بیان شده متصل کنید.

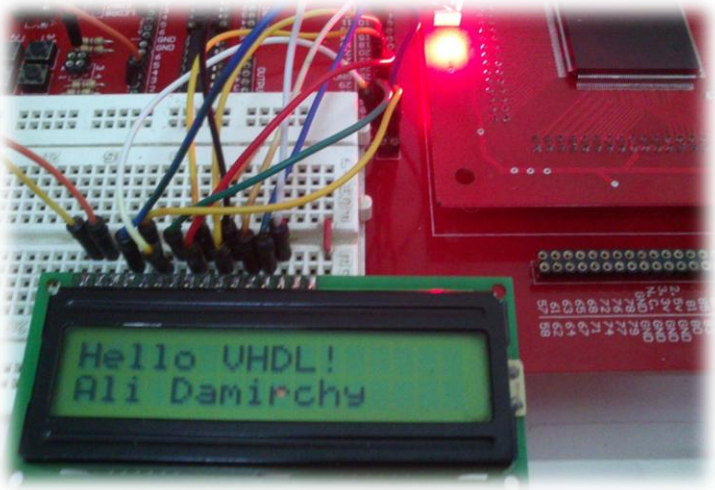
اگر مایل هستید که نور پس زمینه را هم تحت کنترل داشته باشید می توانید آن را به خروجی یکی از بافرها وصل کنید.

توجه کنید که LCD های موجود در بازار 5 ولتی هستند. ولی بعلت اینکه سطح ولتاژ ۳.۳ ولت در منطق TTL نیز سطح منطقی یک محسوب می شود مشکلی در ارتباط با FPGA نداریم و مستقیما آن را به FPGA متصل می کنیم ولی توجه کنید که پایه ی VDD ال سی دی -که همان تغذیه آن است- را به ۵ ولت متصل کنید. این بدان معناست که کنترلر LCD با ۵ ولت روشن می شود ولی برای ارتباط با FPGA با منطق ۳.۳ ولتی هم کار می کند.

مدار زیر را ببندید :



تصویری از نتیجه نهایی:



آزمایش ۵

کار با FPGA در نرم افزار Altium با یک مثال ساده (چشمک زن)

مقدمه آزمایش

شرکت های سازنده ی قطعه همچون Altera یا Xilinx برای قطعه ی خود نرم افزار مختص به خود را ارائه می دهند مانند Quartus و ISE که در آنها می توان برنامه ی نوشته شده ی خود را برای قطعه مان قابل فهم کنیم ولی همان طور که در آزمایش های قبلی هم دیدیم برای برپایی یک ارتباط با یک وسیله ی دیگر چنان درگیر ریزه کاری ها و پیچیدگی های اولیه ی کدنویسی می شویم که شاید این باعث دلسرد شدن از ادامه ی کار بشود. اجازه دهید با یک مثال موضوع را بهتر بیان کنم، مثلا اگر بخواهیم برای نمایش دوخط متن بر روی LCD چندین خط کد بنویسم تصور کنید که برای یک پروژه ی کمی بزرگتر چه وضعیتی پیش خواهد آمد!

اینجاست که می توانیم در طرح خود از کدهای آماده نوشته شده توسط برنامه نویسان حرفه ای استفاده کنیم (هسته ها) و طرح خود را با بهره بری از آنها توسعه دهیم و بیشتر تمرکز خود را بر روی اجزای کاربردی تر و مهمتری متمرکز کنیم.

شرکت Altium نیز به عنوان یکی از پیشگامان این عرصه نرم افزار Altium Designer را تولید کرده است. این نرم افزار قابلیت های فراوانی دارد که می توان به طراحی PCB، Embedded OS، FPGA اشاره کرد. در این برنامه در محیط FPGA می توان سیستم های گسترده ای را طراحی و کد نویسی کرد و حتی در همان محیط قطعه را پروگرام کرد.

یکی از معروف ترین هسته ای که این نرم افزار امکان پیاده سازی روی قطعه را می دهد، هسته ی پردازنده ی TSK3000 است. مزیت اولیه این هسته، پشتیبانی از زبان C می باشد که به مراتب زبان ساده تری نسبت به VHDL یا Verilog است.

در این آزمایش قصد معرفی این نرم افزار با یک مثال ساده داریم.

برای انجام پروژه های FPGA با Altium نیازمند چند پیشنیازهایی هستیم که در زیر توضیح می دهم:

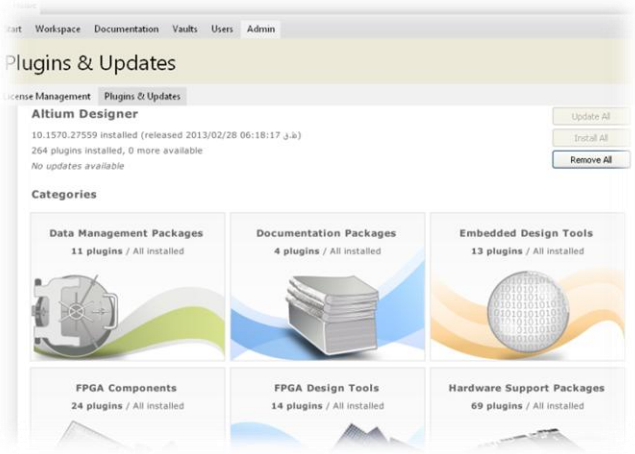
- برای هر نوع قطعه ای که استفاده می کنید باید ابزار کامپایلر و سنتز- مسیریابی خاص آن قطعه را نیز روی سیستم خود نصب داشته باشید. مثلا ISE یا Quartus

ⓘ ترجیحا قبل از نصب Altium ابزار مناسب را نصب کنید.

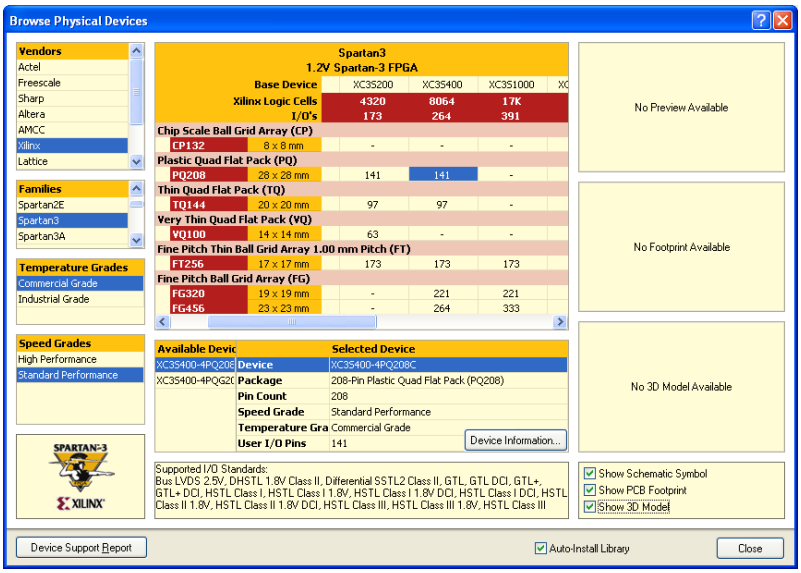
- پس از نصب نرم افزار Altium Designer چند پلاگین دیگر هم باید نصب شوند تا شناسایی قطعه با مشکل مواجه نشود. برای اینکار پس از اجرای برنامه در حالی که CD نصب آلتیوم در درایو قرار دارد، به صفحه ی Home بروید. (از منوی View > Home را بزنید.) و از سربرگ Admin زیر مجموعه Plugin & Updates را انتخاب کنید. از این محل می توانید پلاگین ها را مدیریت کنید.

