DIGITAL DESIGN WITH FPGA CAMP

DAY 1 TIMING DIAGRAM AND LOGIC DESIGN

TIMING DIAGRAM

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

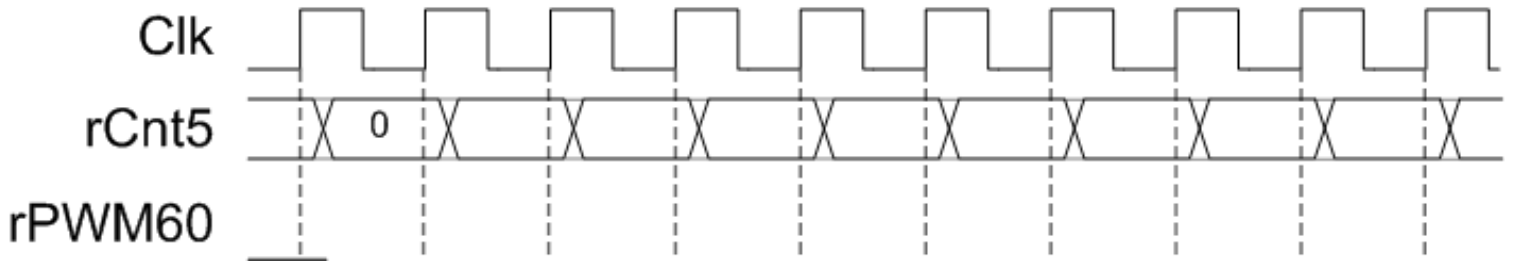
OVERVIEW

- การออกแบบวงจร จะเริ่มจากการวาด Timing diagram ที่เราต้องการขึ้นมาก่อน โดยเริ่มจากการวาดสัญญาณ input และ output ที่ต้องการ เป็น specification ของระบบ
- หลังจากนั้น เราจะเริ่มออกแบบว่า ควรใช้วงจร logic อะไรในการสร้างสัญญาณ output ให้ได้ตามที่ต้องการ และลองวาด timing ของสัญญาณจากวงจร logic ที่เราคิดไว้ขึ้นมาทีละสัญญาณ เช่น counter นับสัญญาณ หรือสัญญาณ pulse ที่มีค่า ‘1’/’0’ ตามจังหวะที่ต้องการ
- การวาดมักจะเริ่มจากสัญญาณควบคุมจังหวะการทำงานก่อน ว่าช่วงไหนทำงานอะไรบ้าง ส่วนใหญ่มักจะออกแบบด้วย state machine counter หรือ สัญญาณขนาด 1-bit ที่มีจังหวะ 1/0 ตามที่ต้องการ แล้วจึงค่อยวาดส่วนที่เป็นสัญญาณข้อมูลที่จะนำมาประมวลผล ซึ่งจะมีขนาดหลาย bit
- สุดท้าย จะเป็นการจูน timing ของสัญญาณที่ทำงานด้วยกัน ให้ทำงานตรง clock cycle เดียวกัน โดยการใส่ Flip Flop เพื่อเลื่อนจังหวะการทำงานให้ตรงกัน

Q1: PULSE WIDTH MODULATION TIMING DIAGRAM

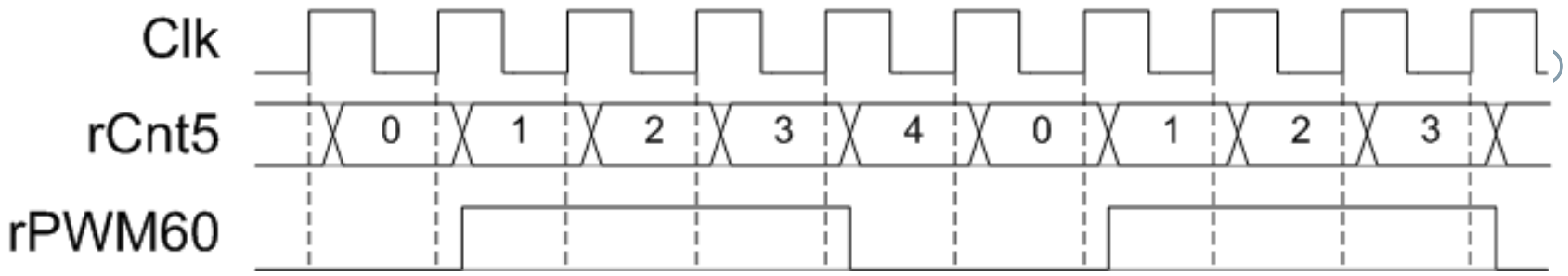
```
u_rCnt5 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt5(2 downto 0)   <= "000";
        else
            if ( rCnt5(2 downto 0)=4 ) then
                rCnt5(2 downto 0)   <= "000";
            else
                rCnt5(2 downto 0)   <= rCnt5(2 downto 0) + 1;
            end if;
        end if;
    end if;
End Process u_rCnt5;

u_rPWM60 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rPWM60      <= '0';
        else
            if ( rCnt5(2 downto 0)=0 ) then
                rPWM60  <= '1';
            elsif ( rCnt5(2 downto 0)=3 ) then
                rPWM60  <= '0';
            else
                rPWM60  <= rPWM60;
            end if;
        end if;
    end if;
End Process u_rPWM60;
```

Q: ลองวาด timing ของสัญญาณจาก code ด้านซ้ายมือ



A1: PULSE WIDTH MODULATION TIMING DIAGRAM

```
u_rCnt5 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt5(2 downto 0)   <= "000";
        else
            if ( rCnt5(2 downto 0)=4 ) then
                rCnt5(2 downto 0)   <= "000";
            else
                rCnt5(2 downto 0)   <= rCnt5(2 downto 0) + 1;
            end if;
        end if;
    end if;
End Process u_rCnt5;

u_rPWM60 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rPWM60      <= '0';
        else
            if ( rCnt5(2 downto 0)=0 ) then
                rPWM60  <= '1';
            elsif ( rCnt5(2 downto 0)=3 ) then
                rPWM60  <= '0';
            else
                rPWM60  <= rPWM60;
            end if;
        end if;
    end if;
End Process u_rPWM60;
```



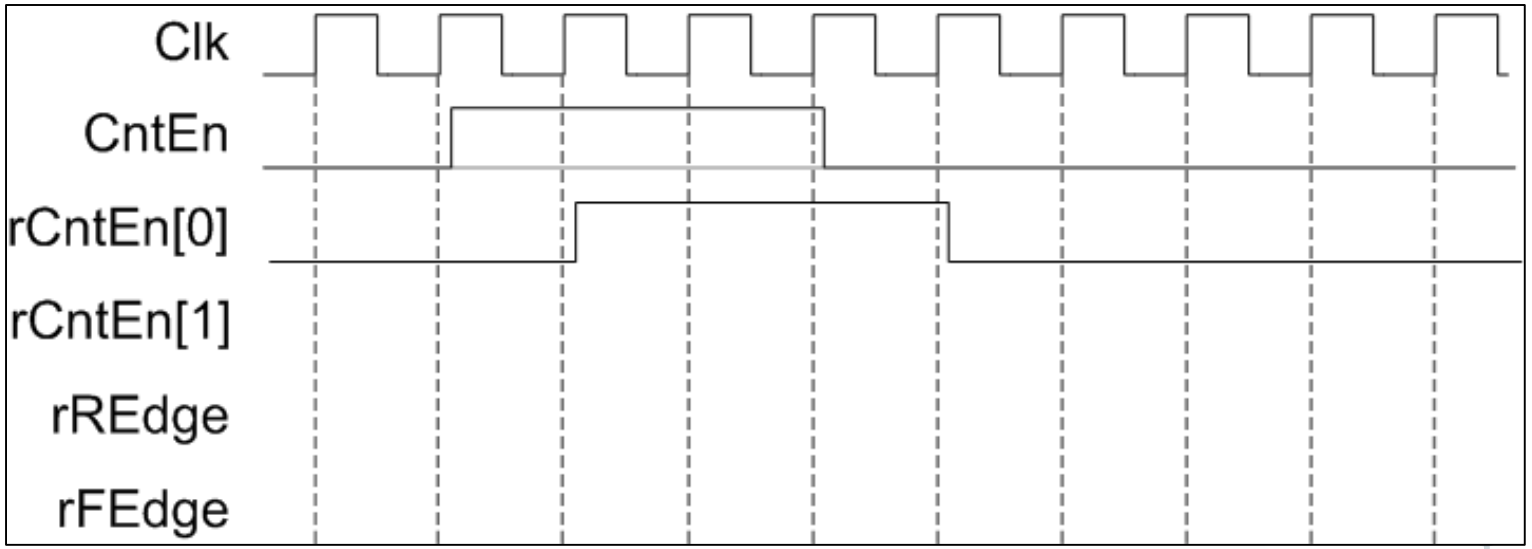
ตัวอย่างนี้ เป็นวงจรพื้นฐานการสร้างสัญญาณ PWM ที่มี duty cycle ตามที่ต้องการ ดังในโจทย์คือ จะสร้างสัญญาณที่จะ ON เป็น ‘1’ ทั้งหมด 60% ของคาบ นั่นก็คือ จะเป็น ‘1’ อยู่ 3 clock และเป็น ‘0’ อยู่ 2 clock

