DIGITAL DESIGN WITH FPGA CAMP

DAY 3 SYSTEM DESIGN BY TX UART

COURSE REVIEW

- การใช้ภาษา VHDL เพื่อออกแบบ logic gate ต่าง ๆ เช่น Counter Multiplexer Shift register แบบ synchronous
- การวาด timing diagram จาก code ที่เขียนไว้
- การออกแบบวงจรพื้นฐานจาก timing diagram ที่กำหนดไว้
- การทดสอบระบบที่ได้ออกแบบ โดยใช้ simulation

AGENDA

- การออกแบบวงจรที่ซับซ้อนขึ้น โดยโจทย์ระบุเพียง input/output และ timing diagram ที่ต้องการ
- การทดสอบวงจรที่ออกแบบ โดยใช้ simulation -> LAB
- การทดสอบวงจรที่ออกแบบ ลงบนบอร์ดจริง -> LAB
- การวิเคราะห์และแก้ไขปัญหาที่เกิดขึ้นบนบอร์ดจริง -> วันต่อไป

ขั้นตอนการออกแบบวงจร

การออกแบบวงจรที่ซับซ้อนขึ้น โดยโจทย์ระบุเพียง input/output และ timing diagram ที่ต้องการ

- วิเคราะห์โจทย์ที่ได้มา เพื่อออกแบบว่าควรมี signal และ logic พื้นฐานที่ต้องใช้อะไรบ้าง
- วาด timing diagram ของ signal ที่ต้องการไว้
- ออกแบบวงจรจาก timing diagram ที่กำหนดไว้โดยใช้ภาษา VHDL

วิเคราะห์โจทย์

โดยทั่วไป จะแบ่งโจทย์ออกเป็นส่วนย่อย ๆ คือ

1) **ส่วนที่เป็นตัวควบคุมหลัก** มีลำดับขั้นตอนการทำงานที่แน่นอน (Sequential) จะออกแบบโดยใช้ State machine

2) **วงจรย่อย ๆ** อื่น ๆ ที่อาจจะทำงานตลอดเวลา หรือทำงานเพียงบางช่วง ซึ่งทำงานพร้อม ๆ กันหลาย ๆ ตัว (Concurrent) จะออกแบบโดยใช้ logic พื้นฐานต่าง ๆ ที่รู้จักกัน เช่น

- Counter สำหรับนับจำนวนข้อมูลที่ทำงาน หรือเป็น delay counter ตามช่วงเวลาที่กำหนด
- Shift register เพื่อแปลงข้อมูลจาก Parallel เป็น Serial หรือกลับกันคือ แปลงจาก Serial เป็น Parallel
- Set/Clear Flip Flop เพื่อสร้างจังหวะการเริ่ม/จบการทำงานของสัญญาณอื่น ๆ เช่น สร้างสัญญาณ enable ให้ counter นับ หรือสร้างสัญญาณ enable ให้ shift register เลื่อนข้อมูล
- D Flip Flop เพื่อ delay สัญญาณไปทีละ clock เพื่อรอจังหวะการทำงานให้พร้อมกับสัญญาณอื่น ๆ

STATE MACHINE

STATE MACHINE

- STATE สามารถเข้ารหัสได้ 2 แบบ คือ binary หรือ one-hot
- หากมีจำนวน State ไม่มาก และสัญญาณที่ใช้ในการเปลี่ยน State ไม่ซับซ้อน ควรใช้ binary coding
- หาก State มีจำนวนมาก หรือสัญญาณที่ใช้ในการเปลี่ยน State นั้นมีจำนวนมาก ควรใช้ one-hot coding

State by Binary Coding (N-bit = 2^n State)

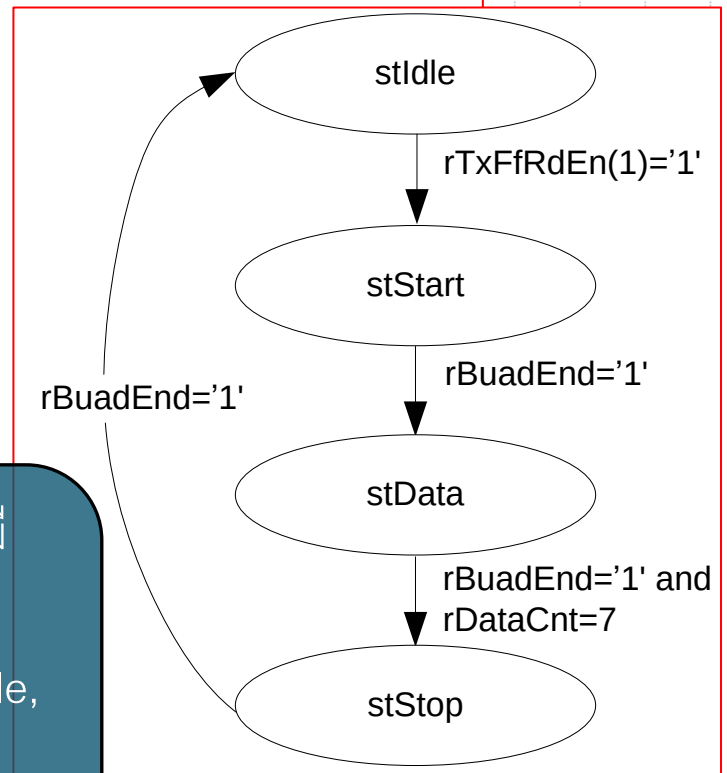
State	Value
State0	00b
State1	01b
State2	10b
State3	11b

State by One-hot Coding (N-bit = N State)

State	Value
State0	0001b
State1	0010b
State2	0100b
State3	1000b

STATE MACHINE (BINARY CODING)

```
type    StateType is
                (
                    stIdle,
                    stStart,
                    stData,
                    stStop
                );
signal  rState      : StateType;
```




```
u_rState : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rState  <= stIdle;
        else
            case ( rState ) is
                when stIdle     =>
                    if ( rTxFfRdEn(1)='1' ) then
                        rState  <= stStart;
                    else
                        rState  <= stIdle;
                    end if;

                when stStart    =>
                    if ( rBuadEnd='1' ) then
                        rState  <= stData;
                    else
                        rState  <= stStart;
                    end if;

                when stData     =>
                    if ( rBuadEnd='1' and rDataCnt="111" ) then
                        rState  <= stStop;
                    else
                        rState  <= stData;
                    end if;

                when stStop     =>
                    if ( rBuadEnd='1' ) then
                        rState  <= stIdle;
                    else
                        rState  <= stStop;
                    end if;
            end case;
        end if;
    end if;
End Process u_rState;
```

มีเพียง 4 State และเงื่อนไขการเปลี่ยน State มีจำนวนไม่มาก การเข้ารหัสแบบ binary คือใช้สัญญาณขนาด 2-bit แทน 4 State (“00”: stIdle, “01”: stStart, “10”: stData, “11”: stStop) จะเหมาะสมกับการทำงาน