ⓘ چون احتمال اینکه این صفحه در آینده نمایان نشود (به علت های لود نکردن سی دی، کرک و ...) توصیه می شود تمام این پلاگین ها را در همان وحله ی اول پس از نصب نرم افزار، نصب کنید.

برای نصب پلاگین ها روی Install All کلیک کنید و پس مدت کمی طولانی، پلاگین ها نصب میشوند.



ⓘ برای چک کردن درستی مراحل بالا از منوی View > Device View را بزنید و سپس از منوی Tools > Browse Physical Devices را انتخاب کنید. در پنجره ای که باز می شود باید بتوانید قطعه ی مورد نظرتان را پیدا کنید. در غیر اینصورت نمی توانید مراحل بعدی را ادامه دهید.

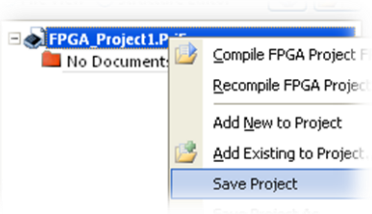


آماده سازی سخت افزار

یکی از ال ای دی های روی برد را (به عنوان مثال) به پایه ی ۲۰ وصل کنید.

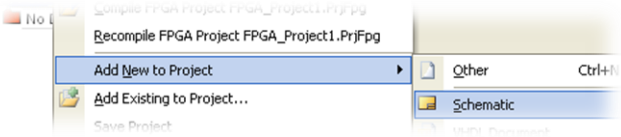
آماده سازی نرم افزار

پس از طی مراحل قبلی حالا شما می توانید در Altium Designer پروژه ی FPGA بسازید. برای اینکار از منوی File > New > Project > FPGA Project را برگزینید. این پروژه را با رعایت قوانین نام گذاری در محلی مناسب ذخیره کنید.



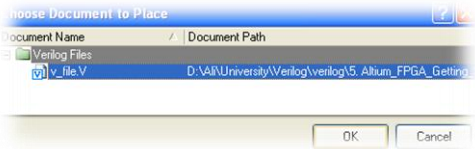
در این آزمایش ساده ترین روش کار(شماتیکی) با FPGA در Altium را ادامه می دهیم تا اساس کلی بهتر بیان شوند.

پس به این پروژه (که خالی است) یک سند شماتیکی اضافه کنید. (روی پروژه کلیک راست کنید و از زیر مجموعه یAdd New to Project گزینه ی Schematic را بزنید. و این شماتیک را نیز ذخیره کنید.

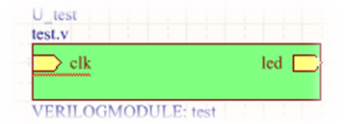


و مجدد روی پروژه کلیک راست کنید و Add Existing to project... را انتخاب کرده و فایل وریلاگی که در آزمایش ۱ (چشمک زن- مقسم فرکانس) نوشته بودید را انتخاب کنید.

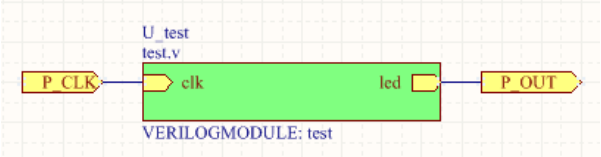
به شماتیک باز گردید و از منوی Design > Create Sheet Symbol From Sheet or HDL را بزنید و فایل ویریلاگی که اضافه کرده اید را انتخاب کنید و OK را بزنید.



سپس یک بلوک سبز رنگ که به موس چسبیده ظاهر می شود. این بلوک را در یک جای مناسب در شماتیک قرار دهید.



از منوی Place > Port را انتخاب کنید و پورت های این بلوک را نام گذاری کنید.



یک بار پروژه را کامپایل کنید تا اگر ایرادی داشت در پنجره ی Messages نمایان شود. (از منوی Project > Compile Document Sheet1.SchDoc را بزنید)

حالا برای تخصیص پایه ها (پورت ها) به پین های FPGA، روی پروژه کلیک راست کرده و از زیر مجموعه ی Add New to Project مورد Constraint File را بزنید. ابتدا باید نوع و اسم قطعه ی مورد نظرتان را معرفی کنید و سپس پین پین ها را تخصیص دهید. برای این کار در هنگامی که Constraint File را که اضافه کرده بودید را باز کردید، از منوی Design > Add/Modify Constraint... > Part... را بزنید. و از پنجره ی Choose Physical Device قطعه ی مورد نظرتان را پیدا کنید و روی OK کلیک کنید.

به عنوان مثال همان طور که در آزمایش اول معرفی شد برد آموزشی که در اختیار شماست دارای قطعه SPARTAN III XC3S400-4PQ208C می باشد، پس از لیست Vendor مورد Xilinx را انتخاب کنید و از قسمت Families مورد Spartan3 را انتخاب کنید و و از سطرPQ208 ، ستون XC3S400 را انتخاب کنید. همان طور که می بینید عدد ۱۴۱ نمایان تعداد I/O های قطعه ی ماست سپس روی OK کلیک کنید تا خط زیر –که معرف قطعه ی ماست- به فایل Constraint اضافه شود.

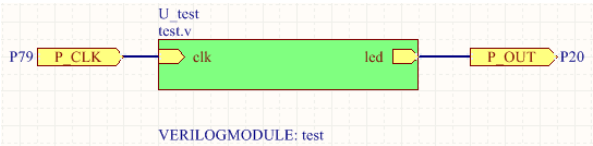
```
Record=Constraint | TargetKind=Part | TargetId=XC3S400-4PQ208C
```

مجدد از منوی Design > FPGA Signal Manager... را بزنید، در این پنجره لیستی از تمام پورت هایی که باید به پین ها تخصیص پیدا کنند نمایش داده می شود. دکمه ی Assign Unconstrained Signal را بزنید تا به هر سیگنالی یک شماره پایه داده شود و OK را بزنید. در پنجره ی بعد به ترتیب روی دکمه Validate Changes و سپس Execute Changes کلیک کنید و بعد از تمام شدن عملیات پنجره را ببندید. همان طور که مشاهده می کنید دو خط دیگر به فایل Constraint اضافه شده که با تغییر شماره پایه می توانید پورت ها را به هر پین دلخواه دیگری تخصیص دهید.

```
Record=Constraint | TargetKind=Port | TargetId=P_CLK | FPGA_PINNUM=P79
Record=Constraint | TargetKind=Port | TargetId=P_OUT | FPGA_PINNUM=P20
```

حالا این فایل را نیز در کنار پروژه خود ذخیره کنید.

و به شماتیک باز گردید و از منوی Design > Synthesize را بزنید. در صورتی که مراحل قبلی را درست انجام داده باشید کنار هر پورت عددی نمایان می شود که نمایانگر شماره پین تخصیص یافته به آن پورت میباشد.



برای سنتز بهینه تر می توانید از سنتز XST -که خاص Xilinx است- استفاده کنید. برای اینکار از منوی Project > Project Option... کلیک کنید. در پنجره ی باز شده، از سربرگ Synthesis و قسمت Synthesizer مورد XST Synthesizer را انتخاب کنید. (توجه کنید که این ابزار باید قبلا نصب شده باشد.)

جهت مدیریت فایلهای محدودیت از منوی Project >Configuration Manager... را انتخاب کنید.

