

5.5.3 EDA 开发综合实例 3：SmartDesign 的使用

在 Libero 中，除了可以编写程序实现相应设计外，还可通过可视化操作方式（“SmartDesign”软件），对现成的模块进行连线和拼装，实现特定的功能。

下例采用可视化方法实现 1 位全加器，再改造为 2 位串行进位加法器，操作过程既有通过编写代码建立模块，也有调用现成模块，还有通过 IP 核创建实例模块，并对多个模块进行拼装和测试。

5.5.3.1 使用半加器构造全加器

通过半加器来构造全加器的方法在 4.7.3 中讨论了，以下的模块及其连接均基于图 4-21 完成。

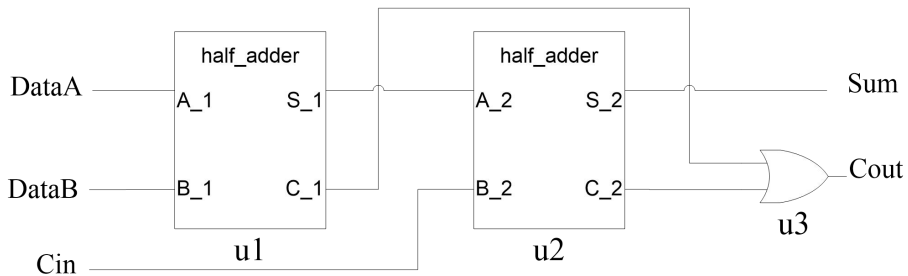


图 4-21 构造全加器 full_adder

1. 新建工程

打开 Libero SoC，选择“Project”菜单的“New Project”命令，输入项目名称、选择项目存放路径，选择语言 Verilog（如图 5-62 所示）。设备的选择同 5.5.2 的实例 2。

New project

← **Project details**
Specify project details

Project Details

Device Selection

Device Settings

Design Template

Project name: Adders_Smart

Project location: D:/Liberoprj

Description:

Preferred HDL type: Verilog

☐ Enable block creation

2. 新建 SmartDesign 设计

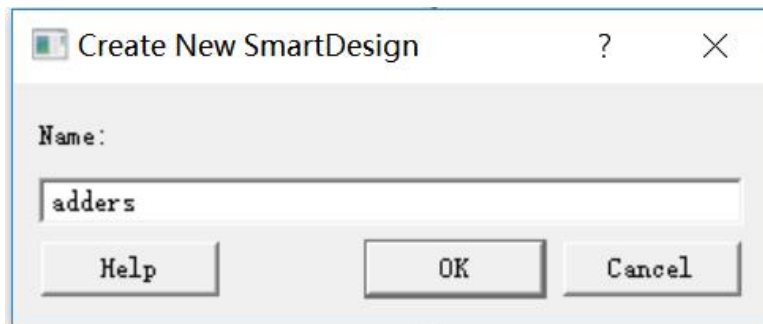
在“Design Flow”中选择“Create SmartDesign”，如图所示：

Design Flow

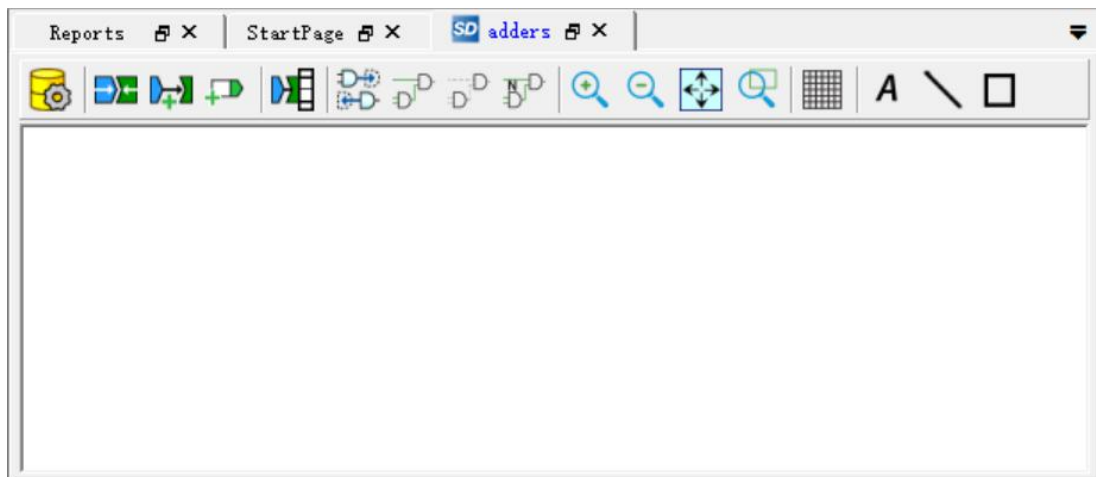
Please select a root

Tool	
	▶ Create Design
SD	Create SmartDesign
	Create HDL
SD TB	Create SmartDesign Testbench
	Create HDL Testbench

在弹出的对话框中输入设计名称，如图所示：

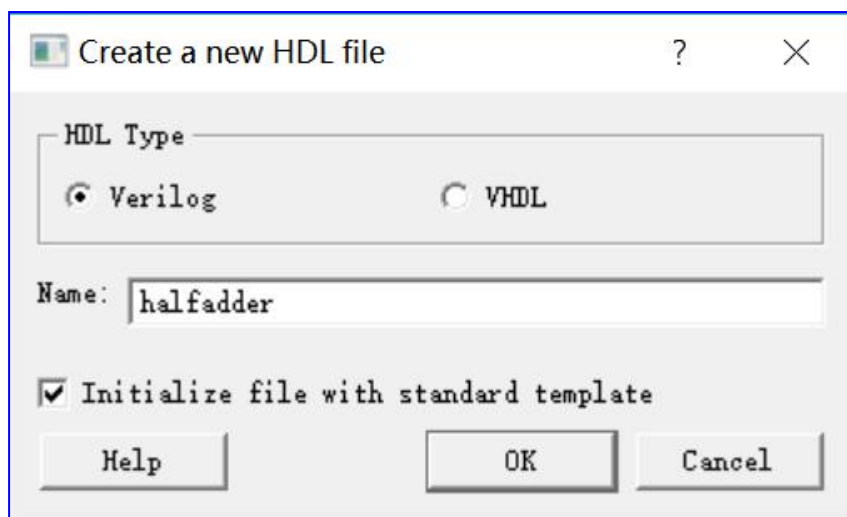


工作区中会显示打开了“adders”设计的画布，但画布是一片空白，如图所示。

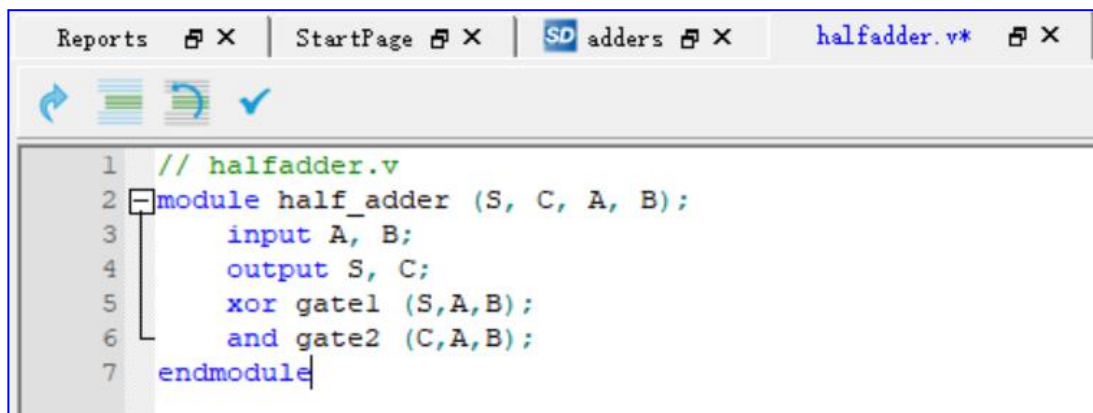


3. 添加半加器模块

在“Design Flow”中选择“Create HDL”按钮，输入并新建 Verilog 程序文件。注意不用输入文件后缀名，如图所示：



在打开的文件中输入半加器程序代码，代码同 4.7.3 中的半加器设计。



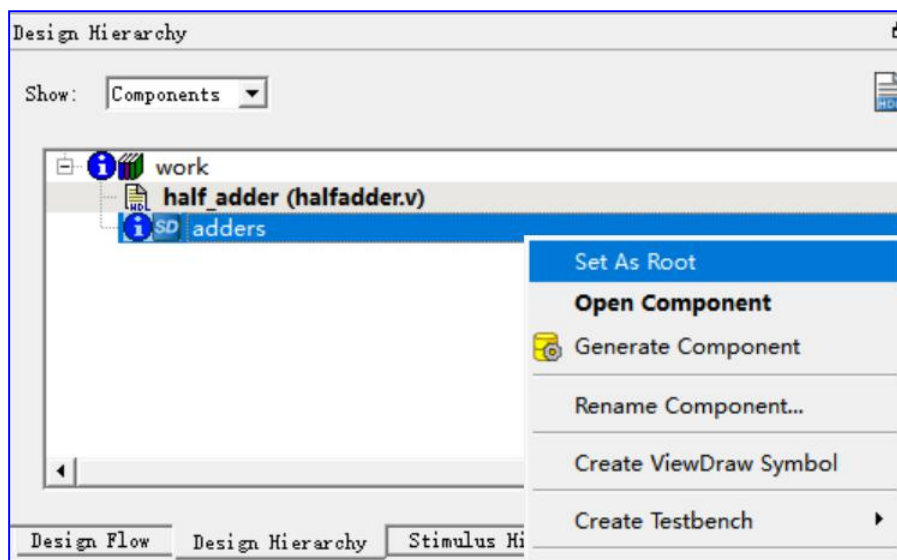
代码:

```

// halfadder.v
module half_adder (S, C, A, B);
    input A, B;
    output S, C;
    xor gate1 (S,A,B);
    and gate2 (C,A,B);
endmodule

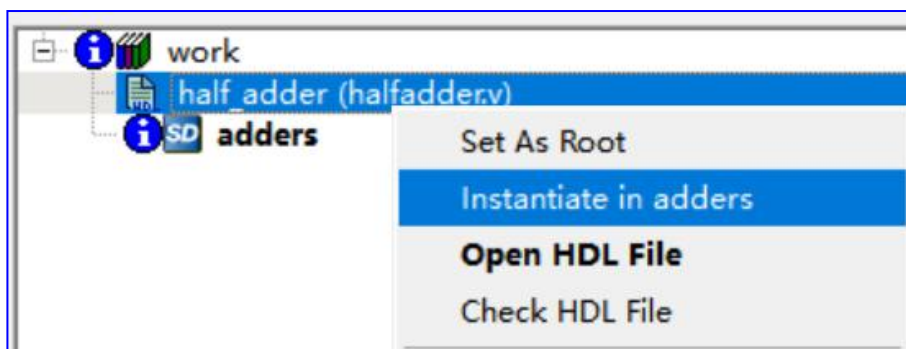
```

切换到“Design Hierarchy”窗口，项目会把第一个建立的模块或设计作为“根”(Root)，并加粗显示，如果项目中的根不是“adders”，则对着“adders”按右键，选择“Set As Root”进行修改，如图所示：

