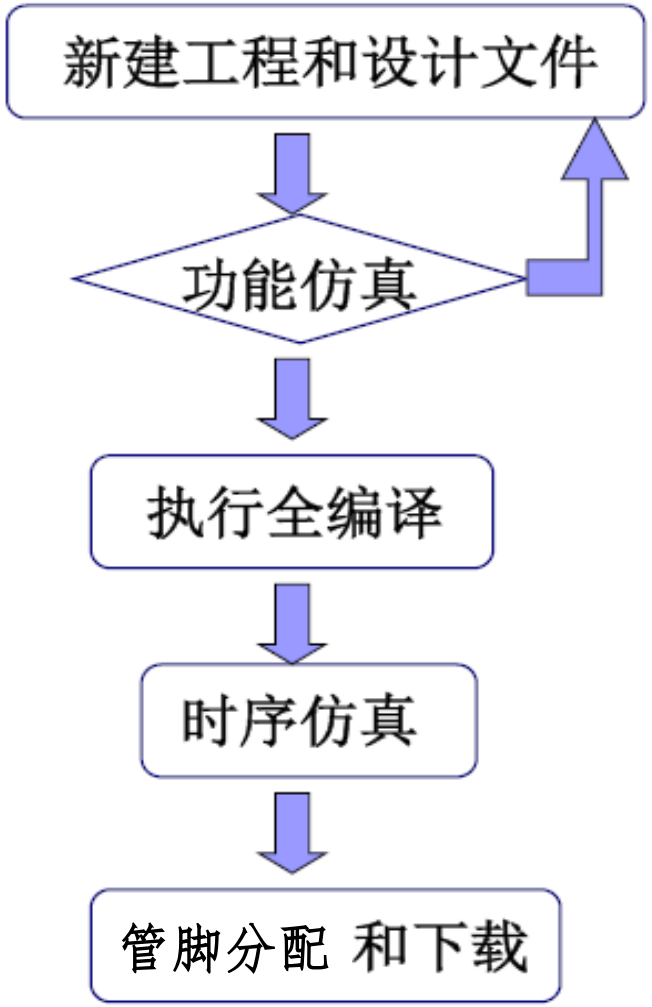
计算机科学基础 C

回　顾

回 顾



硬件描述语言、原理图

建立波形文件
检查逻辑功能是否正确

综合、布局布线

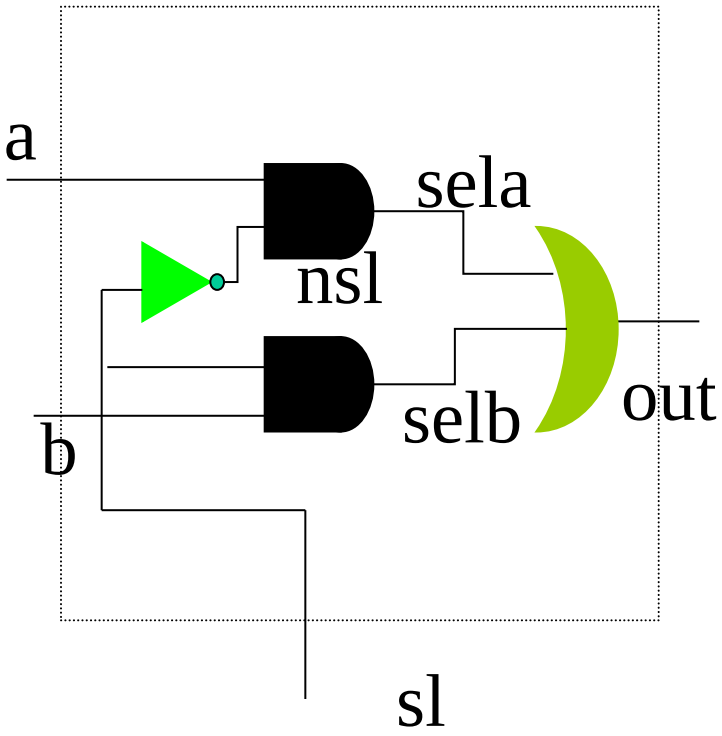
验证电路的时序

下载验证

Module八股

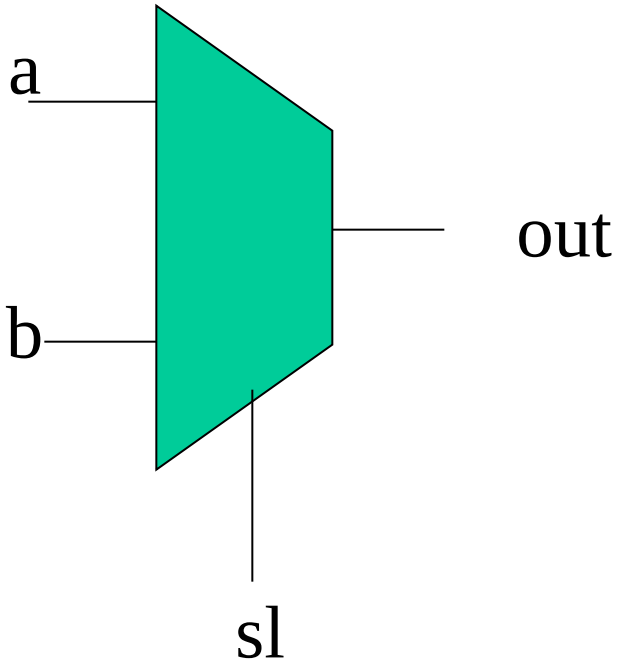
1. module 名字 (出入口声明);
2. 出入口方向位宽声明；　　//用于指定input/output
3. reg/wire;　　//always中被赋值的为reg，连线用wire；
4. always块列写时序或组合电路块;
5. assign列写连线或组合电路块
6. endmodule

结构描述




```
module muxtwo (out, a, b, sl);
input a,b,sl;
output out;
not  u1（ns1，sl）；
and  u2（sela，a，nsl）；
and  u3 （selb，b，sl）；
or   u4（out，sela，selb）；
endmodule
```

行为描述




```
module muxtwo (out, a, b, sl);
input a,b,sl;
output out;
Reg out;
always @(sl or a or b)
    if (!sl)  out = a;
    else out = b;
endmodule
```

Verilog 语法强化

课程内容

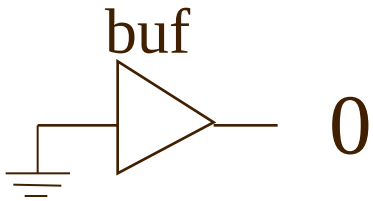
