```
always @(....)
  begin
    ......
    end
```

每一个 always block 都是一个独立电路. 一直都在 loop
常见的错误是将一个变量在两个 always block 中 赋值

```
reg y;
reg a,b, clear;
 alway @*
     if(clear) y=1'b0;


always @*
    y=a&b;
```

以上代码是不可综合, 为同时被两个电路更新, 这样的电路不存在

常见的错误还有一个就是 Incomplete branch and incomplete output assignment
举例一个下电路:

```
always @*
    if (a>b)
        gt =1'b1;
    else if (a==b)
        eq=1'b1;
```

错误: ① if else 不完整. 没有考虑 a<b 的情况, 当 a<b 时, gt 和 eq 应该等于多少, 电路没有. 那么 verilog 在这种情况下会使用 latch 来记忆输出等于 previous value 这就不好了
② a>b 时. eq=? a==b 时 gt=? 都是同情况
在 case 都是一样

3.7.2 Guidelines

The always block is a flexible and powerful language construct. However, it must be used with care to infer correct and efficient circuits and to avoid any discrepancy between synthesis and simulation. Following are the coding guidelines for the description of combinational circuits:

- Assign a variable only in a single always block.
- Use blocking statements for combinational circuits.
- Use `@*` to include all inputs automatically in the sensitivity list.
- Make sure that all branches of the if and case statements are included.
- Make sure that the outputs are assigned in all branches.
- One way to satisfy the two previous guidelines is to assign default values for outputs in the beginning of the always block.
- Describe the desired full case and parallel case in code rather than using software directives or attributes.
- Be aware of the type of routing network inferred by different control constructs.
- Think hardware, not C code.