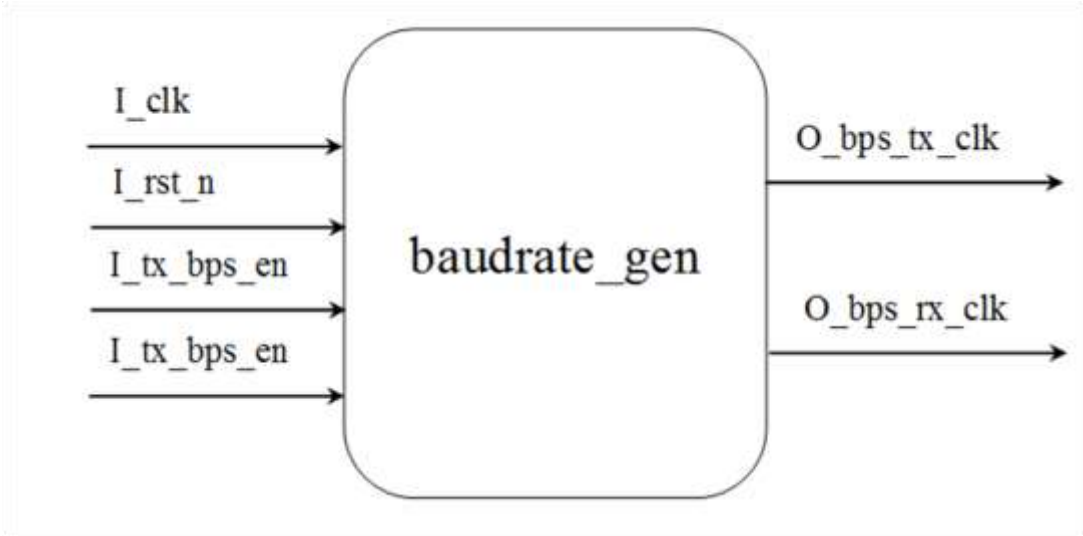
UART Note

Design Flow

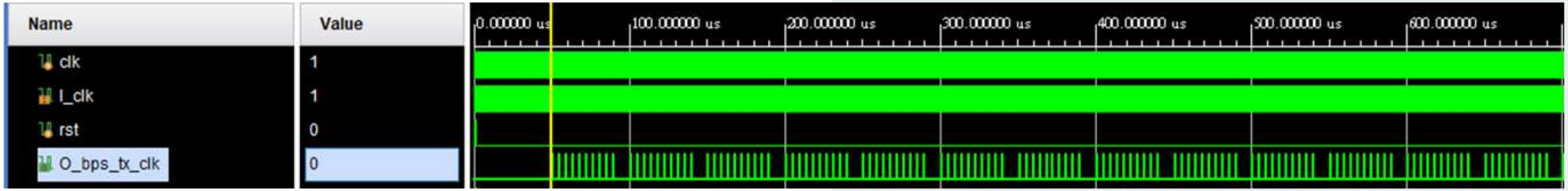
- 1,編寫傳送的verilog代碼，並往PC上不斷發送0x00~0xFF **的數據**， PC上的串口調適接收號 以16HEX 表現。
- 2,編寫接收模塊，用收到的數據點亮LED。
- 3,最終目標，將接收模塊以及發送模塊包再一起，然後從PC的串口發收數據到FPGA，FPGA接收到數據在PC上顯示。