- ❋ Verilog 基础语法
- ❋ Verilog 设计技巧与实践

Verilog 基础语法

- 逻辑值
- 运算与逻辑
- 信号类型
- 位拼接
- always 和 assign
- if 和 case
- 代码八股
- 状态机八股

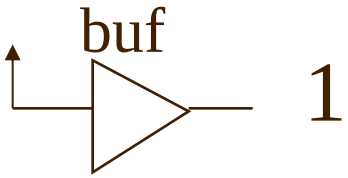
逻辑值

四种逻辑值



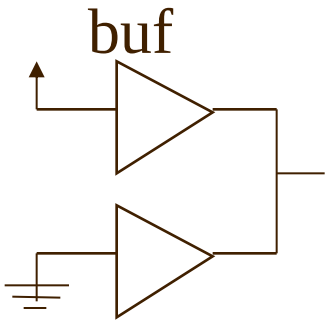
0

0：低、伪、逻辑低、地、VSS、负插入



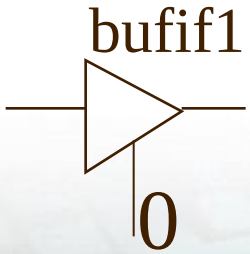
1

1：高、真、逻辑高、电源、VDD、正插入



X

X：不确定、逻辑冲突无法确定其逻辑值



Z

HiZ：高阻抗、三态、无驱动源

运算与逻辑

运算与逻辑

✱ 运算：

☞ 加 减 乘 除 模

＋ － × / %

乘除模用的少
除法和模不可综合

✱ 逻辑：

☞5个值、4个位、3个减约、2个移位、1个条件 （共19个）

- ➢与 或 非 等 不等
- ➢与 或 异或 非
- ➢与 或 异或
- ➢左移 右移 大于（等于） 小于（等于）
- ➢? :

5个值操作

✱5个值操作：（与　或　非　等　不等）

对两个信号逻辑值的操作，结果只有真假(1/0)

用于描述逻辑关系，同时成立ok则&&，之一成立ok则||，不成立ok则！

- ☞ a: 5’b10101　b: 5’b00011
- ☞与&&　a&&b =
- ☞或||　a||b =
- ☞非!　!a =
- ☞等==　a==b =
- ☞不等!=　a!=b =