STATE MACHINE (ONE-HOT)

1

```
type     StateType is
                  (
                      stIdle,
                      stWaitFf,
                      stGenReq,
                      stWtRdFf,
                      stTrans
                  );
signal   rState   : StateType;
```

2

```
u_rState : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rState      <= stIdle;
        else
            case ( rState ) is
                -- Wait start pulse
                when stIdle     =>
                    -- Receive request from user
                    if ( PattReq='1' ) then
                        rState  <= stWaitFf;
                    else
                        rState  <= stIdle;
                    end if;

                -- Wait until FIFO is ready
                when stWaitFf   =>
                    -- Ddr is not busy
                    -- Free space is more than 32 words
                    if ( (rCmd='0' and MtDdrWrBusy='0'
                          and WrFfWrCnt(15 downto 6)/=("11"&x"FF"))
                    -- Ddr is not busy
                     or (rCmd='1' and MtDdrRdBusy='0') ) then
                        rState  <= stGenReq;
                    else
                        rState  <= stWaitFf;
                    end if;
```

3

```
when stGenReq   =>
    -- Write Command is received
    if ( rCmd='0' and MtDdrWrBusy='1' ) then
        rState  <= stTrans;
    -- Read Command is received
    elsif ( rCmd='1' and MtDdrRdBusy='1' ) then
        rState  <= stWtRdFf;
    else
        rState  <= stGenReq;
    end if;

-- Wait 32 words available
when stWtRdFf   =>
    if ( RdFfRdCnt(15 downto 5)/=0 ) then
        rState  <= stTrans;
    else
        rState  <= stWtRdFf;
    end if;

-- Transfer data to/from DDR
when stTrans    =>
    -- End of burst transfer
    if ( rBurstCnt(4 downto 0)=31 ) then
        -- End transfer when next address is end address
        if ( rMtDdrAddr(28 downto 7)=cEndPattReq(21 downto 0) ) then
            rState  <= stIdle;
        -- Continue for next burst transfer
        else
            rState  <= stWaitFf;
        end if;
    -- Still not complete current burst transfer
    else
        rState  <= stTrans;
    end if;
end case;
```

มี 5 State แต่เงื่อนไขการเปลี่ยน State มีสัญญาณจำนวนมาก การเข้ารหัสแบบ one-hot จะทำให้ทำงานได้มีประสิทธิภาพมากกว่า เพราะถ้าใช้ binary จะใช้เวลาประมวลผลนาน

TX UART OVERVIEW

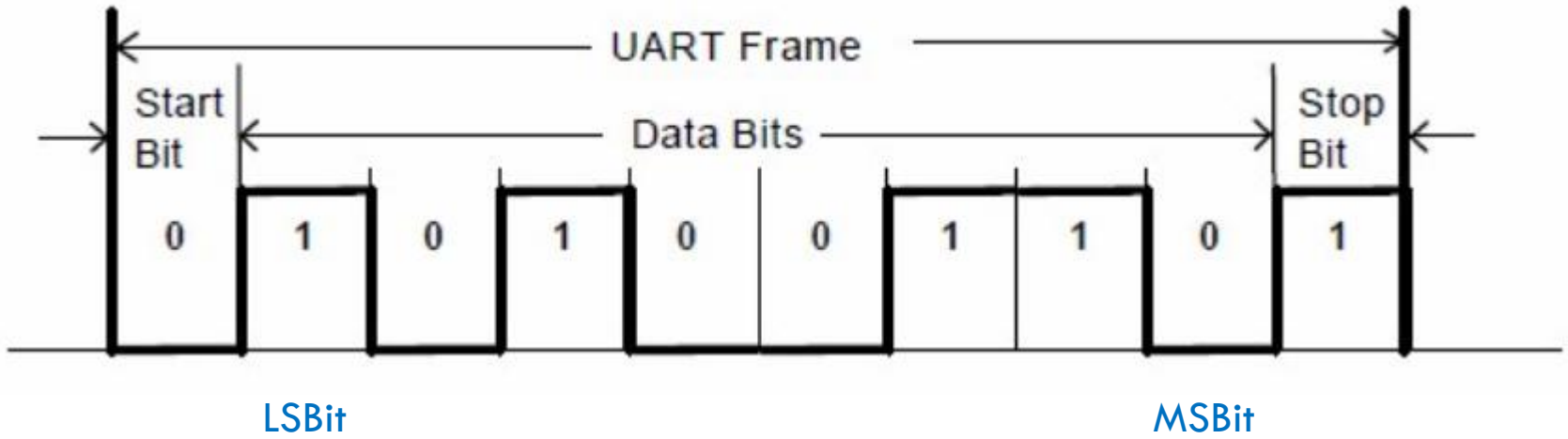
UART

- Universal Asynchronous Receiver and Transmitter
- It is a serial data protocol.
- It can send data without using clk. How?
- Transmitter and receiver must agree on data format and transmission speed
- Data format:
  - How many bits?, parity?, number of stop bits?
- Transmission speed
  - Buad rate (bit/sec)

UART TIMING (8-BIT DATA, STOP=1-BIT, NO PARITY, BAUD RATE=115200)

Sending data byte 0x65

1-bit period = 1/115200 = 8.68 usec



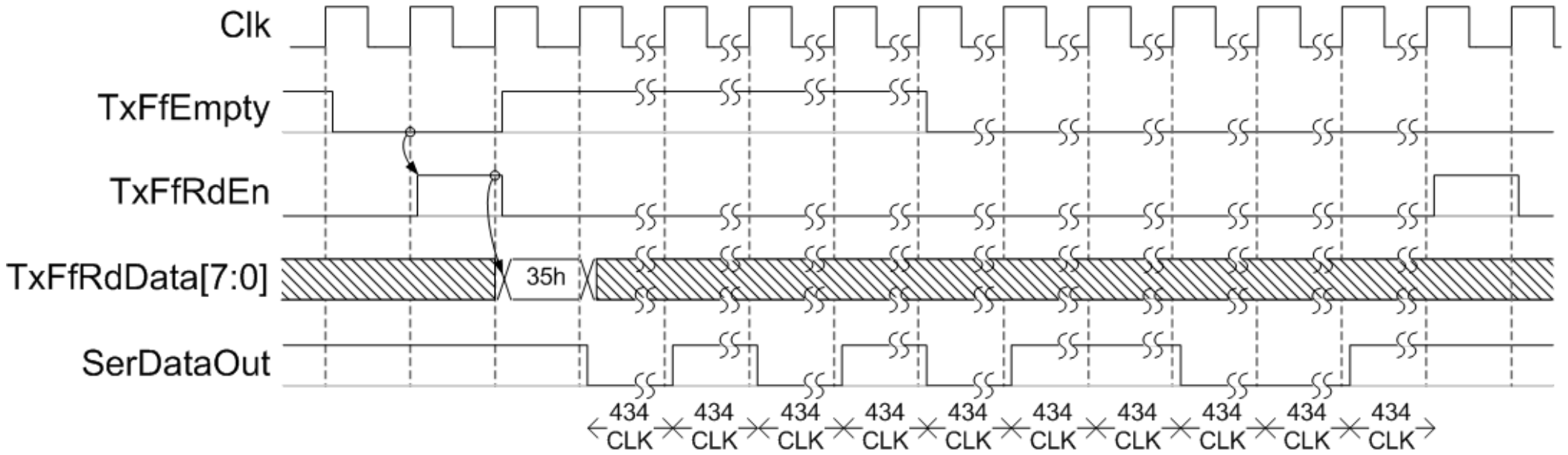
Q: ถ้าวงจรทำงานที่ clock ความถี่ 50 MHz (1 clock = 20 ns)
ต้องใช้ counter เพื่อ delay ไปกี่ clock เพื่อจะรับ/ส่งข้อมูลแต่ละ bit

