DIGITAL DESIGN WITH FPGA CAMP

DAY 4 RX UART

ELECTRICAL ENGINEERING
SINCE 1929
CHULALONGKORN UNIVERSITY
TESA
NDR Solution
DG DESIGN GATEWAY
Silicon craft

RX UART SPECIFICATION

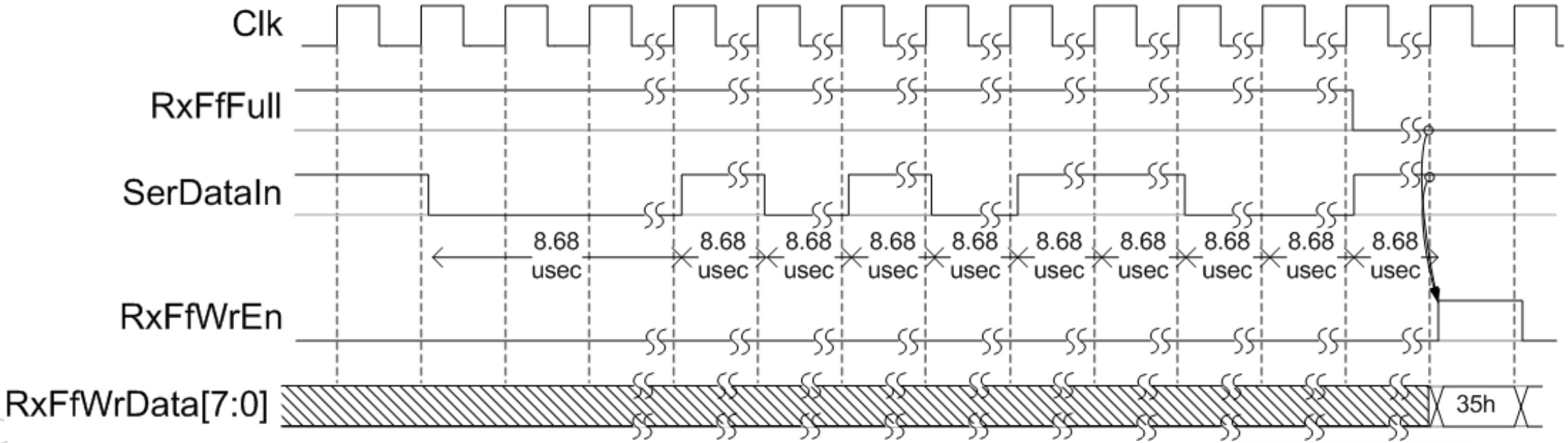
```
Entity RxSerial Is
    Port
    (
        RstB        : in    std_logic;
        Clk         : in    std_logic;

        SerDataIn   : in    std_logic;

        RxFfFull    : in    std_logic;
        RxFfWrData  : out   std_logic_vector( 7 downto 0 );
        RxFfWrEn    : out   std_logic
    );
End Entity RxSerial;
```

1. ความถี่ CLK เปลี่ยนเป็น 100 MHz (เดิม TX UART ใช้ความถี่ 50 MHz)
2. รับสัญญาณแบบ UART มาทางสัญญาณ SerDataIn โดยมีรูปแบบเหมือนเดิม คือ 8-bit data, STOP=1-bit, No parity, Buad Rate=115200 bps)
3. เมื่อได้รับข้อมูลสมบูรณ์ (หลังจากได้ STOP bit แล้ว) ให้สร้างสัญญาณ RxFfWrEn='1' เป็น pulse 1 clock พร้อมกับข้อมูล 8-bit ที่ได้มาไปที่ RxFfWrData
4. ก่อนที่จะสร้างสัญญาณ RxFfWrEn='1' ให้ตรวจสอบสัญญาณ RxFfFull ว่ามีค่าเป็น '0' ก่อน หากไม่ใช่ ยกเลิกการเขียนข้อมูลนั้นไป