4个位操作

✱4个位操作：（与 或 异或 非）

对两个信号对应位的操作，

用于对数据内各个位进行操作，加工。

- ☞ a: 5’b10101　　b: 5’b00011
- ☞与&　　a & b ＝
- ☞或|　　a | b ＝
- ☞异或^　　a ^ b ＝
- ☞ 非~　　～a　＝

3个归约操作

✱3个归约操作：（与 或 异或）

对一个信号从右向左相邻为操作的结果在与左侧相邻为操作，

- ☞ a: 5’b10101
- ☞与&　　&a =
- ☞或|　　| a =
- ☞异或^　　^a =

2个移位关系

✱2个移位操作：（左移　右移）

对操作数移n位补零

- ☞ a: 5’b10101
- ☞左移位 <<:　　a << 2 =
- ☞右移位 >>:　　a >> 3 =

✱2个关系操作：（小于（等于）　大于（等于））

- ☞a: 5’b10101　b: 5’b00011
- ☞小于　　a<b =　　a<=b=
- ☞大于　　a>b =　　a>=b=

1个条件操作

✱1个条件操作：（？ ：）
与assign一同生成带条件的组合电路

☞Eg： assign c = (a>b) ? a : b;

逻辑的应用

✱判断一个8位输入a是奇数还是偶数 ？

- ☞ a%2
- ☞

逻辑的应用

✱一个位宽为8输入，是否是处于 0 – 15 之间的数 ？

☞(a >= 0) && (a <= 15)

☞

逻辑的应用

❋待补充 ……

❋发现之后请分享

信号类型

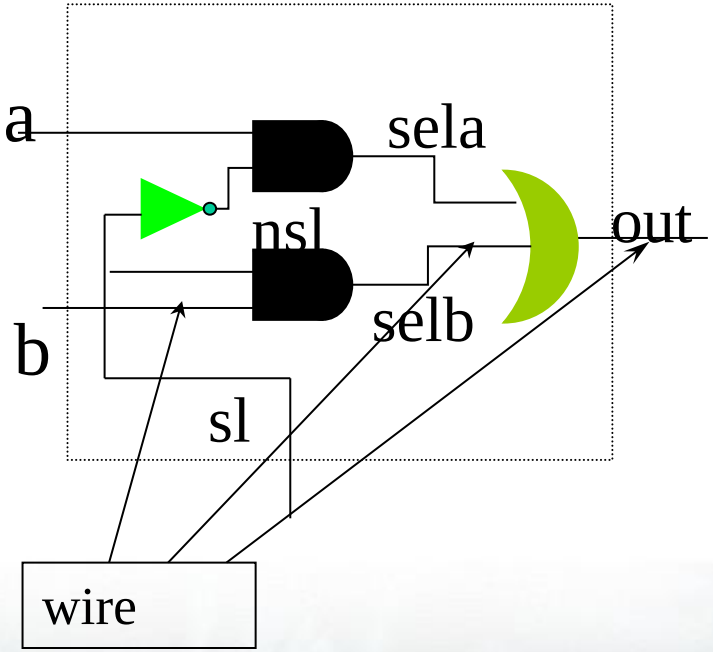
信号类型

- 连线：
- 寄存器：
- 寄存器组：
- 整数：
- 参数：
- 预处理：

信号类型

✱ 连线：

- ☞ 多用于描述物理连线，如各个子模块间的连接等。
- ☞ 连线不可存储数据。
- ☞ 不明确声明信号类型，默认 **wire** 类型。
- ☞ **wire a;** //定义一个1位位宽的线a
- ☞ **wire[7:0] b;** //定义一个8位位宽的线b



信号类型

✱ 寄存器：

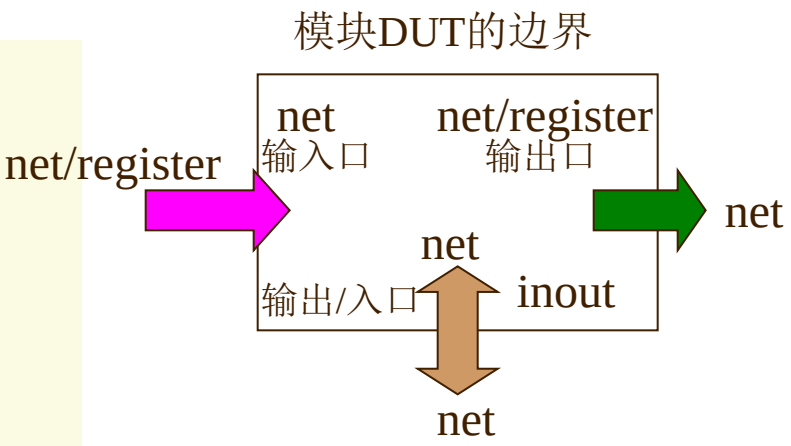
- ☞用于描述存储体，存无符号数。
- ☞ **reg a;** //定义一个1位位宽寄存器类型a
- ☞ **reg[7:0] b;** //定义一个8位位宽寄存器类型b

