Assign statements are good for for simple logic expressions.

For complex behavior, we need a more powerful means of expression.

The always block gives us a higher level of abstraction to do this.

Assign statements were evaluated in any order.

Within procedural blocks (like always), statements will be executed sequentially.

A sequential paradigm will allow description of more complex behavior.
always Block

- One way procedural blocks are formed is with the always block.
- Within the always block, statements are executed sequentially.
- This has nothing to do with the order in which the logic operates.
- External to the always block, everything inside appears to happen continuously.
- You can think of the always block as an infinite loop.
always Block

The always statement is structured like this (Verilog 2001):

```verilog
always @(sensitivity list) <begin> <procedural statements> <end>
```

An example (verilog 2001 style)...

```verilog
module ao2_gate(
    output reg y,
    input    a, b, c, d);

    reg tmp1, tmp2;

    always @(a,b,c,d) begin
        tmp1 = a & b;
        tmp2 = c & d;
        y = tmp1 | tmp2;
    end
endmodule
```
always Block

- Whenever a variable in the sensitivity list changes, the always block wakes up and executes its enclosed statements.
- If variables are omitted from the sensitivity list, the block will not wake up when you want it to!
- `<begin>,<end>` will be needed if multiple statements exist.
- Variables on the LHS inside the always block must be of type reg.
- Synthesis ignores the sensitivity list.

```verilog
reg tmp1, tmp2;
always @(a,b,c,d) begin
    tmp1 = a & b;
    tmp2 = c & d;
    y = tmp1 | tmp2;
end
```
always Block

- Reg variables retain only the last assigned values within always

```verilog
reg tmp1, tmp2;
always @(a,b,c,d,e,f)
  begin
    y = a & b;
    y = c & d;
    y = e & f;
  end
```

- y only retains the value computed by `y = e & f;` when always block concludes.
- When always block has multiple statements, `begin/end` pair is used
Two types of procedural assignment statements are used within always

- Blocking:
  - The blocking assignment operator is the equals sign \(=\)
  - So named because blocking assignments must evaluate RHS and complete the assignment without interruption from any other statement.
  - The blocking assignment also blocks other following assignments until the current one is done.
  - Blocking statements execute in the order they are specified.
always Block

Two types of procedural assignment statements

▶ Nonblocking
  ▶ The nonblocking assignment operator is the less-than-or-equals-to operator ”<=”
  ▶ The nonblocking assignment evaluates the RHS at the beginning of a time step and schedules the LHS update for the end of the time step.
  ▶ Between the evaluation of the RHS and update of LHS, other statements may execute.
  ▶ The nonblocking statement does not block any other statements from being evaluated.

▶ Nonblocking execution has two parts:
  ▶ 1. Evaluate RHS at beginning of time step
  ▶ 2. Update LHS at end of time step
always Block

Blocking and non-blocking assignments
Blocking Assignments Used

module fbosc1(
    output reg y1, y2;
    input clk, rst);

    //always blocks can execute in any order

    //"A" procedural block
    always @(posedge clk, posedge rst)
        if (rst) y1 = 0; //reset
        else y1 = y2;

    //"B" procedural block
    always @(posedge clk, posedge rst)
        if (rst) y2 = 1; //preset
        else y2 = y1;

endmodule
always Block

Blocking and non-blocking assignments
Nonblocking Assignments Used

module fbosc2(
    output reg y1, y2;
    input clk, rst);

//always blocks can execute in any order

//"A" procedural block
always @ (posedge clk, posedge rst)
    if (rst) y1 <= 0; //reset
    else       y1 <= y2;

//"B" procedural block
always @ (posedge clk, posedge rst)
    if (rst) y2 <= 1; //preset
    else       y2 <= y1;

endmodule
Coding Guidelines
Thanks to Cliff Cummings, Sunburst Design

- When modeling sequential logic, use nonblocking assignments.
- When modeling latches, use nonblocking assignments.
- When modeling combo logic with an always block, use blocking assignments.
- When modeling both sequential and combo logic within the same always block, use nonblocking assignments.
- Do not mix blocking and nonblocking assignments in the same always block.
- Do not make assignments to the same variable from more than one always block.

For more information, see: Nonblocking Assignments in Verilog Synthesis, Coding Styles That Kill!
Available at: http://www.sunburst-design.com