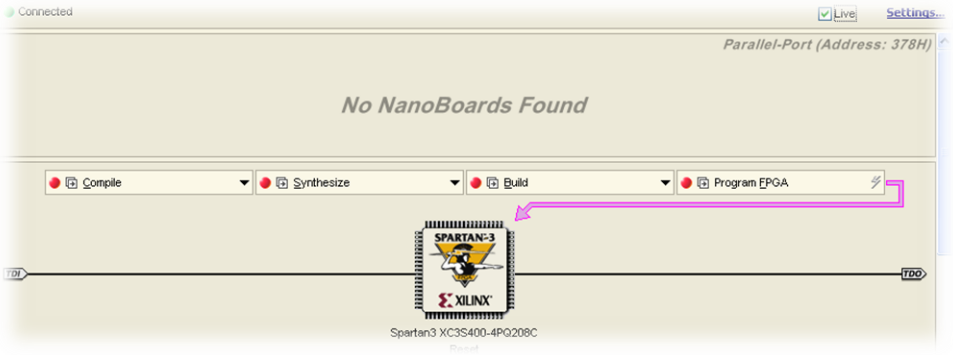
حالا برد آماده ی برنامه ریزی است.

نحوه پراگرام کردن قطعه

برای اینکار ابتدا پروگرامر را به پورت LPT متصل کنید و سوکت JTAG آن را به برد وصل کنبد سپس آداپتور را به برد متصل کنید تا برد روشن شود. اکنون قطعه در حالت ریست قرار دارد.

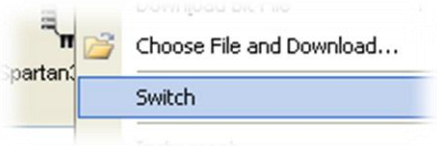
از منوی View > Device View را بزنید تا وارد محیط Device view بشوید. حالا می بایست قطعه ی شما شناسایی بشود.

توجه کنید که تیک Live خورده باشد.

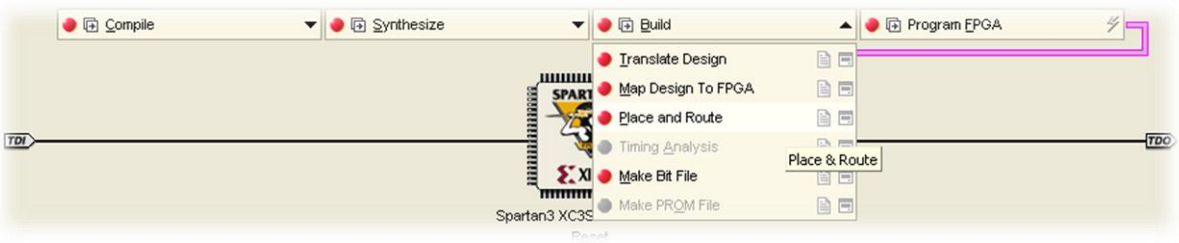


اگر قطعه اشتباهی تشخیص داده شده است مورد دقیق آن را به روش زیر تصحیح کنید:

روی قطعه کلیک راست کنید و از زیر مجموعه ی Switch مورد دقیق و صحیح آن را انتخاب نمایید.



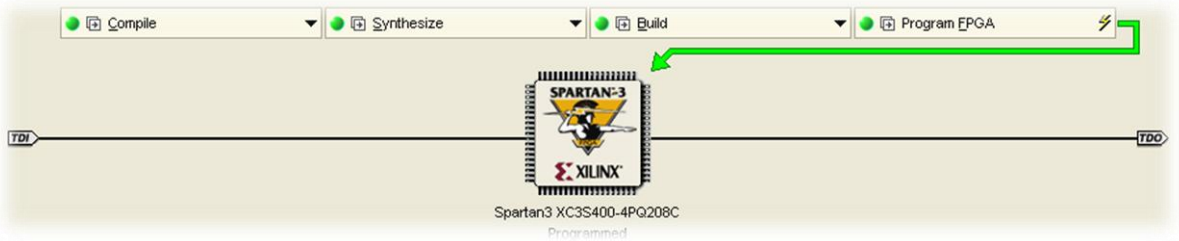
همان طور که به صورت شماتیکی و دیاگرامی هم در اینجا مشخص است این پروژه می بایست ابتدا کامپایل سپس سنتز ، جانمایی، مسیریابی،تولید فایل برنامه و در انتها پروگرام قطعه انجام شود. (مانند همان سیری که در آزمایش اول در نرم افزار Xilinx ISE داشتیم.)



با کلیک روی هر مرحله عملیات مراحل قبل به انظمام همان مرحله اجرا می گردد.

پس روی Program FPGA کلیک کنید تا تمام فرآیند از ابتدا تا انتها اجرا شود.

در صورتی که خطایی رخ ندهد عملیات تا انتها ادامه می یابد و تمام مراحل به رنگ سبز در می آید.



آزمایش ۶

کار با FPGA بدون دانش برنامه نویسی در Altium (شمارنده جانسون)

مقدمه

در این آزمایش جهت آشنایی بیشتر با Altium و ترکیب شماتیک و کد قصد داریم یک شمارنده جانسون (حلقوی دو جهته) را با شماتیک طراحی کنیم. که زمانی که بر روی برد پروگرام می شود باعث حرکت ال ای دی ها به دو جهت راست و چپ بشود.

آماده سازی نرم افزار

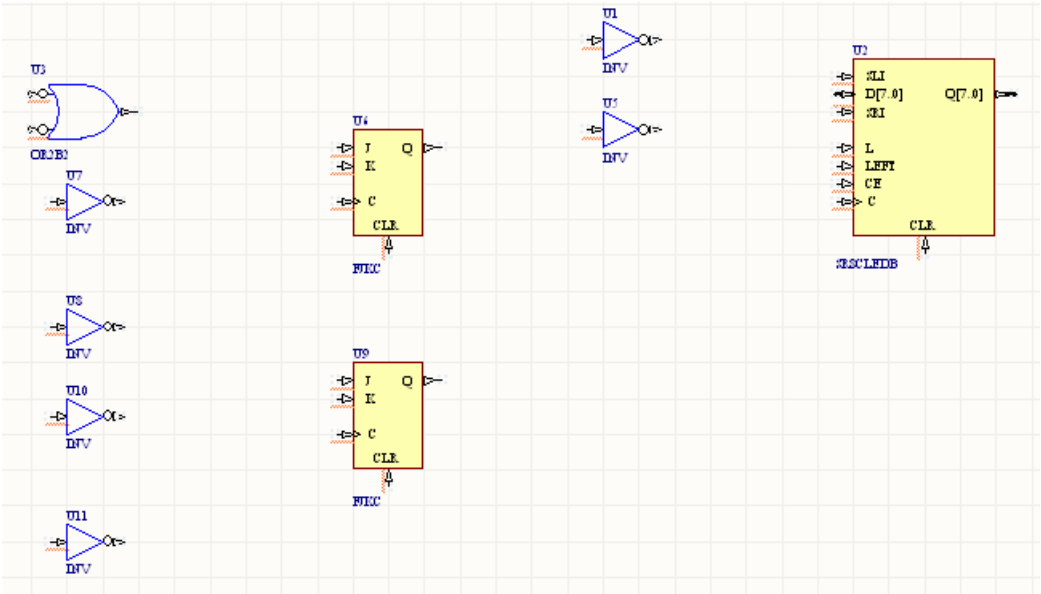
برای شروع کار یک پروژه ی FPGA بسازید و یک شماتیک خالی اضافه کنید و قطعات زیر را به صورت شکل زیر در آن جانمایی کنید:

۱- یک عدد SR8CLEDB از کتابخانه ی FPGA Generic.IntLib

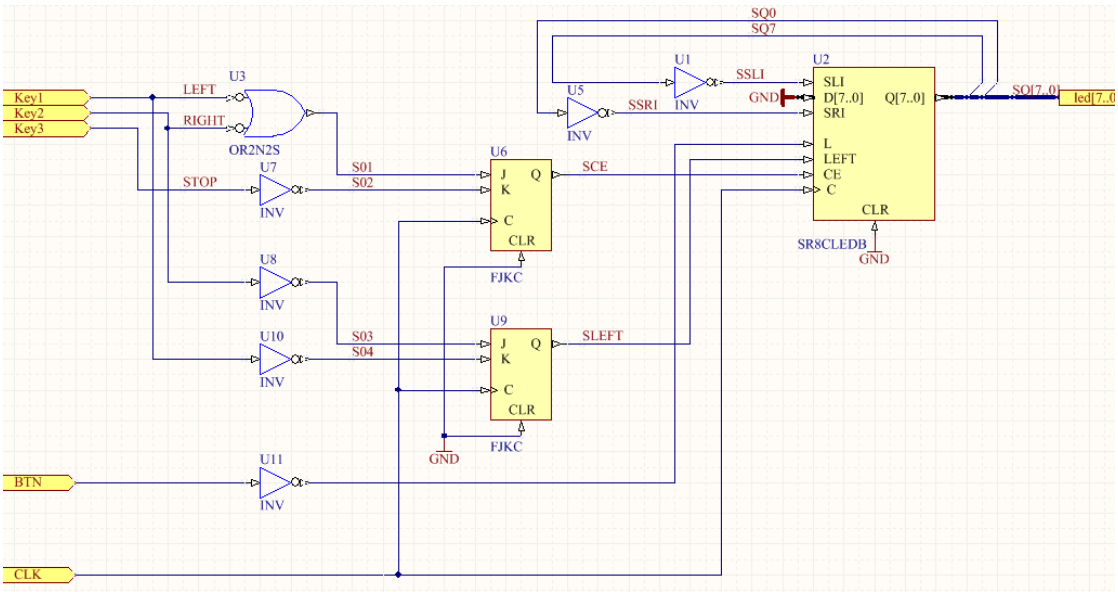
۲- شش عدد INV از کتابخانه ی FPGA Generic.IntLib

۳- یک عدد OR2N2S از کتابخانه ی FPGA Generic.IntLib

۴- دو عدد FJKC از کتابخانه ی FPGA Generic.IntLib



این قطعات را مطابق شکل زیر با Wire به هم وصل کنید و برای ورودی ها و خروجی ها مانند شکل زیر Port متصل کنید:



همین شماتیک را به همراه پروژه می توانید در پوشه ی 6. Scrolling LEDs by SimpleCounter پیدا کنید.

آماده سازی سخت افزار

تمام ۸ ال ای دی های قرمز یا سبز روی برد را به ترتیب به پایه هایی که در فایل ucf. بیان کرده اید متصل کنید.

Constraint1.Constraint

```
Record=Constraint | TargetKind=Part | TargetId=XC3S400-4PQ208C
Record=Constraint | TargetKind=Port | TargetId=BTN | FPGA_PINNUM=P2
Record=Constraint | TargetKind=Port | TargetId=CLK | FPGA_PINNUM=P79
Record=Constraint | TargetKind=Port | TargetId=KEY1 | FPGA_PINNUM=P5
Record=Constraint | TargetKind=Port | TargetId=KEY2 | FPGA_PINNUM=P3
Record=Constraint | TargetKind=Port | TargetId=KEY3 | FPGA_PINNUM=P9
Record=Constraint | TargetKind=Port | TargetId=LED[7..0] |
FPGA_PINNUM=P51,P48,P45,P43,P40,P37,P35,P33
```

- فایل Constraint را به پروژه اضافه کنید و پین ها را به طور صحیح در آن تخصیص دهید. سپس پروژه را کامپایل و سنتز کنید و نتیجه را بر روی برد مشاهده نمایید.

- برای مشاهده ی صحیح عملکرد مدار چه پیشنهادی ارائه می دهید.

آزمایش ۷

پیاده سازی هسته پردازنده ۳۲ بیتی TSK3000 با یک مثال ساده (شمارنده)

مقدمه

در این آزمایش نحوه ی پیاده سازی یک طرح دارای پردازنده را به شما نشان خواهیم داد. این طرح شامل یک هسته ی نرم افزاری پردازنده TSK3000 می باشد که به ۸ عدد LED متصل است. این پردازنده را توسط قطعه کدی که به زبان C خواهیم نوشت، برنامه ریزی می کنیم و سپس کل طرح را به روی سخت افزار FPGA منتقل و پیاده سازی می کنیم.

در انتهای این آزمایش، ردیفی از LED های روی برد به صورت باینری از صفر تا ۲۵۵ می شمرند.

آماده سازی سخت افزار

تمام ۸ ال ای دی های قرمز یا سبز روی برد را به ترتیب به پایه هایی که در فایل ucf. بیان کرده اید متصل کنید.

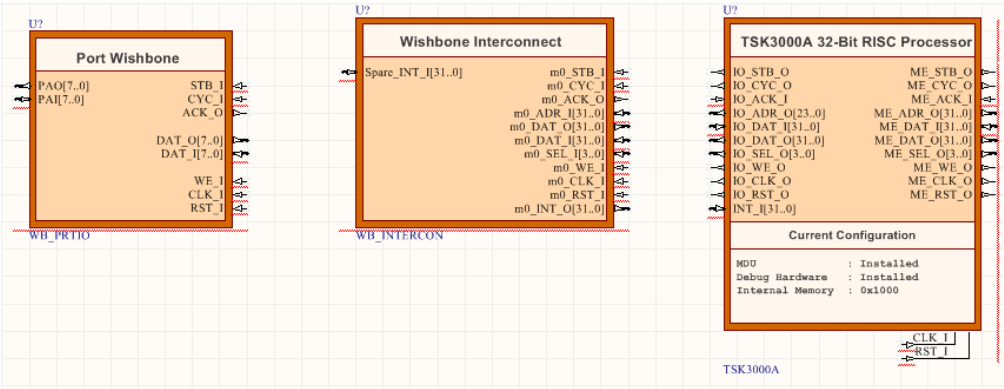
آماده سازی نرم افزار

برای شروع کار یک پروژه ی FPGA بسازید و یک شماتیک خالی اضافه کنید و قطعات زیر را به صورت شکل زیر در آن جانمایی کنید:

۱- یک عدد TSK3000A از کتابخانه ی FPGA Processors.IntLib

۲- یک عدد WB_PRTIO از کتابخانه ی FPGA Peripherals.IntLib

۳- یک عدد WB_INTERCON از کتابخانه ی FPGA Peripherals.IntLib



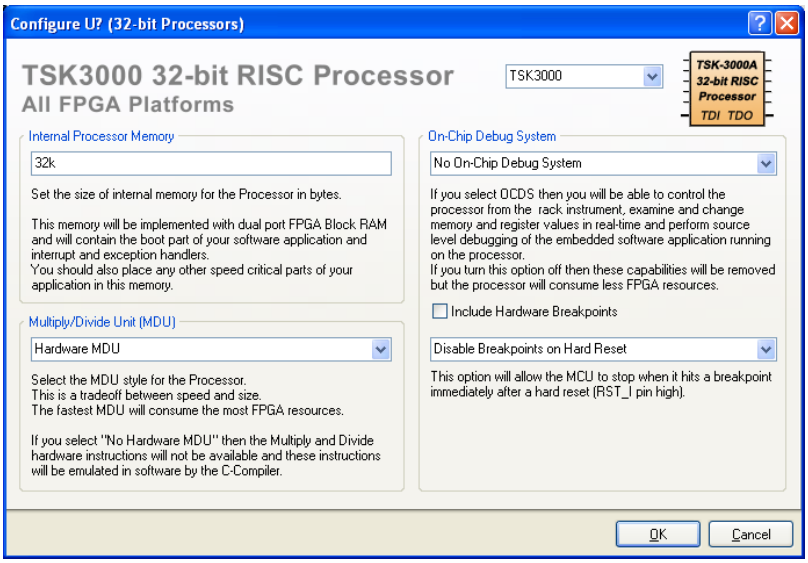
برای Mirror کردن در هنگامی که بلوک را Drag کرده اید دکمه ی x را بفشارید.