信号类型

连线和寄存器的选择

```
module top;
wire y;
 reg a, b;
 DUT u1(y,a,b);
 initial
   begin
   a = 0; b = 0;
   #10 a =1; ....
   end
endmodule
```

```
module DUT(Y, A, B);
output Y;
input A,B:
 wire Y, A, B;
   and (Y, A, B);
endmodule
```



仅过程快中被赋值的用reg，其余全用wire；

出入双向口，实例化外接口用wire。

信号类型

✱ 寄存器组：

- ☞用于定义一组寄存器。
- ☞ reg[1:0] a[15:0]; //定义16个2位位宽寄存器组a，
- ☞ a[5]对应两位寄存器。
- ☞ a[5] ＝ 2’b10;

信号类型

❋ 整数：

- ☞用于描述存储体，存有符号数。
- ☞位宽为32位。
- ☞integer a；　4’d3, 4’b0100, 6’h20

信号类型

✱ 参数：

☞用于位宽暂不确定或实例化时需改写参数的情况

```
module md1(out,in1,in2);
.....
parameter  p1=7, x_word=16'bx,
           file = "/user1/jmdong/design/mem_file.dat";

wire [p1:0] w1;     //用参数来说明wire 的位宽
  ....
    initial  begin  $open(file);  ....... #20000  display("%s",file);
   $stop  end
  ....
endmodule
```

参数

参数值的改写(方法之一，名称重赋值)

举例说明:

```
module mod ( out, ina, inb);
...
parameter cycle = 8, real_constant=2.039,
                    file = "/user1/jmdong/design/mem_file.dat";
...
endmodule

module test;
   ...
  mod mk(out,ina,inb);
  defparam  mk.cycle=6, mk.file="../my_mem.dat";
   ...
endmodule
```

参数值的改写(方法之二，位置重赋值)

```
module mod ( out, ina, inb);
...
parameter cycle = 8, real_constant=2.039,
          file = "/user1/jmdong/design/mem_file.dat";
...
endmodule

module test;
   ...
  mod # (5, 3.20, "../my_mem.dat")  mk(out,ina,inb);
   ...
endmodule
```

信号类型

✱ 预处理：

☞代码编写方便，增加可读性

☞**`define** <宏名> <宏文本>　//无分号

```
`define ON   1'b1
`define OFF  1'b0
`define AND_DELAY  #3
```

位拼接

位拼接

* 位拼接：{　}

  - { a, b[3:0], w, 3’b101}
  - {4{w}}

always 和 assign

Always 块

always@(敏感表)

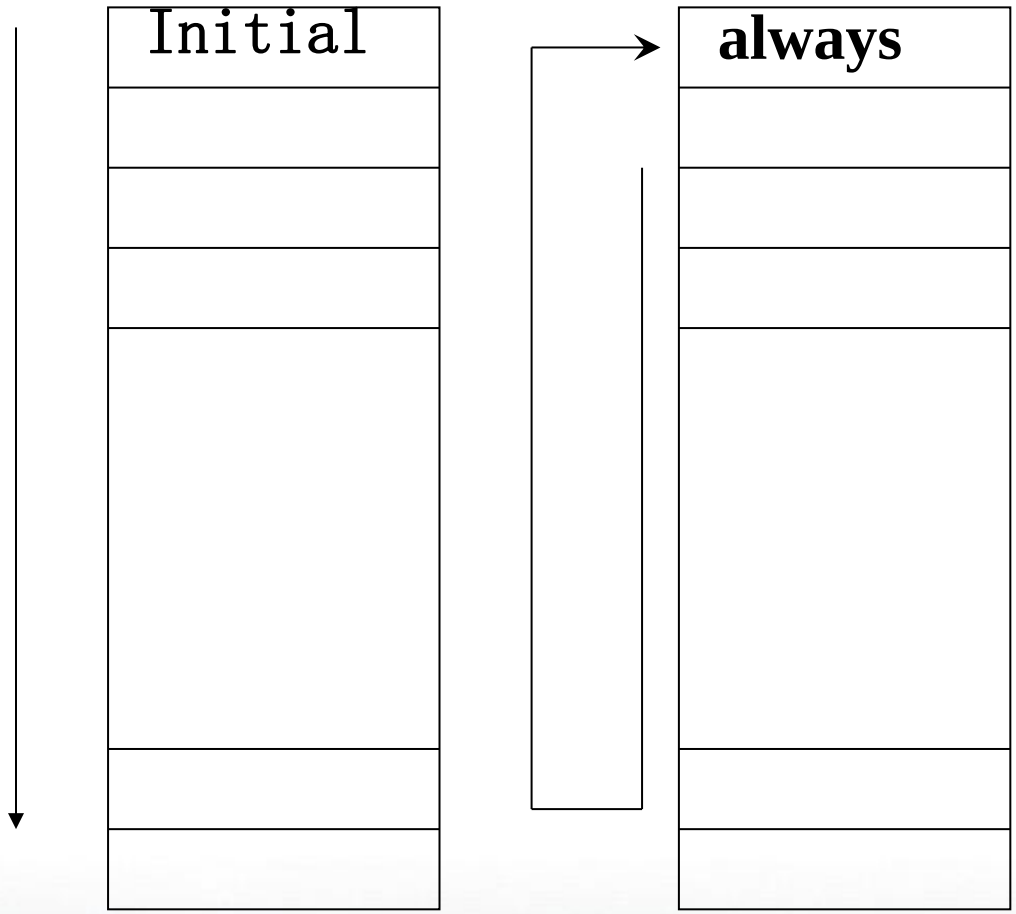
begin

……

end

- ☞由敏感表触发，执行always块内语句
- ☞电平敏感 @(a or b or c)
- ☞沿敏感 @(posedge clock or negedge reset)
- ☞If-else case 只能在过程块中使用

Always 块



Initial 不可综合

assign

```
assign reset_n = iKEY[1];

assign z = left  ?  Left   :   right;
```

- 用于模块间的连线
- 用于条件选择的组合电路
- 用于实现纯组合逻辑

If 和 case

If

Syntax

```
if (expression)
begin
   ...statements...
end

else if (expression)
begin
  ...statements...
end
  ...more else if blocks
else
begin
  ...statements...
end
```

```
if (alu_func == 2'b00)
  aluout = a + b;
else if (alu_func == 2'b01)
  aluout = a - b;
else if (alu_func == 2'b10)
  aluout = a & b;
else // alu_func == 2'b11
  aluout = a | b;
```

Case

Syntax

```
case (expression)
 case_choice1:
 begin
  ...statements...
 end
 case_choice2:
 begin
  ...statements...
 end
 ...more case choices blocks...
 default:
 begin
  ...statements...
 end
endcase
```

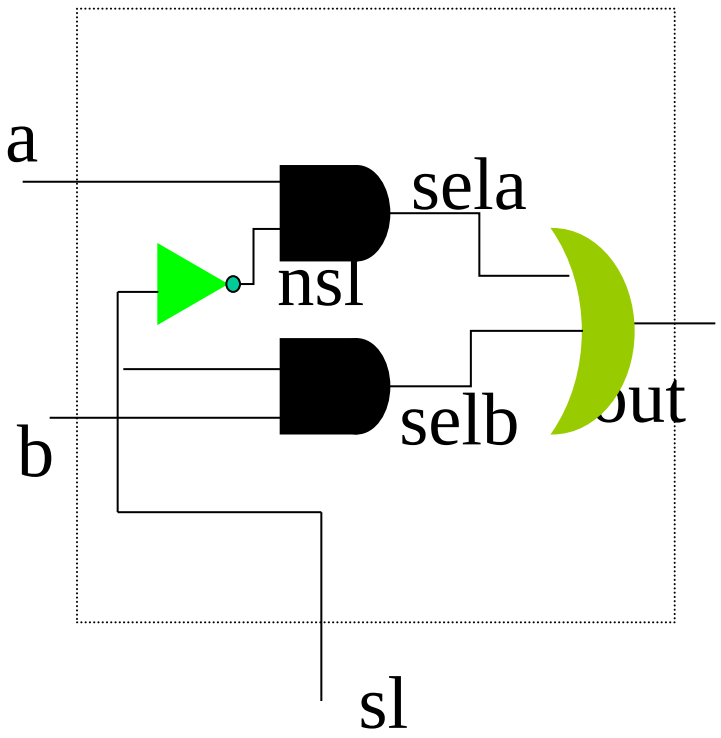
```
case (alu_ctr)
  2'b00:  aluout = a + b;
  2'b01:  aluout = a - b;
  2'b10:  aluout = a & b;
  default: aluout = 1'bx; // Treated as don't cares for
endcase                   // minimum logic generation
```

