Centurion
UNIVERSITY
Shaping Lives...
Empowering Communities...

Clocking Blocks

- Used to specify timing of synchronous signals with respect to clock
- Mainly used in test benches and inside interfaces
- It ensures that all signals are driven or sampled with same clock delays
- Interfaces can contain multiple clocking blocks (1 per clock)

Clocking block syntax

```
clocking block_name clocking_event;
    item list;
endclocking : block_name

clocking bus @(posedge clock1);
  default input #10ns output #2ns;
  input data, ready,
          enable=top.mem1.enable;
  output negedge ack;
  input #1 addr;
endclocking
```

Centurion
UNIVERSITY

Clocking block -example

Example 5-13 Interface with a clocking block

```
interface arb_if(input bit clk);
  logic [1:0] grant, request;
  logic reset;

  clocking cb @(posedge clk);     // Declare cb
    output request;
    input grant;
  endclocking

  modport TEST (clocking cb,      // Use cb
                output reset);

  modport DUT (input request, reset, output grant);
endinterface
```

Centurion University
Shaping Lives...
Empowering Communities...

Input and output skews

Input and output signals are sampled at a clocking event.

For an input skew, the signal is sampled at skew units before the clock event.

For an output or inout, the signal is driven simulation time units after the corresponding clock.

Input and output skews

A skew value is a constant expression and can be specified by a parameter.

If a number is used for the time then skew is interpreted based on timescale.

An input skew of 1step indicates that the signal is sampled at the end of previous time step.

Clocking block example with skew and parameters

```
clocking cb @(posedge clk);
    parameter INPUT_SKEW = 1;
    parameter OUTPUT_SKEW = 2;
    default input #INPUT_SKEW output #OUTPUT_SKEW;
    input #3step req1;
    input req2;
    output #4ns grant;
endclocking
```

Program Blocks

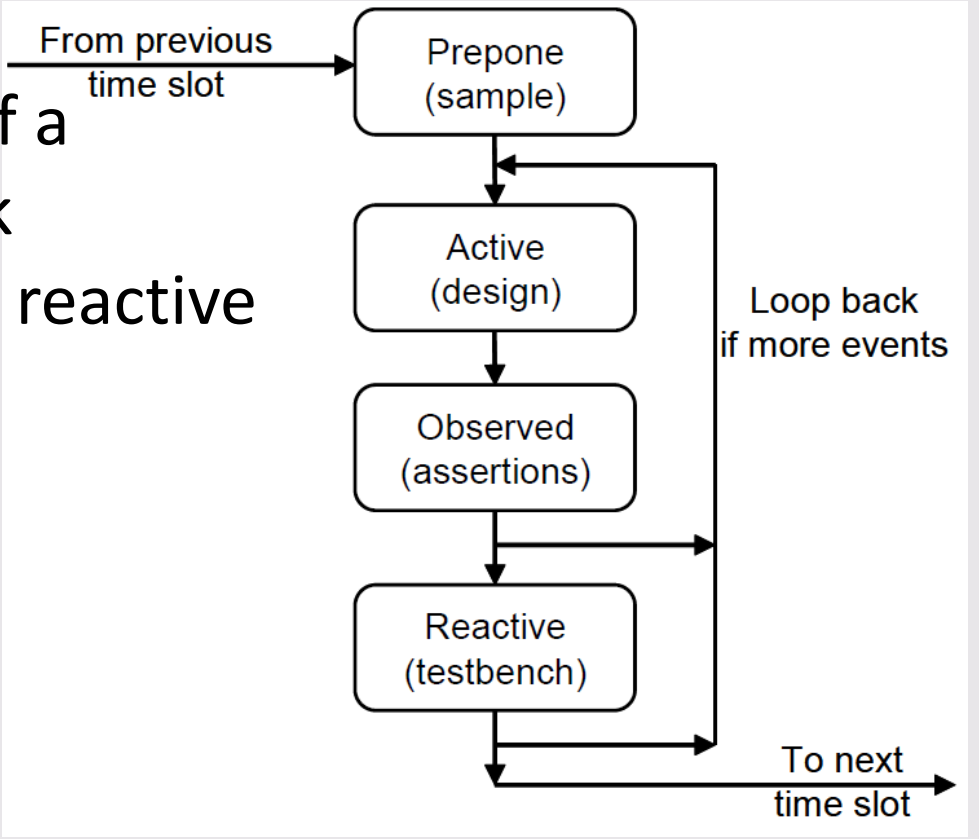
- Race conditions can occur if module is used as top level for both design and Test bench
- A program block is similar to a module. It is used for testbench code.

```
program
  helloWorld();
  initial
  begin: hello
    $display("Hello
  World");
  end

  initial
  begin: there
    $display(“Hello
  There”);
  end
endprogram:
  helloWorld
```

System Verilog Timeslot Evaluation

- Operations of a program block happen in the reactive region of the simulation.



Program Blocks

Centurion University
Shaping Lives... Empowering Communities...

- Programs can be instantiated inside modules, but not the other way around.
- Program blocks may contain one or more initial blocks, but may not contain always, UDPs, modules, interfaces, or other programs.
  - Usage is normally an initial block and use a forever loop for the whole test to evaluate

Program Block – Example

Example 5-15 Testbench using interface with clocking block

```
program automatic test (arb_if.TEST arbif);
...
  initial begin
    arbif.cb.request <= 2'b01;
    $display("@%0d: Drove req=01", $time);
    repeat (2) @arbif.cb;
    if (arbif.cb.grant != 2'b01)
      $display("@%0d: a1: grant != 2'b01", $time);
    end

endprogram : test
```