سفارش سازی بلوک ها

روی بلوک TSK3000 کلیک راست کنید و Configure U?(TSK3000A)... را بزنید:

توجه کنید که U? همان designator پیشفرض بلوک می باشد که بعدا به آن شماره (annotate) می دهیم.

- مقدار Internal Processor Memory را به ۳۲ کیلوبایت افزایش دهید.
- و قسمت On-Chip Debug System را به No On-Chip Debug System تغییر دهید.

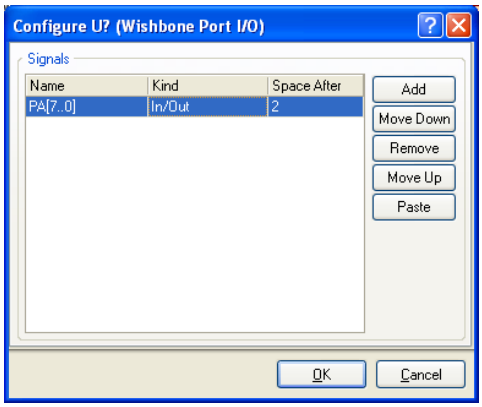


OK را بزنید و به شماتیک بازگردید.

روی بلوک WB_PRTIO کلیک راست کنید و Configure U?(WB_PRTIO)... را بزنید:

مجددا توجه کنید که U? همان designator پیشفرض بلوک می باشد.

- ستون Kind را چک کنید که In/Out باشد.



OK را بزنید و به شماتیک بازگردید.

روی بلوک WB_INTERCON کلیک راست کرده و ...Configure U?(WB_INTERCON) را بزنید:

مجددا توجه کنید که U? همان designator پیشفرض بلوک می باشد.

سمت راست پایین پنجره:

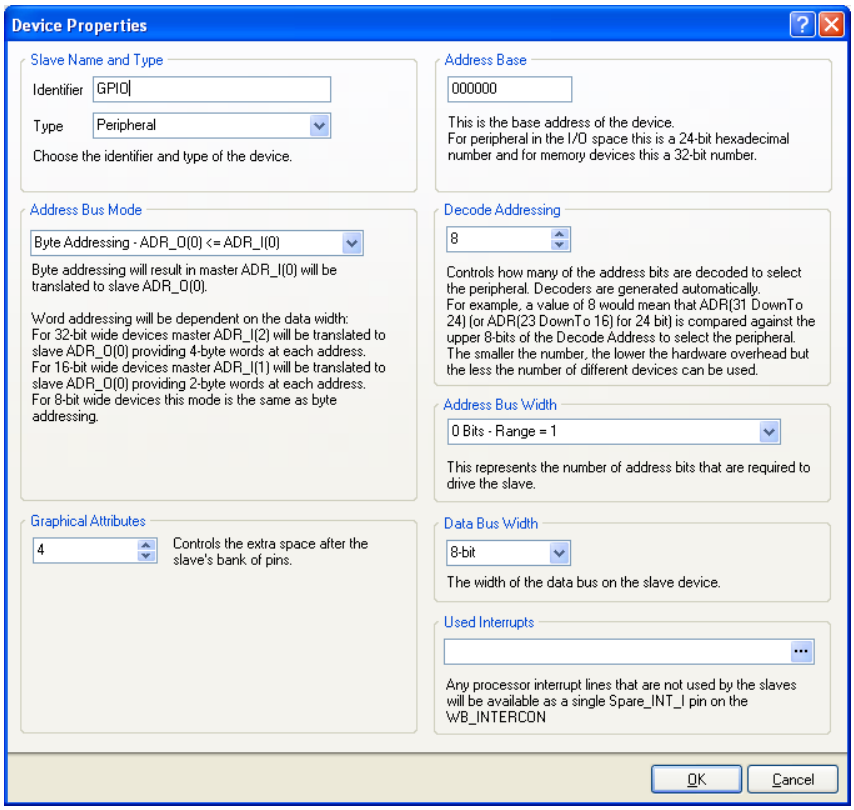
- قسمت Unused Interrupts را به Connect to GND تغییر دهید.

این کار باعث جلوگیری از وقوع وقفه های ناخواسته (که بی استفاده مانده اند) می شود.

- قسمت Master Address Size را به 24-Bit (Peripheral I/O) تغییر دهید.

چون طول باس I/O که TSK3000A دارد ۲۴ بیتی است.

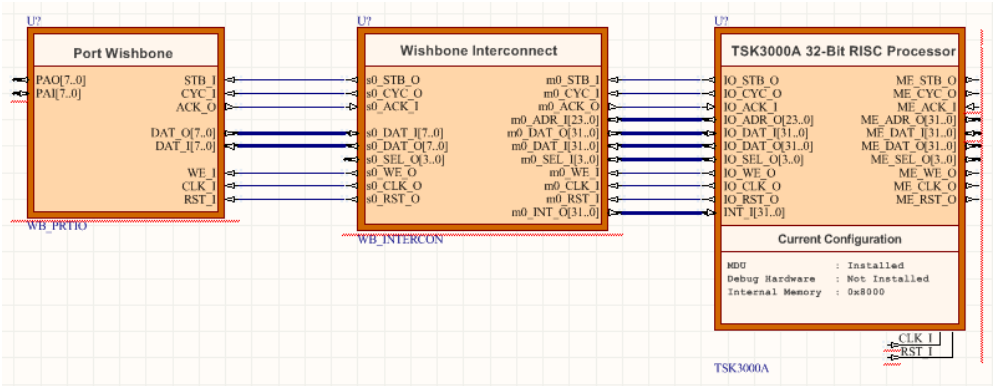
- روی ...Add Device کلیک کنید:
  - یک نام دلخواه را در Identifier وارد کنید. به عنوان مثال: GPIO
  - قسمت Address Bus Mode را به Byte Addressing تغییر دهید.
  - قسمت Data Bus Width را به 8-bit تغییر دهید.



توجه داشته باشید که آدرس پایه 0xFF00_0000 می باشد. در حالی که در این پنجره به مقدار 000000 تنظیم شده است. در واقع بلوک interconnect متوجه هست که فضای I/O های TSK3000A از آدرس 0xFF00_0000 شروع می شود و اتوماتیک آن را تبدیل می کند.