Baud rate selection moudule



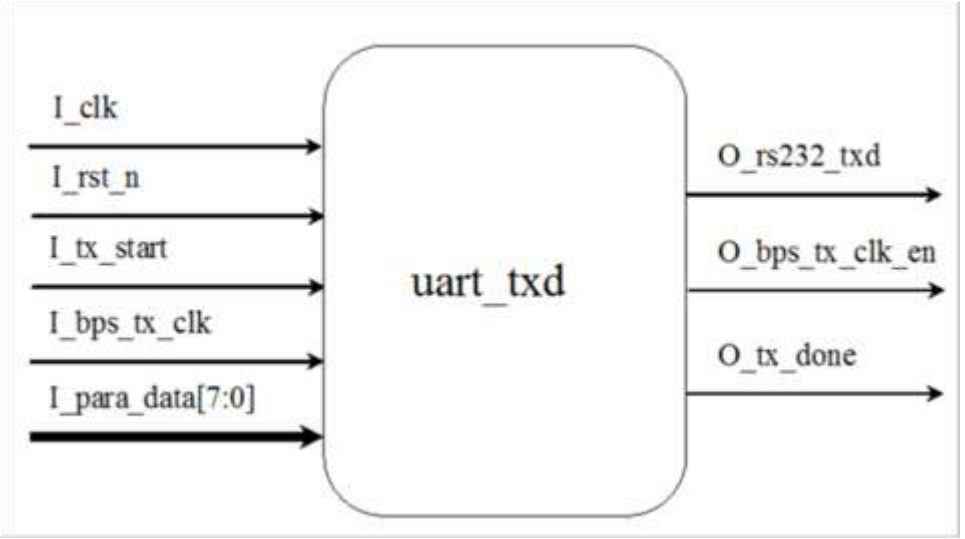

```
always@(posedge I_clk)
begin
    if(I_rst)
        R_bps_tx_cnt <= 13'd0;
    else if(I_bps_tx_clk_en == 1'b1)
        begin
            if(R_bps_tx_cnt == C_bps_select)
                R_bps_tx_cnt <= 13'd0;
            else
                R_bps_tx_cnt <= R_bps_tx_cnt + 1'b1;
        end
    else
         R_bps_tx_cnt <= 13'd0;
end

assign O_bps_tx_clk = (R_bps_tx_cnt ==13'd1)?1'b1:1'b0;
```