TX UART INPUT/OUTPUT

```
Entity TxSerial Is
    Port
    (
        RstB        : in    std_logic;
        Clk         : in    std_logic;

        TxFfEmpty   : in    std_logic;
        TxFfRdData  : in    std_logic_vector( 7 downto 0 );
        TxFfRdEn    : out   std_logic;

        SerDataOut  : out   std_logic
    );
End Entity TxSerial;
```



TX UART DESIGN STEP

1. ออกแบบวงจรสร้างสัญญาณสำหรับ SHIFT BIT ทุก ๆ 434 CLOCK
2. ออกแบบวงจรแปลงข้อมูล data input จาก 8-bit ให้ออกมาทีละ bit โดยเริ่มจาก LSB ไป MSB พร้อมทั้งเติม START BIT และ STOP BIT
3. ออกแบบวงจรสร้างสัญญาณ TxFfRdEn เพื่ออ่านข้อมูลเข้ามาประมวลผล

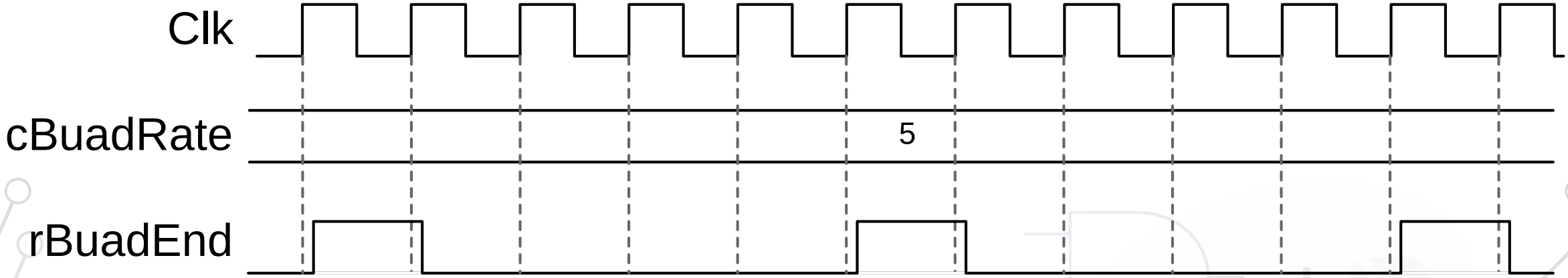
TX UART (QUESTION1)

คำถาม 1 : ออกแบบวงจรสร้าง PULSE 1 CLOCK

ให้ออกแบบวงจรที่สร้าง pulse ที่มีค่าเป็น ‘1’ เป็นเวลา 1 clock cycle ในทุก ๆ คาบเวลา

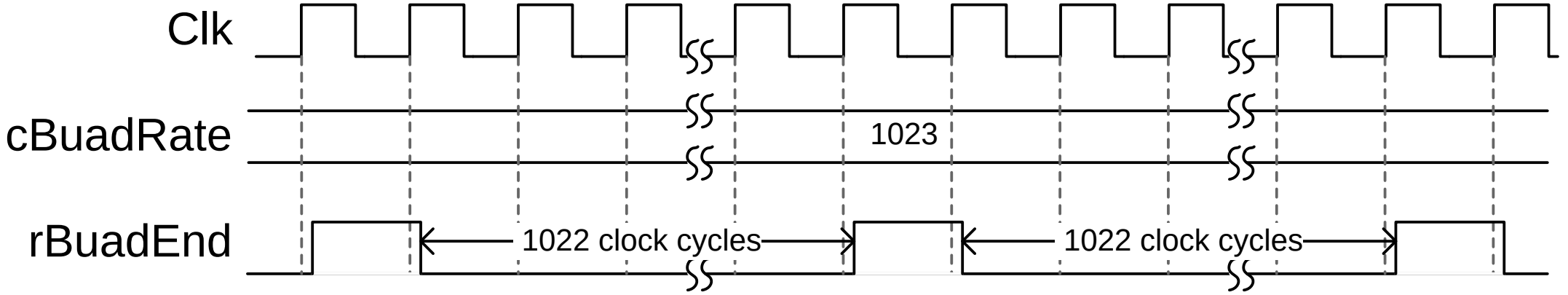
โดยรองรับคาบเวลาที่มีค่าตั้งแต่ 1 – 1023 เช่น

1) เมื่อคาบเวลามีค่าเป็น 5 จะสร้าง pulse เป็น ‘1’ เป็นเวลา 1 clock ในทุก ๆ 5 clock ดังนี้



คำถาม 1 : ออกแบบวงจรสร้าง PULSE 1 CLOCK

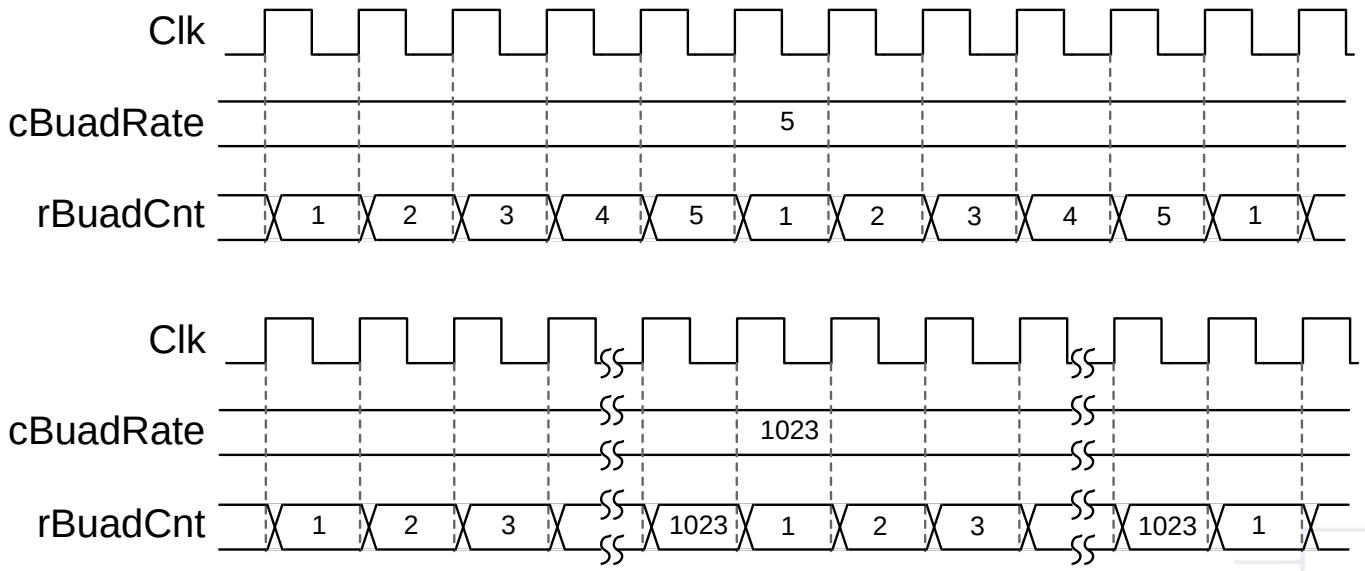
2) เมื่อคาบเวลามีค่าเป็น 1023 จะสร้าง pulse เป็น ‘1’ เป็นเวลา 1 clock ในทุก ๆ 1023 clock ดังนี้



