เครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่

AMPLITUDE RESPONSE ANALYZER

โดย

นายปิยะพันธ์ คงเศรษฐ

นายสราวุฒิ บัวศรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่

AMPLITUDE RESPONSE ANALYZER

โดย

นายปิยะพันธ์ คงเศรษฐ 44015016

นายสราวุฒิ บัวศรี 44015031

อาจารย์ที่ปรึกษา

ผศ.เกรียงไกร วงศ์โรจนภรณ์

รศ.ดร.สุวิพล สิทธิชีวภาค

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่

AMPLITUDE RESPONSE ANALYZER

ผู้จัดทำ

1. นายปิยะพันธ์ คงเศรษฐ 44015016

2. นายสราวุฒิ บัวศรี 44015031

..... อาจารย์ที่ปรึกษา

(ผศ.เกรียงไกร วงศ์โรจนภรณ์)

..... อาจารย์ที่ปรึกษา

(รศ.ดร.สุวิพล สิทธิชีวาภาค)

เครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่

AMPLITUDE RESPONSE ANALYZER

โดย ปิยะพันธ์ คงเศรษฐ 44015016

สราวุฒิ บัวศรี 44015031

อาจารย์ที่ปรึกษา ผศ.เกรียงไกร วงศ์โรจนภรณ์

รศ.ดร.สุวิพล สิทธิชีวภาค

บทคัดย่อ

โครงงานนี้เป็นโครงงานสร้าง อุปกรณ์ ใช้สำหรับตรวจสอบอุปกรณ์ เพื่อหาผลตอบสนองทางความถี่โดยเครื่องมือนี้จะสร้างความถี่ที่รู้ขนาด ความแรงสัญญาณ และความถี่สัญญาณป้อนให้กับอุปกรณ์ที่จะทดสอบจากนั้นวัดผลตอบสนองของอุปกรณ์ทดสอบ เพื่อมาคำนวณหาผลตอบสนองทางด้านขนาด

ABSTRACT

This project is about building and instrument for checking amplitude and phase response of testing device. The instrument will generate sine wave with can vary amplitude and frequency apply to device under test (DUT) and measure amplitude response of the device and calculate amplitude and phase response display on computer.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

การศึกษาระบบเป็นสิ่งสำคัญสำหรับการเรียนรู้และพัฒนาความรู้เละเทคโนโลยีให้ก้าวหน้าและการศึกษาระบบนี้อาจทำได้จากการคำนวณและการวัดผลโดยอุปกรณ์ และอุปกรณ์ที่สร้างขึ้นมานี้ เป็นอุปกรณ์ที่ใช้ตรวจสอบผลตอบสนองของระบบที่มีผลต่อความถี่ต่างๆ เครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่นี้มีหลักการทำงานคล้ายกับเครื่องวิเคราะห์โครงข่ายไฟฟ้า แต่เครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่นี้สามารถสร้างมาได้ด้วยราคาที่ไม่สูงและสามารถตรวจสอบระบบได้อย่างถูกต้อง

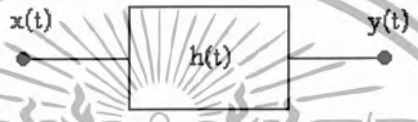
1.2 เนื้อหาของปริญญานิพนธ์

เนื้อหาของปริญญานิพนธ์บทที่ 1 จะเป็นบทนำจะกล่าวถึงความเป็นมาและเนื้อหาโดยรวมบทที่ 2 จะกล่าวถึงทฤษฎีและหลักการที่เกี่ยวข้องกับกับโครงงานนี้ เช่น ทฤษฎีเกี่ยวกับออปแอมป์ และระบบโดยรวมของโครงงาน ส่วนบทที่ 3 จะเกี่ยวของกับการคำนวณและการสร้างโครงงาน มีการคำนวณเกี่ยวกับการผลิตความถี่ด้วยการทำงานของวงจรด้วยภาษา VHDL โดยการประยุกต์และออกแบบทั้งหมดใช้โปรแกรม Max Plus II ในการสร้างสัญญาณไซน์เวฟออกมาและใช้โปรแกรมภาษาซีในการประมวลผลของสัญญาณที่ทำการทดสอบและแสดงผลสู่จอภาพ

รวมทั้งวงจรที่ใช้งานจริง บทที่ 4 จะเป็นผลที่ได้ทดลองในส่วนต่างๆของวงจรที่จะนำมาสร้างเป็นเครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่นี้ ส่วนบทที่ 5 เป็นบทวิจารณ์และบทสรุปผลที่ได้คำนวณและได้ทดลองใช้งานจริงและจะกล่าวเกี่ยวกับข้อดีและข้อบกพร่องและแนวทางแก้ไข

บทที่ 2
ทฤษฎีและหลักการ

เครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่ เป็นเครื่องมือที่ใช้วัดแสดงลักษณะของวงจร อุปกรณ์หรือระบบทางไฟฟ้า ในการหาผลตอบสนองของความถี่ของวงจร อุปกรณ์หรือระบบทางไฟฟ้านั้นๆ สามารถทำได้โดยป้อนสัญญาณคลื่นรูปไซน์ (Sinusoidal) ที่มีความถี่ตั้งแต่ 1Hz – 20kHz และมีแอมพลิจูดของสัญญาณ 2 Vp-p ป้อนให้กับวงจร อุปกรณ์หรือระบบทางไฟฟ้า หลังจากนั้นจะทำการเปรียบเทียบสัญญาณขาเข้า (Input Signal) กับสัญญาณขาออก (Output Signal) เพื่อหาผลตอบสนองความถี่ของระบบนั้นๆ

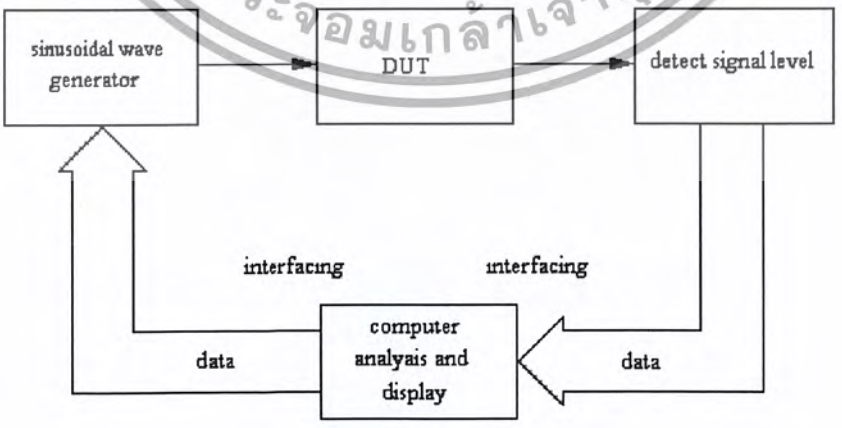


รูปที่ 2.1 ระบบที่จะทดสอบ

2.1 ส่วนประกอบเบื้องต้นของเครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่

2.1.1 หน่วยประมวลผลวิเคราะห์และแสดงผล (Computer analysis and display)

ในโครงงานนี้ใช้คอมพิวเตอร์ส่วนบุคคล (PC) ในการวิเคราะห์และแสดงผลการตอบสนองความถี่ของระบบซึ่งจะเป็นตัวควบคุมการผลิตความถี่โดยการส่งบิทควบคุมไปยังส่วนสร้างสัญญาณกวาดความถี่และจะรับข้อมูลจากวงจรตรวจวัดระดับสัญญาณผลตอบสนองความถี่ จากนั้นจะแสดงผลที่ได้จากการตรวจวัดระบบที่หน้าจอแสดงผล



รูปที่ 2.2 แผนผังส่วนประกอบเบื้องต้นของเครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่

2.1.2 ส่วนสร้างสัญญาณกวาดความถี่ (Sine Wave Generator)

การสร้างสัญญาณรูปคลื่นไซน์จะใช้ FPGA (Field Programmable Gate Arrays) ในการผลิตความถี่ซึ่งจะใช้บิทควบคุมการผลิตจากหน่วยประมวลผลวิเคราะห์และแสดงผล จากนั้นจะทำการกรองเอาเฉพาะความถี่ที่ต้องการก่อนจะส่งสัญญาณความถี่ไปยังอุปกรณ์ที่จะทดสอบ

2.1.3 วงจรตรวจวัดระดับสัญญาณผลตอบสนองความถี่ (Detect signal level)

ส่วนนี้จะใช้วงจรตรวจจับแรงดันยอด จากนั้นทำการแปลงสัญญาณอนาล็อกเป็นดิจิตอลโดยวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (Analog to Digital) เพื่อส่งให้คอมพิวเตอร์แสดงผล

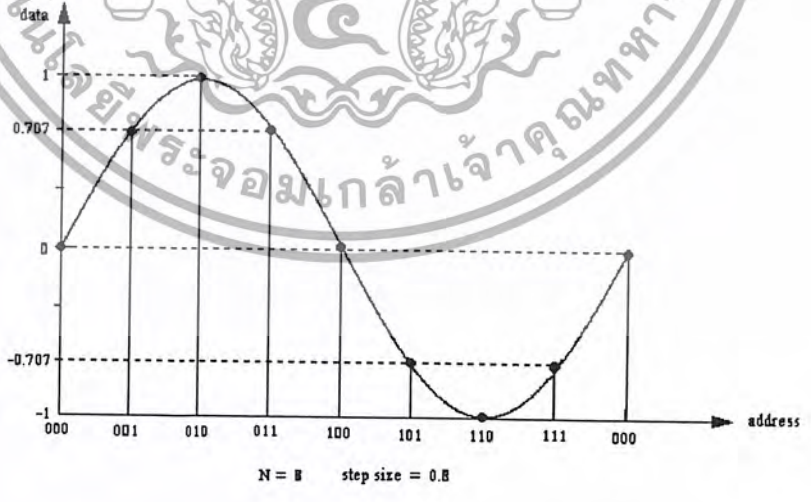
2.1.4 อุปกรณ์ที่จะทำการทดสอบ (Device Under Test)

ต้องเป็นอุปกรณ์ประเภทที่มีการทำงานแบบเชิงเส้น (Linear) จึงจะได้ผลการวัดโดยเครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่ทำการตรวจวัดผลตอบสนองอย่างถูกต้องแม่นยำที่สุด

2.2 ส่วนสร้างสัญญาณกวาดความถี่ (Sinusoidal Wave Generator)

2.2.1 ทฤษฎีสร้างรูปคลื่นแบบไดเร็กดิจิตอลซินธิไซเซอร์ (Direct Digital Synthesizer)

แนวคิดในการสร้างรูปคลื่นแบบไดเร็กดิจิตอลซินธิไซเซอร์หรือ DDS คือการกำเนิดรูปคลื่นโดยอาศัยวิธีการทางดิจิตอลซึ่งในเครื่องกำเนิดความถี่ทั่วๆไปนั้นจะใช้วิธีการทางอนาล็อกเฟสล็อกลูป หรือการกำเนิดรูปคลื่นโดยใช้คริสตอล การสร้างรูปคลื่นแบบไดเร็กดิจิตอลซินธิไซเซอร์จะเก็บข้อมูลของสัญญาณที่จะกำเนิดขึ้นมานั้นให้ครบคาบของสัญญาณซึ่งถ้าหากเรายิ่งเก็บข้อมูลของสัญญาณจำนวนมากเท่าไรก็จะทำให้สัญญาณที่ผลิตออกมาใกล้เคียงกับความเป็นจริงมากเท่านั้น แต่จะมีข้อเสียคือจะสิ้นเปลืองเนื้อที่ในหน่วยความจำมากตามไปด้วยหน่วยความจำมากตามไปด้วย



รูปที่ 2.3 ตัวอย่างการเก็บข้อมูลของสัญญาณไซน์ (Sine Wave) ของ 1 คาบเวลา

ซึ่งการเก็บข้อมูลจะต้องใช้การสุ่ม (Sampling) ข้อมูลบนสัญญาณแต่ละจุดด้วยเวลาที่เท่ากันทุกจุดดังนั้นจุดบนสัญญาณที่ต้องการเก็บคือ 360/จำนวนข้อมูล เช่นต้องการเก็บข้อมูล 1024 ค่า ดังนั้นจะต้องทำการเก็บข้อมูลทุกๆ 360/1024 สามารถทำเป็นตารางได้ดังนี้การเก็บข้อมูลจำนวน N ค่าพิจารณาได้จากตารางที่ 2.1 หลังจากที่เราได้ข้อมูลมาแต่ละจุดแล้ว จะต้องมาทำการจัดค่า (Quantization) ให้มีค่าเป็นทางดิจิตอลเมื่อเราได้ข้อมูลที่มีค่าเป็นเลขฐานสองแล้วก็จะนำค่าข้อมูลเหล่านี้ไปเก็บในหน่วยความจำโดยการจัดเรียงกันไปคือค่าของสัญญาณจุดแรกบนสัญญาณจะถูกเก็บที่ตำแหน่ง (Address) แรกค่าของข้อมูลที่สองจะถูกเก็บในตำแหน่ง (Address) ถัดไปจนครบหมดทุกค่า

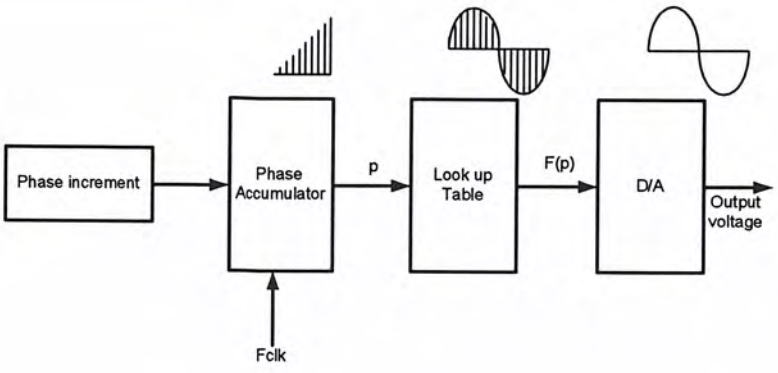
นอกจากสัญญาณไซน์แล้วเราสามารถสร้างสัญญาณชนิดอื่นได้อีกมากโดยใช้โปรแกรมคำนวณค่าของสัญญาณต่างๆโดยการคำนวณค่าของสัญญาณแต่ละจุด โดยใช้หลักการเดียวกับการกำเนิดคลื่นไซน์ที่ได้กล่าวมาข้างต้นนั่นเอง

ตารางที่ 2.1 ตารางแสดงการเก็บข้อมูลของสัญญาณไซน์ (Sine Wave)

ข้อมูลที่	เฟส	ข้อมูล
0	$0.\pi/N$	$f(0)=\sin(0.\pi/N)$
1	$1.\pi/N$	$f(1)=\sin(1.\pi/N)$
2	$2.\pi/N$	$f(2)=\sin(2.\pi/N)$
3	$3.\pi/N$	$f(3)=\sin(3.\pi/N)$
*	*	*
*	*	*
*	*	*
*	*	*
N – 3	$(N-3).\pi/N$	$f(N-3)=\sin((N-3).\pi/N)$
N – 2	$(N-2).\pi/N$	$f(N-2)=\sin((N-2).\pi/N)$
N – 1	$(N-1).\pi/N$	$f(N-1)=\sin((N-1).\pi/N)$

2.2.2.ทฤษฎี Direct Digital Frequency Synthesizer (DDS)

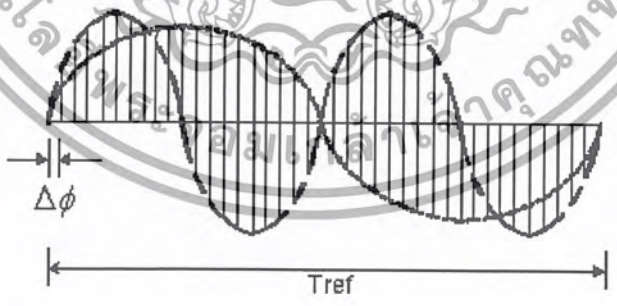
เทคโนโลยีไดเร็กดิจิตอลซินธิไซเซอร์ (DDS : Direct Digital Synthesis) ในรูปแบบของการใช้สัญญาณดิจิตอลมาควบคุมการสร้างความถี่ โดยการนำ FPGA มาใช้ประยุกต์ใช้งาน



รูปที่ 2.4 แสดง Block Diagram โครงสร้างของวงจรสังเคราะห์ความถี่ที่ใช้หน่วยความจำเป็นค่าขนาดของสัญญาณเอาไว้

วงจรสังเคราะห์ความถี่ดิจิตอลแบบโดยตรงจะอาศัยหน่วยความจำเพื่อเก็บค่าขนาดของสัญญาณที่เฟสต่างๆเอาไว้ ซึ่งรูปสัญญาณที่เก็บเอาไว้อาจจะเป็นรูปสัญญาณซายน์ (Sine) สัญญาณรูปฟันเลื่อย (Sawtooth) หรือรูปสัญญาณอื่นๆ ที่ผู้ใช้งานได้ทำการออกแบบไว้ เมื่อต้องการรูปสัญญาณออกมาที่เอาท์พุต ก็จะทำการเรียกข้อมูลที่เฟสต่างๆ ที่เก็บเอาไว้ออกมา เพื่อป้องกันให้กับวงจรแปลงสัญญาณดิจิตอลเป็นอนาล็อก ดังรูปที่ 2.4

หากต้องการจะเปลี่ยนความถี่ของสัญญาณก็ทำได้โดย การเปลี่ยนอัตราการเรียกขอมูลออกจากหน่วยความจำ (เป็นความเร็วของสัญญาณนาฬิกา) หรือเปลี่ยนลำดับการเรียกข้อมูลที่เฟสต่างๆ เพื่อให้จำนวนของข้อมูลในหนึ่งคาบของสัญญาณเปลี่ยนไป ดังเช่นในรูปที่ 2.5 จะเห็นว่ามีรูปของสัญญาณสองความถี่รูปสัญญาณที่มีความถี่สูงกว่าจะมีจำนวนข้อมูลที่ถูกเรียกออกมาน้อยกว่ารูปสัญญาณที่มีความถี่ต่ำกว่า



รูปที่ 2.5 วงจรสร้างสัญญาณซายน์แบบเชิงเลข

มีข้อสังเกตคือ ระยะระหว่างข้อมูลที่ถูกเรียกออกมาของสัญญาณทั้งสองความถี่มีค่าเท่ากัน ดังนั้นความละเอียดในการเปลี่ยนความถี่จะขึ้นอยู่กับขนาดของหน่วยความจำที่ใช้เก็บข้อมูล

หลักการของวงจรสังเคราะห์ความถี่ดิจิตอลแบบโดยตรง (Direct Digital Frequency Synthesizer) นั้นจะมีสัญญาณนาฬิกา (Fclk) นำมาป้อนให้กับส่วนของ Phase Accumulator ซึ่งวงจรนี้จะทำการบวกวนซ้ำด้วยค่าที่รับเข้ามาจากส่วนเพิ่มเฟส ผลของการบวกจะได้ค่าออกมาเป็นเฟสสะสม P ค่าสะสมเฟส P นี้จะมีลักษณะเพิ่มขึ้นด้วยจำนวนเท่ากันทุกครั้งที่ทำการบวกตามจังหวะสัญญาณนาฬิกาอ้างอิง (Fclk) ค่าเฟสสะสม P จะนำไปชี้ในตารางเปิดดู (Look up table) ซึ่งได้จัดเก็บค่าขนาดของรูปสัญญาณไว้แล้ว ซึ่งจะได้สัญญาณ F(P) ออกมา ซึ่งเป็นค่าแบบดิจิตอล มาทำการเปลี่ยนเป็นค่าอนาล็อกด้วยวงจร Digital–to–Analog Converter (DAC) ก็จะได้ Output Voltage เป็นสัญญาณแบบที่ต้องการ

ความถี่ที่สามารถกำเนิดได้จากโครงสร้างนี้ คือ

$$\text{Fout (min)} = \frac{Fclk}{2^n}$$

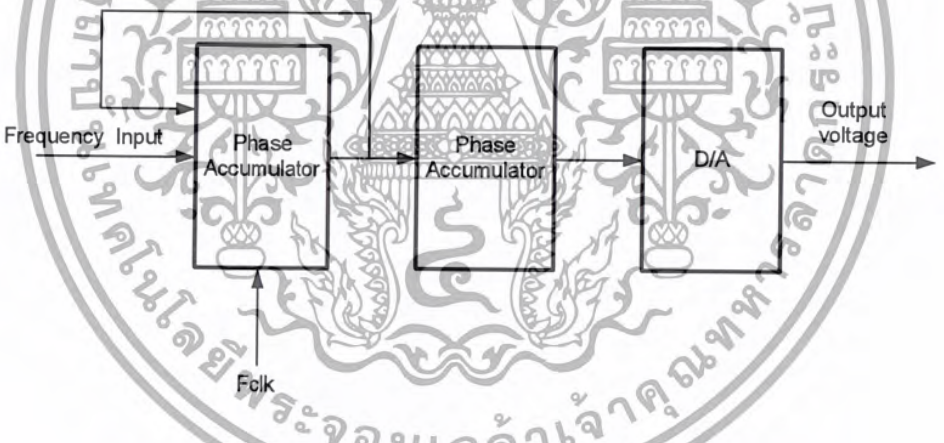
$$\text{Fout} = \frac{(Frequency\ Input \times Fclk)}{2^n}$$

เมื่อ Fout = Frequency Output

Fclk = Frequency Clock

N = จำนวน bit ของ Phase Accumulator

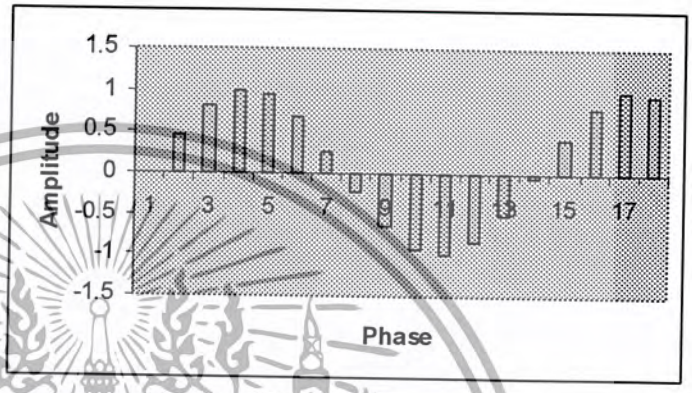
ถ้าค่า N มีค่ามาก Step Size ของความถี่จะมีความละเอียดมากขึ้นด้วย



รูปที่ 2.6 บล็อคไดอะแกรมของการสร้างสัญญาณด้วยหลักการ DDS

จากรูปเป็นตัวอย่างของการกำเนิดสัญญาณซายน์แบบดิจิตอลที่มีค่า Phase Increment เท่ากับ 55

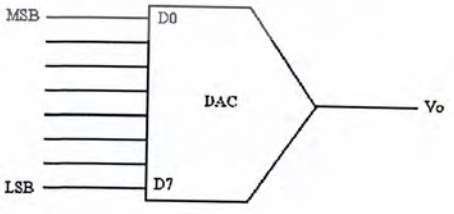
Time	Phase	Amplitude
1	0	0
2	55	0.462
3	110	0.819
4	165	0.991
5	220	0.94
6	275	0.676
7	330	0.259
8	385	-0.216
9	440	-0.643
10	495	-0.924
11	550	-0.996
12	605	-0.843
13	660	-0.5
14	715	-0.044
15	770	0.423
16	825	0.793
17	880	0.985
18	935	0.954



รูปที่ 2.7 เป็นตัวอย่างของการกำเนิดสัญญาณซายน์แบบดิจิตอล ที่มีค่า Phase Increment เท่ากับ 55

2.3 การเปลี่ยนสัญญาณดิจิตอลเป็นอนาล็อก (Digital to Analog Convertor : DAC)

หลักการทำงานของ DAC คือ การนำเอากลุ่มของบิท (BIT) จากคอมพิวเตอร์หรืออุปกรณ์ดิจิตอล เปลี่ยนแปลงเป็นระดับแรงดันอนาล็อกเอาท์พุทของ DAC เป็นระดับความแตกต่างของแต่ละบิทของดิจิตอลอินพุทหลักการพื้นฐานของ (DAC) บล็อกไดอะแกรมของ DAC แสดงในรูปเอาท์พุทที่สร้างขึ้นจาก DAC เป็นได้ทั้งแรงดันและกระแส



รูปที่ 2.8 บล็อกไดอะแกรมของ D/A Converter

เอาท์พุทชนิดใดก็ตามของ DAC ที่ผลิตขึ้นมาได้จากวงจรที่นำมาใช้ในการเปลี่ยนดิจิตอลเป็นอนาล็อกจำนวนของความแตกต่างของระดับแรงดันและกระแสที่สร้างขึ้นที่เอาท์พุทของ DAC จะสัมพันธ์กับจำนวนของบิทที่นำมาเปลี่ยนจากสมการ

$$N = 2^n \tag{2.1}$$

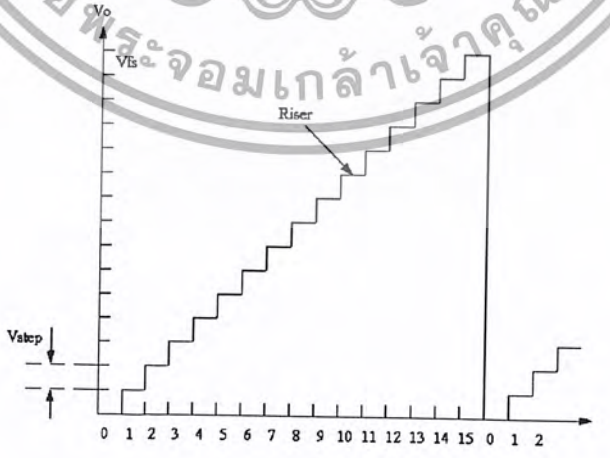
เมื่อ N คือ จำนวนของระดับความแตกต่างด้านเอาท์พุทที่สร้างขึ้นและ n คือจำนวนของบิทอินพุทที่นำมาเปลี่ยนจำนวนของระดับความแตกต่างที่สร้างขึ้นที่เอาท์พุทของ DAC จะขึ้นอยู่กับขอบเขตการจำแนกของอุปกรณ์ที่ใช้งานจำนวนบิทของอินพุทจะใช้บิทที่สูงที่สุดในการคำนวณเช่นอินพุทของ DAC จำนวน 10 บิทสามารถเปลี่ยนระดับสัญญาณได้ 1024 ระดับ การเปลี่ยนแปลงเป็นรูปอื่นเป็นคุณสมบัติหนึ่งที่สำคัญของ DAC ในการนำไปประยุกต์ใช้งานในหลายๆด้านหลักการหนึ่งในการเปลี่ยนแปลงสัญญาณดิจิตอลในรูปของ $N\left(N = 2^n\right)$ และสามารถคิดในรูปของเปอร์เซ็นต์ได้จากสมการ

$$Percent\ Resolution = \frac{1}{2^n} \times 100\% \tag{2.2}$$

เช่น 10 บิท DAC

$$\begin{aligned} Percent\ Resolution &= \frac{1}{2^{10}} \times 100\% \\ &= \frac{1}{1024} \times 100\% \\ &= 0.098\% \end{aligned}$$

จากตัวอย่างของเอาท์พุทของ 10 บิท DAC มีความแน่นอน 0.098 เปอร์เซนต์ของเอาท์พุทสูงสุด (Full Scale) ซึ่งคือระดับแรงดันหรือกระแสที่สร้างขึ้นที่เอาท์พุทของ DAC ที่สมมติขึ้นว่าเลข 1 ไบนารี่ที่เป็นอินพุทแต่ละตัวเปลี่ยนแปลงเป็นรูปอื่นไม่ได้จำกัดแต่ในความเป็นจริง DAC ไม่สามารถมีจำนวนถึง Ideal Full Scale เนื่องจากการจำกัดจำนวนของอินพุท ตัวอย่างเช่น DAC ที่แสดงในรูป 2.8 มีอินพุท 4 เส้นกราฟของ *Vo* และอินพุทไบนารี่สำหรับ 4 บิท DAC สามารถสร้างได้ดังแสดงใน รูปที่ 2.9



รูปที่ 2.9 คุณสมบัติการแปลง DAC 4 บิท

จะสังเกตได้ว่ามีระดับความแตกต่างของแรงดันที่เป็นไปได้ 16 ระดับและ 15 ระดับของขาขึ้นถ้าเป็นเอาท์พุท Full Scale จะมีขอบขาขึ้น 16 ขอบซึ่งหมายถึงว่าค่า Vo สูงสุดเอาท์พุทจะไม่ถึง Vfs อีกหนึ่งขั้นขนาดของเอาท์พุทหนึ่งขั้นเรียกว่า 1 LSB ของดิจิตอลอินพุทเปลี่ยนสภาวะการเพิ่มขึ้นของเอาท์พุท (แรงดันหรือกระแส) สำหรับแต่ละขั้นหาได้จากจำนวนของขั้นและ Vfs ซึ่งมีความสัมพันธ์กับดังนี้

$$\text{ขนาดขั้น} = \frac{Vfs}{2^n} \tag{2.3}$$

เมื่อ n คือจำนวนของอินพุทไบนารี่และ Vfs คือแรงดัน Full Scale ของ Ideal DAC เช่น Vo ของ 4 บิท (DAC) เป็นไปตามทฤษฎีมี Vfs เท่ากับ $10V$ และอินพุทไบนารี่ 12 ฐาน 10 ได้ Vo เท่ากับ

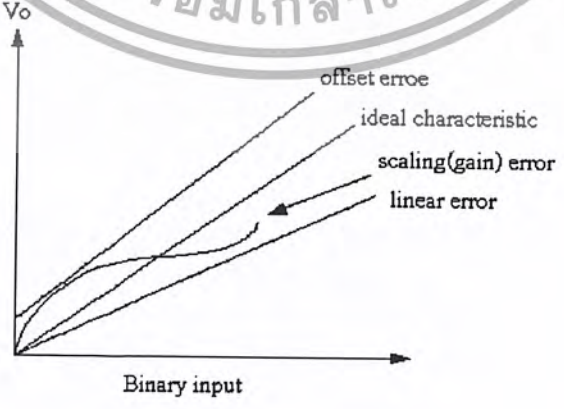
$$\text{ขนาดขั้น} = \frac{Vfs}{2}$$

$$= 10\text{V}/16$$

$$\text{เพราะฉะนั้น } Vo = 0.625\text{V} \times 12$$

$$= 7.5\text{V}$$

การจำแนกของ (DAC) จะใช้เป็นตัวบอกความเที่ยงตรงของศักย์เพราะว่าการจำแนกเป็นตัวกำหนดข้อจำกัดของความเที่ยงตรงของการเปลี่ยนแปลง อย่างไรก็ตามความเที่ยงตรงและการจำแนกไม่ใช่สิ่งเดียวกันตัวอย่างเช่น 16 บิท (DAC) จะพิจารณาถึงการจำแนกสูงสุด (65,536) แต่ไม่ใช่สิ่งจำเป็นที่ถูกต้องในการ หาค่า Vo ซึ่งจะหาได้จากค่าอินพุทที่ให้มาภายใต้เงื่อนไขอุดมคติเอาท์พุทของ (DAC) จะมีความถูกต้อง $+\frac{1}{2}Vstep$ (หรือ $+\frac{1}{2}LSB$) อย่างไรก็ตามอาจมีความผิดพลาดได้ใน DAC แต่ละชนิด ความคลาดเคลื่อนบนเอาท์พุทบนตัวคอนเวอร์เตอร์ แสดงดังรูป 2.10 เป็นรูปผลของการเปลี่ยนแปลงความคลาดเคลื่อนของทรานเฟอร์ฟังก์ชั่นของ DAC อุดมคติ



รูปที่ 2.10 กราฟของ DAC อุดมคติและผลของความคลาดเคลื่อน

OFFSET ERROR เป็นผลที่เกิดขึ้นที่เอาท์พุทของ DAC ไม่เป็น 0 เมื่ออินพุทไบนารี่เป็น 0 ทำให้เกิดค่าคงที่เลื่อนให้ Vo ให้เกิดย่านของไบนารี่อินพุท

GAIN ERROR หรือเรียกอีกอย่างหนึ่งว่า Scaling Error จะสร้างขนาดขึ้นให้ใหญ่กว่าหรือเล็กกว่าขนาดปกติซึ่งเป็นสาเหตุให้ค่า Vo เบี่ยงเบนจากค่าความเป็นจริงของไบนารี่อินพุท

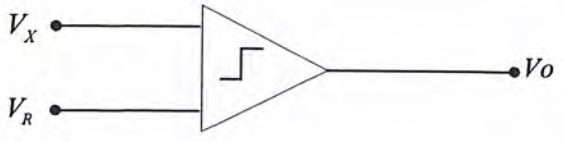
LINEARITY ERROR เป็นความคลาดเคลื่อนอีกชนิดหนึ่งที่เป็นสาเหตุทำให้ DAC ไม่เป็นเชิงเส้น ตัวอย่างเช่นถ้าเกณฑ์ของ DAC ไม่คงที่สำหรับไบนารี่เอาท์พุทจะเปลี่ยนแปลงขนาดของขั้นที่สร้างขึ้น

คุณลักษณะของ DAC ที่สำคัญอีกอย่างคือความสัมพันธ์เกี่ยวกับเวลาที่ใช้ในการเปลี่ยนแปลงคุณสมบัตินี้เรียกว่า Setting Time เป็นการวัดการตอบสนองทางด้านความเร็วของ DAC

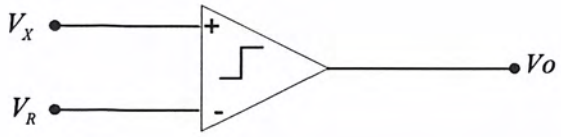
2.4 การเปลี่ยนสัญญาณอนาล็อกเป็นดิจิตอล (Analog to Digital Converters)

รูปแบบสัญญาณไฟฟ้าที่เราพบเห็นและคุ้นเคยในชีวิตประจำวัน จะอยู่ในรูปแบบของสัญญาณที่ต่อเนื่องกันหรือเรียกว่าสัญญาณอนาล็อกซึ่งแต่เดิมการนำเอาสัญญาณไฟฟ้าดังกล่าวมาประมวลผล (Process) ใช้ในรูปแบบที่มีประโยชน์จะกระทำในรูปแบบอนาล็อกนั่นเอง แต่เมื่อเทคนิคการประมวลผลสัญญาณทางดิจิตอลได้รับการพัฒนาขึ้นมาเนื่องจากพบว่าในรูปแบบของดิจิตอล การประมวลผล เก็บสื่อสารและแสดงผล จะกระทำได้ง่ายและมีประสิทธิภาพมากกว่า ดังนั้นการเปลี่ยนรูปแบบสัญญาณ (Convertsion) จึงได้มีความจำเป็นขึ้น จากสัญญาณอนาล็อกที่มีอยู่ตามธรรมชาติถูกเปลี่ยนมาเป็นสัญญาณดิจิตอลโดย Analog to Digital Converters (ADC) และประมวลผลโดยตัวประมวลผลทางดิจิตอล (Digital Processors) เช่น คอมพิวเตอร์ จากนั้นจะถูกนำมาแสดงผลหรือถูกเปลี่ยนกลับมาอยู่ในรูปอนาล็อกและใช้งานได้ง่ายกว่าโดยใช้ Digital to Analog Converters(DAC)

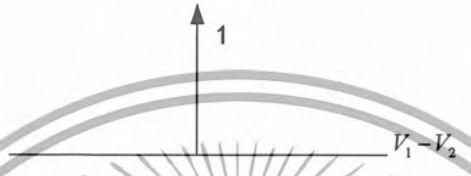
วิธีการแปลงสัญญาณอนาล็อกเป็นดิจิตอลแบบง่ายๆ โดยใช้หลักการของวงจรคอมพาราเตอร์ แรงดันอินพุทที่ไม่ทราบค่า V_X จะต่อเข้ากับขาอินพุทขาหนึ่งของอนาล็อกคอมพาราเตอร์และแรงดันอ้างอิงที่ขนาดแปรตามเวลา V_R ต่อเข้ากับอีกอินพุทของคอมพาราเตอร์ลักษณะของทรานเฟอร์ฟังก์ชั่นของคอมพาราเตอร์ แสดงในรูปที่ 2.12 ถ้าแรงดันอินพุท V_1 มากกว่าอินพุท V_2 แล้วแรงดันเอาท์พุทจะเป็น “1” ถ้า V_1 น้อยกว่า V_2 เอาท์พุทจะเป็น “0”



รูปที่ 2.11 วิธีการพื้นฐานของ ADC



$$Vo = 1 \ , \ V_1 > V_2$$
$$Vo = 0 \ , \ V_1 < V_2$$



รูปที่ 2.12 ทรานเฟอร์ฟังก์ชั่นของคอมพาราเตอร์

วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (ADC) ที่ใช้งานทั่วๆ ไปมีหลายชนิดเช่น Counter type ADC, Integrating ADC, Sucessive Appoximation ADC หรือ Parallel (Flash) ADC เป็นต้น

2.4.1 ทฤษฎีการสุ่มตัวอย่าง Sampling

ในการแปลงสัญญาณอนาล็อกเป็นดิจิตอลนั้น วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอลจะต้องใช้เวลาช่วงหนึ่งในการจัดการ ซึ่งช่วงเวลาดังกล่าวขึ้นอยู่กับหลายตัวแปร เช่น

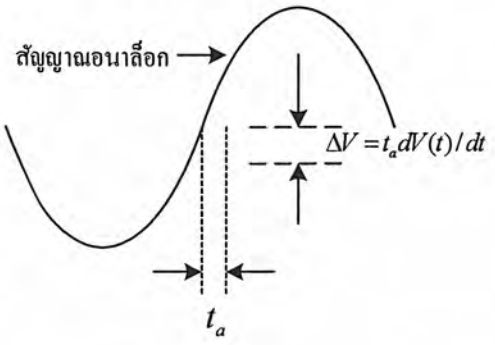
1.ความละเอียดของการเปลี่ยนสัญญาณ

2.เทคนิคของการแปลงสัญญาณ

3.ความเร็วในการทำงานของอุปกรณ์ร่วมอื่นๆ ความเร็วของการแปลงสัญญาณนี้จำเป็นสำหรับใช้งานเฉพาะอย่างและความแม่นยำที่ต้องการ

Aperture Time คือช่วงเวลาในการแปลงสัญญาณ ซึ่งคำว่า Aperture Time โดยทั่วไปหมายถึงช่วงเวลาที่เกิดความไม่แน่นอนในการวัดและผลก็เกิดความผิดพลาด (Error) ต่อค่าที่วัดได้

ในรูปที่ 2.13 สัญญาณอนาล็อก $V(t)$ มีอัตราการเปลี่ยน dV/dt ช่วง Aperture Time t_a ดังนั้นช่วงการเปลี่ยนแปลงสัญญาณอนาล็อกจะเท่ากับ V โดย $\Delta V = t_a dV(t)/dt$



รูปที่ 2.13 แสดงความผิดพลาดจากการวัดใน Aperture Time

ดังนั้นหากเวลาที่วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล ใช้ในการเปลี่ยนสัญญาณในเวลา t_a นี้รหัสของสัญญาณดิจิตอลที่ได้อาจจะตรงกับขนาดของสัญญาณอนาล็อกค่าใดค่าหนึ่งในช่วงนี้ และส่วนอื่นๆที่เหลือคือความผิดพลาดที่เกิดขึ้น ซึ่งแน่นอนในบางครั้งเป็นไปได้ที่รหัสของสัญญาณดิจิตอลจะตรงกับค่าของสัญญาณอนาล็อกที่ถูกต้อง

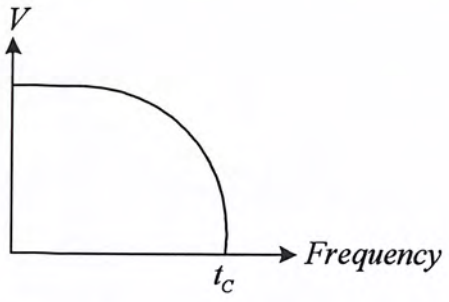
2.4.2 Sample and Hold และ Aperture error

วงจร Sample and Hold จะทำการสุ่มสัญญาณอินพุทและนำสัญญาณนั้นมาเก็บไว้ในเวลาหนึ่งได้ ซึ่งส่วนใหญ่จะใช้การประจุแรงดันนั้นในตัวเก็บประจุที่รั่วไหลค่าหนึ่ง ดังนั้นในเมื่อแรงดันอินพุทสามารถคงอยู่ได้นานพอ ทำให้วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลไม่จำเป็นต้องมีเวลาในการแปลง (Conversion Time) อย่างรวดเร็วนัก Aperture Time ของ Sample and Hold คือเวลาตั้งแต่เริ่มสุ่มสัญญาณจนตัวเก็บประจุมีค่าแรงดันจนถึงค่าที่สุ่มซึ่งสำหรับ Sample and Hold แล้ว Aperture Time ขึ้นอยู่กับ Bandwidth และ Switching Time ของอุปกรณ์แอคทีฟที่ใช้ในวงจรซึ่งหาและสร้างได้ง่ายและราคาถูกกว่าการสร้างวงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลความเร็วสูง

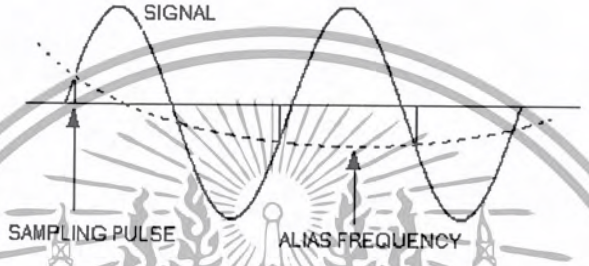
มีปัญหาว่าอัตราการสุ่มสัญญาณนั้นควรจะมีค่าเท่าใดที่จะไม่ทำให้ข้อมูลสูญเสียไปเมื่อสัญญาณนั้นถูกเปลี่ยนกลับมาเช่นเดิมอันนี้ขึ้นอยู่กับความถี่ของสัญญาณอนาล็อก และทฤษฎีการสุ่มที่กล่าวไว้ว่า "ถ้าสัญญาณต่อเนื่องซึ่งมีความถี่และฮาร์โมนิกส์ ไม่เกิน f_c แล้วสัญญาณดังกล่าวจะสามารถเปลี่ยนกลับมา ได้อย่างเดิมได้โดยไม่สูญเสียรายละเอียดหรือเพี้ยนไปถ้าอัตราการสุ่มไม่น้อยกว่า $2f_c$ ต่อวินาที"

2.4.3 Frequency folding and Aliasing

จากทฤษฎีการสุ่มสามารถอธิบายลักษณะรูปสเปกตรัม (Spectum) ของสัญญาณในรูปที่ 2.13 แสดงให้เห็นสเปกตรัมของสัญญาณที่ถูกสุ่มซึ่งแบนวิดท์ไม่เกินกว่า f_c ในขณะที่สัญญาณนี้จะถูกสุ่มด้วยความถี่ f_s ขบวนการมอดูเลชั่น(Modulation) จะทำให้แถบสเปคตรัมของสัญญาณสุ่มขนาดกว้างออกจาก f_s เป็น $2f_s, 3f_s, \ldots$ ได้ดังรูปที่ 2.14



รูปที่ 2.14 แสดง Spectrum ของสัญญาณอนาล็อกที่ถูกสุ่ม



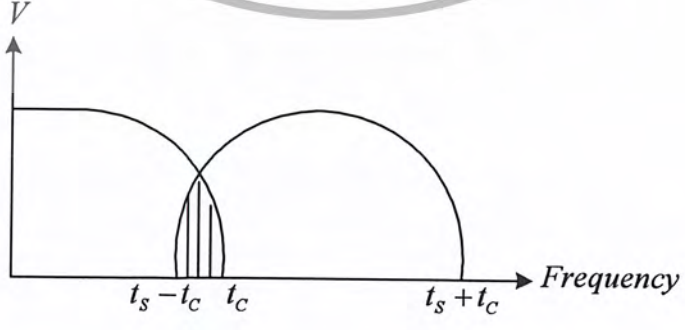
รูปที่ 2.15 หลังจากการสุ่มเกิด Frequency Folding

ถ้าความถี่ของสัญญาณสุ่ม ไม่สูงพอหลังจากการสุ่มสเปกตรัมบางส่วนของ f_s จะมาซ้อนทับกับสเปกตรัมของสัญญาณ ซึ่งเรียกว่า Frequency Folding หากเป็นเช่นนี้ก็จะทำให้เกิดความเพี้ยนแก่สัญญาณอนาล็อกจากการซ้อนกันของสเปกตรัม เมื่อสัญญาณถูกเปลี่ยนกลับให้อยู่ในรูปเดิม และถ้าเลื่อนความถี่ของการสุ่มสูงขึ้นจนโอกาสการซ้อนกันของสเปกตรัมหมดไป $(f_s - f_c) = f_c$ จะทำให้การเปลี่ยนกลับของสัญญาณหลังจากถูกสุ่มก็ยังคงเหมือนเดิมได้

จากที่กล่าวมาแสดงการสนับสนุนทฤษฎีการสุ่มที่ว่าให้ $f_s < 2f_c$ นั้นคือการกำจัดการซ้อนกันของสเปกตรัมได้สองวิธีคือ

1.ใช้อัตราการสุ่มที่สูงพอ

2.การทำการกรองความถี่ของสัญญาณอนาล็อกก่อนการสุ่มเพื่อให้ Bandwidth ไม่เกินไปกว่า $f_s/2$



รูปที่ 2.16 การเกิด Alias Frequency Folding จากการสุ่มด้วยความถี่ต่ำกว่า 2 เท่าของสัญญาณอนาล็อก

ในทางปฏิบัติแล้วจะยังคงเกิด Frequency Folding ได้เสมอจากส่วนฮาร์โมนิคของสัญญาณรบกวนรวมทั้งสเปกตรัมของสัญญาณรบกวนที่ยังคงอยู่แม้ว่าจะทำการกรองความถี่ก่อนหน้ามาแล้วก็ตาม การกำจัดการซ้อนกันของสเปกตรัมนี้ วิธีที่ได้ผลก็คือพยายามให้การสุ่มสัญญาณเป็นไปอย่างรวดเร็วมากที่สุด

ผลของการใช้อัตราการสุ่มที่ไม่เหมาะสม อีกประการหนึ่งเกิดขึ้น ดังรูปที่ 2.16 เรียกว่า Alias Frequency ซึ่งเกิดกับสัญญาณที่เปลี่ยนกลับมาเช่นเดิมหลังจากถูกสุ่มแล้ว

2.4.4 วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (Analog to Digital Converter)

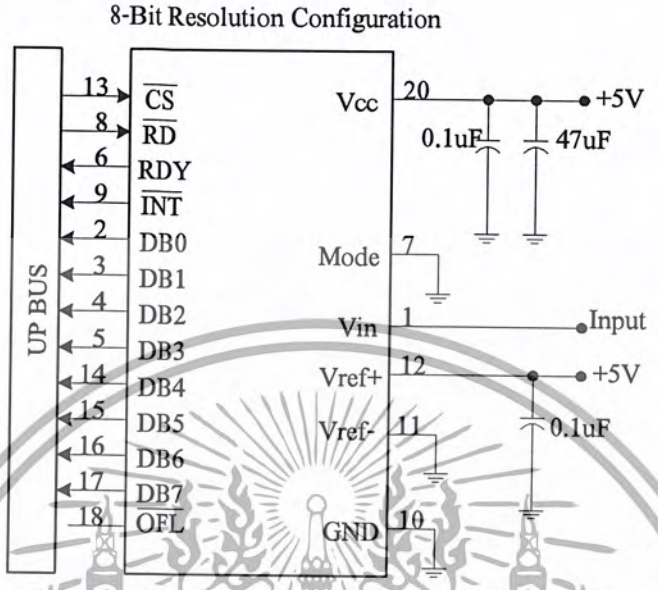
ในการติดต่อสื่อสารกันระหว่างระบบอนาล็อกและระบบดิจิตอลนี้เป็นไปไม่ได้เลยที่จะให้สัญญาณอนาล็อกเข้ามาทำงานในระบบดิจิตอล ดังนั้นจึงต้องมีการแปลงสัญญาณอนาล็อกให้เป็นดิจิตอล ส่วนที่ทำหน้าที่ดังกล่าวนี้เรียกว่า "วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล" วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลแบบใช้วงจรเปรียบเทียบหรือแบบ "แฟลช"

สำหรับแหล่งสัญญาณที่ต้องการความรวดเร็วสูงมากๆเช่น การแปลงสัญญาณภาพโทรทัศน์เรดาห์จำเป็นต้องใช้วงจรแปลงสัญญาณอนาล็อกเป็นสัญญาณดิจิตอลแบบพิเศษ ที่เรียกว่า Flash (Parallel) ADC

วงจรแบบนี้เป็นแบบที่นิยมใช้กันมากเนื่องจากมีความเร็วในการทำงานสูงมาก จะเห็นว่าการทำงานของแบบนี้จะประกอบไปด้วยตัวต้านทานต่อเป็นตัวแบ่งแรงดัน มีออปแอมป์ต่อเป็นวงจรเปรียบเทียบและมีวงจรถอดรหัสซึ่งเป็นตัวกำหนดจำนวนของตัวเลขฐานสองที่จะใช้เป็นเอาท์พุทในรูปจะให้เอาท์พุทออกมาเพียง 4 บิทเท่านั้น

หลักการทำงานก็ คือ จะใช้คอมพาราเตอร์ในการเปรียบเทียบสัญญาณอนาล็อกอินพุทกับแรงดันอ้างอิงที่แบ่งแรงดันให้สอดคล้องกับรหัสดิจิตอลโดยใช้ตัวต้านทานและแปลงเอาท์พุทจากคอมพาราเตอร์ให้ตรงกับรหัสดิจิตอล ซึ่งจะเห็นว่าอุปสรรคทางด้านความเร็วจะถูกกำจัดเพียง Propagation time ของคอมพาราเตอร์เท่านั้น แต่อุปสรรคที่สำคัญต่อการพัฒนาวงจรบนชิพไอซี คือ วงจรนี้ต้องการคอมพาราเตอร์ $2^n - 1$ ตัวซึ่งเป็นจำนวนที่มากพอสมควร

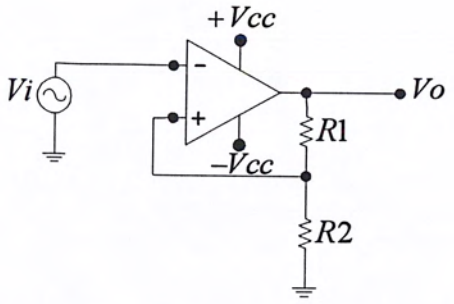
วงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล จะใช้ไอซีเบอร์ ADC0820 เป็นแบบ 8 บิท และมีอัตราการสุ่มสูงสุดได้ที่ 40MHz



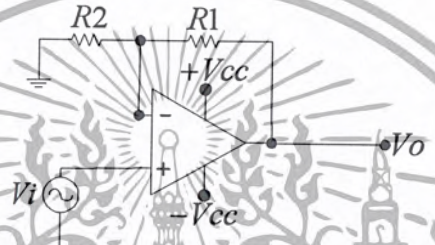
รูปที่ 2.17 วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล

2.5 ออปแอมป์กับวงจรการป้อนกลับแบบบวก (Positive Feedback)

วิธีที่ง่ายที่สุดในการป้อนให้แรงดันอินพุทเป็นแบบฮิสเตอร์รีซีส(Hysteresis) กับวงจรเปรียบเทียบแรงดัน (Comparator) ก็คือ ด้วยการแบ่งแรงดันส่วนหนึ่งจากเอาท์พุทแล้วนำป้อนเข้ากับอินพุทโดยจะต่อแบบโดยตรง หรือผ่านค่าความต้านทานและตัวเก็บประจุก่อนก็ได้ ซึ่งถ้าหากนำเอาแรงดันที่ป้อนกลับนั้นไปต่อเข้ากับขั้วอินพุทบวกของออปแอมป์ ก็จะเรียกว่าการป้อนกลับแบบบวก (Positive Feedback) ดังแสดงใน รูปที่ 2.18(ก) หรือถ้านำแรงดันป้อนกลับไปต่อเข้ากับขั้วอินพุทลบก็จะเรียกว่าการป้อนกลับแบบลบ (Negative Feedback) ตาม รูปที่ 2.18(ข) แต่สำหรับการนำไปใช้งานนั้น การป้อนกลับแบบลบ



(ก) การป้อนกลับแบบบวก (Positive Feedback)

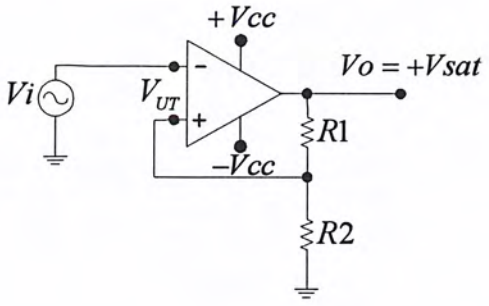


(ข) การป้อนกลับแบบลบ (Negative Feedback)

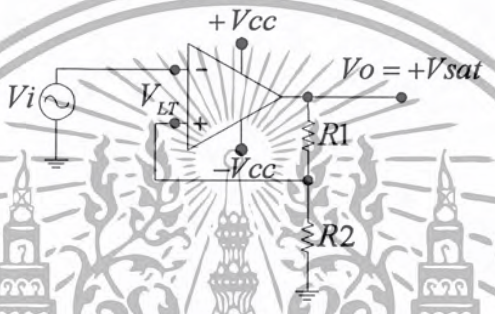
รูปที่ 2.18 วงจรการป้อนกลับ

จะให้เสถียรภาพของการทำงานดีกว่าแบบการป้อนกลับแบบบวกดังจะได้กล่าวต่อไป แต่ถึงอย่างไรก็ตามการป้อนกลับแบบบวก ก็จะยังให้ผลที่ดีในเรื่องของฮิสเตอร์รีซีส (Hysteresis) การป้อนกลับเพื่อให้เกิดแรงดันฮิสเตอร์รีซีสที่อินพุท

ดังได้กล่าวมาแล้วว่าวิธีการป้อนแรงดันฮิสเตอร์รีซีสนั้นสามารถจะกระทำได้โดยการนำเอาวงจรแบ่งแรงดันมาต่อเข้ากับเอาท์พุท เพื่อแบ่งแรงดันนี้มาส่วนหนึ่งแล้วป้อนกลับเข้ากับอินพุทของออปแอมป์ ดังรูปที่ 2.19(ก) ซึ่งเป็นการป้อนกลับแบบบวก



(ก) UPPER THRESHOLD (V_{UT})



(ข) LOWER THRESHOLD(V_{LT})

รูปที่ 2.19 แสดงวิธีการป้อนกลับด้วยวงจรแบ่งแรงดันเพื่อทำให้เกิดแรงดันฮิสเตอร์รีซีสกับอินพุท

ในตราบใดที่แรงดันอินพุท Vi ยังไม่มีการเคลื่อนที่ตัดแกนระดับแรงดันอ้างอิง ($Vref$) จะเป็นค่าเท่ากับ $0V$, V_{LT} หรือ V_{UT} ก็ตาม แต่แรงดันเอาท์พุทก็จะยังคงสถานะเดิมอยู่ตลอดไป ให้ย้อนไปดูวงจรตาม รูปที่ 2.19 ซึ่งเป็นวงจรขยายอินเวอร์ติงและมีแรงดันอ้างอิง $Vref = 2V$ ป้อนที่อินพุทบวก อยู่ ดังนั้นถ้าแรงดันถ้าแรงดันอินพุท Vi – มีค่าน้อยกว่าแรงดันอ้างอิง $Vref = 2V$ แล้วเอาท์พุท Vo ก็จะมีค่าเท่ากับ $+Vsat$ ตลอด หรือจะดูได้จากตาราง รูปที่ 2.3 ดังนั้นแรงดันเอาท์พุทนี้ก็จะไปตกคร่อมกับวงจรแบ่งแรงดันและได้แรงดันส่วนหนึ่งที่ตกคร่อมค่าความต้านทาน $R2$ แล้วป้อนกลับเข้าขั้วอินพุทบวก ดังรูปที่ 2.19(ก) ซึ่งสามารถคำนวณได้จากสมการ

$$(V_{UT}) = \frac{R_2}{R_1 + R_2}(+Vsat) \tag{2.9}$$

ค่าแรงดันนี้ก็จะกลายเป็นแรงดันอ้างอิงค่าบวกที่เกิดจากเอาท์พุท $Vo = +Vsat$ และเรียกแรงดันนี้ว่า UPPER THRESHOLD VOLTAGE (V_{UT}) ซึ่งถ้าหากแรงดันอินพุท Vi มีค่าเพิ่มขึ้นและตัดกับแกน

ระดับแรงดัน (V_{UT}) นี้เอาท์พุท Vo ก็เปลี่ยนสถานะมาเป็น $-Vsat$ (ตาม รูปที่ 2.19(ข)) โดยจะคงค่านี้อยู่ตลอดเวลาตราบที่อินพุท Vi ยังมีค่ามากกว่าแรงดัน (V_{UT})

ดังนั้นในขณะเมื่อเอาท์พุท $Vi = -Vsat$ t สถานะภาพของการป้อนกลับก็จะได้ตามรูปที่ 2.19(ข) และค่าแรงดันป้อนกลับที่ต่อเข้ากับอินพุทบวกนั้นจะได้ตามสมการข้างล่างนี้

$$V_{H} = V_{UT} - V_{LT} \tag{2.10}$$

แทนค่าแรงดัน V_{UT} และ V_{LT} จาก สมการที่ 2.9 และ 2.10 ลงใน สมการที่ 2.8

$$V_{H} = \frac{R_2}{R_1 + R_2} Vsat - \frac{R_2}{R_1 + R_2}(-Vsat) \tag{2.11}$$

$$V_{H} = \frac{2R_2}{(R_1 + R_2)} Vsat$$

ดังนั้นค่าความสัมพันธ์ระหว่าง R_1 กับ R_2 ก็สามารถจะทำให้แรงดันฮิสเตอร์รีซีสมีค่าเปลี่ยนแปลงได้ ยกตัวอย่างเช่น ถ้าหากค่าความต้านทาน R_2 ลดลงในขณะที่ R_1 คงที่แล้ว ผลที่ตามมาก็คือจะทำให้แรงดันฮิสเตอร์รีซีสลดลงตาม ซึ่งจากนั้นก็จะเป็นผลเสียต่อการลดสัญญาณรบกวนให้กับวงจรดังได้กล่าวมาแล้ว

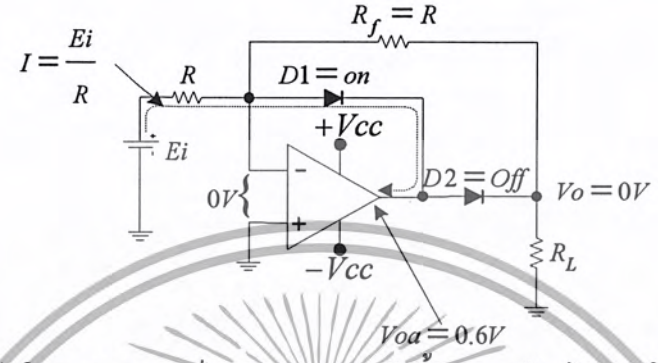
2.6 วงจรฮาล์ฟเวฟเร็กติไฟเออร์

วงจรกลับทิศทางรูปคลื่น หรือฮาล์ฟเวฟเร็กติไฟเออร์นี้ จะสามารถนำกระแสได้เพียงครึ่งลูกคลื่นสัญญาณไฟฟ้าเท่านั้นซึ่งอาจจะเป็นเฉพาะครึ่งลูกคลื่นบวกหรือไม่ก็อาจเป็นครึ่งลูกคลื่นลบอย่างใดอย่างหนึ่งก็ได้ส่วนที่เหลืออีกครึ่งลูกคลื่นจะถูกกั้นไม่ให้กระแสไหลผ่าน

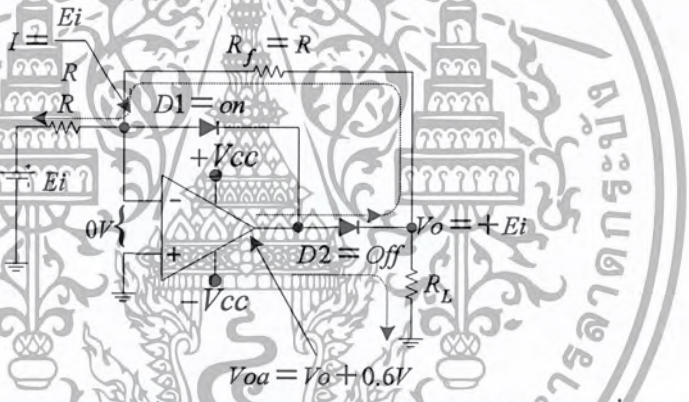
2.6.1 ชนิดเอาท์พุทเป็นบวกโดยใช้วงจรขยายอินเวอร์ติ้ง

วงจรขยายอินเวอร์ติ้งจะทำการกลับขั้วสัญญาณที่ได้จากไดโอดสองตัวดังรูปที่ 2.20 ทำให้ได้วงจรฮาล์ฟเวฟเร็กติไฟร์ที่สามารถทำงานได้ตามอุดมคติเมื่อ Ei เป็นบวกดังรูปที่ 2.20(ก) ไดโอด $D1$ จะนำกระแสทำให้เกิดแรงดันเอาท์พุทของออปแอมป์ Voa มีค่าเป็นลบ(ที่ขาอินพุททั้งสองของไดโอดจะยอมให้กระแสไหลผ่านน้อยมากกระแสจึงไหลไปยังจุดอื่นมากกว่า) หรือประมาณ –0.6 โวลท์ อนึ่งในทางปฏิบัติจริงแทบจะไม่มีกระแสไหลผ่าน Rf เลยเนื่องจากค่า Rf มีค่าความต้านทานมากกว่าความต้านทานของไดโอด $D1$ (ในขณะนั้น) กระแสเกือบทั้งหมดจะไหลผ่าน $D1$ และ Vo จึงเท่ากับ 0 โวลท์ตามเดิม

ใน รูปที่ 2.20(ข) หรือเมื่อแรงดันอินพุทเป็นลบจะทำให้ค่า Voa มีค่าเป็นบวก $D2$ จะนำกระแสซึ่งไหลผ่าน Rf วงจรในขณะนี้จึงเหมือนกับวงจรขยายแบบอินเวอร์ติ้งโดย $R = Rf$ และ $Vo = -(-Ei) = +Ei$ สำหรับกระแส I ที่ได้จะสามารถคำนวณได้จากค่าของ Ei/R และอัตราขยายที่ได้มีค่าเท่ากับ $-Rf/Ri$



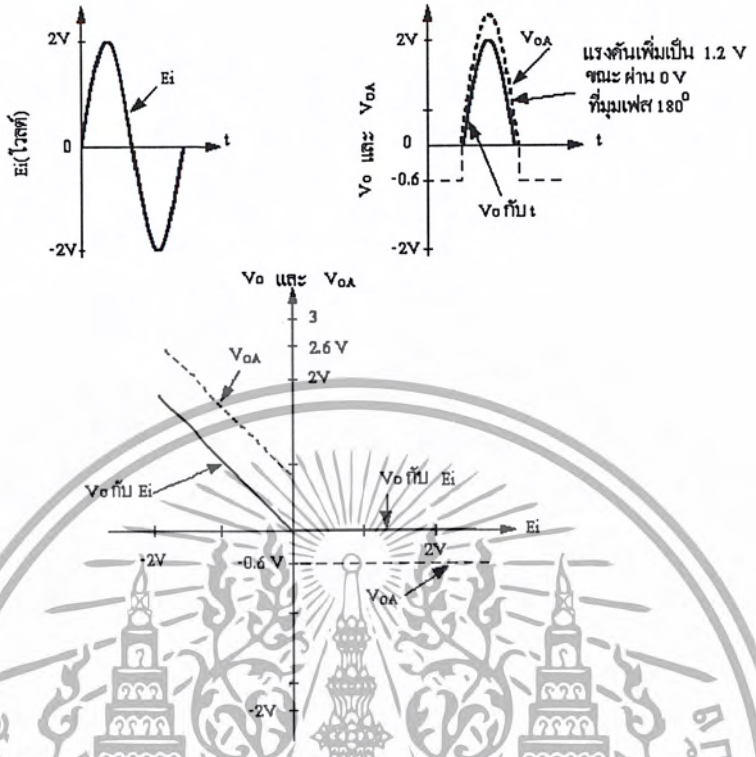
(ก) แรงดันอินพุทของวงจรเป็นบวก Vo ถูกสกัดกั้นกระแสโดยไดโอดจึงมีค่าเท่ากับ 0 โวลท์



(ข) เมื่อแรงดันอินพุทของวงจรเป็นลบ Vo จะมีค่าเป็นบวกและเท่ากับ Ei ที่เป็นลบ

รูปที่ 2.20 วงจรอินเวอร์ติ้งฮาล์ฟเวฟเร็กติไฟเออร์

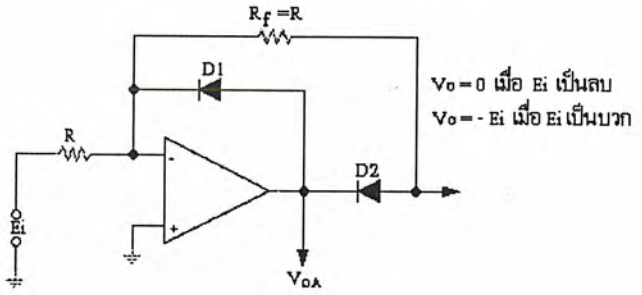
รูปคลื่นที่เกิดจากวงจรนี้จะเป็นดังรูป 2.21 โดยค่า Vo ที่เป็นลูกคลื่นจะมีค่าได้เฉพาะที่เป็นบวก ในรูปที่2.20(ข) จะเป็นการแสดงค่าเทรสโฮลหรือค่าเริ่มต้นเปลี่ยนแปลงสถานะของซิลิคอนไดโอดทั่วไป ซึ่งจำเป็นต้องใช้แรงดันค่าหนึ่งจึงจะสามารถทำงานได้อย่างไรก็ตามเมื่อเราใช้วงจรป้อนกลับของออปแอมป์จะทำให้สามารถแก้ปัญหาเรื่องค่าเทรสโฮลนี้ได้ซึ่งสามารถทำให้ไดโอดกลับทิศทางไฟฟ้าได้ แม้ว่าแรงดันอินพุทที่ได้จากวงจรอยู่ในช่วง –0.6 ถึง +0.6 โวลท์



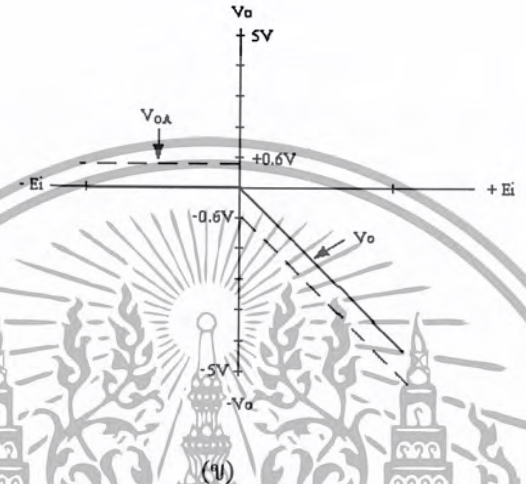
รูปที่ 2.21 แสดงรูปคลื่นที่ได้จากวงจรอินเวอร์ติ้งฮาล์ฟเวฟเรกติไฟเออร์

2.6.2 ชนิดเอาท์พุทลบโดยใช้วงจรขยายอินเวอร์ติง

ไดโอดในรูปที่ 2.22 เมื่อนำมากลับทิศทางดังรูปที่ 2.22(ก) จะพบว่าแรงดันอินพุทมีค่าเป็นบวกเท่านั้นที่สามารถส่งผ่านและถูกกลับขั้วได้ (ได้ค่าแรงดันเอาท์พุทเป็นลบเมื่ออินพุทเป็นบวกนั่นเอง) *Vo* จะมีค่า 0 โวลท์เสมอเมื่อแรงดันอินพุทมีค่าเป็นลบวงจรนี้สามารถสรุปคุณสมบัติของวงจรได้ดังรูปที่2.28 (ข)



(ก)



(ข)

รูปที่ 2.22 การกลับการวางตัวของไดโอด

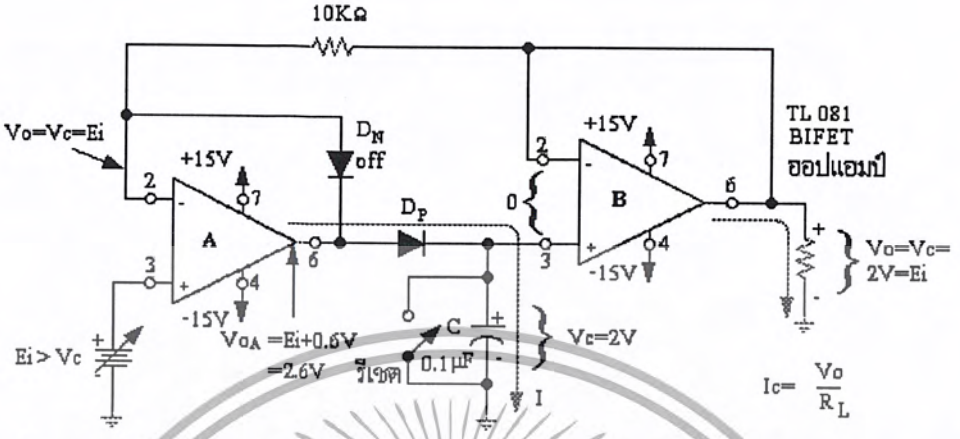
2.7 วงจรตรวจจับแรงดันยอด

โดยทั่วไปจะทำการติดตามแรงดันตลอดเวลาแล้วทำการเก็บแรงดันสูงสุดที่เกิดขึ้น หรือแรงดันยอดเอาไว้ในตัวเก็บประจุถ้ามีแรงดันยอดที่สูงกว่าค่าที่เคยเก็บไว้แล้วตัวเก็บประจุจะทำการเก็บค่าใหม่เข้าไปและคายประจุก็ต่อเมื่อมีการเปิดใช้สวิทซ์กลธรรมดา หรือสวิตซ์อิเล็กทรอนิกส์ดังนั้นวงจรชนิดนี้อาจเรียกอีกอย่างหนึ่งว่าวงจรติดตามแรงดันยอด

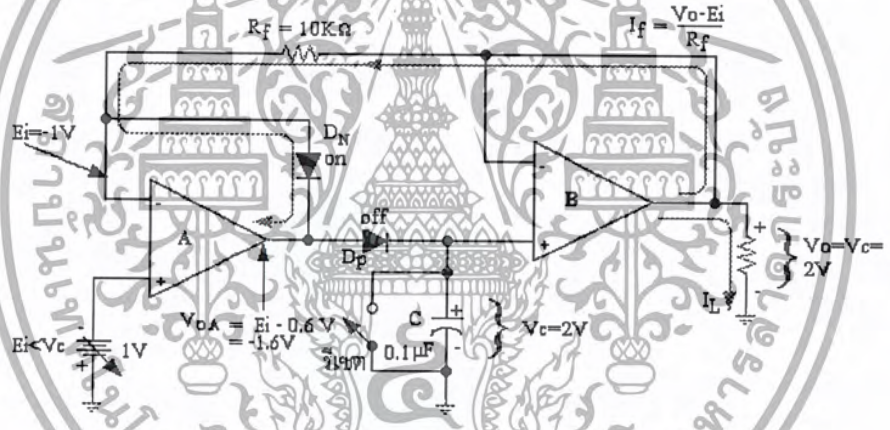
วงจรในรูปที่ 2.23 ประกอบด้วยออปแอมป์ 2 ตัว ตัวต้านทาน 1 ตัว ตัวเก็บประจุและสวิตซ์รีเซ็ตออปแอมป์ *A* จะเป็นวงจรฮาล์ฟเวฟเรกติไฟเออร์ซึ่งจะทำการประจุกระแสให้กับตัวเก็บประจุก็ต่อเมื่อแรงดันอินพุท *E* มีค่ามากกว่า *Vc* ส่วนออปแอมป์ *B* จะเสมือนบัฟเฟอร์ซึ่งจะทำการสะท้อนแรงดันที่เกิดขึ้นที่ตัวเก็บประจุที่ไปยังเอาท์พุทของมัน ดังนั้นแรงดันเอาท์พุทของวงจรนี้จะมีค่าเท่ากับ *Vc* เสมอเนื่องจากความต้านทานอินพุทมีค่าสูงมากของวงจรบัฟเฟอร์ดังนั้นจะไม่มีการคายประจุเกิดขึ้นที่ตัวเก็บประจุ

จาก รูปที่ 2.23(ก) เมื่อมีค่ามากกว่า *Vc* ประมาณ 0.6 โวลท์ ไดโอด *Dp* จะอนุญาตให้กระแสไหลผ่านตัวเก็บประจุได้ แรงดัน *Vc* นี้จะสะท้อนไปที่เอาท์พุทของออปแอมป์ *B* ด้วยและเมื่อแรงดันอินพุท ของวงจรมีค่าต่ำกว่า *Vc* ไดโอด *Dp* จะสกัดไม่ให้กระแสไหลผ่านย้อนตัวมันกระแสก็จะไหลผ่านไดโอด *Dn* จากออปแอมป์ *B* ไปยังออปแอมป์ *A* ดัง รูปที่ 2.23(ข) สำหรับวงจรนี้ข้อควรพิจารณา

ข้อควรพิจารณาเพิ่มก็คือ เราควรใช้ไดโอดที่มีกระแสรั่วต่ำมาก และออปแอมป์ B ควรมีอินพุทอิมพิแดนซ์สูงมาก ตัวเก็บประจุควรเป็นชนิดที่มีกระแสรั่วต่ำมีค่าดูดกลืนของไดอิเล็กตริคต่ำ



(ก) เมื่อ Ei มีค่ามากกว่า Vc ตัวเก็บประจุ C จะทำการประจุกระแสซึ่งไหลจาก Ei ผ่าน Dp



(ข) เมื่อ Ei มีค่าน้อยกว่า Vc ตัวเก็บประจุ C จะคงค่าแรงดันยอดเดิมเอาไว้

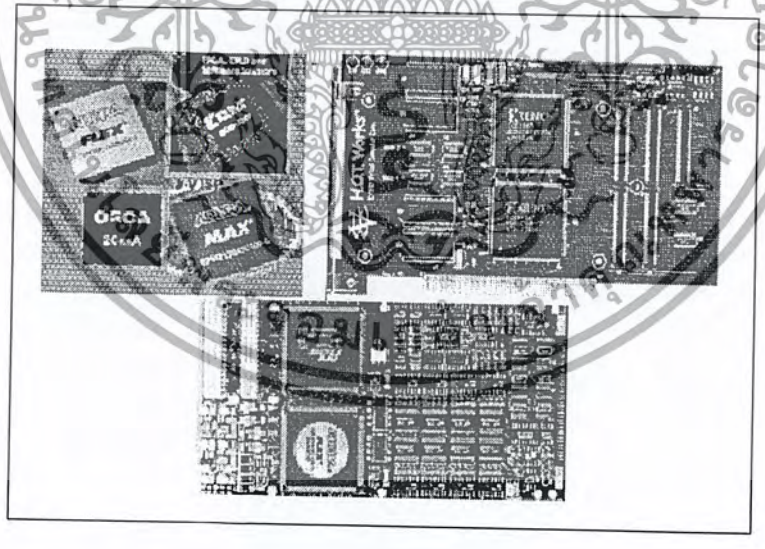
รูปที่ 2.23 วงจรตรวจจับแรงดันยอดชนิดบวกแล้วคงค่าแรงดันยอดเอาไว้

บทที่ 3
การคำนวณและการสร้าง

3.1 การออกแบบวงจรเชิงเลขด้วยอุปกรณ์ FPGA

อุปกรณ์ FPGA (Field Programmable Gate Array) เป็นอุปกรณ์ที่ใช้ในการโปรแกรมวงจรที่ได้ออกแบบลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชั่นการทำงานตามที่ออกแบบไว้ ในการทำ FPGA ซึ่งเป็นวิธีการออกแบบ IC (Integrated Circuit) แบบ Semicustom อีกวิธีหนึ่ง เมื่อเทียบกับการทำ ASICs (Application Specific Integrated Circuits) แล้วนั้นก็มีทั้งข้อดีและข้อเสีย คือ การทำ FPGA จะมีข้อจำกัดในด้านขนาดของวงจรเพราะภายในอุปกรณ์ FPGA จะมีจำนวนเกท (Gate) ให้ใช้จำนวนจำกัดและการทำ FPGA ก็เหมาะสำหรับการทำผลิตภัณฑ์ต้นแบบหรือเพื่อผลิตในปริมาณต่ำ ส่วนข้อดีของการทำ FPGA ก็คือระยะเวลาที่ใช้ในการทำตั้งแต่เขียนรหัส (Code) อธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลด (Download) นั้นน้อยกว่าการทำ ASIC มากและการตรวจสอบหรือแก้ไขการออกแบบก็ทำได้สะดวก

การทำ FPGA ในปัจจุบันมีประสิทธิภาพและความสะดวกมากขึ้น ทั้งนี้ก็เนื่องจากทางบริษัทผู้ผลิตอุปกรณ์ FPGA ได้เพิ่มความสามารถของอุปกรณ์ FPGA โดยเพิ่มจำนวนองค์ประกอบภายในหรือปรับปรุงโครงสร้างสถาปัตยกรรมภายในและยังได้เพิ่มประสิทธิภาพของซอฟต์แวร์ที่ใช้ทำ PPR (Partitioning Placement and Routing) สำหรับอุปกรณ์นั้นๆ ด้วย ลักษณะของตัวFPGAและการนำไปใช้งานแสดงดังในรูปที่ 3.1



รูปที่ 3.1 ลักษณะของตัว FPGA และการนำไปใช้งาน

สำหรับตัวอุปกรณ์ FPGA นั้นก็มีโครงสร้างพื้นฐานเทคโนโลยีที่ใช้สร้างตลอดจนเทคนิควิธีการโปรแกรมที่แตกต่างกันสำหรับผู้ผลิตแต่ละราย นอกจากนั้นอุปกรณ์ FPGA ของแต่ละผู้ผลิตก็มีโครงสร้างและความสามารถที่แตกต่างกันบางส่วน ในการใช้งานนั้นอุปกรณ์ FPGA สามารถนำไปประยุกต์ใช้งานได้

3.2 การออกแบบโดยใช้ภาษาอธิบายการทำงานของฮาร์ดแวร์

ในการออกแบบวงจรเชิงเลขนั้นทำได้โดยการวาดวงจรหรือใช้ภาษาอธิบายฮาร์ดแวร์ในขั้นตอนนี้เป็นขั้นตอนที่ไม่แตกต่างกันระหว่างการออกแบบด้วย FPGA และ ASIC ในกรณีที่ใช้ภาษาอธิบายฮาร์ดแวร์ แต่ในกรณีที่ออกแบบโดยวิธีการวาดวงจรจะแตกต่างกันโดยที่การทำวิธีนี้จะต้องคำนึงถึงเทคโนโลยีที่จะใช้ซึ่งแต่ละเทคโนโลยีก็มีความแตกต่างกันไป จะเห็นได้ว่าการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ทำได้สะดวกกว่าเพราะการทำด้วยวิธีนี้ไม่ต้องคำนึงถึงเทคโนโลยีที่จะใช้ (Technology independence) และที่สำคัญการออกแบบด้วยวิธีนี้สามารถที่จะแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี

ในการเขียนโค้ดสิ่งที่ต้องคำนึงถึงคือเขียนอย่างไรจึงจะสามารถสังเคราะห์เป็นวงจรได้และให้คุณสมบัติของวงจรตามที่กำหนด เพราะลักษณะการเขียนโค้ดจะมีผลโดยตรงกับวงจรที่ได้ เนื่องจากในการสังเคราะห์วงจรนั้นซอร์ฟแวร์สังเคราะห์วงจร (Synthesis Tools) จะทำการสังเคราะห์ตามโค้ดที่เขียน ถ้าอธิบายการทำงานของวงจรเดียวกันแต่เขียนโค้ดในลักษณะที่ต่างกันเมื่อสังเคราะห์แล้วจะได้วงจรที่ต่างกัน และจากวงจรที่ต่างกันเมื่อนำไปทำต้นแบบด้วย FPGA หรือการทำ ASIC แล้วจะได้ไอซีที่มีคุณสมบัติต่างกันทั้งในด้านของขนาดหรือความเร็ว (Area and Time) ส่วนการเขียนโค้ดลักษณะใดเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดนั้นก็ขึ้นอยู่กับประสบการณ์ในการออกแบบ

3.2.1 การจำลองการทำงานของวงจร (Simulation)

ขั้นตอนนี้เป็นขั้นตอนที่สำคัญเพราะเป็นขั้นตอนที่ใช้ตรวจสอบฟังก์ชั่นการทำงานของวงจรว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหน เพื่อที่จะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะใช้ซอร์ฟแวร์สำหรับทำการจำลองการทำงานของวงจร เช่น V-System และ Model Sim ของบริษัท Model Technology

3.2.2 การสังเคราะห์วงจร

ในขั้นตอนนี้จะใช้ซอฟต์แวร์สังเคราะห์วงจร (Synthesis tools) ทำการสังเคราะห์โค้ดเพื่อให้ได้เป็นวงจรขึ้นมา แต่ต้องตรวจสอบด้วยว่าซอฟต์แวร์นั้นๆสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการใช้หรือไม่โดย FPGA ที่นิยมใช้งานเช่นของบริษัท Xilinx ตระกูล XC4000 และบริษัท Altera ตระกูล FLEX 10 K ซอฟต์แวร์สังเคราะห์วงจรที่นิยมใช้เช่นโปรแกรม Leonardo Spectrum ของบริษัท Exemplar Logic ซึ่งในขั้นตอนนี้ซอฟต์แวร์สังเคราะห์วงจรจะแปลงโค้ดและทำการออปติไมซ์ (Optimization) เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ นอกจากนี้ยังสามารถกำหนดข้อบังคับสำหรับวงจรได้เช่น ข้อบังคับในเรื่องของเวลา (Time Constraints) หรือข้อบังคับในเรื่องของพื้นที่ ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอนออปติไมซ์เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการออปติไมซ์คือการเทียบ (Mapping) วงจรให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างสถาปัตยกรรมภายในอุปกรณ์ FPGA ในกรณีของ Xilinx ตระกูล XC4000 และ Altera ตระกูล FLEX 10 K จะเทียบโดยใช้วิธี LUT (Look Up Table) เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอฟต์แวร์สังเคราะห์วงจรก็จะมีการรายงานผลว่าวงจรที่ออกแบบไปนั้นเป็นอย่างไร เช่น มีความหน่วง (Delay) เท่าไร ใช้ทรัพยากรต่างๆใน FPGA อะไรบ้าง เป็นต้น

3.2.3 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ให้เป็นส่วนย่อยๆ สำหรับลงใน CLBs, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้เพื่อช่วยลดความหนาแน่นในตอนทำการเชื่อมต่อสัญญาณ (Routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำ โดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจรเช่น เกท (Gate), ฟลิปฟลอป (Flipflop) ลงในทรัพยากรต่างๆ ที่มีอยู่ภายในอุปกรณ์ FPGA (CLBs, IOBs, BUFT และ Edge Decoder) หลังจากทำขั้นตอนนี้เสร็จแล้วสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนซอฟต์แวร์ที่ใช้ในขั้นตอนนี้ขึ้นอยู่กับตัว FPGA ที่ใช้งานเช่น FPGA ของบริษัท Xilinx จะใช้ Xilinx Foundation Series 2.1i ซึ่งซอฟต์แวร์ตัวนี้จะรวมเอาซอฟต์แวร์ย่อยอื่นๆอีก เพื่อให้การทำ PPR (Partitioning, Placement and Routing) เป็นไปอย่างต่อเนื่อง ส่วน FPGA ของบริษัท Altera จะใช้Altera MAX+II

3.2.4 การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Partitioning) มาแล้วว่าควรจะอยู่ในตำแหน่งใดในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่นวงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทาง (Route) ได้ง่ายหรือช่วยลดความหน่วง

จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือตัว Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด

3.2.5 การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA เช่น ระหว่าง CLBs หรือระหว่าง CLBs กับ IOBs ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอร์ฟแวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมดหรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับโดยสามารถทำขั้นตอนนี้ได้โดยใช้ซอร์ฟแวร์เช่นกันหรือทำการเชื่อมต่อสัญญาณด้วยตัวเอง (Manual Layout) ก็ได้แต่ทางที่ดีควรใช้ซอร์ฟแวร์ทำดีกว่าโดยให้ทำการค้นหาเส้นทางหลายๆครั้งเพื่อหาครั้งที่ดีที่สุด นอกจากนั้นการกำหนดข้อบังคับทางเวลา (Time Constraints) จะช่วยให้ผลที่ได้จากการทำการเชื่อมต่อสัญญาณดีขึ้นได้

3.2.6 การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่วงจรผ่านขั้นตอนต่างๆจนกระทั่งผ่านการทำ PPR (Partitioning, Placement and Routing) แล้วนั้น ถึงตอนนี้ก็สามารถที่จะดาวน์โหลดลงในอุปกรณ์ FPGA ได้แล้วในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้ให้เป็นข้อมูลวงจร (Configuration data) ซึ่งอยู่ในรูปของบิทสตรีม (Bit-Stream) ก่อนแล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชั่นการทำงานตามวงจรที่ออกแบบไว้

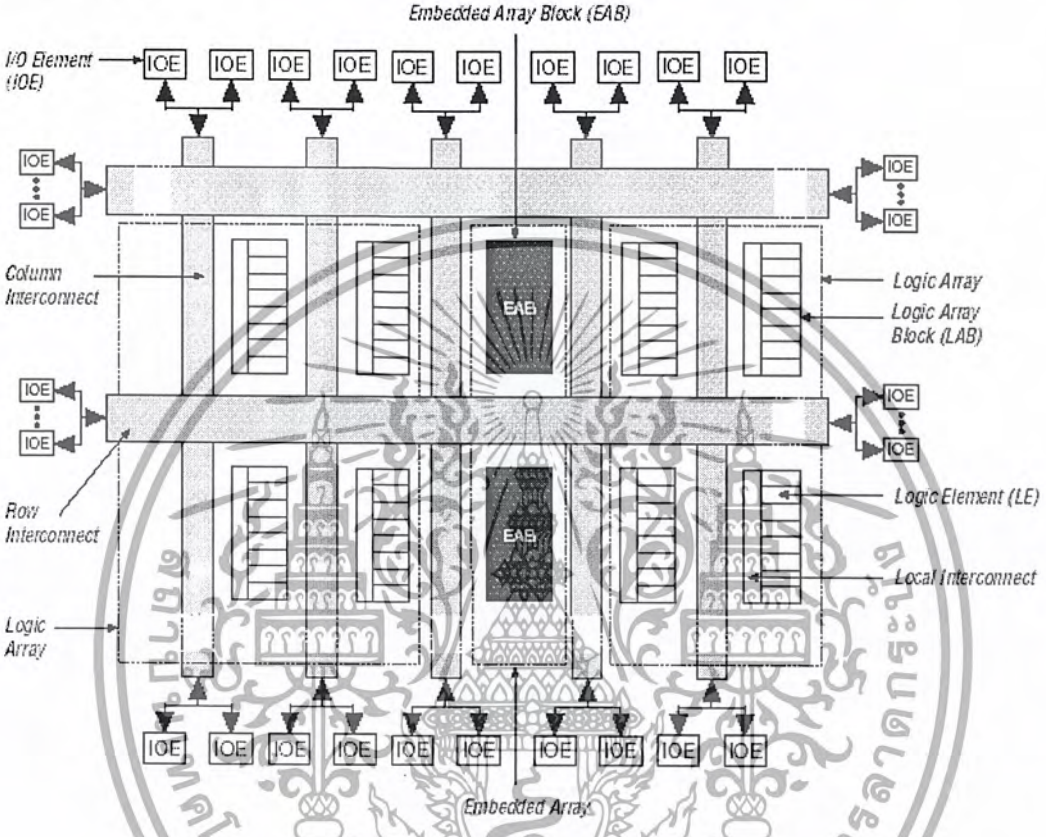
จากที่อธิบายมาทั้งหมดจะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้น ทำได้สะดวกกว่าการทำ ASIC มากเพราะใช้เวลาน้อยกว่ามาก ส่วนสำคัญที่ใช้ในการทำ FPGA คือซอร์ฟแวร์ที่ใช้ตั้งแต่การเขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอร์ฟแวร์ที่ใช้ต้องเป็นซอร์ฟแวร์ที่ใช้ทำงานต่อเนื่องกัน

3.3 โครงสร้างของ FPGA ตระกูล FLEX 10 K

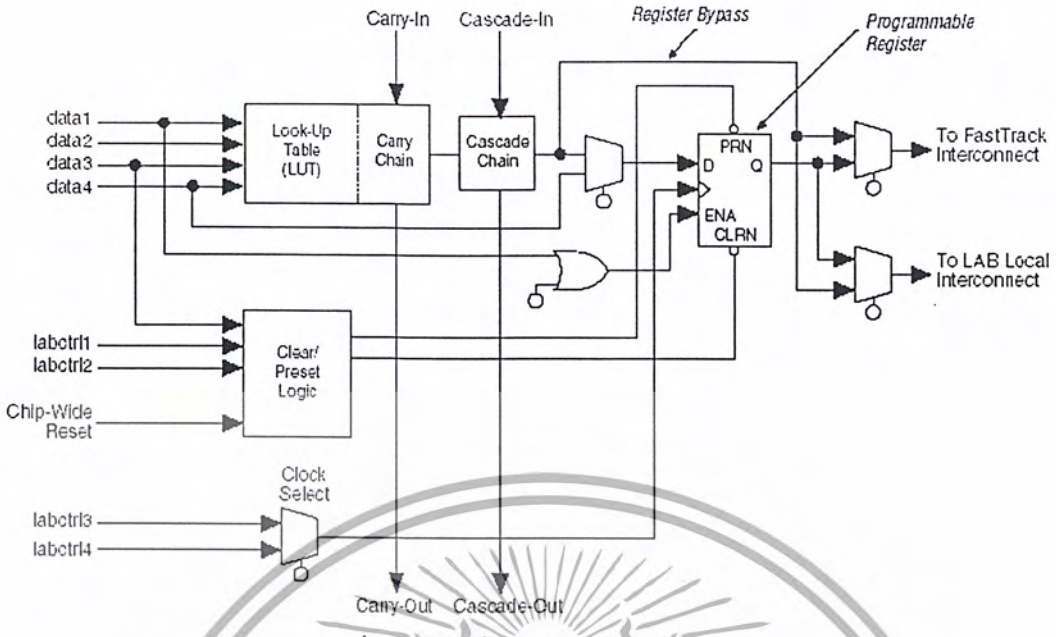
FPGA ของบริษัท Altera ตระกูล FLEX 10 K เป็นอุปกรณ์ที่มีความหนาแน่นเกทประมาณตั้งแต่ 10,000 – 250,000 เกท โดยการจัดโครงสร้าง (Configuration) จะใช้วิธีโหลดโครงสร้างเข้าไปใน SRAM ภายใน ซึ่งหมายความว่าถ้าไม่ได้มีการจ่ายไฟเลี้ยงให้ โครงสร้างที่จัดเอาไว้ก็จะหายไป FPGA ประเภทนี้จะสามารถโปรแกรมซ้ำได้ไม่จำกัดจำนวนครั้ง และการทำงานตามลอจิกฟังก์ชั่นจะใช้วิธีการเปิดตารางดู (Look Up table : LUT) โดยโครงสร้างของ FPGA ตระกูล FLEX 10 K แสดงดังรูปที่ 3.5 โดยในโครงสร้างของ FPGA ตระกูล FLEX 10 K สามารถที่จะแบ่งเป็นส่วนต่างๆ ได้ดังนี้

Logic Element (LE)

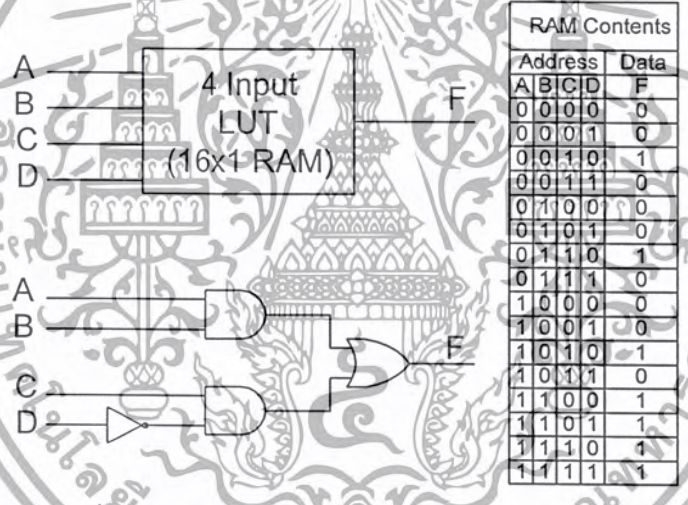
ใน รูปที่ 3.3 แสดงโครงสร้างภายในของ LE โดยการกระทำทางบูลีนของลอจิกเกทจะสร้างด้วยวิธีการ LUT โดย LUT คือ 1x16 SRAM ซึ่ง LUT เพียงตัวเดียวสามารถนำมาทำโครงข่ายของลอจิกเกทที่มี 4 อินพุท และ 1 เอาท์พุท โดยโครงข่ายของลอจิกเกทจะถูกแปลงไปเป็นตารางค่าความจริง (Truth Table) ดังแสดงในรูปที่ 3.4



รูปที่ 3.2 แสดงโครงสร้างของ FPGA ตระกูล FLEX 10K

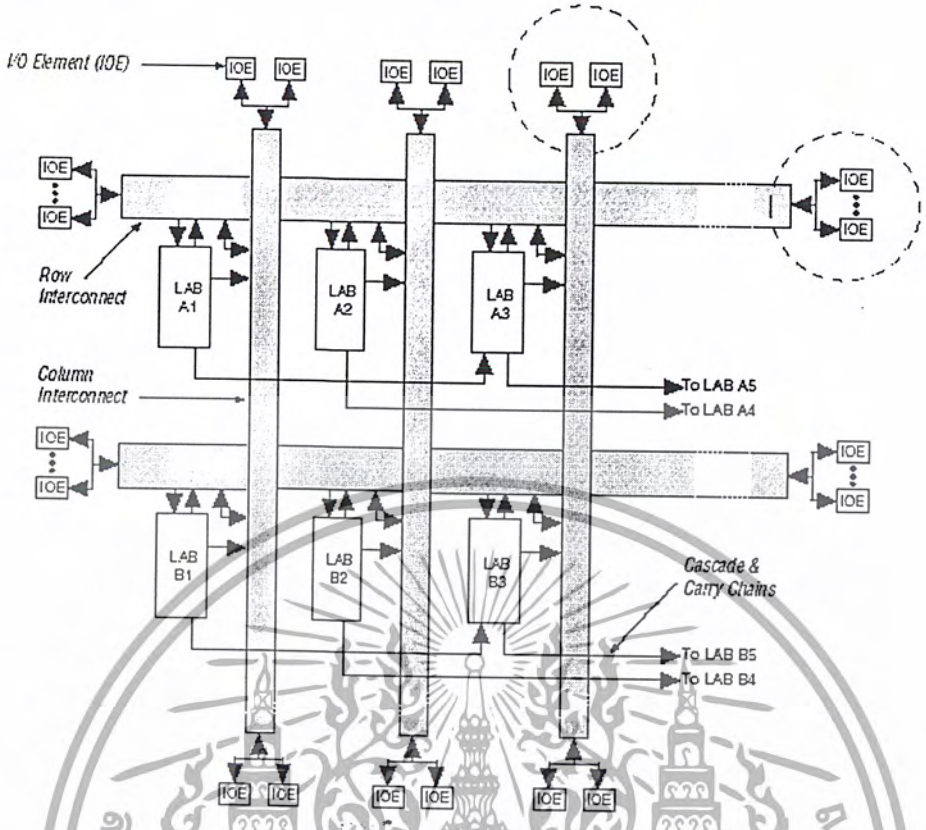


รูปที่ 3.3 แสดงโครงสร้างภายในของ LE



รูปที่ 3.4 แสดงการใช้งาน LUT เป็นโครงข่ายของลอจิก

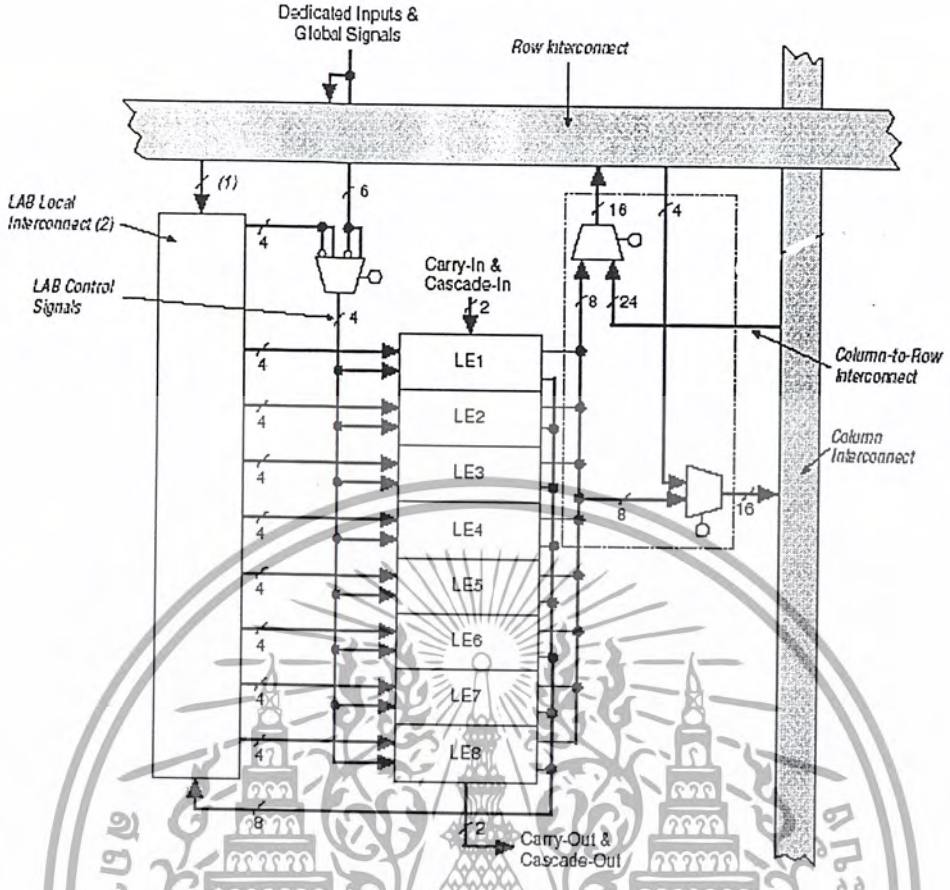
ถ้าโครงข่ายของลอจิกเกทมีความซับซ้อนขึ้นจะต้องใช้ LUT ของแต่ละ LE เป็นจำนวนหลายตัว โดยเอาท์พุทของ LUT จะส่งต่อไปยังฟลิปฟลอปและต่อไปยังโครงข่ายการเชื่อมต่อ (Interconnection Network) ดังแสดงใน รูปที่ 3.5



รูปที่ 3.5 แสดงโครงข่ายของการเชื่อมต่อ

Logic Array Block (LAB)

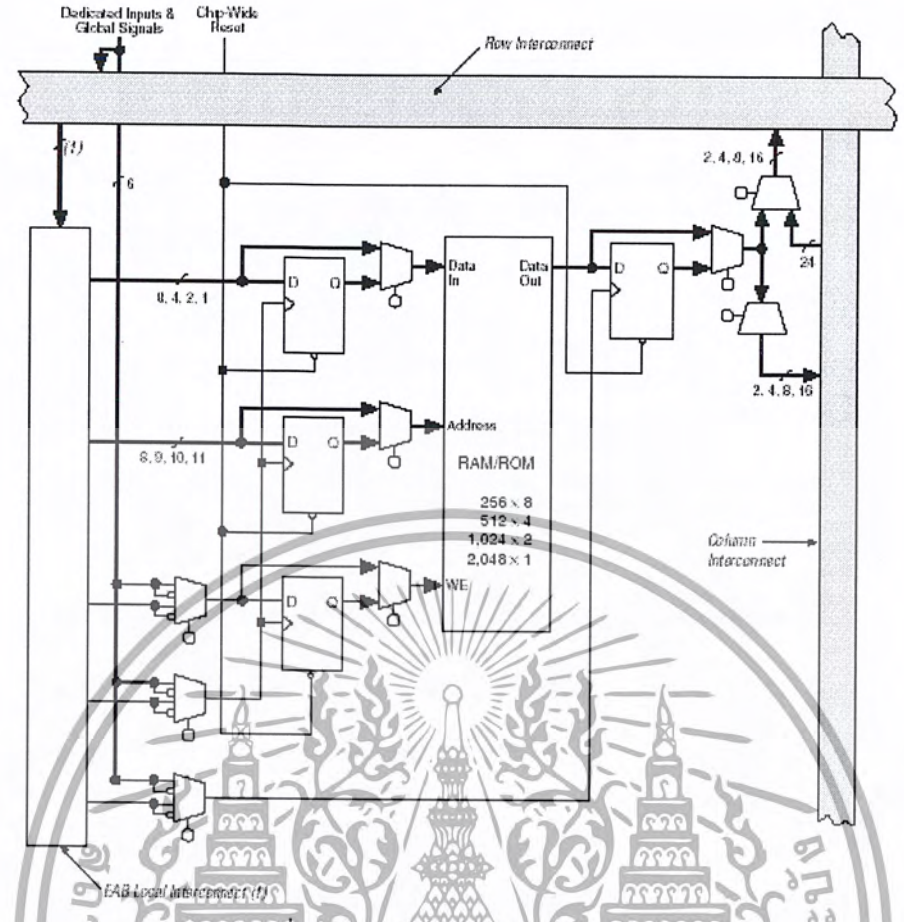
LAB 1 ตัว จะประกอบไปด้วย 8 LE ดังแสดงใน รูปที่ 3.6



รูปที่ 3.6 แสดงโครงสร้างภายในของ LAB

Embedded Array Block (EAB)

สถาปัตยกรรมโดยทั่วไปของ FLEX 10 K จะมีลักษณะของ LAB ที่มีการจัดเรียงแบบเมตริกซ์ และ EAB ซึ่งมีการเชื่อมต่อผ่านทางแถวและคอลัมน์ โดยในแต่ละแถวจะมี 1 EAB ซึ่ง 1 EAB จะมีขนาด 2048 บิท และสามารถกำหนดความกว้าง (Width) ความลึก (Depth) ของ EAB ได้โดยไม่ส่งผลต่อความเร็ว



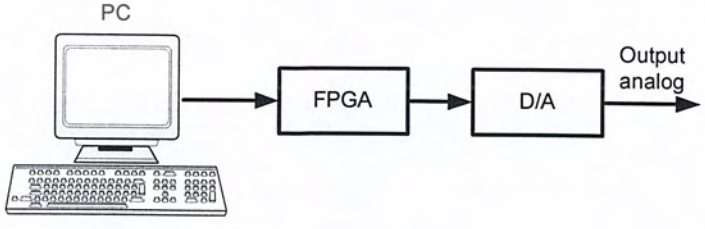
รูปที่ 3.7 แสดงโครงสร้างภายใน EAB

Input Out Element (IOE)

IOE จะถูกต่ออยู่กับขา I/O โดยจะประกอบด้วยส่วนของวงจรที่เป็น Tri State และส่วนที่เป็นฟลิปฟลอป ซึ่งเป็น option

3.4 การออกแบบ

การออกแบบตัวกำเนิดสัญญาณไซน์แบบดิจิตอลด้วย FPGA เป็นการออกแบบโดยการนำเอา Personal Computer (PC) กับ FPGA ต่อทำงานร่วมกัน ดัง รูปที่ 3.8



รูปที่ 3.8 ขั้นตอนการทำงานของ ตัวกำเนิดสัญญาไซน์

3.5 การสร้างสัญญาณรูปคลื่นไซน์

ตารางที่ 3.1 การเก็บค่าขนาด (Amplitude) ในหน่วยความจำ

Phase	Amplitude
0	10000001
1	10000100
2	10000111
3	10001010
4	10001110
5	10010001
6	10010100
7	10010111
8	10011010
9	10011101
10	10100000
11	10100011
12	10100110
13	10101001
14	10101100
15	10101111
16	10110010
17	10110101
18	10110111
19	10111010
20	10111101
21	11000000
22	11000010
23	11000101
24	11001000
25	11001010
26	11001111
27	11010010
28	11001101
29	11010100
30	11010110

Phase	Amplitude
31	11011001
32	11011011
33	11011101
34	11011111
35	11100001
36	11100011
37	11100101
38	11100111
39	11101001
40	11101010
41	11101100
42	11101110
43	11101111
44	11110001
45	11110010
46	11110011
47	11110101
48	11110110
49	11110111
50	11111000
51	11111001
52	11111010
53	11111011
54	11111100
55	11111101
56	11111101
57	11111110
58	11111110
59	11111100
60	11111111
61	11111111
62	11111111
63	11111111

Phase	Amplitude
64	11111111
65	11111111
66	11111111
67	11111111
68	11111110
69	11111110
70	11111101
71	11111101
72	11111100
73	11111100
74	11111011
75	11111010
76	11111001
77	11111000
78	11110111
79	11110110
80	11110101
81	11110011
82	11110010
83	11110001
84	11101111
85	11101110
86	11101100
87	11101010
88	11101001
89	11100111
90	11100101
91	11100011
92	11100001
93	11011111
94	11011101
95	11011011
96	11011001

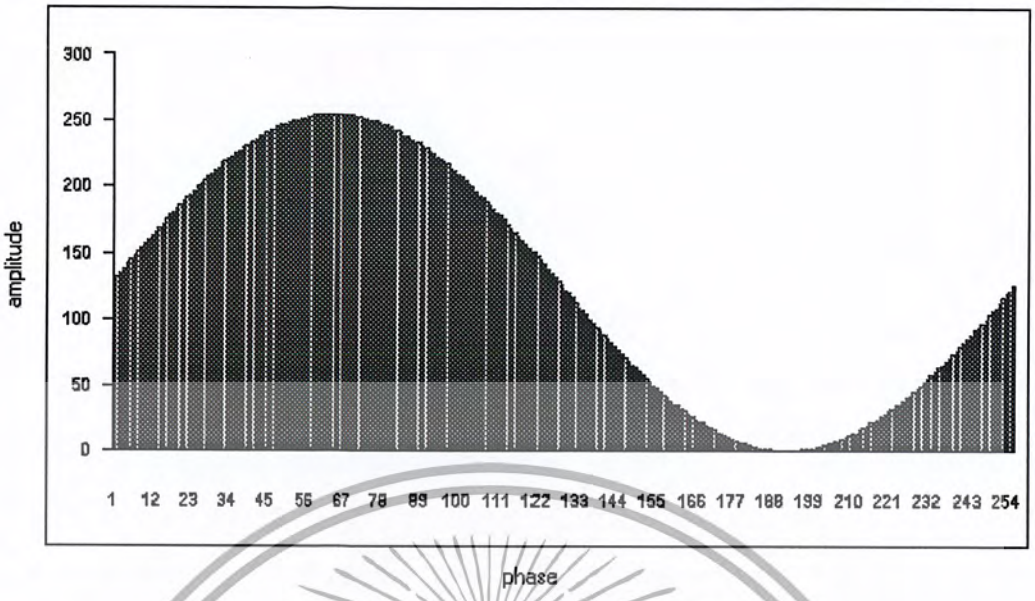
Phase	Amplitude
97	11010110
98	11010100
99	11010010
100	11001111
101	11001101
102	11001010
103	11001000
104	11000101
105	11000010
106	11000000
107	10111101
108	10111010
109	10110111
110	10110101
111	10110010
112	10101111
113	10101100
114	10101001
115	10100110
116	10100011
117	10100000
118	10011101
119	10011010
120	10010111
121	10010100
122	10010001
123	10001110
124	10001010
125	10000111
126	10000100
127	10000001
128	01111110

Phase	Amplitude
129	01111011
130	01111000
131	01110101
132	01110001
133	01101110
134	01101011
135	01101000
136	01100101
137	01100010
138	01011111
139	01011100
140	01011001
141	01010110
142	01010011
143	01010000
144	01001101
145	01001010
146	01001000
147	01000101
148	01000010
149	00111111
150	00111101
151	00111010
152	00110111
153	00110101
154	00110010
155	00110000
156	00101101
157	00101011
158	00101001
159	00100110
160	00100100

Phase	Amplitude
161	00100010
162	00100000
163	00011110
164	00011100
165	00011010
166	00011000
167	00010110
168	00010101
169	00010011
170	00010001
171	00010000
172	00001110
173	00001101
174	00001100
175	00001010
176	00001001
177	00001000
178	00000111
179	00000110
180	00000101
181	00000100
182	00000011
183	00000011
184	00000010
185	00000010
186	00000001
187	00000001
188	00000000
189	00000000
190	00000000
191	00000000
192	00000000

Phase	Amplitude
193	00000000
194	00000000
195	00000000
196	00000001
197	00000001
198	00000010
199	00000010
200	00000011
201	00000011
202	00000100
203	00000101
204	00000110
205	00000111
206	00001000
207	00001001
208	00001010
209	00001100
210	00001101
211	00001110
212	00010000
213	00010001
214	00010011
215	00010101
216	00010110
217	00011000
218	00011010
219	00011100
220	00011110
221	00100000
222	00100010
223	00100100
224	00100110

Phase	Amplitude
225	00101001
226	00101011
227	00101101
228	00110000
229	00110010
230	00110101
231	00110111
232	00111010
233	00111101
234	00111111
235	01000010
236	01000101
237	01001000
238	01001010
239	01001101
240	01010000
241	01010011
242	01010110
243	01011001
244	01011100
245	01011111
246	01100010
247	01100101
248	01101000
249	01101011
250	01101110
251	01110001
252	01110101
253	01111000
254	01111011
255	01111110



รูปที่ 3.9 ผลจากนำค่าในตารางมาวาดกราฟ

การคำนวณหาความถี่ขาออกของวงจรผลิตสัญญาณรูปคลื่นไซน์ สามารถคำนวณได้จากสูตรคำนวณดังนี้

$$f_{out} = \frac{(\Delta phase \times CLKIN)}{2^{32}}$$

เมื่อ $\Delta phase$ = ค่าของ tuning word ซึ่งมี 28 บิท

$CLKIN$ = ความถี่ของสัญญาณนาฬิกา(CLOCK)

f_{out} = ความถี่ขาออกของวงจร

ในการสร้างสัญญาณรูปคลื่นไซน์และกวาดความถี่ตามที่ต้องการเราจะต้องกำหนดความถี่เริ่มต้นและความถี่สิ้นสุด เพราะฉะนั้นจะต้องคำนวณหาค่าจำนวนค่าของ Tuning word ($\Delta phase$) และจากนั้นทำการแบ่งค่าความถี่เป็น 2 ถึง 512 ช่วง ระยะห่างของความถี่เท่าๆกันเพื่อให้โปรแกรมสั่งงานให้วงจรสร้างสัญญาณรูปไซน์กวาดความถี่ไปตามต้องการ

ตัวอย่าง ต้องการหาผลตอบสนองความถี่ในช่วง 1kHz ถึง 2 kHz หาช่วงของความถี่ f_{out} เพื่อคำนวณหา ค่าของ Tuning word ซึ่งมี 28 บิท ($\Delta phase$) และใช้ $\Delta phase$ ในการควบคุมความถี่ที่ผลิตจากวงจรสร้างสัญญาณรูปคลื่นไซน์ใช้ความถี่ของสัญญาณนาฬิกา 12 MHz

ตารางที่ 3.2 ตารางความถี่และค่าตัวเลขควบคุมความถี่

ความถี่ Fout (Hz)	Tuning word (Decmal)
1000	22369
1010	22592
1020	22816
1030	23040
1040	23263
1050	23487
1060	23711
1070	23934
1080	24158
1090	24382
1100	24606
1110	24829
1120	25053
1130	25277
1140	25500
1150	25724
1160	25948
1170	26171
1180	26395
1190	26619
1200	26843
1210	27066
1220	27290
1230	27514
1240	27737
1250	27961
1260	28185
1270	28408
1280	28632
1290	28856
1300	29080

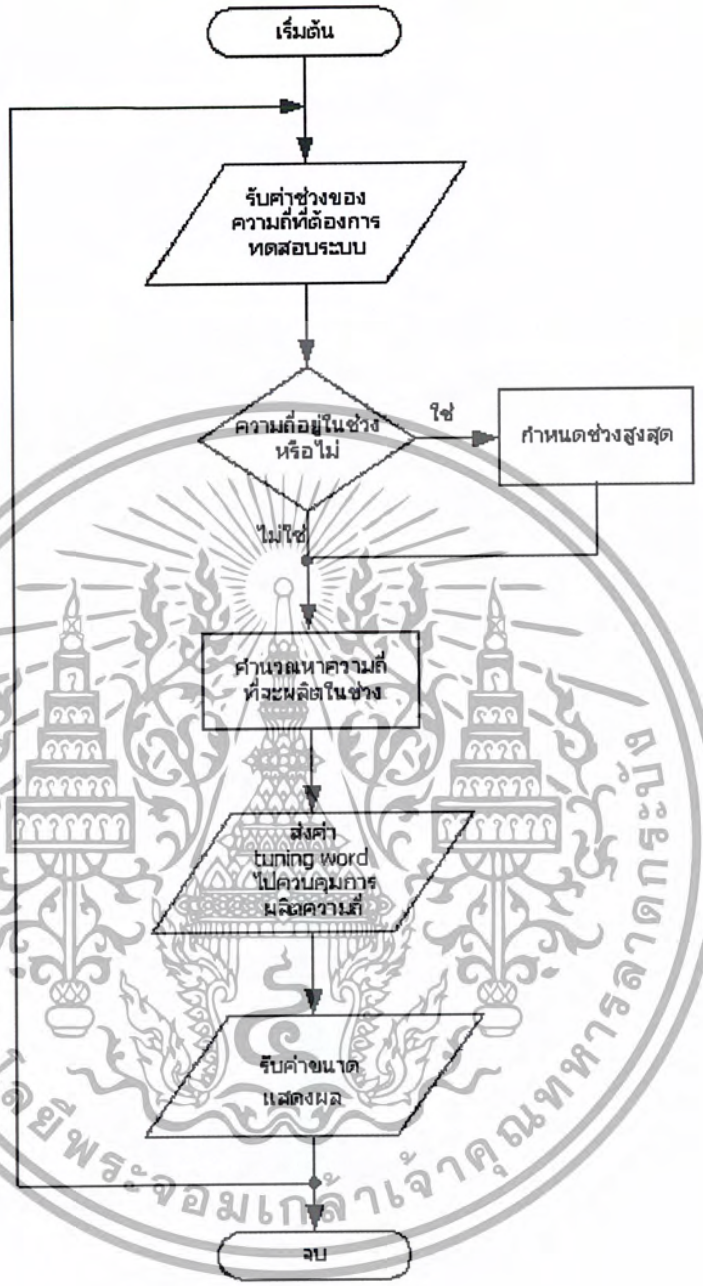
ความถี่ Fout (Hz)	Tuning word (Decmal)
1310	29303
1320	29527
1330	29751
1340	29974
1350	30198
1360	30422
1370	30645
1380	30869
1390	31093
1400	31316
1410	31540
1420	31764
1430	31988
1440	32211
1450	32435
1460	32659
1470	32882
1480	33106
1490	33330
1500	33553
1510	33777
1520	34001
1530	34225
1540	34448
1550	34672
1560	34896
1570	35119
1580	35343
1590	35567
1600	35790
1610	36014
1620	36238
1630	36461

ความถี่ Fout (Hz) ความถี่ Fout (Hz)	Tuning word (Decmal)
1640	36685
1650	36909
1660	37133
1670	37356
1680	37580
1690	37804
1700	38027
1710	38251
1720	38475
1730	38698
1740	38922
1750	39146
1760	39370
1770	39593
1780	39817
1790	40041
1800	40264
1810	40488
1820	40712
1830	40935
1840	41159
1850	41383
1860	41606
1870	41830
1880	42054
1890	42278
1900	42501
1910	42725
1920	42949
1930	43172
1940	43396
1950	43620
1960	43843

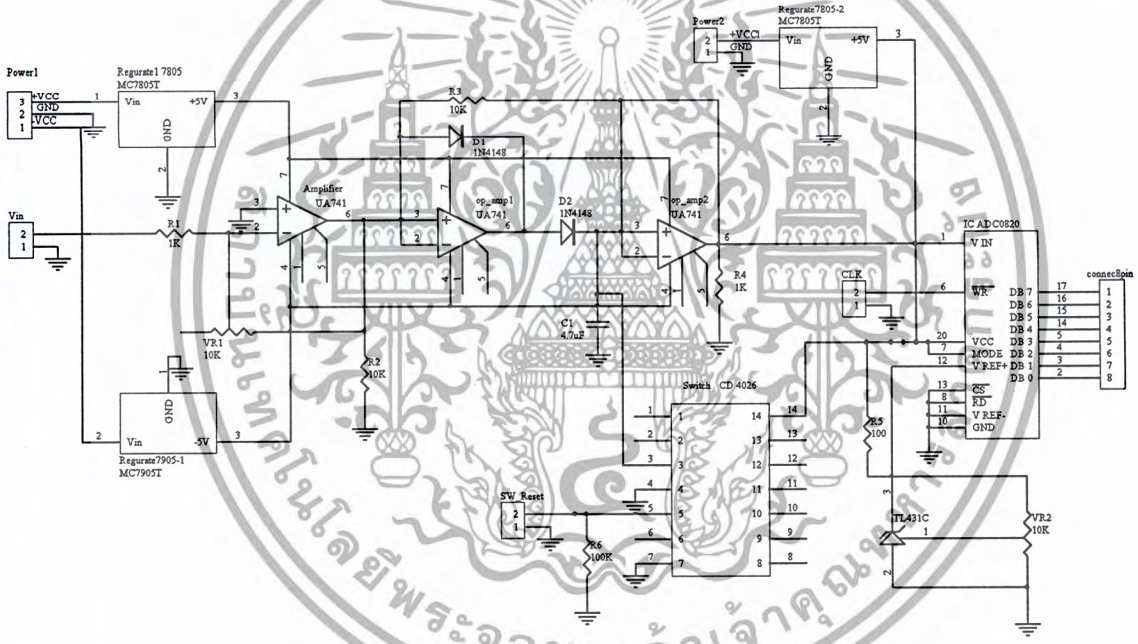
ความถี่ Fout (Hz)	Tuning word (Decmal)
1970	44067
1980	44291
1990	44515
2000	44738

ค่าที่ได้ตามตารางข้างต้นจะได้จากการคำนวณจากคอมพิวเตอร์และนำค่าของ tuning word ($\Delta phase$) นี้ไปแปลงเป็นเลขฐานสองเพื่อนำค่าที่ได้ควบคุมการสร้างความถี่ เช่น ต้องการความถี่ขาออกเท่ากับ 1000Hz จะได้tuning word = 22369 จากนั้นแปลงเป็นเลขฐานสองจะได้เท่ากับ " 0000 0000 0000 0101 0111 0110 0001 "

3.6 ขั้นตอนการทำงานของโปรแกรมควบคุมและแสดงผล



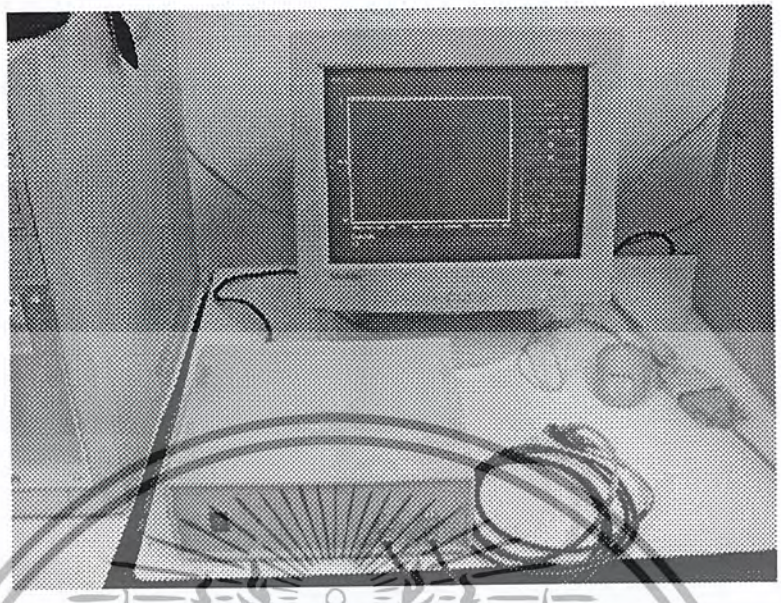
รูปที่ 3.10 แสดงขั้นตอนการทำงานถึงการสร้างสัญญาณและการรับสัญญาณกลับมาเพื่อแสดงผล



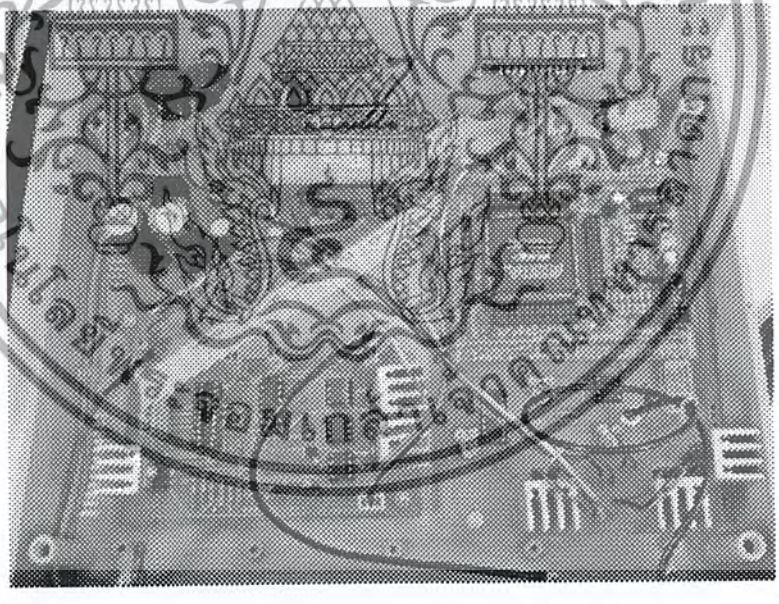
รูปที่ 3.11 วงจรจับยอดแรงดันและวงจรแปลงสัญญาณอนาล็อกเป็นดิจิตอล

Title		
Size B	Number	Revision
Date: 5-Apr-2004		Sheet of
File: C:\ADVPCB\adc.ddb		Drawn By:

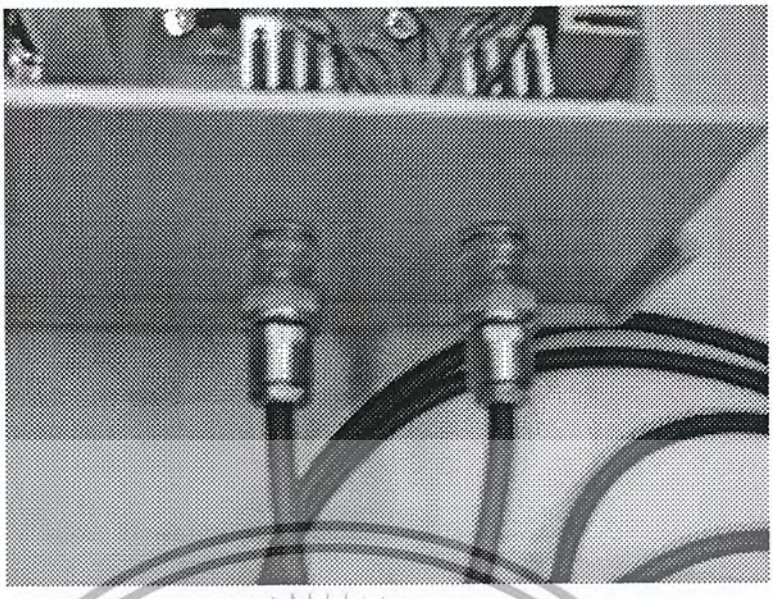
3.7 แสดงชิ้นงานที่สร้างขึ้น



รูปที่ 3.12 รูปการแสดงผลและการต่อสายวัดสัญญาณ



รูปที่ 3.13 การนำอุปกรณ์ลงกล่อง



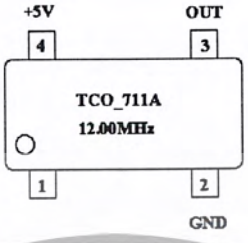
รูปที่ 3.14 การต่อสาย BNC ใช้เพื่อต่อเข้ากับระบบที่ทดสอบ

รูปที่ 3.15 การต่อสาย DB-25 เข้ากับ Computer

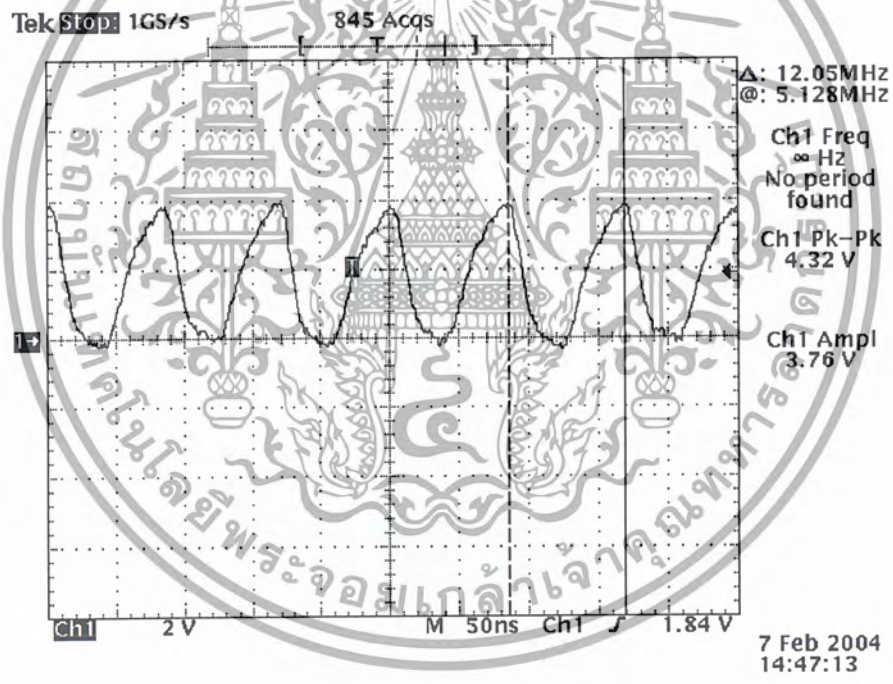
บทที่ 4

การทดลองและผลการทดลอง

4.1 สัญญาณนาฬิกา (CLK) ที่ป้อนให้กับ FPGA

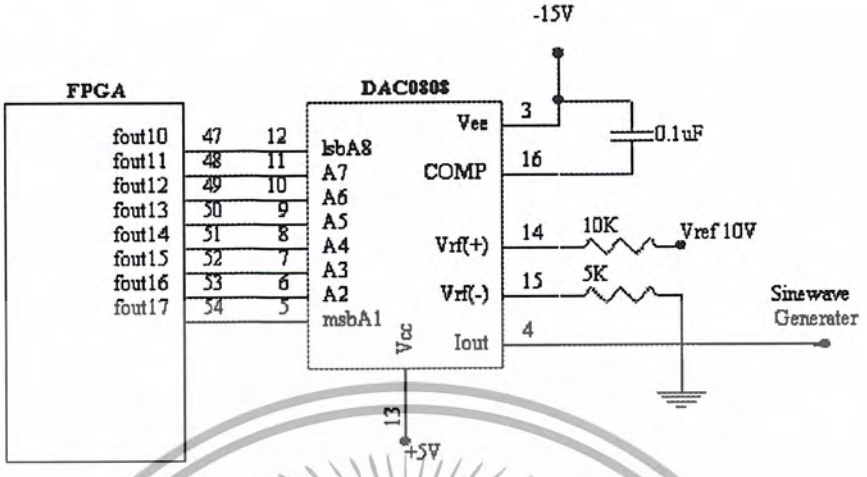


รูปที่ 4.1 ตัวป้อนสัญญาณนาฬิกาให้กับ FPGA



รูปที่ 4.2 รูปสัญญาณนาฬิกาความถี่ 12 MHz แอมปลิจูด 3.76 V

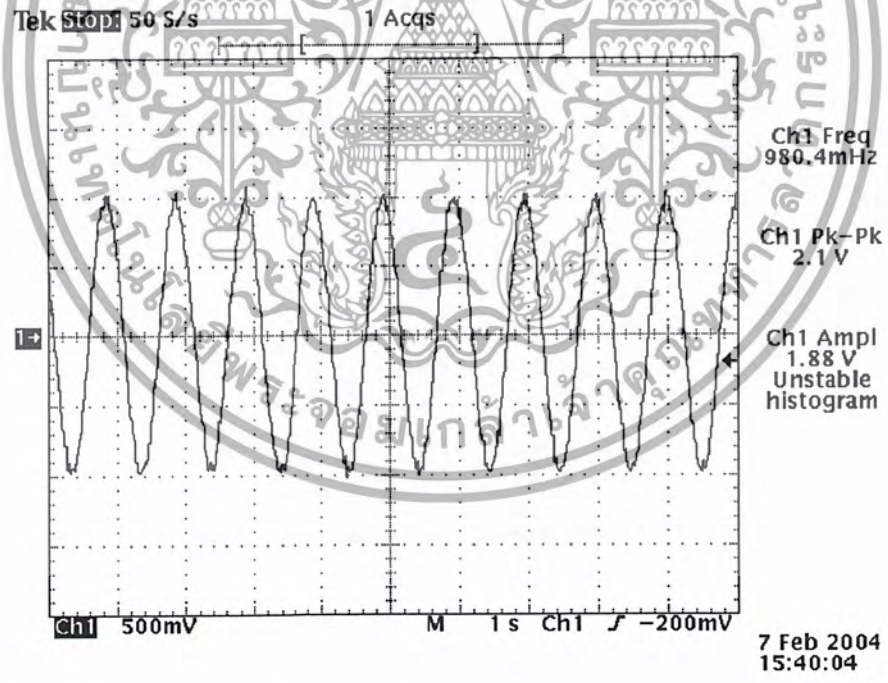
4.2 ส่วนสร้างสัญญาณไซน์เวฟความถี่ 1 ถึง 20kHz



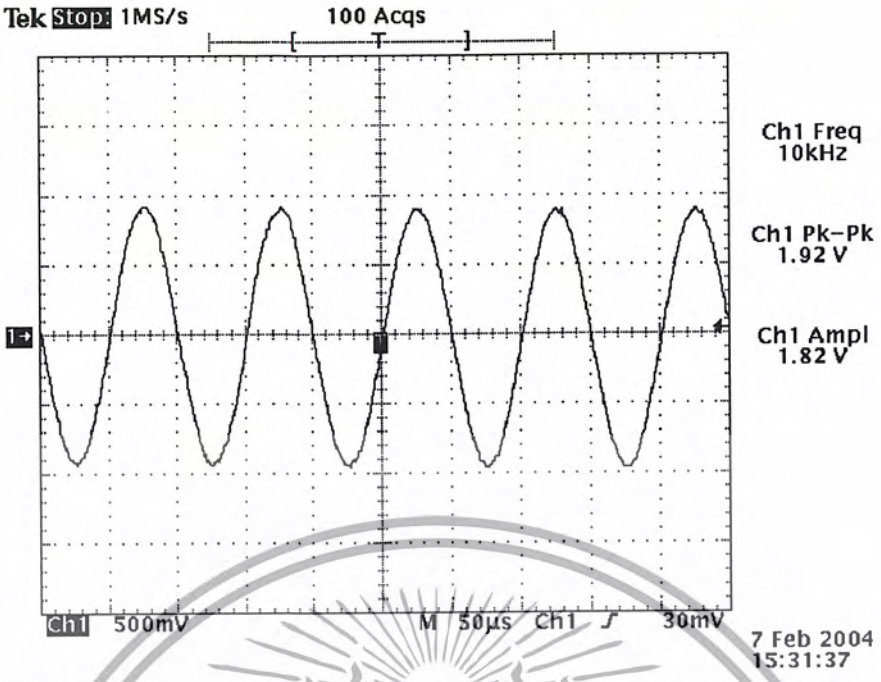
รูปที่ 4.3 วงจรสร้างสัญญาณไซน์เวฟความถี่ 1Hz – 20kHz

4.3 สัญญาณที่ใช้ในการทดสอบระบบ

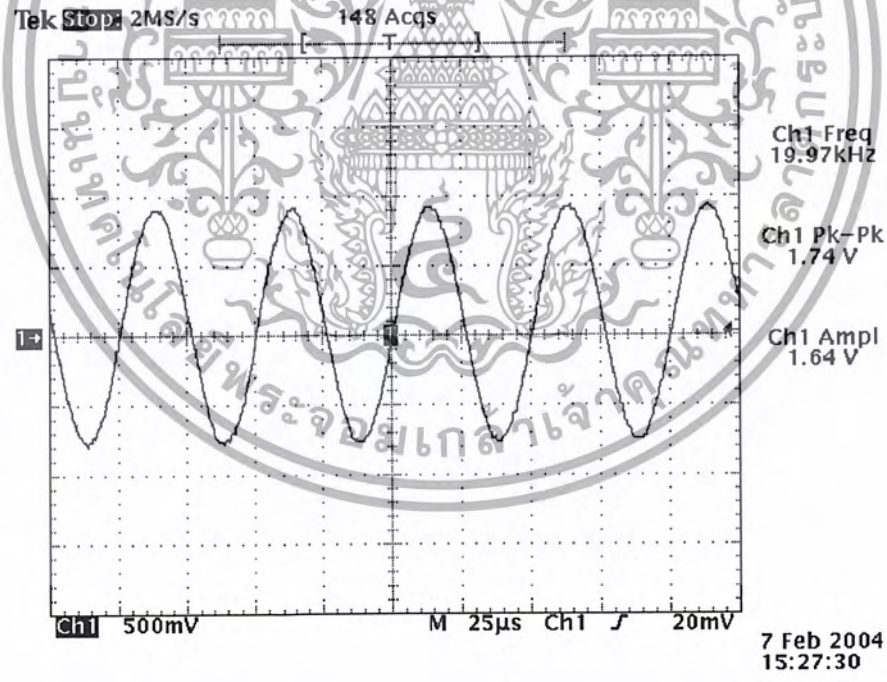
สัญญาณที่ได้จากส่วนสร้างสัญญาณไซน์เวฟ ซึ่งสามารถกำหนดความถี่ได้โดยสั่งทางแป้นพิมพ์



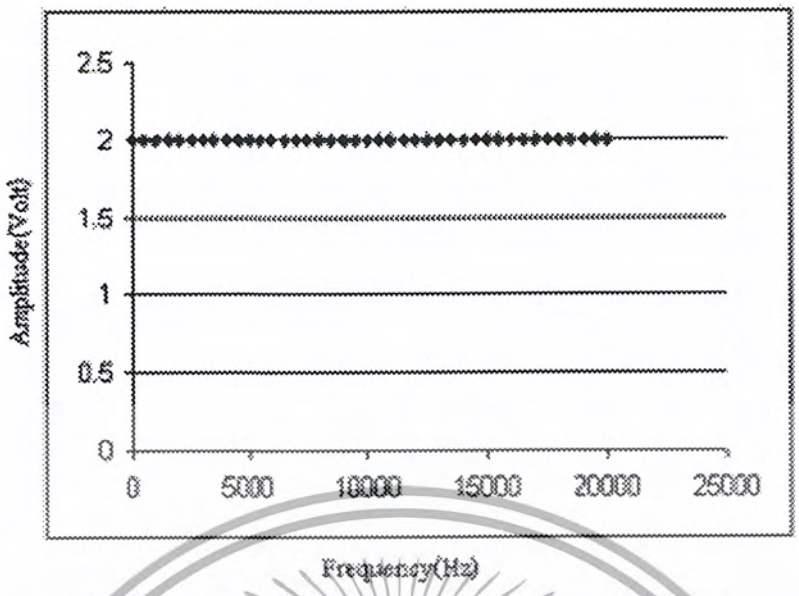
รูปที่ 4.4 รูปสัญญาณไซน์เวฟความถี่ 1 Hz แอมปลิจูด 2 Vp_p



รูปที่ 4.5 รูปสัญญาณไซน์เวฟ 2 kHz แอมปลิจูด 2 Vp_p

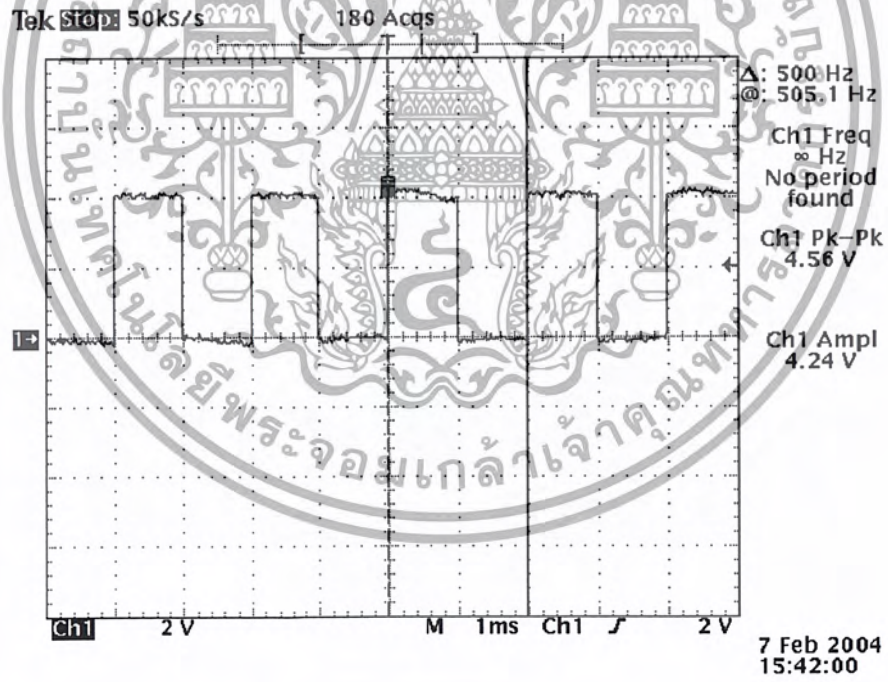


รูปที่ 4.6 รูปสัญญาณ Sine wave ความถี่ 20 kHz แอมปลิจูด 2 Vp_p



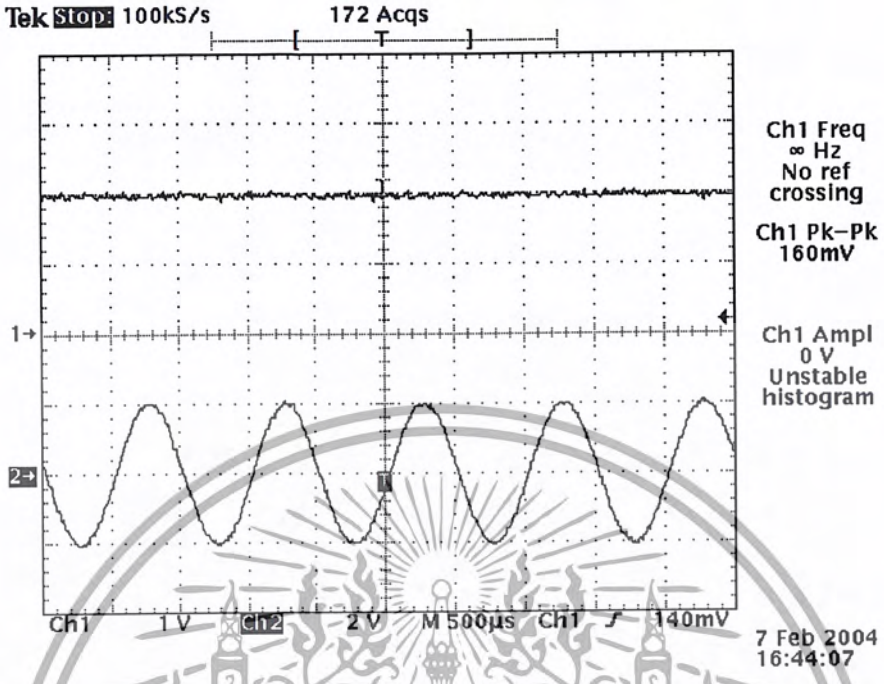
รูปที่ 4.7 แสดงความสัมพันธ์ระหว่างขนาดและ ความถี่ของชุดสร้างความถี่

4.4 สัญญาณอินพุทที่เข้า ADC0808

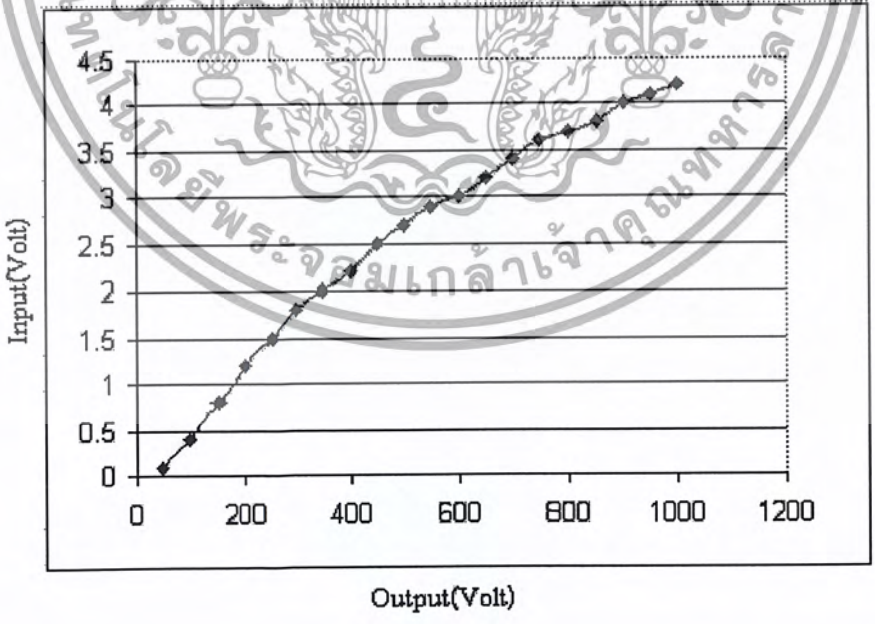


รูปที่ 4.8 รูปสัญญาณที่ขา 5 ของ ADC 0808 ความถี่ 500Hz

4.5 สัญญาณจากวงจรตรวจจับแรงดันยอด



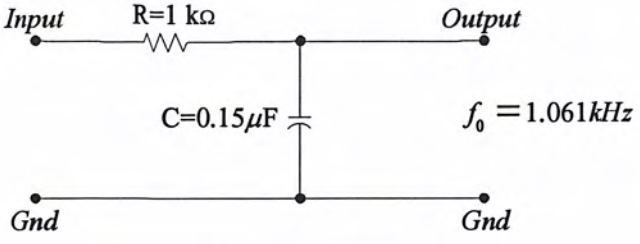
รูปที่ 4.9 สัญญาณการตรวจจับแรงดันยอดที่ความถี่ 500Hz แอมปลิจูด 2Vp-p



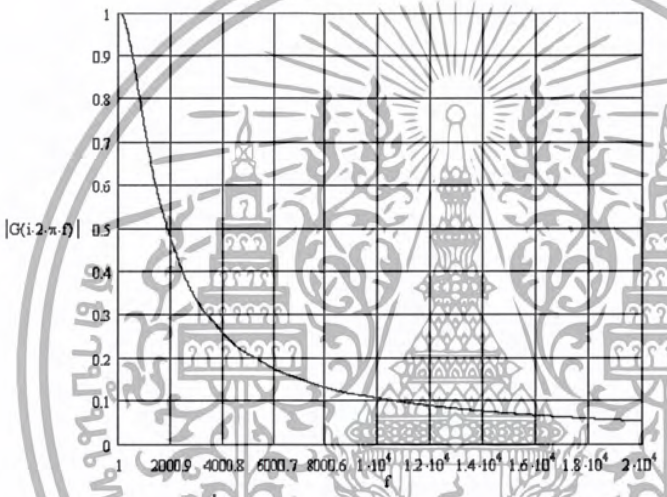
รูปที่ 4.10 แสดงความสัมพันธ์ระหว่างอินพุทและเอาท์พุทของชุดวงจรจับยอดแรงดัน

4.6 ผลการเปรียบเทียบระหว่างการคำนวนและจากการใช้เครื่องวิเคราะห์ทดสอบวงจรกรองความถี่แบบต่างๆ