除頻器的概念，計數完，拉H，其餘時候都是L

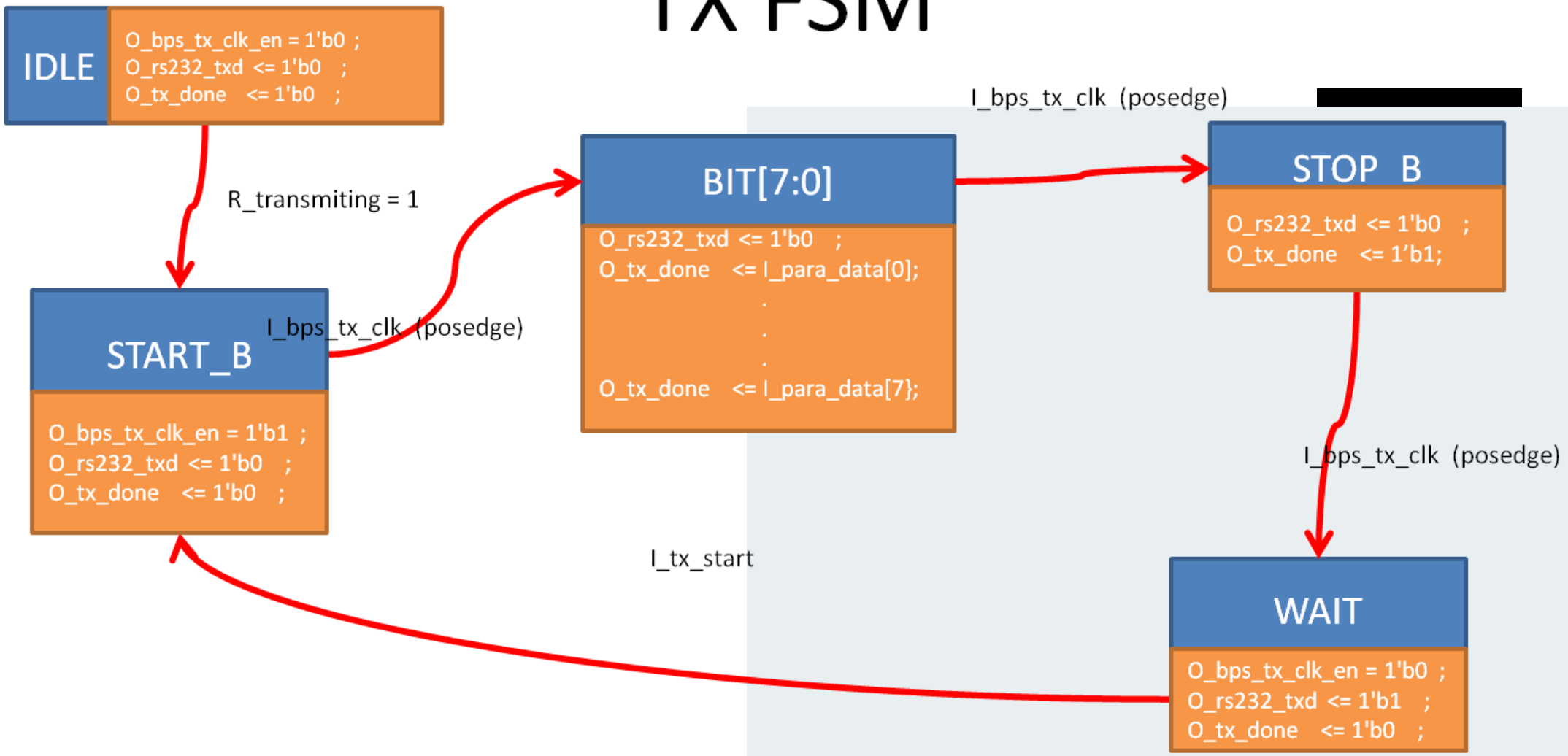


看到BAUD RATE的CLK ，程式每10個CLK，一個CYCLE，所以示波器上面有間隔，因為有EN PIN，
所以每8bit 的資料的CLK，包含START BIT 跟STOP BIT所以總共有10 BIT，用到10 個CLK。

Tx moudule


I_clk
I_rst_n
I_tx_start
I_bps_tx_clk
I_para_data[7:0]
uart_txd
O_rs232_txd
O_bps_tx_clk_en
O_tx_done

TX FSM



IDLE 初始資料
R_transmiting 為H，進入到START_B
START_B 將BPS_CLK_EN打開BAUD 的CLK，在下個BAUD的CLK上升的時候開始傳遞資料。
BIT傳8BIT的資料，傳到第7BIT進入STOP_B
STOP_B O_TX_DONE 拉H，在下個CLK近來時候進入WAIT 等到下一筆要傳書的資料，並且O_RS232_TXD傳1告知已進入WAIT將關閉CLK，等到I_TX_STRATFLAG近來在傳下筆資料。

```
reg R_transmiting = 1'b0;
// logic for flag transmiting
always@(posedge I_clk)
begin
    if(I_rst)
        R_transmiting <= 1'b0;
    else if(O_tx_done)
        R_transmiting <=1'b0;
    else if(I_tx_start)
        R_transmiting <= 1'b1;
end
```

當I_tx_start 近來，要重送資料的時候，R_tramsmiting 為開始傳送資料的FLAG

```
// FSM state logic

always@(posedge I_clk)
begin
    if(I_rst)
        curr_state = IDLE;
    else
        curr_state = next_state;
end
```

將FSM分成三個部分進行撰寫，第一個針對狀態的邏輯。

```
//FSM next state logic

always@(*)
begin
    case(curr_state)
        IDLE:
            begin
                if(R_transmiting)
                begin
                    next_state = START_B;
                end
                else
                    next_state = IDLE;
            end
        START_B :
            begin
                if(I_bps_tx_clk)

                    next_state = B_0;
                else
                    next_state = curr_state;
            end
        B_0     :
            begin
                if(I_bps_tx_clk)
                    next_state = B_1;
                else
                    next_state = curr_state;
            end
        B_1     :
            begin
```

