Sample Midterm

Assertion Questions

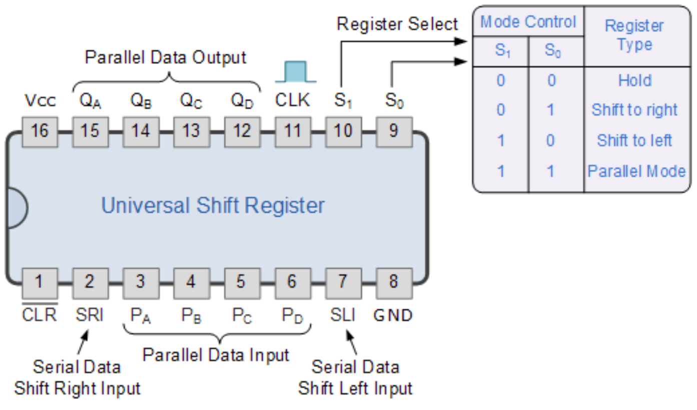
1. Check that an 8-bit counter overflows correctly
2. Mutex checker; Ensures that a and b never evaluate true at the same time.
3. The assert_unchange assertion continuously monitors the start_event at every positive edge of the triggering event, clk. When this signal (or expression) evaluates TRUE, the assertion monitor ensures that the test_expr will not change values within the next num_cks number of clocks.
4. The assert_next assertion validates proper cycle timing relationships between two events in the design. When a start_event evaluates true,then the test_expr must evaluate true exactly num_ cks number of clock cycles later.

1. assert property(@(posedge clk) counter==8'hff |=>counter==8'h00)
2. assert property(@(posedge clk) a!=1'b0 || b!=1'b0)

SystemVerilog General Questions

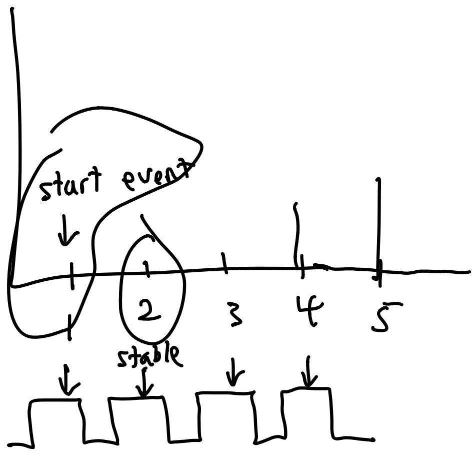
1. What does an interface do ? — interface is named bundle of nets or variables. Encapsulates functionality & connectivity
2. What is a clocking block? — timing for sampling and driving clocking block is implicit and relative to CB's clock — clocking blocks separates timing and synchronization details from procedural statements Interaction between TB and DUT must be handled correctly or non-deterministic results occur
3. Write the interface/clocking block/modport for testing a 4-bit universal shift register

4-bit Universal Shift Register 74LS194



Mode Control		Register Type
S_1	S_0	
0	0	Hold
0	1	Shift to right
1	0	Shift to left
1	1	Parallel Mode

num_cks = 3



Constraint Questions

4. Write the constraint block for the Universal Shift register
5. Write the assertions for the same

assuming that start_event is the first clk cycle

3. assert property(@(posedge clk) start_event==1'b1 |=> $stable(expr)[* num_cks]
4. assert property(@(posedge clk) start_event |=> ##[num_cks] test_expr

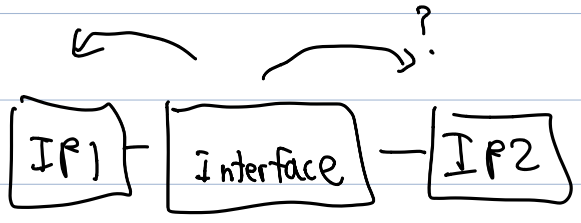
SystemVerilog

Interface

Clking Blks

Modport

Assertions



1. Overflow correctly?

8 bit counter

255 → 0

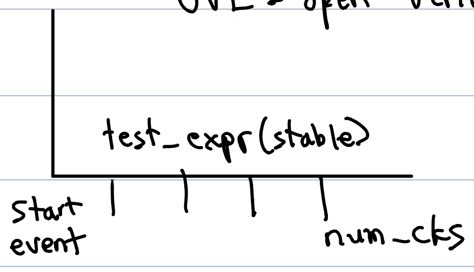
```
assert
property(@(posedge clk)
(counter == 8'hff |=> counter = 8'h00))
```