两种代码

两种代码

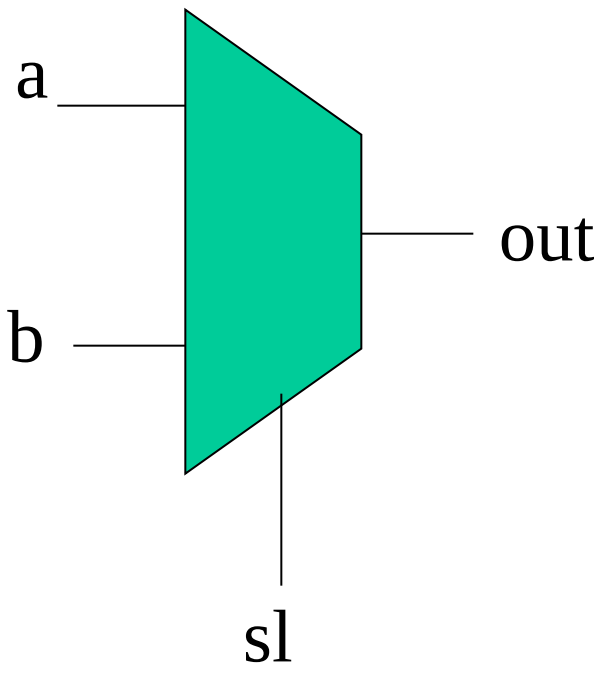
- 结构描述
- 行为级RTL描述

结构描述




```
module muxtwo (out, a, b, sl);
input a,b,sl;
output out;
    not  u1（ns1，sl）；
    and  u2（sela，a，nsl）；
    and  u3 （selb，b，sl）；
    or   u4（out ，sela，selb）；
endmodule
```

行为描述




```
module muxtwo (out, a, b, sl);
input a,b,sl;
output out;
Reg out;
   always @(sl or a or b)
        if (!sl)  out = a;
            else out = b;
endmodule
```

Verilog八股文

Module八股

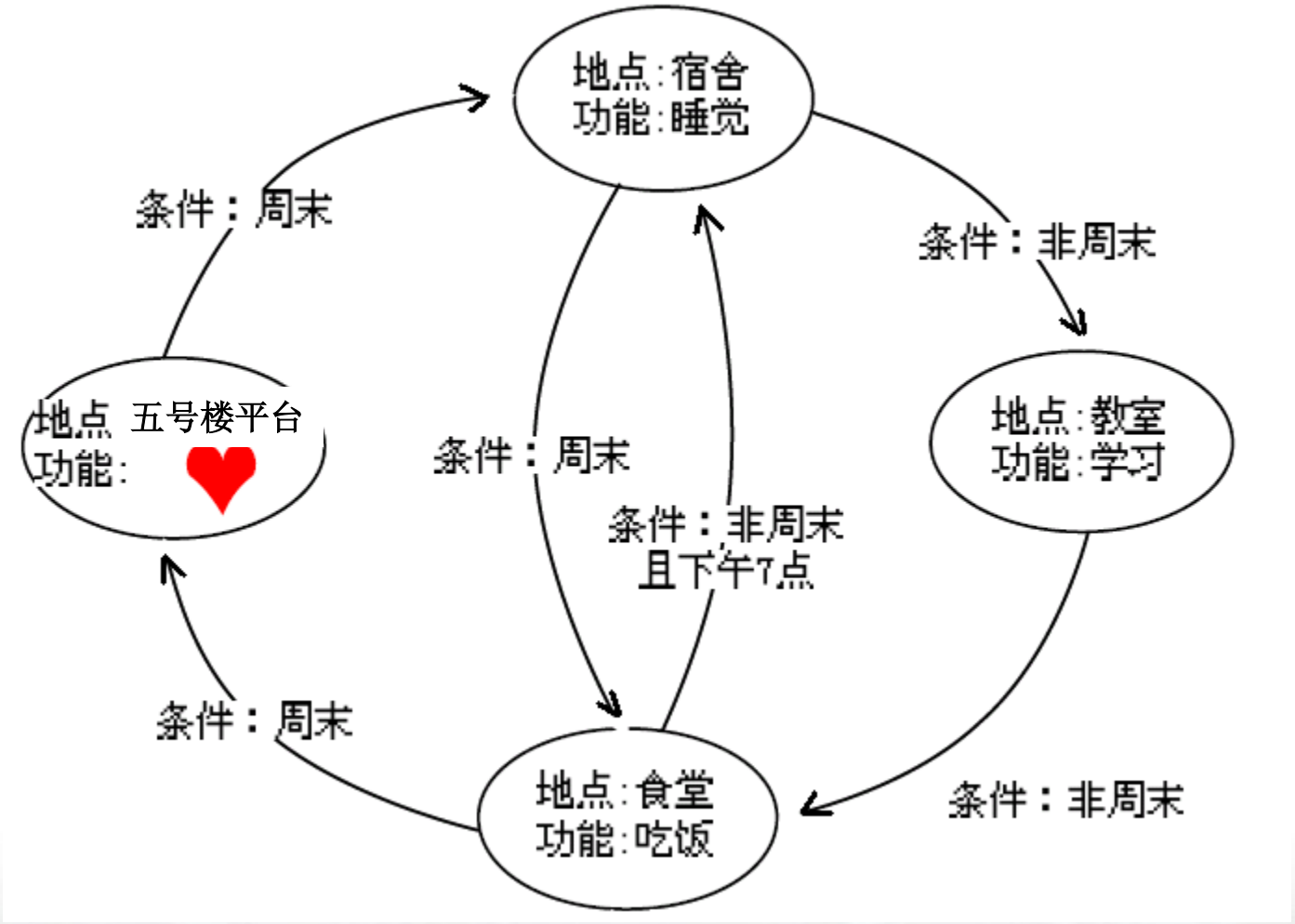
1. module 名字 (出入口声明);
2. 出入口方向位宽声明；　　//用于指定input/output
3. reg/wire;　　//always中被赋值的为reg，连线用wire；
4. always块列写时序或组合电路块;
5. assign列写连线或组合电路块
6. endmodule

