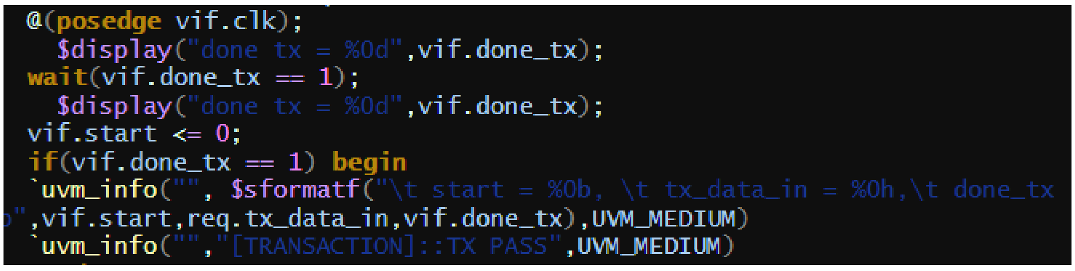
UART Bugs and Fixes

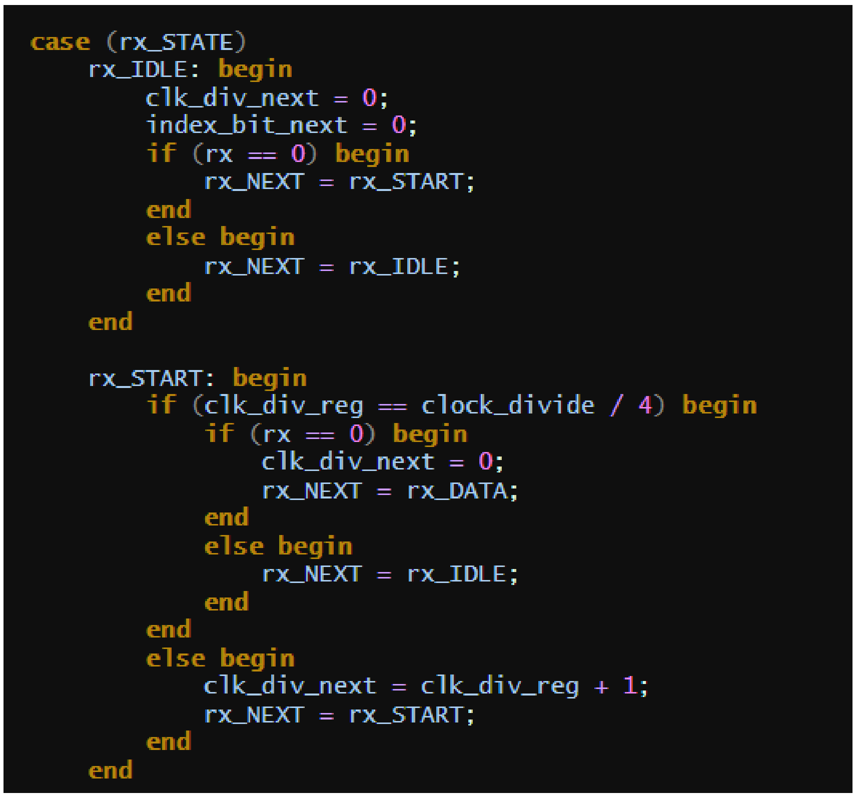
1. TX Start Condition Issue

In the driver code, the transaction should begin when done_tx is high (i.e., wait(vif.done_tx == 1)). However, the DUT outputs done_tx = 0, preventing the transaction from starting.



```
  @(posedge vif.clk);
    $display("done tx = %0d",vif.done_tx);
  wait(vif.done_tx == 1);
    $display("done tx = %0d",vif.done_tx);
  vif.start <= 0;
  if(vif.done_tx == 1) begin
  `uvm_info("", $sformatf("\t start = %0b, \t tx_data_in = %0h,\t done_tx
",vif.start,req.tx_data_in,vif.done_tx),UVM_MEDIUM)
  `uvm_info("","[TRANSACTION]::TX PASS",UVM_MEDIUM)
```

Upon reviewing the uart_tx module, it was found that in the state machine implementation, the next state is not specified within the if conditions for each state. Only the else blocks assign the next state. This causes undefined behavior when the if conditions are true. To fix this, the next state should explicitly remain the current state when inside the if condition.