RX UART SPECIFICATION



RX UART DESIGN STEP

1. คำนวณจำนวน CLOCK ที่ต้อง delay ไว้ในการรับสัญญาณแต่ละ bit จาก UART (115200 bps โดยใช้ 100 MHz) และออกแบบ counter เพื่อใช้ Sampling สัญญาณ
2. ออกแบบวงจรแปลงข้อมูล data input จาก 1-bit ให้กลายเป็นข้อมูล 8-bit โดยตัดสัญญาณ START BIT และ STOP BIT ทิ้ง และส่งข้อมูลนี้ผ่านสัญญาณ RxFfWrData
3. ออกแบบวงจรสร้างสัญญาณ RxFfWrEn เพื่อเขียนข้อมูลที่รับได้ออกไป

Note: การสุ่มสัญญาณนั้น จะไม่ทำที่ขอบสัญญาณ ให้สุ่มที่จุดกึ่งกลางของแต่ละ bit ซึ่งจะเป็นตำแหน่งข้อมูลที่เสถียรแล้ว

RECOMMENDED CODING STYLE

1. SIGNAL WITH/WITHOUT FLIP-FLOP

การออกแบบวงจร สัญญาณภายในจะมีอยู่ 2 ประเภท คือ

1) Combination signal without Flip-Flip ซึ่งจะเป็นสัญญาณที่จะทำงานทันทีเมื่อได้รับ input มา โดยการทำงานจะไม่เกี่ยวข้องกับ clock

2) Signal ที่จะมีการเปลี่ยนค่าตาม input ก็ต่อเมื่อเจอขอบขาขึ้น/ขาลงของ clock

ให้ประกาศสัญญาณประเภทแรก โดยขึ้นต้นตัว w

ส่วนสัญญาณประเภทที่สอง ให้ประกาศขึ้นต้นด้วยตัว r

Style for FPGA design

```
    signal  wRxFfRdEn   : std_logic;
    signal  rTxFfWrEn   : std_logic;
    signal  rTxFfWrData : std_logic_vector( 7 downto 0 );

Begin

-------------------------------------------------------------
-- Output Assignment
-------------------------------------------------------------

    RxFfRdEn    <= wRxFfRdEn;
    TxFfWrEn    <= rTxFfWrEn;
    TxFfWrData  <= rTxFfWrData;

-------------------------------------------------------------
-- Internal
-------------------------------------------------------------

    wRxFfRdEn   <= not RxFfEmpty;

    u_rTxFfWrEn : Process (Clk) Is
    Begin
        if ( rising_edge(Clk) ) then
            if ( RstB='0' ) then
                rTxFfWrEn   <= '0';
                rTxFfWrData <= (others=>'0');
            else
                if ( TxFfWrCnt(4 downto 3)/="11" ) then
                    rTxFfWrEn   <= '1';
                    rTxFfWrData <= rTxFfWrData + 1;
                else
                    rTxFfWrEn   <= '0';
                    rTxFfWrData <= rTxFfWrData;
                end if;
            end if;
        end if;
    End Process u_rTxFfWrEn;
```

Note:

1. สำหรับการออกแบบบน FPGA : การประกาศชื่อสัญญาณที่ต่างกัน ทำให้เราระมัดระวังเวลาจะนำสัญญาณที่ขึ้นต้นด้วยตัว w เพราะเป็นสัญญาณที่มี delay มาแล้ว หากเรานำไปใช้ ค่า delay รวมของวงจรเรา จะเกิดจากวงจรปัจจุบันที่กำลังออกแบบ รวมกับ delay ของสัญญาณ w เองด้วย ดังนั้นเวลาในการประมวลผลรวมกว่าจะได้ output จะมากกว่าสัญญาณที่ขึ้นต้นด้วยตัว r ซึ่งจะส่งผลทำให้วงจรเราทำงานได้ที่ความถี่ clock ที่ต่ำลง ดังนั้น process เกือบทั้ง code จะทำงานภายใต้ clock และสร้างสัญญาณ r แทบทั้งหมด เฉพาะสัญญาณจำนวนน้อย ที่ทำงานไม่ทันจริง ๆ จึงจะสร้างโดยใช้ w เพื่อทำงานทันที

2. สำหรับการออกแบบบน ASIC : Coding style ที่นิยมใช้งานเป็นมาตรฐาน คือ แยกส่วน code ที่เป็น combination ทั้งหมดเป็น w ส่วนสัญญาณ r นั้นจะเป็นแค่ D Flip Flop ธรรมดาเท่านั้น ดังนั้น process เกือบทั้ง code จะเป็นการสร้างสัญญาณ w แทบทั้งหมด มี r เพียง process สุดท้าย ที่เป็น DFF

2. 1 PROCESS 1 SIGNAL

```
u_rTxFfWrEn : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rTxFfWrEn   <= '0';
        else
            if ( TxFfWrCnt(4 downto 3)/="11" ) then
                rTxFfWrEn   <= '1';
            else
                rTxFfWrEn   <= '0';
            end if;
        end if;
    end if;
End Process u_rTxFfWrEn;

u_rTxFfWrData : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rTxFfWrData <= (others=>'0');
        else
            if ( rTxFfWrEn='1' ) then
                rTxFfWrData <= rTxFfWrData + 1;
            else
                rTxFfWrData <= rTxFfWrData;
            end if;
        end if;
    end if;
End Process u_rTxFfWrData;
```

การแยกการสร้าง signal ออกจากกัน ทำให้เราสามารถวิเคราะห์วงจรได้ง่าย ว่าสัญญาณใดจะเป็น critical path และการสร้างสัญญาณนั้น ๆ สิ้นเปลืองไปไหม สามารถ optimize ได้อีกไหม รวมถึงสามารถตรวจสอบได้ง่าย ว่าเราเขียน code ผิดไปหรือเปล่า

3. OTHER SIGNAL DECLARATION

```
------------------------------------------------------------
-- Constant Declaration
------------------------------------------------------------

    constant    cBaudrate    : std_logic_vector( 7 downto 0 ) := x"75";

------------------------------------------------------------
-- Signal Declaration
------------------------------------------------------------

    type    SerStateType is
                    (
                        stIdle  ,
                        stStart ,
                        stData  ,
                        stStop
                    );
    signal  rSerState   : SerStateType;
    signal  rBuadCnt    : std_logic_vector( 7 downto 0 );
```

- Constant ต่าง ๆ ขึ้นต้นด้วยตัว c
- ชื่อ State จะขึ้นต้นด้วย st
- สัญญาณที่เป็น State จะลงท้ายด้วย State
- สัญญาณที่เป็น counter จะลงท้ายด้วย Cnt

การแยกชื่อสัญญาณให้ชัดเจน จะทำให้เราจำสัญญาณ และจำการทำงานของมันได้ง่าย เวลาที่ code มีความซับซ้อนมาก

4. IF WITH ELSE

```
u_rDataLat : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( rDataEn='1' ) then
            rDataLat    <= DataIn;
        else
            rDataLat    <= rDataLat;
        end if;
    end if;
End Process u_rDataLat;
```

เมื่อเขียน IF ภายใต้ process จะต้องใส่ else ทุกครั้ง แม้ว่าจะเป็นการใส่ค่าเดิม เพื่อเป็นการยืนยันว่า เราตั้งใจจะใส่ค่าเดิมลงไป ไม่ใช่ลืมเขียน code ลงไป

5. DELAY SIGNAL

```
    signal  rFfRdEn     : std_logic_vector( 3 downto 0 );
    signal  rFfRdData0  : std_logic_vector( 7 downto 0 );
    signal  rFfRdData1  : std_logic_vector( 7 downto 0 );
    signal  rFfRdData2  : std_logic_vector( 7 downto 0 );

Begin

------------------------------------------------------------------
-- Output Assignment
------------------------------------------------------------------


------------------------------------------------------------------
-- Internal
------------------------------------------------------------------

    u_rFfRdEn : Process (Clk) Is
    Begin
        if ( rising_edge(Clk) ) then
            if ( RstB='0' ) then
                rFfRdEn     <= "0000";
            else
                rFfRdEn     <= rFfRdEn(2 downto 0) & wFfRdEn;
            end if;
        end if;
    End Process u_rFfRdEn;

    u_rFfRdData : Process (Clk) Is
    Begin
        if ( rising_edge(Clk) ) then
            rFfRdData2      <= rFfRdData1;
            rFfRdData1      <= rFfRdData0;
            rFfRdData0      <= FfRdData;
        end if;
    End Process u_rFfRdData;
```

บางครั้งเราต้อง delay สัญญาณไปหลาย ๆ CLOCK เพื่อรอเวลาในการทำงานพร้อมกับสัญญาณอื่น ๆ โดย

- ถ้าสัญญาณที่เราต้องการ delay นั้นมีขนาด 1-bit ให้ขยายสัญญาณเป็นหลาย ๆ bit แทน
- ถ้าสัญญาณที่เราต้องการ delay นั้นมีขนาดหลาย bit ให้เติมด้านท้ายเป็นหมายเลข 0 1 2 เพื่อบอกจำนวน CLOCK ที่เรา delay มา

6. COMMENT

ในการออกแบบ ควรใส่ comment กำกับไว้ให้มากที่สุด เพื่อบอกไอเดียในการออกแบบ ขยายความหมายของ code ของเรา เพื่อทำให้สามารถกลับมาอ่านได้เข้าใจ

CODING TECHNIQUE

ADD FLIP FLOP TO INPUT FROM DEVICE PIN

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

    signal  rSerDataIn  : std_logic;

Begin

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


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

```
    u_rSerDataIn : Process (Clk) Is
    Begin
        if ( rising_edge(Clk) ) then
            rSerDataIn      <= SerDataIn;
        end if;
    End Process u_rSerDataIn;
```

เมื่อเราอ่านข้อมูลจาก ขาของ device ตรง ๆ ไม่ใช่สัญญาณจาก logic ภายในด้วยกันเอง ให้นำสัญญาณไปผ่าน D Flip Flop เพื่อ synchronous ให้มีการเปลี่ยนแปลงของสัญญาณตามจังหวะขอบขาขึ้น/ขาลงของ clock ก่อนนำมาใช้งาน

ให้สร้างวงจรอ่านข้อมูลจาก rSerDataIn เท่านั้น ไม่อ่านข้อมูลจาก SerDataIn ซึ่งเป็น signal จากขา FPGA

ความรู้เพิ่มเติม

7.1 ข้อควรระวังในการออกแบบ http://bit.ly/fpgawarn

8.1 ตัวอย่าง coding style http://bit.ly/fpgacodingstyle

SIMULATION CONDITION

1. สร้างเงื่อนไขรับข้อมูล 3 ตัวอย่างต่อเนื่องกัน คือ 0xA5, 0xFF, 0x00 โดยให้แต่ละ bit กว้าง 8.68 usec แล้วตรวจสอบดูว่า สามารถรับค่าได้อย่างถูกต้องหรือไม่

2. สร้างเงื่อนไขการส่งข้อมูลโดยที่ stop='0' แล้วตรวจสอบว่า RxFfWrEn ต้องไม่ถูก set เป็น '1'

3. สร้างเงื่อนไขการส่งข้อมูลโดยที่ Full='1' แล้ว ตรวจสอบว่า RxFfWrEn ต้องไม่ถูก set เป็น '1'

4. ทำเหมือนข้อ 1 อีกครั้ง แต่ปรับความกว้าง bit ให้เล็กลงเหลือ 8 usec แล้วตรวจสอบดูว่า วงจรสามารถรับค่าได้อย่างถูกต้องหรือไม่