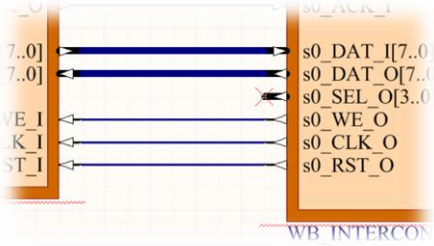
OK را بزنید و به شماتیک بازگردید.

بلوک ها را مطابق شکل زیر بهم وصل کنید:



برای سهولت در سیم کشی ابتدا یک بلوک را به بلوک مجاور نزدیک کنید و رها کنید تا بهم وصل شوند و سپس در حالی که کلید Ctrl را نگه داشته اید، بلوک را به جای اول خور باز گردانید.

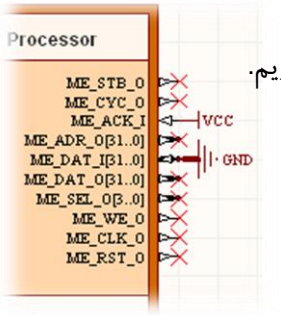
حالا دو جفت پین s0_SEL_O[3..0] –که روی بلوک GPIO موجود نیست ولی روی بلوک interconnect موجود است- باقی می ماند. این پین ها در واقع هریک بایت انتخاب پین، در زمانی که با لوازم۳۲ بیتی اتصال دارد، مورد استفاده میگیرد. از آنجا که اکنون با بلوک GPIO ،که ۸ بیتی تنظیم شده است، اتصال دارد این پین ها استفاده نمی شوند. لذا بر روی آن No ERC قرار میدهیم. (از منوی Place > Directives > No ERC)



سمت راست بلوک TSK3000A مربوط به باس حافظه است که اکنون با آن کاری نداریم.

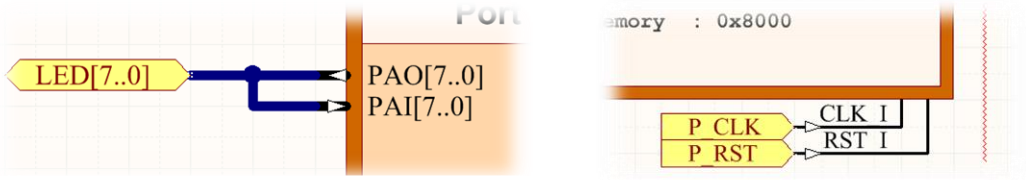
بجز دو پین که ورودی هستند و نباید آنها را بدون اتصال رها کنیم:

- پین ME_ACK_I که باید به Vcc Power Port متصل شود.
- پین ME_DAT_I که باید به GND Bus Power Port متصل شود.



حالا برای ورودی های CLK_I و RST_I پورت متصل کنید. و خروجی GPIO را هم با باس به صورت زیر به پورت متصل و نام گذاری کنید:

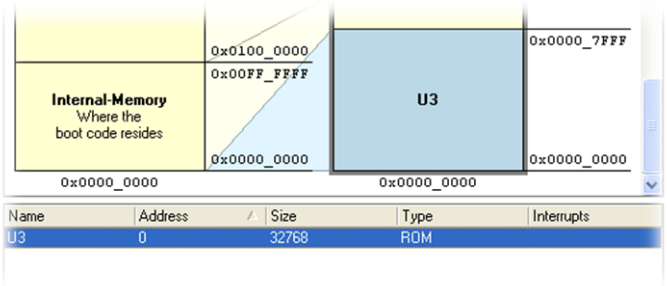
هنگام تعیین نام پورت متصل به باس، علاوه برنام، طول باس را هم به صورت مقابل باید بیان کرد: LED[7..0]



حالا یک بار پروژه را Annotate کنید. (از منوی Tools > Annotate Schematics Quietly... را بزنید)

حالا کل پروژه را ذخیره کنید.

برای تخصیص حافظه به پردازنده TSK3000A روی آن کلیک راست کنید و Configure Processor Memory را بزنید. طبق آنچه در این پنجره به صورت گرافیکی هم نشان داده شده میزان 32768 (32K) بایت به پردازنده تخصیص یافته است.

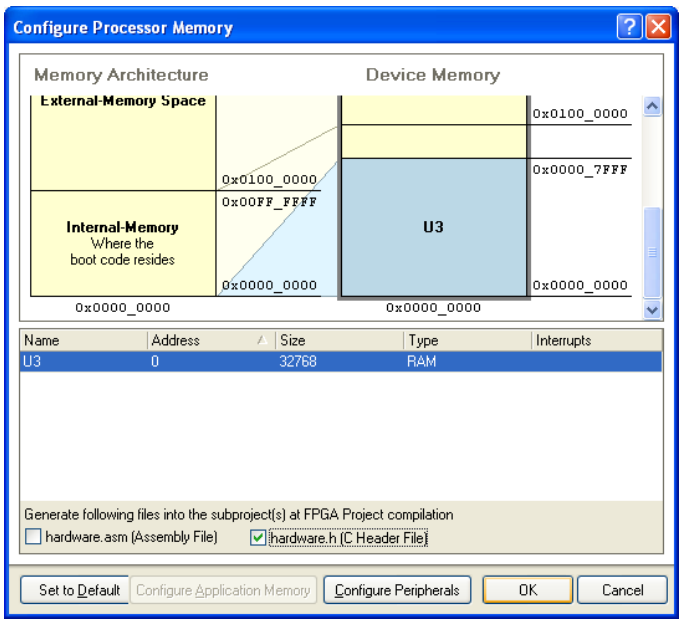


اگر نوع حافظه ROM تنظیم شده است روی آن دبل کلیک کنید و نوع آن را به RAM تغییر دهید و سپس OK را بزنید و به پنجره اصلی تنظیمات بازگردید.

⚠ سپس توجه کنید که تیک hardware.h (C Header File) حتما فعال باشد.

ⓘ این کار باعث تولید فایل hardware.h در پروژه ی embedded (جهت برنامه نویسی پردازنده) می شود.

پس اکنون این پنجره باید بصورت زیر باشد:



OK را بزنید و به شماتیک بازگردید.

جهت تنظیم و معرفی لوازم جانبی متصل به پردازنده، روی آن کلیک راست کنید و Configure Processor Peripheral... را بزنید. و از پنجره باز شده دکمه ی Import From Schematic را بزنید و در پاسخ پرسش Yes را کلیک کنید. پس از آن پنجره ای باز می شود که لیست لوازم جانبی متصل را نشان می دهد. روی بخش Do not import از interconnect که با U2 مشخص شده کلیک کنید و آن را به Import تغییر دهید. این کار را برای تمام زیر مجموعه های آن نیز انجام دهید. و سپس OK را بزنید و به پنجره اصلی تنظیمات بازگردید.

⚠ سپس توجه کنید که تیک hardware.h (C Header File) حتما فعال باشد.

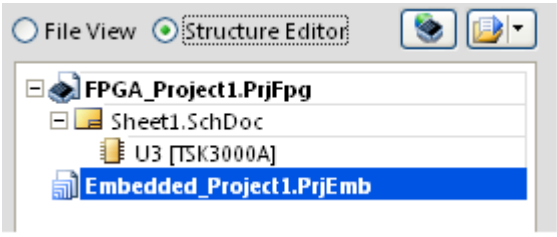
حالا کل پروژه را مجددا ذخیره کنید. بخش سفارش سازی و طراحی سخت افزار تمام شد در ادامه به برنامه نویسی پردازنده و نحوه ی ساخت پروژه ی embedded می پردازیم.