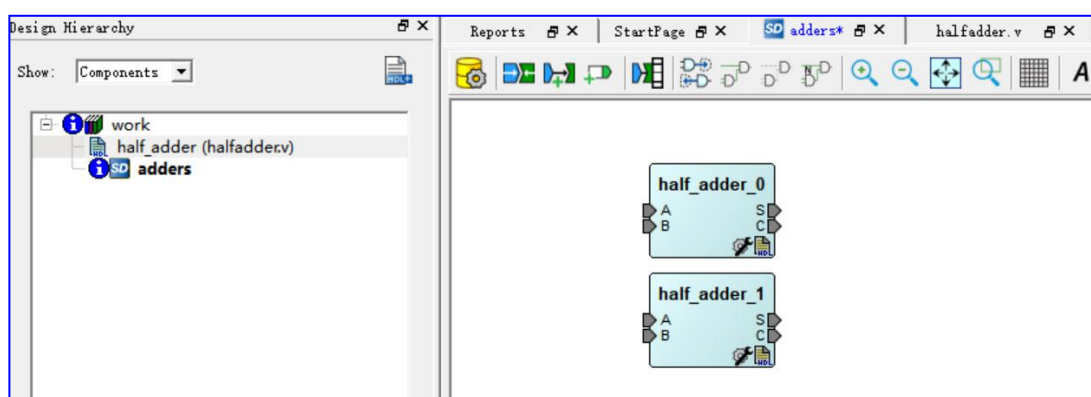


4. 在设计中添加“半加器”模块

对着“half_adder”模块右键，选择“Instantiate in adders”。

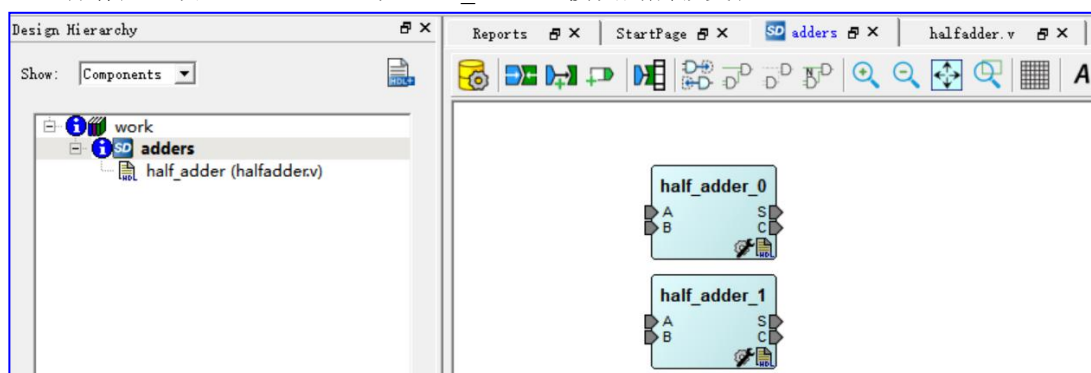


或者按着“half_adder”模块拖拽至“adders”的画布（Canvas）上。重复操作两次，在此需要两个半加器来构造全加器。操作结果如图所示：



在图中可看到，添加两个模块后，设计中自动设定了模块的实例化名称（half_adder_0 和 half_adder_1），直接点击可修改其实例名，双击可打开对应的源程序文件（halfadder.v）。

保存后，留意“adders”与“half_adder”模块的层级变化。

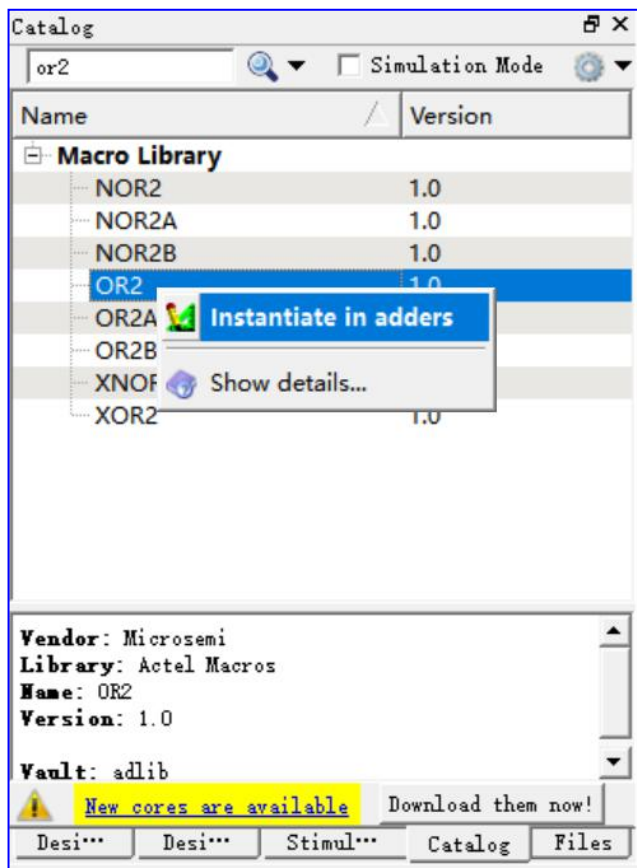


5. 在设计中添加“或”模块

切换到“Catalog”窗口，该处列出了 Libero SoC 提供的各种现成可使用的 IP 核，包括宏单元（Macro Library）、基本块（Basic Blocks）等多种分类，资源丰富。

在搜索栏输入“or2”（也可直接在“Macro Library”列表中找），可找到在此需要用到的“or2”宏单元（即 2 输入“或”门）。点击右键，选择“Instantiate in adders”（如图所

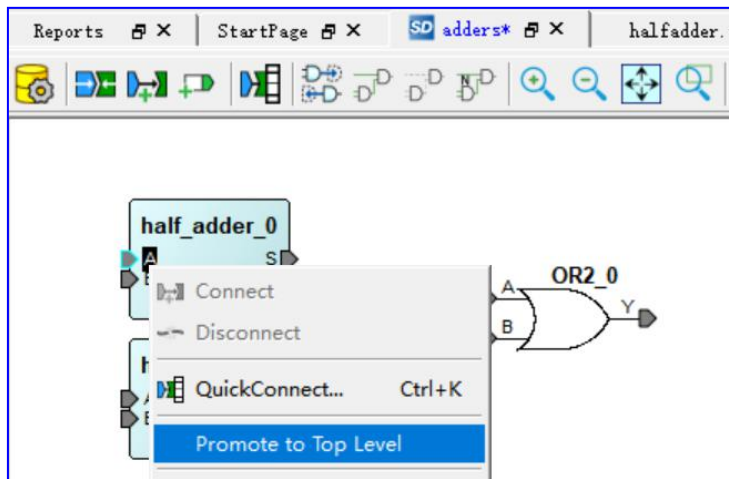
示), 或者通过拖拽操作, 添加至“adders”的画布上。



6. 连线到顶层

整个画布就是一个“芯片”的设计, 而刚才添加的模块只是该“芯片”的内部零件, 故需要定义这些子模块中哪些端口是连接到整个设计的对外(输入/输出)端口上。

对着“half_adder_0”模块的“A”端口按右键, 选择“Promote to Top Level”(如图), 可把该子模块的端口连接至顶层。

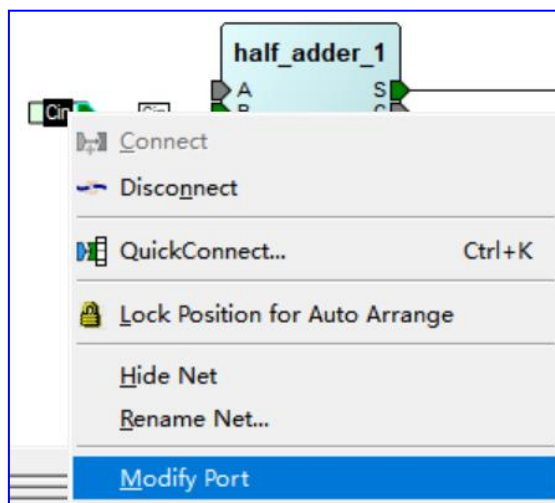


通过类似操作，将“half_adder_0”模块的“B”端口、“half_adder_1”的“B”端口
“half_adder_1”的“S”端口、“or2_0”模块的“Y”端口连接到顶层。

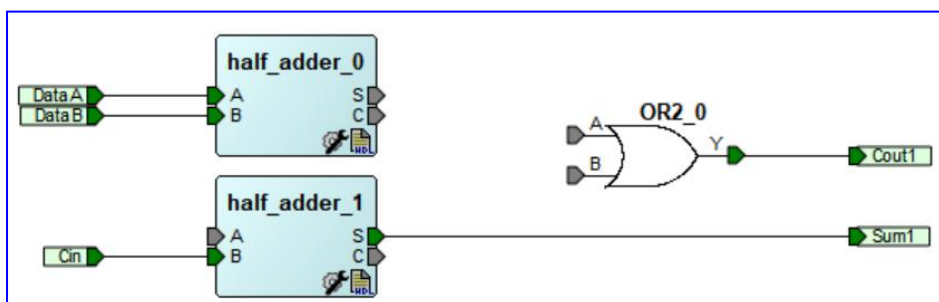
由于“half_adder_0”和“half_adder_1”都有名为“B”的端口，故连接到顶层时会有命名上的冲突，如图所示为弹出的对话框，如果选择“是”，则连接到一个新的端口，系统自动给这个端口改名（如“B_0”），如果选择“否”，就会将“half_adder_0”和“half_adder_1”的“B”端口都连接到同一个对外的顶层端口上。



可修改端口的名称，对着要修改的端口名按右键，选择“Modify Port”（或直接双击），输入新的端口名称（如图所示）。

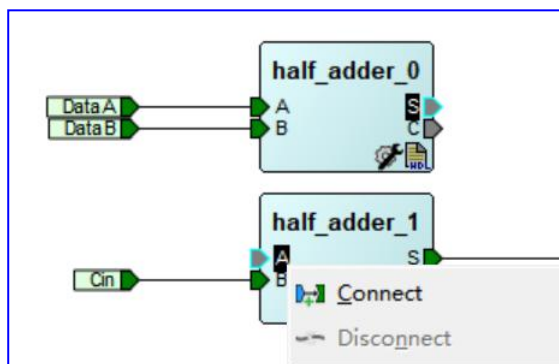


连接并修改端口名后的结果如下图所示：



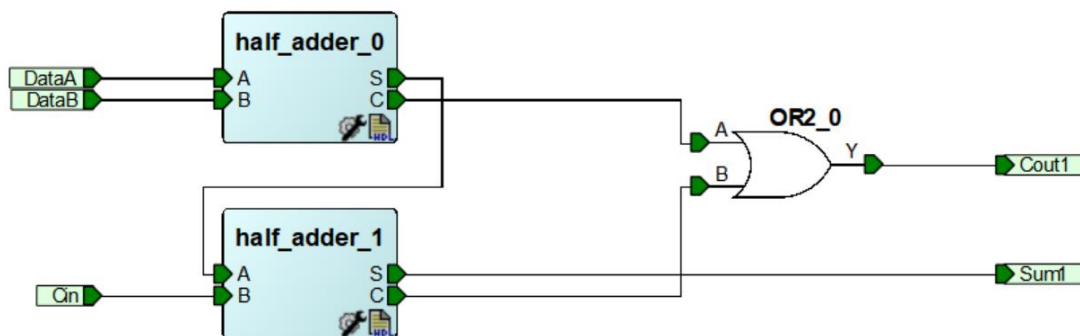
7. 进行内部连线

选择“half_adder_0”的“S”端口，按下“Ctrl”键同时，点击选择“half_adder_1”的“A”端口，同时选中了两个端口，点击右键，选择“Connect”命令，就可把这两个选中的端口进行连线。如图 5-75 所示：



用同样的方法，把“half_adder_0”的“C”端口与“or2_0”的“A”端口、“half_adder_1”的“C”端口与“or2_0”的“B”端口进行连接。

模块的布局看起来可能会比较乱，连线可能会产生视觉上的重叠，可调整各模块的位置达到较好的显示效果，如图所示：



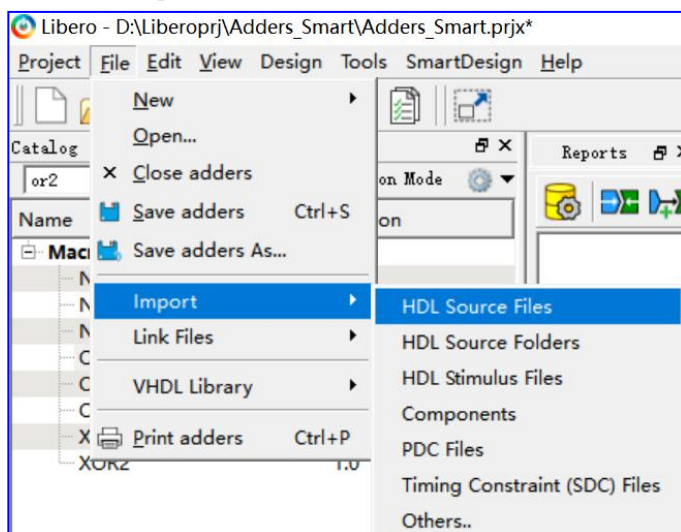
至此，一个全加器的设计已完成。

5.5.3.2 与现有的全加器对比

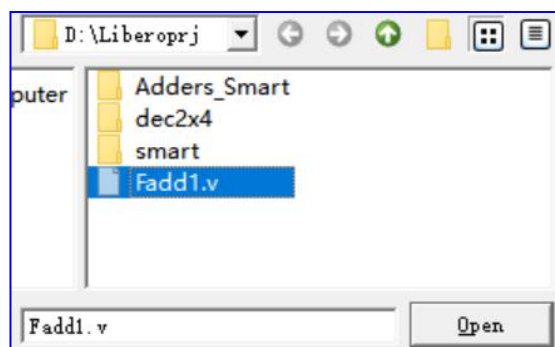
在 5.5.1 综合实例 1 中，已经设计了全加器的模块并保存在磁盘中（“Fadd1.v”文件），在此可调用原来已经完成的模块，将两种设计思路都放到设计中，一起进行验证并对结果进行对比。