第二個部分，為針對進入下個狀態條件邏輯

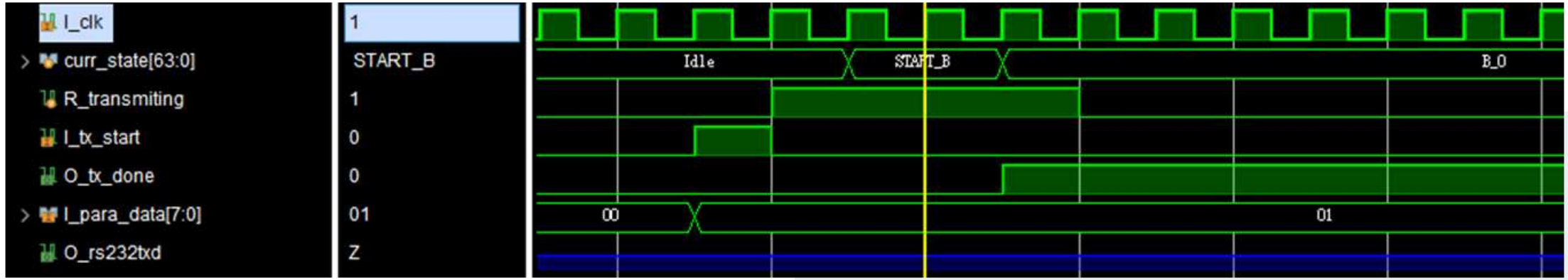
```
// output logic
always@(*)
begin
    case(curr_state)
        IDLE:
            begin
                O_bps_tx_clk_en = 1'b0  ;
                O_rs232_txd  <= 1'b0    ;
                O_tx_done    <= 1'b0    ;
            end
        START_B :
            begin
                O_bps_tx_clk_en = 1'b1  ;
                O_rs232_txd  <= 1'b0    ;
                O_tx_done    <= 1'b0    ;
            end
        B_0     :
            begin
                O_rs232_txd  <= 1'b0    ;
                O_tx_done    <= I_para_data[0];
            end
        B_1     :
            begin
                O_rs232_txd  <= 1'b0    ;
                O_tx_done    <= I_para_data[1];
            end
        B_2     :
            begin
                O_rs232_txd  <= 1'b0    ;
                O_tx_done    <= I_para_data[2];
            end
        B_3     :
```

第三個部分，為每個狀態的OUTPUT 的邏輯。

```
always@(posedge I_clk)
begin
    if(I_rst)
        begin
            R_cnt_1s <= 31'd0;
            R_data_reg = 7'd0;
            R_tx_start_reg =1'd0;
        end
    else if(R_cnt_1s == 31'd5000)
        begin
             R_cnt_1s         <= 31'd0                ;
             R_data_reg       <= R_data_reg + 1'b1     ;
             R_tx_start_reg <= 1'b1                 ;
        end
    else
        begin
          R_cnt_1s             <= R_cnt_1s + 1'b1     ;
          R_tx_start_reg     <= 1'b0              ;
        end
end
```

