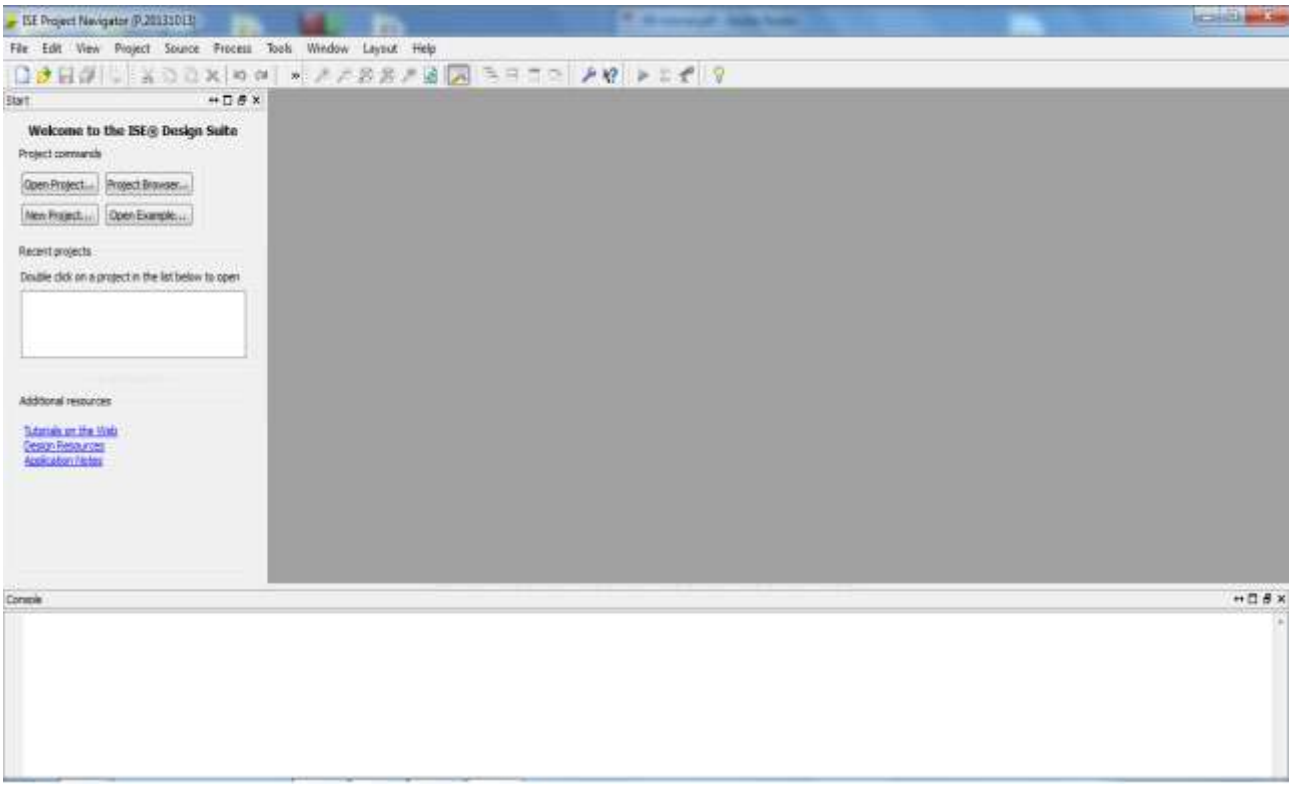
آموزش پیاده سازی و شبیه سازی طراحی های HDL
در ابزار Xilinx ISE 14.7

۱. مقدمه:

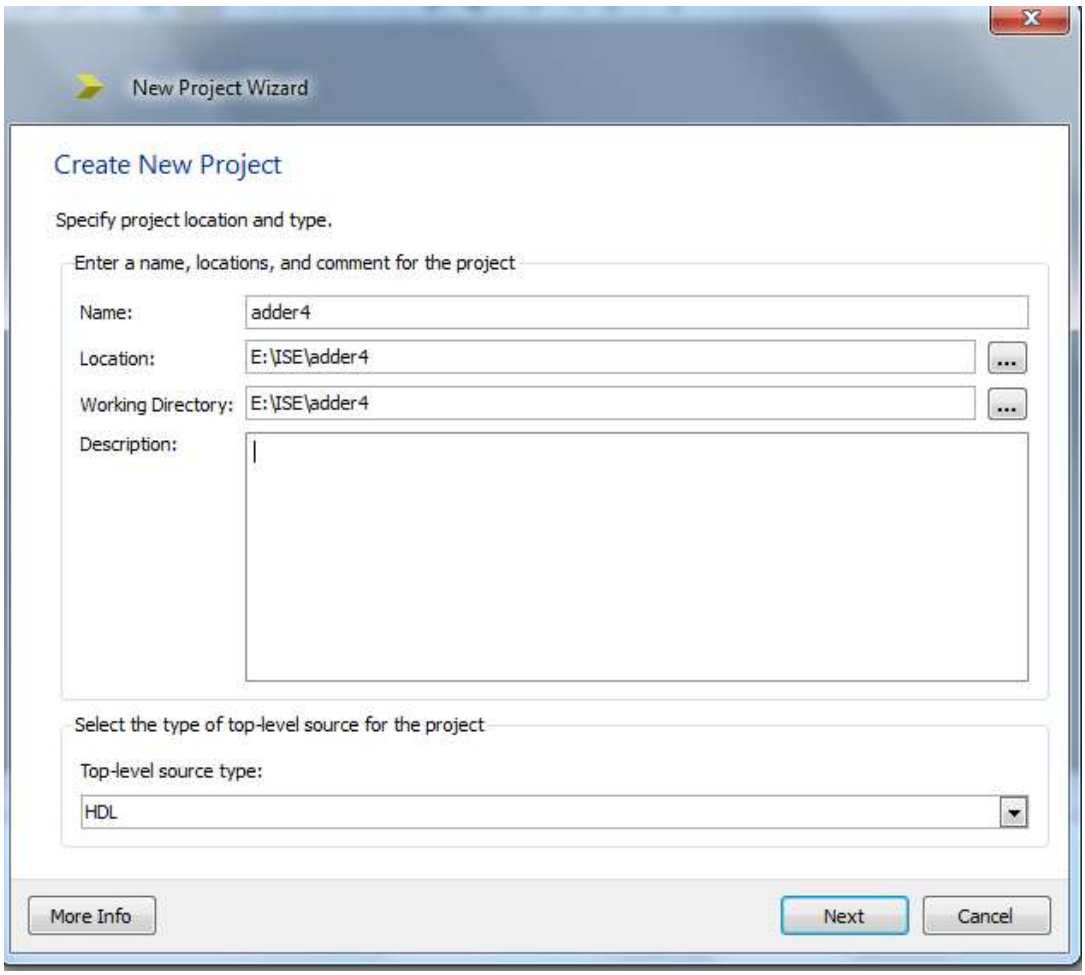
در این فایل آموزشی، شما نحوه کار با ابزار ISE متعلق به شرکت Xilinx را یاد خواهید گرفت. شما با استفاده از این ابزار می توانید کد های Verilog خود را رویFPGA های شرکت Xilinx سنتز کرده، آنها را مورد سنجش و ارزیابی قرارداده و عملکرد آنها را شبیه سازی کنید.

۲. ساخت پروژه:

نرم افزار Project Navigator را اجرا کنید. پنجره ای مانند زیر مشاهده خواهید کرد.



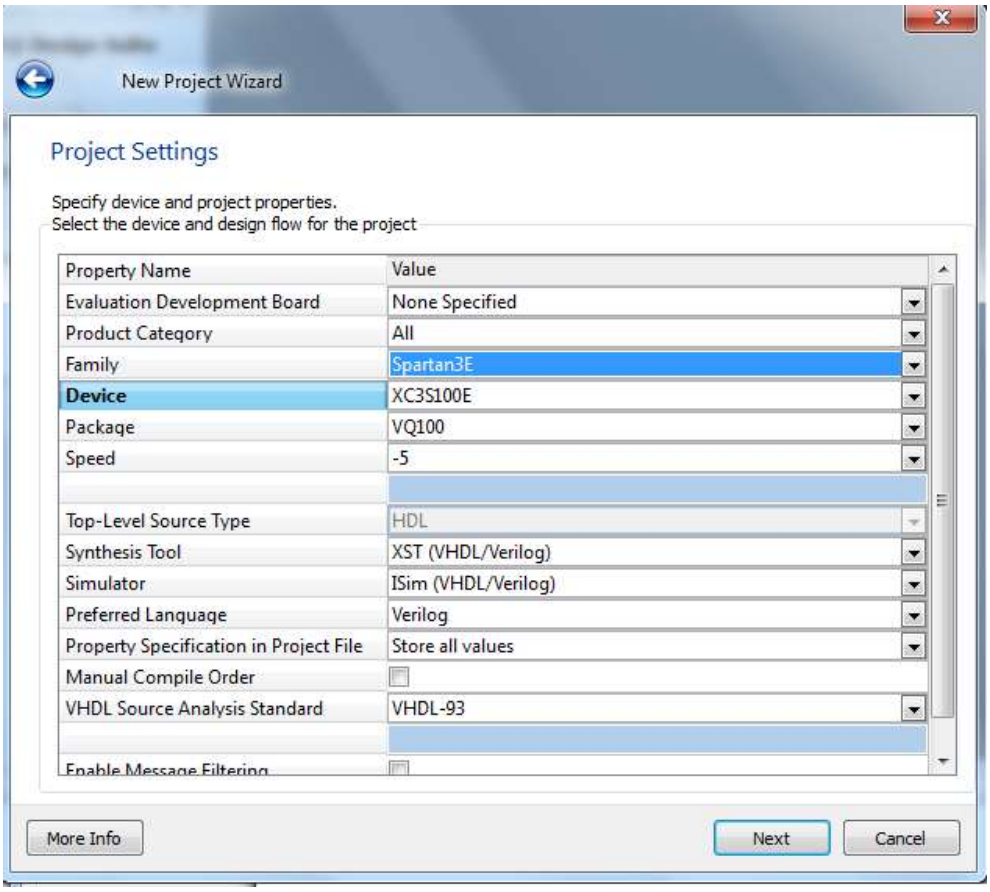
نامی دلخواه برای پروژه انتخاب می کنیم ، مسیر آن را روی دیسک مشخص کرده و نوع ورودی طراحی را HDL انتخاب می کنیم:



در صفحه بعد پارامتر ها را به ترتیب زیر انتخاب می کنیم:

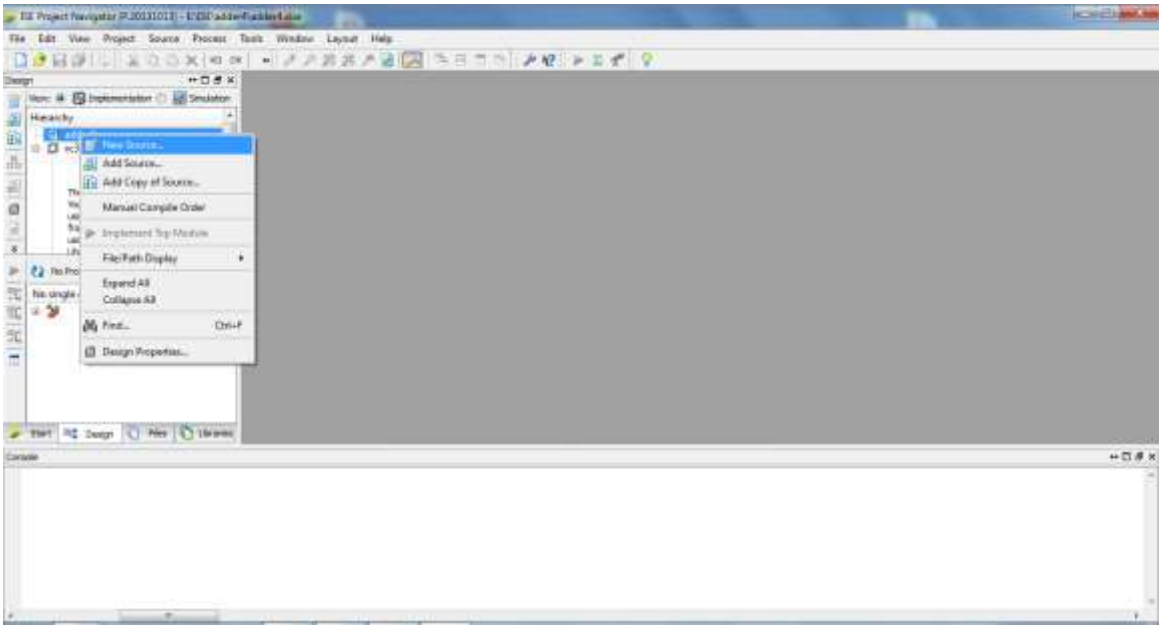
نوع FPGA که قصد داریم طراحی را روی آن انجام دهیم به طور مثال به شرح زیر انتخاب می کنیم :