八股实例

```
module alu(alu_out, zero, opcode, data, accum);
  input [7:0] data, accum;
  input [2:0] opcode;
  output [7:0] alu_out;
  output zero;

  reg zero;
  reg [7:0] alu_out;

  `define pass_accum  3'b000, 3'b001, 3'b110, 3'b111
  `define Add         3'b010
  `define And         3'b011
  `define Xor         3'b100
  `define pass_data   3'b101

  always @(accum)
  zero = (!accum);

  always @(accum or data or opcode)
    case (opcode)
      `Add        : alu_out = accum + data;
      `And        : alu_out = accum & data;
      `Xor        : alu_out = accum ^ data;
      `pass_data  : alu_out = data ;
      `pass_accum : alu_out = accum;
      default     : alu_out = 8'bx;
    endcase
endmodule
```

有限状态机

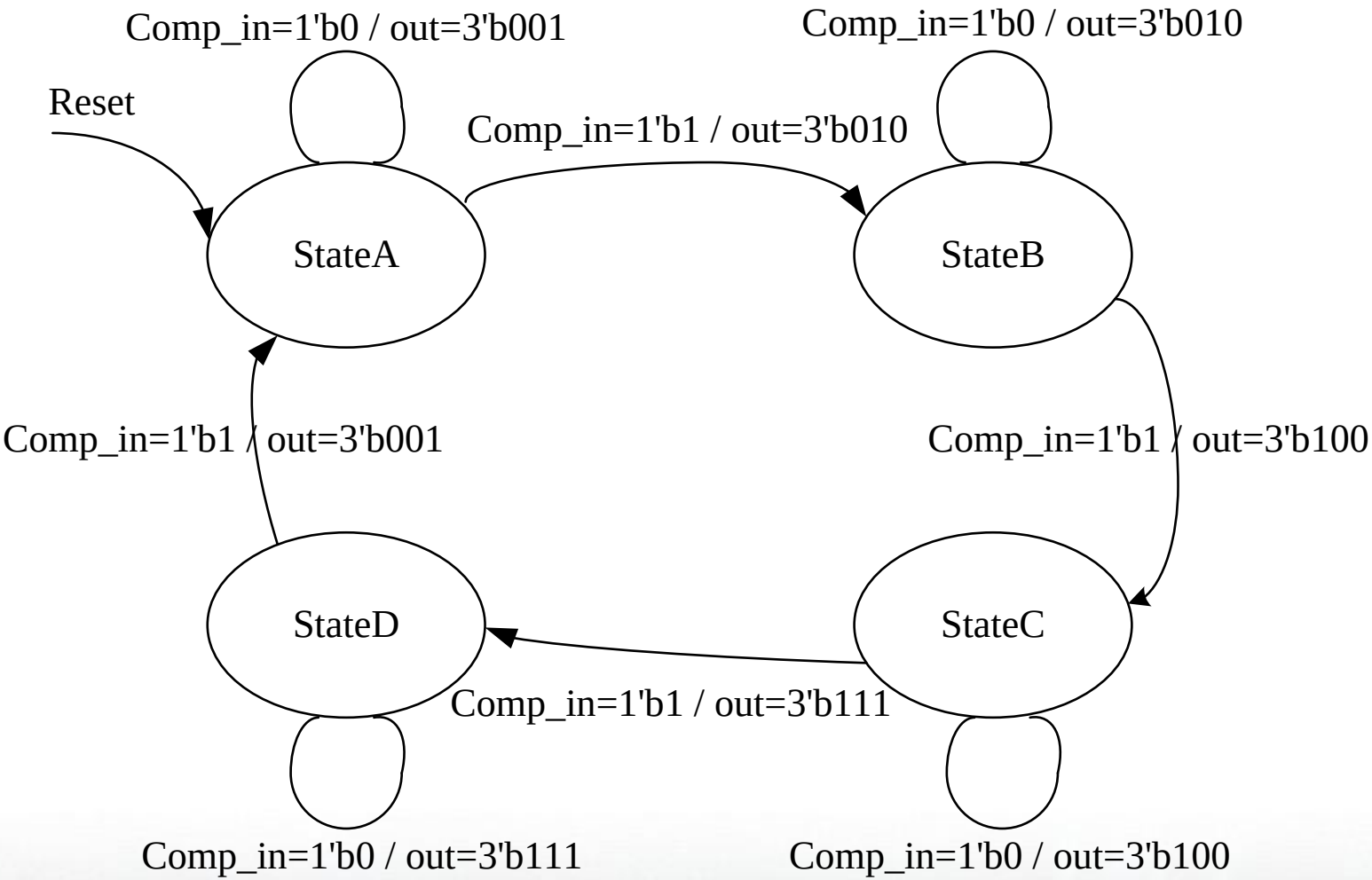
有限状态机


地点:宿舍
功能:睡觉
条件：周末
条件：非周末
地点 五号楼平台
功能:
条件：周末
地点:教室
功能:学习
条件：非周末
且下午7点
条件：周末
地点:食堂
功能:吃饭
条件：非周末

有限状态机

- **摩尔状态机：**摩尔状态机的输出仅仅依赖于当前状态，而与输入条件无关。

- **米勒状态机：**米勒型状态机的输出不仅依赖于当前状态，而且取决于该状态的输入条件。


Comp_in=1'b0 / out=3'b001
Comp_in=1'b0 / out=3'b010
Reset
Comp_in=1'b1 / out=3'b010
StateA
StateB
Comp_in=1'b1 / out=3'b001
Comp_in=1'b1 / out=3'b100
StateD
StateC
Comp_in=1'b1 / out=3'b111
Comp_in=1'b0 / out=3'b111