資料傳輸的TOP 邏輯，作連續資料的傳輸，每5000次 +1，並且打開TX的EN PIN，進行傳輸，防止資料傳輸錯誤。

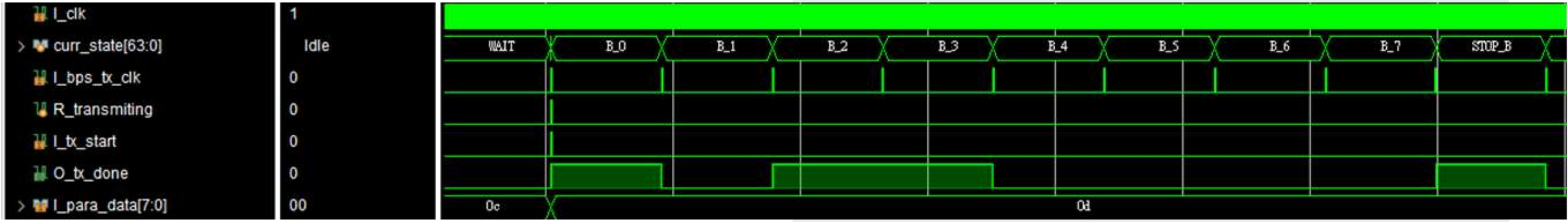
開始傳輸資料



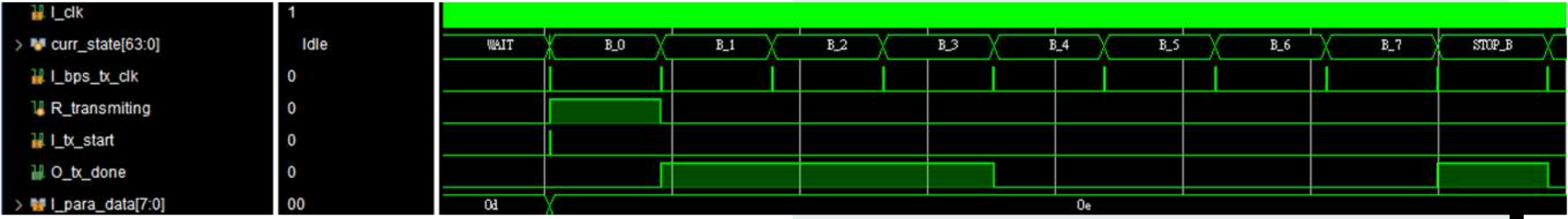
傳輸資料 01 => TX_DONE 0001

傳輸資料 02 => TX_DONE 0010

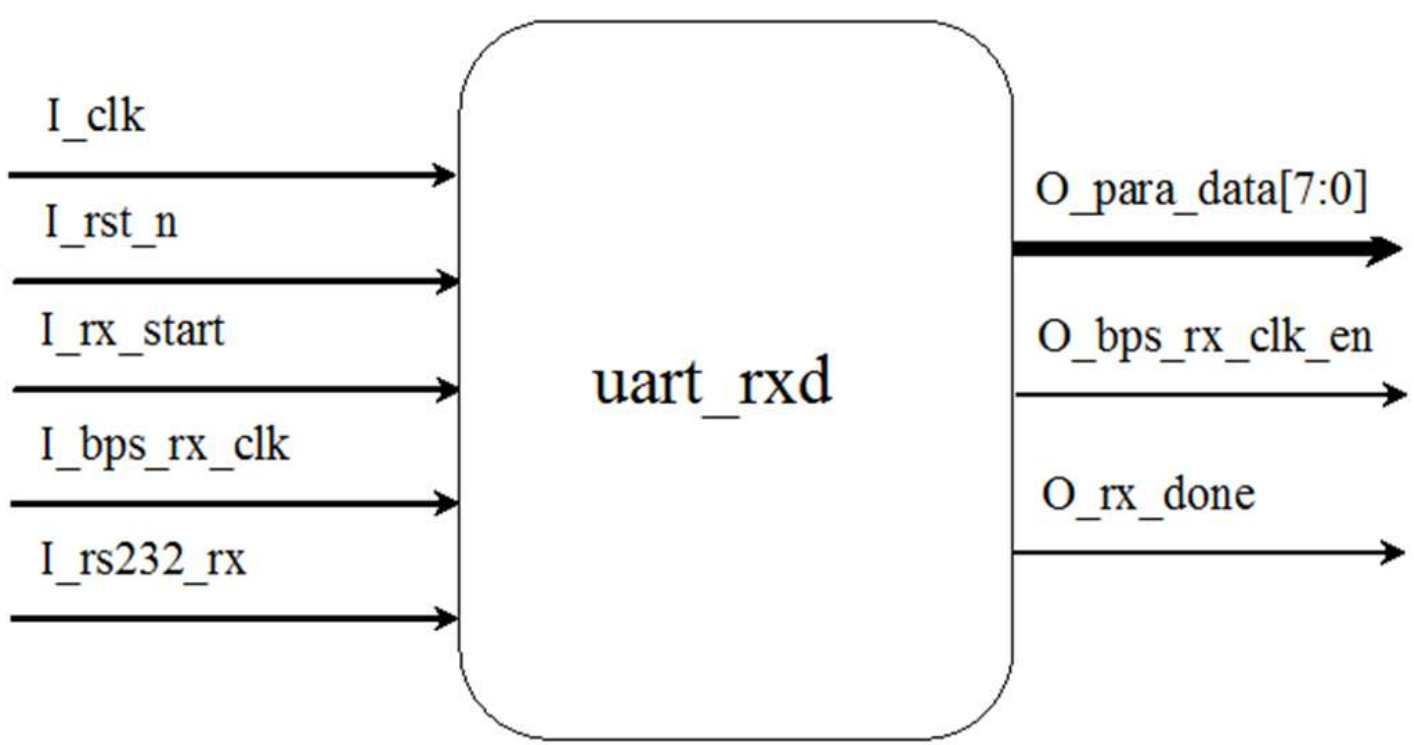
傳輸資料0d=> TX_DONE 1101



傳輸資料0e => TX_DONE 1110

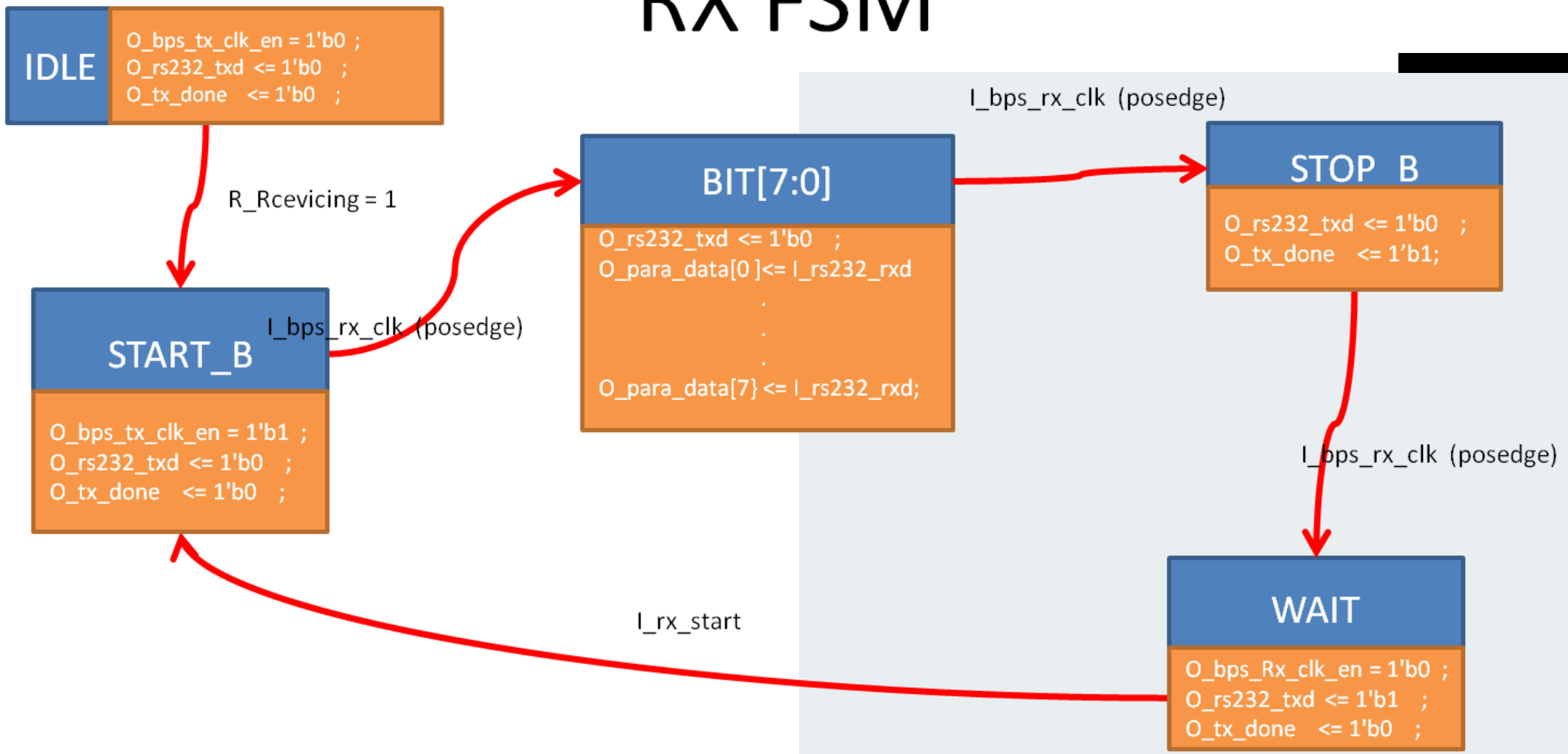


RX moudule



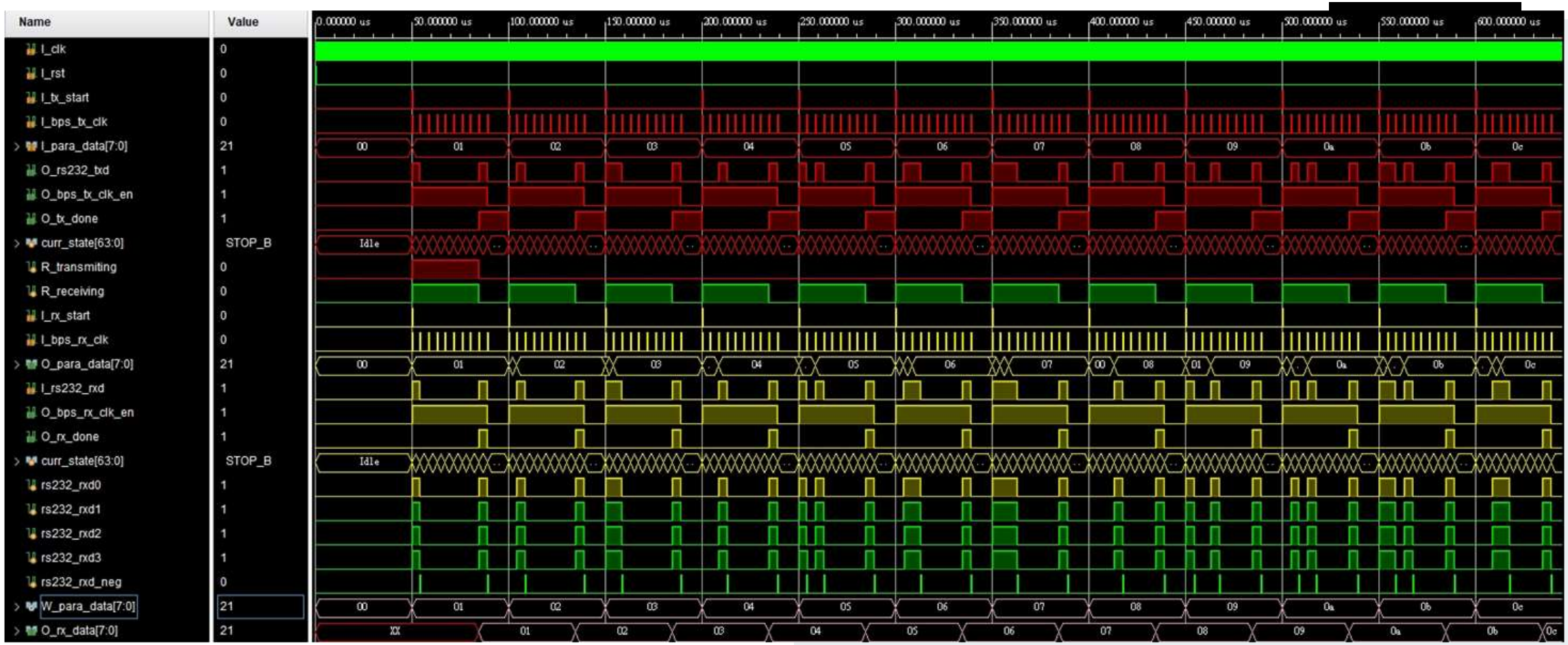
I_clk	system clocks