- سری Spartan3E
- نوع XC3S100E
- بسته بندی VQ100
- رده سرعت 5
- ابزار شبیه سازی ISim (VHDL/Verilog)
- زبان انتخابی Verilog



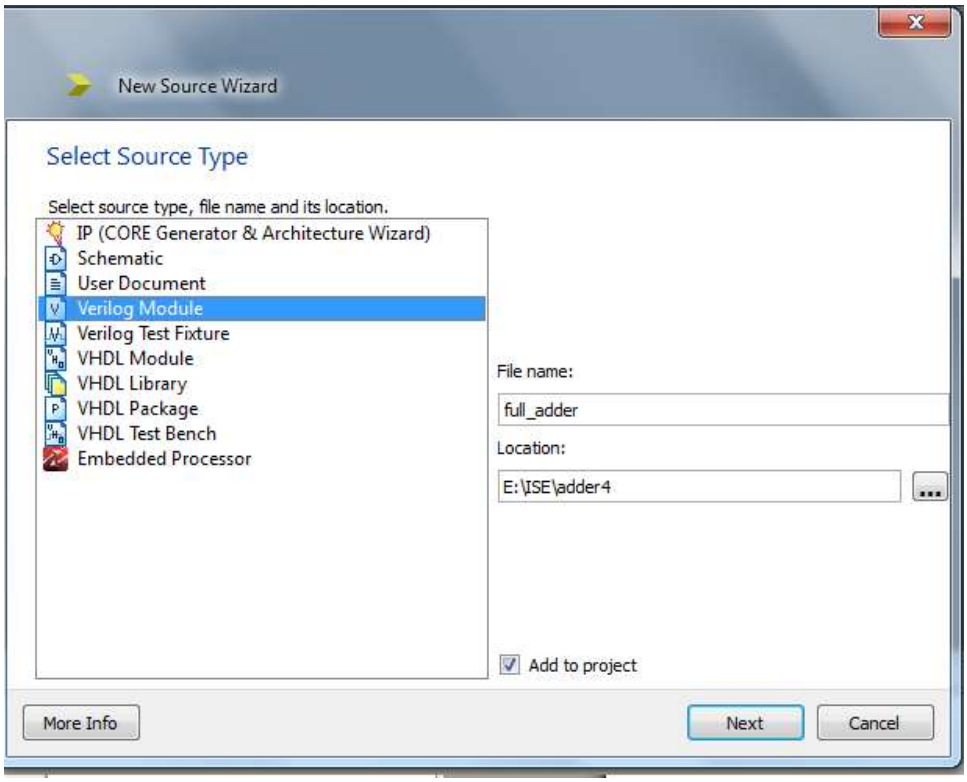
۲.۱ اضافه کردن فایلهای طراحی

در میانه ساخت پروژه و در صفحه بعد امکان اضافه کردن فایل های طراحی را خواهیم داشت:



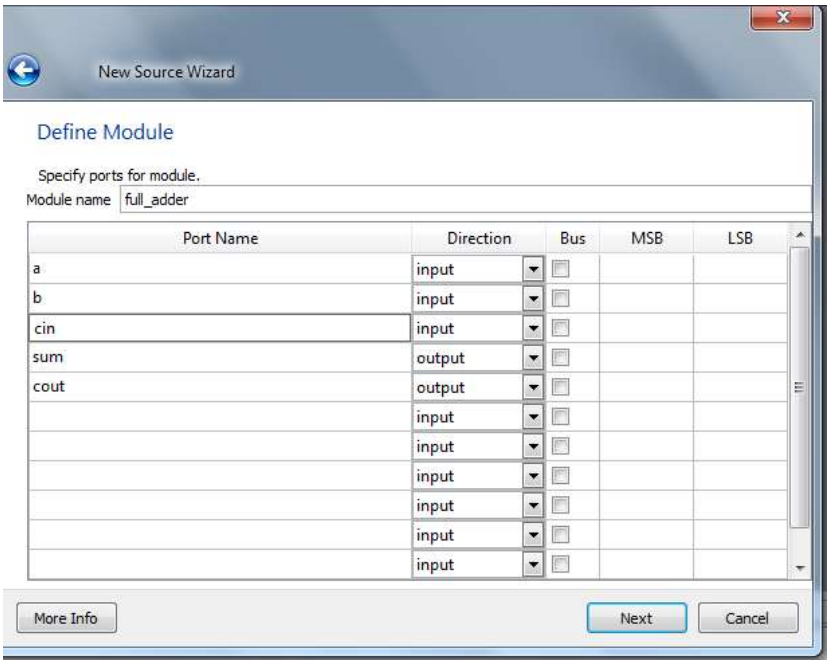
در این صفحه گزینه New Source را انتخاب می کنیم و در صفحه بعد گزینه ها را به این ترتیب انتخاب می کنیم:

Verilog Module - > File Name = full_adder

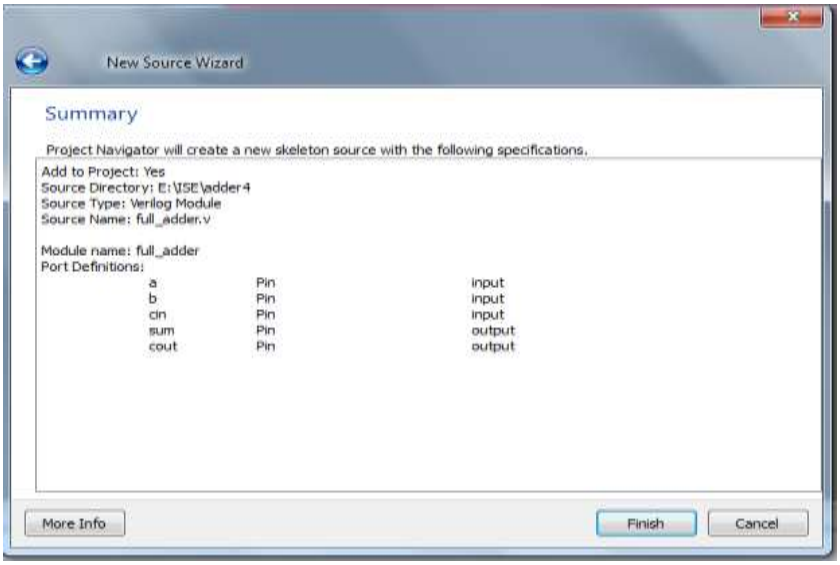


سپس گزینه Next را می زنیم. در این قسمت Interface (سیگنال های ورودی/ خروجی) ماژول طراحی شده را مشخص می کنیم:

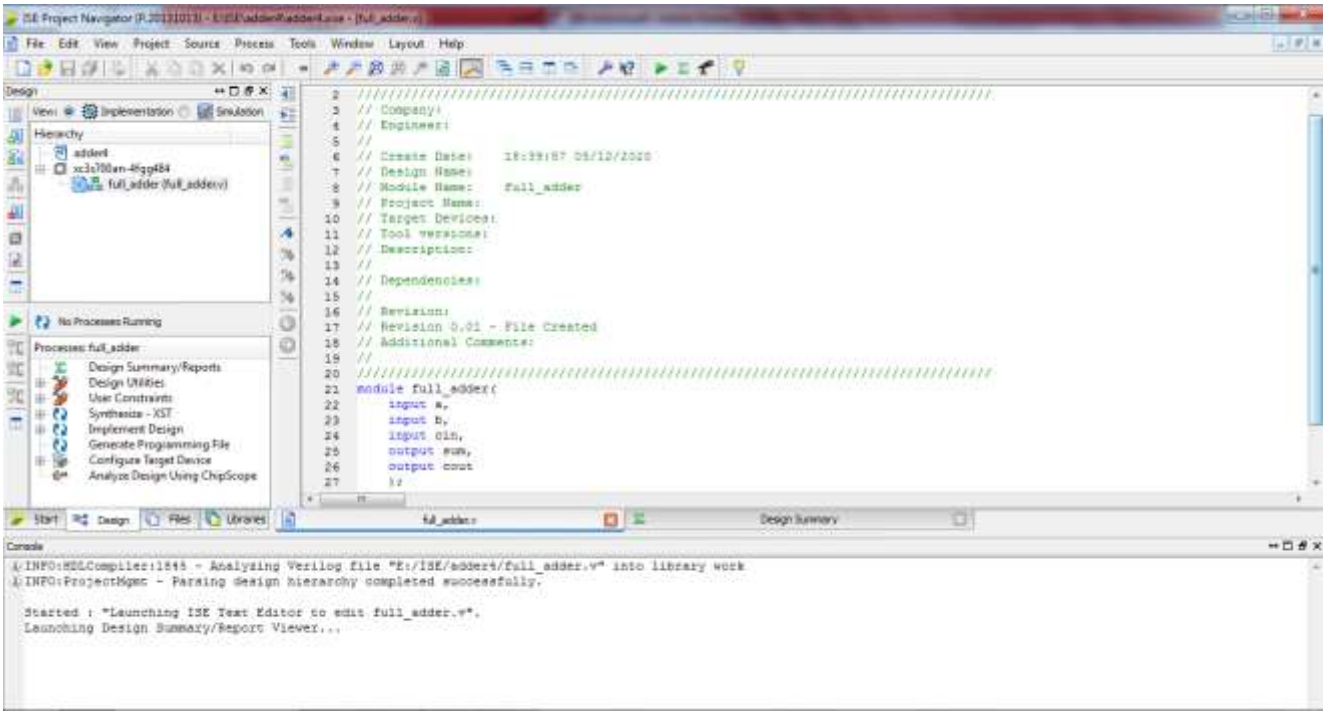
مطابق شکل ۵ سیگنال یک تمام جمع کننده را تعریف کرده که در این بین cin , b, a سیگنالهای ورودی و sum و cout سیگنالهای خروجی هستند.



در صفحه آخر این wizard شاهد خلاصه فرآیندی که طی کردید خواهید بود:



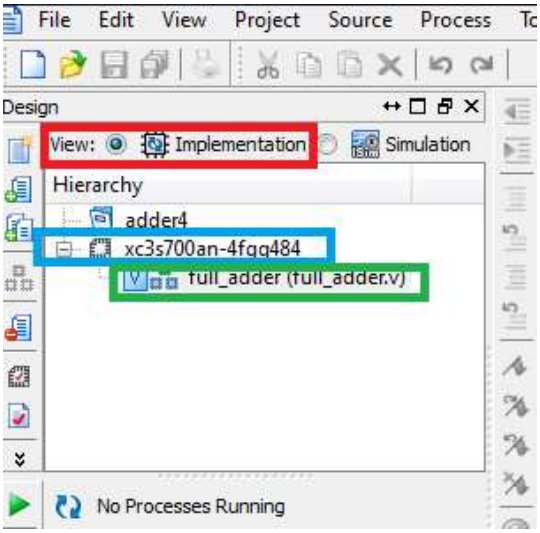
در این قسمت Finish را بزنید. صفحه زیر را مشاهده می کنید:



۳. آشنایی با اجزاء صفحه اصلی Project Navigator

در این صفحه اجزا زیر حائز اهمیت هستند.

در کادر سمت چپ عناصر زیر قابل مشاهده هستند:



- بخش view که برای (سنتز و پیاده سازی) یا (شبیه سازی) مدار از آن می توانید استفاده کنید.(کادر قرمز)
- عنصر Top- Level طراحی ما که سطح بالا ترین بخش طراحی ما است.(کادر سبز)
- مشخصات چیپ FPGA انتخاب شده برای پیاده سازی و سنتز(کادر آبی). با کلیک راست روی آن و انتخاب گزینه Design Properties می توا نید چیپ و جزییات پروژه را که در wizard اول انتخاب کردید، تغییر دهید.

در کادر سمت راست صفحه نیز ، شما Editor متن را مشاهده می کنید که می توا نید کد Verilog خود را با استفاده از آن تکمیل کنید.

۴. تکمیل طراحی

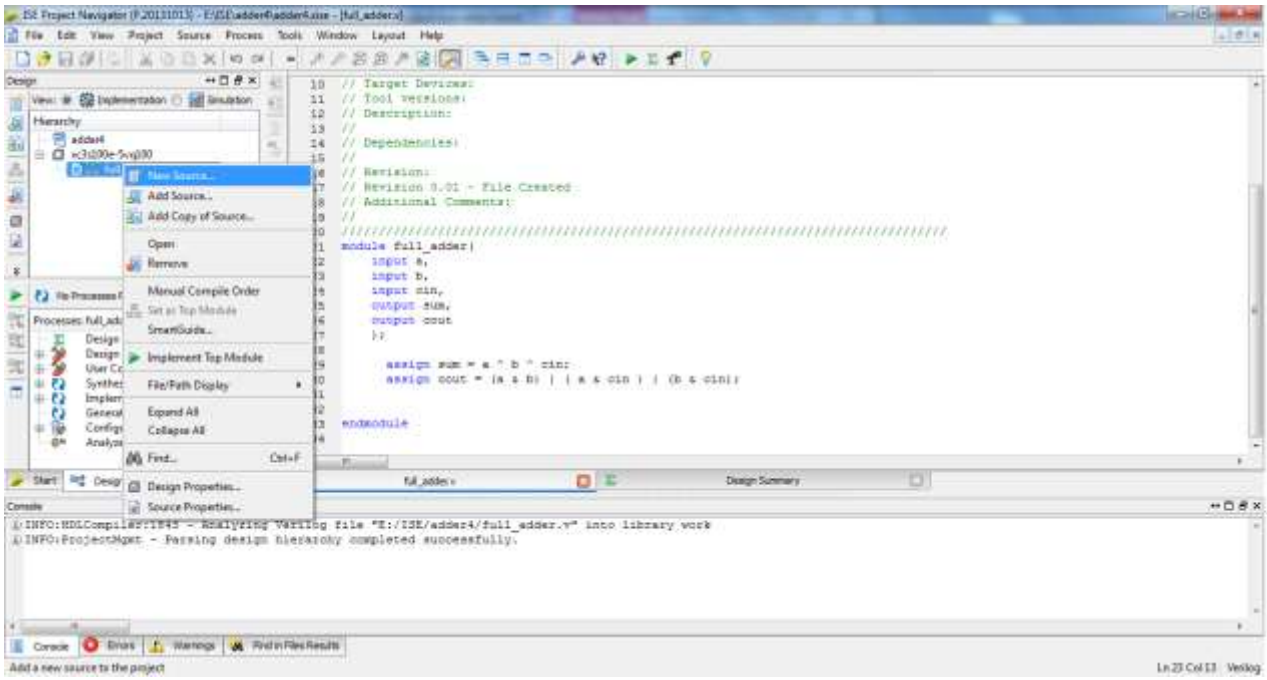
در ابتدای کار کد ما به صورت اتوماتیک تا این مرحله تکمیل شده است:

```
// Module Name:    full_adder
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module full_adder(
    input a,
    input b,
    input cin,
    output sum,
    output cout
    );


endmodule
```

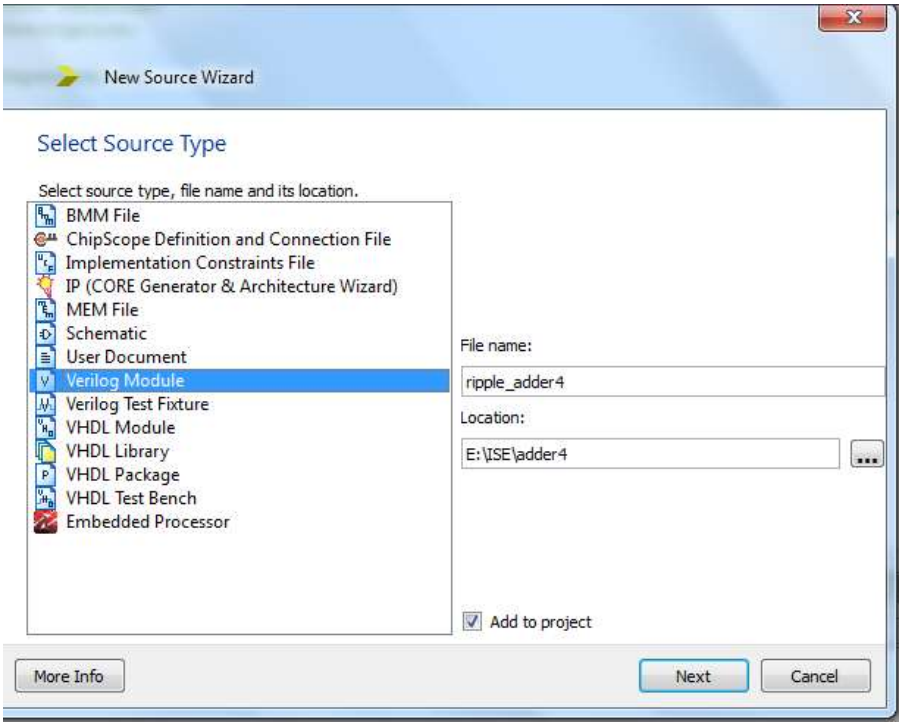
ما دو خط ۲۹و ۳۰ را اضافه می کنیم تا یک Full Adder داشته باشیم:

```
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module full_adder(
    input a,
    input b,
    input cin,
    output sum,
    output cout
    );

      assign sum = a ^ b ^ cin;
      assign cout = (a & b) | ( a & cin ) | (b & cin);


endmodule
```

طراحی خود را ذخیره می کنیم و در بخش Files، کلیک راست کرده و گزینه New Source را انتخاب می کنیم :

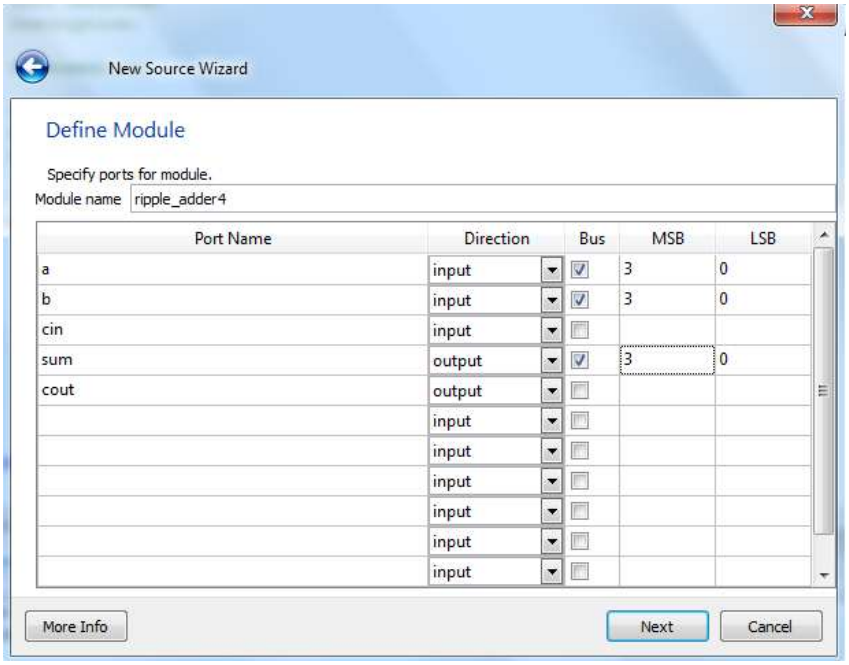


در این بخش نیز یک Verilog Module با نام ripple_adder4 می سازیم:

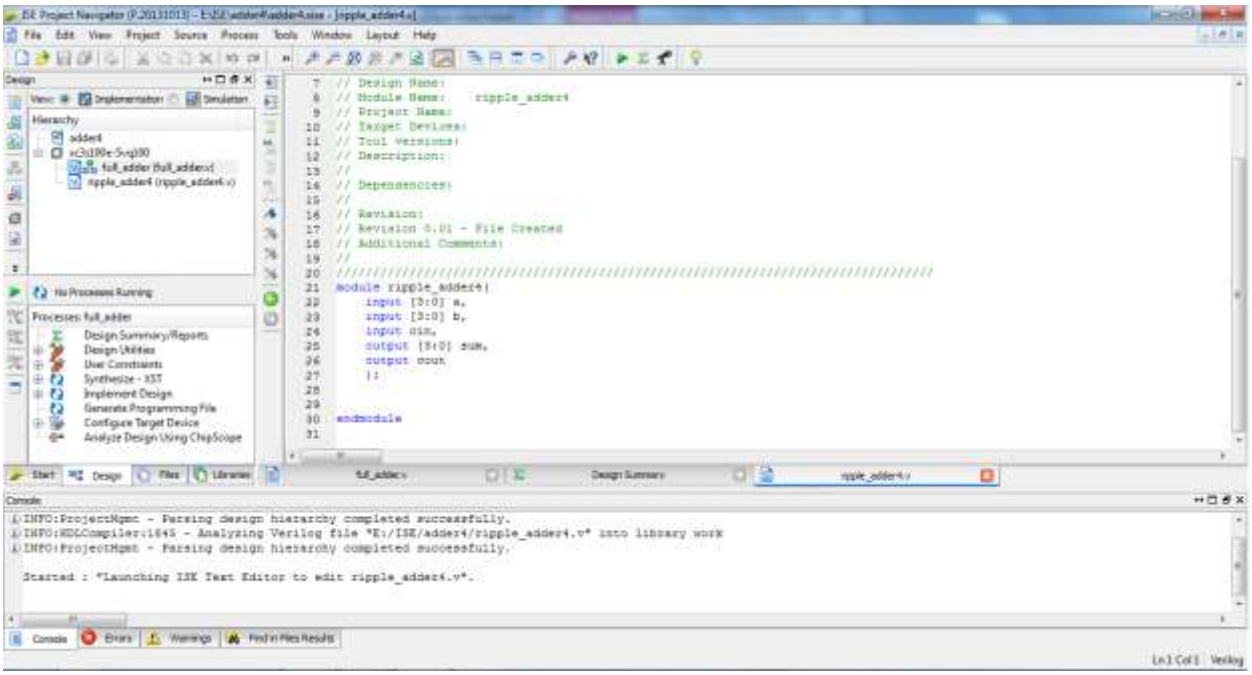


ماژولی که قصد داریم بسازیم یک جمع کننده ۴ بیتی است لذا سیگنالهای ورودی a و b و خروجی sum آن چهار بیتی هستند و سیگنالهای cin و cout همانند یک Full_Adder تک بیتی هستند.

برای سیگنالهای چند بیتی در جدول تعریف Portها، نوع Bus را انتخاب می کنیم و تعاریف را مطابق شکل زیر انجام می دهیم. به مقادیر MSB و LSB و همچنین جهت (Direction) توجه داشته باشید:



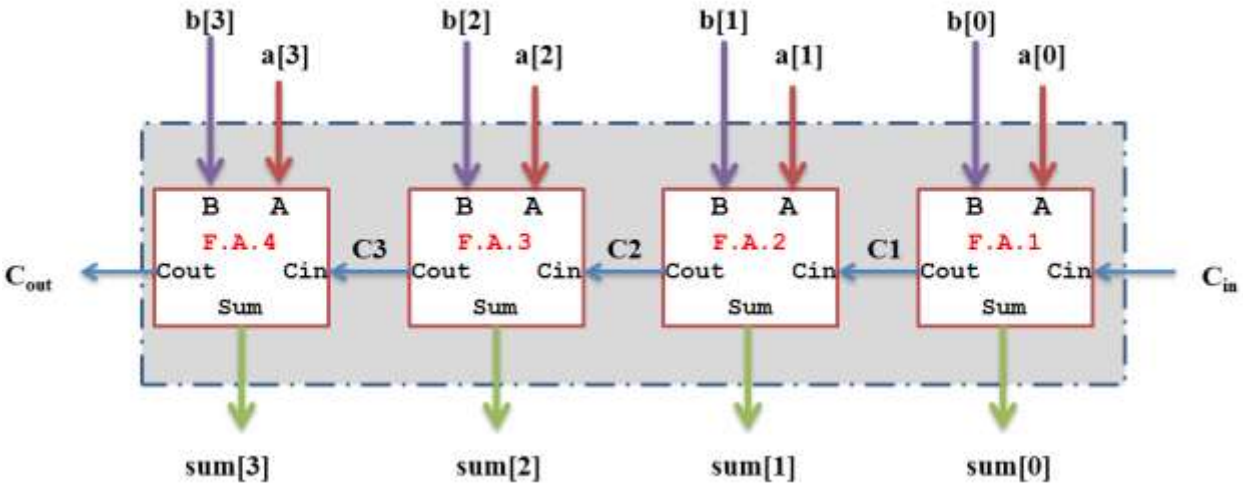
این wizard را تکمیل می کنیم تا به صفحه Editor برسیم :



چون هدف طراحی Structural یک جمع کننده چهار بیتی است، با اتصال و سیم بندی چهار Full Adder یک Ripple Carry Adder چهار بیتی می سازیم. کد این طراحی به شرح زیر است:

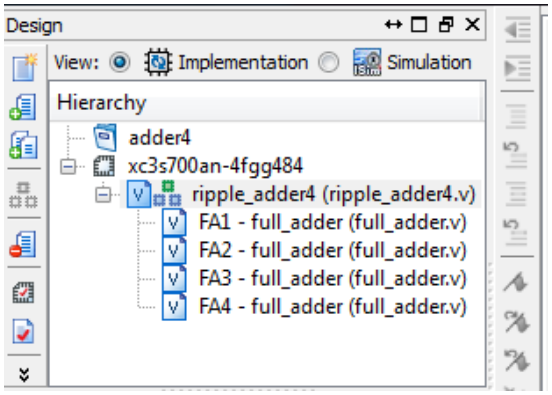
```
module ripple_adder4(
    input [3:0] a,
    input [3:0] b,
    input cin,
    output [3:0] sum,
    output cout
    );

wire c1, c2, c3;
full_adder FA1 (a[0], b[0], cin, sum[0], c1);
full_adder FA2 (a[1], b[1], c1, sum[1], c2);
full_adder FA3 (a[2], b[2], c2, sum[2], c3);
full_adder FA4 (a[3], b[3], c3, sum[3], cout);

endmodule
```

این طراحی ساختاری دقیقا معادل شماتیک زیر است. کادر نقطه چین آبی رنگ، ماژول ripple_adder4 را نشان می دهد. توجه دارید که c1، c2 و c3 در این طراحی جزو سیگنالهای ورودی خروجی اصلی نیستند و به عنوان اتصالات داخل ماژول جمع کننده چهار بیتی از آنها بهره جسته ایم. به همین دلیل جهت تطابق با Syntax زبان Verilog آنها را باید تعریف کرده و در ابتدای ماژول آنها را به عنوان Wire تعریف کرده ایم.

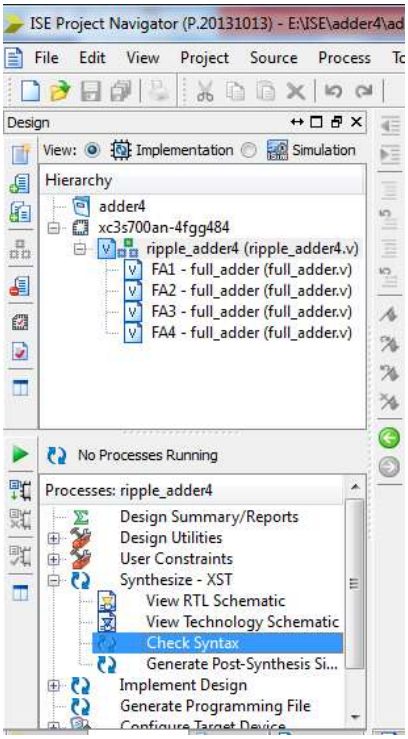


۵. سنتز و پیاده سازی روی FPGA

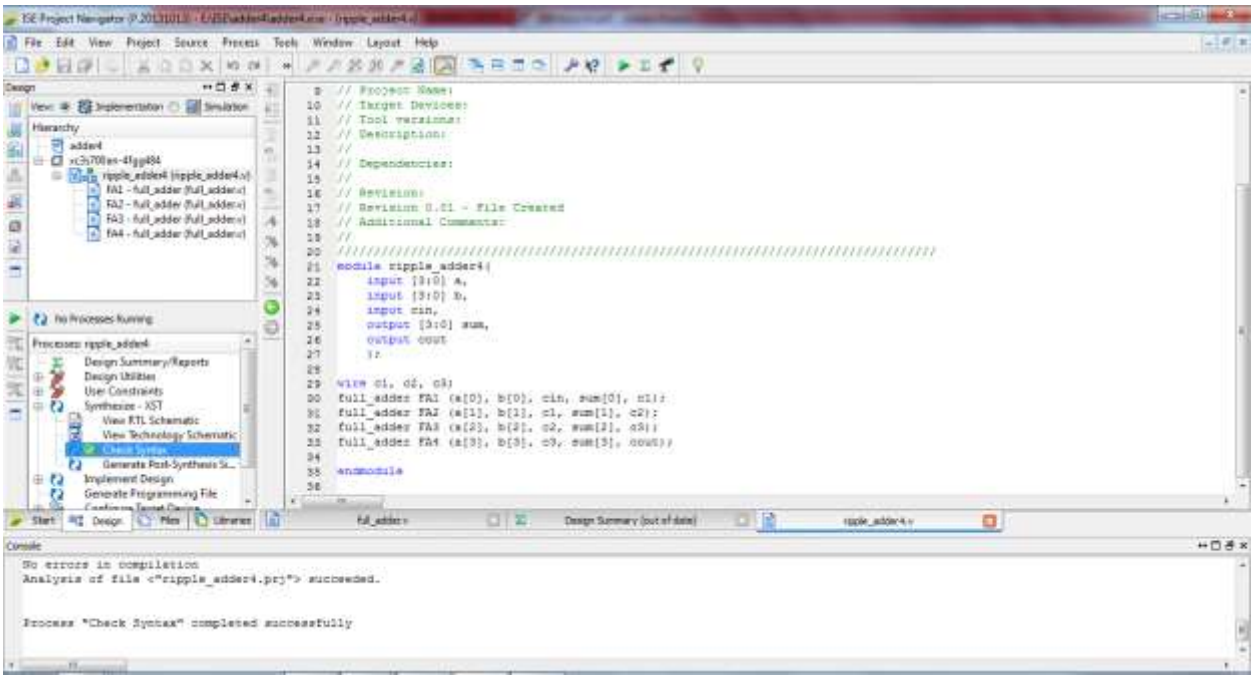
حال که طراحی خود را تکمیل کرده ایم، قصد داریم آن را سنتز کنیم و خروجی طراحی خود را بر روی FPGA مشاهده کنیم. پس از ذخیره کردن کد خود ، روی تب Design کلیک می کنیم. مشاهده می کنیم که Top Level Module در طراحی ما به ripple_adder4 تغییر یافته است که با کلیک روی علامت + در کنار اجزای ساختاری آن که شامل چهار تمام جمع کننده است لیست می شوند:



با کلیک روی ripple_adder4 و انتخاب کزینه Implementation در View گزینه های مربوط به سنتز در بخش Processes قابل مشاهده خواهد بود. روی علامت + دربخشSyntesis-XST کلیک کنید. ساختاری مثل زیر مشاهده می شود.

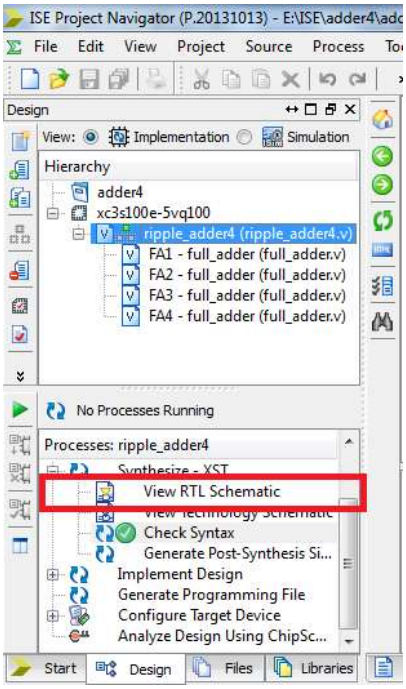


روی گزینه Check Syntax ، دابل کلیک کنید تا اجرا شود. در صورتی که کد شما خطایی نداشته باشد علامت Check (تیک) سبز رنگ روی این گزینه نشان داده می شود و در بخش Console با پیغام پایان موفقیت آمیزکنترل Syntax مواجه می شوید:

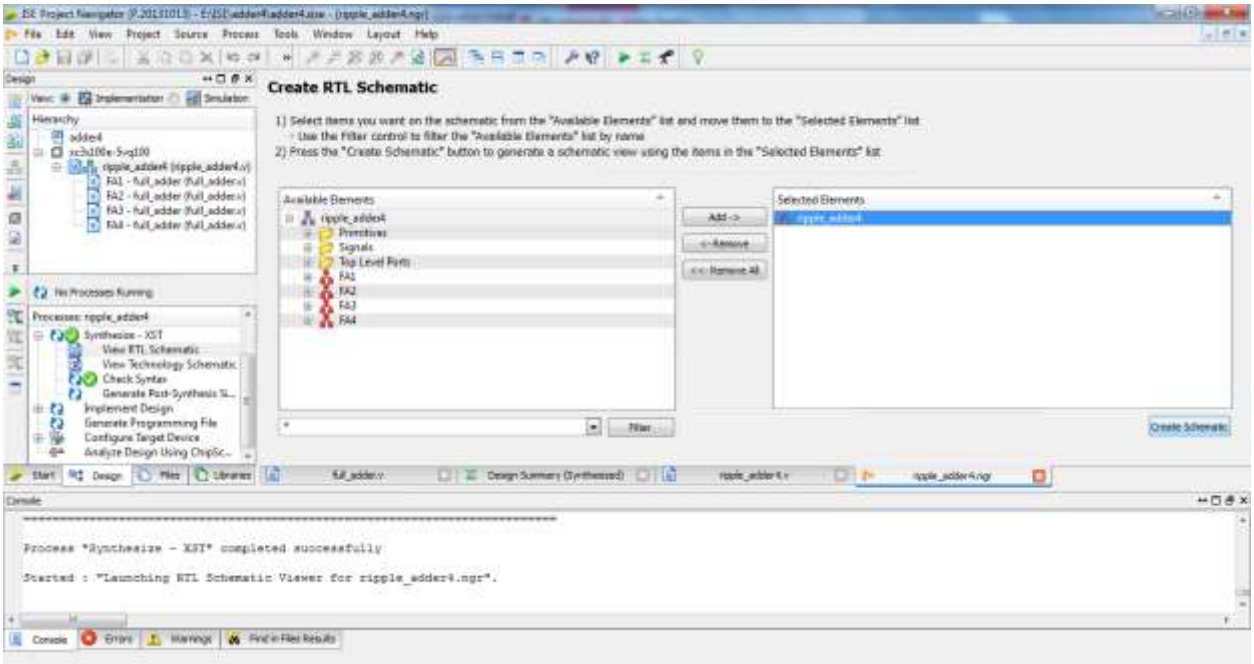


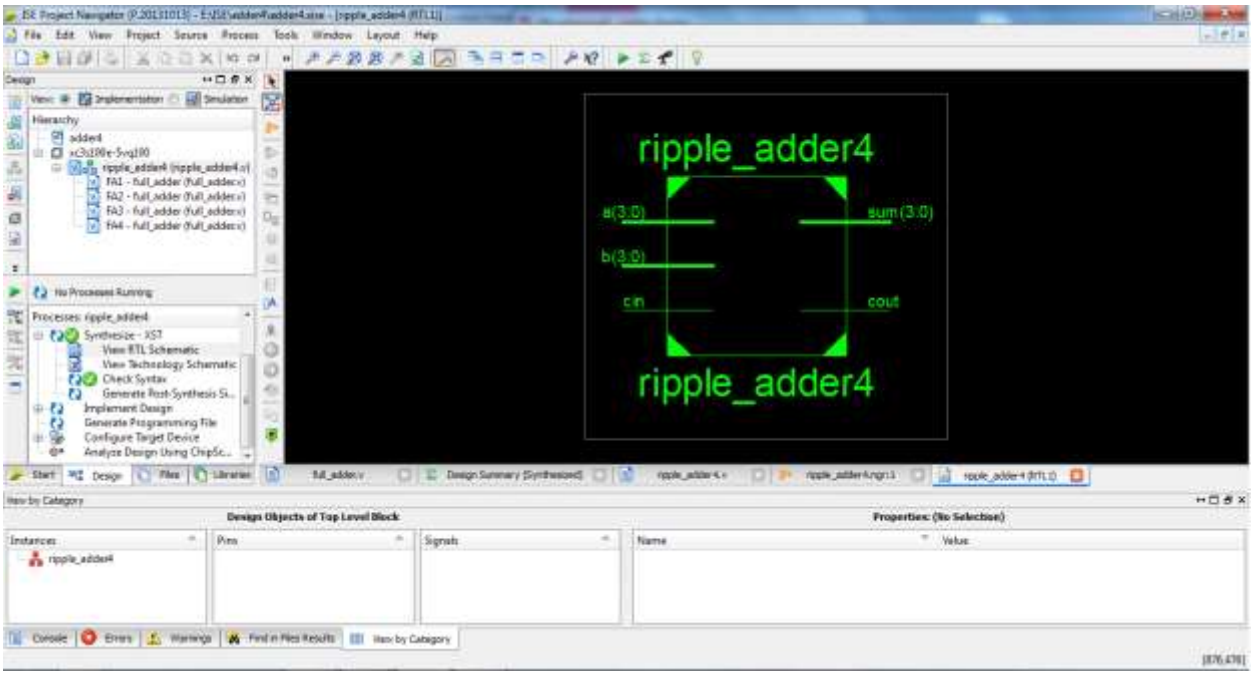
۵.۱ مشاهده شماتیک حاصل از سنتز:

برای مشاهده شماتیک مدارمان در سطح منطقی روی View RTL Schematic کلیک کنید.

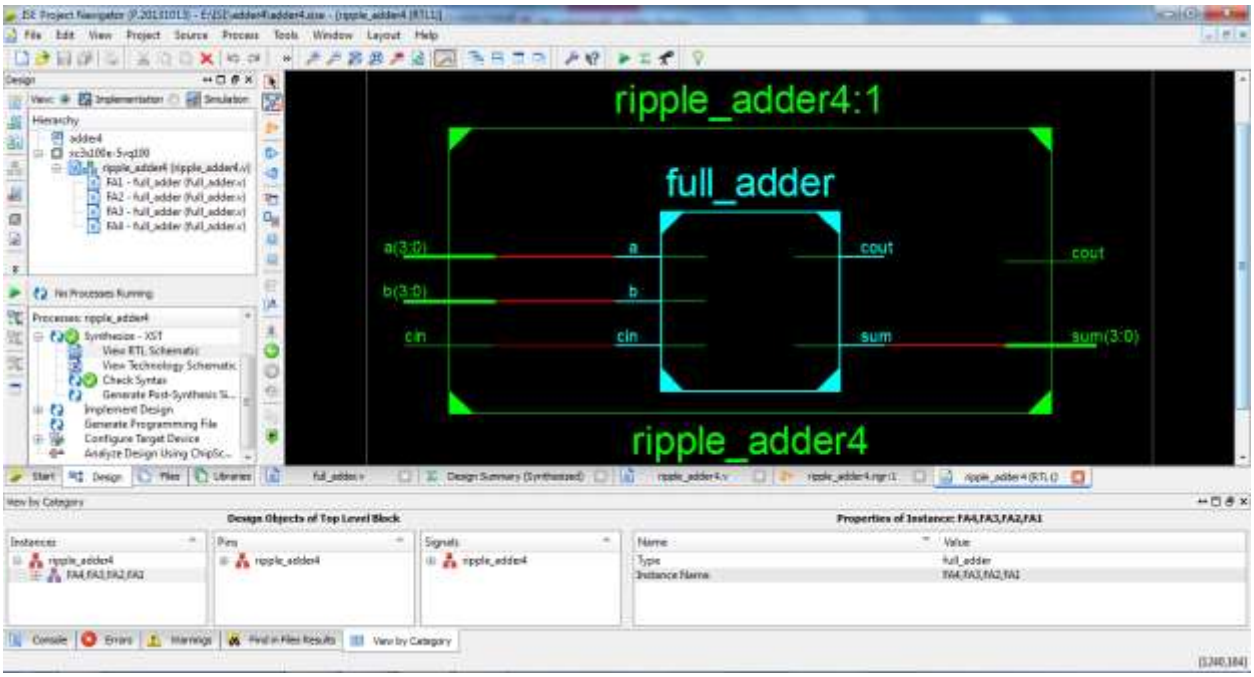


ساختار زیر مشاهده خواهد شد:

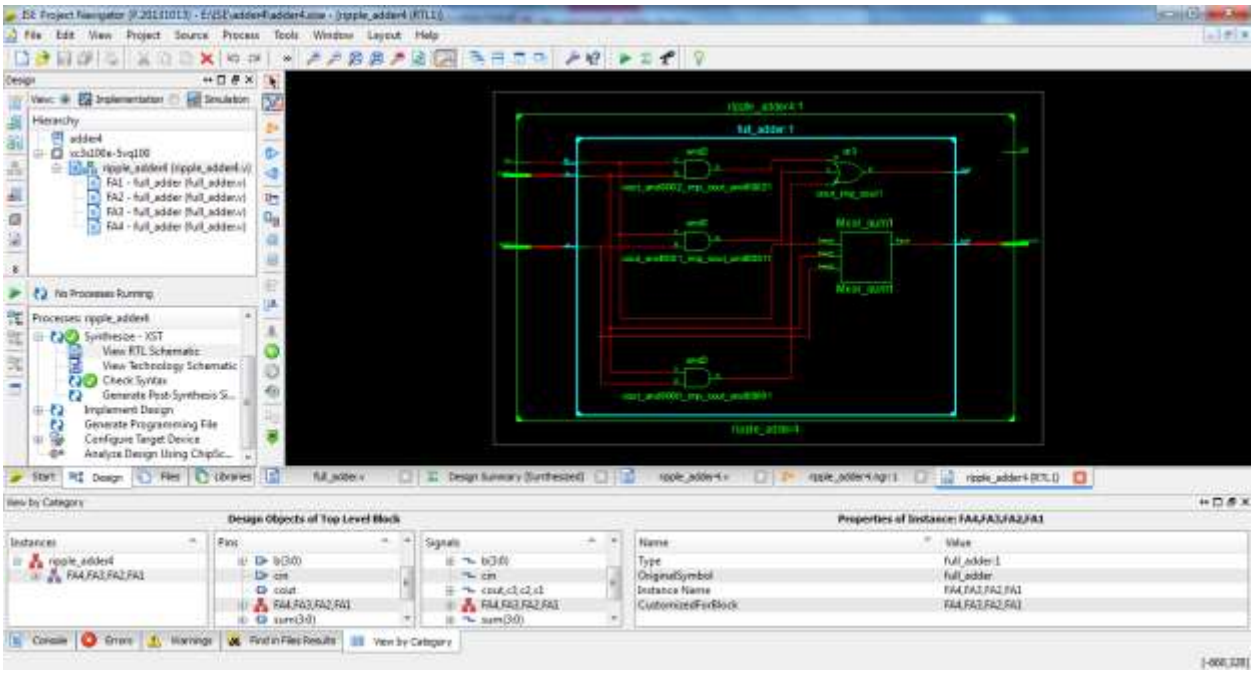




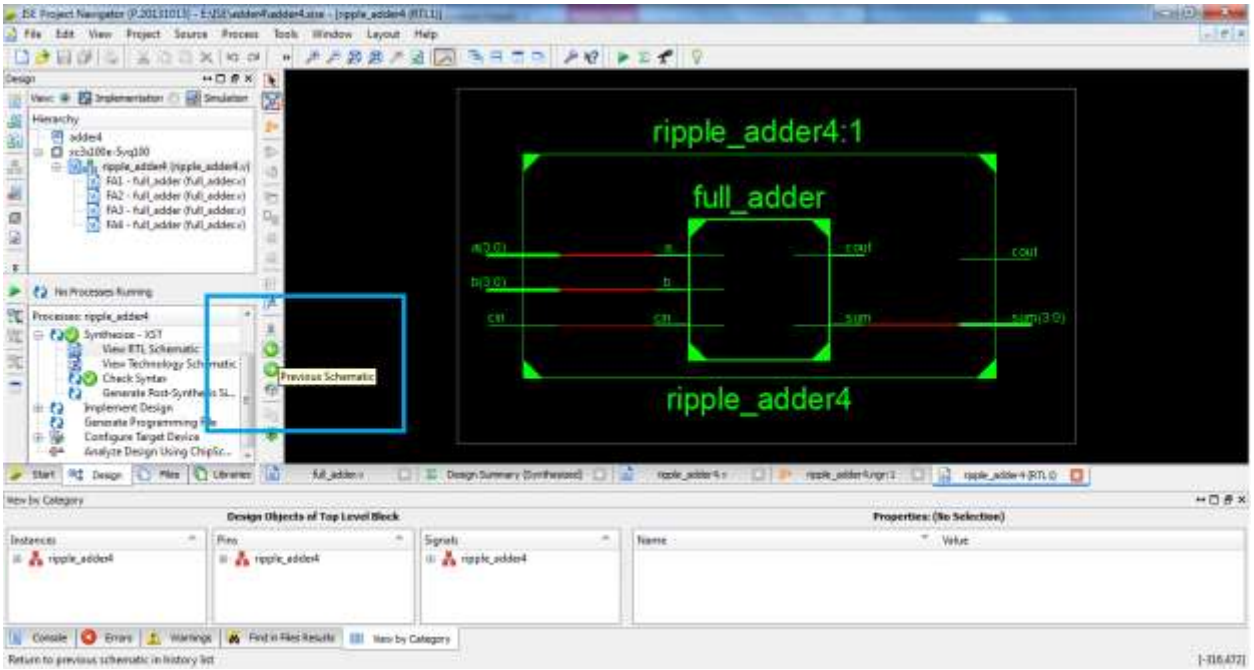
اگر روی بلوک اصلی طراحی دابل کلیک کنید ، اجزا داخلی آن دیده می شود:



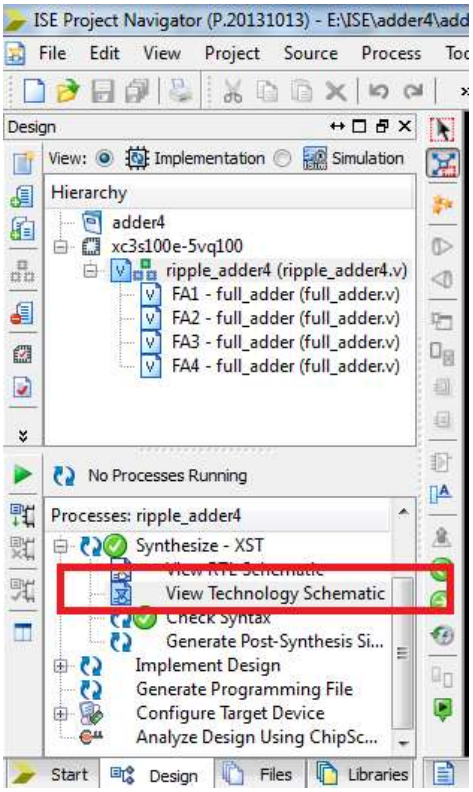
با کلیک مجدد روی بخش جمع کننده ساختار آن قابل مشاهده خواهد بود. درزیر شکل مرحله بعد آورده شده است:

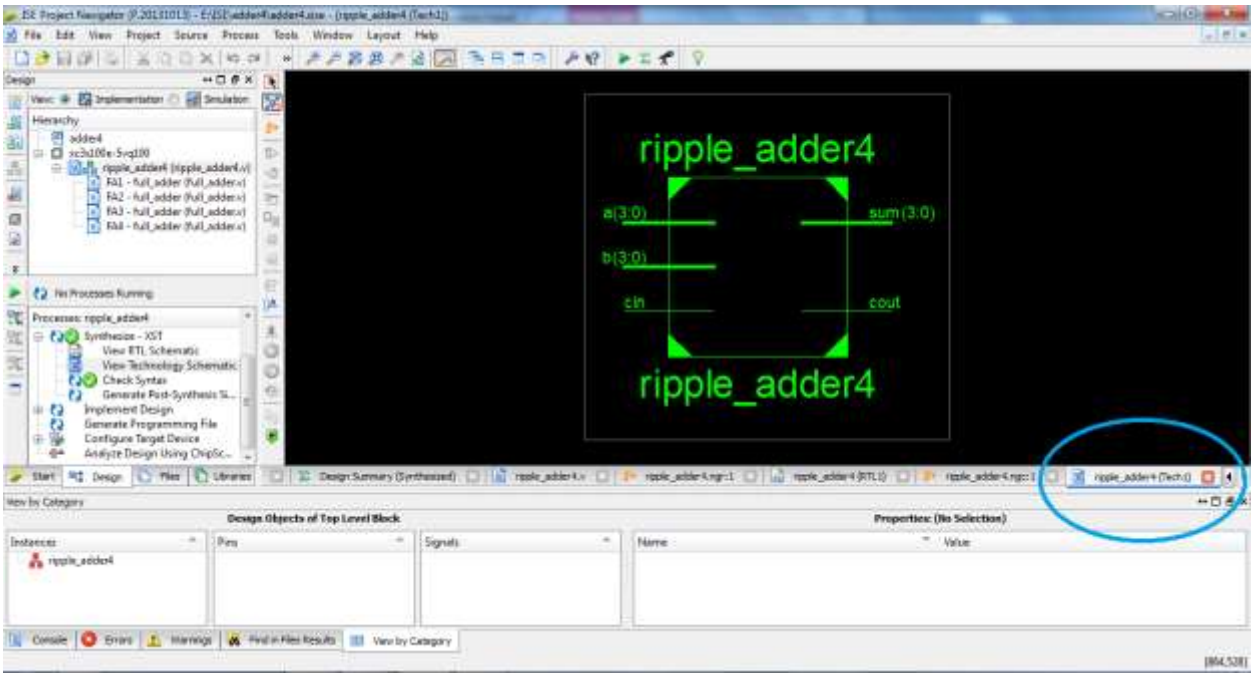


برای بازگشت به سطح بالاتر طراحی گزینه Previous Schematic را می توانید انتخاب کنید.(کادر آبی)

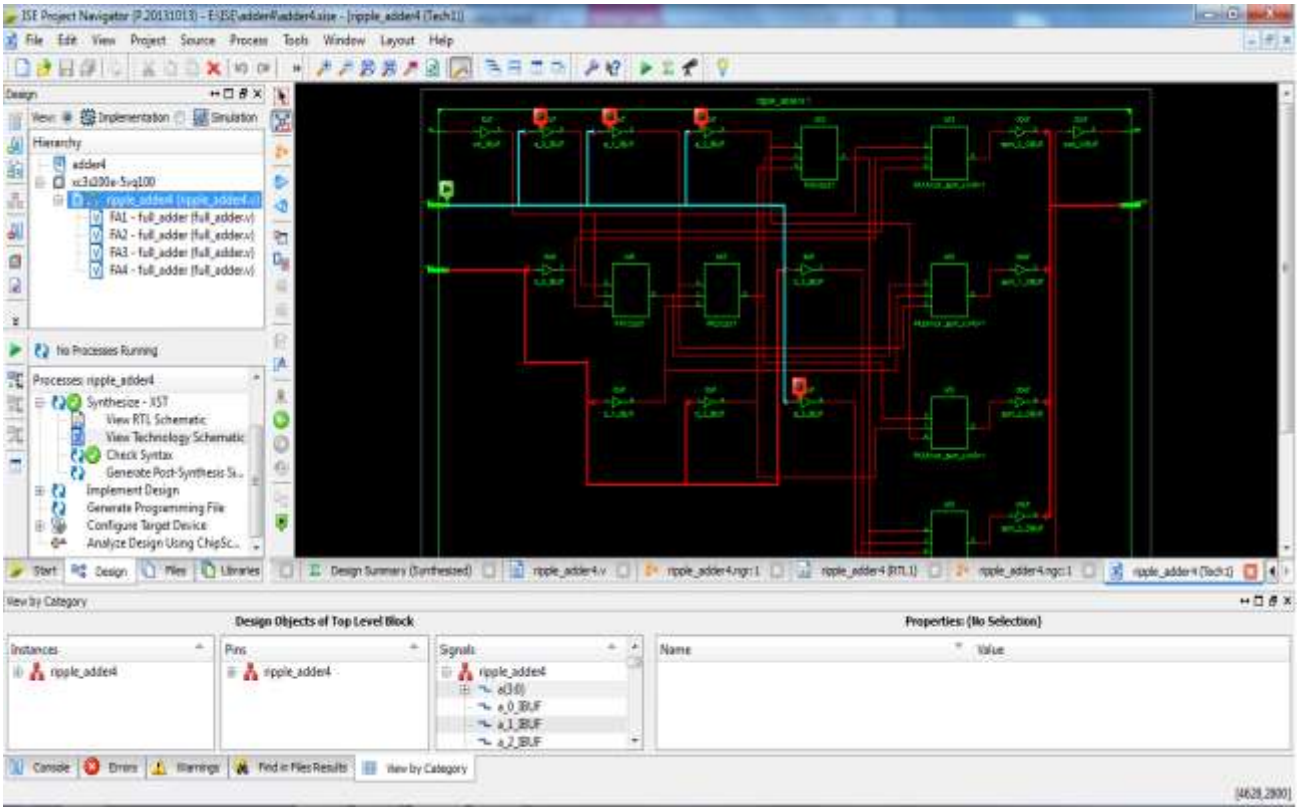


حال می خواهیم نتیجه سنتز مدار و پیاده سازی آن با استفاده از منابع FPGA را مشاهده کنیم. برای این کار روی View Technology Schematic کلیک کنید:

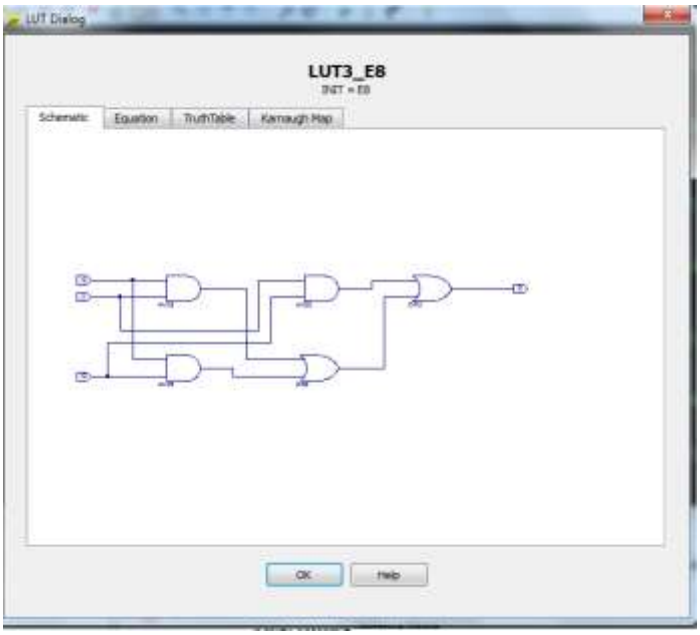


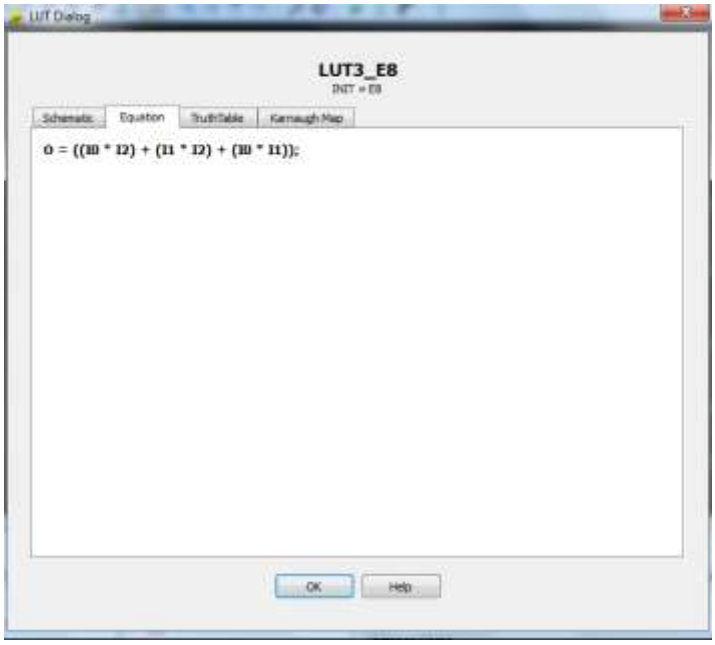


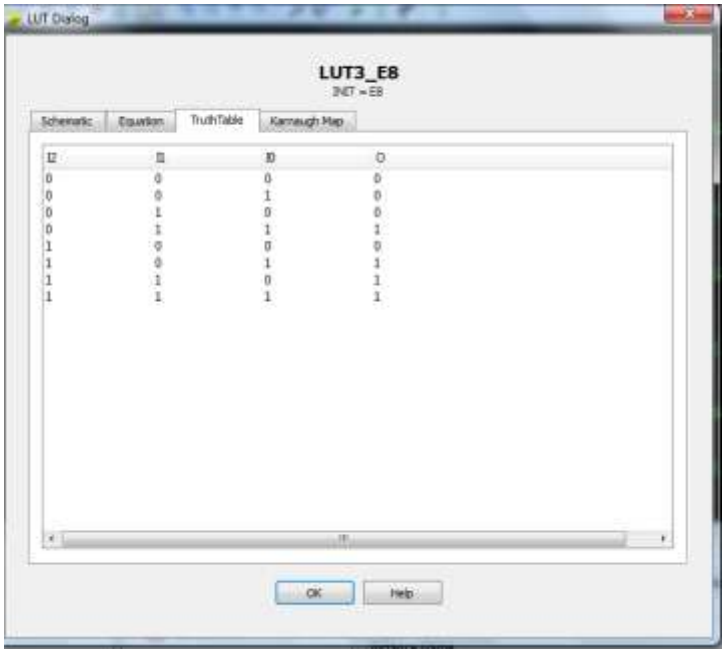
سپس روی بلوک اصلی طراحی کلیک کنید تا ساختار زیر مشاهده شود. شکل زیر پیاده سازی توابع ما توسط LUTها در FPGA را نشان می دهد.