Q: วงจรอะไรบ้างที่ควรนำมาใช้ในโจทย์นี้

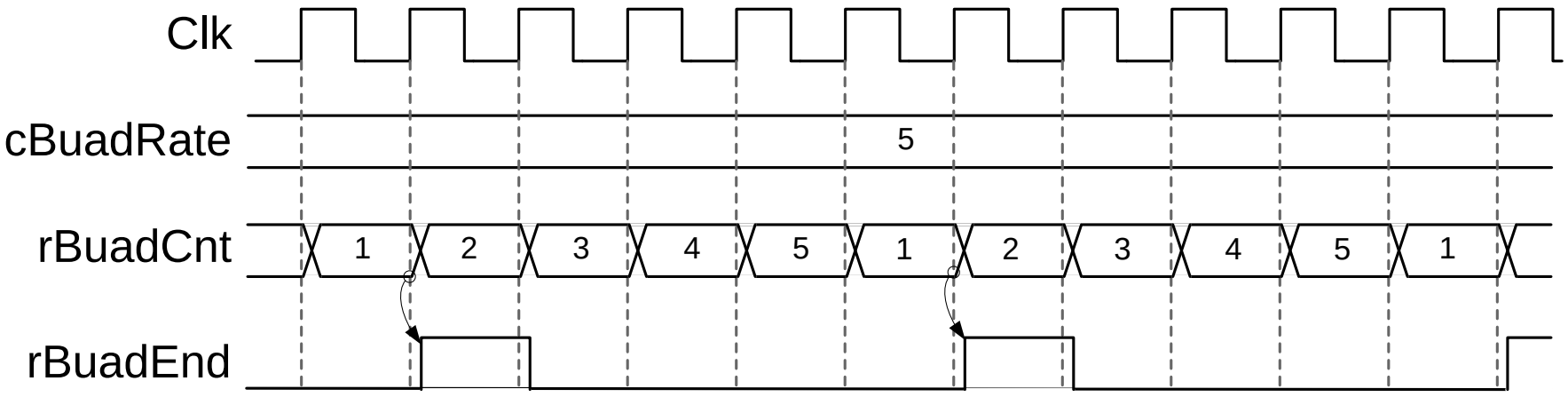
คำตอบ 1 : ออกแบบวงจรสร้าง PULSE 1 CLOCK

1) สร้าง counter ที่ปรับคาบได้ตั้งแต่ 1 – 1023 = 10-bit counter



คำตอบ 1 : ออกแบบวงจรสร้าง PULSE 1 CLOCK

2) ใช้ comparator เพื่อหาจุดที่ counter จะมีค่านี้เพียง 1 ครั้งในแต่ละคาบการทำงาน



Q: ค่าเท่าไหร่ดีที่จะ compare

VHDL CODE 1

```
------------------------------------------------------------------
-- Constant declaration
------------------------------------------------------------------

    constant    cbuadCnt    : integer := 434;

------------------------------------------------------------------
-- Signal declaration
------------------------------------------------------------------

    signal  rBuadCnt    : std_logic_vector( 9 downto 0 );
    signal  rBuadEnd    : std_logic;
```

```
u_rBuadCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rBuadCnt    <= conv_std_logic_vector(cbuadCnt, 10);
        else
            if ( rBuadCnt=1 ) then
                rBuadCnt    <= conv_std_logic_vector(cbuadCnt, 10);
            else
                rBuadCnt    <= rBuadCnt - 1;
            end if;
        end if;
    end if;
End Process u_rBuadCnt;

u_rBuadEnd : Process (Clk) Is
Begin
    if ( rising_edge(CLK) ) then
        if ( RstB='0' ) then
            rBuadEnd    <= '0';
        else
            if ( rBuadCnt=1 ) then
                rBuadEnd    <= '1';
            else
                rBuadEnd    <= '0';
            end if;
        end if;
    end if;
End Process u_rBuadEnd;
```

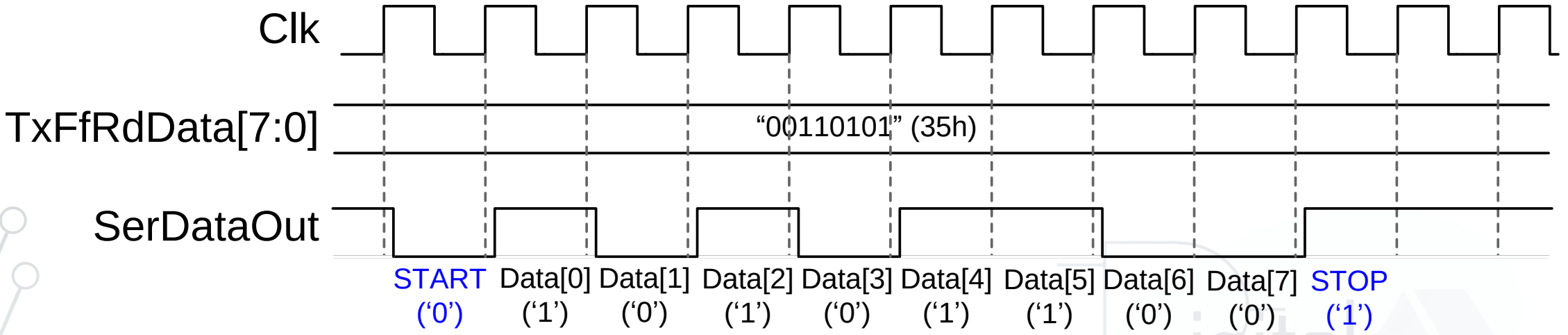
TX UART (QUESTION2)

คำถาม 2 : ออกแบบวงจรเพื่อส่ง DATA ขนาด 1 BIT

ให้ออกแบบวงจรที่รับ data ขาเข้ามาขนาด 8-bit และทยอยส่งออกไปทีละ bit โดยเริ่มต้นจาก LSB ไป MSB

การออกแบบ จะใช้ protocol แบบ UART คือ ต้องเริ่มต้นด้วย start bit (ค่า ‘0’)

ตามด้วย data 8-bit (data0, data1,…, data7) และจบด้วย stop bit (ค่า ‘1’)

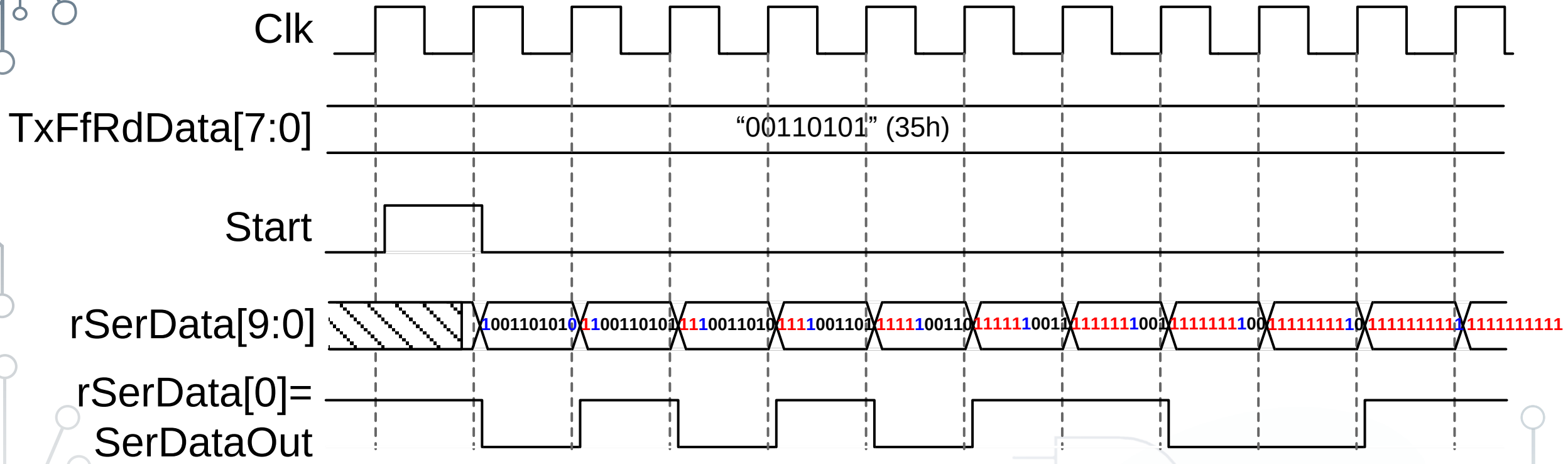


คำตอบ 2 : ออกแบบวงจรเพื่อส่ง DATA ขนาด 1 BIT

1) หลักการของการแปลงสัญญาณที่ input เป็นแบบ Parallel ขนาด 8-bit ให้ output ออกเป็น Serial ขนาด 1-bit คือ วงจรที่เรียกว่า parallel – to – serial ซึ่งตรงกับการทำงานของ shift register นั่นเอง

2) การส่งออกต้องมี start bit และ stop bit ด้วย เพื่อคงหลักการของการใช้ shift register อยู่เหมือนเดิม เมื่อเราต้องการส่งข้อมูลทั้งหมด 10 clock เราจึงปรับ shift register ของเรา ให้รับ input เป็นขนาด 10-bit แทน เพื่อจะได้โหลดค่า start bit และ stop bit เข้าไปพร้อมกับ data input เลย

คำตอบ 2 : ออกแบบวงจรเพื่อส่ง DATA ขนาด 1 BIT



VHDL CODE 2

```
-------------------------------------------------------------------------
-- Signal declaration
-------------------------------------------------------------------------

  signal  rSerData    : std_logic_vector( 9 downto 0 );

-------------------------------------------------------------------------
-- Output assignment
-------------------------------------------------------------------------

   SerDataOut  <= rSerData(0);

-------------------------------------------------------------------------
-- DFF
-------------------------------------------------------------------------

   u_rSerData : Process (Clk) Is
   Begin
       if ( rising_edge(Clk) ) then
           if ( RstB='0' ) then
               rSerData        <= (others=>'1');
           else
               if ( Start='1' ) then
                   rSerData(9)     <= '1';
                   rSerData(8 downto 1) <= TxFfRdData;
                   rSerData(0)     <= '0';
               else
                   rSerData    <= '1' & rSerData(9 downto 1);
               end if;
           end if;
       end if;
   End Process u_rSerData;
```

คำถาม 2 เพิ่มเติม: ออกแบบวงจรเพื่อส่ง DATA ขนาด 1 BIT

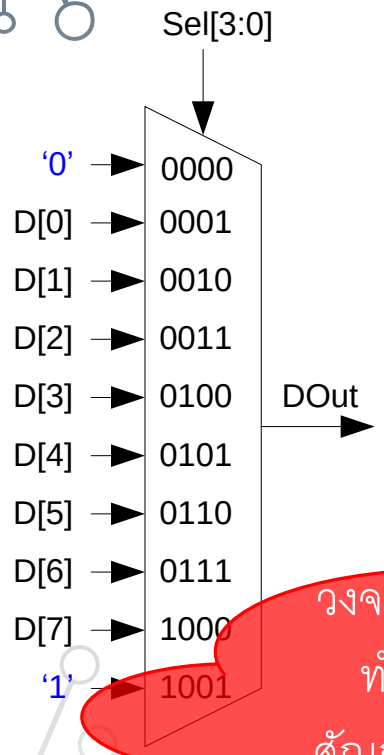
1) หลักการของการแปลงสัญญาณที่ input เป็นแบบ Parallel ขนาด 8-bit ให้ output ออกเป็น Serial ขนาด 1-bit คือ วงจรที่เรียกว่า parallel – to – serial ซึ่งตรงกับการทำงานของ shift register นั่นเอง

2) การส่งออกต้องมี start bit และ stop bit ด้วย เพื่อคงหลักการของการใช้ shift register อยู่เหมือนเดิม เมื่อเราต้องการส่งข้อมูลทั้งหมด 10 clock เราจึงปรับ shift register ของเรา ให้รับ input เป็นขนาด 10-bit แทน เพื่อจะได้โหลดค่า start bit และ stop bit เข้าไปพร้อมกับ data input เลย

Q: นอกจาก Shift register แล้ว โดยทั่วไป เมื่อเจอโจทย์ข้อนี้ คนมักจะออกแบบด้วยอะไร

คำตอบ 2 เพิ่มเติม : ออกแบบวงจรเพื่อส่ง DATA ขนาด 1 BIT

ส่วนใหญ่เมื่อเห็น specification แล้ว มักจะใช้วงจร mux 10-to-1 แทนการใช้ shift register




```
u_rSerData : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rSerData    <= (others=>'0');
        else
            if ( Start='1' ) then
                rSerData(7 downto 0) <= TxFfRdData;
            else
                rSerData(7 downto 0) <= rSerData(7 downto 0);
            end if;
        end if;
    end if;
End Process u_rSerData;
```

```
u_rSerDataOut : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rSerDataOut <= '1';
        else
            case ( Sel(3 downto 0) ) is
                when "0000" => rSerDataOut  <= '0';
                when "0001" => rSerDataOut  <= rSerData(0);
                when "0010" => rSerDataOut  <= rSerData(1);
                when "0011" => rSerDataOut  <= rSerData(2);
                when "0100" => rSerDataOut  <= rSerData(3);
                when "0101" => rSerDataOut  <= rSerData(4);
                when "0110" => rSerDataOut  <= rSerData(5);
                when "0111" => rSerDataOut  <= rSerData(6);
                when "1000" => rSerDataOut  <= rSerData(7);
                when others => rSerDataOut  <= '1';
            end case;
        end if;
    end if;
End Process u_rSerDataOut;
```

วงจร mux 10-to-1 เป็นวงจรที่ใช้เวลาประมวลผลนาน ทำให้วงจรทำงานได้ที่ความถี่ต่ำลง และการสร้างสัญญาณ Sel เพื่อเลือกสัญญาณออกให้ถูกต้อง จะทำให้กินทรัพยากรของระบบมาก

TX UART (QUESTION3)

คำถาม 3 : ออกแบบวงจรสร้างสัญญาณอ่านข้อมูล

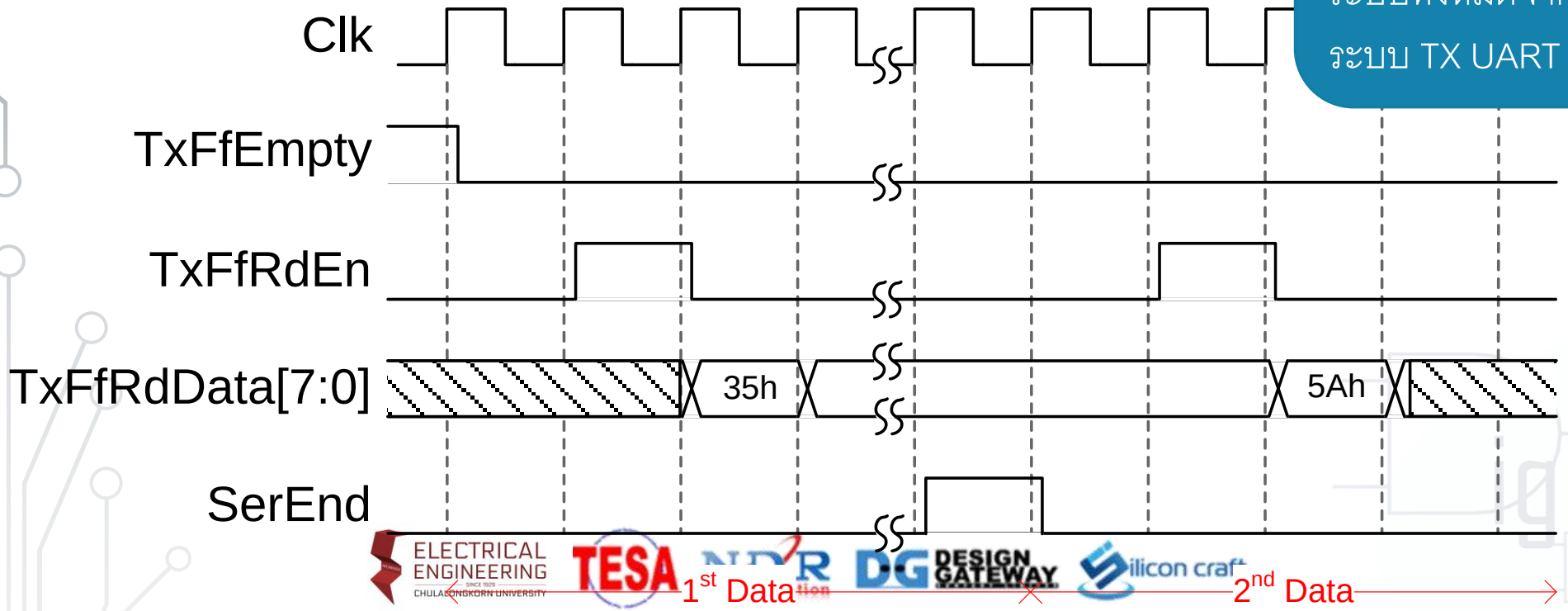
ให้ออกแบบวงจรสร้างสัญญาณ RdEn เพื่ออ่านข้อมูลจาก FIFO โดยมีขั้นตอนในแต่ละรอบคือ

1) ตรวจสอบสัญญาณ Empty ว่ามีค่าเท่ากับ ‘0’ ก่อน

2) สร้างสัญญาณ RdEn ออกไปเป็น Pulse 1 clock cycle

3) รอจนสัญญาณ SerEnd ขึ้นเป็น ‘1’ จึงจะกลับไปเริ่มต้นทำงานรอบใหม่

Note: สัญญาณ TxFfRdData เป็นสัญญาณ input ที่ไม่ได้ใช้งานในโจทย์ข้อนี้ แต่ได้วาด timing diagram ไว้ก่อน เพื่อเป็นข้อมูลที่จะนำไปใช้รวมระบบทั้งหมดจากคำถามทั้ง 3 ข้อ ให้กลายเป็นระบบ TX UART ที่สมบูรณ์

