Verilog红宝书_基本语法_下

阿东

恒创科技

简 介

- 大家可以叫我阿东，我在通信行业做了6年的芯片设计，做了几款大型路由器和交换机芯片，写了6年的Verilog，对Verilog是熟悉的不能再熟悉了，对数据通信、QOS有深入研究和实现，精通数据通信各种协议，对通信网络有较深理解。精通ASIC、FPGA和Verilog架构、方案、实现设计。
- 希望我的经历能让大家掌握项目开发的编码规范和方案设计，给大家的学习和发展略进绵薄之力。大家使用过程中遇到什么问题，可以发邮件:1530384236@qq.com，QQ:1530384236。
- 后续还会推出更多FPGA相关资料，包括方案设计、视频教程、项目实战等，请大家关注我们。

- **淘宝店铺（本店专注于FPGA开发板开发）：**

http://shop67541132.taobao.com

http://shop69029874.taobao.com

- **开发板介绍（位于电子发烧友的阿东Verilog技术专版）：**

http://bbs.elecfans.com/jishu_348985_1_1.html

目　录

上 期 回 顾

1. 前途+钱途
2. 什么是HDL？
3. 为什么使用HDL？
4. VHDL还是Verilog?
5. Verilog的历史
6. Verilog的用途
7. Verilog设计层次
8. Verilog一个具体例子
9. Verilog基本语法-标识符
10. Verilog基本语法-注释
11. Verilog基本语法-数字
12. Verilog基本语法-数据类型
13. Verilog基本语法-运算符&表达式
14. Verilog基本语法-条件语句
15. Verilog基本语法-case语句

第一期主要讲解了Verilog的历史、重要性、基础语法前半部分。

上 期 总 结

1、Verilog基本语法和C语言有点类似，但是思想却完全不同。C语言编译的结果是指令，而Verilog编译的结果是电路，需要使用电路设计的思想去写Verilog。

2、大家要多看波形，建立时序概念，熟练画时序图进行时序设计。

3、现在绝对主流是Verilog进行数字设计。

4、Verilog最重要的是方案设计，掌握基本语法即可。

Verilog建模方式

- 结构级描述方式
  - 用基本单元(primitive)或低层元件(component)的连接来描述系统以得到更高的精确性，特别是时序方面。
  - 在综合时用特定工艺和低层元件将RTL描述映射到门级网表
- 数据流描述方式
  - 通过对数据流在设计中的具体行为的描述来建模
  - 一般采用连续赋值语句
- 行为级描述方式
  - 采用对信号行为级的描述方法来建模
  - 一般是always 块语句和assign块语句描述的称为行为建模方式

结构级描述方式

- 模块定义结构
- 模块端口
- 例化语句
- 实例

结构级描述方式-----模块定义结构

一个设计实际上是由一个个module 组成的，一个模块module 的结构如下：

module module_name (

port_list

) ;

Declarations_and_Statements

endmodule

```
module LED (
                    //input
input               sys_clk             ,    //system clock;
input               sys_rst_n           ,    //system reset, low is active;

                    //output
output reg [7:0]    LED
                );


//Parameter define
parameter WIDTH = 8                          ;
parameter SIZE  = 8                          ;
parameter WIDTH2 = 18                        ;
parameter Para = 100000                      ;

//Reg define
reg     [SIZE-1:0]          counter          ;
reg     [WIDTH2-1:0]        count            ;

//Wire define

//*************************************************************************************
//**                              Main Program
//**
//*************************************************************************************

// count for add counter
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (sys_rst_n ==1'b0)
            count <= 18'b0;
        else
            count <= count + 18'b1;
end
```

在结构建模中，描述语句主要是实例化语句，包括对Verilog HDL 内置门如与门（and）异或门（xor）等的例化，如全加器的xor 门的调用。

端口队列port_list 列出了该模块通过哪些端口与外部模块通信，在Verilog 2001语法中还包括输入输出、wire/reg类型、位宽定义。

结构级描述方式-----模块端口

- 模块的端口可以是输入端口、输出端口或双向端口。缺省的端口类型为线网类型（即wire 类型）。输入端口默认为wire类型，不需要定义，输出或双向端口能够声明为wire/reg 型，使用reg必须显式声明，使用wire也强烈建议显式声明。

- Verilog 2001语法中的输入输出包括端口名、输入输出、wire/reg类型、位宽定义，极大的减少了端口声明占用的代码行数。当前已经非常普及。

  例子：

```
module LED (
                    //input
input               sys_clk             ,    //system clock;
input               sys_rst_n           ,    //system reset, low is active;

                    //output
output reg [7:0]    LED
                );
```

✓ 强烈推荐2001语法的端口定义。

该例子包括输入、输出、姓名名称、位宽、输出的信号类型。

结构级描述方式-----例化

- 一个模块能够在另外一个模块中被引用，这样就建立了描述的层次。
- 模块实例化语句形式如下：

module_name instance_name(port_associations) ;

信号端口可以通过位置或名称关联；但是关联方式不能够混合使用。

名称关联形式如下：

使用名称关联的例子：

```
// instantance RR_CORE
RR_CORE  U_RR_CORE  (
.rr_en          ( rr_en          ),
.port_vld       ( key            ),
.last_sel_port  ( last_sel_port  ),

.port_win       ( port_win       )

);
```

✓ 推荐使用名称进行关联。

✓ 大规模电路设计的时候，悬空的端口最好使用XX_nc（wire类型）进行显式声明，提高检视代码的效率（芯片项目里面代码检视的工作量占到和写代码的时间差不多）。

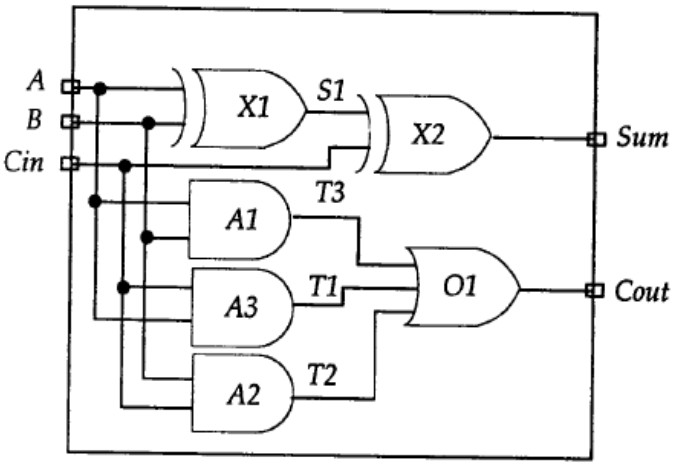
结构级描述方式-----实例

- 下面是一位全加器的结构级描述，采用Verilog基本单元(门)描述。

```
module FA_struct (
input A,
input B,
input Cin,
output  wire Sum,
output  wire Count
);
wire S1, T1, T2, T3;

 xor x1 (S1, A, B);
xor x2 (Sum, S1, Cin);
and A1 (T3, A, B );
and A2 (T2, B, Cin);
and A3 (T1, A, Cin);
or O1 (Cout, T1, T2, T3 );

 endmodule
```



- ✓ 结构级Verilog适合开发小规模元件，如ASIC和FPGA的单元
  - Verilog内部带有描述基本逻辑功能的基本单元(primitive)，如and门。
  - 综合产生的网表通常是结构级的。

数据流描述方式

- 连续赋值语句
- 阻塞赋值语句
- 非阻塞赋值语句
- 数据流建模具体实例

数据流描述方式-----连续赋值

- 数据流的描述是采用连续赋值语句(assign)语句来实现的。

- 连续赋值语句用于组合逻辑的建模。 等式左边是wire 类型的变量。等式右边可以是常量、由运算符如逻辑运算符、算术运算符参与的表达。

如下几个实例：

```
wire [3:0] Z, preset, clear; //线网说明
assign z = preset & clear;  //连续赋值语句
wire cout, cin ;
wire [3:0] sum, a, bB;
. . .
assign {cout, sum} = a + b + cin;
```

✓

实际电路中赋值语句的执行其实会有ps级别的延迟，这个延迟是线、门器件本身特性造成的，不过只要是同步电路，那么ps级别延迟就可以忽略不计，因为后续的寄存器是在时钟上升沿采样，ps级别延迟和一个典型的时钟周期来比，完全不用考虑。

注意：

1、连续赋值语句的执行是：只要右边表达式任一个变量有变化，表达式立即被计算，计算的结果立即赋给左边信号。

2、连续赋值语句之间是并行语句，因此与位置顺序无关。

数据流描述方式-----阻塞赋值

- “=”用于阻塞的赋值，凡是在组合逻辑（如在assign 语句中）赋值的请用阻塞赋值。阻塞赋值“=”在begin 和end 之间的语句是顺序执行，属于串行语句。

一个组合逻辑的例子：

```
always @(*) begin
    if ( new_vld_after == 1'b1 )
        port_win = new_port_after ;
    else if ( new_vld_before ==1'b1 )
        port_win = new_port_before ;
    else
        port_win = last_sel_port ;
end
```

注意：
1、always语句的敏感变量如果不含有时钟，即always（*）这样描述，那么也属于组合逻辑，需要使用阻塞赋值。

数据流描述方式-----非阻塞赋值

- “<=”用于阻塞的赋值，凡是在时序逻辑（如在always语句中）赋值的请用非阻塞赋值，非阻塞赋值“<=” 在begin 和end 之间的语句是并行执行，属于并行执语句。

一个时序逻辑的例子：

```
always @(posedge sys_clk or negedge sys_rst_n) begin
    if (sys_rst_n ==1'b0)
        clk_cnt <= 26'b0;
    else
        clk_cnt <= clk_cnt + 26'b1;
end
```

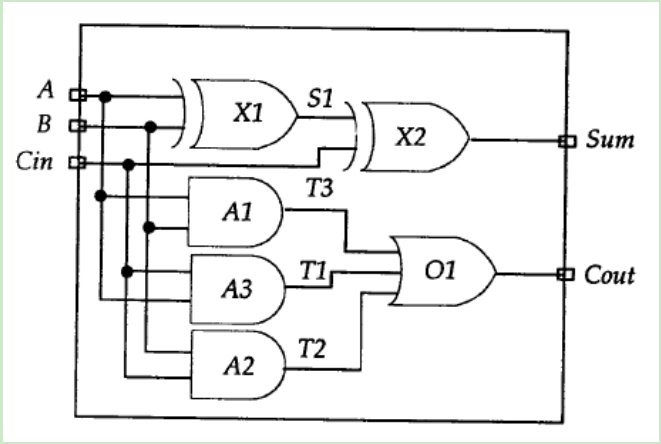
注意：

1、时序逻辑值的是带有时钟的always块逻辑，只有always带有时钟，这个逻辑才能综合为寄存器。

✓禁止阻塞和非阻塞赋值同时使用。

数据流描述方式-----实例

```
module FA_flow(
input A,
input B,
input Cin,
output wire Sum,
output wire Count
);
wire S1,T1,T2,T3;
assign # 2 S1 = A ^ B;
assign # 2 Sum = S1 ^ Cin;
assign #2 T3 = A & B;
assign #2 T1 = A & Cin;
assign #2 T2 = B & Cin ;
endmodule
```



- ✓ 数据流的建模方式就是通过对数据流在设计中的具体行为的描述的来建模。最基本的机制就是用连续赋值语句。
- ✓ 在连续赋值语句中，某个值被赋给某个线网变量（信号），语法如下：

  assign [delay] net_name = expression;

  如：assign #2 A = B;
- ✓ 在数据流描述方式中，还必须借助于HDL提供的一些运算符，如按位逻辑运算符 ：逻辑与（&），逻辑或（|）等。

行为级描述方式

- 顺序语句块
- 过程赋值语句块
- 状态机
- 实例

行为级别描述方式-----顺序语句块

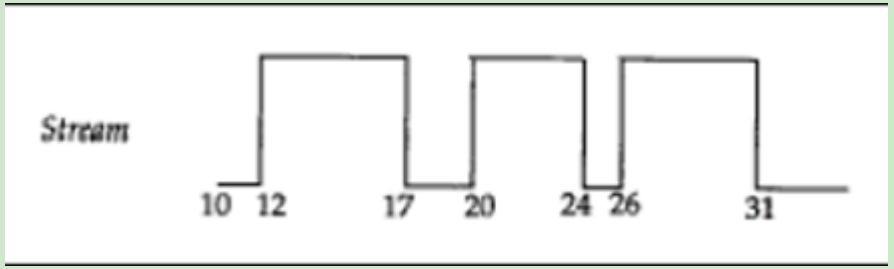
- 语句块提供将两条或更多条语句组合成语法结构上相当于一条语句的机制。本文主要讲Verilog HDL的顺序语句块(begin . . . end)：语句块中的语句按给定次序顺序执行。
- 顺序语句块中的语句按顺序方式执行。每条语句中的时延值与其前面的语句执行的模拟时间相关。一旦顺序语句块执行结束，跟随顺序语句块过程的下一条语句继续执行。

顺序语句块的语法如下：

```
begin [:block_id{declarations}]
procedural_statement (s)
end
```

例如：

```
// 产生波形:
begin
#2 Stream = 1;
#5 Stream = 0;
#3 Stream = 1;
#4 Stream = 0;
#2 Stream = 1;
#5 Stream = 0;
end
```



假定顺序语句块在第10个时间单位开始执行。两个时间单位后第1条语句执行，即第12个时间单位。此执行完成后，下1条语句在第17个时间单位执行(延迟5 个时间单位)。然后下1条语句在第20个时间单位执行，以此类推。

行为级别描述方式-----过程赋值语句

- Verilog HDL中提供两种过程赋值语句initial和always语句，用这两种语句来实现行为的建模。这两种语句之间的执行是并行的，即语句的执行与位置顺序无关。这两种语句通常与语句块（begin ....end）相结合，语句块中的执行是按顺序执行的。

- **initial 语句**

  initial语句只执行一次，即在设计被开始模拟执行时开始（0时刻）。通常只用在对设计进行仿真的测试文件中，用于对一些信号进行初始化和产生特定的信号波形。

例子如上产生一个信号波形：

```
initial
begin
#2 Stream = 1;
#5 Stream = 0;
#3 Stream = 1;
#4 Stream = 0;
#2 Stream = 1;
#5 Stream = 0;
end
```

注意：

✓ initial只能使用在仿真中，是不可综合语法，很多初学者在开始的时候以为initial是可以综合的。

✓ Initial不可以综合。

行为级别描述方式-----过程赋值语句

- **always语句**

  always语句与initial语句相反，是被重复执行，执行机制是通过对一个称为敏感变量表的事件驱动来实现的，下面会具体讲到。always 语句可实现组合逻辑或时序逻辑的建模。

例子1：

```
initial
clk = 0 ;
always
#5 clk = ~clk;
```

因为always 语句是重复执行的，因此，clk是初始值为0 的，周期为10 的方波。

例子2：D 触发器

```
always @ ( posedge clk or negedge rst ) begin
    if ( rst == 1' b0 )
        q <= ' b 0;
    else
        q <= d;
end
```

上面括号内的内容称为敏感变量，即整个always 语句当敏感变量有变化时被执行，否则不执行。因此，当rst为0时，q被复位，在时钟上升沿时，d被采样到q。

注意：
1、对组合逻辑的always 语句，敏感变量建议使用*替代。
2、对组合逻辑器件的赋值采用阻塞赋值 “=”。
3、时序逻辑器件的赋值语句采用非阻塞赋值 “<=”；

行为级别描述方式-----状态机

- 有限状态机英文名字，Finite State Machine，简称状态机，缩写为FSM。有限状态机是指输出取决于过去输入部分和当前输入部分的时序逻辑电路。有限状态机又可以认为是组合逻辑和寄存器逻辑的一种组合。状态机特别适合描述那些发生有先后顺序或者有逻辑规律的事情，其实这就是状态机的本质。状态机就是对具有逻辑顺序或时序规律的事件进行描述的一种方法。

根据状态机的输出是否与输入条件相关，可将状态机分为两大类，即摩尔（Moore）型状态机和米勒（Mealy）型状态机。

- **Mealy 状态机：时序逻辑的输出不仅取决于当前状态，还取决于输入。**
- **Moore 状态机：时序逻辑的输出只取决于当前状态。**

根据实际写法，状态机还可以分为一段式、二段式和三段式状态机。

- **一段式：把整个状态机写在一个 always 模块中，并且这个模块既包含状态转移，又含有组合逻辑输入/输出。**
- **二段式： 状态切换用时序逻辑，次态输出和信号输出用组合逻辑。**

三段式： 状态切换用时序逻辑，次态输出用组合逻辑，信号输出用时序逻辑。

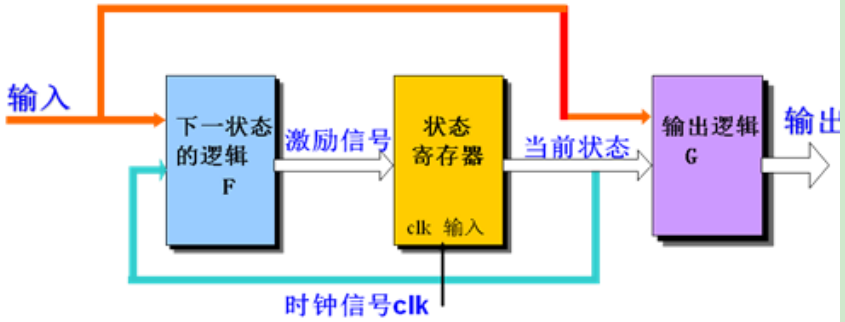
说明：

实际应用中三段式使用最多，也最为可靠，避免了状态和输入输出的干扰，推荐大家使用第三种写法，我们实际项目中基本全部是第三种写法。本文也着重讲解三段式。

行为级别描述方式-----状态机

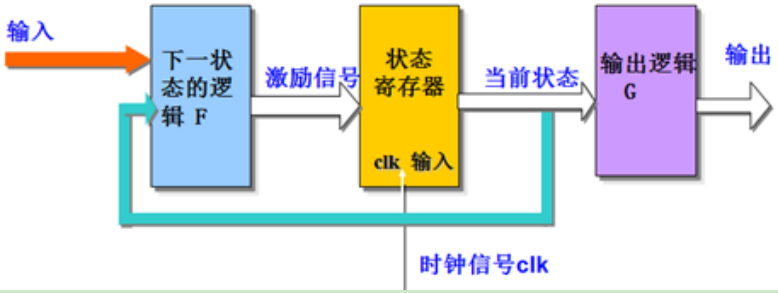
·Mealy 状态机

下一个状态 = F(当前状态，输入信号)；
输出信号 = G(当前状态，输入信号)；



·Moore 状态机

下一个状态 = F(当前状态，输入信号)；
输出信号 = G(当前状态)；



行为级别描述方式-----状态机

三段式状态机

两段式直接采用组合逻辑输出，而三段式则通过在组合逻辑后再增加一级寄存器来实现时序逻辑输出。这样做的好处是可以有效地滤去租个逻辑输出的毛刺，同时可以有效地进行时序计算与约束，另外对于总线形式的输出信号来说，容易使总线数据对齐，从而减小总线数据间的偏移，减小接收端数据采样出错的频率。

三段式状态机的基本格式是：

第一个always语句实现同步状态跳转；

第二个always语句实现组合逻辑；

第三个always语句实现同步输出。

Verilog描述状态机需要注意的事项：

- **定义模块名和输入输出端口；**
- **定义输入、输出变量或寄存器；**
- **定义时钟和复位信号；**
- **定义状态变量和状态寄存器；**
- **用时钟沿触发的 always 块表示状态转移过程；**
- **在复位信号有效时给状态寄存器赋初始值；**
- **描述状态的转换过程：符合条件，从一个状态到另外一个状态，否则留在原状态；**
- **验证状态转移的正确性，必须完整和全面。**

行为级别描述方式-----状态机例子

一个三段式状态机例子：【配套光盘中进阶类中 7分频的分频器】

```
module divider7_fsm (
//input
input       sys_clk        ,     // system clock;
input       sys_rst_n      ,     // system reset, low is active;

              //output
output reg  clk_divide_7         // output divide 7 clk
              );

//reg define
reg [6:0]      curr_st      ;    // FSM current state
reg [6:0]      next_st      ;    // FSM next    state
reg            clk_divide_7 ;    // generated clock, divide by 7

//wire define

//parameter define

//one hot code design
parameter S0 = 7'b0000000;
parameter S1 = 7'b0000001;
```

```
parameter S2 = 7'b0000010;
parameter S3 = 7'b0000100;
parameter S4 = 7'b0001000;
parameter S5 = 7'b0010000;
parameter S6 = 7'b0100000;
```

行为级别描述方式-----状态机

```
//generate FSM next state
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (sys_rst_n ==1'b0) begin
            curr_st <= 7'b0;
        end
        else  begin
            curr_st <= next_st;
        end
end

//FSM state logic
always @(*) begin
    case (curr_st)
        S0: begin
                next_st = S1;
            end
        S1: begin
                next_st = S2;
            end
        S2: begin
                next_st = S3;
            end
        S3: begin
                next_st = S4;
            end
        S4: begin
                next_st = S5;
            end
        S5: begin
                next_st = S6;
            end
        S6: begin
                next_st = S0;
            end
        default: next_st = S0;
    endcase
end
```

```
//control divide clock offset
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (sys_rst_n ==1'b0) begin
            clk_divide_7 <= 1'b0;
        end
        else if ((curr_st == S0) | (curr_st == S1) | (curr_st == S2)
                | (curr_st == S3))
            clk_divide_7  <= 1'b0;
        else if ((curr_st == S4) | (curr_st == S5) | (curr_st == S6))
            clk_divide_7  <= 1'b1;
        else
            ;
end

endmodule
//end of RTL code
```

行为级别描述方式-----状态机

说明：

1、 本状态机采用独热码设计，简称one-hot code，独热码编码的最大优势在于状态比较时仅仅需要比较一个位，从而一定程度上简化了译码逻辑。

2、 一般状态机状态编码使用二进制编码、格雷码、独热码。

各种编码比较:

二进制编码、格雷码编码使用最少的触发器，消耗较多的组合逻辑，而独热码编码反之。独热码编码的最大优势在于状态比较时仅仅需要比较一个位，从而一定程度上简化了译码逻辑。虽然在需要表示同样的状态数时，独热编码占用较多的位，也就是消耗较多的触发器，但这些额外触发器占用的面积可与译码电路省下来的面积相抵消。

Binary（二进制编码）、gray-code（格雷码）编码使用最少的触发器，较多的组合逻辑，而 one-hot（独热码）编码反之。one-hot 编码的最大优势在于状态比较时仅仅需要比较一个 bit，一定程度上从而简化了比较逻辑，减少了毛刺产生的概率。另一方面，对于小型设计使用 gray-code 和 binary 编码更有效，而大型状态机使用 one-hot 更高效。

行为级别描述方式----实例

```
module LED (
                //input
input           sys_clk          ,    //system clock;
input           sys_rst_n        ,    //system reset, low is active;

                //output
output reg [7:0]  LED
            );


//Parameter define
parameter WIDTH = 8                     ;
parameter SIZE  = 8                     ;
parameter WIDTH2 = 18                   ;
parameter Para = 100000                 ;

//Reg define
reg    [SIZE-1:0]       counter          ;
reg    [WIDTH2-1:0]     count            ;

//Wire define

//*****************************************************************************
//**                          Main Program
//**
//*****************************************************************************

// count for add counter
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (sys_rst_n ==1'b0)
            count <= 18'b0;
        else
            count <= count + 18'b1;
end
```

```
// counter for delay time to LED display
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (sys_rst_n ==1'b0)
            counter <= 8'b0;
        else if ( count == Para)
            counter  <= counter + 8'b1;
        else ;
end

// ctrl LED pipeline display when counter is equal 10 or 20
always @(posedge sys_clk or negedge sys_rst_n) begin
        if (sys_rst_n ==1'b0)
            LED <= 8'b0;
        else begin
            case (counter)
                8'd10     : LED <= 8'b10000000   ;
                8'd20     : LED <= 8'b01000000   ;
                8'd30     : LED <= 8'b00100000   ;
                8'd40     : LED <= 8'b00010000   ;
                8'd50     : LED <= 8'b00001000   ;
                8'd60     : LED <= 8'b00000100   ;
                8'd70     : LED <= 8'b00000010   ;
                8'd80     : LED <= 8'b00000001   ;
                default   : LED <= 8'b00000000   ;
            endcase
        end
end

endmodule
//end of RTL code
```

- ✓ 行为方式的建模是指采用对信号行为级的描述的方法来建模。
- ✓ 在表示方面，类似数据流的建模方式，但一般是always 块语句和assign块语句描述的归为行为建模方式。
- ✓ 行为建模方式通常需要借助一些行为级的运算符如加法运算符（+），减法运算符（-）等。

本 期 总 结

1、当前数字逻辑规模越来越大，使用结构化描述已经越来越不现实，也没有必要，当前行为描述方式已经占据绝对的主导地位，大家学习的时候知道有这两种方式即可，重点学习行为描述。

2、我们经历了很多个芯片项目，全部使用的是行为模式方式。

3、基于Verilog进行方案设计是重点之重。

附录1　暴 风 系 列 开 发 板 简 介

中端2C5 FPGA开发板208元套餐(包括开发板+USB Blaster+1602):

适合没有任何基础、有一定基础，可以用来做学习和一般项目、NIOS。

- 淘宝网址：

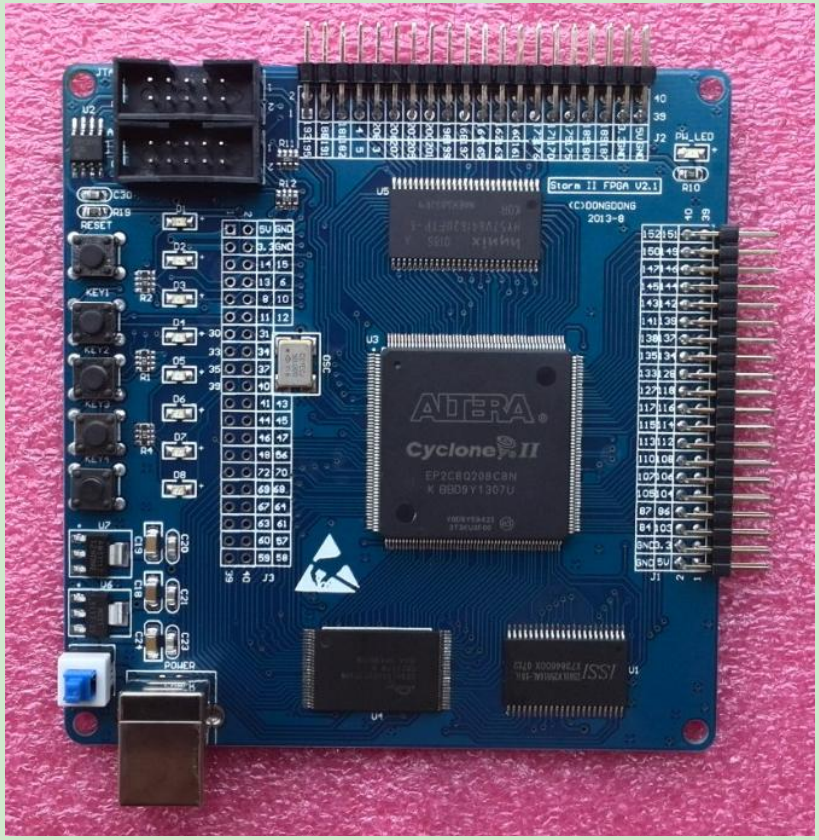
http://item.taobao.com/item.htm?spm=a230r.1.14.37.3IZ4fw&id=18309760936

附录2　暴 风 系 列 开 发 板 简 介

中端2C8 FPGA开发板288元套餐(包括开发板+USB Blaster+1602):

适合没有任何基础、有一定基础、基础较好，可以用来做学习和一般项目、NIOS。

板载SDRAM/SRAM/FLASH



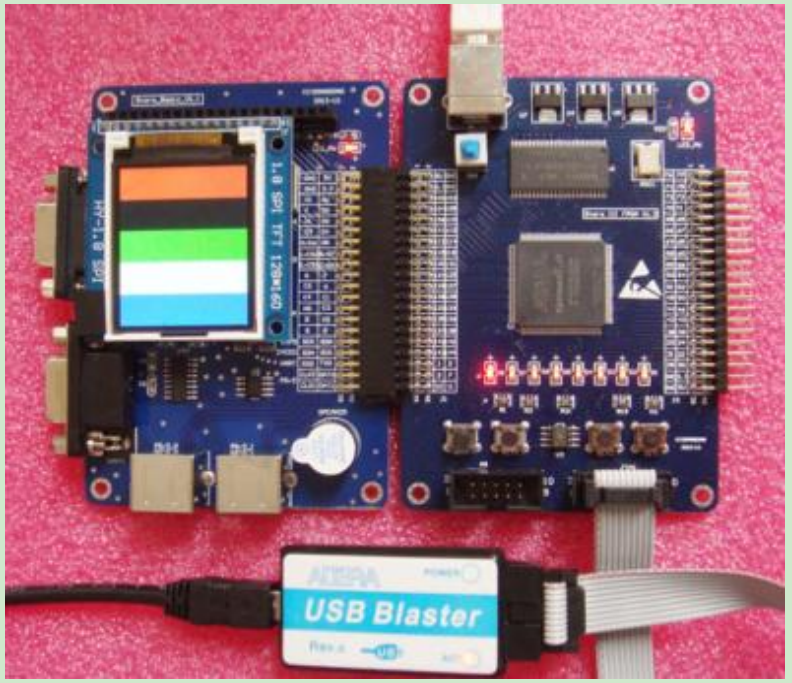
- **淘宝网址：**

http://item.taobao.com/item.htm?spm=a1z10.1.w4004-1006554551.33.2vMMCN&id=27124996625

附录3　暴 风 系 列 开 发 板 简 介

高端3C10 FPGA开发板288元套餐(包括核心板+扩展板+USB Blaster+亚克力壳):

适合没有任何基础、有一定基础、基础较好，可以用来做学习和高级项目、NIOS。



- 淘宝网址：

http://item.taobao.com/item.htm?spm=a1z10.1.w4004-1006554551.5.Z5bNtc&id=17385063452

附录4 暴风系列开发板简介

高端4CE15 FPGA开发板420元套餐(包括核心板+扩展板+USB Blaster+亚克力壳)：

适合有一定基础、基础较好，想一步到位（学习和复杂项目都可以）的同学，可以用来做学习和高级项目、NIOS。

板载SDRAM/SRAM/FLASH

- 淘宝网址：

http://item.taobao.com/item.htm?spm=a1z10.1.w4004-2613814661.5.IHzdr5&id=25969336978

播放结束，谢谢观看！