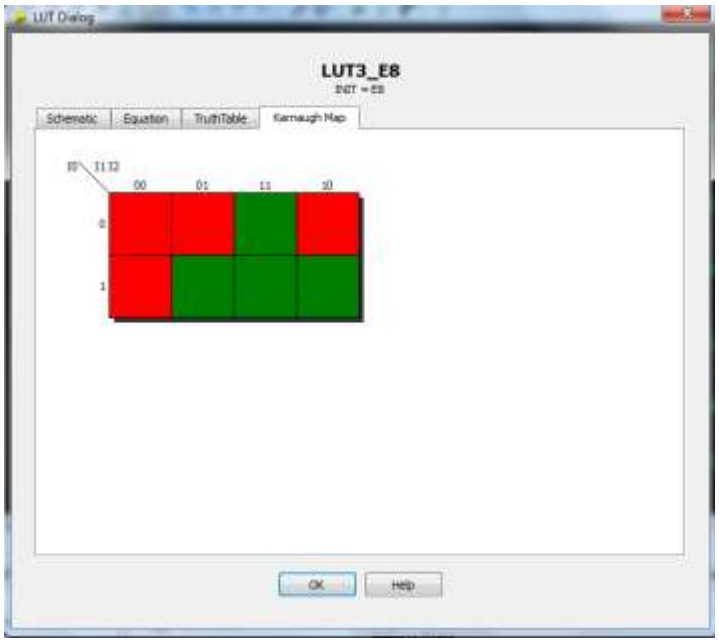


اگر روی LUT دابل کلیک کنیم پنجره ای باز می شود که جزئیات آن LUT را نشان می دهد. در این پنجره شماتیک، تابع پیاده سازی شده، جدول صحت و جدول کارنو آن قابل مشاهده است.


LUT Dialog
LUT3_E8
INIT = E8
Schematic
Equation
TruthTable
Karnaugh Map
OK
Help

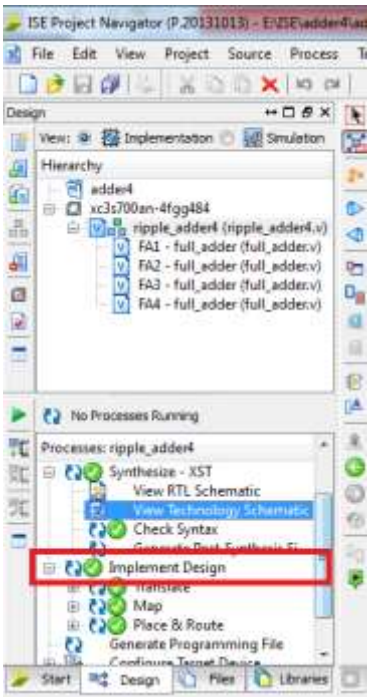

LUT Dialog
LUT3_E8
INIT = E8
Schematic
Equation
TruthTable
Karnaugh Map
O = ((I0 * I2) + (I1 * I2) + (I0 * I1));
OK
Help



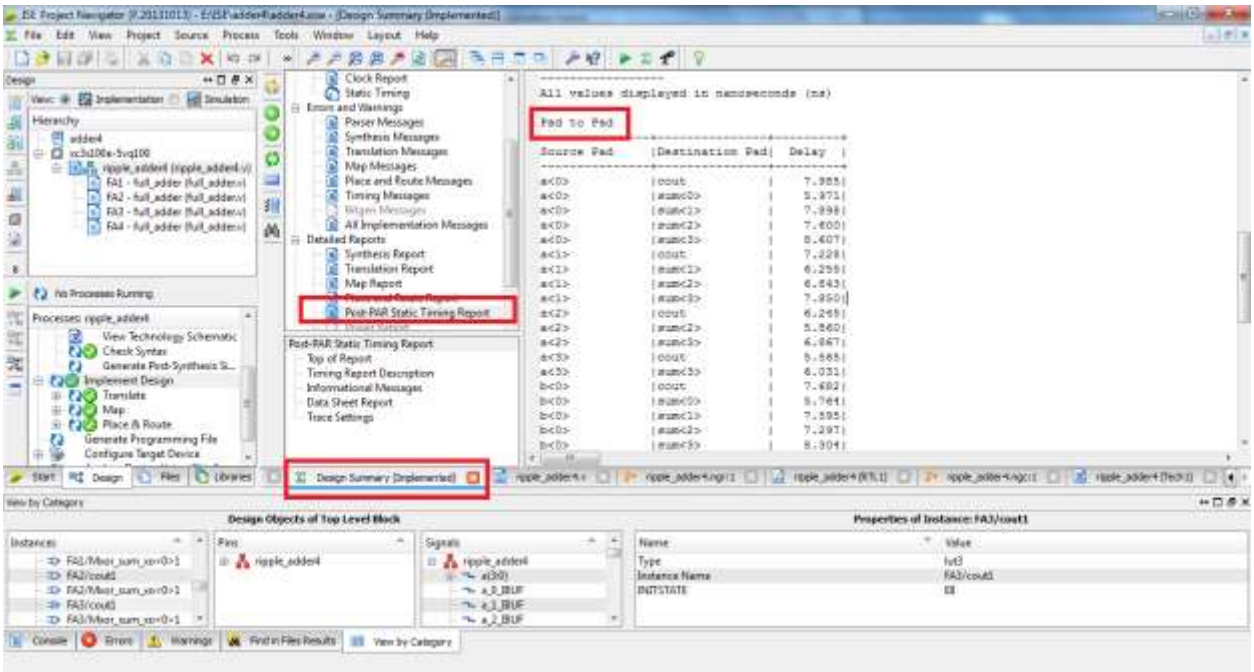


۵.۲ سنجش تأخیر مدار ترکیبی:

در این مرحله قصد داریم مسیر بحرانی مدار را تشخیص داده و آن را با دانسته های تئوری خود ملاحظه کنیم. برای این کار در قسمت Processes روی Implement Design دابل کلیک کنید. منتظر شوید تا گامهای آن تکمیل شود:



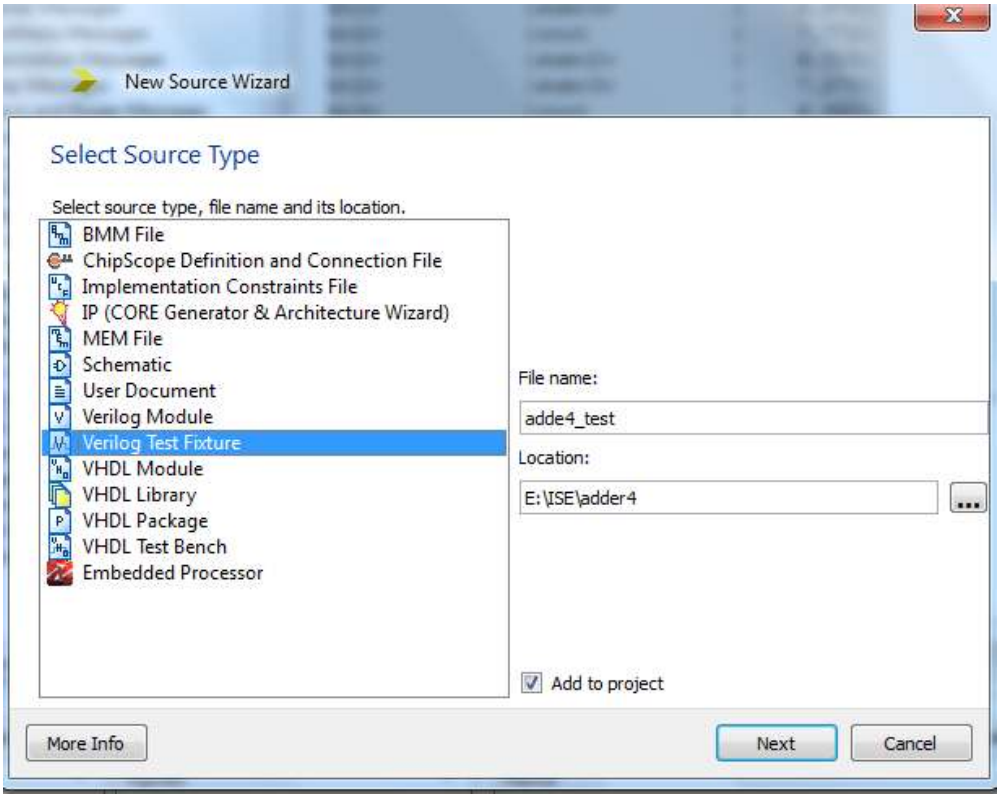
به بخش View Design Summary بروید.در بین گزارشات در بخش Detailed Reports، گزارش تاخیر های استاتیک (Static Timing Report) را انتخاب کنید. در پنجره باز شده کمی به سمت پایین اسکرول کنید تا به بخش گزارشات Pad-to-Pad برسید.



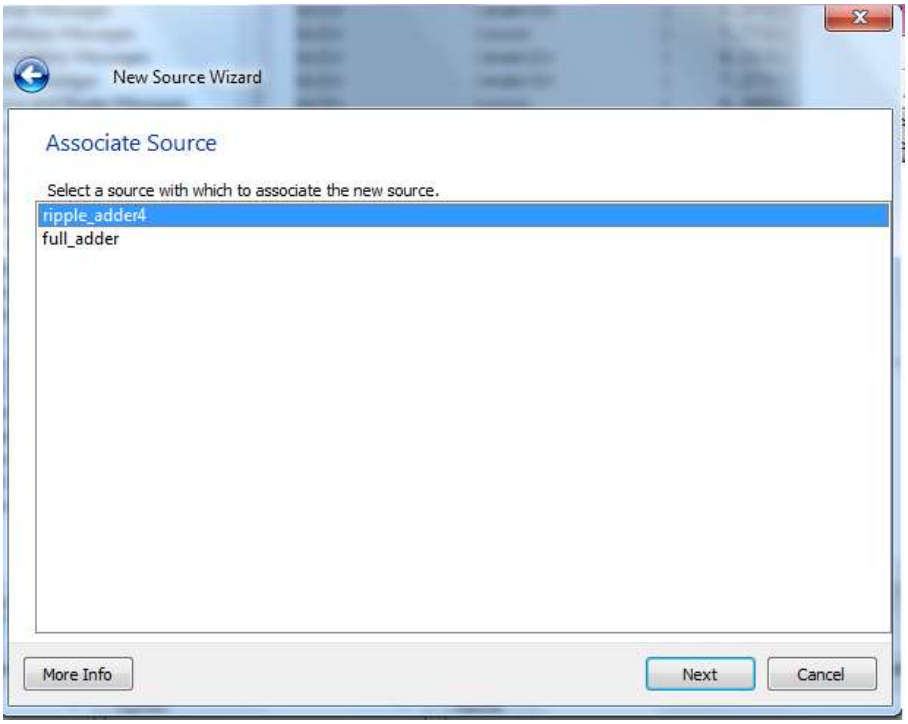
۶. شبیه سازی

حال تصمیم داریم صحت عملکرد مدار خود را بسنجیم. برای این کار ابتدا باید یک فایل که مقادیر ورودی آزمایش را به مدار ما بدهد بسازیم. این فایلها در VHDL عنوان Test Bench و در Verilog به نام Test Fixture شناخته می شود.

