# Behavioral Timing Controls

## Procedural timing controls

- Delay based timing control
- Event based timing control
- Level sensitive timing control

## Delay based timing control

- Regular delay control
- Intra assignment delay control
- Zero delay control

## Regular delay control

Format:

#<delay_value> <target> = RHS expression;

- It is used to delay the execution of the corresponding procedural statement by the defined delay value.
- Regular delays defer the execution of the entire assignment.

e.g. : #10 c = a + b;

**#(2:4:6,3:5:7) or** out = a | b;

## Intra-assignment delay control

Format:

<target> = #<delay_value> RHS expression;

- Intra-assignment delays compute the RHS expression at the current time and defer the assignment of the computed value to the LHS target by the defined delay value.

e.g. : c = **#10** a + b;

## Zero delay control

- Statements with zero delay execute at the end of the current simulation time, when all other statements in the current simulation time have executed.

```
initial
  begin
     a = 1’b1;
     b = 1’b0;
  end

initial
  begin
     #0 a = 1’b0;
     #0 b = 1’b1;
  end
```

## Event based timing control

- An event is the change in the value on a register or a net.
- Implicit event: The value changes on nets and registers can be used as events to trigger the execution of a statement.
- The event can also be based on the direction of the change.
  - **posedge** : change towards the value 1
  - **negedge** : change towards the value 0

## Types of event based timing control

- Regular event control
- Named event control
- Event or control

## Regular event control

Format:
@(event) statement;

```
e.g.:   @ (ena)    q = d;

        @ (posedge t_in)  q = ~q;

        q = @ (negedge clk) d;

        @ (posedge clk_in)  q_out = d_in;
```

## Named event control

- Events can be declared with an identifier.
- The declared can be triggered conditionally or unconditionally, & these doesn’t hold any data.
- The triggered events can be made to execute the a block of statements / assignments waiting for this trigger.
- Event is declared by keyword: **event** event_name;
- Event triggered as: **->** event_name;
- Usage: **@** <event_name>

## Example of named event control

```
module ex1(out, sig);
output out;
input sig;
reg out;
wire sig;
event shoot;
always @(negedge sig)
begin
if (sig == 1'b0)
-> shoot;
else
out = 1'b0;
end
```

```
always @(shoot)
//if (shoot == 1'b1)
out = 1'b1;
```

## Event **or** control

- The logical or of any number of events can be expressed such that the occurrence of any one of the events triggers the execution of the procedural statement that follows it.

Format:

always @ (event1 **or** event2 **or** event3)

always @ (**posedge** event1 **or negedge** event2)

# Event or control -Example

```
always @ (in1 or in0 or
   sel_in)
begin
  if (sel_in == 1'b0)
  mux_out = in0;
  else
  mux_out = in1;
end
```

```
always @ (posedge clk_in or
   negedge set_in_n)
begin
  if (set_in_n == 1'b0)
  q_out = 1'b1;
  else
  q_out = d_in;
end
```

## Level sensitive control

- Execution of statements will be delayed (wait) till a condition becomes true.
- The nature of the wait statement is level-sensitive, as opposed to basic event control (specified by the @ character), which is edge-sensitive.

Format:

**wait** (condition) <statement>

## Level sensitive control - Example

```
module lat(d_in, ena_in_n, q_out);
output q_out;
input d_in, ena_in_n;

reg q_out;
always @(d_in or ena_in_n)
begin
   wait (ena_in_n == 1’b0)
   q_out = d_in;
end
endmodule
```

## Reference

1. Samir Palnitkar,"Verilog HDL: A Guide to Digital Design and Synthesis" Prentice Hall, Second Edition,2003
2. T.R.Padmanabhan and B.Bala Tripura Sundari, "Design Through Verilog HDL" Wiley Student Edition