การออกแบบจึงเริ่มต้นด้วยการสร้าง counter พื้นฐาน สร้างจังหวะ 5 clock ขึ้นมา คือ เป็น counter นับ 0-4 แล้วค่อยนำ counter มาเปรียบเทียบเพื่อหาจังหวะการ set เป็น ‘1’ และ ‘0’ ให้ได้ตาม cycle ที่ต้องการ ดังในตัวอย่าง Set/Clear Flip Flop ชื่อ rPWM60 นี้ จะเป็น ‘1’ เมื่อเจอ Cnt=0 และเป็น ‘0’ เมื่อเจอ Cnt=3

Q2: EDGE DETECTION TIMING DIAGRAM

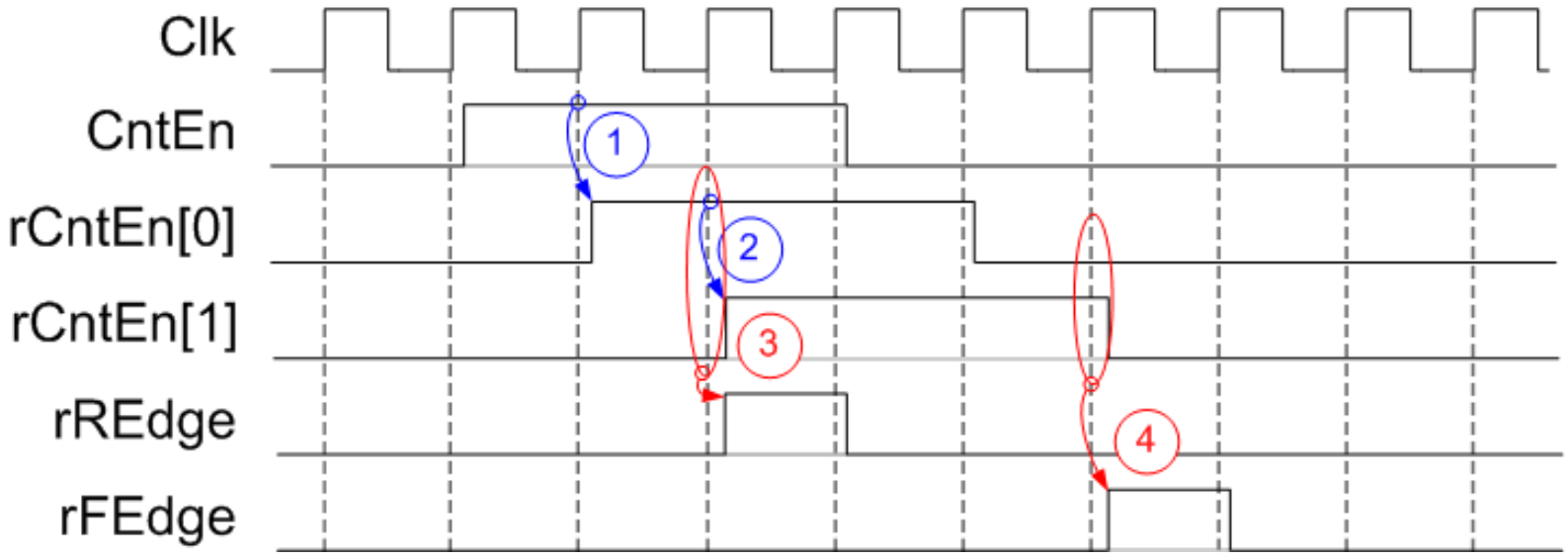
```
u_rCntEn : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCntEn(1 downto 0)  <= "00";
        else
            rCntEn(1)       <= rCntEn(0);
            rCntEn(0)       <= CntEn;
        end if;
    end if;
End Process u_rCntEn;

u_rREdge : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( rCntEn(1 downto 0)="01" ) then
            rREdge      <= '1';
        else
            rREdge      <= '0';
        end if;
    end if;
End Process u_rREdge;

u_FEdge : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( rCntEn(1 downto 0)="10" ) then
            rFEdge      <= '1';
        else
            rFEdge      <= '0';
        end if;
    end if;
End Process u_FEdge;
```

Q: ลองวาด timing ของสัญญาณที่เหลือ จาก code ด้านซ้ายมือ



A2: EDGE DETECTION TIMING DIAGRAM

```
u_rCntEn : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCntEn(1 downto 0)  <= "00";
        else
            rCntEn(1)       <= rCntEn(0);
            rCntEn(0)       <= CntEn;
        end if;
    end if;
End Process u_rCntEn;

u_rREdge : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( rCntEn(1 downto 0)="01" ) then
            rREdge      <= '1';
        else
            rREdge      <= '0';
        end if;
    end if;
End Process u_rREdge;

u_FEdge : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( rCntEn(1 downto 0)="10" ) then
            rFEdge      <= '1';
        else
            rFEdge      <= '0';
        end if;
    end if;
End Process u_FEdge;
```

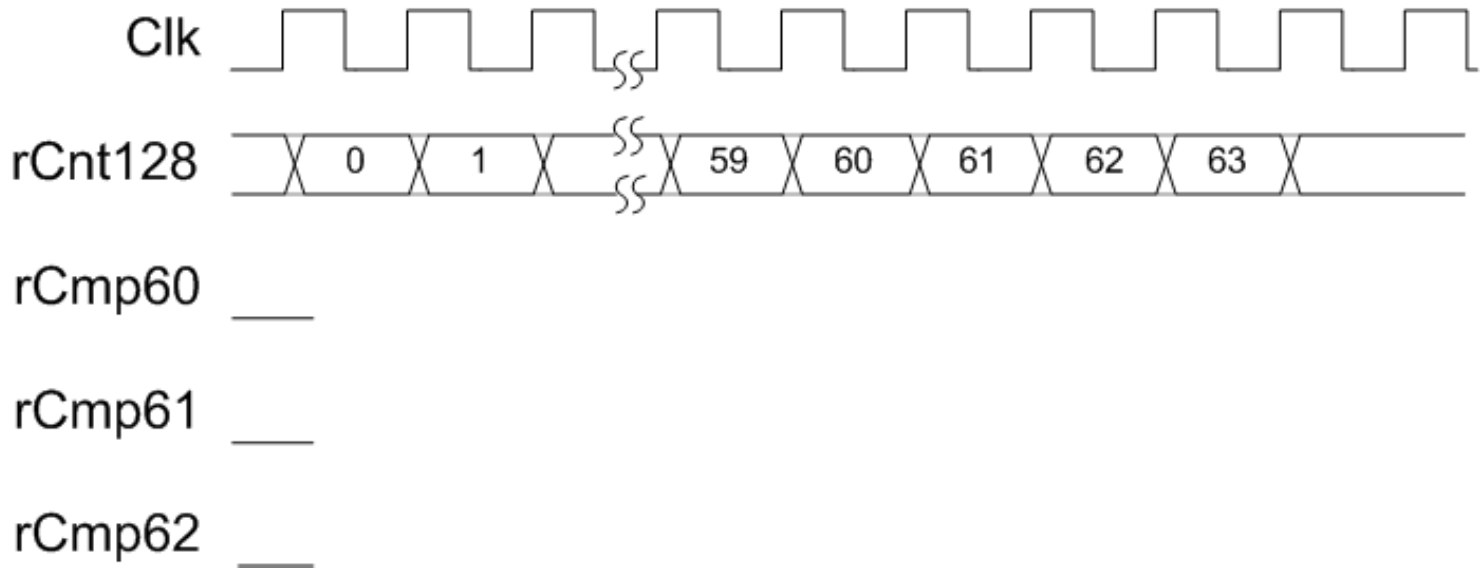


ตัวอย่างนี้ เป็นวงจรพื้นฐานที่สำคัญในการออกแบบ เมื่อต้องการทำงานเพียง 1 clock ในช่วงขอบขาขึ้นหรือขอบขาลงของสัญญาณ ๆ หนึ่ง โดยเราจะนำสัญญาณนั้นมาผ่าน D Flip-Flop 2 ตัว เพื่อให้มี delay ค่าระหว่างกัน และนำมาเปรียบเทียบค่า “01” เพื่อหาช่วงขอบขาขึ้น หรือเปรียบเทียบกับค่า “10” เพื่อหาช่วงขอบขาลง

Q3: COMPARATOR TIMING DIAGRAM

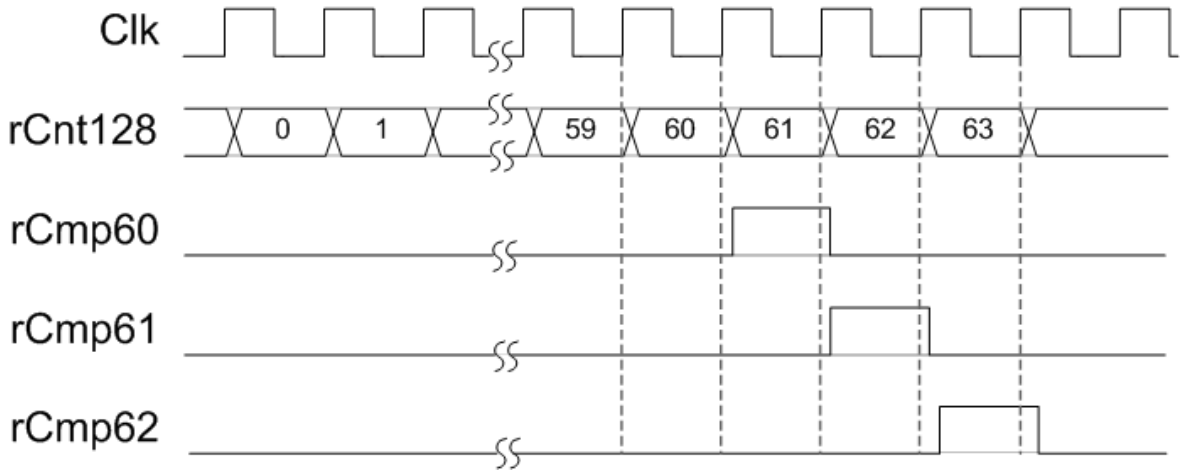
```
u_rCnt128 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt128(6 downto 0) <= (others=>'0');
        else
            rCnt128(6 downto 0) <= rCnt128(6 downto 0) + 1;
        end if;
    end if;
End Process u_rCnt128;

u_rCmp : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCmp60  <= '0';
            rCmp61  <= '0';
            rCmp62  <= '0';
        else
            if ( rCnt128(6 downto 0)=60 ) then
                rCmp60  <= '1';
            else
                rCmp60  <= '0';
            end if;

            if ( rCnt128(6 downto 0)=61 ) then
                rCmp61  <= '1';
            else
                rCmp61  <= '0';
            end if;

            if ( rCnt128(6 downto 0)=62 ) then
                rCmp62  <= '1';
            else
                rCmp62  <= '0';
            end if;
        end if;
    end if;
End Process u_rCmp;
```

Q: ลองวาด timing ของสัญญาณที่เหลือ จาก code ด้านซ้ายมือ



A3: COMPARATOR TIMING DIAGRAM

```
u_rCmp : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCmp60  <= '0';
            rCmp61  <= '0';
            rCmp62  <= '0';
        else
            if ( rCnt128(6 downto 0)=60 ) then
                rCmp60  <= '1';
            else
                rCmp60  <= '0';
            end if;

            if ( rCnt128(6 downto 0)=61 ) then
                rCmp61  <= '1';
            else
                rCmp61  <= '0';
            end if;

            if ( rCnt128(6 downto 0)=62 ) then
                rCmp62  <= '1';
            else
                rCmp62  <= '0';
            end if;
        end if;
    end if;
End Process u_rCmp;
```



ตัวอย่างนี้ เป็นตัวอย่างวงจรที่เรามักจะออกแบบกัน เมื่อต้องการทำงานในช่วง จังหวะที่ counter มีค่า 60 61 และ 62 ก็จะเขียน code สำหรับสร้างวงจร comparator ขนาด 7 bit ขึ้นมา เพื่อหาช่วงที่มีค่าเป็น 60 61 และ 62 ตามลำดับ

Q: เราสามารถออกแบบสัญญาณ rCmp60/61/62 ให้ประหยัดกว่านี้ได้หรือเปล่า ???

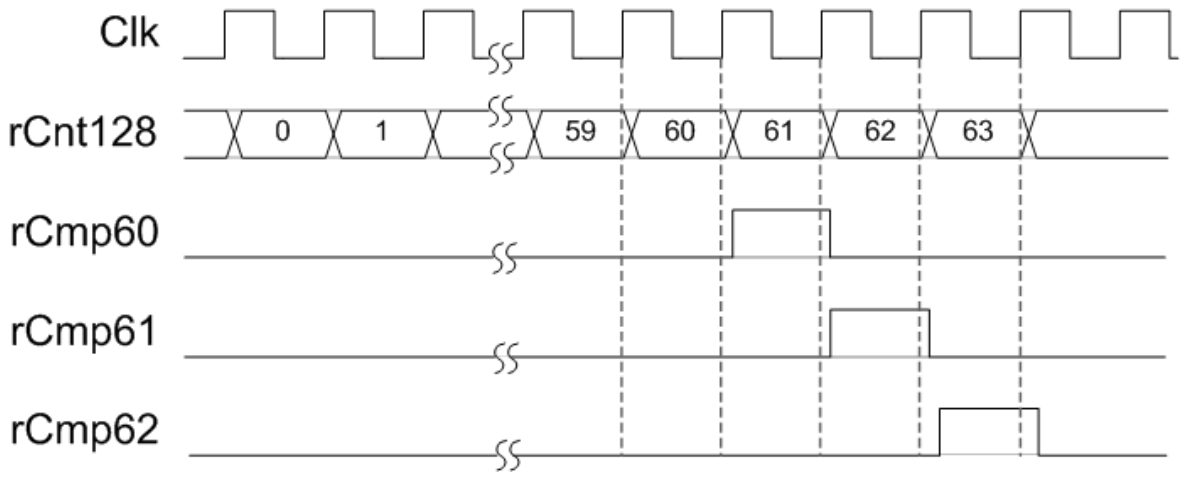
DESIGN BY TIMING DIAGRAM

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

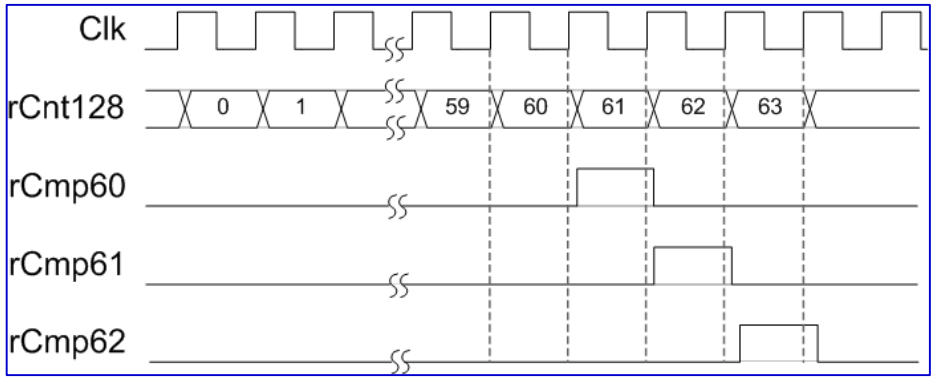
Q4: COMPARATOR OPTIMIZATION

```
u_rCmp : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCmp60  <= '0';
            rCmp61  <= '0';
            rCmp62  <= '0';
        else
            if ( rCnt128(6 downto 0)=60 ) then
                rCmp60  <= '1';
            else
                rCmp60  <= '0';
            end if;

            if ( rCnt128(6 downto 0)=61 ) then
                rCmp61  <= '1';
            else
                rCmp61  <= '0';
            end if;

            if ( rCnt128(6 downto 0)=62 ) then
                rCmp62  <= '1';
            else
                rCmp62  <= '0';
            end if;
        end if;
    end if;
End Process u_rCmp;
```

Q: เราสามารถออกแบบสัญญาณ rCmp60/61/62 ให้ประหยัดกว่านี้ได้หรือเปล่า ???



A4: COMPARATOR OPTIMIZATION

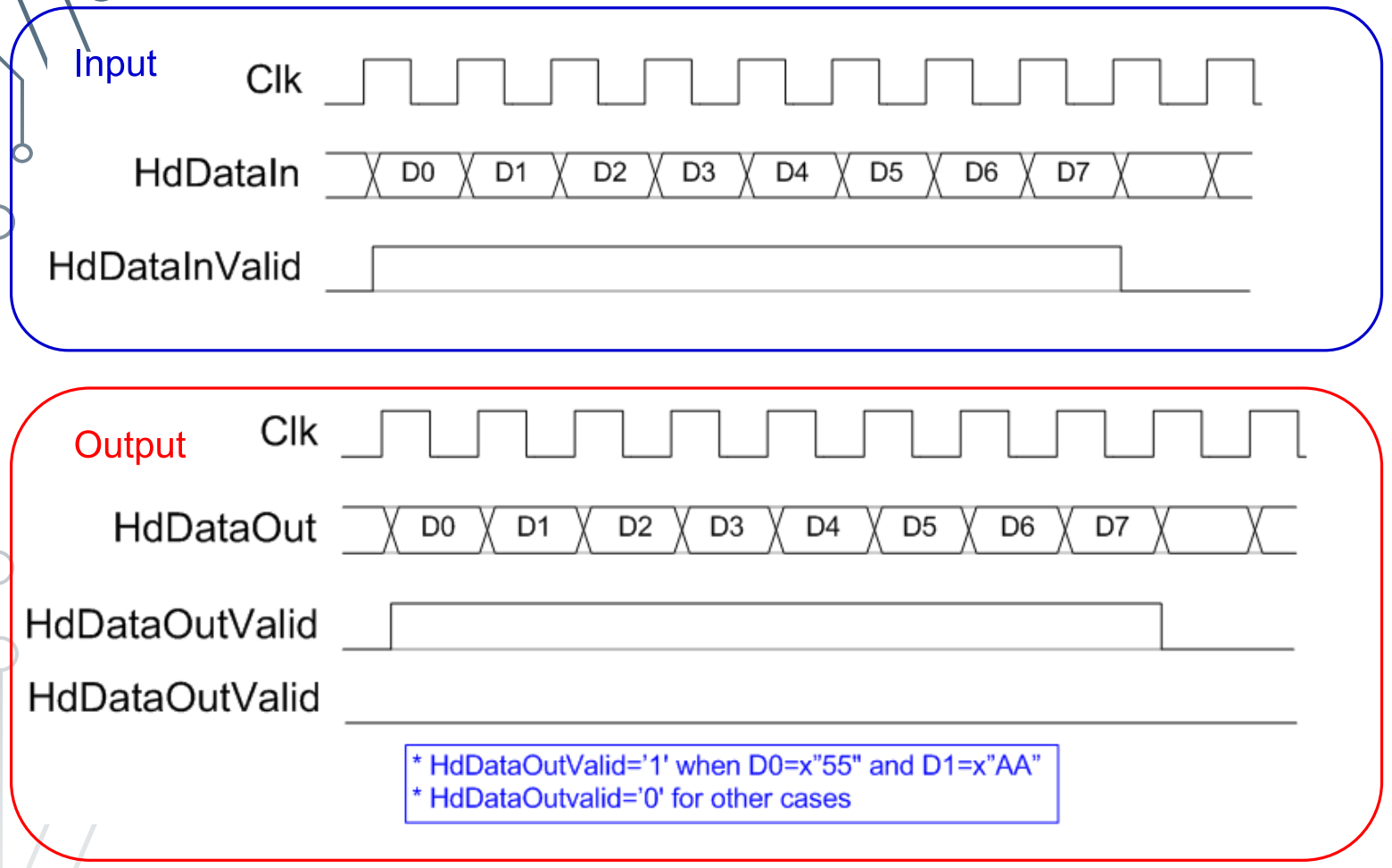



```
u_rCmp : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCmp60  <= '0';
            rCmp61  <= '0';
            rCmp62  <= '0';
        else
            rCmp61  <= rCmp60;
            rCmp62  <= rCmp61;
            if ( rCnt128(6 downto 0)=60 ) then
                rCmp60  <= '1';
            else
                rCmp60  <= '0';
            end if;
        end if;
    end if;
End Process u_rCmp;
```

ถ้าเรามอง timing diagram ให้ดี จะเห็นว่าสัญญาณ rCmp61 และ rCmp62 นั้น สามารถสร้างได้จากการ delay สัญญาณ rCmp60 ไปอีก 1 clock และ 2 clock ตามลำดับ ดังนั้น เราสามารถปรับวงจรใหม่ โดยการใส่ DFF ลงไปแทน เพื่อสร้าง rCmp61 และ rCmp62 ซึ่งจะประหยัดกว่าการใช้ comparator 7 bit ลงได้มาก

นี่คือตัวอย่างให้เห็นประโยชน์ของการออกแบบโดยวาด timing diagram ขึ้นมา แล้วจึงเขียน code โดยใช้ logic ให้เหมาะสมกับ timing diagram ที่ต้องการ ไม่ใช่วาด timing diagram เพื่อตรวจสอบ code ที่เราเขียนไปแล้วเท่านั้น

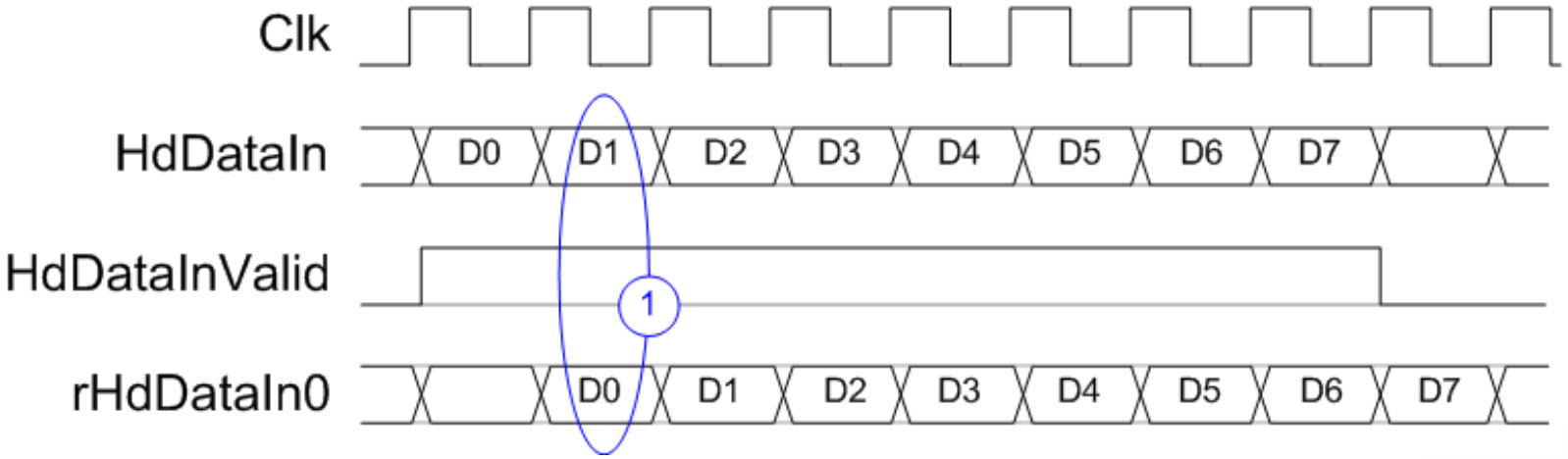
Q5: SYNC TIMING



Q: ให้ออกแบบวงจร ที่จะส่งผ่านข้อมูลทั้งหมด 8 ตัว จาก HdDataIn ไปที่ HdDataOut ก็ต่อเมื่อสัญญาณ 2 byte แรก คือ D0=x"55" และ D1=x"AA" เท่านั้น หากเป็นค่าอื่น จะไม่ส่งผ่านไป โดยควบคุมผ่านสัญญาณ HdDataOutValid ที่จะเป็น '1' เพื่อบอกว่าข้อมูลนี้ถูกต้อง ส่งผ่านไปได้ และเป็น '0' เพื่อบอกปฏิเสธข้อมูลนี้ไป ไม่ส่งต่อ

A5.1: HOW TO COMPARE DATA

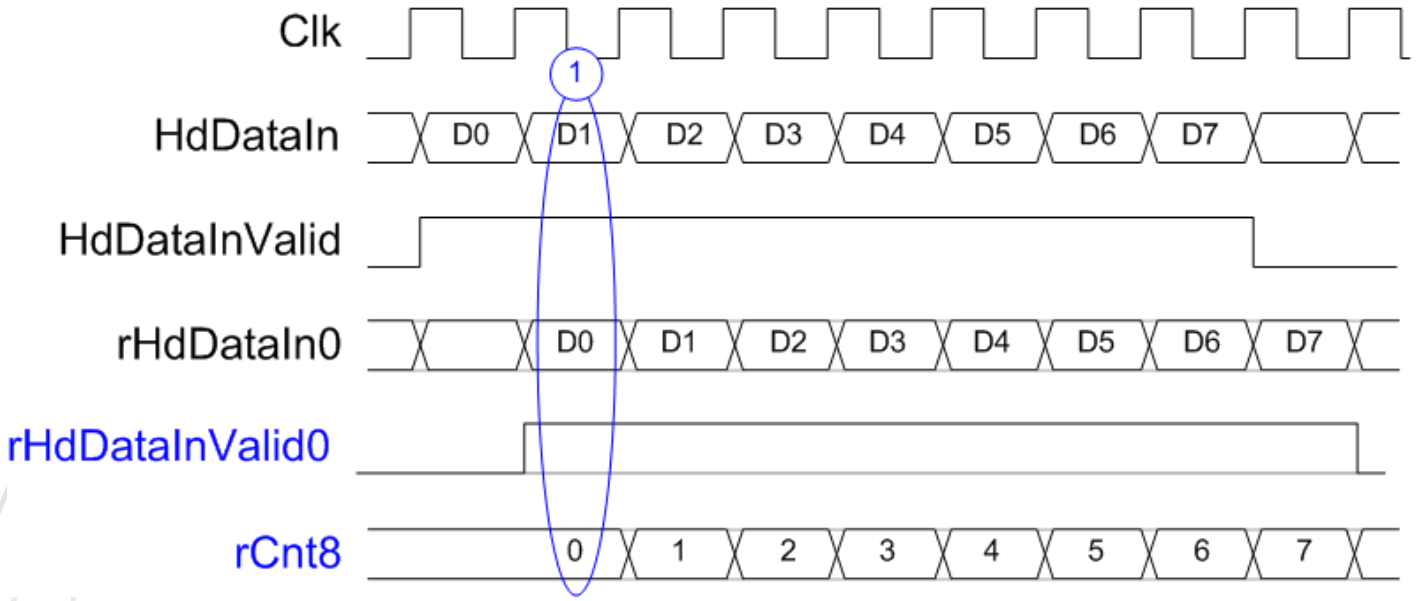
เมื่อต้องเปรียบเทียบข้อมูลก่อนหน้า และข้อมูลปัจจุบัน (D0 และ D1) พร้อมกัน จำเป็นต้องสร้าง logic เพื่อเก็บข้อมูลก่อนหน้าไว้รอข้อมูลปัจจุบันที่กำลังจะเข้ามาตัวต่อไป นั่นคือ หน้าที่และคุณสมบัติของ DFF ดังนั้น จะสร้างสัญญาณ rHddataIn0 มา เพื่อเก็บค่าข้อมูลก่อนหน้าไว้ สัญญาณก็จะเหมือน delay ไป 1 clock ดังรูป



ที่จังหวะที่ 1 ในรูป เราสามารถเปรียบเทียบข้อมูล D0 และ D1 ได้แล้ว ต่อไปคือการหาจังหวะการเปรียบเทียบให้ตรงกับจังหวะที่ 1

A5.2: HOW TO ENABLE COMPARATOR

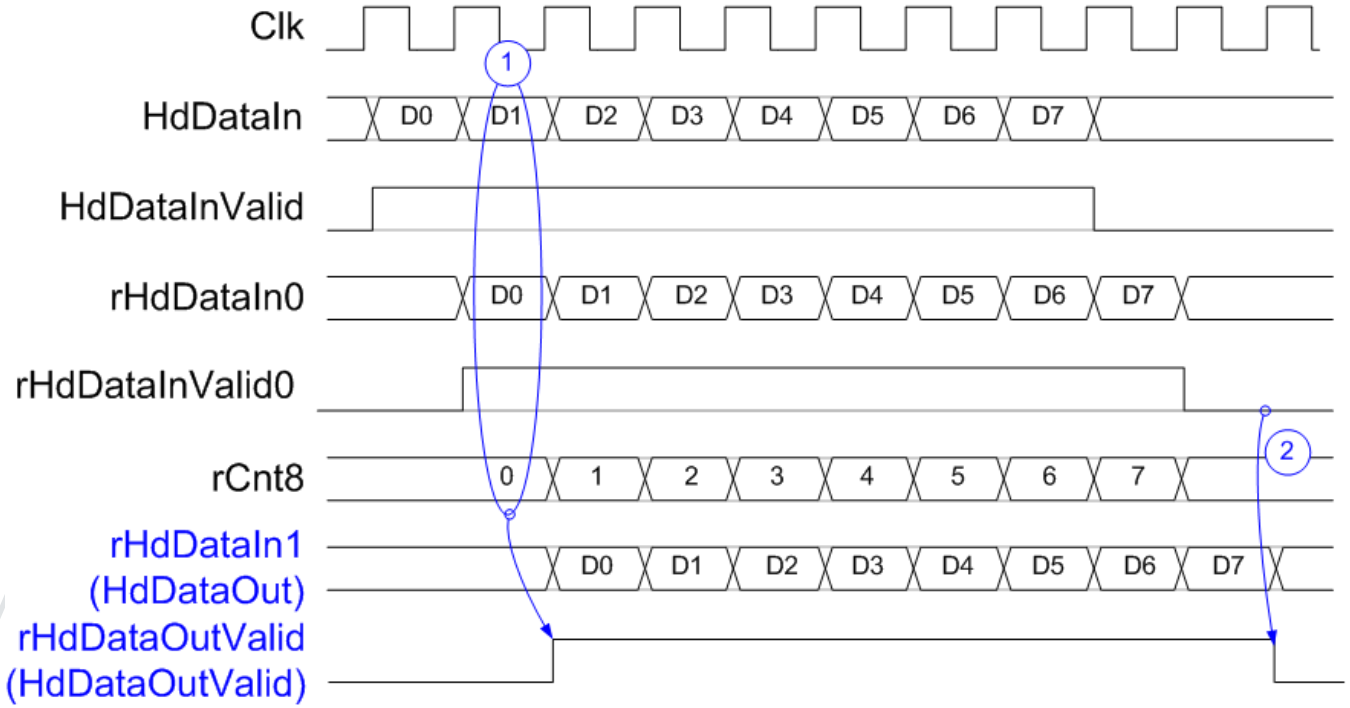
- เมื่อวิเคราะห์วงจร จะพบว่า period ในการทำงานของเรา จะยาวทั้งหมด 8 clock cycles ดังนั้นเราสามารถกำหนดช่วงจังหวะในการทำงานได้ โดยใช้วงจร counter นับ 0-7 ได้
- เพื่อให้จังหวะการทำงานสอดคล้อง เราจะ delay HdDataInValid ไปอีก 1 clock ทำให้ timing ที่ได้จะตรงกับสัญญาณ rHdDataIn0 และใช้สัญญาณนี้เป็นตัวให้จังหวะการนับ



เมื่อเราสร้าง rCnt8 ไว้แล้ว ซึ่งเป็นสัญญาณที่จะนับตามสัญญาณ rHdDataInValid0 เราจึงสามารถสร้างเงื่อนไขบอกจังหวะที่ 1 ได้ เพื่อตรวจสอบค่าของ D0=x"55" และ D1=x"AA" ได้ โดยการเปรียบเทียบ rCnt8=0 และ rHdDataInValid0='1'

A5.3: HOW TO GENERATE OUTPUT

เมื่อวงจรเราต้องทำงานตาม clock ดังนั้นเราจะทราบผลของการเปรียบเทียบ D0 และ D1 ว่ามีค่าถูกต้องหรือไม่ในจังหวะ clock ถัดไป ดังนั้นเราจะสามารถสร้างสัญญาณ HdDataOutValid ได้ในจังหวะ clock ถัดไป ทำให้สัญญาณข้อมูล HdDataOut ที่จะส่งออก ต้อง delay ออกไปเช่นกัน จึงต้องสร้างสัญญาณ HdDataOut มาจาก rHdDataIn1



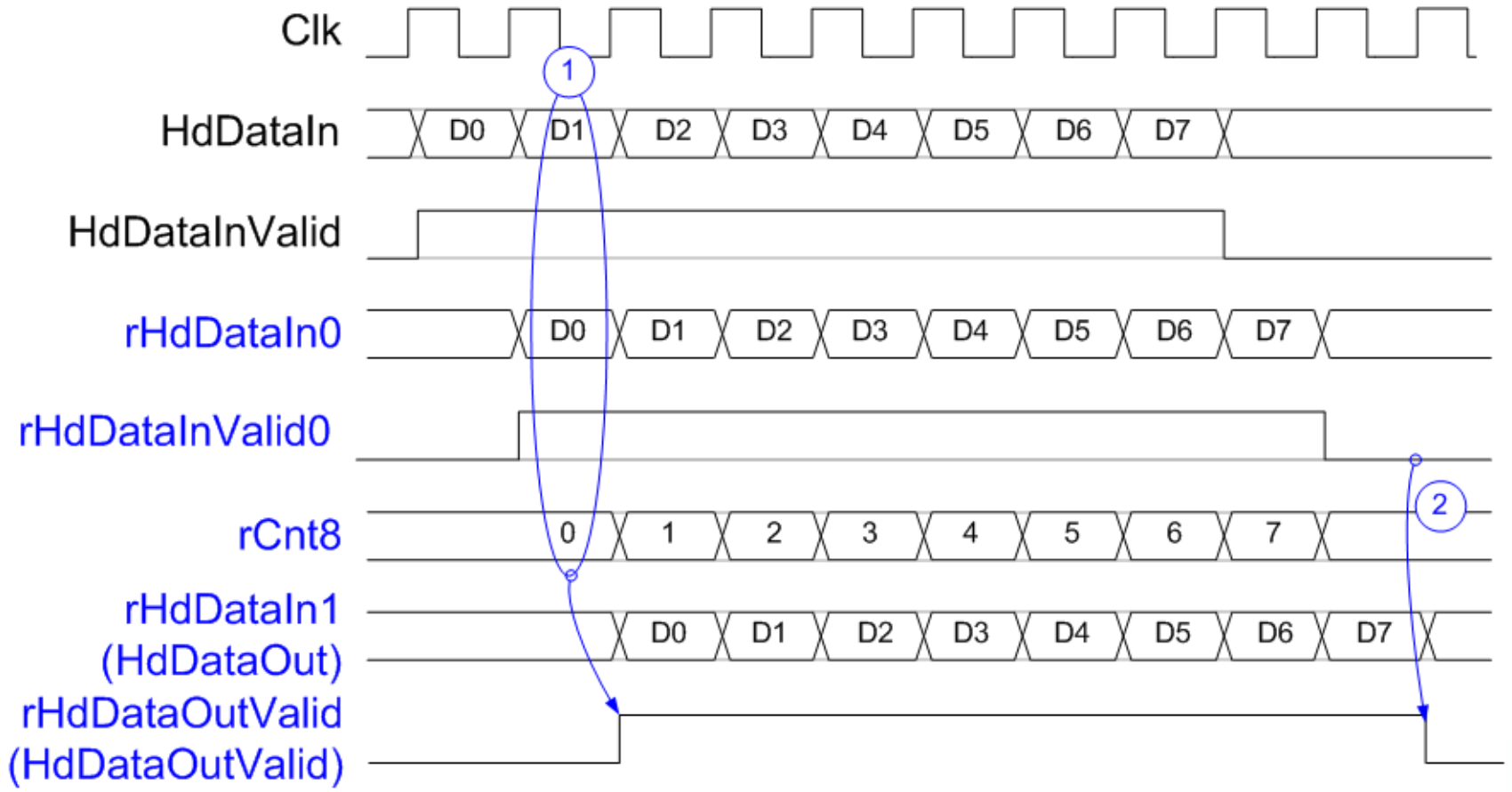
สัญญาณ rHdDataOutValid จะสร้างโดยใช้ Set-Clear FlipFlop โดยเงื่อนไข คือ

- Set to '1' เมื่อ rCnt8=0, rHdDataInValid0='1', rHdDataIn0=x"55" และ HdDataIn=x"AA"
- Clear to '0' เมื่อ rHdDataInValid0='0'

TIMING DIAGRAM TO LOGIC CODING

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

Q6: VHDL CODING FROM TIMING DIAGRAM



Q: ลองเขียน Code เพื่อสร้างสัญญาณสีฟ้าทั้งหมดจาก Input สีดำ

A6.1: BLOCK TEMPLATE CODING

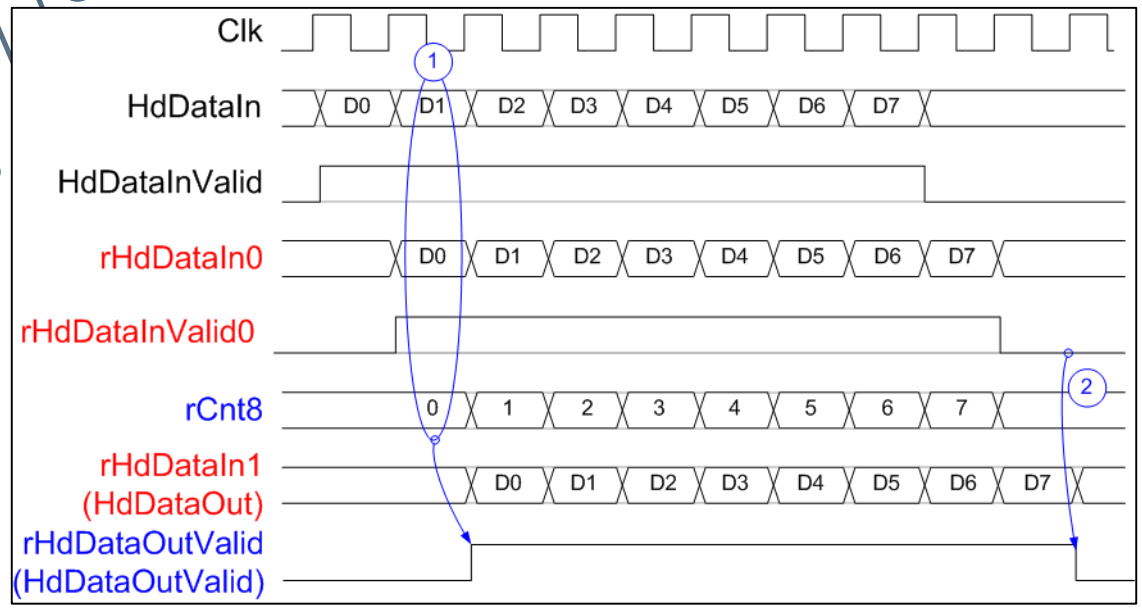
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

Entity Test1 Is
    Port
    (
        RstB            : in    std_logic;
        Clk             : in    std_logic;

        HdDataIn        : in    std_logic_vector( 7 downto 0 );
        HdDataInValid   : in    std_logic;

        HdDataOut       : out   std_logic_vector( 7 downto 0 );
        HdDataOutValid  : out   std_logic
    );
End Entity Test1;

Architecture rtl Of Test1 Is

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


Begin

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



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

End Architecture rtl;
```

1. ประกาศ library มาตรฐานที่ใช้งาน

2. ประกาศ input/output ของกล่อง พร้อมตั้งชื่อกล่อง (Test1)

3. ขึ้นโครงสร้างของ module โดยประกาศ Architecture, Begin, End Architecture

A6.2: CODING FOR DELAY SIGNAL



1. ประกาศสัญญาณที่จะใช้งาน

2. สร้างสัญญาณ data 2 ตัว ผ่าน DFF 2 ชั้น โดยไม่มี reset condition

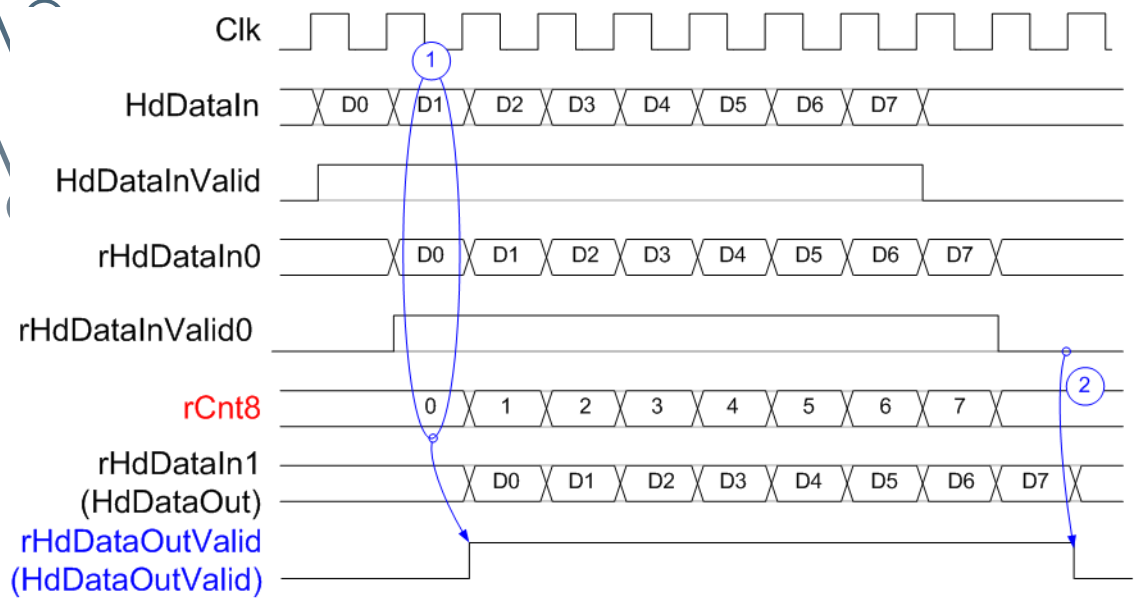
3. สร้างสัญญาณ valid ผ่าน DFF 1 ตัว โดยมี reset condition เพื่อไม่ให้เป็นค่า '1' ในช่วงที่วงจรยังไม่พร้อมทำงาน

1
```
signal  rHdDataIn0      : std_logic_vector( 7 downto 0 );
signal  rHdDataIn1      : std_logic_vector( 7 downto 0 );
signal  rHdDataInValid0 : std_logic;

Begin

-- Output assignment
```

2
```
-- Delay data
u_rHdDataIn0 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        rHdDataIn1  <= rHdDataIn0;
        rHdDataIn0  <= HdDataIn;
    end if;
End Process u_rHdDataIn0;
```

3
```
-- Delay valid
u_rHdDataInValid0 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        -- Control signal should have reset condition
        if ( RstB='0' ) then
            rHdDataInValid0 <= '0';
        else
            rHdDataInValid0 <= HdDataInValid;
        end if;
    end if;
End Process u_rHdDataInValid0;
```

A6.3: CODING FOR COUNTER




```
signal   rHdDataIn0       : std_logic_vector( 7 downto 0 );
signal   rHdDataIn1       : std_logic_vector( 7 downto 0 );
signal   rHdDataInValid0  : std_logic;
signal   rCnt8            : std_logic_vector( 2 downto 0 );
```

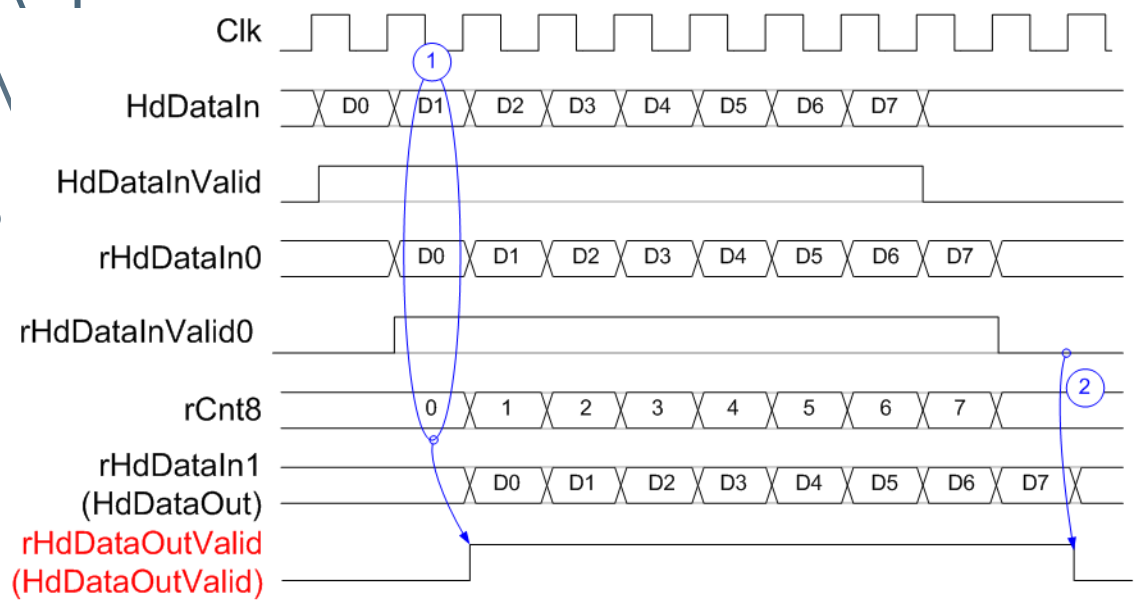
```
-------------------------------------------------------------------
-- DFF
-------------------------------------------------------------------

    -- Delay data
    u_rHdDataIn0 : Process (Clk) Is

    -- Delay valid
    u_rHdDataInValid0 : Process (Clk) Is
```

```
u_rCnt8 : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( rHddataInValid0='1' ) then
            rCnt8(2 downto 0)   <= rCnt8(2 downto 0) + 1;
        else
            rCnt8(2 downto 0)   <= "000";
        end if;
    end if;
End Process u_rCnt8;
```

ประกาศสัญญาณ rCnt8 เพื่อนับ 0-7 (3 bit)

และเขียน code counter ที่จะนับตามสัญญาณ rHdDataInValid0

A6.4: CODING FOR OUTPUT

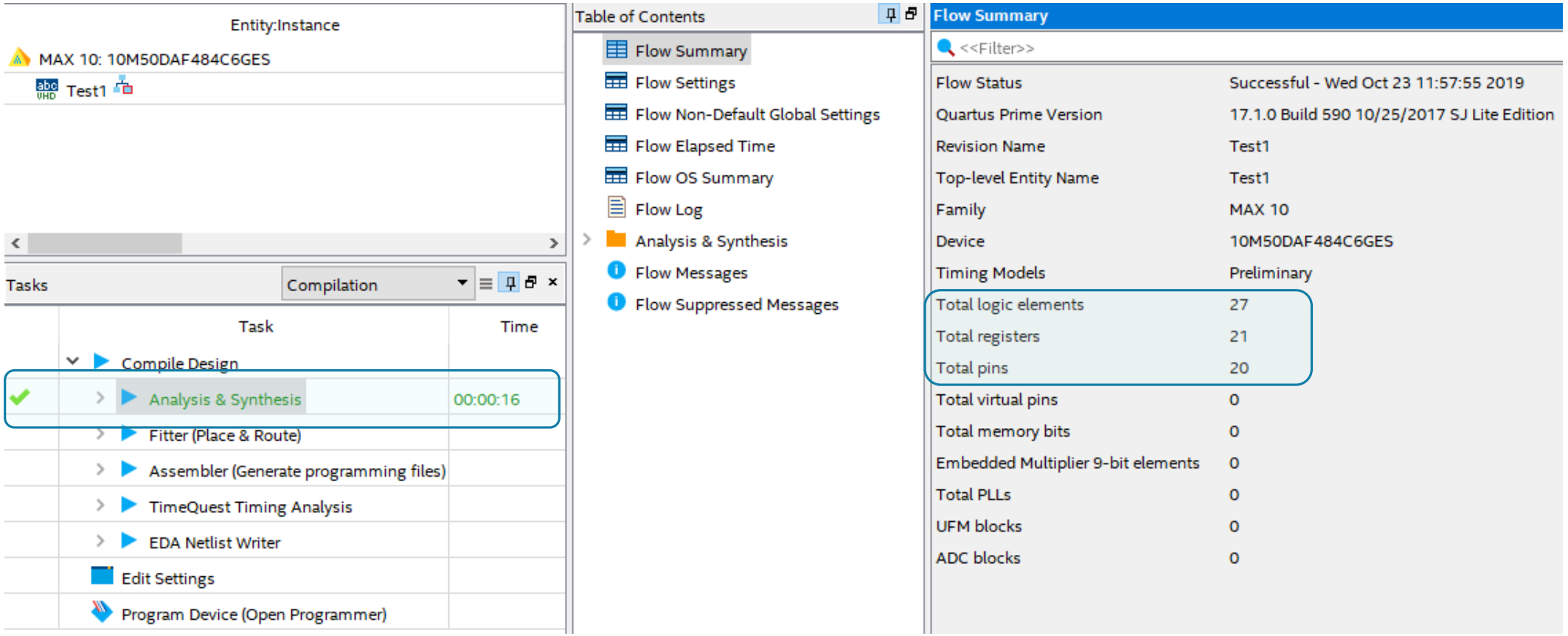


1. ประกาศสัญญาณ rHdDataOutValid และเขียน code เพื่อสร้าง set/clear Flipflop ตามเงื่อนไขที่ได้ออกแบบไว้ เนื่องจากเป็นสัญญาณ control ที่ต้องระวังค่าช่วงที่วงจรไม่พร้อมทำงาน จึงต้องใส่เงื่อนไข reset ไว้ด้วย

2. เขียน code เพื่อ assign output จากสัญญาณภายในที่สร้างไว้

```
    signal  rCnt8           : std_logic_vector( 2 downto 0 );
    signal  rHdDataOutValid : std_logic;
Begin

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

    HdDataOut       <= rHdDataIn1;
    HdDataOutValid  <= rHdDataOutValid;

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

    -- Delay data
    u_rHdDataIn0 : Process (Clk) Is

    -- Delay valid
    u_rHdDataInValid0 : Process (Clk) Is

    u_rCnt8 : Process (Clk) Is

    u_rHdDataOutValid : Process (Clk) Is
    Begin
        if ( rising_edge(Clk) ) then
            if ( RstB='0' ) then
                rHdDataOutValid <= '0';
            else
                -- Start of valid data streaming
                if ( rHdDataInValid0='1' and rCnt8(2 downto 0)=0 and
                     HdDataIn(7 downto 0)=x"AA" and rHdDataIn0=x"55" ) then
                    rHdDataOutValid     <= '1';
                -- End of data streaming
                elsif ( rHddataInValid0='0' ) then
                    rHdDataOutValid     <= '0';
                else
                    rHdDataOutValid     <= rHdDataOutValid;
                end if;
            end if;
        end if;
    End Process u_rHdDataOutValid;
```

A6.5: SYNTHESIS RESULT



Code จะสามารถ synthesis ได้สำเร็จ และมี report ของ LE/register ที่ใช้งานแสดงขึ้นมา