Comp_in=1'b0 / out=3'b100

两段式状态机

两段式状态机

```
module fsm2(out,comp_in,clk,clr);

input comp_in, clk, clr;

output[2:0] out;

reg[2:0] out;

reg[2:0] CS,NS;

parameter stateA = 3'b000, stateB = 3'b001,

          stateC = 3'b010, stateD = 3'b100;
```

两段式状态机

```
always @ (posedge clk or negedge clr)
begin if(!clr)
            CS <= stateA;
        else
            CS <= NS;
end
```

两段式状态机

```
always @(CS or comp_in)
begin NS = 3'bx;
        case(CS)
stateA: begin
      if(!comp_in) begin NS = stateA; out = 3'b001; end
        else begin NS = stateB;        out=3’b010;end end
stateB: begin
        if(!comp_in) begin NS = stateB; out = 3’b010; end
        else begin NS = stateC; out = 3’b100; end end
stateC: begin
        if(!comp_in) begin NS = stateC; out = 3'b100; end
        else begin NS = stateD; out=3’b111;end end
stateD: begin
        if(!comp_in) begin NS = stateD; out = 3'b111; end
        else begin NS = stateA; out=3’b001;end end
        endcase end
```

二段式状态机八股：

1. Module 名（出入口）；
2. input output
3. reg cs, ns, out;
4. parameter 状态；
5. 时序always cs<=ns;
6. 组合always case(cs, in) ns＝？ out = ?

知识点小结

- 连线：
- 寄存器：
- 寄存器组：
- 整数：
- 参数：
- 预处理：