ALWAYS flavours for RTL

- RTL is synonymous with "synthesiable subset of verilog"
- reg can be changed only in one always.
- in RTL there is no "#" - no delays allowed.
- at most one clock can be specified in flipflop always.

full classic always of flipflops.

- all assignments must be "<="
- all registers will be made as flipflops.
- all variables driven in this always must be "reg".
- reset is asynchronous

```
always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
        somereg <= 0;
    end else begin
        if (enable)
            somereg <= (somereg *5) + 17;
    end
end
```

classic always of flipflops with sync reset

```
always @(posedge clk) begon
    if (!rst_n) begin
        somereg <= 0;
    end else begin
        if (enable)
            somereg <= (somereg *5) + 17;
    end
end
```

- similar to first one, just without rst_n in sensitivity list

full new form always of flipflops.

- all assignments must be "<="
- all registers will be made as flipflops.
- reset is asynchronous
- all affected variables (that are changed by this always) must be regs.

```
always_ff @(posedge clk or negedge rst_n) begon
    if (!rst_n) begin
        somereg <= 0;
    end else begin
        if (enable)
            somereg <= (somereg *5) + 17;
    end
end
```

- why this is needed? don't ask me.
- also ok without negedge of rst_n, making the reset synchronous.

classic combinatorial always

```
always @* begin
    aareg = inputx + 2;
    bbreg = aareg -1;
end
```

- bbreg will get updated aareg version. so this mode is similar to "C" code mechanics.
- all assignments are "=".
- all regs will become wires, **UNLESS!!!!** latch is created.
- notice that loops are not a good thing: No one should be assigned after used!

latch always

happens when register may NOT get a new value. think of **case** without **default** clause.

```
always @* begin
    if (enable)
        aareg = inputx + 2;
end
```

newform combinatorial always

```
always_comb begin
     aareg = inputx + 2;
end
```

I dont know what happens to latches here, but there is also always_latch construct defined in the new language.