บทที่ 5
การออกแบบ

5.1 บทนำ

ในบทนี้จะเป็นการนำเสนอรายละเอียดขั้นตอนการออกแบบวงจรครอสโอเวอร์เน็ตเวอร์คแบบรีเคอร์ซีฟ ต้นแบบเป็นวงจรกรองทุกย่านความถี่ผ่านต่อขนานกัน ในขั้นตอนการออกแบบวงจรนี้จะเริ่มจากการหาฟังก์ชันถ่ายโอนของวงจรกรองต้นแบบที่ออกแบบไว้ก่อน โดยจะเริ่มตั้งแต่การกำหนดอันดับ (Order) ความถี่ตัด (cutoff frequency) ความถี่ในการสุ่ม (sampling frequency) และในรายละเอียดอื่นๆ เช่น ความถี่ที่ขอบแถบผ่าน (pass band edge frequency) ความถี่ที่ขอบแถบหยุด (stop band edge frequency) การกระเพื่อมในแถบผ่าน (pass band ripple) และการลดทอนในแถบหยุด (stop band attenuation) โดยใช้การประมาณค่าด้วยฟังก์ชันอิลลิปติก เพื่อคำนวณหาค่าสัมประสิทธิ์ของฟังก์ชันถ่ายโอนออกมาโดยอยู่ในโดเมนดิจิตอล ซึ่งเหมาะสมสำหรับการคำนวณและการนำไปสร้างบน FPGA

ค่าสัมประสิทธิ์ที่ได้จากการคำนวณจะถูกทำการควอนไตซ์ (Quantization) เพื่อจัดข้อมูลของค่าสัมประสิทธิ์ให้มีการแทนตัวเลขโดยตรงได้ (fix-point) โดยกำหนดให้มีขนาดของข้อมูลเท่ากับ 16 บิต ในขั้นตอนการทำงานต่างๆจะใช้โปรแกรม MATLAB ในการคำนวณหาค่าสัมประสิทธิ์ออกมาจากนั้นจึงนำค่าสัมประสิทธิ์ที่ได้ไปสร้างไฟล์ตารางเปิดดูในรูปแบบของภาษา Verilog HDL ด้วยภาษาซี ซึ่งลดความยุ่งยากในการคำนวณ จากนั้นจึงทำการออกแบบตามโครงสร้างฮาร์ดแวร์ของวงจรครอสโอเวอร์บน FPGA ซึ่งออกแบบและพัฒนาด้วยโปรแกรม Xilinx ISE ซึ่งขั้นตอนในการออกแบบแสดงดังรูปที่ 5.1

รูปที่ 5.1 ขั้นตอนการออกแบบวงจรครอสโอเวอร์เน็ตเวอร์คบน FPGA

5.2 การออกแบบวงจรครอสโอเวอร์ด้วยวงจรกรองทุกย่านความถี่ผ่านต่อขนานกัน

ในขั้นตอนการสร้างวงจรครอสโอเวอร์เน็ตเวอร์คแบบสองทางจะเริ่มต้นออกแบบด้วยการกำหนดค่าคุณสมบัติเริ่มต้นของวงจรขึ้นมาก่อนซึ่งเป็นวงจรกรองความถี่ต่ำโดยมีคุณสมบัติของวงจรต้นแบบเป็นดังนี้

วงจรกรองต้นแบบ	อิลลิปติก อันดับ 3		
ความถี่ในการสุ่ม		48	KHz
แถบความถี่ขอบผ่าน ω_p		3	KHz
แถบความถี่ขอบหยุด ω_s		13	KHz
การกระเพื่อมในแถบผ่าน δ_p		0.002	dB
การลดทอนในแถบหยุด δ_s		35	dB

จากข้อกำหนดดังกล่าวสามารถหาฟังก์ชันถ่ายโอนได้เป็น

$$H(Z)=\frac{0.0650659+0.0951390\cdot Z^{-1}+0.0951390\cdot Z^{-2}+0.0650659\cdot Z^{-3}}{1-1.3724767\cdot Z^{-1}+0.8860779\cdot Z^{-2}-0.1931912\cdot Z^{-3}} \tag{5.1}$$

หลังจากได้ฟังก์ชันถ่ายโอนต้นแบบจากสมการที่ (5.1) สามารถแยกให้อยู่ในรูปแบบของผลรวมและผลต่างของสองฟังก์ชันถ่ายโอนด้วยวิธี Pole interacting prototype จะได้วงจรกรองทุกย่านความถี่ผ่านสองสมการเป็น

$$A_0(Z)=\frac{-0.3792592+Z^{-1}}{1--0.3792592\cdot Z^{-1}} \tag{5.2}$$

และ

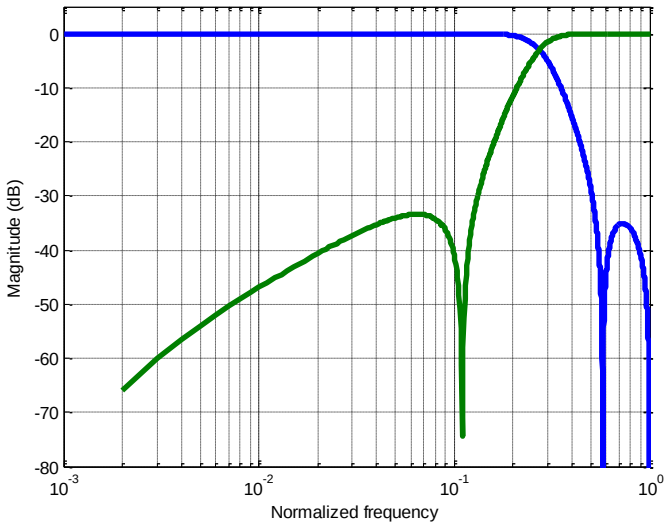
$$A_1(Z)=\frac{0.5093910-0.9932175\cdot Z^{-1}+Z^{-2}}{1-0.9932175\cdot Z^{-1}+0.5093910\cdot Z^{-2}} \tag{5.3}$$

ซึ่งฟังก์ชันถ่ายโอน $A_0(Z)$ และ $A_1(Z)$ เป็นฟังก์ชันถ่ายโอนที่เรียกว่า All-Pass function ที่สามารถนำกลับมาสร้างเป็นวงจรกรองความถี่ต่ำต้นแบบและวงจรกรองความถี่สูงได้ตามสมการที่ (5.4) และ (5.5) ตามลำดับ

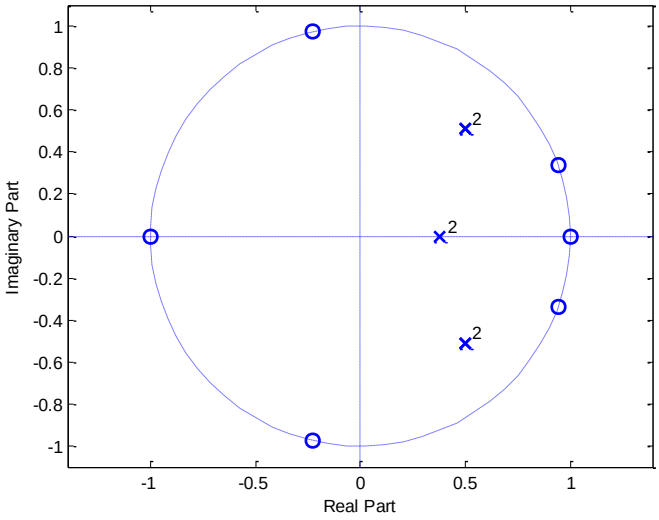
$$H_{LPF}(Z)=\frac{A_0(Z)+A_1(Z)}{2} \tag{5.4}$$

$$H_{HPF}(Z)=\frac{A_0(Z)-A_1(Z)}{2} \tag{5.5}$$

จากสมการที่ (5.4) และ (5.5) สามารถพล๊อตผลตอบสนองความถี่ เส้นทึบเป็นผลตอบสนองความถี่ของวงจรกรองความถี่ต่ำจากสมการที่ (5.4) และเส้นบางเป็นผลตอบสนองความถี่สูงจากสมการที่ (5.5) ซึ่งเรียกว่า Mirror filter [26] โดยมีตำแหน่งของโพลของทั้งสองวงจรอยู่ที่จุดเดียวกัน ซึ่งจะมีคุณลักษณะตรงกันข้ามกับวงจรกรองต้นแบบ และเมื่อนำผลตอบสนองความถี่ทั้งสองมารวมกันจะได้ผลรวมมีค่าคงที่ดังรูปที่ 5.2 และ 5.3

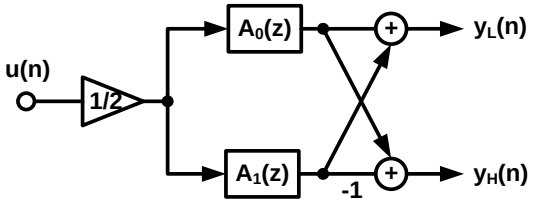


รูปที่ 5.2 ผลตอบสนองความถี่จากสมการที่ (5.4) และ (5.5)



รูปที่ 5.3 ตำแหน่งโพลและซีโร ของฟังก์ชันถ่ายโอนจากสมการที่ (5.4) และ (5.5)

เมื่อได้ค่าฟังก์ชันถ่ายโอน $A_0(Z)$ และ $A_1(Z)$ เป็นฟังก์ชันถ่ายโอนที่เป็นวงจรกรองทุกย่านความถี่ผ่านมาทำการสร้างซึ่งจะมีความซับซ้อนน้อยกว่าวิธีการที่สร้างโดยตรงโดยมีโครงสร้างดังรูปที่ 5.4



รูปที่ 5.4 โครงสร้างของวงจรครอสโอเวอร์ที่สร้างจากวงจรกรองทุกย่านความถี่ผ่านต่อขนานกัน

จากรูปที่ 5.4 สามารถสร้างวงจรกรอง $A_0(Z)$ และ $A_1(Z)$ ด้วยโครงสร้างคณิตศาสตร์การกระจายโดยนำค่าสัมประสิทธิ์จากสมการ 5.2 และ 5.3 มาสร้างตารางเปิดดูจะได้เป็น

ตารางที่ 5.1 ค่าสัมประสิทธิ์ของตารางเปิดดูจากฟังก์ชันถ่ายโอนจากสมการที่ (5.2)

X(n)	X(n-1)	Y(n-1)	Output
0	0	0	0
0	0	1	$- b_0$
0	1	0	1
0	1	1	$1 - b_0$
1	0	0	b_0
1	0	1	0
1	1	0	$1 + b_0$
1	1	1	1

ตารางที่ 5.2 ค่าสัมประสิทธิ์ของตารางเปิดดูจากฟังก์ชันถ่ายโอนจากสมการที่ (5.3)

X(n)	X(n-1)	X(n-2)	Y(n-2)	Y(n-1)	Output
0	0	0	0	0	0
0	0	0	0	1	$- b_1$
0	0	0	1	0	$- b_0$
0	0	0	1	1	$- (b_1 + b_0)$
0	0	1	0	0	1
0	0	1	0	1	$1 - b_1$
0	0	1	1	0	$1 - b_0$
0	0	1	1	1	$1 - (b_1 + b_0)$

ตารางที่ 5.2 ต่อ

0	1	0	0	0	b_1
0	1	0	0	1	0
0	1	0	1	0	$b_1 - b_0$
0	1	0	1	1	$- b_0$
0	1	1	0	0	$1 + b_1$
0	1	1	0	1	1
0	1	1	1	0	$1 + b_1 - b_0$
0	1	1	1	1	$1 - b_0$
1	0	0	0	0	B_0
1	0	0	0	1	$b_0 - b_1$
1	0	0	1	0	0
1	0	0	1	1	$- b_1$
1	0	1	0	0	$1 + b_0$
1	0	1	0	1	$1 + b_0 - b_1$
1	0	1	1	0	1
1	0	1	1	1	$1 - b_1$
1	1	0	0	0	$b_0 + b_1$
1	1	0	0	1	b_0
1	1	0	1	0	b_1
1	1	0	1	1	0
1	1	1	0	0	$1 + b_0 + b_1$
1	1	1	0	1	$1 + b_0$
1	1	1	1	0	$1 + b_1$
1	1	1	1	1	1

จากตารางที่ 5.1 และ 5.2 สามารถนำไปสร้างไฟล์ตารางเปิดดูที่ใช้งานบน FPGA โดยใช้ขนาดในการเก็บข้อมูล 16 บิต ใช้พื้นที่หน่วยความจำ 8 ตำแหน่งและ 32 ตำแหน่ง ซึ่งหากออกแบบด้วยโครงสร้างแบบตรง (Direct form II) จะใช้พื้นหน่วยความจำมากถึง 128 ตำแหน่ง ซึ่งมากกว่าหลายเท่าตัว ดังแสดงในตารางที่ 5.3 เป็นการเปรียบเทียบการใช้พื้นที่ในการออกแบบตารางเปิดดูของวงจรกรองที่ออกแบบด้วยโครงสร้างแบบต่างๆจะเห็นว่าหากออกแบบครอสโอเวอร์แบบสองทางซึ่งเป็นวงจรกรอง 2 ชุด โครงสร้างของ Direct form II และแบบ FIR จะใช้พื้นที่ของตารางเปิดดูเพิ่มขึ้นเป็นเท่าตัวในขณะที่โครงสร้างแบบ Two parallel All-pass ใช้พื้นที่เท่าเดิม

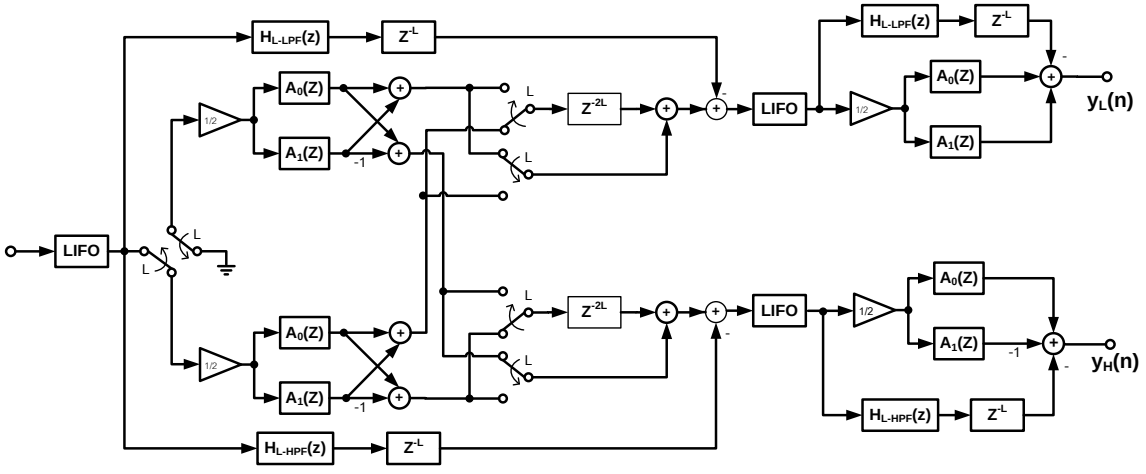
ตารางที่ 5.3 พื้นที่ในการออกแบบตารางเปิดดูของวงจรกรองที่ออกแบบด้วยโครงสร้างแบบต่างๆ

Type	Filter	2 Band Crossover
3rd Order Direct form II	128	256
16 Tab FIR (REMEZ algorithm)*	64K	128K
Two parallel All-pass filter	40	40

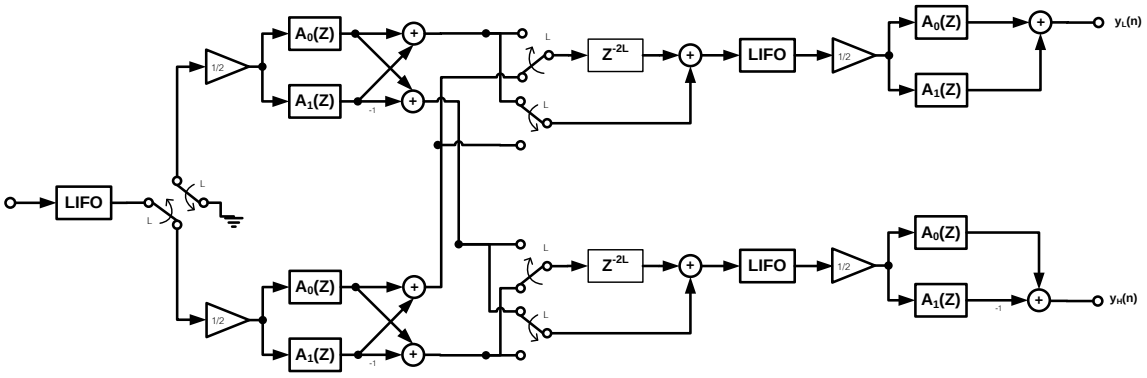
* เป็นการประมาณคุณสมบัติของวงจรกรองต้นแบบที่เป็น IIR และมีการออกแบบโดยตรงไม่ได้ใช้เทคนิคลดพื้นที่หน่วยความจำ

5.3 โครงสร้างวงจรครอสโอเวอร์แบบสองทางที่ให้เฟสเชิงเส้นสมบูรณ์บน FPGA

ในบทที่ผ่านมาได้กล่าวถึงการออกแบบวงจรครอสโอเวอร์เน็ตเวอร์คที่ให้เฟสเชิงเส้นสมบูรณ์ แต่ในการสร้างจริงบน FPGA นั้นได้ทำการตัดวงจรกรองเรซิดิวส์ ออกไปเนื่องจากมีผลกระทบต่อการทำงานน้อยมากเนื่องจากผลการตอบสนองของสัญญาณอิมพัลซ์ มีความยาวน้อยกว่าความยาวในการพลิกกลับสัญญาณของวงจรกลับสัญญาณจึงมีผลต่อระบบโดยรวมน้อยมากโดยแสดงดังรูปที่ 5.5 เป็นระบบที่สมบูรณ์ของวงจรครอสโอเวอร์เน็ตเวอร์ที่ให้เฟสเชิงเส้นสมบูรณ์ และรูปที่ 5.6 เป็นวงจรครอสโอเวอร์เน็ตเวอร์คที่นำไปสร้างบน FPGA



รูปที่ 5.5 วงจรครอสโอเวอร์เน็ตเวอร์คที่ให้เฟสเชิงเส้นสมบูรณ์



รูปที่ 5.6 โครงสร้างวงจรครอสโอเวอร์เน็ตเวอร์คแบบสองทางที่นำไปสร้างบน FPGA

จากโครงสร้างของวงจรครอสโอเวอร์เน็ตเวอร์คดังรูปที่ 5.6 ซึ่งเป็นโครงสร้างที่ให้ผลตอบสนองทางเฟสเชิงเส้นดังได้กล่าวในบทที่ผ่านมาแล้วนั้น สามารถแบ่งออกเป็นภาคต่างๆ เพื่อง่ายต่อการออกแบบบน FPGA อันได้แก่

โมดูลสวิทช์มัลติเพล็กซ์ (Multiplex switch)

โมดูลกลับสัญญาณในเวลาจริง (Real-time time reversal)

โมดูลหน่วงเวลา (Delay)

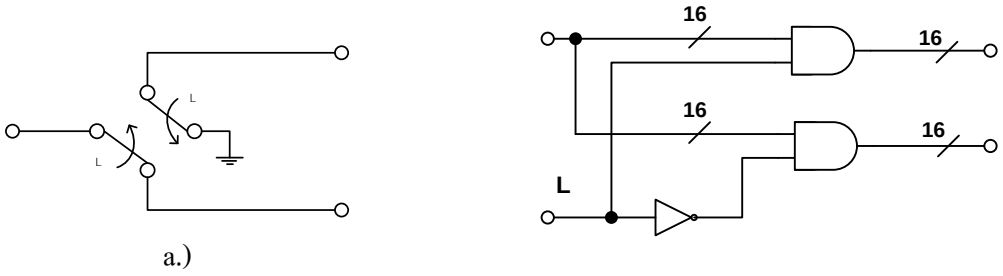
โมดูลกรองความถี่จากโครงสร้างแบบคณิตศาสตร์การกระจาย

5.4 การออกแบบวงจรสวิทช์มัลติเพล็กซ์

จากรูปที่ 5.6 วงจรสวิทช์จะมีอยู่สองส่วนได้แก่ ส่วนแรกหลังจากผ่านวงจรกลับสัญญาณในเวลาจริง (LIFO) และส่วนที่รวมสัญญาณจากวงจรกรองตัวบนและตัวล่าง

5.4.1 การออกแบบวงจรสวิทช์มัลติเพล็กซ์ส่วนที่ 1 (Multiplex switch)

วงจรสวิทช์ส่วนแรกจะเป็นการเลือกสลับสัญญาณให้กับวงจรกรองความถี่ชุดบนหรือชุดล่าง โดยมีจังหวะสลับการทำงานกันทุกๆ L แซมเปิ้ล โดยวงจรกรองที่ไม่ได้สัญญาณจากวงจรกลับสัญญาณก็จะถูกต่อเข้ากับข้อมูล “0” ซึ่งเสมือนต่อลงกราวน์ใน ซึ่งในวิทยานิพนธ์นี้ออกแบบให้มีการประมวลผลขนาด 16 บิต ดังนั้นบัสของวงจรสวิทช์ชุดแรกนี้จะเท่ากับ 16 บิต มีสัญลักษณ์และวงจรดังรูปที่ 5.7



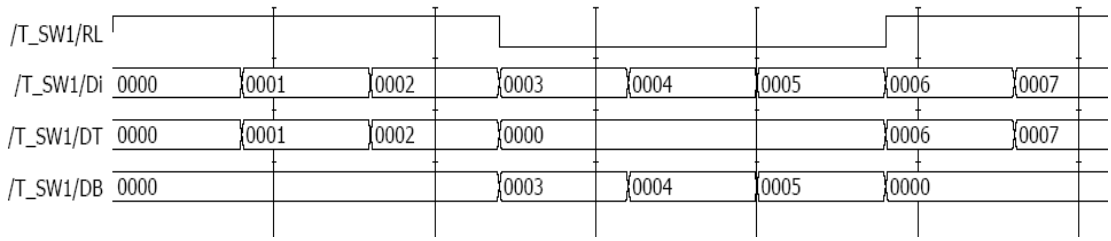
รูปที่ 5.7 สัญลักษณ์ (รูป a) และโครงสร้าง (รูป b) ของวงจรสวิทช์ส่วนที่ 1

จากวงจรในรูปที่ 5.7(b) สามารถนำไปเขียนด้วยภาษาอธิบายฮาร์ดแวร์โดยกำหนดอินพุตและเอาต์พุตขนาด 16 บิต โดยมี RL เป็นสัญญาณควบคุมการมัลติเพล็กซ์ดังนี้

```
module SW1(RL,Di,DT,DB);
   input RL;                  // Multiplex control
   input [15:0] Di;           //  Data input
   output [15:0] DT;          //  Data output to top Filter
   output [15:0] DB;          //   Data output to Bottom Filter

   assign DB[15:0] = Di[15:0] & {16{!RL}};
   assign DT[15:0] = Di[15:0] & {16{RL}};
endmodule
```

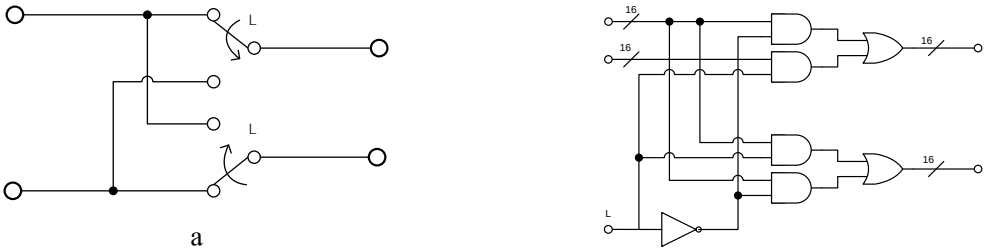
การจำลองการทำงานโดยกำหนดให้ข้อมูลอินพุต Di ที่ถูกควบคุมโดยสัญญาณ RL ซึ่งจะได้ผลจำลองการทำงานดังรูปที่ 5.8 โดยจะเห็นได้ว่าข้อมูลจะถูกส่งไปยังเอาต์พุต DT และ DB สลับกัน



รูปที่ 5.8 ผลการจำลองการทำงานวงจรสวิทช์ส่วนที่ 1

5.4.2 การออกแบบวงจรสวิทช์มัลติเพล็กซ์ส่วนที่ 2 (Multiplex switch)

วงจรสวิทช์ส่วนที่ 2 ทำหน้าที่สลับสัญญาณที่ได้จากวงจรกรองความถี่ด้านบนและด้านล่างโดยที่มีจังหวะสลับการทำงานทุกๆ L แซมเปิ้ลโดยมีขนาดข้อมูลเท่ากับ 16 บิตซึ่งมีสัญลักษณ์และวงจรดังรูปที่ 5.9



a b

รูปที่ 5.9 สัญลักษณ์ (รูป a) และโครงสร้าง (รูป b) ของวงจรสวิทช์ส่วนที่ 2

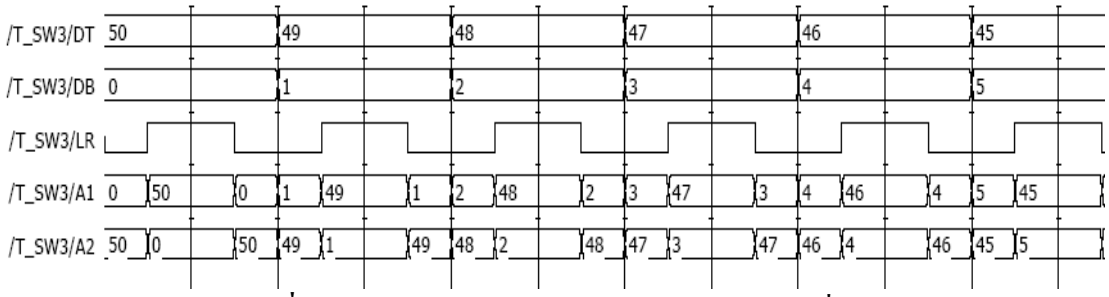
จากวงจรในรูปที่ 5.9(b) สามารถนำไปเขียนด้วยภาษาอธิบายฮาร์ดแวร์โดยกำหนดอินพุตและเอาต์พุตขนาด 16 บิต โดยมี RL เป็นสัญญาณควบคุมการมัลติเพล็กซ์ดังนี้

```
module sw_3(DT,DB,LR,A1,A2);
        input [15:0] DT;            // Data input Bottom Filter
       input [15:0] DB;            // Data input Top Filter
        input LR;                   // Multiplex control
        output [15:0] A1;           //  Data output to Delay
        output [15:0] A2;           //  Data output to Adder

       wire [15:0]w1;
       wire [15:0]w2;
       wire [15:0]w3;
       wire [15:0]w4;

      assign w1[15:0] = DT[15:0] & {16{LR}};
      assign w2[15:0] = DB[15:0] & {16{! LR}};
      assign w3[15:0] = DT[15:0] & {16{! LR}};
      assign w4[15:0] = DB[15:0] & {16{LR}};
      assign A1 = w1 | w2 ;
      assign A2 = w3 | w4 ;
endmodule
```

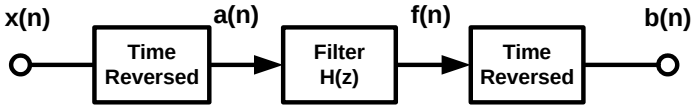
ในการจำลองการทำงานของโมดูลสวิทช์มัลติเพล็กซ์ตัวที่สองจะเห็นว่าเอาต์พุตทั้งสอง (A1, A2) จะเลือกข้อมูลจากอินพุต (DT, DB) ที่มาจากวงจรกรองความถี่โดยจะทำการสลับข้อมูลทุกครั้งที่ L มีการเปลี่ยนแปลงดังแสดงในรูปที่ 5.10



รูปที่ 5.10 ผลการจำลองการทำงานวงจรสวิทช์ส่วนที่ 2

5.5 วงจรกลับสัญญาณในเวลาจริง (Real –time Last In First Out)

การประมวลผลสัญญาณดิจิตอลของวงจรกรองความถี่แบบนอนคอซอลที่เพาเวลและชุลได้นำเสนอบทความ [32, 33] ซึ่งจะประกอบด้วยวงจรกลับสัญญาณในเวลาจริง วงจรกรองความถี่โดยแสดงดังรูปที่ 5.11



รูปที่ 5.11 โครงสร้างวงจรกรองแบบไม่เป็นคอซอลที่สามารถสร้างได้จริง

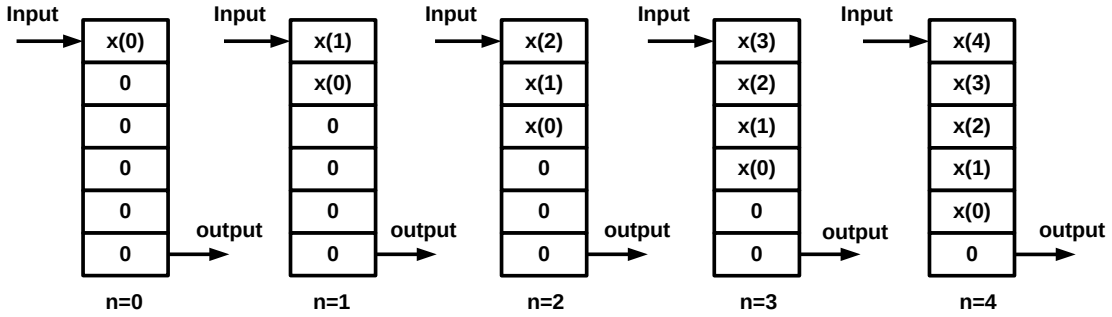
จากรูปที่ 5.11 จะเห็นว่าสัญญาณอินพุต x(n) จะถูกป้อนผ่านวงจรกลับสัญญาณครั้งละ L แซมเปิ้ลจากนั้นจะถูกส่งให้กับวงจรกรองความถี่และถูกป้อนเข้าสู่วงจรกลับสัญญาณอีกครั้งโดยสามารถเขียนในรูปสมการถ่ายโอนได้ดังนี้

$$A(Z) = X(Z^{-1}) \tag{5.6}$$

$$F(Z) = H(Z) \cdot X(Z^{-1}) \tag{5.7}$$

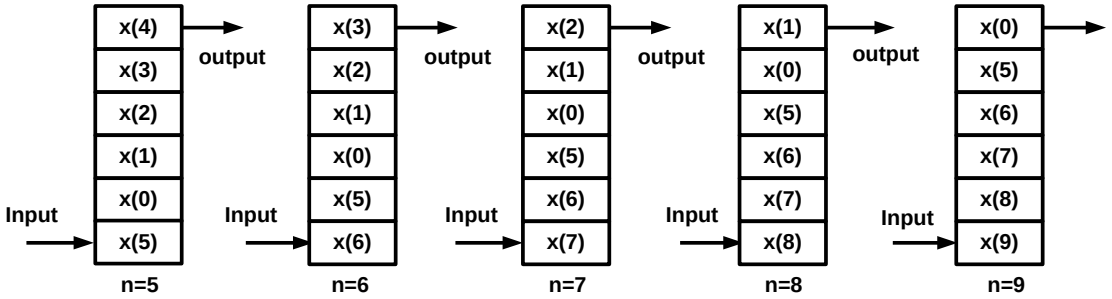
$$B(Z) = F(Z^{-1}) = H(Z^{-1}) \cdot X(Z) \tag{5.8}$$

ในส่วนของวงจรกลับสัญญาณ เพาเวลและชุลได้นำเสนอวิธีการสร้างโดยใช้วงจรชิปรีจีสเตอร์ในการเก็บข้อมูลโดยเมื่อสัญญาณอินพุตในแต่ละแซมเปิ้ลเข้ามาจะถูกเก็บลงในชิปรีจีสเตอร์จนกว่าจะครบค่าความยาวที่จะกลับสัญญาณ (L)



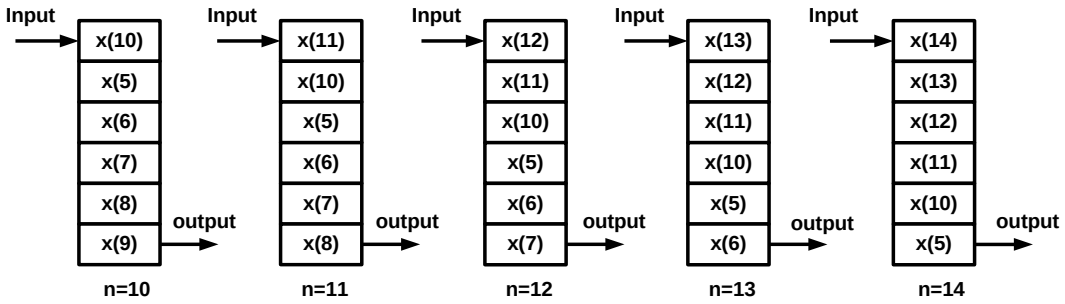
รูปที่ 5.12 การทำงานของวงจรกลับสัญญาณในเวลาจริงที่ n=0 ถึง 4

จากรูปที่ 5.12 จะเห็นว่าสัญญาณอินพุตที่ n=0 ถึง 4 จะถูกเก็บไว้ที่รีจีสเตอร์บนสุดจากนั้นก็จะถูกเลื่อนข้อมูลที่ถูกเก็บไว้ลงมาเพื่อรอรับข้อมูลในแซมเปิ้ลต่อไปโดยขนาดของชิปรีจีสเตอร์จะมีความยาวเท่ากับ L+1



รูปที่ 5.13 การทำงานของวงจรกลับสัญญาณในเวลาจริงที่ n=5 ถึง 9

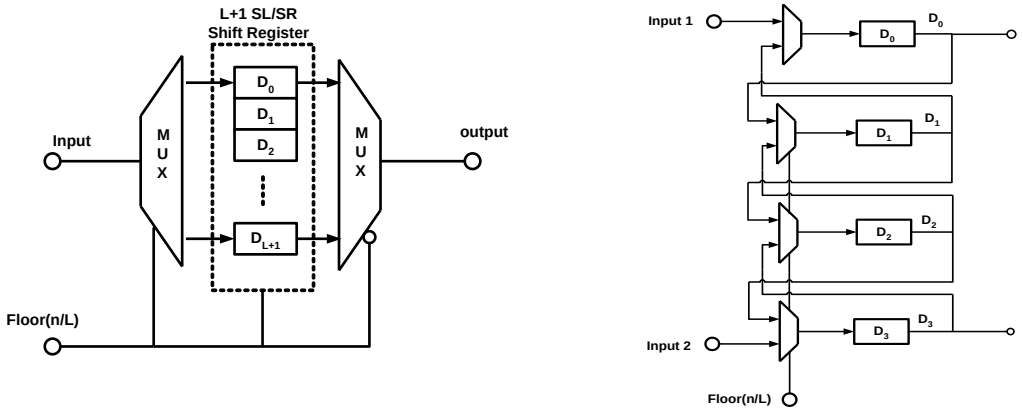
เมื่อเก็บข้อมูลจนครบความยาว L แล้วจากนั้นสัญญาณอินพุตจะถูกเปลี่ยนมาเก็บไว้ที่ตำแหน่งล่างสุดของชิปรีจิสเตอร์แทนและเมื่อทำการเก็บข้อมูลแล้วก็จะทำการเลื่อนข้อมูลขึ้นไปซึ่งเอาต์พุตของวงจรกลับสัญญาณก็จะเปลี่ยนเป็นรีจีสเตอร์ตัวบนสุดแทนด้วยดังแสดงในรูปที่ 5.13



รูปที่ 5.14 การทำงานของวงจรกลับสัญญาณในเวลาจริงที่ n=10 ถึง 14

เมื่อข้อมูลแซมเปิ้ลที่ 10 อินพุตจะเปลี่ยนการเก็บข้อมูลจากด้านบนอีกครั้งดังแสดงในรูปที่ 5.14 ซึ่งจะสลับการทำงานแบบนี้ทุกๆ ความยาว L ซึ่งสามารถเขียนให้อยู่ในรูปแบบวงจรอย่างง่ายได้ดัง

แสดงในรูปที่ 5.15 (a) การทำงานของวงจรกลับสัญญาณในเวลาจริงที่สร้างจากวงจรชิปรีจีสเตอร์ (b) โครงสร้างในส่วนวงจรเก็บและเลื่อนข้อมูล

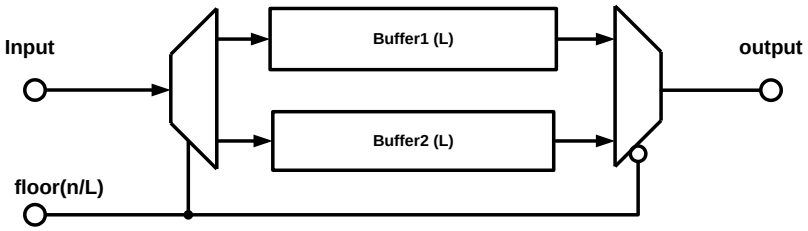


รูปที่ 5.15 วงจรกลับสัญญาณในเวลาจริงที่สร้างจากวงจรชิปรีจีสเตอร์

จากรูปที่ 5.15(b) ความยาว L = 3 และขนาดข้อมูลเป็น 1 บิต ซึ่งในการใช้งานจริงนั้น L จะใช้งานราวๆ 150 ถึง 200 และขนาด 16 บิต ซึ่งนำไปสร้างจริงได้ยากเนื่องจากความซับซ้อนมากซึ่งเมื่อนำไปสร้างบนชิป FPGA จะใช้พื้นที่ในการออกแบบที่มากเกินที่จะสามารถโปรแกรมลงบนตัวชิปได้

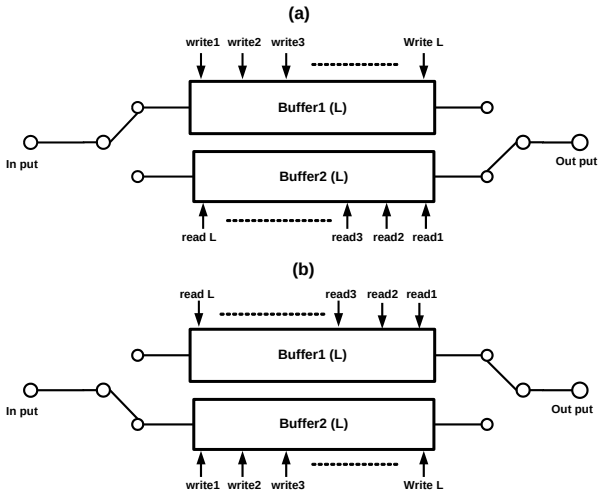
5.5.1 การออกแบบวงจรกลับสัญญาณโดยใช้โครงสร้างของหน่วยความจำ

ในการออกแบบชิปรีจีสเตอร์บนโครงสร้างของ FPGA นั้นจะใช้พื้นที่ในการสร้างที่มากกว่าการออกแบบโดยใช้หน่วยความจำ ซึ่งในการออกแบบใหม่นี้ได้นำเสนอการออกแบบโดยใช้หน่วยความจำมาเก็บข้อมูลแทนชิปรีจีสเตอร์ซึ่งผลที่ได้นั้นมีการใช้พื้นที่น้อยกว่าอย่างมาก



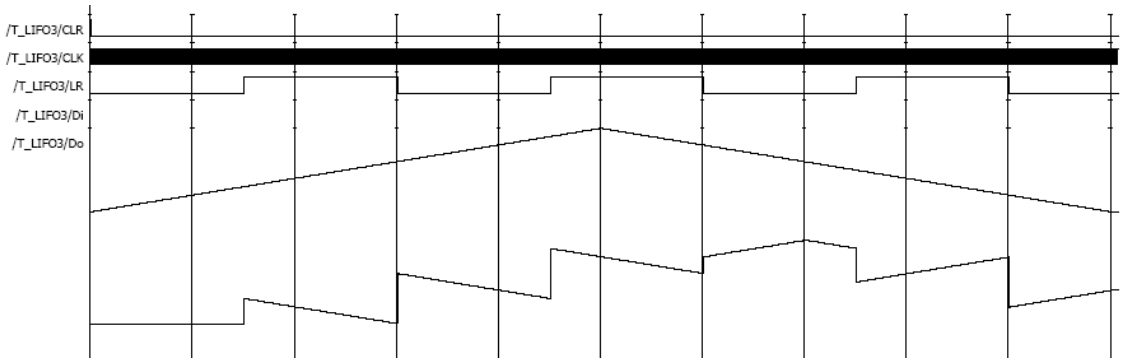
รูปที่ 5.16 วงจรกลับสัญญาณที่ออกแบบด้วยหน่วยความจำ

จากวงจรในรูปที่ 5.16 จะเห็นได้ว่าประกอบด้วยหน่วยความจำสองชุด (Buffer1, Buffer2) ซึ่งแต่ละตัวมีความยาวเท่ากับ L ซึ่งการเก็บข้อมูลและการอ่านข้อมูลออกจะใช้บัฟเฟอร์คนละชุด เช่น ในการเก็บข้อมูลลงในชุดบนข้อมูลเอาต์พุตก็จะใช้ชุดล่างซึ่งจะสลับกันทำงานทุกๆ ความยาว L ดังรูปที่ 5.17 ซึ่งในการสร้างบน FPGA ด้วยโครงสร้างที่นำเสนอนี้จะใช้พื้นที่น้อยกว่าการออกแบบด้วยโครงสร้างของชิปรีจีสเตอร์



รูปที่ 5.17 การอ่านการเขียนข้อมูลในบัฟเฟอร์ (a) บัฟเฟอร์ 1 เขียนข้อมูล บัฟเฟอร์ 2 อ่านข้อมูล (b) บัฟเฟอร์ 2 เขียนข้อมูล บัฟเฟอร์ 1 อ่านข้อมูล

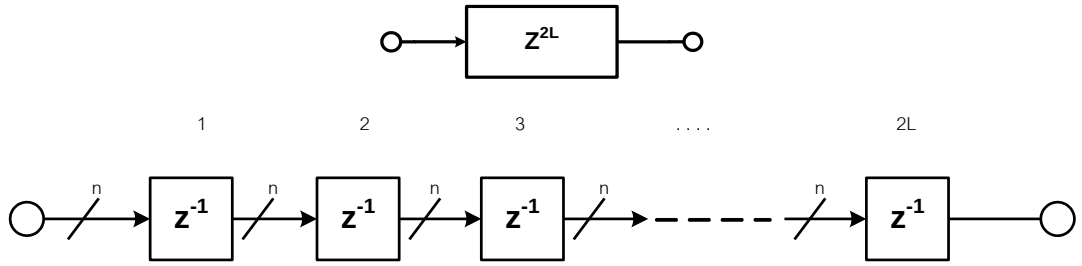
ผลการจำลองการทำงานของโมดูล LIFO ข้อมูล Di จะถูกเก็บข้อมูลเข้าสู่หน่วยความจำทุกๆ สัญญาณนาฬิกา (CLK) จนกระทั่งครบจำนวน L ขาควบคุม L จะทำการกลับสถานะสัญญาณและทำการนำข้อมูลโดยออกมาจากหน่ายความจำที่ทำการเก็บไว้ก่อนแล้วพร้อมกับเก็บข้อมูลชุดใหม่เข้าไปซึ่งจะทำให้สัญญาณที่เข้ามาถูกพลิกกลับเป็นบล๊อกด้วยความยาว L ต่อกันไปเรื่อยๆ ดังแสดงในรูปที่ 5.18



รูปที่ 5.18 ผลการจำลองการทำงานโมดูล LIFO

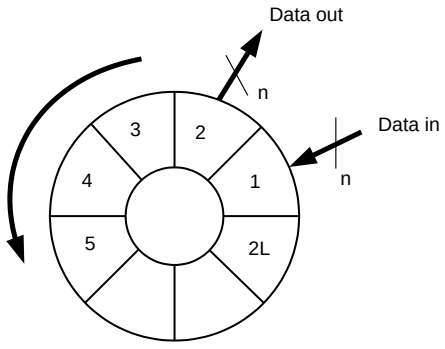
5.6 การออกแบบวงจรหน่วงสัญญาณ

จากที่ได้กล่าวมาแล้วว่าการประมวลผลสัญญาณดิจิตอลส่วนใหญ่จะประกอบไปด้วยวงจรบวก วงจรลบ วงจรคูณ และหน่วงสัญญาณ ซึ่ง ในการออกแบบวงจรหน่วงเวลานั้นสามารถออกแบบโดยการใช้โครงสร้างของชิปรีจีสเตอร์หรือโครงสร้างของหน่วยความจำได้ด้วยเช่นกันโดยโครงสร้างของวงจรหน่วงสัญญาณดังกล่าวแสดงดังรูปที่ 5.19



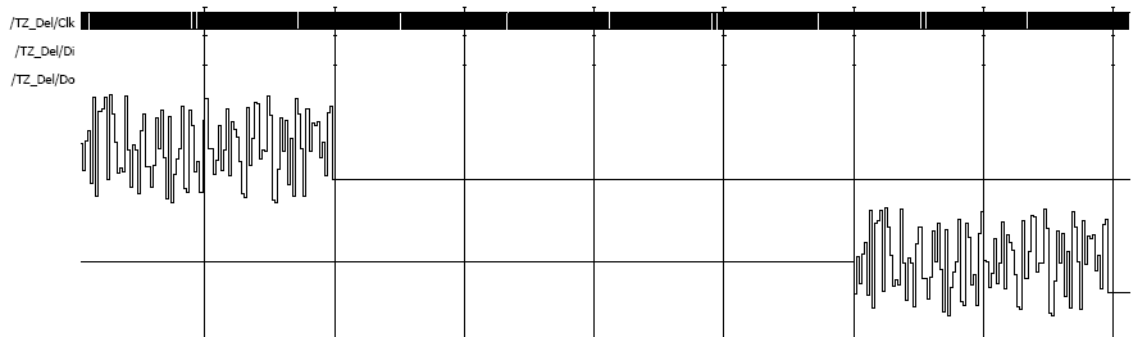
รูปที่ 5.19 โครงสร้างของวงจรหน่วงสัญญาณ

จากรูปที่ 5.19 โครงสร้างดังกล่าวสามารถสร้างได้จากวงจรดีฟลิบฟลอบ (D Flip Flop) โดย ฟลิบฟลอบจำนวนหนึ่งตัวจะหน่วงสัญญาณออกไปหนึ่งแซมเปิ้ล ดังนั้นในการสร้างวงจรหน่วงสัญญาณด้วยโครงสร้างที่กล่าวมานี้จะใช้พื้นที่ในการสร้างที่มากเช่นเดียวกับการออกแบบวงจรกลับสัญญาณในหัวข้อที่ได้กล่าวมาแล้ว จึงได้มีการออกแบบจากวงจรหน่วงสัญญาณด้วยการใช้โครงสร้างของชิปรีจีสเตอร์จะใช้พื้นที่ในการออกแบบที่มากมาเป็นการออกแบบโดยใช้โครงสร้างหน่วยความจำดังแสดงในรูปที่ 5.20 แทนซึ่งเรียกว่าหน่วยความจำแบบวงกลม (circular buffer) โดยในการทำงานนั้นจะทำการเก็บข้อมูล การอ่านข้อมูลด้วยโครงสร้างของ หน่วยความจำสองพอร์ต (dual port RAM) ซึ่งขนาดของหน่วยความจำนั้นจะเป็นตัวกำหนดจำนวนแซมเปิ้ลในการหน่วงสัญญาณ เช่น ในการออกแบบใช้หน่วยความจำขนาด 300 ตำแหน่ง หมายถึงข้อมูลที่เข้ามาในวงจรหน่วงสัญญาณนี้จะถูกหน่วงสัญญาณออกไป 300 แซมเปิ้ล



รูปที่ 5.20 โครงสร้างของวงจรหน่วงสัญญาณที่สร้างจากหน่วยความจำ

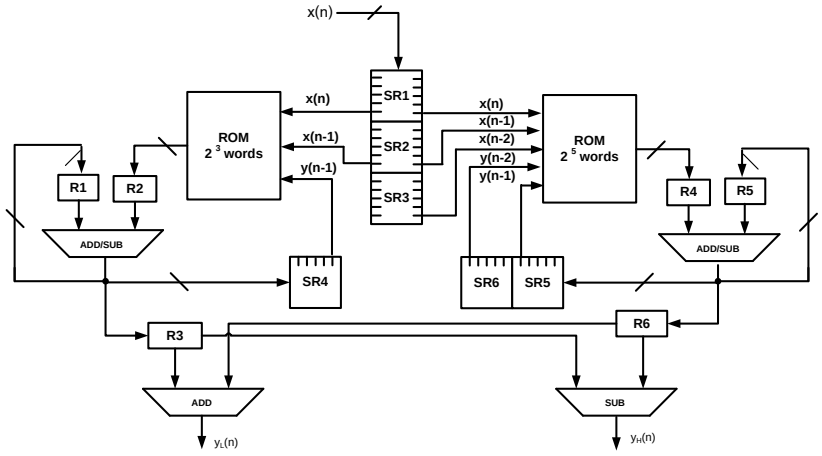
ในการจำลองการทำงานของโมดูลหน่วงเวลา L แซมเปิ้ลข้อมูล Di จะถูกเก็บเข้าไปในหน่วยความจำไปเรื่อยๆ จนครบเวลา L ครั้งจากนั้นข้อมูลที่ถูกเก็บไว้ในชุดแรกออกมา ดังแสดงในรูปที่ 5.21



รูปที่ 5.21 ผลการจำลองการทำงานของโมดูลหน่วงสัญญาณ

5.7 วงจรครอสโอเวอร์เน็ตเวอร์คจากโครงสร้างคณิตศาสตร์การกระจาย

การสร้างวงจรกรองบน เอฟ พี จี เอ ด้วยโครงสร้างของคณิตศาสตร์การกระจายด้วยภาษา Verilog HDL โดยมีโครงสร้างดังรูปที่ 5.22



รูปที่ 5.22 วงจรครอสโอเวอร์เน็ตเวอร์คแบบสองทางจากโครงสร้างของคณิตศาสตร์การกระจาย

จากรูปที่ 5.22 เป็นวงจรดิจิตอลครอสโอเวอร์เน็ตเวอร์คด้วยวงจรกรองความถี่ผ่านตลอดต่อขนาน 2 วงจร (two parallel all-pass filters) ที่สร้างด้วยวงจรคณิตศาสตร์การกระจายโดยจะรับข้อมูลมาจากวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลที่เข้ามาในลักษณะอนุกรม เพื่อให้ง่ายในการสร้างวงจรจะถูกแบ่งเป็นโมดูลย่อยๆตามความเหมาะสมซึ่งสามารถแบ่งออกได้ดังนี้

- โครงสร้างตารางเปิดดูค่า (LUT)
- ชิปรีจิสเตอร์แบบ เข้าขนานออกอนุกรมขนาด 16 บิต (16 Bit PISO)
- ชิปรีจีสเตอร์แบบ เข้าอนุกรมออกอนุกรมขนาด 16 บิต (16 Bit SISO)
- แอกคิวมูเลเตอร์ขนาด16 บิต แบบคิดเครื่องหมาย (16 Bit Accumulator)
- วงจรกำเนิดสัญญาณควบคุมจังหวะการทำงาน (System clock)

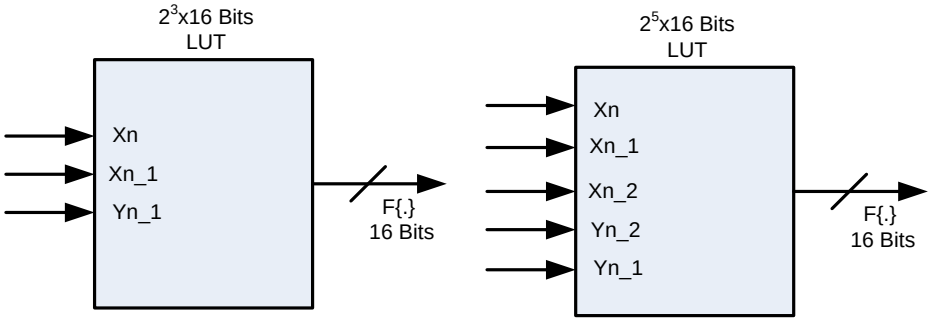
การออกแบบในแต่ละส่วนจะต้องทราบถึงหลักการทำงานของสัญญาณต่างๆที่เข้ามาแล้วทำการกำหนดฟังก์ชันการทำงานกับสัญญาณที่ป้อนเข้ามาว่าต้องการสัญญาณเอาต์พุตจากโมดูลนั้นเป็นอย่างไร การทำงานและผลการทำงานในแต่ละโมดูลนั้นสามารถออกแบบและจำลองการทำงานด้วยโปรแกรม Xilinx ISE และ ModelSim โดยขั้นตอนการออกแบบเป็นดังนี้

5.7.1 การสร้างไฟล์ตารางเปิดดู

จากผลรวมของค่าสัมประสิทธิ์ที่ได้จากตารางที่ 5.1 และ 5.2 สามารถนำไปสร้างตารางเปิดดูในรูปแบบของภาษา Verilog HDL ซึ่งจะมีโครงสร้างของภาษาดังนี้

```
module LUT(Di,Y);
  input [1:0] Di;            // Data input
  output [7:0] Y;          // Data output to top Filter
   always @(Di)
   begin
       case (Di)
            2 ' d0 : Y <= 8' d250;
              2 ' d1 : Y <= 8' d200;
              2 ' d2 : Y <= 8' d150;
              2 ' d3 : Y <= 8' d100;
           default : Y <= 16'b0;
      endcase
  end
endmodule
```

จากตัวอย่างข้างบนเป็นไฟล์ตารางเปิดดูขนาดอินพุต 2 บิตให้เอาต์พุตขนาด 8 บิตซึ่งหากค่าสัมประสิทธิ์มีจำนวนมากขึ้นก็ยิ่งทำให้มีจำนวนเงื่อนไขที่มากขึ้นตามไปด้วย ดังนั้นในการลดความยุ่งยากในการสร้างไฟล์ดังกล่าว จึงได้ออกแบบซอร์ฟแวร์ในการสร้างไฟล์ตารางเปิดดูขึ้นมาโดยสามารถกำหนดค่าของสัมประสิทธิ์ ขนาดของข้อมูลเอาต์พุต ซึ่งวิธีการนี้ทำให้สามารถลดระยะเวลาในการทำงานลงอย่างมาก

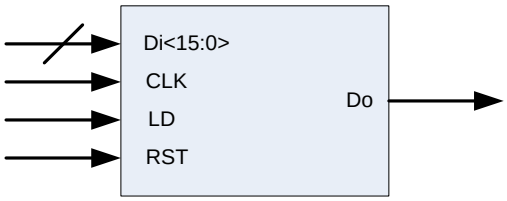


รูปที่ 5.23 สัญลักษณ์ของโมดูลเปิดตารางจากตารางที่ 5.1 และ 5.2

5.7.2 การออกแบบชิปรีจิสเตอร์แบบเข้าขนานออกอนุกรมขนาด 16 บิต (16 Bit PISO)

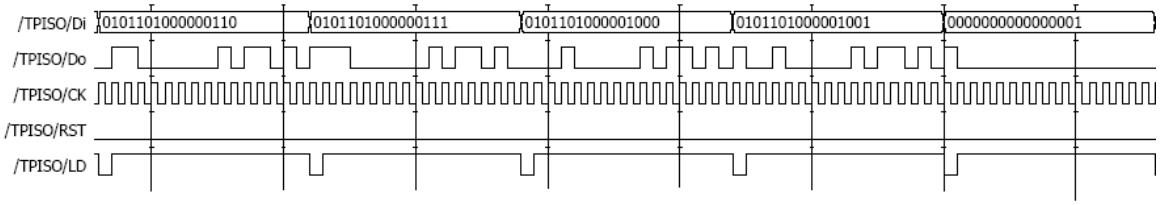
การออกแบบวงจรชิปรีจีสเตอร์แบบเข้าขนานออกอนุกรมขนาด 16 บิตแสดงดังรูปที่ 5.24 มี Di เป็นอินพุตขนาด 16 บิตมี Do สัญญาณเอาต์พุต ซึ่งมีขาที่ใช้ในการควบคุมได้แก่ รีเซ็ต (RST) โหลดข้อมูล (LD) และสัญญาณนาฬิกา (CLK) สามารถเขียนด้วยภาษาอธิบายฮาร์ดแวร์ได้ดังนี้

```
module PISO(Di,Do,CK,RST,LD);
input [width-1:0] Di CK,RST,LD;
output Do;
 reg       [16:0]dat;
            assign Do = dat[0];
            always @ (posedge CK or posedge RST or negedge LD )
            begin
                        if (RST)
                                    dat[16:0] = 0  ;
            else begin
                        if(!LD)
                                    dat[16:0] = {0,Di};
                        else begin
                                    dat[15:0]  = {dat[16:1],0};
                        end
           end
 end
endmodule
```



รูปที่ 5.24 โครงสร้างของโมดูลชิปรีจีสเตอร์แบบเข้าขนานออกอนุกรมขนาด 16 บิต

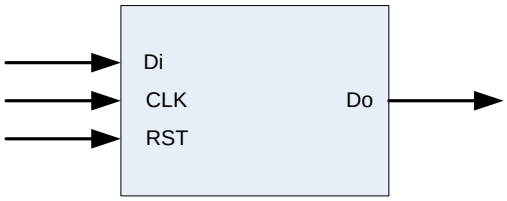
การจำลองการทำงานของโมดูล ข้อมูลจะถูกป้อนให้กับอินพุตก่อนที่จะทำการโหลดข้อมูลลงในชิปรีจีสเตอร์ทั้ง16 บิตก่อนที่จะทำการเลื่อนข้อมูลออกมาทีละบิตทางเอาต์พุตดังแสดงในรูปที่ 5.25



รูปที่ 5.25 ผลการจำลองการทำงานของโมดูลชิปรีจีสเตอร์แบบเข้าขนานออกอนุกรม

5.7.3 การออกแบบชิปรีจีสเตอร์แบบ เข้าอนุกรมออกอนุกรมขนาด 16 บิต (16 Bit SISO)

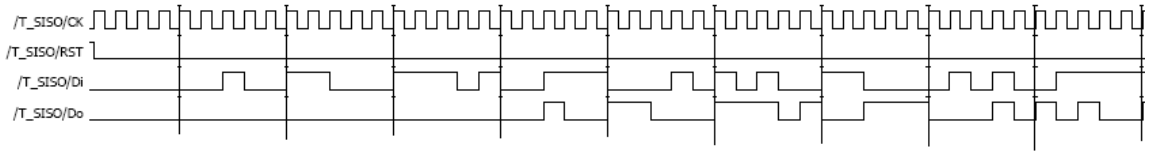
การออกแบบวงจรชิปรีจิสเตอร์แบบเข้าอนุกรมออกอนุกรมขนาด 16 บิตแสดงดังรูปที่ 5.26 โดยที่มี Di เป็นอินพุต Do เป็นสัญญาณเอาต์พุต ซึ่งมีขาที่ใช้ในการควบคุมได้แก่ รีเซ็ต (RST) และสัญญาณนาฬิกา (CLK) สามารถเขียนด้วยภาษาอธิบายฮาร์ดแวร์ได้ดังนี้



รูปที่ 5.26 โครงสร้างของโมดูลชิปรีจีสเตอร์แบบเข้าอนุกรมออกอนุกรมขนาด 16 บิต

```
module SiSo(Di,Do,CK,RST);
  input Di,RST,CK;
output Do;
  reg     [15:0]dat,Do;
  always @(posedge CK or posedge RST)
          if (RST)  begin
              dat[15:0]=0;
              Do=0;
          end
          else begin
              Do=dat[15];
              dat[15:0]  = {dat[14:0],Di};
          end
endmodule
```

เมื่อทำการทดสอบการทำงานของโมดูลโดยการป้อนสัญญาณอินพุตแบบอนุกรมดังแสดงในรูปที่ 5.27 จะเห็นได้ว่าโมดูลดังกล่าวจะทำการหน่วงสัญญาณที่เข้ามาออกไปจำนวน 16 รอบของสัญญาณนาฬิกา โดยที่ข้อมูลที่ผ่านโมดูลดังกล่าวยังมีลักษณะเช่นเดิมแต่ถูกหน่วงไป 16 รอบสัญญาณนาฬิกา

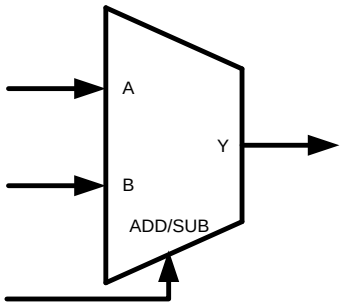


รูปที่ 5.27 ผลการจำลองการทำงานของโมดูลชิปรีจีสเตอร์แบบเข้าอนุกรมออกอนุกรม

5.7.4 การออกแบบแอกคิวมูเลเตอร์ขนาด16 บิต แบบคิดเครื่องหมาย (16 Bit Accumulator)

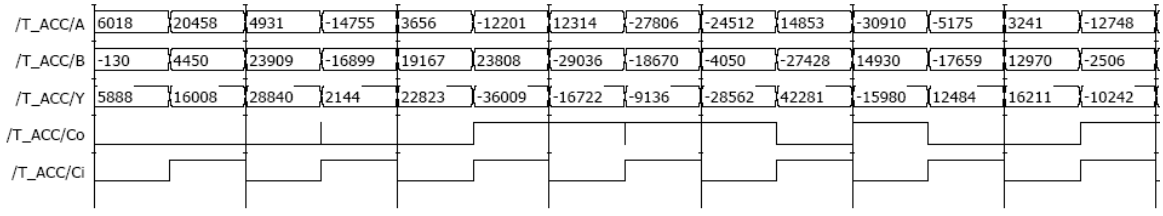
ส่วนประกอบภายในของแอกคิวมูเลเตอร์ที่ทำการออกแบบแสดงดังรูปที่ 5.28 ซึ่งเป็นวงจรบวกลบเลขขนาด 16 บิตแบบคิดเครื่องหมายโดยมี (A) และ (B) เป็นอินพุตส่วน (Y) เป็นเอาต์พุตขนาด 16 บิตสามารถควบคุมการบวกลบข้อมูลด้วยสัญญาณ (add/sub) สามารถเขียนด้วยภาษาอธิบายฮาร์ดแวร์ได้ดังนี้

```
module acc(A,B,Y,Co,Ci);
 input [15:0] A, 15:0] B,Ci;
 output [16:0] Y,Co;
 wire [15:0]Do,t1;
 reg [16:0]result;
            assign Do=B^{16{Ci}};
            assign t1 = A[15]^Do[15];
            assign Co = result[16]^t1;
            assign Y[16]=Co;
            assign Y[15:0] =  result[15:0];
            always @(A or Do or Ci)
                        result = A + Do + Ci;
endmodule
```



รูปที่ 5.28 โครงสร้างของโมดูลแอกคิวมูเลเตอร์

การจำลองการทำงานของโมดูลแอกคิวมูเลเตอร์โดยที่เอาต์พุต (Y) จะเป็นผลบวกหรือผลต่างของข้อมูล A และ B ซึ่งถูกควบคุมด้วยสัญญาณ Ci แสดงดังรูปที่ 5.29

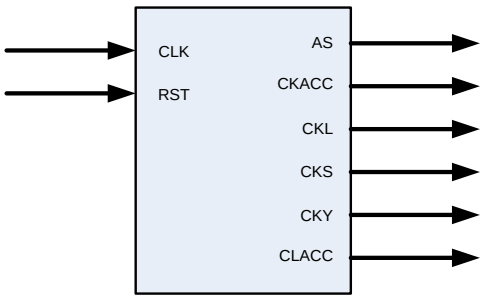


รูปที่ 5.29 ผลการจำลองการทำงานของของโมดูลแอกคิวมูเลเตอร์

5.7.5 การออกแบบวงจรควบคุมจังหวะการทำงาน (System clock)

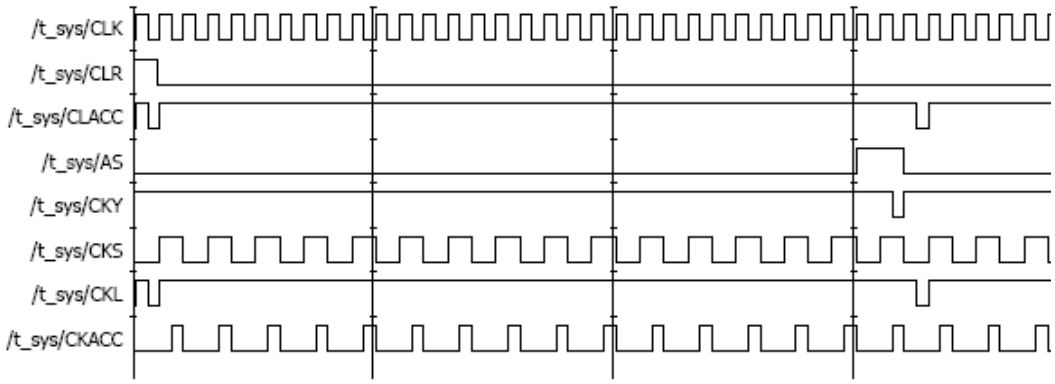
ส่วนประกอบภายในวงจรควบคุมจังหวะการทำงานทั้งหมดที่ทำการออกแบบแสดงดังรูปที่ 5.30 ประกอบไปด้วยสัญญาณนาฬิกาจากภายนอกและสัญญาณรีเซ็ตระบบทั้งหมด โดยการทำงานของโมดูลนี้จะนำสัญญาณนาฬิกาจากภายนอกมาทำการจัดใหม่ให้ระบบทั้งหมดทำงานเข้าจังหวะกันซึ่งในวงจรครอสโอเวอร์เน็ตเวอร์คจะประกอบไปด้วยสัญญาณต่างๆ ดังนี้

- CLK เป็นสัญญาณนาฬิกาจากวงจรออสซิลเลเตอร์ภายนอก
- RST เป็นสัญญาณที่ใช้ในการรีเซ็ตระบบทั้งหมด
- CKL สัญญาณที่ใช้ในการโหลดข้อมูลอินพุตซึ่งมีอัตราการสุ่มข้อมูลที่ 48KHz
- CLKS สัญญาณที่ใช้ในการชิปข้อมูลเข้ามาประมวลผลโดยในแต่ละรอบของการสุ่มข้อมูลมาประมวลผลสัญญาณ CLKS จะกำเนิดสัญญาณนาฬิกา 16 ครั้งเท่ากับขนาดของข้อมูลในการประมวลผลคือ 16 บิต
- CLKACC เป็นสัญญาณนาฬิกาที่ใช้ในการประมวลผลของแอกคิวมูเลเตอร์ โดยสัญญาณนี้จะเกิดขึ้นภายหลังจากการโหลดข้อมูลเสร็จ
- AS สัญญาณที่ใช้ในการควบคุมการบวก ลบ ข้อมูลในแอกคิวมูเลเตอร์โดยจะเปลี่ยนสถานะเป็นลอจิก “1” ทุกๆ บิต MSB ของข้อมูลนั้นๆ
- CKY สัญญาณ ที่ใช้ในการโหลดข้อมูลออกจากแอกคิวมูเลเตอร์ภายหลังจากประมวลผลเสร็จ
- CLACC เป็นสัญญาณที่ใช้ในการเคลียร์ข้อมูลในแอกคิวมูเลเตอร์ก่อนที่จะทำการโหลดข้อมูลในชุดถัดไปมาประมวลผล



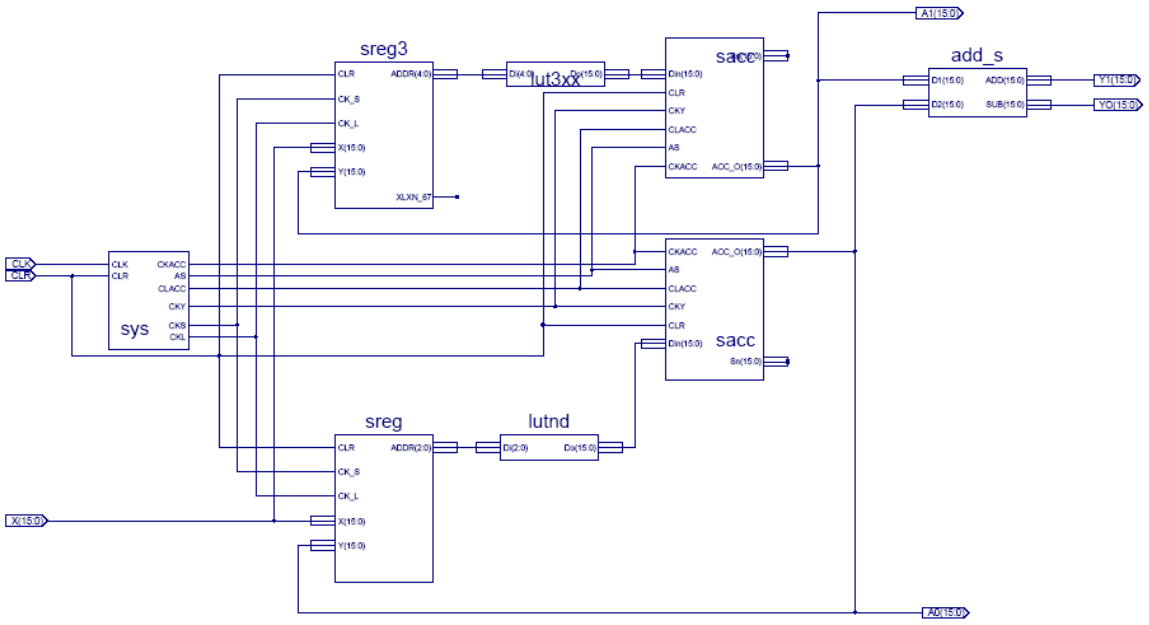
รูปที่ 5.30 โครงสร้างของโมดูลควบคุมจังหวะการทำงาน

การจำลองการทำงานของโมดูลควบคุมจังหวะการทำงานจะทำการป้อนสัญญาณรีเซ็ตและสัญญาณนาฬิกาโมดุลจะสร้างสัญญาณควบคุมออกมาเพื่อนำไปควบคุมการทำงานของโมดูลอื่นๆต่อไปดังแสดงในรูปที่ 5.31



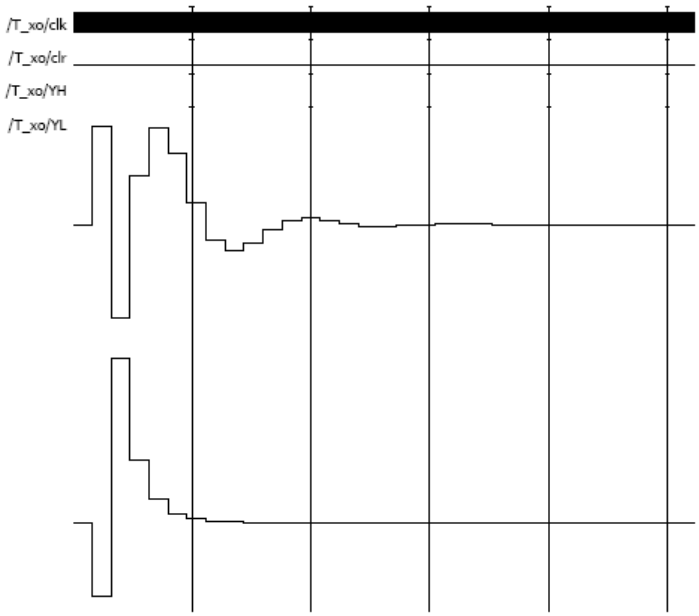
รูปที่ 5.31 ผลการจำลองการทำงานของของโมดูลวงจรควบคุมจังหวะการทำงาน

ซึ่งเมื่อนำโมดูลทั้งหมดที่ได้ออกแบบมาต่อกันเป็นวงจรครอสโอเวอร์เน็ตเวอร์คบน FPGA ตามโครงสร้างในรูปที่ 5.4 จะได้โมดูลของวงจรครอสโอเวอร์เน็ตเวอร์คแบบสองทางเพื่อที่จะนำไปออกแบบรวมกับโมดูลอื่นต่อไปดังแสดงในรูปที่ 5.32



รูปที่ 5.32 วงจรครอสโอเวอร์แบบสองทางจากวงจรกรองทุกความถี่ต่อขนานกันบน FPGA

เมื่อนำโครงสร้างจากรูปที่ 5.32 ไปหาผลตอบสนองสัญญาณอิมพัลซ์ของระบบจะได้ผลจำลองการทำงานของ $A_0(Z)$ และ $A_1(Z)$ ดังรูปที่ 5.33



รูปที่ 5.33 ผลการตอบสนองอิมพัลซ์ของวงจรครอสโอเวอร์เน็ตเวอร์คแบบสองทางบน FPGA

5.8 การสังเคราะห์วงจร

หลังจากที่ได้กล่าวถึงการออกแบบวงจรครอสโอเวอร์เน็ตเวอร์คในแต่ละส่วนเป็นที่เรียบร้อยแล้วขั้นตอนสุดท้ายของการสร้างก็คือการสังเคราะห์วงจรที่ได้ออกแบบเพื่อทำการโปรแกรมลงสู่ FPGA จากซอร์ฟแวร์ที่ใช้ในการพัฒนาจะรายงานผลการสังเคราะห์วงจรรวมทั้งขีดความสามารถต่างๆ ที่วงจรสามารถทำงานได้รวมไปจนถึงการใช้งานทรัพยากรภายใน FPGA ที่ถูกใช้ไปดังแสดงในตารางที่ 5.3 ดังนี้

ตารางที่ 5.4 ผลการสังเคราะห์แต่ละโมดูลของวงจรครอสโอเวอร์เน็ตเวอร์ค

Device utilization summarizing **Selected device : XC2S2000 144 - 5**	**PAPF**	**DELAY** **(L=300)**	**LIFO** **(L=150)**	**SW1**	**SW2**
Number of Slices (1920 Max)	175 (9%)	19 (0.98%)	529 (27%)	18 (0.93%)	18 (0.93%)
Number of Slices Flip Flops (3840 Max)	240 (6%)	18 (0.46%)	42 (1%)	-	
Number of 4 input LUTs (3840 Max)	316 (8%)	34 (0.88%)	646 (16%)	32 (0.83%)	32 (0.83%)
Number of bonded IOBs (97 Max)	81 (83%)	33 (34%)	34 (35%)	49(50%)	65(67%)
Number of BRAMs (12 Max)	-	1 (8%)	-	-	-
Number of GCKs (8 Max)	1 (12%)	1 (12%)	1 (12%)	-	-
Maximum speed (MHz)	220.216	143.730	85.844	-	

* PAPF (Parallel All-pass Filter), LIFO (Last In First Out), sw1 (switch multiplex 1), sw2 (switch multiplex 2)