I_rst_n	system reset.
I_rx_start	beginning single, when the high, send the data I_para_data.
I_bps_rx_clk	baud rate clk, send the one bit when I_bps_tx_clk was high
I_para_data[7:0]	8-bit data
I_rs232_rxd	parallel data.
O_bps_clk_en	enable pin of output of buad rate
O_rx_done	when the one byte sending done will send the high pluse.

RX FSM



IDLE 初始資料
R_Rcevicing 為H，進入到START_B
START_B 將BPS_CLK_EN打開BAUD 的CLK，在下個BAUD的CLK上升的時候開始傳遞資料。
BIT傳8BIT的資料，傳到第7BIT進入STOP_B
STOP_B O_TX_DONE 拉H，在下個CLK近來時候進入WAIT 等到下一筆要傳書的資料，並且O_RS232_TXD傳1告知已進入WAIT將關閉CLK，等到I_RX_STRAT FLAG近來在傳下筆資料。

Simulation UART



透過W_para_data 確認TX接收到的資料，透過O_rx_data確認RX最終收到的資料，如下圖的粉紅色標記的波型。

Demo issue

- 1,
- 亞穩態的問題處理，目前傳輸資料應該是沒有問題，115200傳輸 1bit **時間** 8.68us，包含START bit 跟 STOP BIT 總共有10bit，所以傳送一個byte要86.8us，傳送255個byte **需要**22.185us，模擬上是22.134us，目前沒甚麼問題。
- 2,
- 實際機台燒錄未做，無法測試串口是否符合要求。