2. Mutex checker

```
!(a==1'b1 && b== 1'b1)
```

3. start event → test_expr does not change for num cks

OVL ⇒ open verification language (assertion checker library)



```
start_event |=> $stable(test_expr[*(num_cks-1)])
```

$past
$rose
$fell
$stable

4. start_event |=> ##[num_cks] test_expr

Sys Verilog General Questions

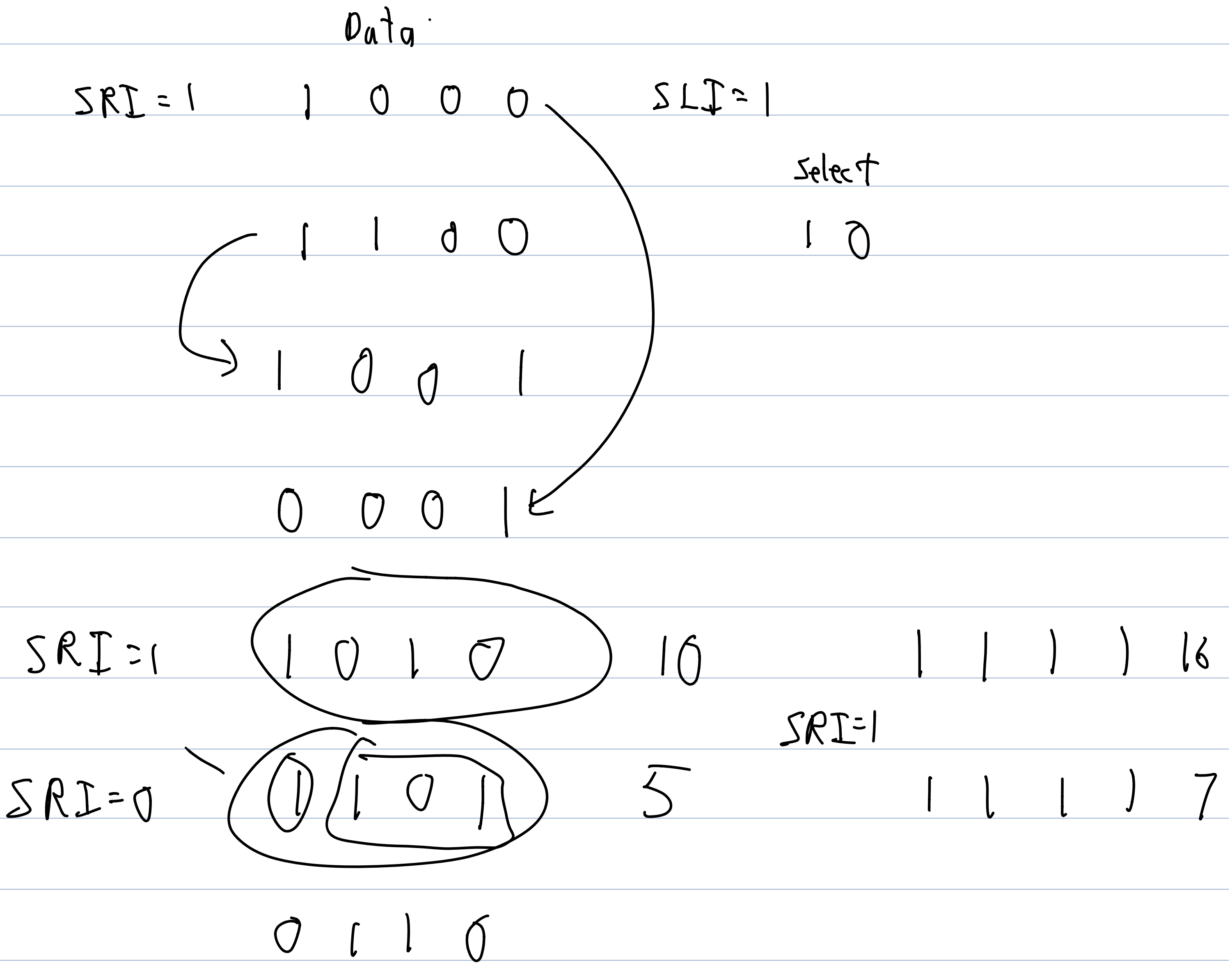
1. Interface

2. Clking block

Universal Shift register

randomize mode

Either, shift left, right, or parallel load



```
assert property(@(posedge clk) (S1==0 && S0==1) |=>
                                $past(Q[3:1]) == Q[2:0] &&
                                Q[3] = $past(SRI));
```