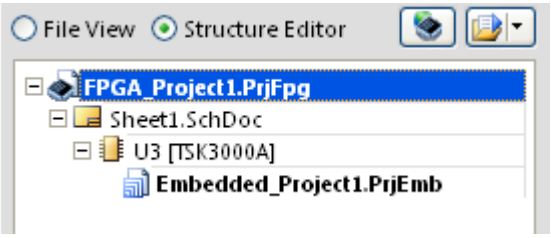
از منوی File > New > Project > Embedded Project را انتخاب کنید.

سپس از منوی File > New > C Source document را انتخاب کنید و تمام آنها را ذخیره کنید.

جهت معرفی و اتصال این پروژه به پروژه ی قبلی مراحل زیر را به ترتیب انجام دهید:

۱. در پنل Project روی Structure Editor کلیک کنید. تا به حالت نمایش Structure Editor در آید.
۲. سپس روی پروژه ی قبلی (یعنی پروژه ی FPGA و نه embedded) کلیک راست کنید و Compile FPGA project را بزنید تا فایل شماتیکی که رسم کرده بودید در زیر آن نمایان شود.
۳. حالا پروژه Embedded را بر روی نماد TSK3000 (زیرمجموعه شماتیک است) بکشید و رها کنید.
۴. سپس به حالت File View بازگردید.





توجه کنید که اکنون باید فایل hardware.h به پروژه ی Embedded شما اضافه شده باشد.

حالا باید پروژه ی Embedded را برای TSK3000A سفارش سازی کنیم:

روی پروژه Embedded کلیک راست کنید و Project Options... را بزنید.

در بخش Device قطعه ی TSK3000A را انتخاب کنید.

در بخش Linker، مورد Stack/Heap را انتخاب کنید سپس قسمت Stack size را برابر 4K قرار دهید.

و قسمت Heap size را پاک کنید.

در قسمت های قبلی ما پردازنده را به 32K بایت حافظه مجهز کردیم.

حالا باید پروژه Embedded را تنظیم کنیم تا کامپایلر متوجه باشد که نرم افزار چگونه و به چه میزان از این حافظه دسترسی دارد. اکنون ما می خواهیم این ۳۲ کیلوبایت را به ۱۶ کیلوبایت حافظه ی ROM (حافظه ی برنامه) و ۱۶ کیلوبایت حافظه ی RAM فرار تقسیم کنیم.

برای اینکار در همین پنجره (Project option) روی سربرگ Configure Memory کلیک کنید.

تیک Automatically synchronize on embedded project compile را بردارید.

سپس روی ستون Application Memory کلیک راست کنید و Delete All را انتخاب کنید تا این بخش خالی و به رنگ آبی در آید.

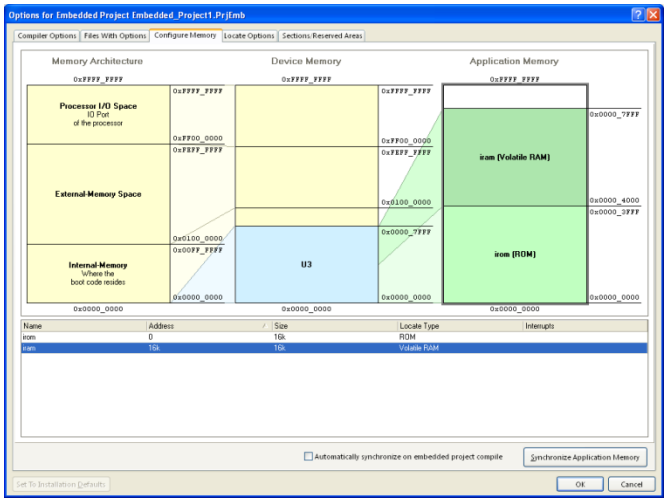
مجدد روی ستون Application Memory کلیک راست کنید و ...Add Memory را بزنید.

- یک نام دلخواه (به عنوان مثال irom) را در بخش Name وارد کنید.
- نوع آن را ROM انتخاب کنید.
- اندازه ی آن را 16K وارد نمایید.
- آدرس شروع (Address Base) این حافظه را برابر با صفر قرار دهید.

مجدد روی ستون Application Memory کلیک راست کنید و ...Add Memory را بزنید.

- یک نام دلخواه (به عنوان مثال iram) را در بخش Name وارد کنید.
- نوع آن را RAM – Volatile انتخاب کنید.
- اندازه ی آن را 16K وارد نمایید.
- آدرس شروع (Address Base) این حافظه را باید 16K قرار دهید. (انتهای حافظه ی قبلی)

اگر خطایی وجود نداشته باشد بلوک به رنگ سبز در می آید در غیر اینصورت قرمز می شود:



OK را بزنید و به صفحه ی اول بازگردید.

حالا این پروژه نیز به درستی تنظیم گردید و آماده ی برنامه نویسی برای پردازنده می باشد.

leds1.c

```
#include <stdint.h>
#include "hardware.h"

volatile uint8_t * const leds = (void *)Base_GPIO;
void main( void ){
    *leds = 255;  // Initialize the LEDs to all OFF
    for ( ;; ){
        (*leds)--;
        for ( int delay = 0; delay < 4000000; delay++ ){
            __nop();  // Two underscores
        }
    }
}
```

همین کد را به همراه پروژه می توانید در پوشه ی 7. Counter LEDs by TSK3000 پیدا کنید.

توجه کنید که Base_GPIO در فایل hardware.h تعریف شده است، که آدرس پایه ی I/O است.
سپس آن را Run تا عیب یابی شود. (از منوی Debug > Run)

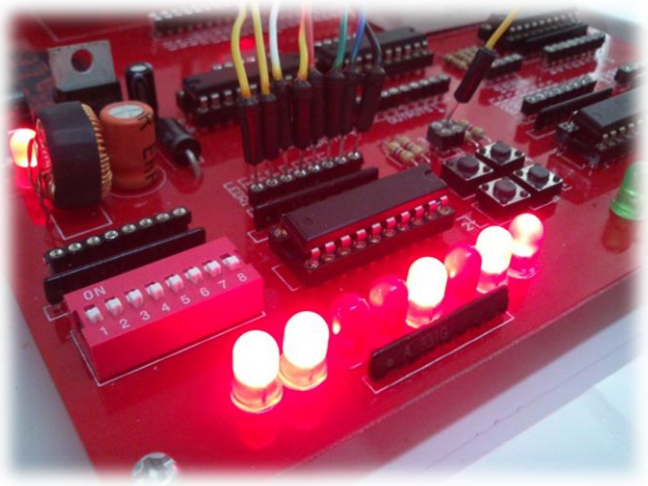
حالا پورت های شماتیک را به پین های FPGA تخصیص دهید:
فایل Constraint را به پروژه ی FPGA اضافه کنید و اسم قطعه و پین ها را در آن بیان کنید.

Constraint1.Constraint

```
Record=Constraint | TargetKind=Part | TargetId=XC3S400-4PQ208C
Record=Constraint | TargetKind=Port | TargetId=LED[7..0] |
FPGA_PINNUM=P51,P48,P45,P43,P40,P37,P35,P33
Record=Constraint | TargetKind=Port | TargetId=P_CLK | FPGA_PINNUM=P79
Record=Constraint | TargetKind=Port | TargetId=P_RST | FPGA_PINNUM=P10
```

یک بار کل پروژه را کامپایل کنید تا اگر خطایی وجود دارد، نمایش داده شود. سپس آن را سنتز کنید.
حالا برد را آماده ی پروگرام کنید و نتیجه را بر روی آن مشاهده کنید.

