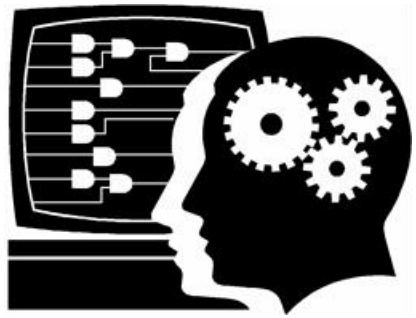
数字系统设计与Verilog HDL

（第6版）



第4章 Verilog设计初步

4.1 Verilog简介

4.2 Verilog模块的结构

4.3 Verilog基本组合电路设计

4.4 Verilog基本时序电路设计

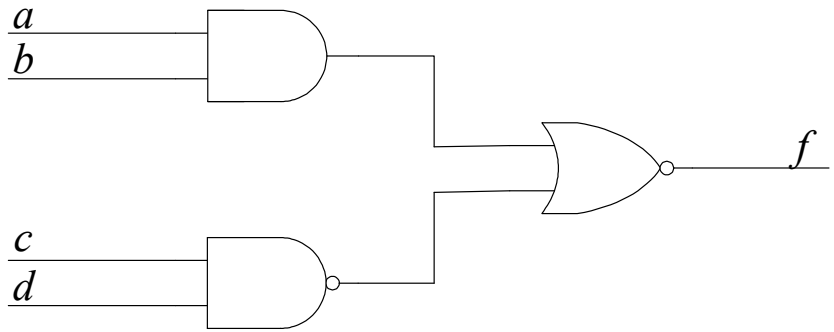
4.1 Verilog简介

- Verilog语言是1983年由GDA（Gateway Design Automation）公司的Phil Moorby首创的，之后Moorby又设计了Verilog-XL仿真器，Verilog-XL仿真器大获成功，也使得Verilog语言得到推广使用。
- 1989年，Cadence收购了GDA，1990年，Cadence公开发表了Verilog HDL，并成立了OVI组织专门负责Verilog HDL的发展。
- Verilog于1995年成为IEEE标准，称为IEEE Standard 1364-1995（Verilog-1995）
- IEEE“1364-2001”标准（Verilog-2001）也获得通过，多数综合器、仿真器都已支持Verilog-2001标准

Verilog语言的特点

- 既适于可综合的电路设计，也可胜任电路与系统的仿真。
- 能在多个层次上对所设计的系统加以描述，从开关级、门级、寄存器传输级（RTL）到行为级，都可以胜任，同时语言不对设计规模施加任何限制。
- 灵活多样的电路描述风格，可进行行为描述，也可进行结构描述；支持混合建模，在一个设计中各个模块可以在不同的设计层次上建模和描述。
- Verilog的行为描述语句，如条件语句、赋值语句和循环语句等，类似于软件高级语言，便于学习和使用。
- 内置各种基本逻辑门，便于进行门级结构描述；内置各种开关级元件，可进行开关级的建模。
- 易学易用，功能强，可满足各个层次设计人员的需要。

4.2 Verilog模块的结构



“与-或-非”电路

```
module aoi(a,b,c,d,f);
/* 模块名为aoi，端口列表a，b，c，d，f */
input a,b,c,d; //模块的输入端口为a，b，c，d
output f;    //模块的输出端口为f
wire a,b,c,d,f;          //定义信号的数据类型
assign f=~((a&b)|(~(c&d)));          //逻辑功
  能描述
endmodule
```

4.2 Verilog模块的结构

- Verilog程序是由模块构成的。每个模块的内容都嵌在module和endmodule两个关键字之间；每个模块实现特定的功能。
- 每个模块首先要进行端口定义，并说明输入和输出口（input、output或inout），然后对模块的功能进行定义。
- Verilog程序书写格式自由，一行可以写几个语句，一个语句也可以分多行写。
- 除了endmodule等少数语句外，每个语句的最后必须有分号。
- 可用 /*……*/ 和 //……对Verilog程序作注释。

1．模块声明

模块声明包括模块名字，模块输入、输出端口列表。
模块定义格式如下：

module 模块名(端口1，端口2，端口3，……);

2．端口（Port）定义

对模块的输入输出端口要明确说明，其格式为：

input 端口名1，端口名2，…… 端口名n;

//输入端口

output 端口名1，端口名2，…… 端口名n;

//输出端口

inout 端口名1，端口名2，…… 端口名n;

//输入输出端口

3．信号类型声明

- 对模块中所用到的所有信号（包括端口信号、节点信号等）都必须进行数据类型的定义。Verilog语言提供了各种信号类型，分别模拟实际电路中的各种物理连接和物理实体。
- 如果信号的数据类型没有定义，则综合器将其默认为是wire型。

4．逻辑功能定义

模块中最核心的部分是逻辑功能定义。

定义逻辑功能的几种基本方法:

（1）用assign持续赋值语句定义

assign语句多用于组合逻辑的赋值，称为持续赋值方式。

（2）用always过程块定义

always过程语句既可以用来描述组合电路，也可以描述时序电路。

（3）调用元件（元件例化）

调用元件的方法类似于在电路图输入方式下调入图形符号来完成设计，这种方法侧重于电路的结构描述。

Verilog 模块的模板

```
module   <顶层模块名> (<输入输出端口列表>);
output   输出端口列表;              //输出端口声明
input    输入端口列表;               //输入端口声明
/*定义数据，信号的类型，函数声明*/
reg   信号名;
//逻辑功能定义
assign <结果信号名>=<表达式>;       //使用assign语句定义逻辑功能
//用always块描述逻辑功能
always @ (<敏感信号表达式>)
  begin
    //过程赋值
    //if-else，case语句
    //while，repeat，for循环语句
    //task，function调用
  end
//调用其他模块
 <调用模块名module_name > <例化模块名> (<端口列表port_list >);
//门元件例化
 门元件关键字 <例化门元件名> (<端口列表port_list>);
endmodule
```

4.3 Verilog基本组合电路设计

【例4.5】　三人表决电路的Verilog描述

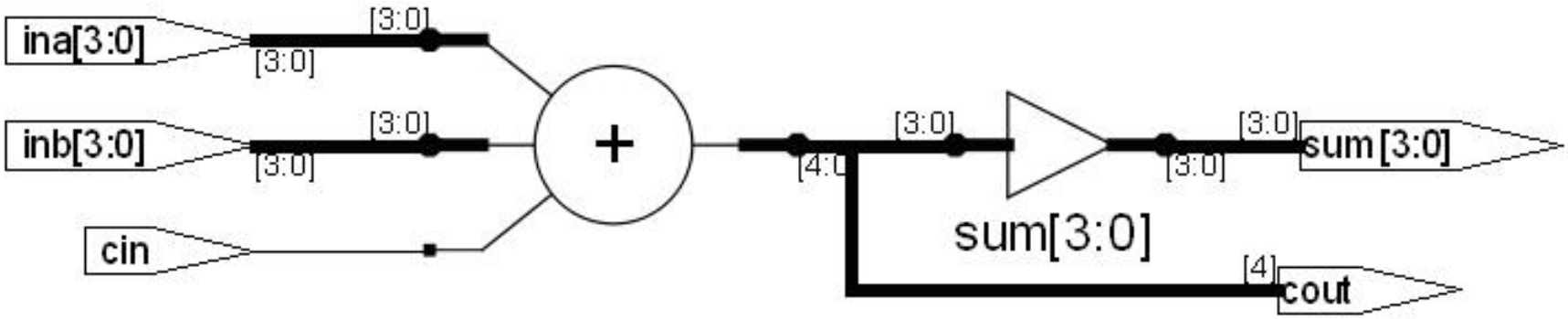
```
module vote(a,b,c,f);          //模块名与端口列表
input a,b,c;                   //模块的输入端口
output f;                      //模块的输出端口
wire a,b,c,f;                  //定义信号的数据类型
assign f=(a&b)|(a&c)|(b&c);
                               //逻辑功能描述
endmodule
```

4.3 Verilog基本组合电路设计

【例4.6】4位二进制加法器的Verilog描述

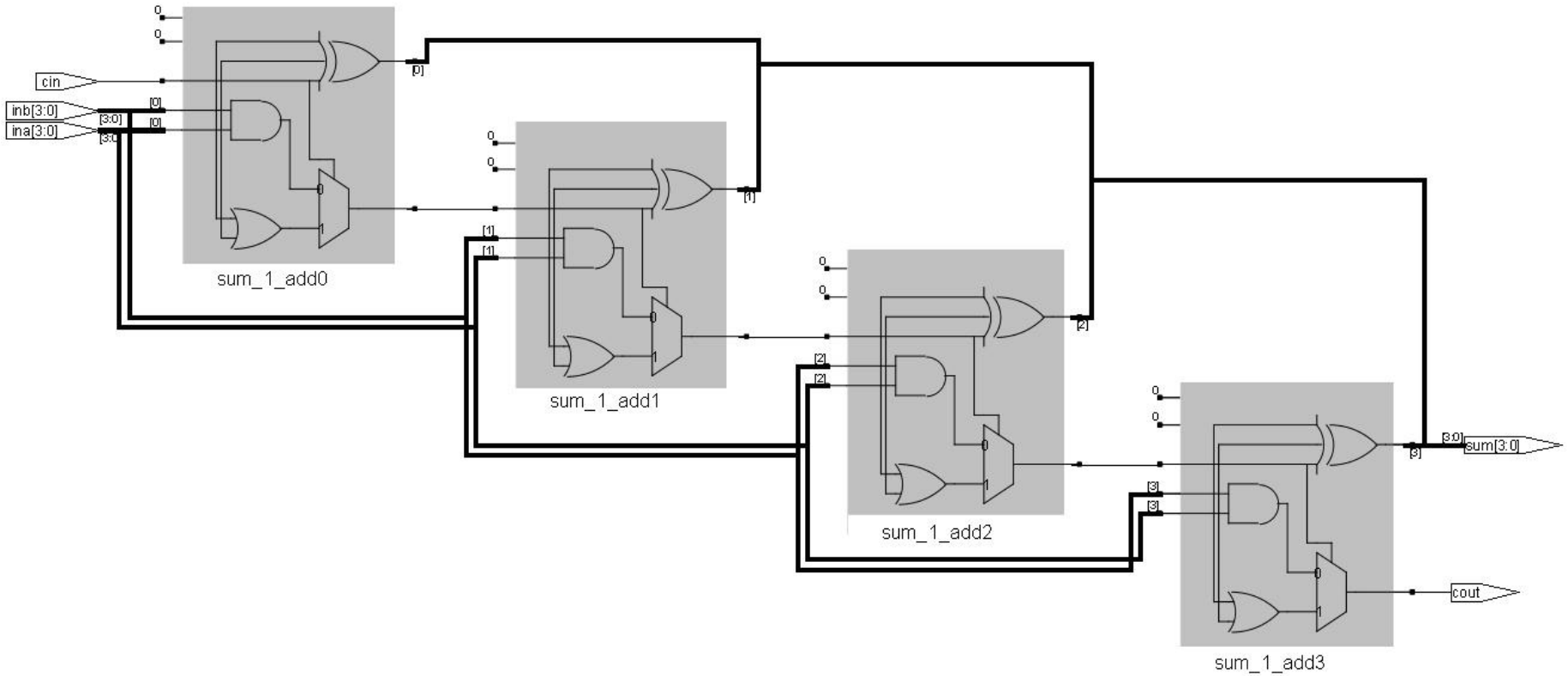
```
module add4_bin(cout,sum,ina,inb,cin);
input cin;
input[3:0] ina,inb;
output[3:0] sum;
output cout;
assign {cout,sum}=ina+inb+cin;
            /*逻辑功能定义*/
endmodule
```

综合 （RTL级）



4位二进制加法器RTL级综合结果

综合（门级）



4位二进制加法器门级综合视图

4.3 Verilog基本组合电路设计

【例4.8】 BCD码加法器

```
module add4_bcd(cout,sum,ina,inb,cin);
input cin; input[3:0] ina,inb;
output[3:0] sum; reg[3:0] sum;
output cout; reg cout;
reg[4:0] temp;
always @(ina,inb,cin)     //always过程语句
  begin  temp<=ina+inb+cin;
     if(temp>9) {cout,sum}<=temp+6;
           //两重选择的IF语句
     else {cout,sum}<=temp;
  end
endmodule
```

4.4 Verilog基本时序电路设计

【例4.9】　基本D触发器的Verilog描述

```
module dff(q,d,clk);
input d,clk;
output reg q;
always @(posedge clk)
begin
  q<=d;
end
endmodule
```

4.4 Verilog基本时序电路设计

带同步清0/同步置1（低电平有效）的D触发器

```
module dff_syn(q,qn,d,clk,set,reset);
input d,clk,set,reset; output reg q,qn;
always @(posedge clk)
  begin
  if(~reset) begin q<=1'b0;qn<=1'b1;end
  //同步清0，低电平有效
else if(~set) begin q<=1'b1;qn<=1'b0;end
  //同步置1，低电平有效
else  begin q<=d; qn<=~d; end
  end
endmodule
```

【例4.11】　带异步清0/异步置1（低电平有效）的D触发器

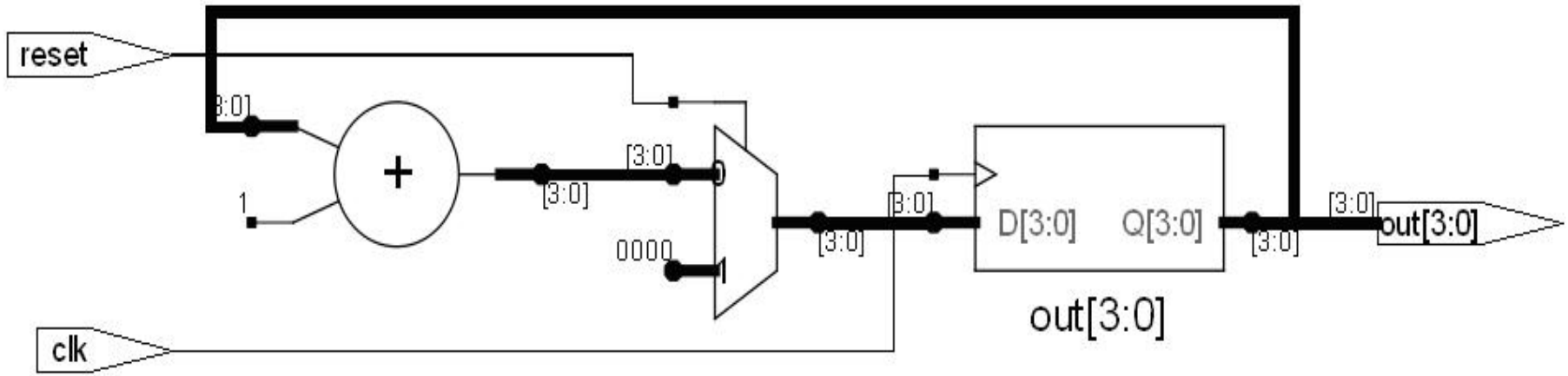
```
module dff_asyn(q,qn,d,clk,set,reset);
input d,clk,set,reset; output reg q,qn;
always @(posedge clk or negedge set or negedge
  reset)
  begin
if(~reset)       begin q<=1'b0;qn<=1'b1; end
  //异步清0，低电平有效
else if(~set) begin q<=1'b1;qn<=1'b0; end
  //异步置1，低电平有效
else        begin  q<=d;qn<=~d; end
  end
endmodule
```

4.4 Verilog基本时序电路设计

【例4.12】　4位二进制加法计数器

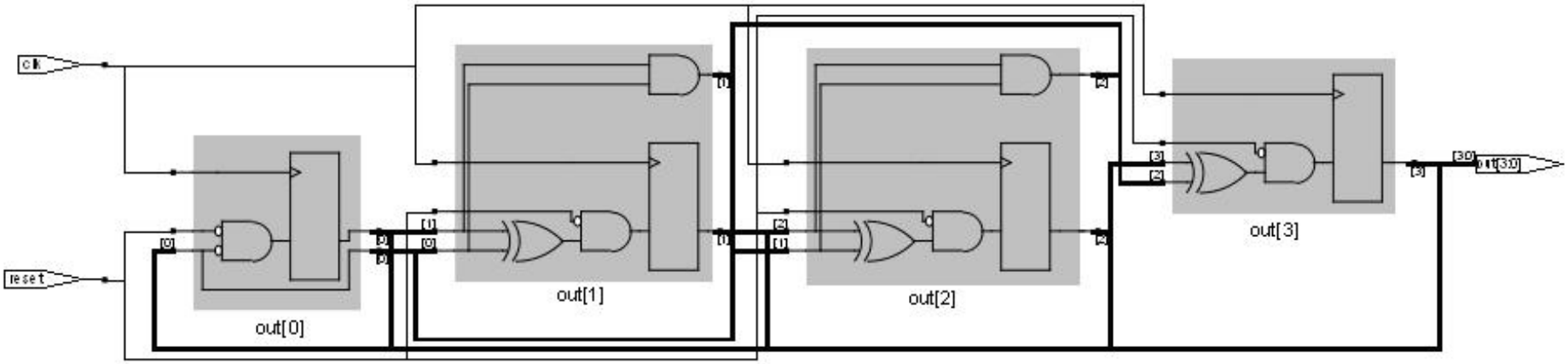
```
module count4(out,reset,clk);
input reset,clk;
output reg[3:0] out;
always @(posedge clk)
begin
if(reset)        out<=0;     //同步复位
else       out<=out+1;       //计数
end
endmodule
```

综合 （RTL级）



4位二进制加法计数器RTL级综合视图
（Synplify Pro）

综合（门级）



4位二进制加法计数器门级综合视图
（Synplify Pro）

习　题　4

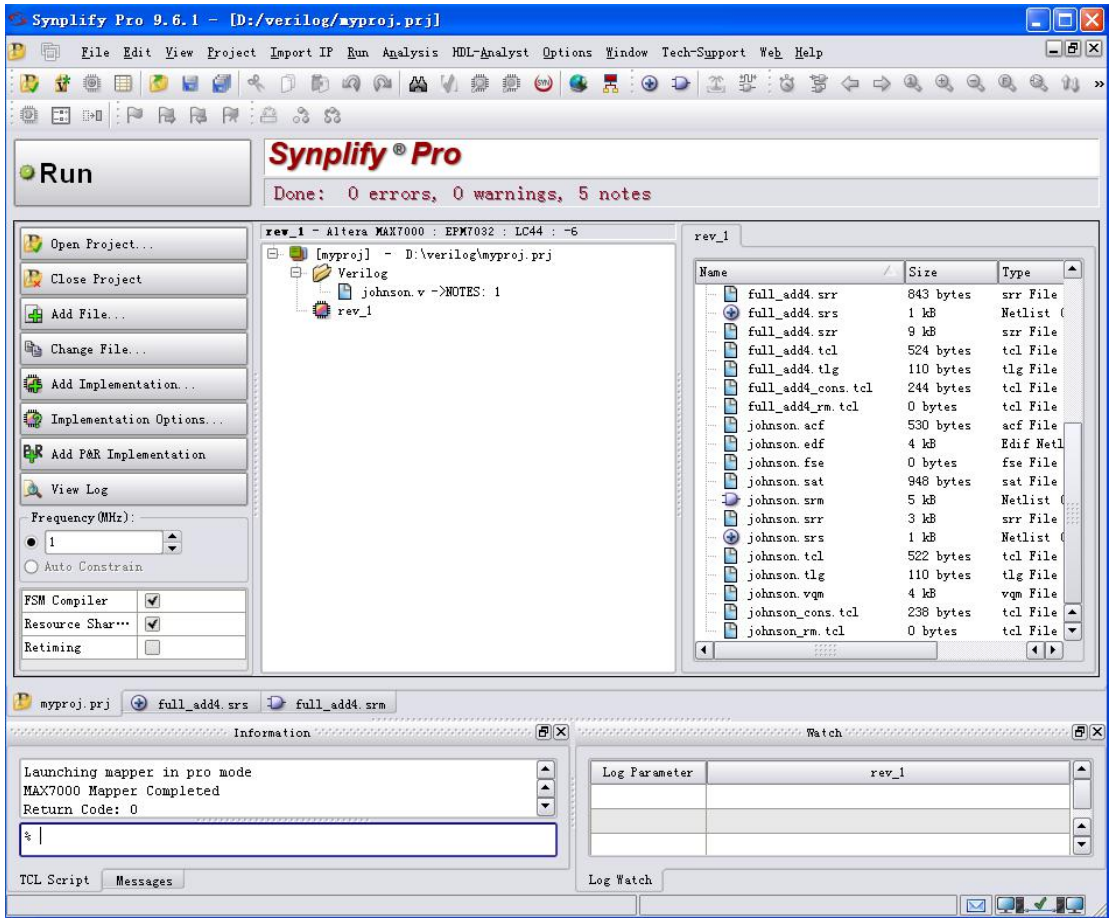
4.1 用Verilog设计一个8位加法器，进行综合和仿真，查看综合和仿真结果。

4.2 用Verilog设计一个8位二进制加法计数器，带异步复位端口，进行综合和仿真，查看综合和仿真结果。

4.3 用Verilog设计一个模60的BCD码计数器，进行综合和仿真，查看综合和仿真的结果

实验与设计

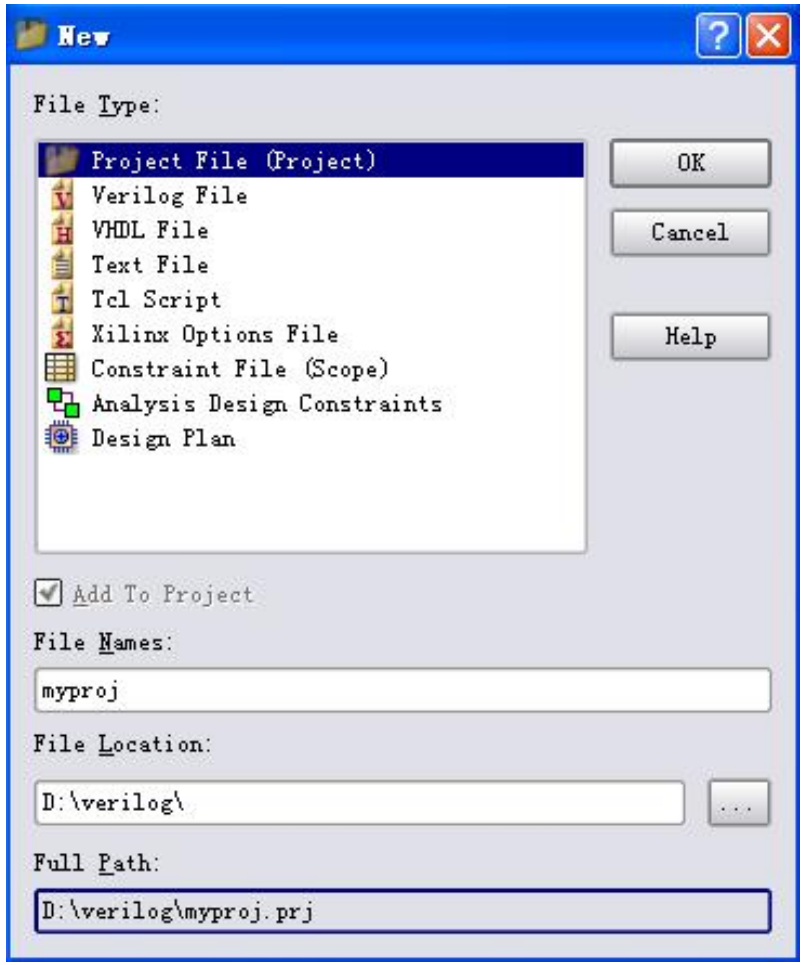
4-1 Synplify Pro综合器的使用方法



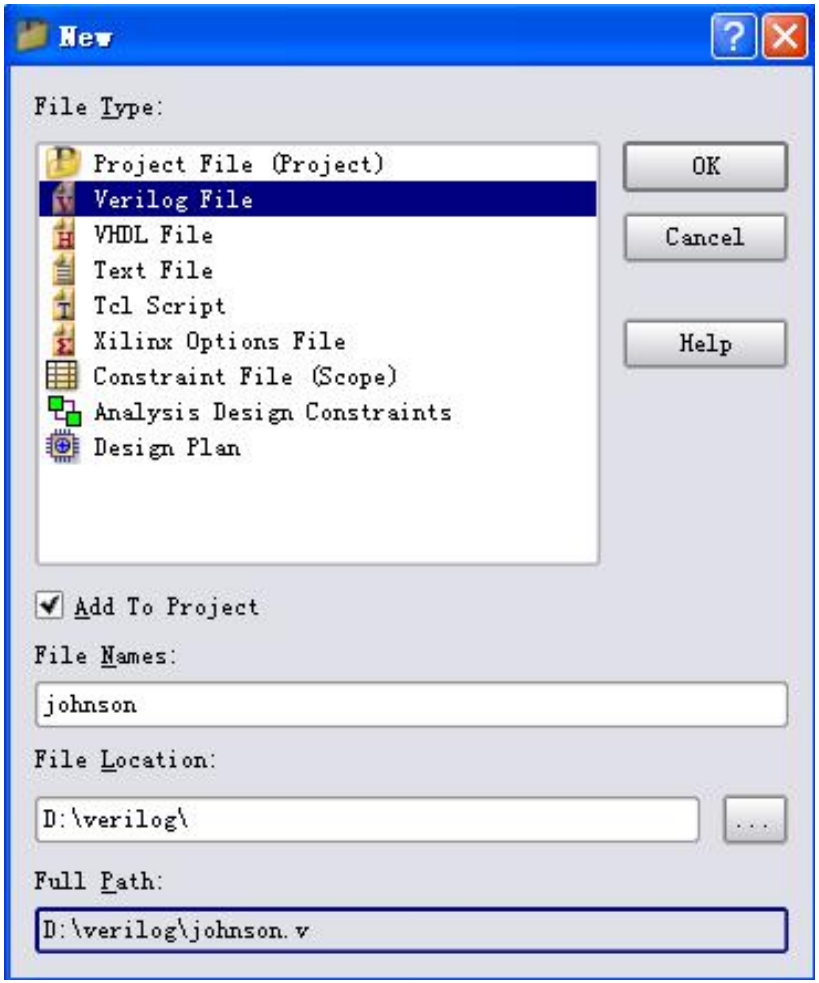
Synplify Pro的工作界面

【例4.15】 Johnson计数器（异步复位）

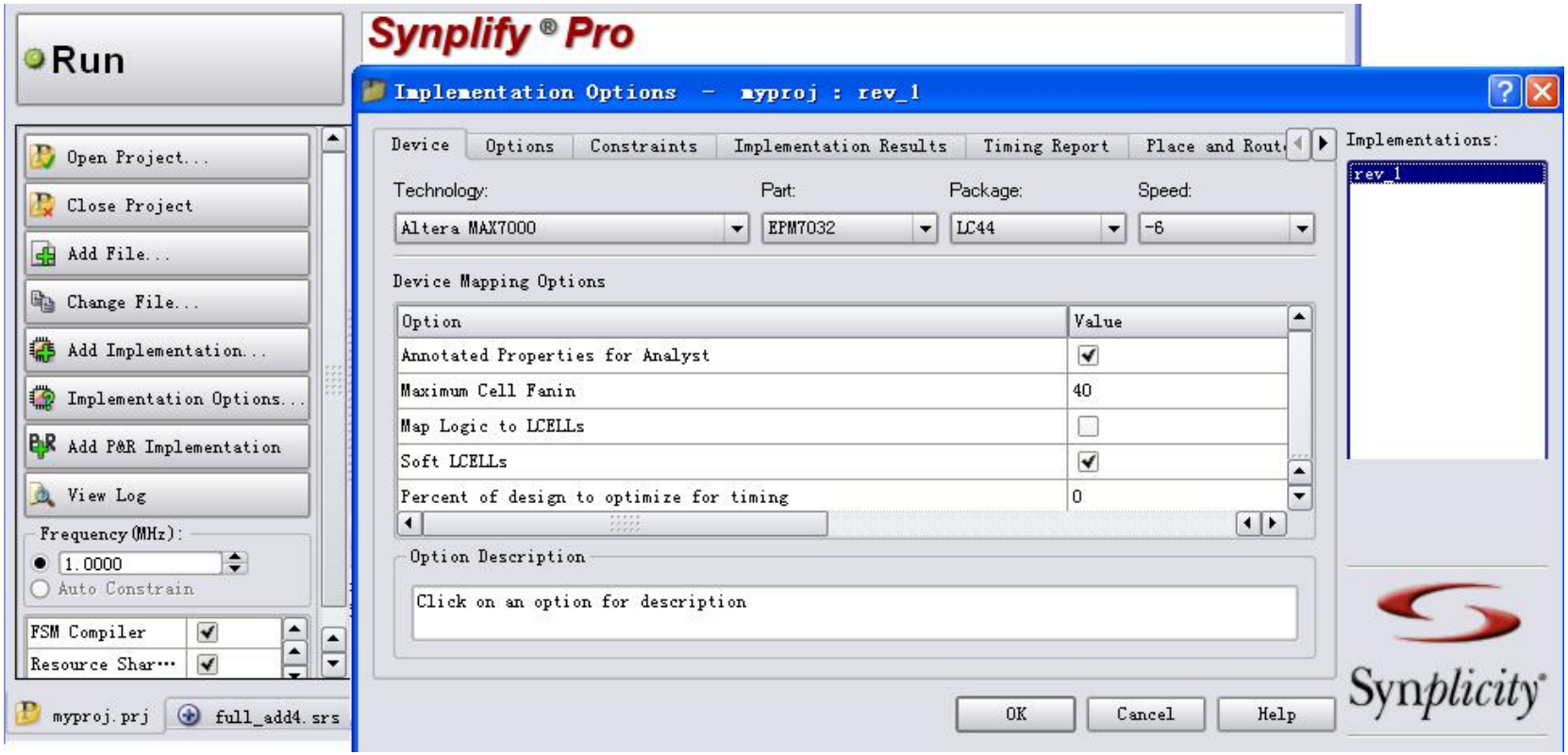
```
module johnson(clk,clr,qout);
parameter WIDTH=4;
input clk,clr;
output reg[(WIDTH-1):0] qout;
always @(posedge clk or posedge clr)
begin  if(clr)                qout<=0;
else       begin      qout<=qout<<1;
                  qout[0]<=~qout[WIDTH-1];
end
end
endmodule
```



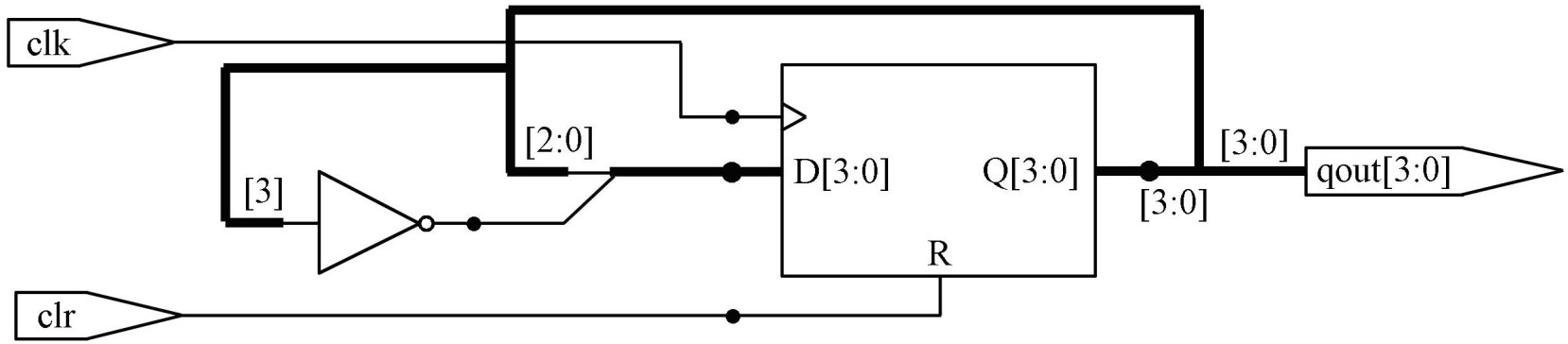
Synplify Pro新建项目对话框



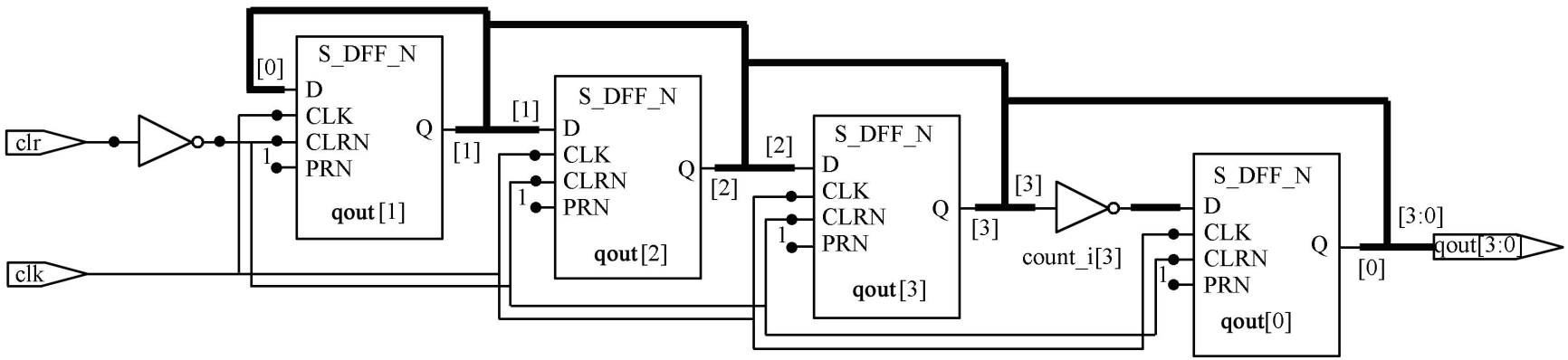
Synplify Pro新建文件对话框



Implementation Option对话框



约翰逊计数器综合后的RTL级原理图



约翰逊计数器综合后的门级原理图（MAX7000器件）