1. 导入全加器模块

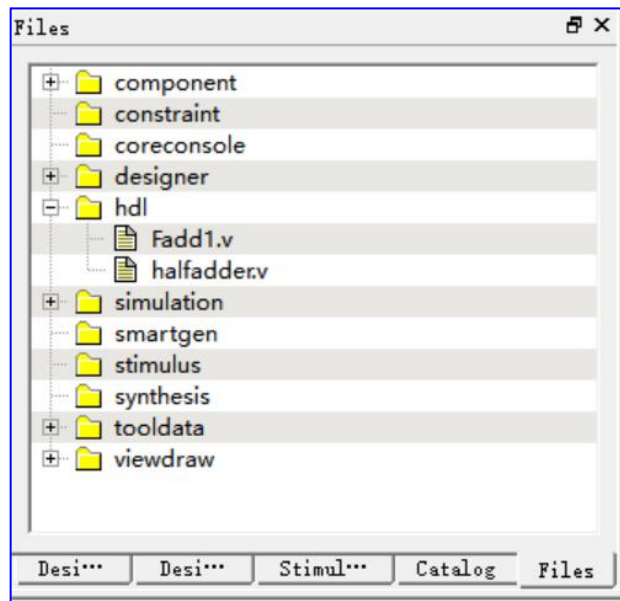
在“File”菜单运行“Import Files”命令（如图）。



找到已保存了的全加器设计文件“Fadd1.v”，点击“Open”按钮导入文件（如图）。



导入文件后，Libero SoC 会复制该文件到项目文件夹的“/hdl”子文件夹下，对该文件中的代码进行改动，不会影响原文件。



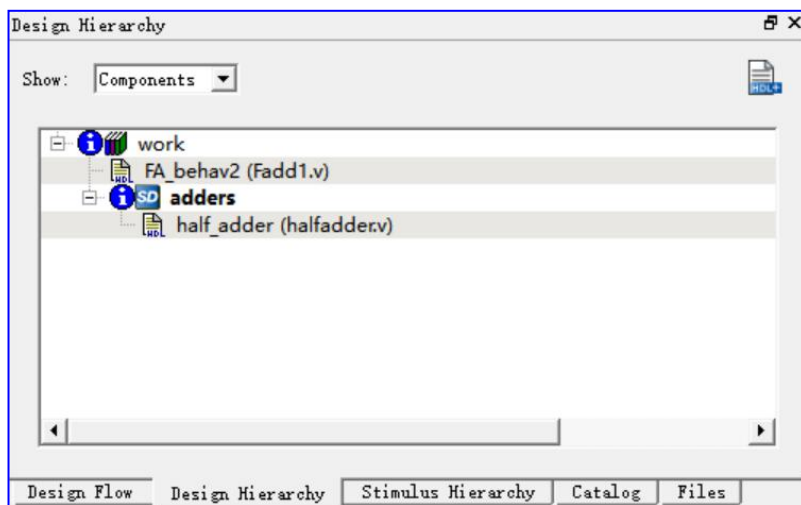
“Fadd1.v” 的代码如下：

```
// Fadd1.v
module FA_behav2(A,B,Cin,Sum,Cout);
input A,B,Cin;
output Sum,Cout;
reg Sum,Cout;

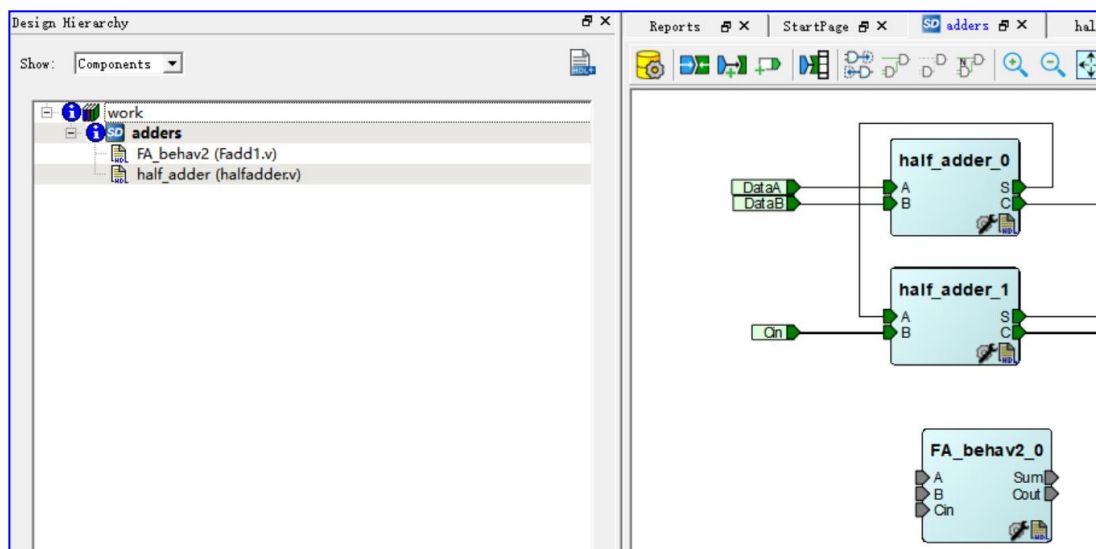
always
@(A or B or Cin) begin
{Cout,Sum}=A+B+Cin;    // 使用拼接运算符
end
endmodule
```

2. 添加全加器模块到设计中

导入文件后，该模块会显示在项目层次结构图中，如图所示：

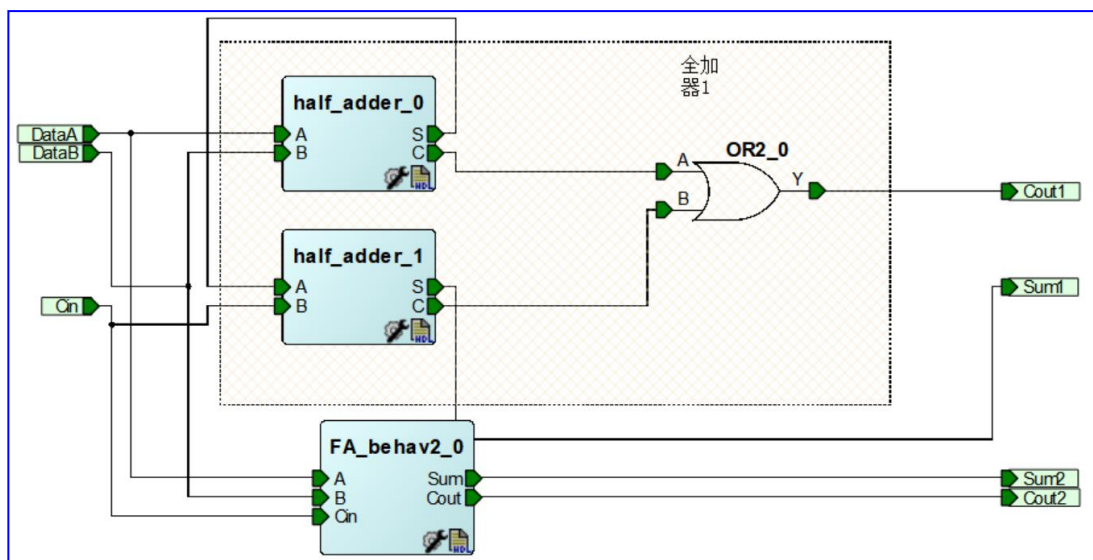


选择“FA_behav2”模块，添加到“adders”设计中（实例化），并保存，层次结构也会发生变化，如图所示：



3. 连线

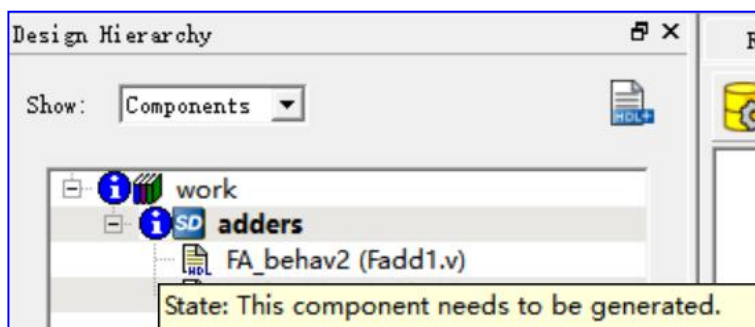
由于“FA_behav2”模块是一个独立的整体，不需要进行内部连线，故全部端口都连接到顶层设计。为了对比两个设计，将“FA_behav2_0”实例的输入（A，B，Cin）与“全加器 1”（半加器组装的）的输入都接到同一个端口（DataA，DataB，Cin），输出端口直接接到顶层设计，如图所示：



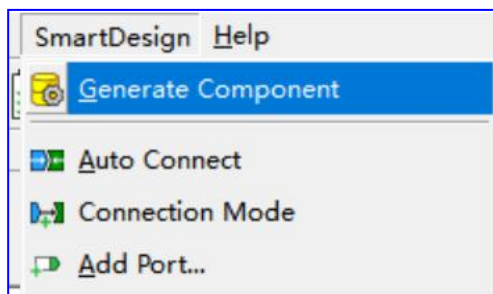
注：在图中，由于全加器 1 是由多个部件组装而成，为了更好的显示效果，加入了一个长方体和文字，这些元素可在画布内添加，而不会影响原来的设计。

4. 生成设计

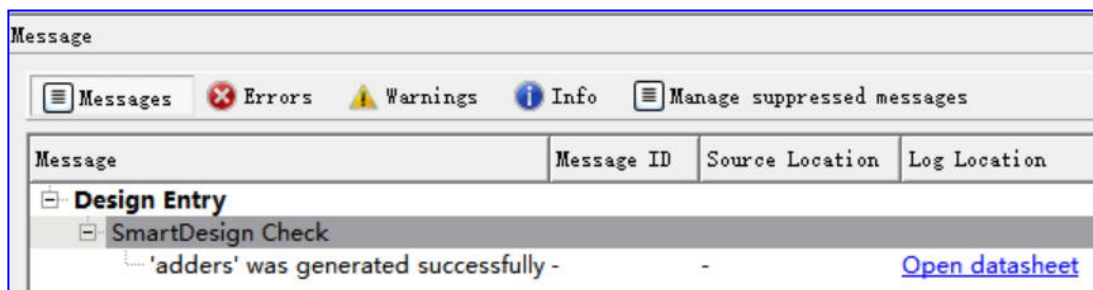
留意 Work 库前面的状态信息，提示目前设计还没有生成。



选择“SmartDesign”菜单的“Generate Component”命令（如图所示），或者工具栏上的  图标，又或者对着画布区空白处按右键，选择“Generate Component”命令。

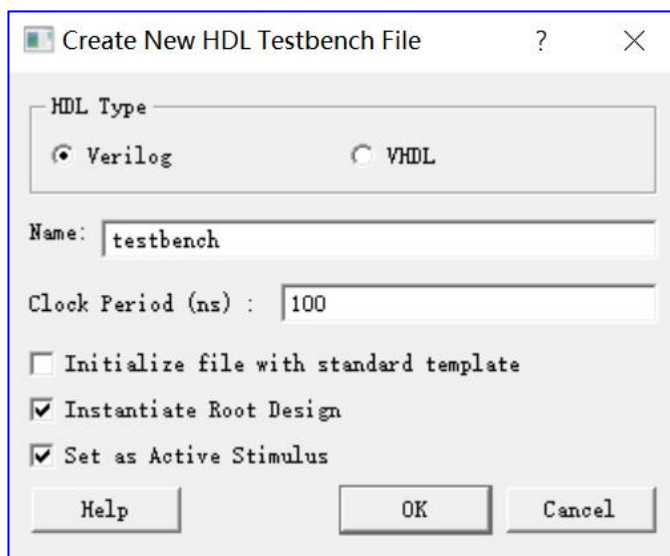


Message 窗口显示生成结果。

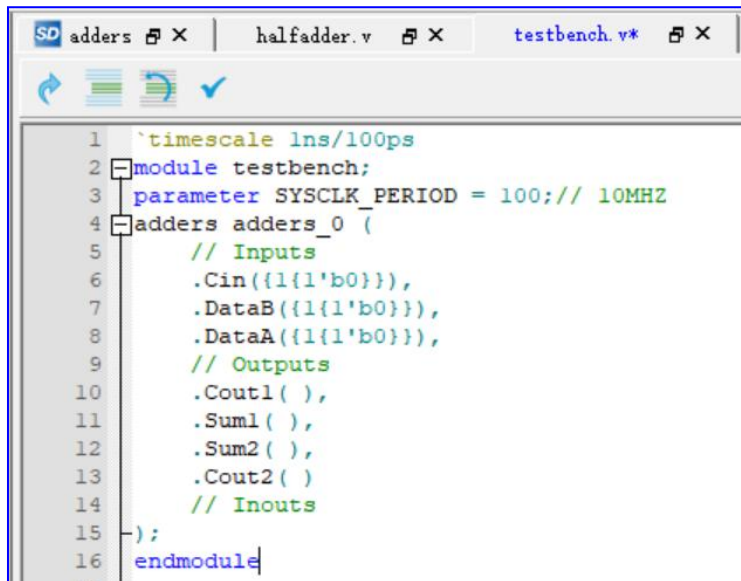


5. 编写测试平台

点击“Design Flow”切换回项目设计流程，点击“Create HDL Testbench”按钮，在打开的对话框中输入测试平台的文件名，如图所示：



Libero 软件会提供默认的文件模板。



```
1 `timescale 1ns/100ps
2 module testbench;
3     parameter SYCLK_PERIOD = 100; // 10MHZ
4     adders adders_0 (
5         // Inputs
6         .Cin({1'b0}),
7         .DataB({1'b0}),
8         .DataA({1'b0}),
9         // Outputs
10        .Cout1( ),
11        .Sum1( ),
12        .Sum2( ),
13        .Cout2( )
14        // Inouts
15    );
16 endmodule
```

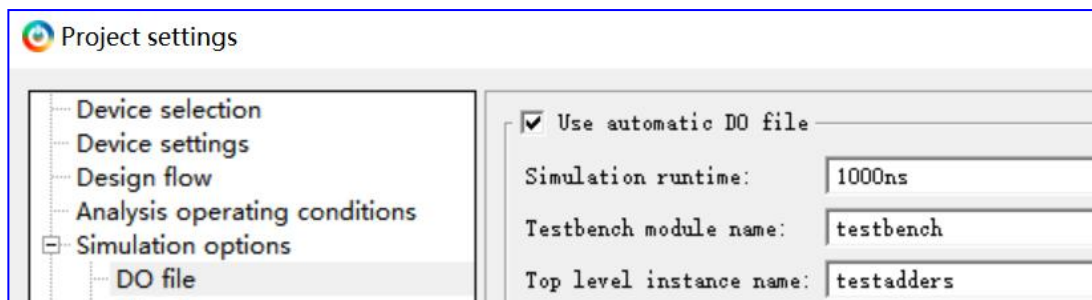
在打开的文件中去掉模板代码，输入以下测试平台代码：

```
`timescale 1ns/1ns
module testbench;
    reg pa,pb,pCin;
    wire Sum1,Cout1,Sum2,Cout2;
    adders testadders(.DataA(pa),.DataB(pb),.Cin(pCin),
        .Sum1(Sum1),.Cout1(Cout1),.Sum2(Sum2),.Cout2(Cout2));
        // 3 个输入，4 个输出

    initial
        begin
            pa=0;pb=0;pCin=0;
            #5 pCin=1;
            #5 pb=1;
            #5 pCin=0;
            #5 pa=1;
            #5 pCin=1;
            #5 pb=0;
            #5 pCin=0;
            #5,$finish;
        end
endmodule
```

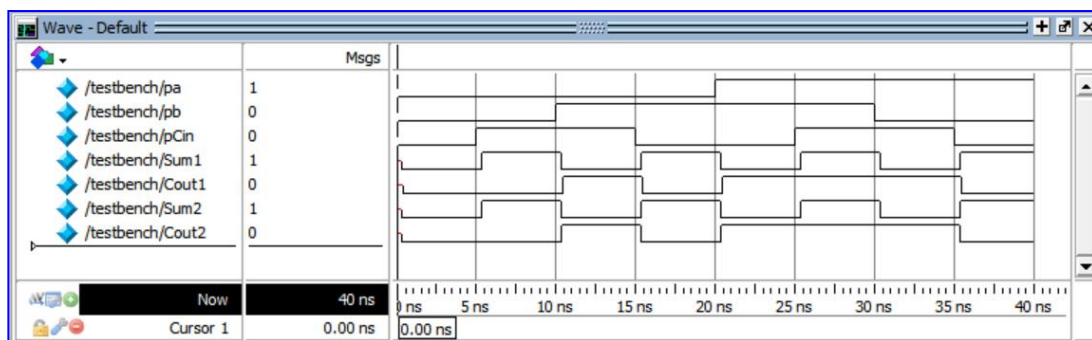
6. 选项配置

在“Project”菜单选择“Project Settings”，在弹出的对话框中选择“Simulation Options”，配置好测试平台模块名称及顶层实例名。如图所示：



7. 功能仿真

在“Design Flow”中，运行“Verify Pre-Synthesized Design”下的“Simulate”功能，进行功能仿真（综合前仿真），仿真结果如图所示：

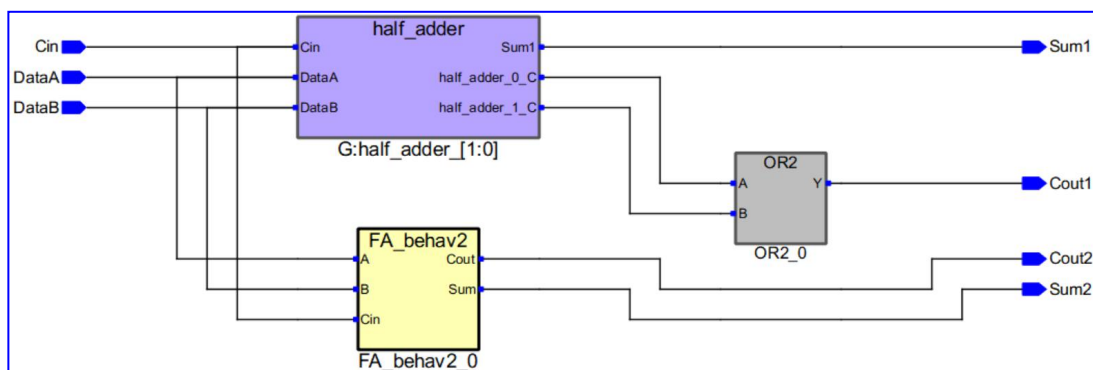


说明：

- (1) 由于测试平台中有“\$finish”操作，故仿真结束时提示是否结束“Are you sure you want to finish?”，在此不要选择“是”，否则 ModelSim 软件将退出。
- (2) 在波形结果中，两个全加器采用相同的输入（pa，pb，pCin），分别产生不同的输出（“Sum1”，“Cout1”为“全加器 1”的输出，“Sum2”，“Cout2”为“FA_behav2”的输出）
- (3) 放大波形，会看到各输出端口有延迟产生（一开始为红色的 x 态，后续的每次输出都略有延迟），这是因为调用了 IP 核（或门），因此仿真时也考虑了这部分的延迟。

8. 综合结果

对设计进行综合，RTL 视图如图所示：

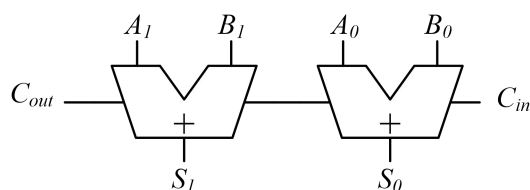


接下来操作包括综合后仿真、布局布线等，实现步骤与 5.5.2 的综合实例 2 相同，不作赘述。

5.5.3.3 造为 2 位串行进位加法器

1. 设计结构图

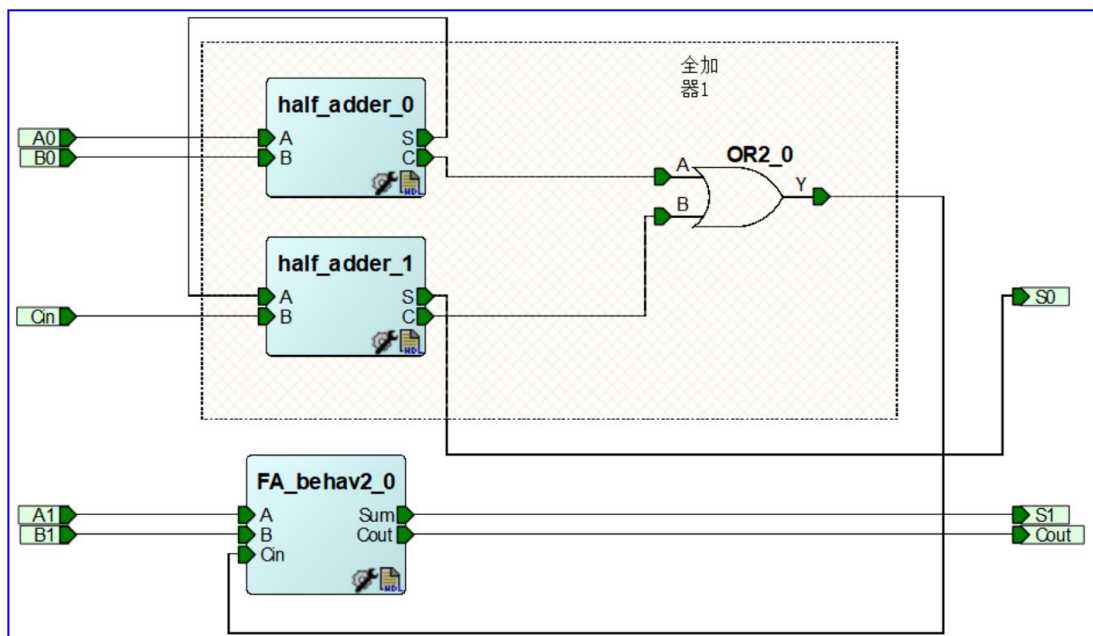
在第二章（2.3.5）中，讨论了如何通过 1 位全加器构造多位串行进位加法器，图 2-34 是一个 4 位的串行进位加法器，稍作删改，可得到如图所示的 2 位串行进位加法器。在此可将现有的 2 个全加器拼装为 2 位的串行进位加法器，。



2. 连线并修改端口名称

按照上图对“adders”设计进行改造，更改端口连线及端口名称为如下图所示的结构。

提示：可以用“disconnect”操作取消连线，用“Delete Top Level Port”去掉多余的顶层端口。

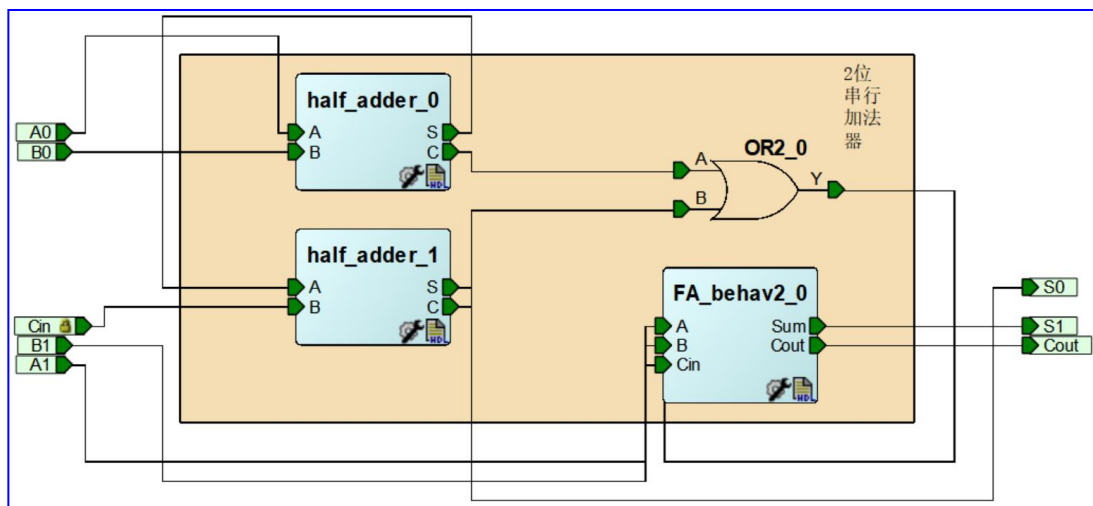


3. 保存并生成设计

选择“File”菜单的“Save adders As”命令，将设计另存为“adder_2”。

选择“SmartDesign”菜单的“Generate Component”命令，或者对着画布区按右键，选择“Generate Component”命令，生成设计。

调整布局，并修改文字提示，整体构成一个2位串行加法器，如图所示：



4. 编写测试平台

新建测试平台，保存文件为“testbench2.v”。代码如下：

```
`timescale 1ns/1ns
```

```

module testbench2;
    reg a0,b0,a1,b1,cin;
    integer i;
    wire s0,s1,cout;

    adder_2 testadder_2(.A0(a0),.B0(b0),.A1(a1),.B1(b1),.Cin(cin),
        .S0(s0),.S1(s1),.Cout(cout));

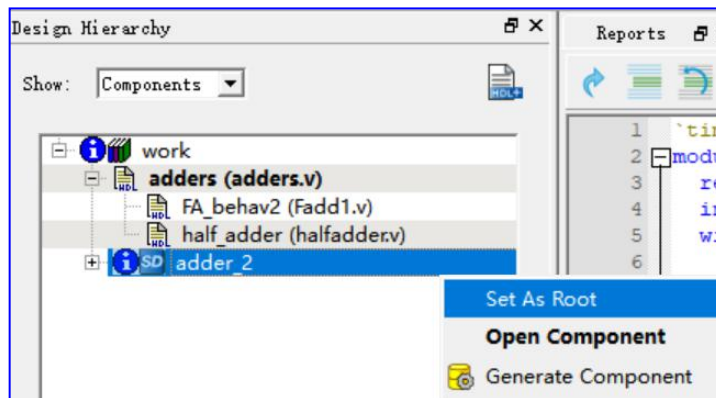
    initial
        begin
            for(i=0;i<32;i=i+1)
                begin
                    {a0,a1,b0,b1,cin}=i;
                    #5;
                end
            #5;$finish;
        end
    endmodule


```

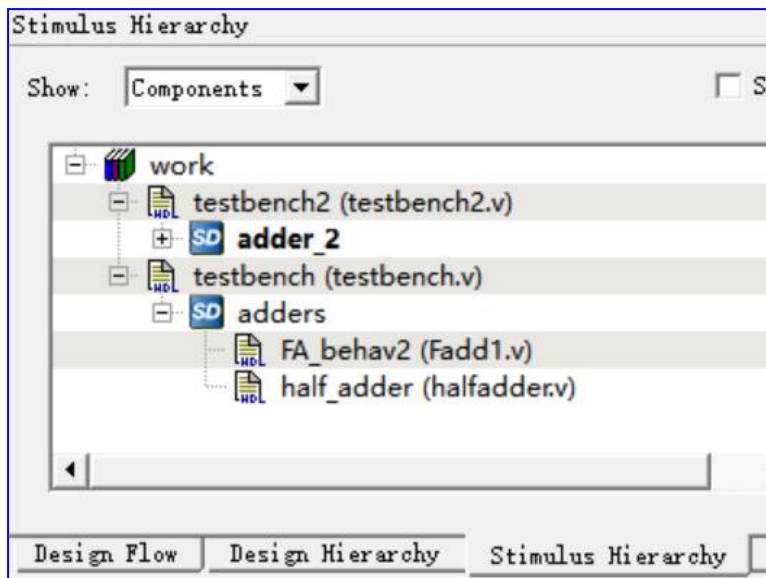
5. 选项配置

由于新建了另一个设计和测试平台，故须调整若干的选项设置：

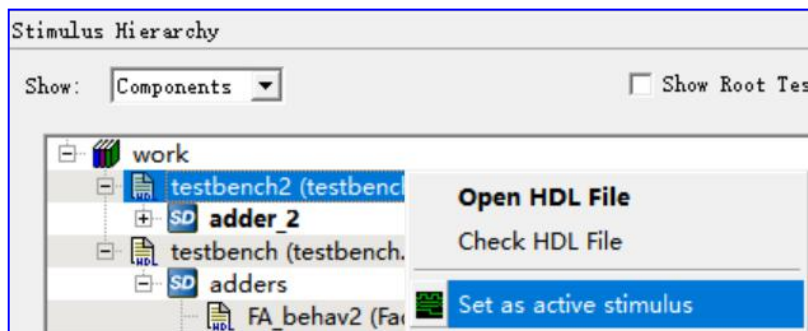
(1) 在“Design Hierarchy”中设置“adder_2”为根，设置方法跟前面是一致的，如图示：



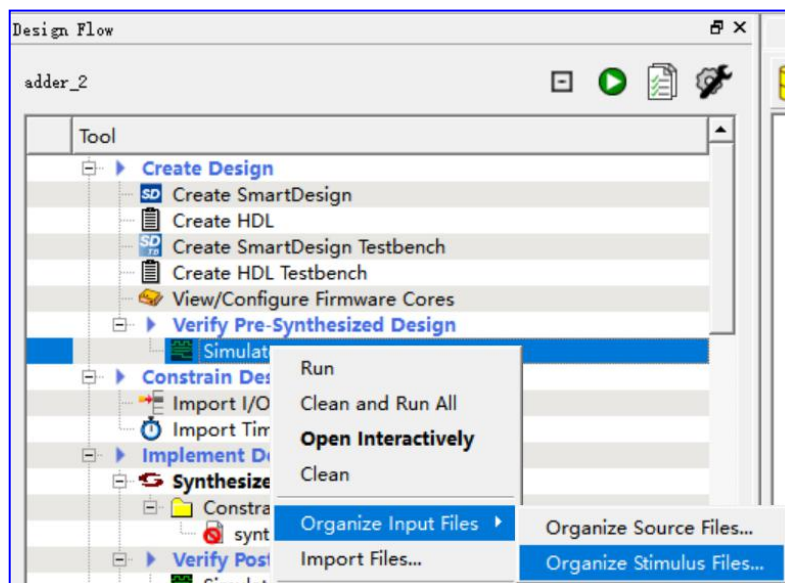
(2) 选择“SmartDesign”菜单的“Generate Component”命令，或在 adder_2 画布界面点击  按钮。在“Stimulus Hierarchy”窗口看到新增的 testbench2 文件、“adder_2”（已设为根）及其关联关系。



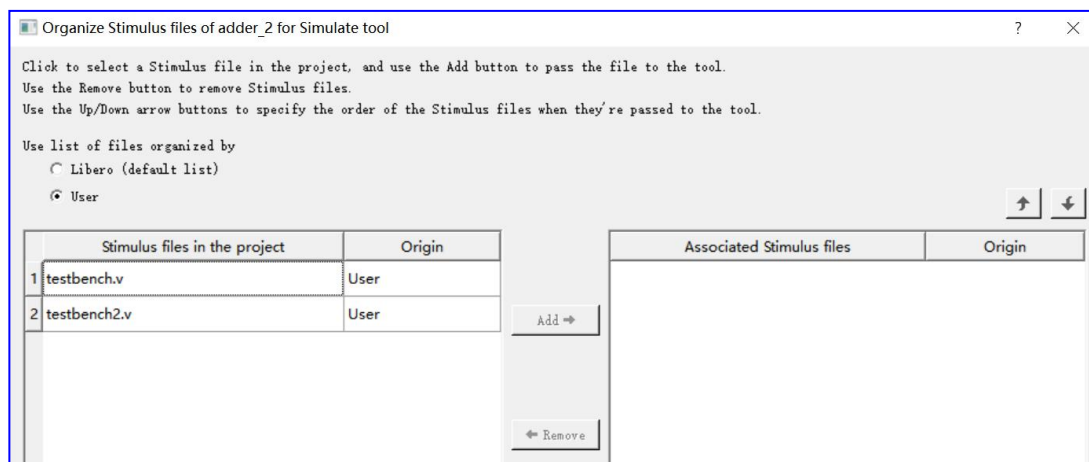
(3) 配置测试平台文件，对着“testbench2”模块按右键，选择“Set as active stimulus”（如图所示），把“testbench2.v”文件设置为关联测试平台（如图所示）。



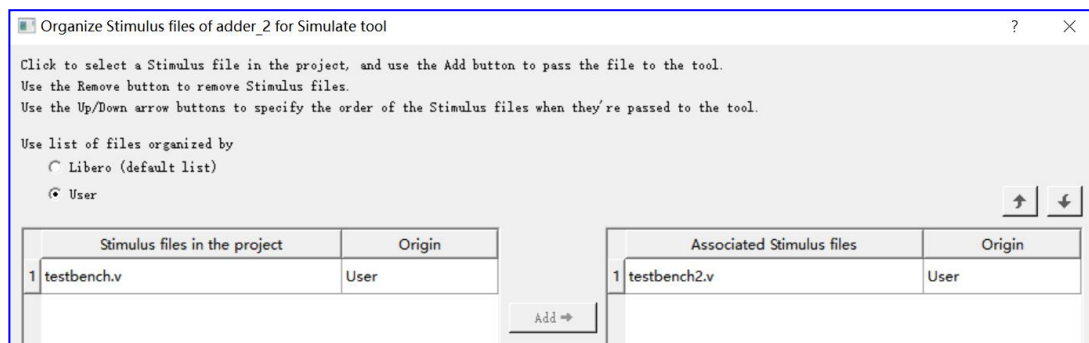
(4) 如果没有做步骤(3)，也可以在“Design Flow”中，对着“Verify Pre-Synthesized Design”下的“Simulate”功能，右键选择“Organize Input Files”下的“Organize Stimulus Files”。



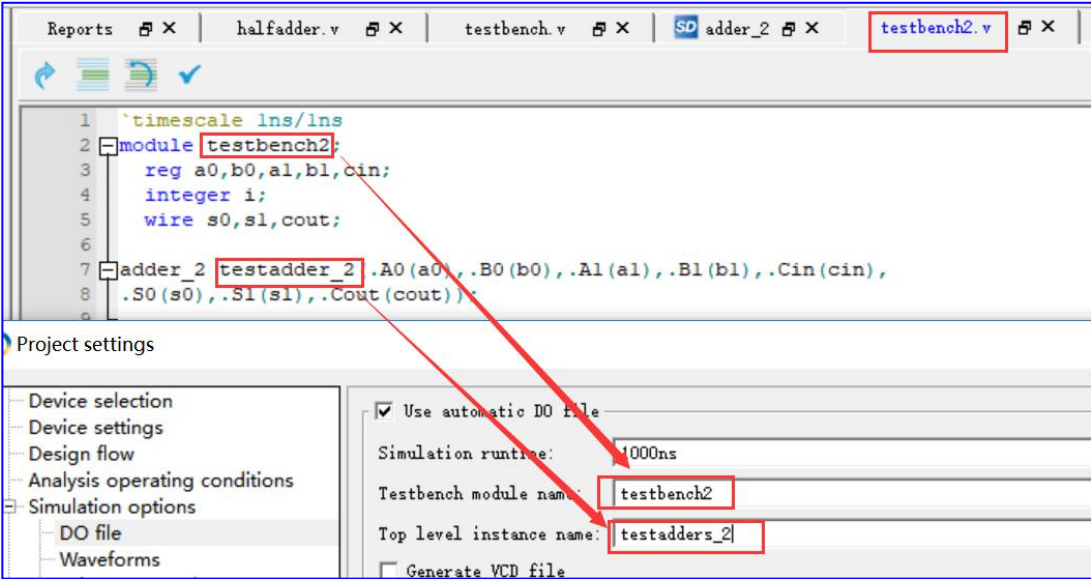
该设计目前有两个测试平台文件，不指定清楚的情况下，Libero 是不知道该从哪个文件开始测试的。



选择“testbench2”文件，文件来源为用户（User）录入（而非系统生成），点击“Add”按钮，则测试平台文件关联完成。

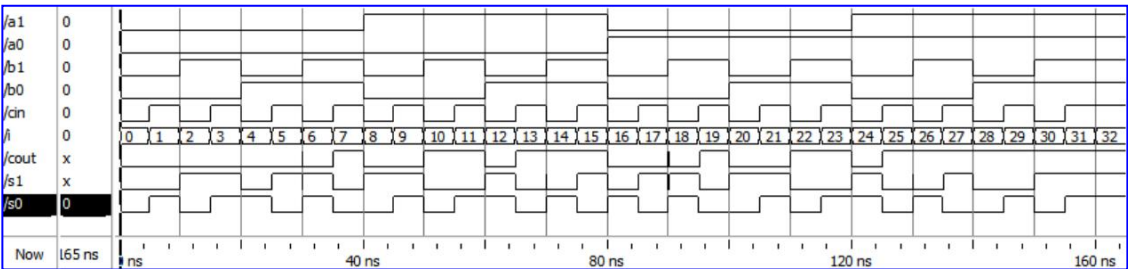


(5) 在“Project”菜单选择“Project Settings”，在弹出的对话框中选择“Simulation Options”，配置好测试平台模块名称及顶层实例名。如图所示：



6. 功能仿真

功能仿真结果如图所示：

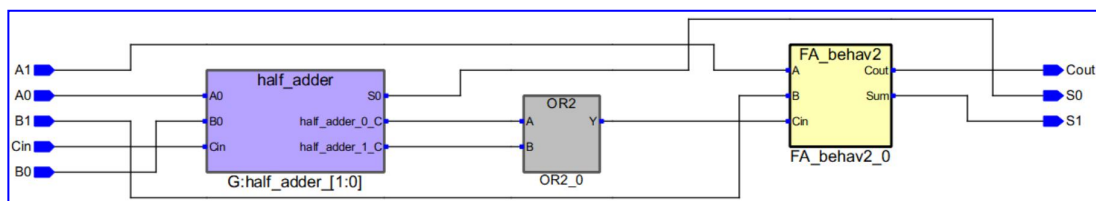


波形说明：

- (1) a1, a0 表示两位输入数据，如“a1=1, a0=0”，则相当于输入两位“A[1:0]=2'b10”；
- (2) 同理，b1, b0 可理解为“B[1:0]”，s1, s0 可理解为“S[1:0]”，计算的内容相当于“{cout,S[1:0]}=A[1:0]+B[1:0]+cin”；
- (3) 调整好变量的顺序，更加方便查看。
- (4) 由于是串行进位，并使用了 IP 核，因此仿真结果中考虑了延迟因素，出现了毛刺。

7. 综合结果

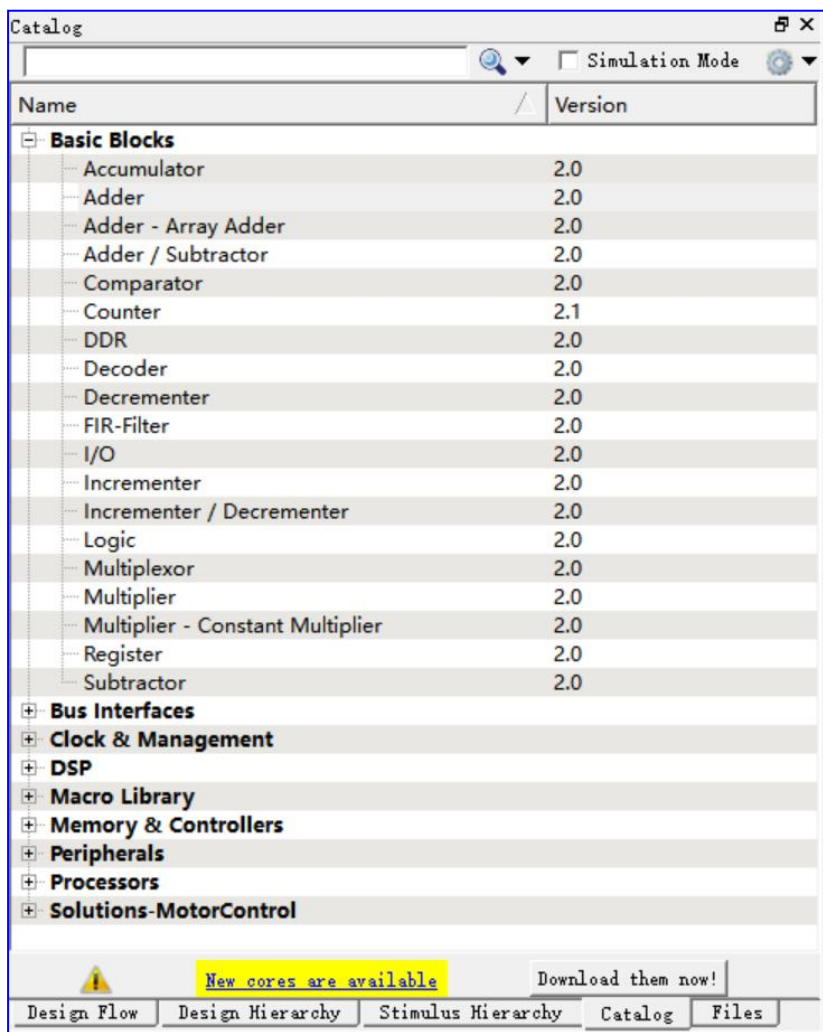
综合结果如图所示：



5.5.3.4 调用 IP 核创建 2 位串行进位加法器

前面是通过拼装的方式实现 2 位串行进位加法器，只是为了说明 SmartDesign 的使用方法，并不是最佳的实现方法。

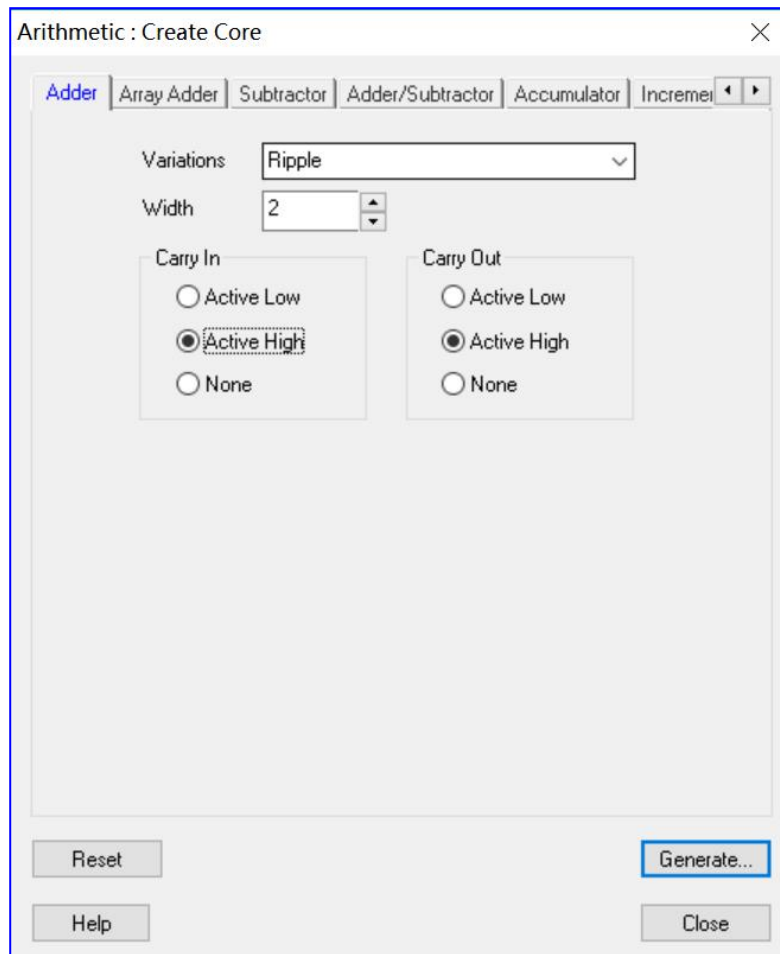
在 Libero SoC 中，可以通过现成的模块非常方便地创建多种功能模块：在 Catalog 窗口中列出了 Microsemi 公司提供的多种 IP 核，从简单的宏单元（基本门等）、中规模的基本块（加法器、编码器等）、总线接口，到大规模的 DSP、处理器等都可以提供使用（如图所示）。



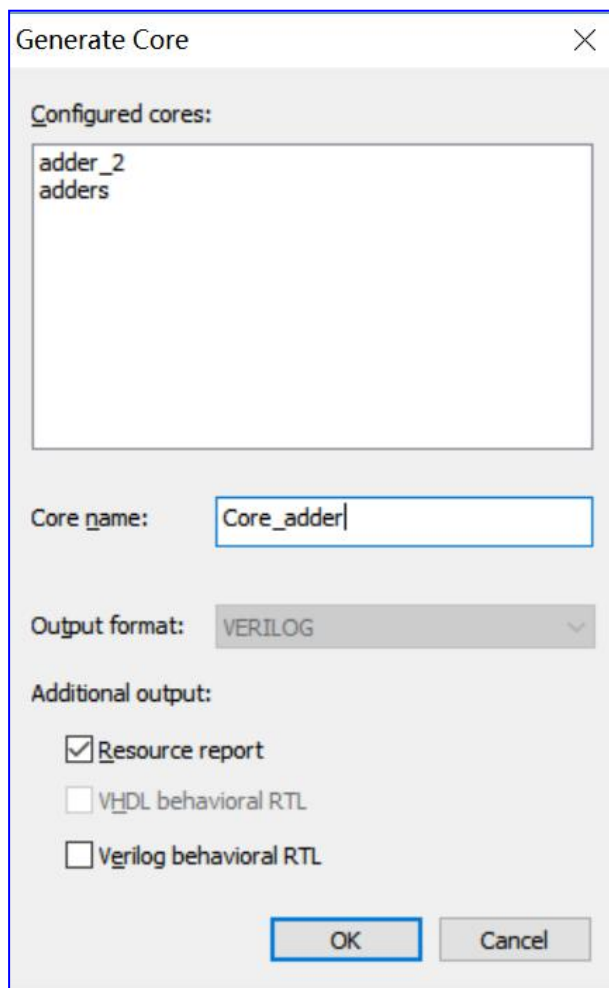
以下操作是调用 IP 核方式创建 2 位串行进位加法器，并与之前的设计进行对比。

1. 创建新核并实例化

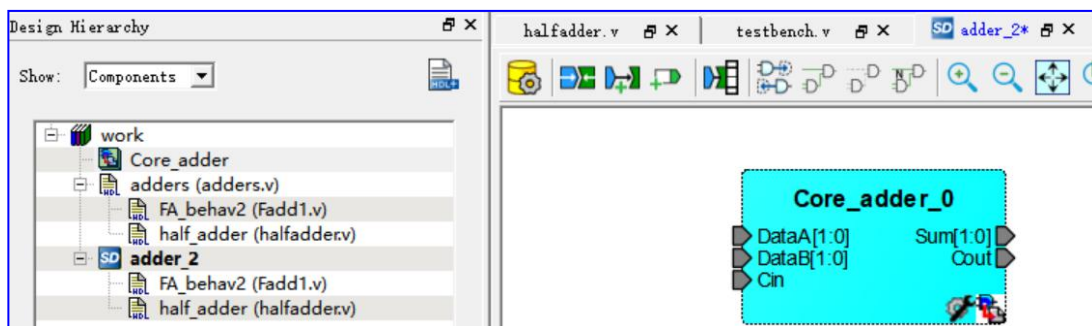
选择“Basic Blocks”下的“Adder”，按右键选择“Configure core”，或直接双击，出现如图所示的对话框。



“Ripple”是代表串行进位，“Width”中最小是 2 位，代表所创建加法器的位数，“Carry In”和“Carry out”中选择“Active High”表示进位输入/出高电平为 1。点击“Generate”按钮，出现如图的对话框，输入新建核的名称。



在项目层次结构图中，可见到 Core_adder，将其添加到画布中（实例化名为“Core_adder_0”），如图所示：



模块建立成功，需要更改加法器的位宽和其它参数，只需对着“Core_adder_0”核按右键，选择“Configure”命令（或直接双击）即可，非常方便。

2. 局部连线

对着任一模块右键，选择“QuickConnect”，可查看及编辑该模块的端口连接，如图

所示为 half_adder_1 模块的引脚及其所连接到的模块端口。

QuickConnect

Instance Pins

half_adder_1

Pin	Connection
A	half_adder_0:S
B	adder_2:Cin
C	OR2_0:B
S	adder_2:S0

对着空白处右键，选择“QuickConnect”，则查看所有未连接的端口，如图所示：

QuickConnect

Instance Pins

All unconnected pins in design

Search: *.*

Pin

Connection

Core_adder_0:Cin

Core_adder_0:Cout

Core_adder_0:DataA[1:0]

Core_adder_0:DataB[1:0]

Core_adder_0:Sum[1:0]

Pins to Connect

Enter a wildcard search in the form of [Instance].[Pin]

adder_2

A0

A1

B0

B1

Cin

Cout

S0

S1

Core_adder_0

Cin

Cout

DataA[1:0]

DataB[1:0]

Sum[1:0]

FA_behav2_0

half_adder_0

half_adder_1

OR2_0

Pin: Cout

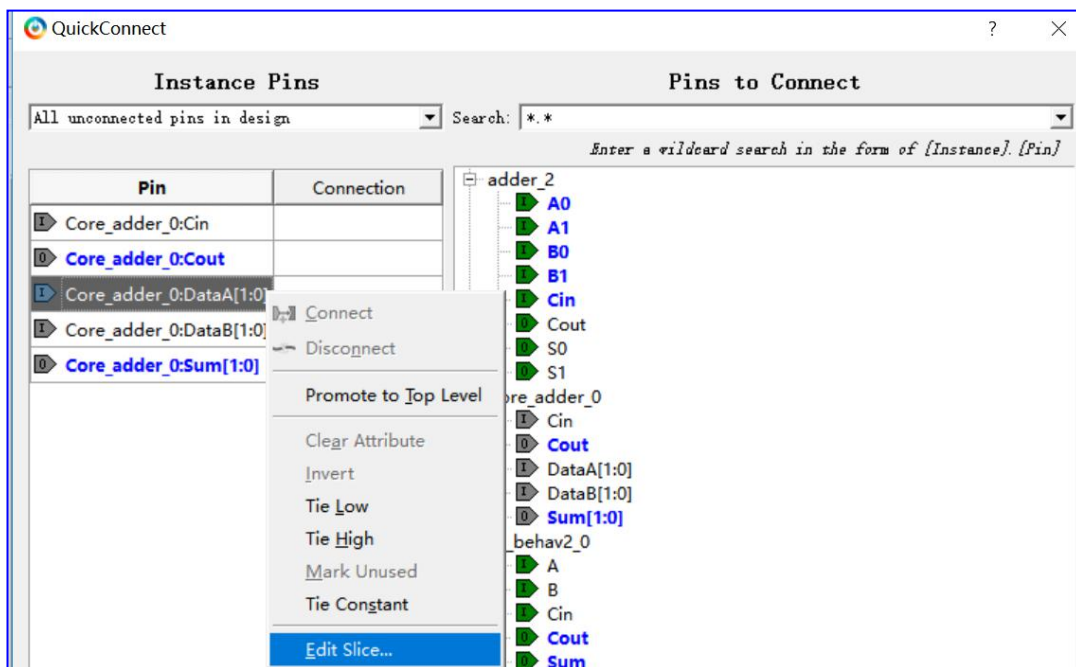
Instance: Core_adder_0

Direction: OUT

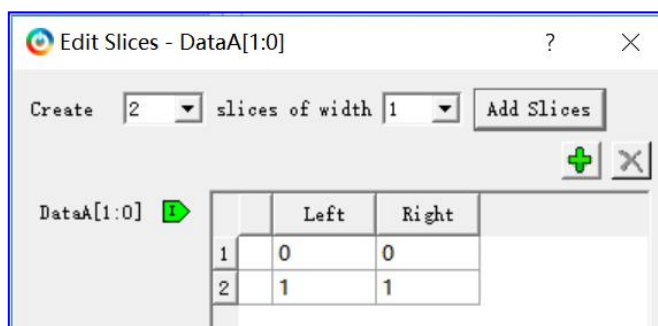
State: Unconnected

目前，新添加的 Core_adder_0 未与其它端口连线。由于该核的端口采用多位向量形式，不能直接与原来的端口进行连接。是否需要改写测试平台才能进行仿真测试？可以通过如下方法，将多位向量形式的端口按每一位单独进行连线操作：

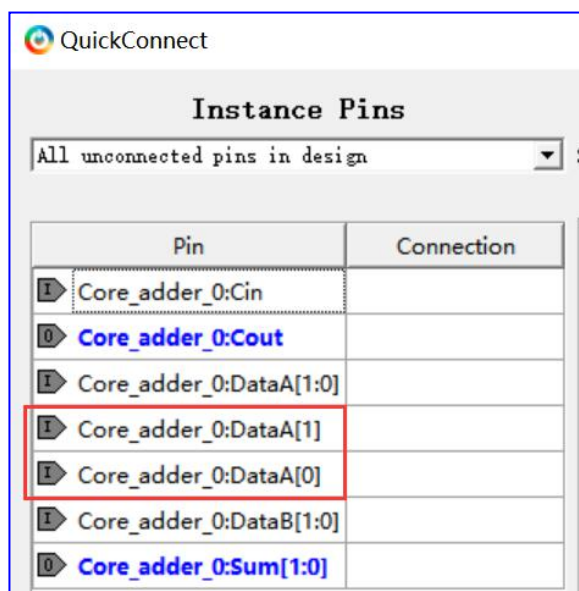
（1）因为需要把新增的“Core_adder_0”的“DataA”端口分成两个单独的位端口以便连接，故对着“DataA”右键，选择“Edit Slice”，如图所示。



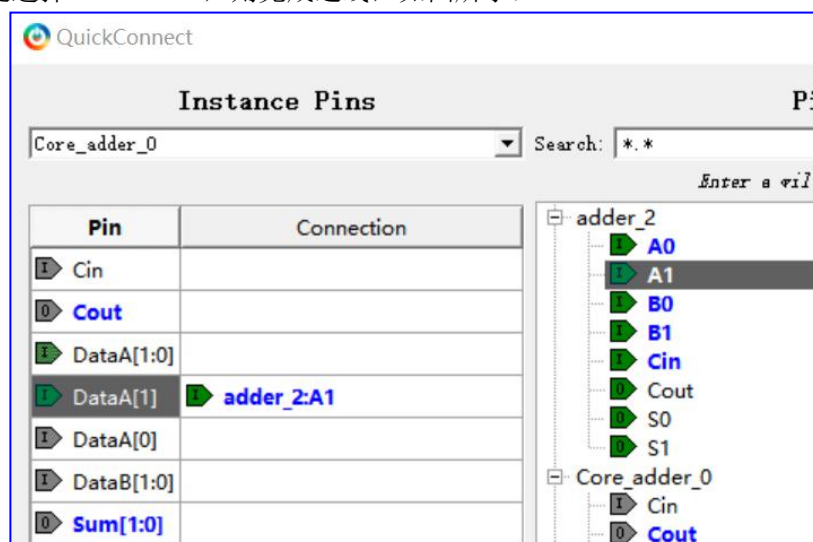
(2) 在弹出的对话框中，点击“Add Slices”按钮，则“DataA”端口分成了独立的两个端口。



(3) 确定后，会看到出现两个对应的端口。端口前面的“I”标志表示输入端口，“O”标志表示输出端口。

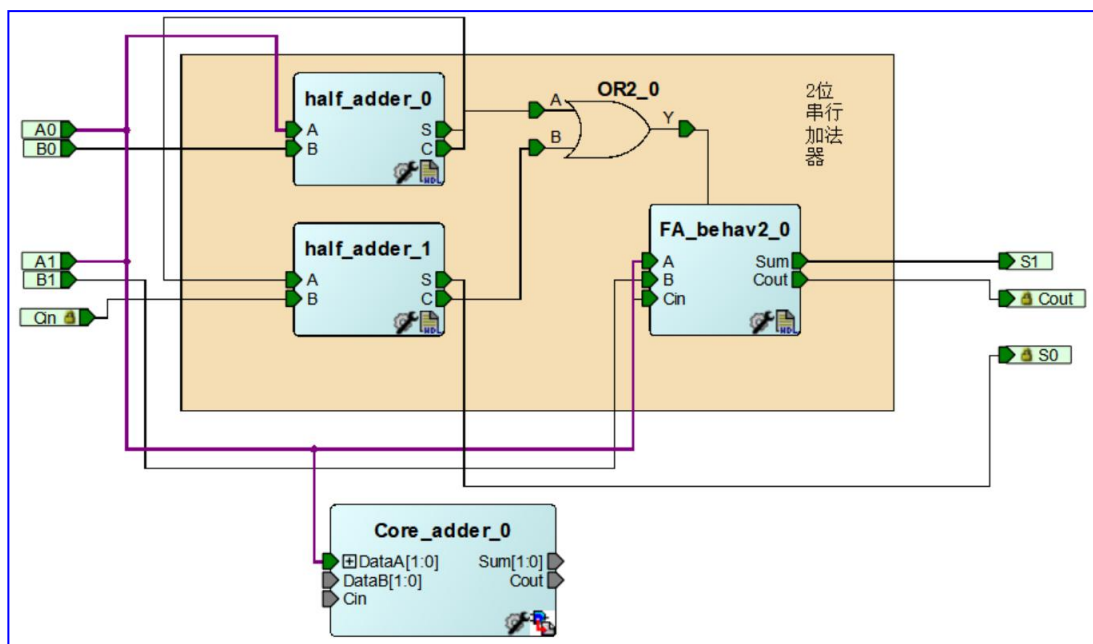


(4) 在“Instance Pins”的下拉列表中，选择“Core_adder_0”，这样可以看到该核的所有已连接及未连接端口。选择左边的“DataA[1]”端口，选择右边“adder_2”下的“A1”端口，右键选择“Connect”，则完成连线，如图所示：

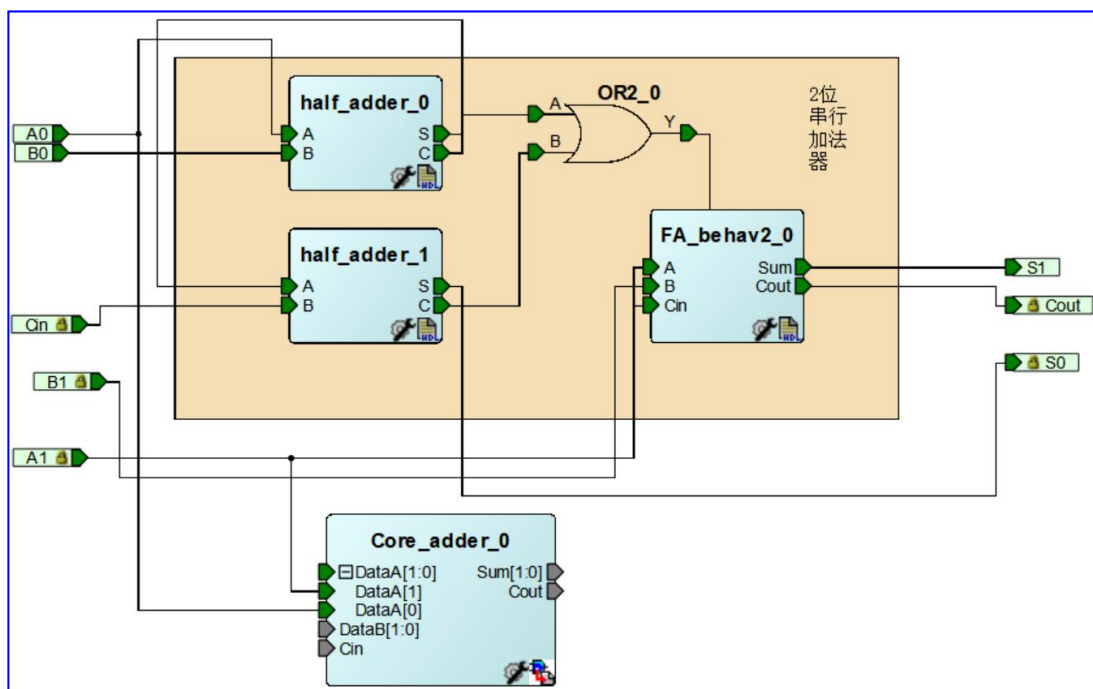


3. 全局连线

用同样的操作方法，把“Core_adder_0”的“DataA”端口的[0]位连接到“adder_2”的“A0”端口。可看到如图所示的连接效果。



从图中可看到，A1 和 A0 同时连接到不同的实例上了，但显示效果不好，也看不出 DataA 每位的具体连接情况。可点击 DataA 前面的加号，展开 DataA 端口的各位，显示效果更为清晰（如图所示）。

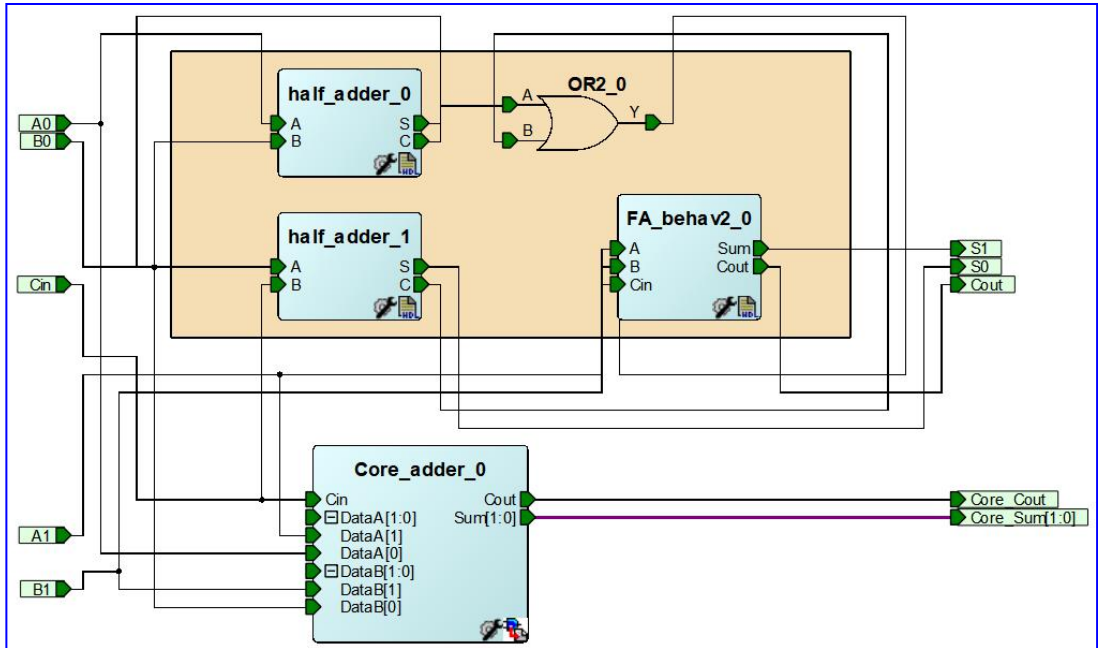


与前述操作类似，也可直接在画布上操作，对“Core_adder_0”进行以下连线操作：

- 将“DataB”端口的[0]位连接到画布顶层“B0”端口；
- 将“DataB”端口的[1]位连接到画布顶层“B1”端口；

- 将“Cin”端口连接到画布顶层“Cin”端口；
- 将“Cout”端口连接到顶层，端口命名为“Core_Cout”；
- 将“Sum[1:0]”端口连接到顶层，端口命名为“Core_Sum”，系统自动显示为“Core_Sum[1:0]”

连接结果如图所示：



4. 保存并生成设计

选择“File”菜单的“Save adders As”命令，将设计另存为“adder_3”。

选择“SmartDesign”菜单的“Generate Component”命令，或者对着画布区按右键，选择“Generate Component”命令，生成设计。

5. 编写测试平台

在前面例子的基础上编写测试平台，输入端口没有变化，输出端口多了两个，故只需增加输出端口的连接即可。新建测试平台文件“testbench3.v”。

```
`timescale 1ns/1ns
module testbench3;
    reg a0,b0,a1,b1,cin;
    integer i;
    wire s0,s1,cout,core_cout;
    wire[1:0] core_sum;

    adder_3 testadder_3(.A0(a0),.B0(b0),.A1(a1),.B1(b1),.Cin(cin),
        .S0(s0),.S1(s1),.Cout(cout),.Core_Cout(core_cout),.Core_Sum
```

```

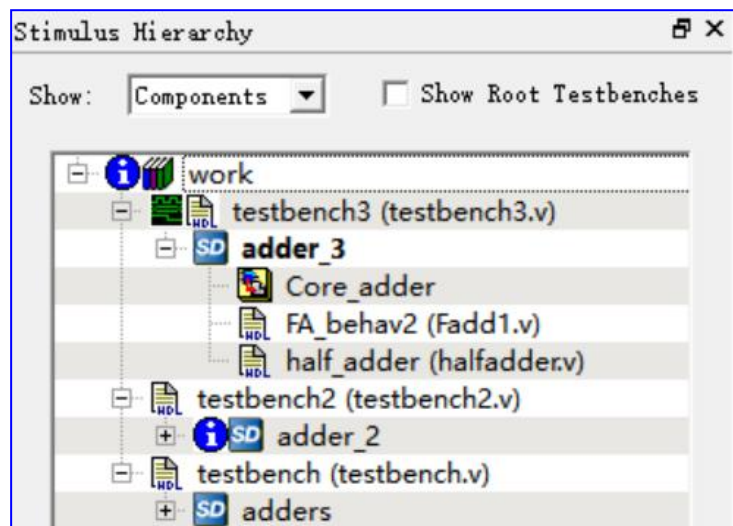
(core_sum));

    initial
    begin
        for(i=0;i<32;i=i+1)
        begin
            {a0,a1,b0,b1,cin}=i;
            #5;
        end
        #5;$finish;
    end
endmodule

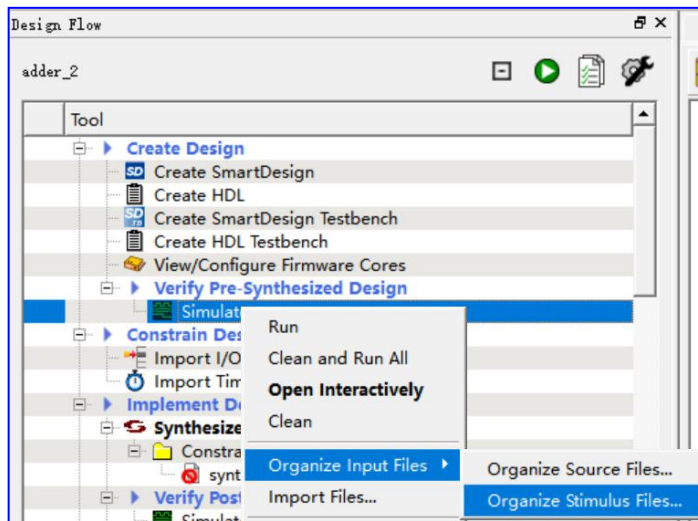
```

6. 选项配置

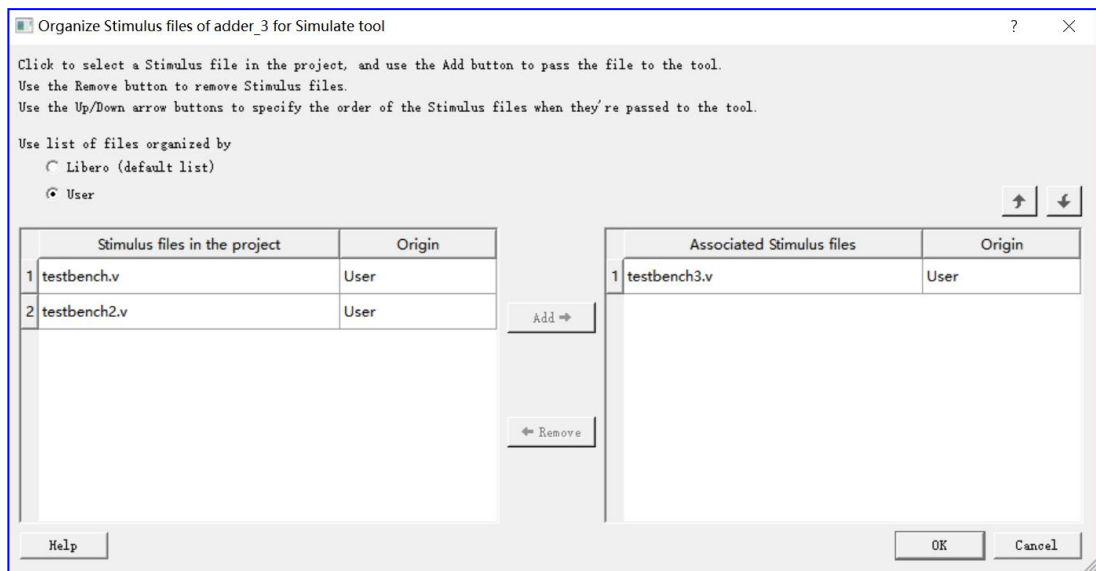
可在“Stimulus Hierarchy”中看到多个画布设计及其测试平台，testbench3 模块前面的波形图图标，表示其为激活状态（仿真启动文件）。如果勾选“Show Root Testbenches”，则只显示 testbench3 及其调用的设计 adder_3。



在“Design Flow”中，对着“Verify Pre-Synthesized Design”下的“Simulate”功能，右键选择“Organize Input Files”下的“Organize Stimulus Files”。

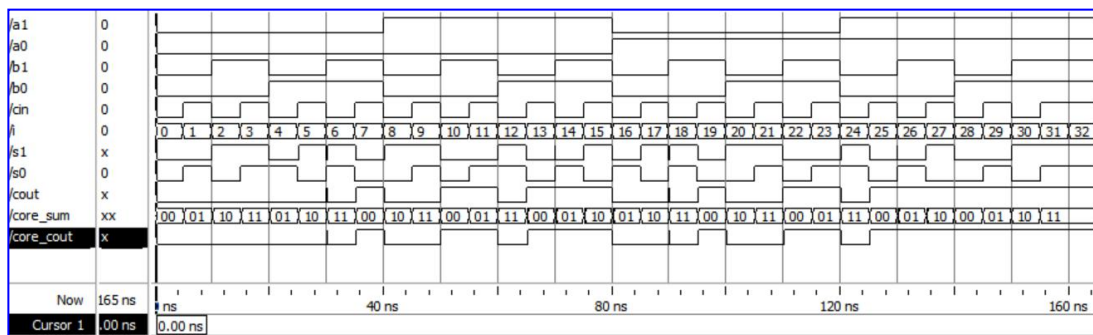


在打开的对话框中，有三个测试平台，testbench3.v 为关联的文件，如果不是的话，可以修改设置，如图所示。



7. 功能仿真

适当调整信号位置，及显示的进制，仿真波形如图所示：



8. 更好的显示效果

从上图中，虽然可以在同一波形图中对比两个设计的结果，但呈现的效果并不直观。可以对测试平台和显示变量进行适当的调整，以达到更好的显示效果。修改测试平台代码如下：

```

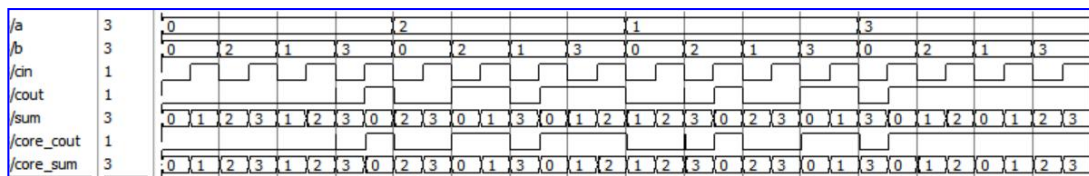
`timescale 1ns/1ns
module testbench3;
    reg a0,b0,a1,b1,cin;
    reg[1:0] a,b;
    integer i;
    wire s0,s1,cout,core_cout;
    wire[1:0] core_sum,sum;
    assign sum={s1,s0};

    adder_3 testadder_3(.A0(a0),.B0(b0),.A1(a1),.B1(b1),.Cin(cin),
        .S0(s0),.S1(s1),.Cout(cout),.Core_Cout(core_cout),.Core_Sum
(core_sum));

    initial
        begin
            for(i=0;i<32;i=i+1)
                begin
                    {a0,a1,b0,b1,cin}=i;
                    a={a1,a0};
                    b={b1,b0};
                    #5;
                end
            #5;$finish;
        end
endmodule

```

在仿真波形结果中，把 i、a1、a0、b1、b0、s1、s0 等变量的显示去掉（对着相应变量右键，选择“delete”），再调整一下变量的显示顺序（鼠标拖拽即可），就可得到如下图的显示结果，该波形显示更直观易懂。



9. 综合结果