تصویری از نتیجه نهایی:



آزمایش ۸

ارتباط با نمایشگرهای کریستال مایع، کاراکتری با پردازنده TSK3000A

مقدمه

همان طور که در آزمایش ۵ نیز اشاره شد برنامه نویسی به زبان های توصیف سخت افزار به ویژه در پروژه های گسترده به علت پیچیدگی های طراحی پروتکل اولیه، مناسب گسترش و توسعه ارتباط با لوازم جانبی نیست و از این رو طراحان حرفه ای سیستم ها از هسته های قابل پیاده سازی در طرح خود استفاده می کنند.

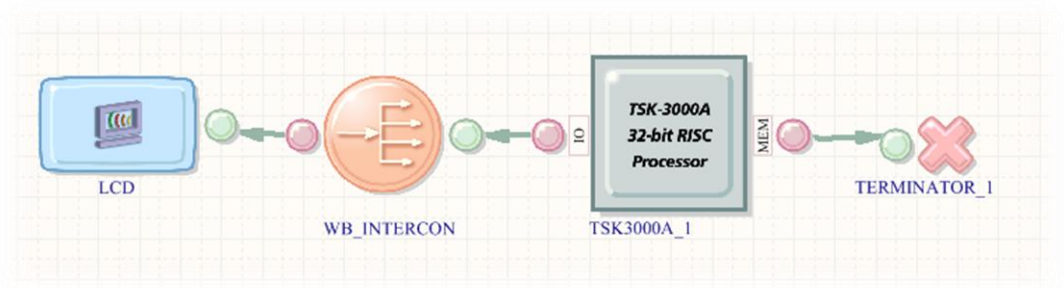
در این آزمایش قصد پیاده سازی آزمایش ۴ (ارتباط با نمایشگرهای کریستال مایع، کاراکتری) به کمک هسته ی نرم افزاری TSK3000A (آزمایش۷) را داریم.

آماده سازی سخت افزار

تمام پایه های ال سی دی را مانند آزمایش ۴ (با همان ترتیب) به پین های برد متصل کنید.

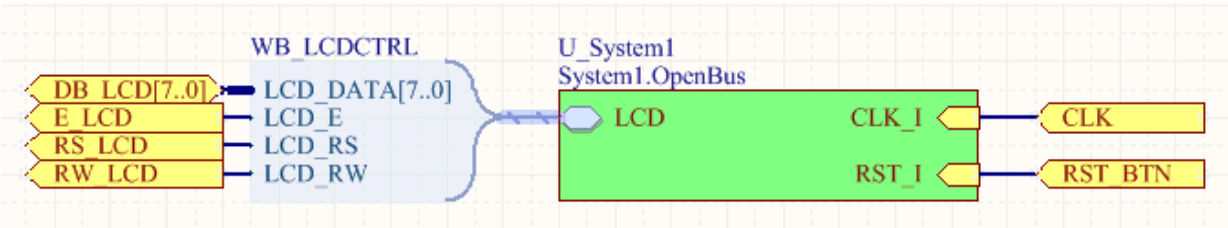
آماده سازی نرم افزار

یک پروژه ی FPGA با یک شماتیک خالی ایجاد کنید و یک فایل OpenBus به پروژه اضافه کنید و طرح زیر را در آن ببندید:



روی بلوک TSK3000A دبل کلیک کنید و میزان حافظه را 16K تنظیم کنید و On-Chip را غیرفعال کنید.

این فایل را نیز ذخیره کنید و به شماتیک بازگردید و از منوی Design > Create Sheet Symbol From Sheet or HDL را بزنید و فایل OpenBus را انتخاب کنید و این بلوک را در جایی مناسب قرار دهید و به صورت زیر پورت ها را متصل کنید:



برای قرار دادن اتصال LCD از منوی Place > Harness > Predefined Harness Connector را بزنید. و روی WB_LCDCTRL کلیک کنید و OK را بزنید.

حالا پروژه Embedded را ایجاد کنید و آن را به پروژه FPGA الصاق کنید. (مراجعه به آزمایش قبل) و یک فایل C به آن اضافه کنید و کد زیر را در آن بنویسید:

C_Source1.c

```
#include "chars.h"
#include <timing.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <devices.h>
#include <drv_lcd.h>
#include <drv_lcd_util.h>
#define MSG_SIZE 128

drv_lcd_t *display;
const uint32_t delay = 1500;
void init(void){
    uint8_t *cdata = chardata;
    display = lcd_open(DRV_LCD_1);
    for (uint32_t i = 0; i < charmax; i++)
    { lcd_set_custom_char(display, i, cdata + i * 8); }

}

void lcd_print(const char *msg){
    uint8_t size;
    for (size = 0; size < MSG_SIZE; size++)
    { if (msg[size] == '\0') break;}

    lcd_write(display, msg, size);
}

void main (void)
{
    char Msg[MSG_SIZE];
    uint32_t count = 0;
    static uint64_t start_tick;
    uint32_t elapsed;

    init();

    lcd_set_cursor(display, 0, 0);
    start_tick = clock_ms();
    lcd_print("Ali Damirchy\n902151008");

    delay_ms(delay);

    while (1)
    {
        lcd_print("\f\r-=Introducing=-\n");
        delay_ms(delay * 2);
        for (uint32_t i = 0; i < 16; i++)
        {lcd_write_char(display, ctop[i]);}
        lcd_goto_xy(display, 0, 1);
        for (uint32_t i = 0; i < 16; i++)
        {lcd_write_char(display, cbot[i]);}
    }

}
```

همین کد و هدر فایلهای مورد نیاز رامی توانید در پوشه ی 8. LCD 2x16 byTSK300 پیدا کنید

کد را ذخیره کنید.

پروژه Embedded را به صورت زیر سفارش سازی کنید:

- Stack Size را به 1K و همچنین Heap Size را به 1K تغییر دهید.
- 12K بایت حافظه ی ROM با آدرس شروع صفر بسازید
- 4K بایت نیز حافظه ی Volatile RAM با آدرس شروع 12Kبسازید

سپس یک فایل SwPlatform نیز به پروژه Embedded اضافه کنید و آن را به صورت زیر تغییر دهید:

- روی دکمه ی Import Form FPGAکلیک کنید تا بلوک LCD به آن اضافه شود.
- سپس روی دکمه Grow Stack Up... کلیک کنید تا درایور LCD نیز به آن اضافه شود.
- سپس روی دکمه Compile Project کلیک کنید و این فایل را ذخیره نمایید.

حالا پورت هایی که در شماتیک قرار داده بود را به پین های روی FPGA تخصیص دهید:

LCD.Constraint

```
Record=Constraint | TargetKind=Part | TargetId=XC3S400-4PQ208C

Record=Constraint | TargetKind=Port | TargetId=CLK | FPGA_PINNUM=P79
Record=Constraint | TargetKind=Port | TargetId=DB_LCD[7..0] |
FPGA_PINNUM=P5,P11,P16,P21,P27,P33,P37,P45
Record=Constraint | TargetKind=Port | TargetId=RS_LCD | FPGA_PINNUM=P52
Record=Constraint | TargetKind=Port | TargetId=RW_LCD | FPGA_PINNUM=P50
Record=Constraint | TargetKind=Port | TargetId=E_LCD | FPGA_PINNUM=P51

Record=Constraint | TargetKind=Port | TargetId=RST_BTN | FPGA_PINNUM=P3
```

برد را آماده برنامه ریزی کنید و نتیجه را بر روی آن مشاهده کنید: