DIGITAL DESIGN WITH FPGA CAMP

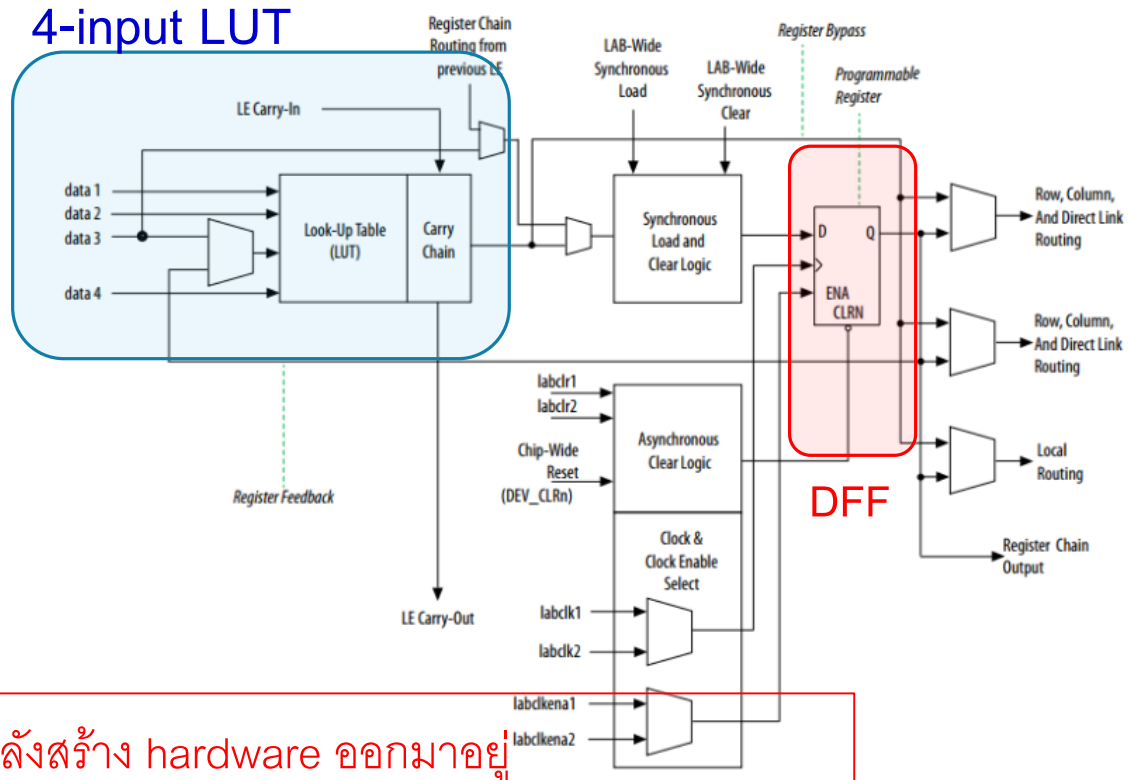
DAY 1 LOGIC SYNTHESIS

VHDL TO LOGIC

OVERVIEW

- VHDL เป็นภาษาที่ใช้สื่อสารกับ tool เพื่อให้ tool นั้นสังเคราะห์วงจร logic ออกมาตามที่เราคิดไว้
- Tool จะแปล code ของเราว่าการทำงานที่เราเขียนนั้นอ้างอิงถึงวงจรอะไรได้บ้าง แล้วสังเคราะห์วงจรออกมา
- บน FPGA ทุก logic จะสร้างโดยอยู่บนพื้นฐานของโครงสร้าง FPGA นั้นคือ การใช้ LE ที่ประกอบด้วย 4-input LUT และ D Flip Flop

Figure 5: LE High-Level Block Diagram for MAX 10 Devices.



*** ดังนั้นการเขียน code ทุกบรรทัด ควรจะต้องเตือนตัวเองว่า กำลังสร้าง hardware ออกมาอยู่

หาก code นั้นไม่ได้จำเป็นจริง ๆ นั่นก็หมายความว่า เรากำลังเสีย LE ที่มีอยู่อย่างจำกัด ไปโดยเปล่าประโยชน์ ***

Q1: VHDL TO LOGIC

```
    signal  rCnt    : std_logic_vector( 3 downto 0 );

Begin

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

    CntOut      <= rCnt;

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

    -- 4-bit increment counter
    u_rCnt : Process (Clk) Is
    Begin
        if ( rising_edge(Clk) ) then
            if ( RstB='0' ) then
                rCnt(3 downto 0)    <= (others=>'0');
            else
                -- Load start value
                if ( CntLd='1' ) then
                    rCnt(3 downto 0)    <= DataIn(3 downto 0);
                -- Counter enable
                elsif ( CntEn='1' ) then
                    rCnt(3 downto 0)    <= rCnt(3 downto 0) + 1;
                -- Hold same value
                else
                    rCnt(3 downto 0)    <= rCnt(3 downto 0);
                end if;
            end if;
        end if;
    End Process u_rCnt;
```

!!! ลองวิเคราะห์ดู !!!

สัญญาณ rCnt ขนาด 4-bit ตัวนี้ ประกอบด้วยวงจรอะไรบ้าง???

A1: VHDL TO LOGIC

```
-- 4-bit increment counter
u_rCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt(3 downto 0)     <= (others=>'0');
        else
            -- Load start value
            if ( CntLd='1' ) then
                rCnt(3 downto 0)     <= DataIn(3 downto 0);
            -- Counter enable
            elsif ( CntEn='1' ) then
                rCnt(3 downto 0)     <= rCnt(3 downto 0) + 1;
            -- Hold same value
            else
                rCnt(3 downto 0)     <= rCnt(3 downto 0);
            end if;
        end if;
    end if;
End Process u_rCnt;
```

1 สร้าง Flip Flop ขนาด 4 bit (เพราะทำงานภายใต้ Clk) และเป็น Flip Flop ที่สามารถ clear ค่าเป็น 0 ได้

2 วงจร Mux เพื่อเลือก input ให้โหลดค่า DataIn เข้าไป โดยมี CntLd เป็นสัญญาณสำหรับเลือก input

3 วงจรบวก และมี CntEn เป็นสัญญาณ input ของ Mux สำหรับเลือกโหลดค่า +1 เข้าไป

4 วงจร Latch ขนาด 4-bit สำหรับคงค่าเดิมไว้

*** code ทุกบรรทัด คือการสร้าง logic เพิ่มขึ้นมาทั้งหมด ***

Q2: LOGIC SYNTHESIS BY FPGA

```
-- 4-bit increment counter
u_rCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt(3 downto 0)    <= (others=>'0');
        else
            -- Load start value
            if ( CntLd='1' ) then
                rCnt(3 downto 0)    <= DataIn(3 downto 0);
            -- Counter enable
            elsif ( CntEn='1' ) then
                rCnt(3 downto 0)    <= rCnt(3 downto 0) + 1;
            -- Hold same value
            else
                rCnt(3 downto 0)    <= rCnt(3 downto 0);
            end if;
        end if;
    end if;
End Process u_rCnt;
```

!!! ลองวิเคราะห์ดู !!!

เมื่อนำไปสังเคราะห์ด้วย FPGA โดยใช้ LE ที่มี 4-input LUT และ DFF สำหรับสร้าง output 1 bit จะต้องใช้ LE ทั้งหมดเท่าไหร่

คำแนะนำ

1. แยกส่วนที่สังเคราะห์ด้วย LUT และ DFF ออกจากกันก่อน แล้วค่อยมาเล็งว่า DFF กับ LUT นั้นใช้ LE ตัวเดียวกันได้หรือไม่

2. การวิเคราะห์ LUT ให้ลองนับจำนวน input ที่ต้องใช้ในการสร้างสัญญาณแต่ละ bit ดู (1 LUT รองรับได้ 4 input)

A2: DRAFT ANALYSIS

```
-- 4-bit increment counter
u_rCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt(3 downto 0)    <= (others=>'0');
        else
            -- Load start value
            if ( CntLd='1' ) then
                rCnt(3 downto 0)    <= DataIn(3 downto 0);
            -- Counter enable
            elsif ( CntEn='1' ) then
                rCnt(3 downto 0)    <= rCnt(3 downto 0) + 1;
            -- Hold same value
            else
                rCnt(3 downto 0)    <= rCnt(3 downto 0);
            end if;
        end if;
    end if;
End Process u_rCnt;
```

วิเคราะห์จำนวน LUT ด้วยตัวเอง

การสร้างสัญญาณ rCnt(0) มี input ทั้งหมดกี่ตัว

RstB, ค่า ‘0’, CntLd, DataIn(0), CntEn, +1, rCnt(0)

เพื่อความง่าย เรามองการบวก 1 เป็น input อีกตัวก่อน

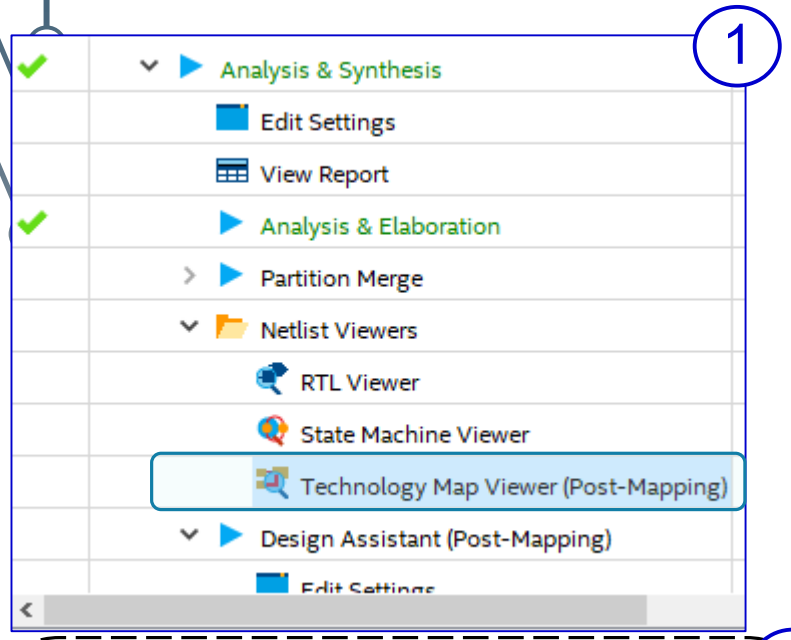
ดังนั้น จะมี input ที่ต้องใช้ในการสร้างทั้งหมด 7 ตัว

คำนวณได้ว่า ควรจะใช้ 4-input LUT อย่างน้อย 2 ตัวสำหรับการสร้าง rCnt0 เพราะ 1 ตัวรองรับ input ได้เพียง 4 input เท่านั้น

เมื่อมี 4 bit ก็ต้องใช้อย่างน้อย 8 ตัว

ประเมินขั้นต่ำ เพราะไม่ได้ประเมินในส่วนวงจรบวกเข้าไป

A2: LOGIC SYNTHESIS BY QUARTUSII



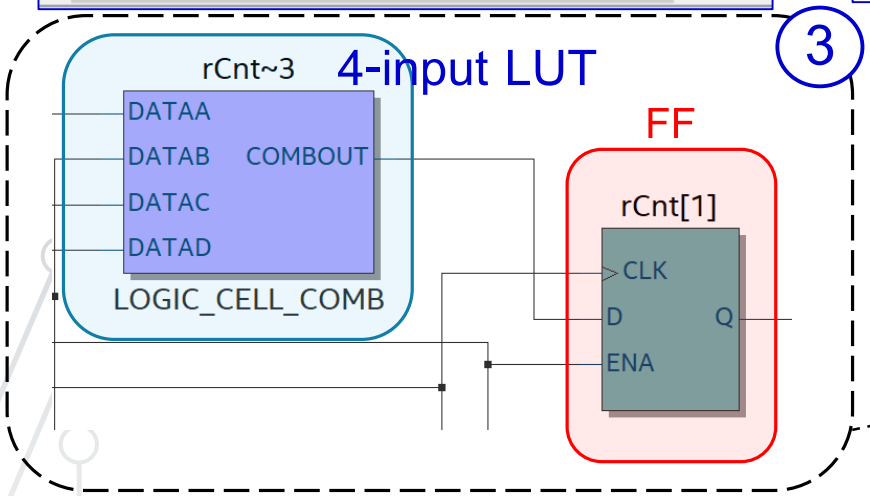
ตรวจสอบผลที่ได้โดยใช้ QuartusII

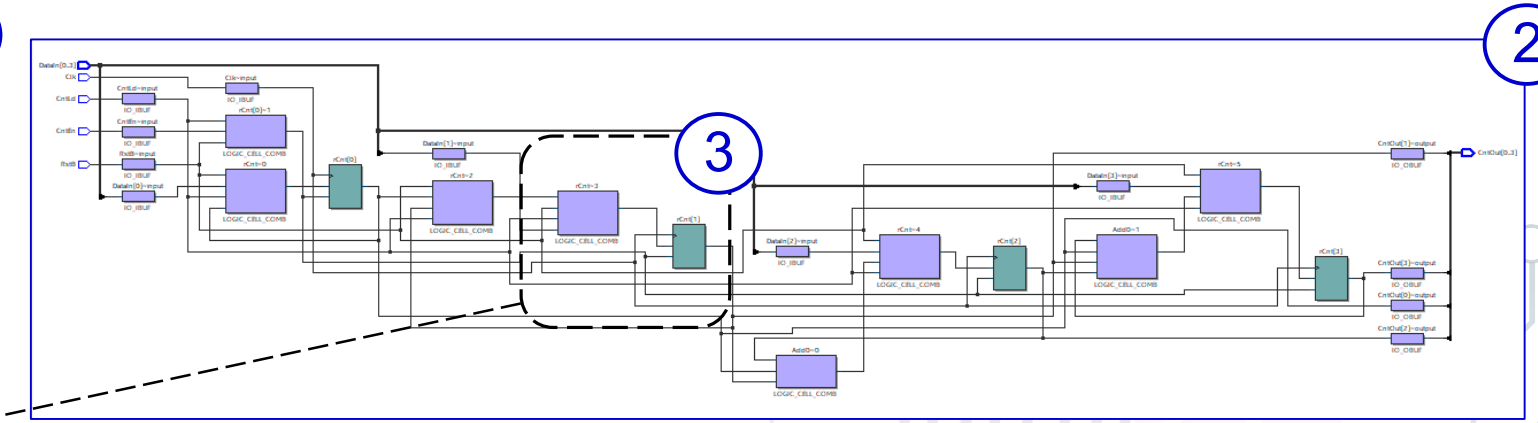
Analysis & Synthesis -> Netlist Viewers -> Technology Map Viewer

จากรูปด้านล่าง สีเขียวคือ Register (FF) สีม่วงอันใหญ่เป็นส่วนของ COMB (LUT)

นับได้ 8 LUT และ 4 FF จะใช้ทั้งหมด 8 LE (4 LE ใช้แต่ LUT อีก 4 LE ใช้ LUT + FF)

รูปที่ 3 แสดงตัวอย่างของการสร้างสัญญาณ rCnt ด้วย LUT และ FF โดยละเอียด





LOGIC OPTIMIZATION

Q1: BASIC COUNTER WITH FIXED PERIOD

คำอธิบาย: code ทั้งแบบที่ 1 และแบบที่ 2 ต่างก็เป็น counter พื้นฐานที่จะใช้สร้างสัญญาณ pulse ขึ้นมาได้ในทุก ๆ MaxVal clock เช่น MaxVal=5 ก็จะสร้าง Counter นับ 0-4 (สำหรับวงจรด้านซ้าย) หรือนับ 1-5 (สำหรับวงจรด้านขวา) เพื่อจะนำ counter นี้ไป compare ค่าแล้วสร้าง pulse ออกมาทุก ๆ 5 clocks นั่นเอง

1

```
-- 4-bit increment counter
-- Count from 0 to (Maxval-1)
u_rCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt(3 downto 0)    <= "0000";
        else
            if ( rCnt(3 downto 0)=(Maxval(3 downto 0)-1) ) then
                rCnt(3 downto 0)    <= "0000";
            else
                rCnt(3 downto 0)    <= rCnt(3 downto 0) + 1;
            end if;
        end if;
    end if;
End Process u_rCnt;
```

2

```
-- 4-bit increment counter
-- Count from 1 to Maxval
u_rCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt(3 downto 0)    <= "0001";
        else
            if ( rCnt(3 downto 0)=Maxval(3 downto 0) ) then
                rCnt(3 downto 0)    <= "0001";
            else
                rCnt(3 downto 0)    <= rCnt(3 downto 0) + 1;
            end if;
        end if;
    end if;
End Process u_rCnt;
```

Q1 : Code ทั้งสองแบบนี้ สามารถสร้าง pulse ได้เช่นเดียวกัน

คิดว่าวงจรทั้งสองนี้ ใช้ LE เท่ากันหรือไม่ ถ้าไม่เท่า แบบไหนใช้น้อยกว่ากัน???

A1: BASIC COUNTER WITH FIXED PERIOD

```vhdl
-- 4-bit increment counter
-- Count from 0 to (Maxval-1)
u_rCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt(3 downto 0)    <= "0000";
        else
            if ( rCnt(3 downto 0)=(Maxval(3 downto 0)-1) ) then
                rCnt(3 downto 0)    <= "0000";
            else
                rCnt(3 downto 0)    <= rCnt(3 downto 0) + 1;
            end if;
        end if;
    end if;
End Process u_rCnt;
```

1

Total logic elements	11
Total registers	4

```vhdl
-- 4-bit increment counter
-- Count from 1 to Maxval
u_rCnt : Process (Clk) Is
Begin
    if ( rising_edge(Clk) ) then
        if ( RstB='0' ) then
            rCnt(3 downto 0)    <= "0001";
        else
            if ( rCnt(3 downto 0)=Maxval(3 downto 0) ) then
                rCnt(3 downto 0)    <= "0001";
            else
                rCnt(3 downto 0)    <= rCnt(3 downto 0) + 1;
            end if;
        end if;
    end if;
End Process u_rCnt;
```

2

Total logic elements	9
Total registers	4

A: แบบซ้ายจะใช้ LE มากกว่า เนื่องจากต้องนำ LE ส่วนหนึ่งมาสร้างวงจรลบ จากการเขียนเปรียบเทียบ MaxVal(3 downto 0)-1

ย้ำเตือนอีกครั้งว่า code ทุกบรรทัดของ VHDL คือการสร้าง hardware ขึ้นมา ดังนั้นผู้เขียน code ควรจะมีความรู้พื้นฐานที่จะ analyze ตัว code ด้วยว่า สามารถเขียนให้ใช้ resource ได้เหมาะสมแล้วหรือยัง

Q2: COUNTER 1-128

คำอธิบาย: code ทั้งแบบที่ 1 และแบบที่ 2 ต่างก็เป็น counter สำหรับนับเลขตั้งแต่ 1-128 เหมือนกัน

1

```
-- 8-bit increment counter
-- Count from 1 to 128
u_rCnt : Process (clk) Is
Begin
   if ( rising_edge(clk) ) then
      if ( RstB='0' ) then
         rCnt(7 downto 0)  <= x"01";
      else
         if ( rCnt(7 downto 0)=128 ) then
            rCnt(7 downto 0)  <= x"01";
         else
            rCnt(7 downto 0)  <= rCnt(7 downto 0) + 1;
         end if;
      end if;
   end if;
End Process u_rCnt;
```

2

```
-- 8-bit increment counter
-- Count from 1 to 128
u_rCnt : Process (clk) Is
Begin
   if ( rising_edge(clk) ) then
      if ( RstB='0' ) then
         rCnt(7 downto 0)  <= x"01";
      else
         if ( rCnt(7)='1' ) then
            rCnt(7 downto 0)  <= x"01";
         else
            rCnt(7 downto 0)  <= rCnt(7 downto 0) + 1;
         end if;
      end if;
   end if;
End Process u_rCnt;
```

Q2 : Code ทั้งสองแบบนี้ ใช้ LE เท่ากันหรือไม่ ถ้าไม่เท่า แบบไหนใช้น้อยกว่ากัน???

A2: COUNTER 1-128

```
-- 8-bit increment counter
-- Count from 1 to 128
u_rCnt : Process (Clk) Is
Begin
   if ( rising_edge(Clk) ) then
      if ( RstB='0' ) then
         rCnt(7 downto 0)  <= x"01";
      else
         if ( rCnt(7 downto 0)=128 ) then
            rCnt(7 downto 0)  <= x"01";
         else
            rCnt(7 downto 0)  <= rCnt(7 downto 0) + 1;
         end if;
      end if;
   end if;
End Process u_rCnt;
```

1

Total logic elements	12
Total registers	8

```
-- 8-bit increment counter
-- Count from 1 to 128
u_rCnt : Process (Clk) Is
Begin
   if ( rising_edge(Clk) ) then
      if ( RstB='0' ) then
         rCnt(7 downto 0)  <= x"01";
      else
         if ( rCnt(7)='1' ) then
            rCnt(7 downto 0)  <= x"01";
         else
            rCnt(7 downto 0)  <= rCnt(7 downto 0) + 1;
         end if;
      end if;
   end if;
End Process u_rCnt;
```

2

Total logic elements	10
Total registers	8

A: แบบซ้ายจะใช้ LE มากกว่า เนื่องจากตอน compare ค่า 128 นั้น สร้างวงจร comparator ขนาด 8 bit ขึ้นมา ในขณะที่วงจรด้านขวามือนั้น ทราบว่า สัญญาณนั้นจะมีค่า 1 – 128 เท่านั้น และค่า 128 เป็นค่าเดียวที่ bit7 จะมีค่าเป็น ‘1’ ดังนั้น เราสามารถเปรียบเทียบด้วย bit7 ตัวเดียว เพื่อแทนค่า 128 ได้เลย ขนาดของ comparator ที่ต้องใช้งานจึงเหลือเพียง 1 bit เท่านั้น

code ทุกบรรทัดของ VHDL คือการสร้าง hardware ขึ้นมา

ดังนั้นผู้เขียน code ควรจะมีความรู้พื้นฐานที่จะ analyze ตัว code ด้วยว่า

สามารถเขียนให้ใช้ resource ได้เหมาะสมแล้วหรือยัง

ความรู้เพิ่มเติม

3.3 **ตัวอย่างการสังเคราะห์วงจร**

https://forfpgadesign.wordpress.com/2017/11/16/hdlcoderesource1/

9.1 **ตัวอย่างเทคนิคการออกแบบ** http://bit.ly/fpgatech1

9.2 **ปัญหาชวนคิด** (1) https://forfpgadesign.wordpress.com/2017/12/02/9-2-know-1/

9.3 **ปัญหาชวนคิด** (2) https://forfpgadesign.wordpress.com/2017/12/03/9-3-know2/