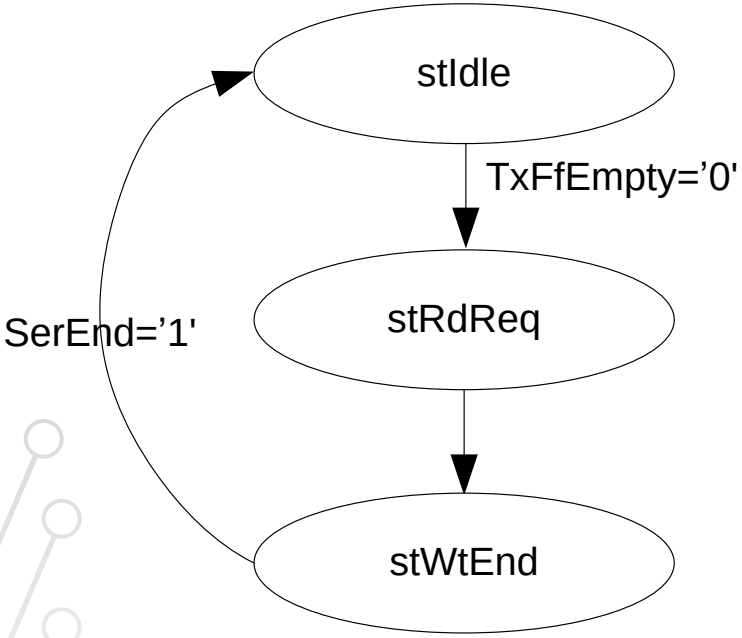


คำตอบ 3 : ออกแบบวงจรสร้างสัญญาณอ่านข้อมูล

โจทย์ข้อนี้เน้นการทำงานตามลำดับ คือ มีลำดับที่ชัดเจนทั้งหมด 3 ขั้นตอน

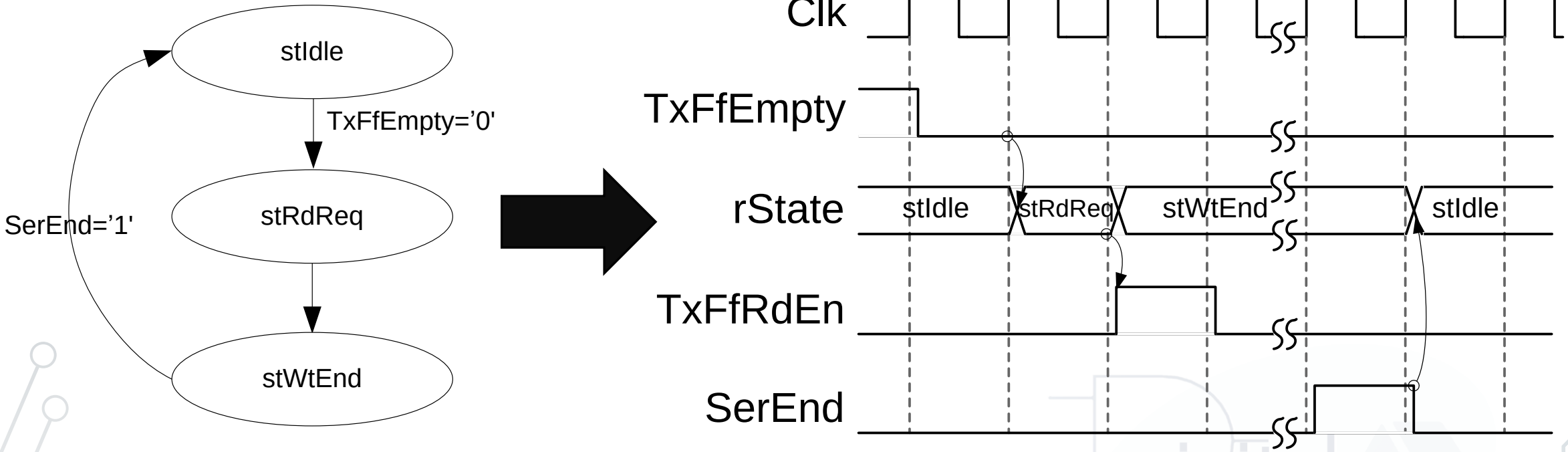
การออกแบบวงจรที่ทำงานตามลำดับนั้น นิยมใช้ State machine

สำหรับข้อนี้ จึงสามารถออกแบบได้โดยใช้ State machine ทั้งหมด 3 State ได้แก่



1) stIdle เพื่อตรวจสอบสัญญาณ TxFfEmpty ว่ามีค่าเท่ากับ '0' หรือยัง เพื่อเริ่มต้นทำงาน
2) stRdReq เพื่อสร้างสัญญาณ TxFfRdEn เป็น pulse 1 clock ออกไป
3) stWtEnd เพื่อรอจนกว่าสัญญาณ SerEnd มีค่าเท่ากับ '1' เพื่อจบรอบการทำงาน

คำตอบ 3 : ออกแบบวงจรสร้างสัญญาณอ่านข้อมูล



VHDL CODE 3

1

```
-----------------------------------------
-- Signal declaration
-----------------------------------------

    type     SerStateType is
                        (
                            stIdle   ,
                            stRdReq  ,
                            stWtEnd
                        );
    signal   rState   : SerStateType;

    signal   rTxFfRdEn    : std_logic;

-----------------------------------------
-- Output assignment
-----------------------------------------

    TxFfRdEn     <= rTxFfRdEn;
```

2

```
-----------------------------------------
-- DFF
-----------------------------------------

u_rState : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB = '0' ) then
            rState  <= stIdle;
        else
            case ( rState ) is
                when stIdle     =>
                    if ( TxFfEmpty='0' ) then
                        rState  <= stRdReq;
                    else
                        rState  <= stIdle;
                    end if;

                when stRdReq    =>
                    rState  <= stWtEnd;

                when stWtEnd        =>
                    if ( SerEnd='1' ) then
                        rState  <= stIdle;
                    else
                        rState  <= stWtEnd;
                    end if;

            end case;
        end if;
    end if;
End Process u_rState;
```

3

```
u_rTxFfRdEn : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rTxFfRdEn   <= '0';
        else
            if ( rState=stRdReq ) then
                rTxFfRdEn   <= '1';
            else
                rTxFfRdEn   <= '0';
            end if;
        end if;
    end if;
End Process u_rTxFfRdEn;
```

TX UART (QUESTION4)

คำถาม 4 : ออกแบบวงจร TXUART

ให้ออกแบบวงจร TXUART โดยมีลำดับดังนี้

1) ตรวจสอบสัญญาณ TxFfEmpty ว่ามีค่าเท่ากับ ‘0’ ก่อน

2) สร้างสัญญาณ TxFfRdEn ออกไปเป็น Pulse 1 clock cycle

3) นำข้อมูล TxFfRdData ที่อ่านได้ ส่งออกไปตาม UART protocol โดยมี buad rate=115200

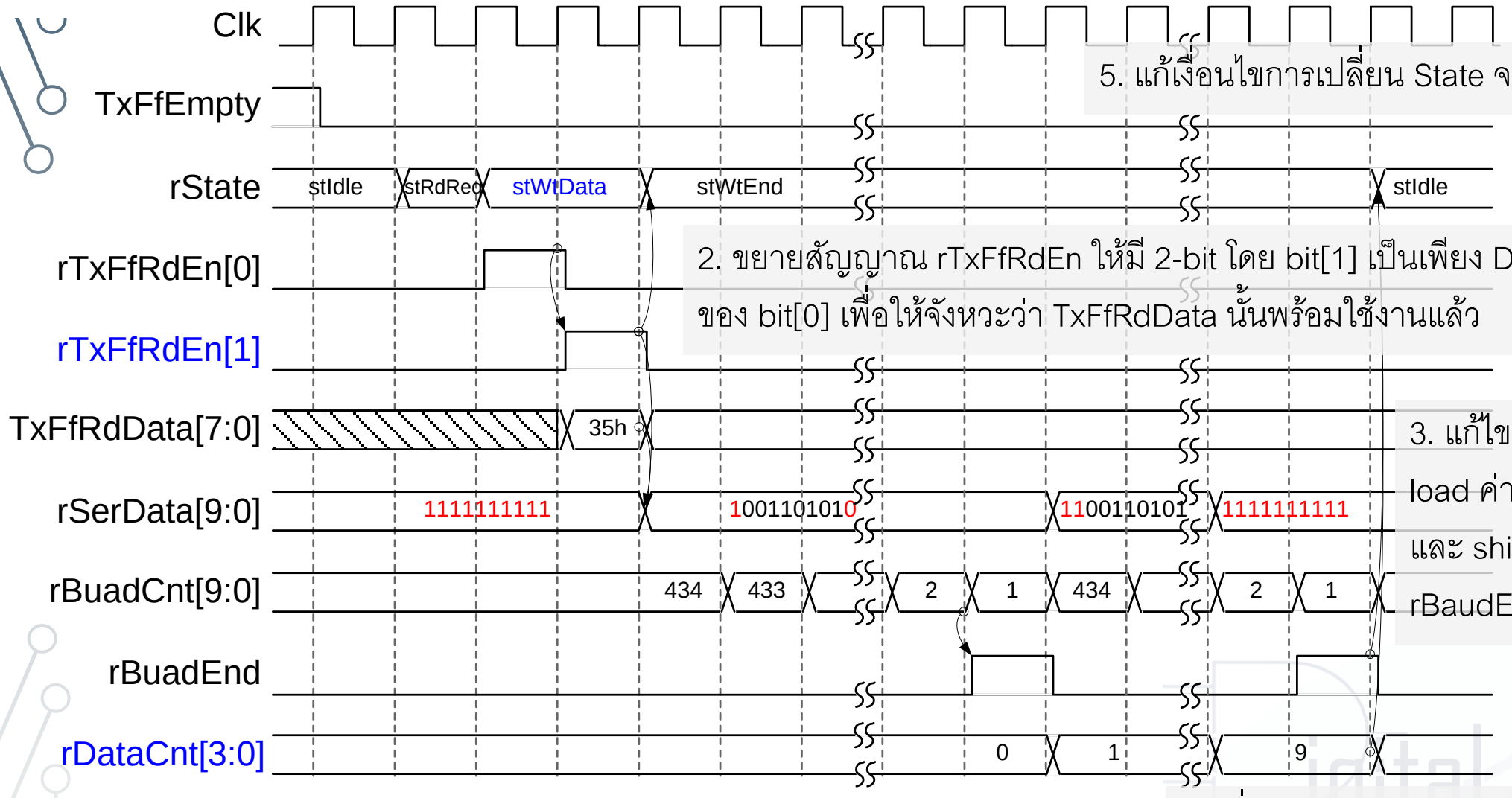
    3.1) ข้อมูลประกอบด้วย START bit (‘0’) + Data 8-bit (เริ่มจาก LSB ไป MSB) + STOP bit (‘1’)

    3.2) ข้อมูลจะ shift เพื่อส่งข้อมูลถัดไปทุก ๆ 434 clock

เมื่อส่งข้อมูลครบ ให้กลับไปขั้นตอนที่ 1 ใหม่

คำตอบ 4 : ออกแบบวงจร TXUART

1. เพิ่ม State ชื่อ stWtData เพื่อรอให้ data input (TxFfRdData) พร้อมใช้งานก่อน



5. แก้เงื่อนไขการเปลี่ยน State จาก stWtEnd เป็น stIdle

2. ขยายสัญญาณ rTxFfRdEn ให้มี 2-bit โดย bit[1] เป็นเพียง DFF ของ bit[0] เพื่อให้จังหวะว่า TxFfRdData นั้นพร้อมใช้งานแล้ว

3. แก้ไขสัญญาณ rSerData ให้ load ค่าจาก rTxFfRdEn[1]='1' และ shift ค่าเมื่อเจอ rBaudEnd='1'

4. เพิ่มสัญญาณ rDataCnt เพื่อนับว่าได้ shift สัญญาณ data ออกไปครบหรือยัง

VHDL CODE 4

```
-------------------------------------------------------------
-- Constant declaration
-------------------------------------------------------------

    constant    cbuadCnt    : integer := 434;

-------------------------------------------------------------
-- Signal declaration
-------------------------------------------------------------

    type    SerStateType is
                    (
                        stIdle  ,
                        stRdReq ,
                        stWtData,
                        stWtEnd
                    );
    signal  rState  : SerStateType;

    signal  rTxFfRdEn   : std_logic_vector( 1 downto 0 );
    signal  rSerData    : std_logic_vector( 9 downto 0 );
    signal  rBuadCnt    : std_logic_vector( 9 downto 0 );
    signal  rBuadEnd    : std_logic;
    signal  rDataCnt    : std_logic_vector( 3 downto 0 );

Begin

-------------------------------------------------------------
-- Output assignment
-------------------------------------------------------------

    TxFfRdEn    <= rTxFfRdEn(0);
    SerDataOut  <= rSerData(0);
```

1

4

```
u_rTxFfRdEn : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rTxFfRdEn       <= "00";
        else
            rTxFfRdEn(1)    <= rTxFfRdEn(0);
            if ( rState=stRdReq ) then
                rTxFfRdEn(0)    <= '1';
            else
                rTxFfRdEn(0)    <= '0';
            end if;
        end if;
    end if;
End Process u_rTxFfRdEn;
```

2

```
u_rSerData : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rSerData        <= (others=>'1');
        else
            if ( rTxFfRdEn(1)='1' ) then
                rSerData(9)     <= '1';
                rSerData(8 downto 1) <= TxFfRdData;
                rSerData(0)     <= '0';
            elsif ( rBuadEnd='1' ) then
                rSerData    <= '1' & rSerData(9 downto 1);
            else
                rSerData    <= rSerData;
            end if;
        end if;
    end if;
End Process u_rSerData;
```

3

TXUART DESIGN

1. Complete HDL Code of TXUART
2. Syntax Check (Quartus II)
3. Write testbench to confirm timing diagram (ModelSim)
4. Test on real board (SOF file)

SIMULATION CONDITION

1. ให้สัญญาณ TxFfEmpty เป็น ‘0’ ยาว ๆ เพื่อดูว่า TxSerial สามารถส่ง data ออกมาได้อย่างน้อย 3 ตัวต่อกันอย่างถูกต้อง โดยเปลี่ยนค่า TxFfRdData อย่างน้อย 3 ค่าคือ x”A5”, x”00” และ x”FF”
2. ให้ TxFfEmpty นั้นมีค่าเป็น ‘0’ เป็นช่วงสั้น ๆ ประมาณ 2-3 clock และรอจน TxSerial ส่ง data ทีละตัว แล้ววนลูปกลับไปปล่อย TxFfEmpty เป็น ‘0’ ใหม่อีกครั้ง ด้วยค่า data ใหม่ ทำแบบนี้เพื่อส่งข้อมูล 3 ตัว คือ x”A5”, x”00” และ x”FF” ตรวจสอบดูว่า data ที่ส่งออกมาถูกต้องไหม ขนาดความกว้างของ start bit, data และ stop bit ยังถูกต้องอยู่หรือเปล่า

BOARD CONNECTION