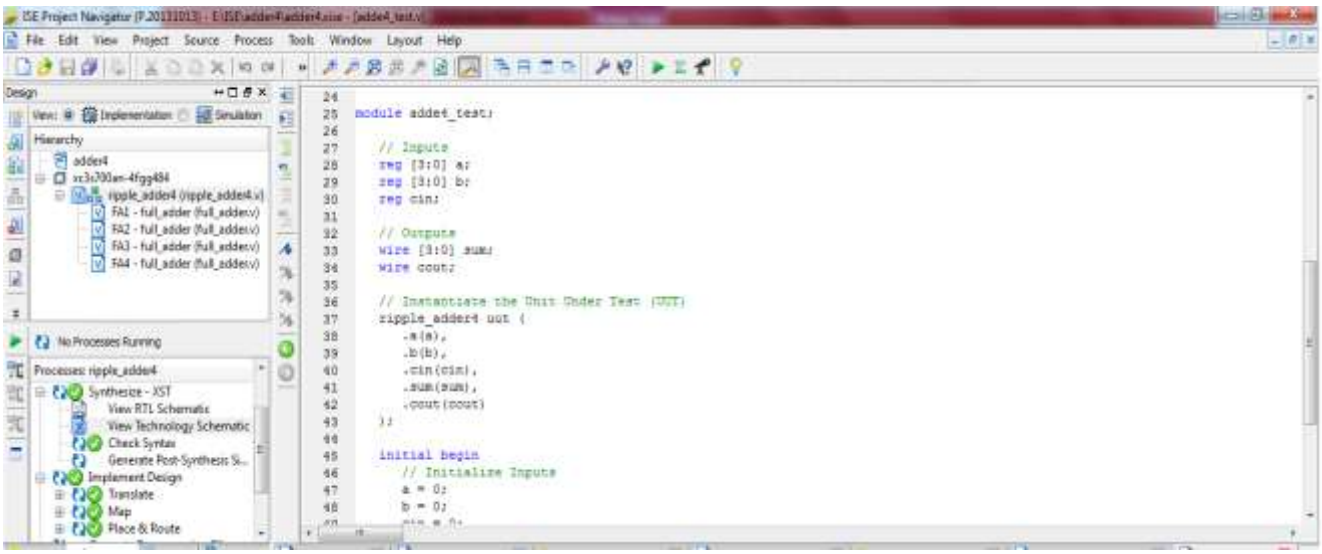
با نوجه به این طراحی خود را تا کنون با Verilog پیش برده ایم، برای ادامه روند شبیه سازی یک فایل Test Fixture به طراحی خود اضافه میکنیم. برای این کار ، گزینه New Source را انتخاب کرده و یک Test Fixture می سازیم مانند زیر:



در بخش بعد، انتخاب می کنیم که Test Fixture برای تست کدام ماژول خواهد بود که ripple_adder4 را انتخاب می کنیم:



با انجام این کارفایل Test Fixture به پروژه ما اضافه می شود:



همانطور که مشاهده می کنید این فایل یک نمونه (Instance) از ماژول adder_4 می سازد و به پایه های ورودی آن مقدار های دلخواه در طول زمان می دهد.

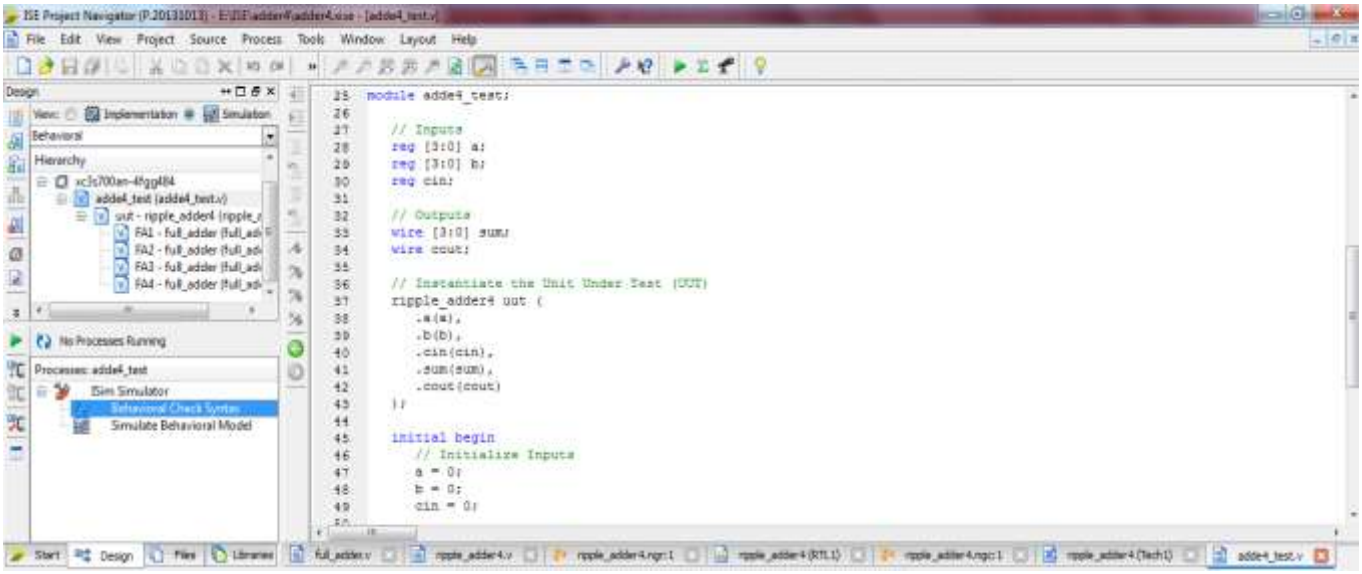
توجه داشته باشید که فایلهای Test Fixture در عمل سنتز نمی شوند، فقط در فرآیند شبیه سازی استفاده می شوند.

ما مقادیر دلخواه جهت تست به مدار می دهیم مانند زیر:

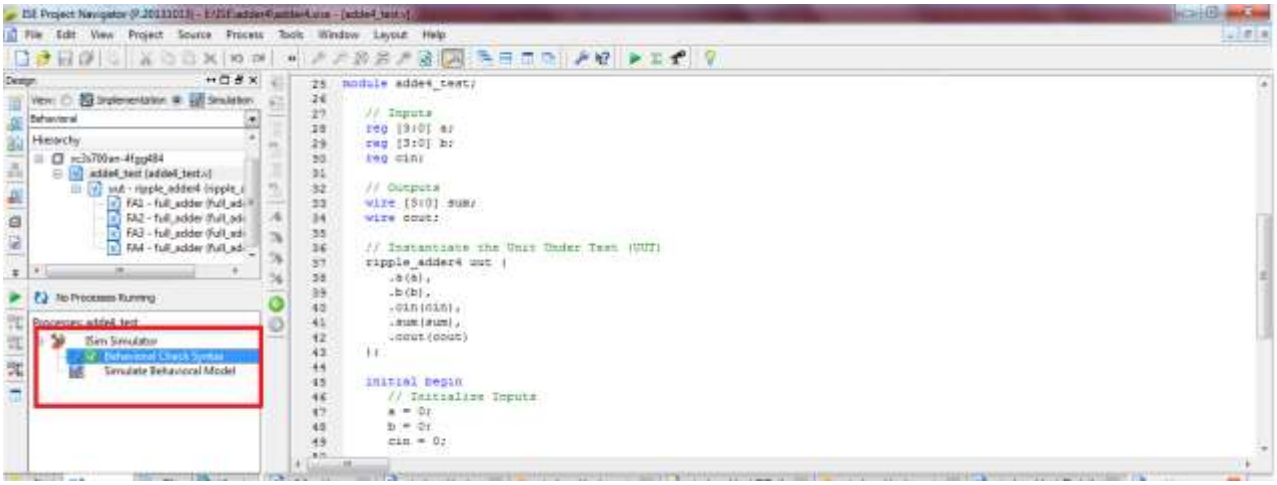
```
44
45      initial begin
46          // Initialize Inputs
47          a = 0;
48          b = 0;
49          cin = 0;
50
51          // Wait 100 ns for global reset to finish
52          #100;
53          a = 4'b0001; b = 4'b0000; cin= 1'b0;
54
55          #100;
56          a = 4'b1010; b = 4'b0011; cin= 1'b0;
57
58          #100;
59          a = 4'b1101; b = 4'b1010; cin= 1'b1;
60
61          #100;
62
63          // Add stimulus here
64
65      end
66
```

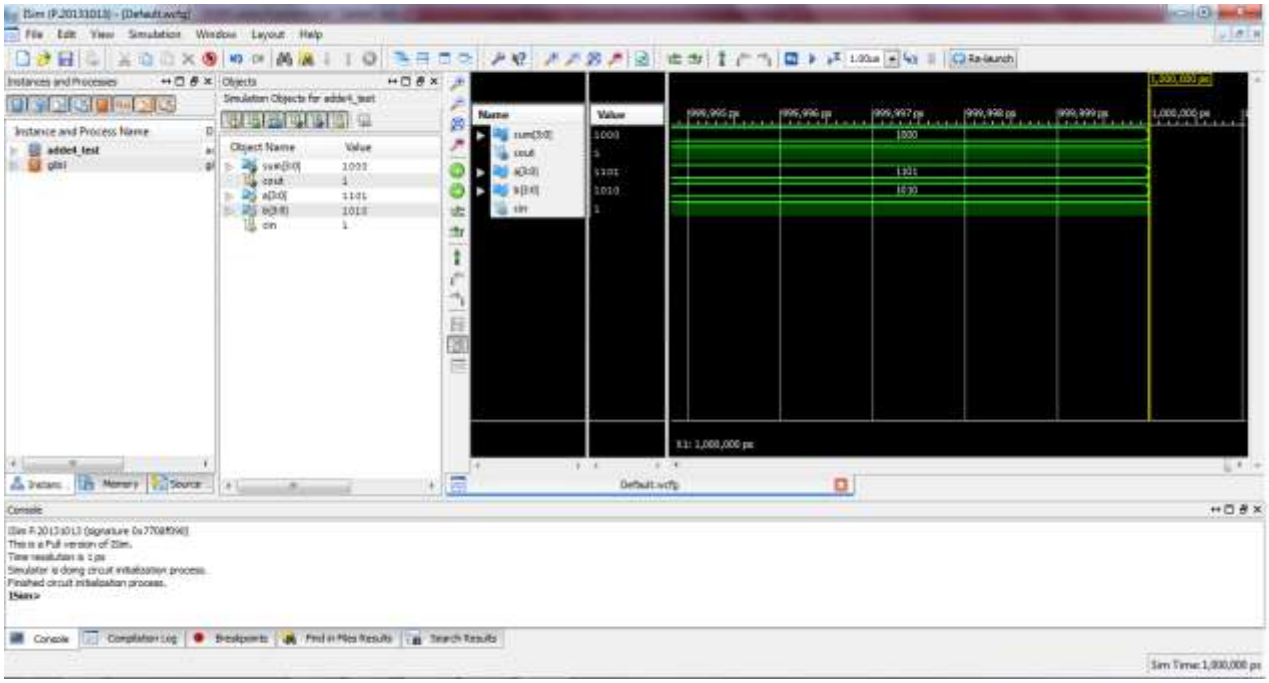
توجه داشته باشید که مقادیر a و b سیگنالهای ورودی ۴ بیتی هستند و cin تک بیتی است.

پس از تکمیل این فایل و ذخیره آن نوبت به شبیه سازی می رسد. برای این کار روی فایل Test Fixture کلیک می کنیم و گزینه Simulate Behavioral Model را انتخاب کنید.



با این کار ISE Simulator که به طور خلاصه آن را ISim می نامیم اجرا می شود:





برای مشاهده تمام تغییرات، روی آیکون ZoomOut کلیک کنید. این کار باعث می شود تمام تغییرات در بازه زمانی اجرای شبیه سازی به طور کلی قابل مشاهده باشد.

اگر توجه کنید، همانطور که در شکل زیر مشاهده می شود، شبیه سازی این مرحله با فرض تاخیر صفر در گیتها و عملکرد ایده آل آنهاست، لذا با تغییر ورودی، بلافاصله خروجی نیز تغییر می یابد.