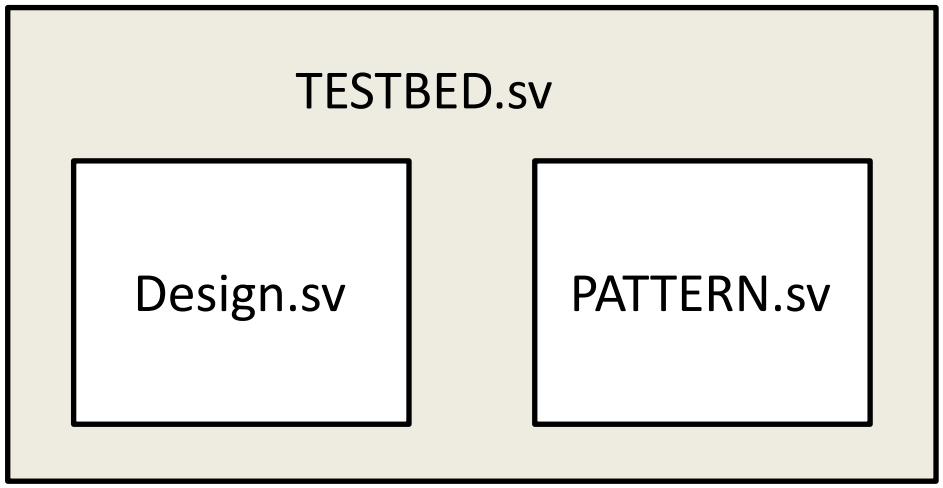
Pattern Hint

陳東軒

Pattern

- Generate clock
- Generate test stimulus and check answer



Reminders

- Add $finish; to end simulation.
  - End of pattern (Pass)
  - Spec violation (Fail)

If your simulation never stops…
Check for
①Proper $finish added or not
②Unbreakable loop in your pattern
③Unbreakable loop in design

- If display fail, give pattern some time before ending simulation.
  - Ex : $display….;
  - repeat(2)@(negedge clk); or repeat(2) #(CYCLE);
  - $finish;

- Generate input to design at negedge clock not posedge.
  - Ex : @(negedge clk);
    in_data = $urandom_range(3,0);

- In pattern, you should use "===" or "!==" instead of "==", "!=".
  - Need to consider unknown in pattern design

Hint

- Declare integers as you wish.
  - Ex : PATNUM, patcount, latency...
- Multiple initial blocks are allowed.
- Make good use of tasks.
- High level coding is allowed and recommended.
  - Ex : for loops, while loops, if without elses

Clock

- (Optional) Use `define to easily modify your numbers.
  - Declare before module

- Ex :

```
`define CYCLE_TIME 10

module pattern(…)
/* some pattern design */

real CYCLE =  `CYCLE_TIME; (Don’t forget “ ` ”)
always #(CYCLE/2.0) clk = ~clk;

/* some pattern design */
endmodule
```

Test Flow Hint

- General test flow and some useful tasks that are used a lot.
  - You can design your own flow and add your tasks.
- Ex：

```
initial begin
    reset_task;
    for(patcount=0;patcount<PAT_NUM;patcount=patcount+1)begin
        input_task;
        wait_out_valid_task;
        check_ans_task;
    end
    YOU_PASS_task;
end
```

Some Example Tasks

- Asynchronous reset :
  - Resets all signals at the falling edge of the reset signal.
- Clock should be forced to 0 before reset signal is given.

```
task reset_task; begin
    rst_n     = 'd1;
    in_valid  = 'd0;
    /*
        Signals you wish to reset.
    */
    force clk = 0;

    #CYCLE; rst_n = 'd0;
    #CYCLE; rst_n = 'd1;

    if( /* Check if reset done properly */ )begin
        $display ("/* Your error message */");
        repeat(2) #CYCLE;
        $finish;
    end

    #CYCLE; release clk;
end endtask
```

```
task wait_out_valid_task; begin
    latency = 0;
    while(out_valid===1'd0)begin
        latency = latency + 1;
        if( latency==/* Maximum cycles allowed */ )begin
            $display ("/* Your error message */");
            repeat(2)@(negedge clk);
            $finish;
        end
        @(negedge clk);
    end
    total_latency = total_latency + latency;
end endtask
```