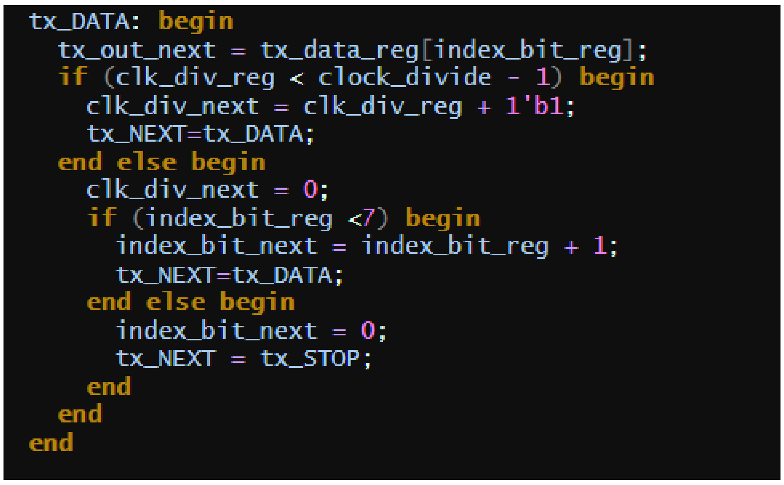
```
case (rx_STATE)
    rx_IDLE: begin
        clk_div_next = 0;
        index_bit_next = 0;
        if (rx == 0) begin
            rx_NEXT = rx_START;
        end
        else begin
            rx_NEXT = rx_IDLE;
        end
    end

    rx_START: begin
        if (clk_div_reg == clock_divide / 4) begin
            if (rx == 0) begin
                clk_div_next = 0;
                rx_NEXT = rx_DATA;
            end
            else begin
                rx_NEXT = rx_IDLE;
            end
        end
        else begin
            clk_div_next = clk_div_reg + 1;
            rx_NEXT = rx_START;
        end
    end
```

---

2. Infinite Loop in tx_DATA State

In the tx_DATA state, the index_bit_reg (a 3-bit register) is used to track the bit position in tx_data_reg. The loop condition checks if index_bit_reg < 8. Since a 3-bit register can range from 0 to 7, this condition is always true, causing the loop to never exit and resulting in an infinite loop.

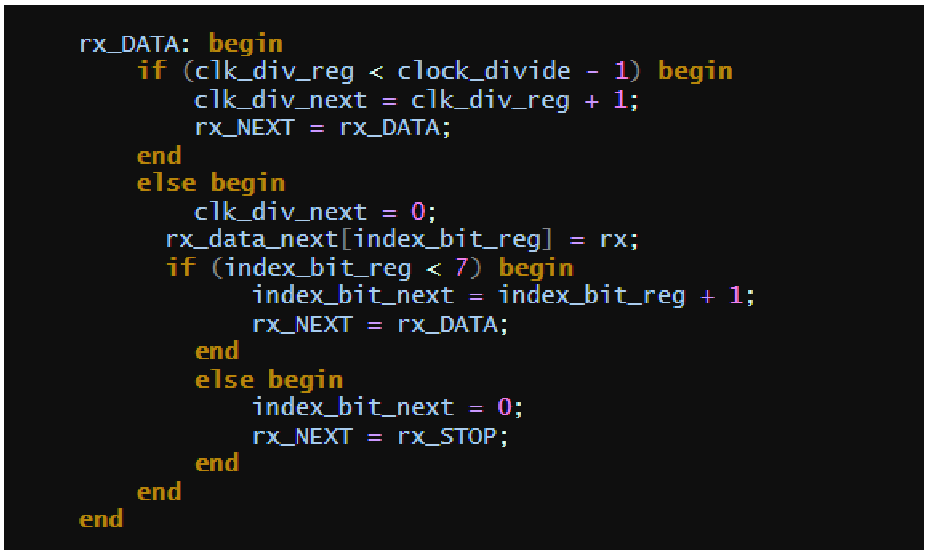
To resolve this, the condition should be modified to index_bit_reg < 7, allowing the loop to eventually exit. This change lets the state machine reach the tx_DONE state, where done_tx is asserted, enabling the TX process to complete successfully.



```
tx_DATA: begin
  tx_out_next = tx_data_reg[index_bit_reg];
  if (clk_div_reg < clock_divide - 1) begin
    clk_div_next = clk_div_reg + 1'b1;
    tx_NEXT=tx_DATA;
  end else begin
    clk_div_next = 0;
    if (index_bit_reg <7) begin
      index_bit_next = index_bit_reg + 1;
      tx_NEXT=tx_DATA;
    end else begin
      index_bit_next = 0;
      tx_NEXT = tx_STOP;
    end
  end
end
```

---

3. Similar Issue in RX
The similar issue exists in the uart_rx module, where index_bit_reg <= 7 causes a never-ending loop. Changing this condition to index_bit_reg < 7 allows the state machine to transition to the next state and complete the reception properly.



```
rx_DATA: begin
    if (clk_div_reg < clock_divide - 1) begin
        clk_div_next = clk_div_reg + 1;
        rx_NEXT = rx_DATA;
    end
    else begin
        clk_div_next = 0;
      rx_data_next[index_bit_reg] = rx;
      if (index_bit_reg < 7) begin
            index_bit_next = index_bit_reg + 1;
            rx_NEXT = rx_DATA;
        end
        else begin
            index_bit_next = 0;
            rx_NEXT = rx_STOP;
        end
    end
end
```