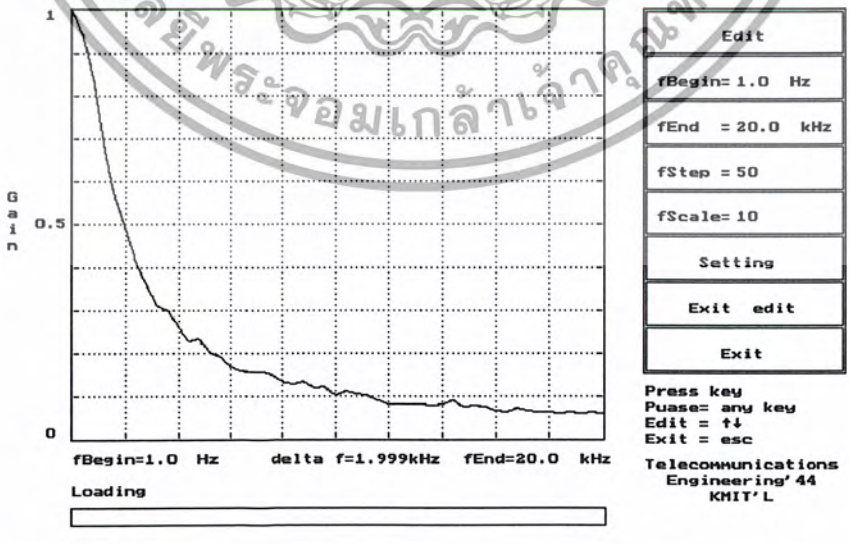
4.6.1 Low Pass Filter



รูปที่ 4.11 วงจร Low Pass Filter

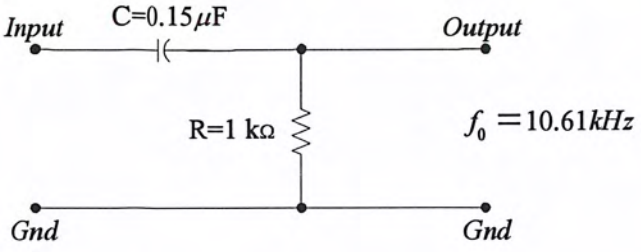


รูปที่ 4.12 ผลการคำนวน Low Pass filter

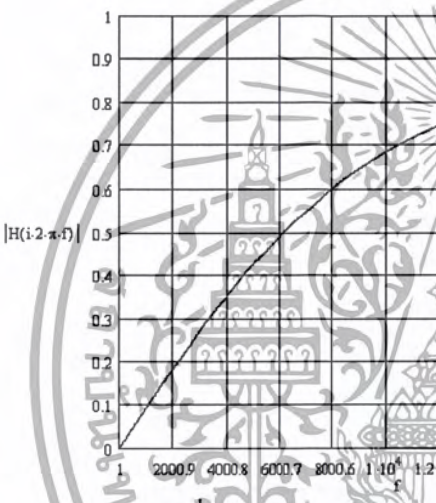


รูปที่ 4.13 ผลการใช้เครื่องวิเคราะห์ทดสอบ Low Pass Filter

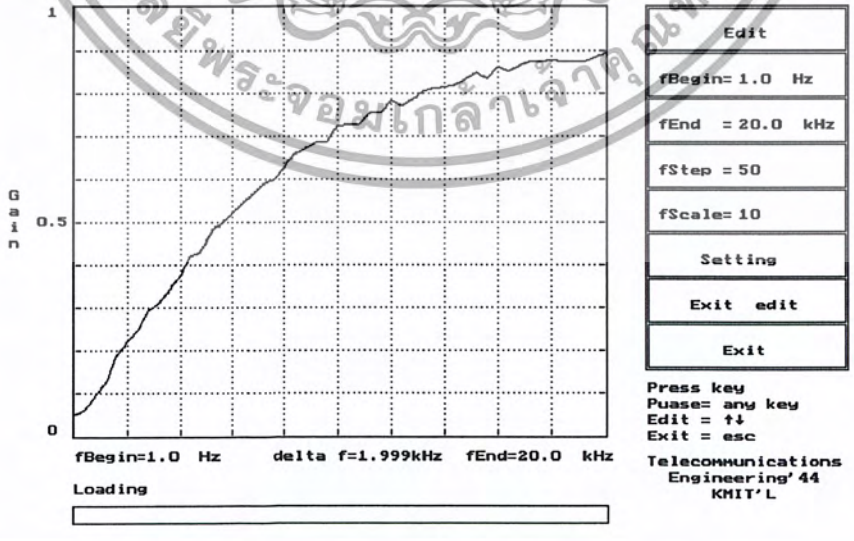
4.6.2 วงจร Hight Pass Filter



รูปที่ 4.14 วงจร High Pass Filter



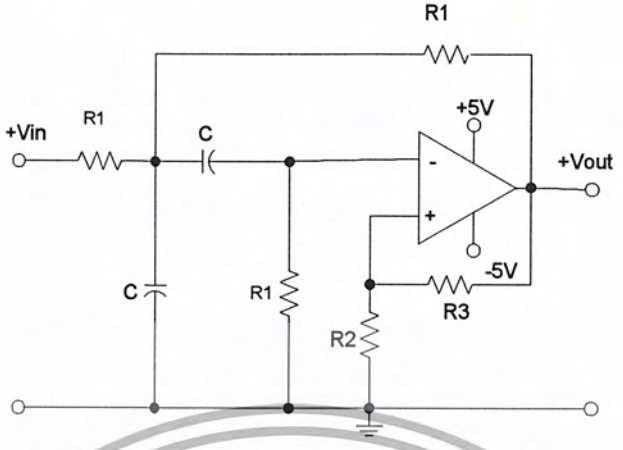
รูปที่ 4.15 ผลการคำนวน High Pass Filter



รูปที่ 4.16 ผลการใช้เครื่องวิเคราะห์ทดสอบ High Pass Filter

4.6.3 วงจร Band Pass Filter

$R_1 = 500\Omega\ , R_2 = 1k\Omega, R_3 = 1k\Omega, C = 0.1\mu F$

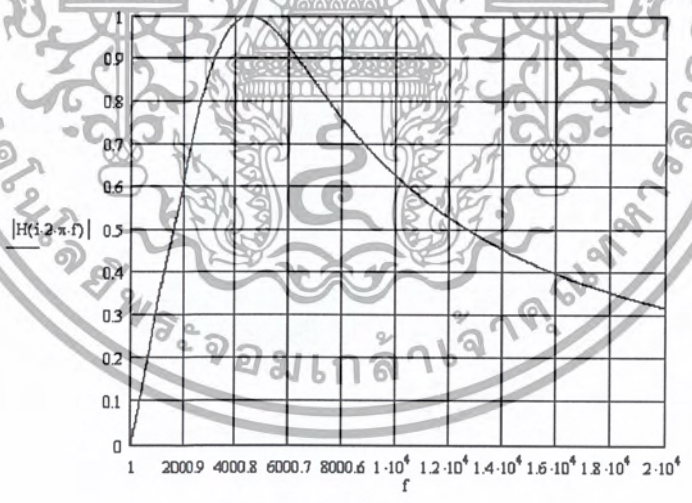


$K = 4\times10^{4}$

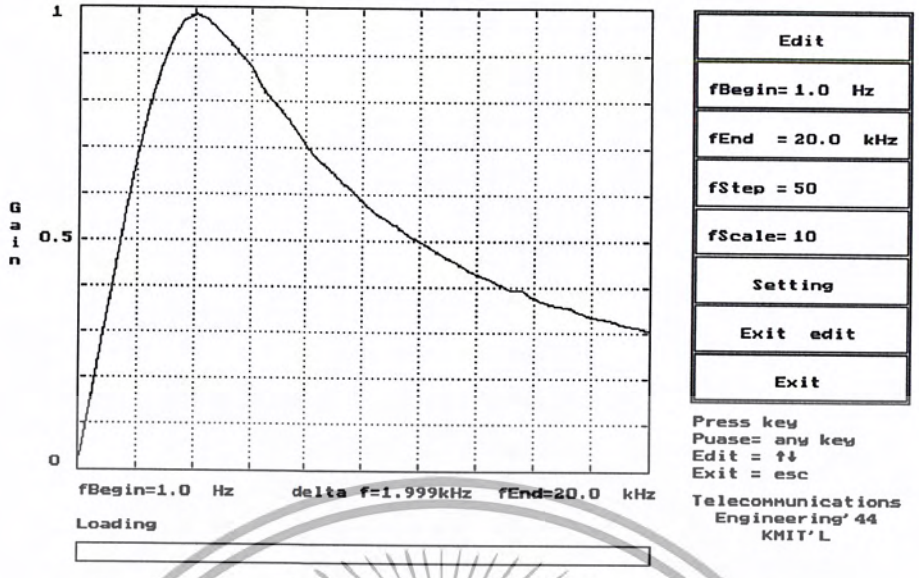
$BW = 6.366\times10^{3}$

$f_0 = 4.502\times10^{3}$

รูปที่ 4.17 วงจร Band Pass Filter



รูปที่ 4.18 ผลการคำนวน Band Pass Filter

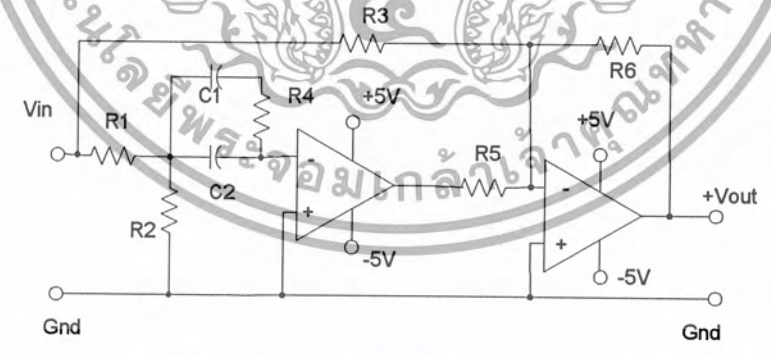


รูปที่ 4.19 ผลการใช้เครื่องวิเคราะห์ทดสอบ Band Pass Filter

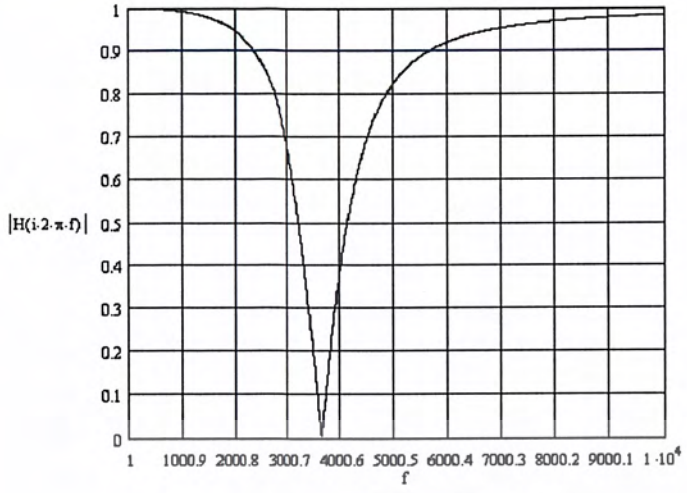
4.6.4 วงจร Band Reject Filter

$R1=20k\Omega$,$R2=1k\Omega$,$R3=R4=100k\Omega$,$R5=50k\Omega$,$R6=100k\Omega$

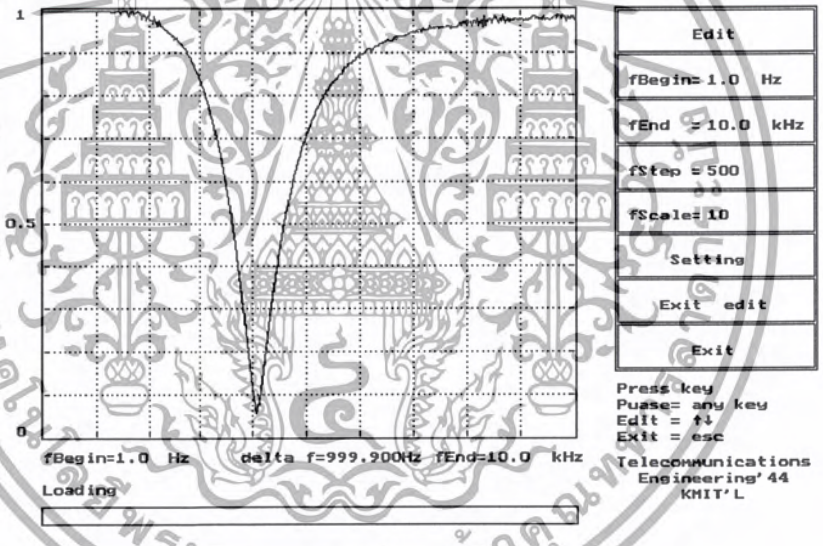
$K=1, f_0=3.647kHz, BW=1.592kHz$



รูปที่ 4.20 วงจร Band Reject Filter



รูปที่ 4.21 ผลการคำนวน Band Reject Filter



รูปที่ 4.22 ผลการใช้เครื่องวิเคราะห์ทดสอบ Band Reject Filter

บทที่5
บทวิจารณ์ และบทสรุป

5.1 สรุปผลการทดลอง

จากการทดลองและสร้างเครื่องวิเคราะห์ผลตอบสนองต่อขนาดที่แปรผันไปตามความถี่ซึ่งทำให้ได้มีความรู้ทฤษฎีการกำเนิดสัญญาณดิจิตอลแบบตรง(Direct Digital Frequency Synthesizer) ฮาร์ดแวร์ ซอฟต์แวร์ และหลักการสร้าง

ซึ่งในส่วนของฮาร์ดแวร์ได้ใช้อุปกรณ์ FPGA ในการสร้างสัญญาณไซน์เวฟเพื่อใช้ในการทดสอบ ใช้อุปกรณ์แปลงสัญญาณ Digital to Analog ใช้แปลงสัญญาณดิจิตอลจาก FPGA เป็นอนาล็อกเพื่อใช้ทดสอบระบบใช้อุปกรณ์แปลงสัญญาณ Analog to Digital แปลงสัญญาณจากระบบทดสอบเป็นดิจิตอลส่งเข้าคอมพิวเตอร์เพื่อทำการคำนวณและแสดงผลใช้วงจรตรวจจับยอดแรงดันที่วัดเข้ามาจากระบบทดสอบเพื่อหาค่าแรงดันสูงสุดเพื่อใช้ในการคำนวณหาค่าอัตราการขยาย

ในส่วนของซอฟต์แวร์ได้อธิบายการทำงานของวงจรด้วยภาษา VHDL โดยการประยุกต์และออกแบบทั้งหมดใช้โปรแกรม Max Plus II ในการสร้างสัญญาณไซน์เวฟออกมาและใช้โปรแกรมภาษาซีในการประมวลผลของสัญญาณที่ทำการทดสอบและแสดงผลสู่จอภาพ

5.2 ปัญหาที่พบ

- แหล่งจ่ายไฟไม่คงที่จะทำให้การทดสอบระบบได้ผลที่ไม่เที่ยงตรง
- ในการผลิตความถี่ต่ำจะใช้เวลามากกว่าความถี่สูง เนื่องจากมีคาบเวลาที่ยาวนานกว่า
- มีสัญญาณรบกวนจากการผลิตสัญญาณรูปคลื่นไซน์
- อุปกรณ์เสียหายบ่อย

5.3 แนวทางแก้ไข

- ใช้อุปกรณ์เรคติไฟเออร์เพื่อทำให้แรงดันที่ป้อนให้ระบบคงที่
- ในการผลิตความถี่ต่ำถ้าต้องการให้ได้สัญญาณที่เร็วขึ้นจะต้องใช้คาบเวลาในการสร้างที่เร็วขึ้น
- กรองความถี่สูงจากแหล่งจ่ายไฟ
- ลดแรงดันลงเพื่อลดกำลังไฟฟ้าในตัวอุปกรณ์

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงลงได้ เนื่องมาจากการแนะนำให้คำปรึกษาและชี้ แนวทางจาก ผศ.เกรียงไกร วงศ์โรจนภรณ์ รศ.ดร.สุวิพล สิทธิชีวภาค ซึ่งเป็นอาจารย์ที่ปรึกษาเอาใจใส่ดูแลเป็นอย่างดี ตลอดจนบุคคลทุกคนที่มีส่วนร่วมในปริญญานิพนธ์ชิ้นนี้ ผู้จัดทำจึงขอขอบพระคุณในความอนุเคราะห์ของทุกท่านไว้ ณ ที่นี้ด้วย

หนังสืออ้างอิง

1. J.L.Hilburn and D.E.Jhonson.,"Manual of Active filter Degn",McGraw-Hill Book Company, New York,1973
2. R.C.Jaeger, "Microelecrtonic Circuit Design",McGraw-Hill Companies,Inc,New York,1997
3. S.Sjonhom and L.Lindh,." VHDL for Designers",Prentice Hall,1997
4. D.L.Perry, "VHDL",New York:McGraw-Hill,1995

ภาคผนวก

โปรแกรมแสดงผล

```
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <string.h>
#include <iostream.h>
#include <dos.h>
#define dl  1000 //delay freq
void window(int,int,int,int,float,float,int,int,int,int,int);
void scalexy(int,int,int,int,int,int,int,int);
float linear(float,float,float,float,float);
void fvalue(int,int,char*,float);
void ivalue(int,int,char*,int);
float edit(int,int,int,int,int,int,float,int);
float inedit(int,int,float);
void crlword(int,double);
int det(void);
int main(void)
      {
       int input=0,y_wide,x_wide;
       int x1,y1,x2,y2;
       int nx=10,ny=10;
       float fbegin=1,fend=20000,scale,fout[1000];
       int inportg=0xf300,inportp=0xf301,outportfq=0x378;
       int Nx=50,scalex=10,n;
       int y_bar=330,yb[30],ye[30],delta,cell=8,bar=2,barsensor=0;
       float delt;
       float editvle[50];
       clrscr();
       int gdriver = DETECT, gmode, errorcode;
       initgraph(&gdriver, &gmode, "");
       errorcode = graphresult();
       if (errorcode != grOk)
            {
            printf("Graphics error: %s\n", grapherrormsg(errorcode));
            printf("Press any key to halt:");
      getch();
      exit(1);
            }
       x1=50;
       y1=60;
       x_wide=400;
       y_wide=320;
       y2=y1+y_wide;
       x2=x1+x_wide;
```

```
        while(1)
        {
        if(kbhit()!=0)
        {
           int a;
           input=getch();
           if(input!=27&&input!=32&&input!=0)input=32;
           if(input==27) return 0;
           if(input==32)barsensor=1;
           }
               delt=(y_bar-y1)/float(cell);
               for(n=0;n<=cell;n++)
               {
               yb[n]=((y1+1)+(n-1)*delt);
               ye[n]=((y1-1)+(n)*delt);
               }
         scale=(fend-fbegin)/float(Nx);
         for(n=0;n<=Nx;n++)
         {
         fout[n]=fbegin+(n*scale);
         }
        if(input!=32)
          {
           setcolor(8);
           outtextxy(1,(y1+y_wide/2)-12*2,"G");
           outtextxy(1,(y1+y_wide/2)-12,"a");
           outtextxy(1,(y1+y_wide/2),"i");
           outtextxy(1,(y1+y_wide/2)+12,"n");
           setcolor(3);
           fvalue(x1,y2+12,"fBegin=",fbegin);
           fvalue(((x1+x2)/2)-6*8,y2+12,"delta f=",
                  (fend-fbegin)/float(scalex));
           fvalue(x2-8*13,y2+12,"fEnd=",fend);
           outtextxy(x1-20,y1,"1");
           outtextxy(x1-28,(y1+y2)/2-4,"0.5");
           outtextxy(x1-20,y2-8,"0");
           edit(x2+30,y1,x2+180,y_bar,cell,bar,0,0);
           setcolor(3);
           fvalue(x2+100,(yb[2]+ye[2])/2,"",fbegin);
           fvalue(x2+100,(yb[3]+ye[3])/2,"",fend);
           ivalue(x2+100,(yb[4]+ye[4])/2,"",Nx);
           ivalue(x2+100,(yb[5]+ye[5])/2,"",scalex);
           scalexy(x1,y1,x_wide,y_wide,fbegin,fend,Nx,scalex);
window(x1,y1,x_wide,y_wide,fbegin,fend,Nx,scalex,inportg,inportp,
         outportfq);
           }
           editvle[2]=fbegin;
           editvle[3]=fend;
           editvle[4]=Nx;
           editvle[5]=scalex;
           editvle[6]=1;
           editvle[7]=1;
           if(barsensor==1)
              {
              int b=1;
```

```
        while(b!=27)
              {
              if(b==72)bar=bar-1;
              if(b==80)bar=bar+1;
              if(bar>cell)bar=2;
              if(bar<2)bar=cell;

editvle[bar]=edit(x2+30,y1,x2+180,y_bar,cell,bar,editvle[bar],b);
              if(bar==cell&&b==13)return 0; //exit
              if(bar!=cell-1)
                    {
                    b=getch();
                    if(bar==6&&b==13)//setting
                    {
                    restorecrtmode();
                    clrscr();
                    printf("\n\n\t\t\tAmplitud Response Analyzer");
                    printf("\n\n\t\t\t\t          By");
                    printf("\n\n\t\t\tMr.Piyapan Congset 44015016");
                    printf("\n\t\t\tMr.Sarawut Buasri  44015031");
                    printf("\n\n\n\t\t\t\t      setting");
                    printf("\n\n\t\t\t    1 ) position graph x");
                    printf("\n\n\t\t\t    2 ) position graph y");
                    char setting=0;;
                    while(setting!=27)
                    {
                    setting=getch();
                    clrscr();
                    printf("\n\n\t\t\tAmplitud Response Analyzer");
                    printf("\n\n\t\t\t\t          By");
                    printf("\n\n\t\t\tMr.Piyapan Congset 44015016");
                    printf("\n\t\t\tMr.Sarawut Buasri  44015031");
                    printf("\n\n\n\t\t\t\t      setting");
                    printf("\n\n\t\t\t    1 ) position graph x");
                    printf("\n\n\t\t\t    2 ) position graph y");
                    switch(setting)
                                {
                                case '1':
                                printf("\n\n\t\tx1 :");
                                scanf("%d",&x1);
                                break;
                                case '2':
                                printf("\n\n\t\ty1 :");
                                scanf("%d",y1);
                                break;
                                }
                    }
                    setgraphmode(getgraphmode());
                    b=27;
    setcolor(8);
    outtextxy(1,(y1+y_wide/2)-12*2,"G");
    outtextxy(1,(y1+y_wide/2)-12,"a");
    outtextxy(1,(y1+y_wide/2),"i");
    outtextxy(1,(y1+y_wide/2)+12,"n");
    setcolor(3);
    fvalue(x1,y2+12,"fBegin=",fbegin);
```

```
            fvalue(((x1+x2)/2)-6*8,y2+12,"delta f=",
                  (fend- fbegin)/float(scalex));
            fvalue(x2-8*13,y2+12,"fEnd=",fend);
            outtextxy(x1-20,y1,"1");
            outtextxy(x1-28,(y1+y2)/2-4,"0.5");
            outtextxy(x1-20,y2-8,"0");
            edit(x2+30,y1,x2+180,y_bar,cell,bar,0,0);
            setcolor(3);
            fvalue(x2+100,(yb[2]+ye[2])/2,"",fbegin);
            fvalue(x2+100,(yb[3]+ye[3])/2,"",fend);
            ivalue(x2+100,(yb[4]+ye[4])/2,"",Nx);
            ivalue(x2+100,(yb[5]+ye[5])/2,"",scalex);
          edit(x2+30,y1,x2+180,y_bar,cell,bar,0,0);
          scalexy(x1,y1,x_wide,y_wide,fbegin,fend,Nx,scalex);
          window(x1,y1,x_wide,y_wide,fbegin,fend,Nx,scalex,inportg,inportp,
                 outportfq);
                          }
                      }
                 else
                      { b=getch();if(b==13)b=27;}//exit edit
                      }
                 barsensor=0;
                 input=0;
                 fbegin=editvle[2];
                 fend=editvle[3];
                 Nx=editvle[4];
                 scalex=editvle[5];
                 clearviewport();
                }
          }
          closegraph();
          return 0;
          }
void window(int x1,int y1,int x_wide,int y_wide,float fbegin,float
fend,int Nx,int scalex,int inportg,int inportp,int outportfq)
  {
      float i,x2,y2,x3,x4,y4;
      float deltax,deltay,point;
      y2=y1+y_wide;
      x2=x1+x_wide;
        setcolor(YELLOW);
        rectangle(x1-1,y1-1,x2+1,y2+1);
        float scale,fout[1000],xscr[1000];
        int n,delphase;
        float datag[1000],datap[1000];
        if(Nx>=(x2-x1)) Nx=x2-x1;
        if(Nx<=1)Nx=1;
        scale=(fend-fbegin)/float(Nx);
        point=(x2-x1)/float(Nx);
        deltax=(x2-x1)/float(scalex);
        for(n=0;n<=Nx;n++)
        {
        fout[n]=fbegin+(n*scale);
        xscr[n]=x1+(n*point);
        int offsetg=0;
        int amplit=1;
            delphase=n*scale+fbegin;  //control freq
```

```
      crlword(outportfq,delphase);
      setcolor(7);
      outtextxy(x1,y2+12*3,"Loading");
      setcolor(BLUE);
      rectangle(x1-1,y2+52,x2+2,y2+65);
      setcolor(7);
      if(xscr[n]<x2-4)
       {
            outtextxy(xscr[n],y1+y_wide+55,"|");
       }
      delay((dl/fout[n])+50);
int gainin=0;
gainin=det();
 datag[xscr[n]]=offsetg+((y2+y1)/2)-(amplit)*
                linear(-((y_wide/2)-1),0,(y_wide/2)-1,255,gainin);
  }
     if(n>=Nx)
     {
      for(n=0;n<=Nx;n++)
       {
       setcolor(BLACK);
       if(xscr[n]<x2-2)
       {
            outtextxy(xscr[n],y1+y_wide+55,"|");
       }
       }
     }
 setcolor(BLACK);
 outtextxy(10,10,"operate...");//del oporate
 setcolor(3);
 for(n=1;n<=Nx;n++)
 {
 setcolor(BLACK);
 int i;
 if(n==1)
 {
 line(xscr[0],y1,xscr[0],y2);
 }
 for(i=xscr[n-1];i<=xscr[n];i++)
 {
 line(i+1,y1,i+1,y2);
 }
 scalexy(x1,y1,x_wide,y_wide,fbegin,fend,Nx,scalex);
 setcolor(3);
 if(xscr[n]==x2)
 setcolor(YELLOW);
 line(xscr[n]+1,y1,xscr[n]+1,y2);
 delay(5000/Nx);
float  xg1,yg1,xg2,yg2,xp1,yp1,xp2,yp2;
xg1=xscr[n-1];
yg1=datag[xscr[n-1]];
xg2=xscr[n];
yg2=datag[xscr[n]];
xp1=xscr[n-1];
yp1=datap[xscr[n-1]];
xp2=xscr[n];
yp2=datap[xscr[n]];
```

```
        setcolor(3);
        if(yg1>y1&&y1>yg2&&yg1<y2)
        {
             line(xg1,yg1,linear(xg1,yg1,xg2,yg2,y1),y1);
        }
        if(yg1<y1&&y1<yg2&&yg2<y2)
        {
              line(linear(xg1,yg1,xg2,yg2,y1),y1,xg2,yg2);
        }
        if(yg1>y2&&y2>yg2&&yg2>y1)
        {
              line(linear(xg1,yg1,xg2,yg2,y2),y2,xg2,yg2);
        }
        if(yg1<y2&&y2<yg2&&yg1>y1)
        {
              line(xg1,yg1,linear(xg1,yg1,xg2,yg2,y2),y2);
        }
        if(yg1>y1&&y1<yg2&&yg1<y2&&y2>yg2)
        {
              line(xg1,yg1,xg2,yg2);
        }
         if(yg1<y1&&yg2>y2)
        {

line(linear(xg1,yg1,xg2,yg2,y1),y1,linear(xg1,yg1,xg2,yg2,y2),y2);
        }
        if(yg1>y2&&yg2<y1)
        {

line(linear(xg1,yg1,xg2,yg2,y2),y2,linear(xg1,yg1,xg2,yg2,y1),y1);
        }
        scalexy(x1,y1,x_wide,y_wide,fbegin,fend,Nx,scalex);
        delay(500/Nx);
        }
  }
void scalexy(int x1,int y1,int x_wide,int y_wide,int fbegin,int fend,int
Nx,int scalex)
{
     int x2,y2,x3,x4,y4,nyg=10,nyp=10;
     float i,deltax,deltayg,deltayp,point;
     y2=y1+y_wide;
     x3=x1;
     x2=x1+x_wide;
      deltax=(x2-x1)/float(scalex);
      setcolor(8);
      for(i=1;i<scalex;i++)
      {
      line(x1+i*deltax,y1+1,x1+i*deltax,y1+3);
      line(x1+i*deltax,y2-1,x1+i*deltax,y2-3);
      setlinestyle(4,0x2222, 1);
      line(x1+i*deltax,y1+3,x1+i*deltax,y2-3);
      setlinestyle(0,0,1);
      }
      deltayg=(y2-y1)/float(nyg);
      deltayp=(y2-y1)/float(nyp);
      for(i=1;i<nyg;i++)
      {
```

```
        line(x1+1,y1+i*deltayg,x1+3,y1+i*deltayg);
        line(x2-1,y1+i*deltayg,x2-3,y1+i*deltayg);
        setlinestyle(4,0x2222, 1);
        line(x1+3,y1+i*deltayg,x2-3,y1+i*deltayg);
        setlinestyle(0,0,1);
        }
        setcolor(BLACK);
}
void fvalue(int x,int y,char* txt,float fin)
{
        // setcolor(WHITE);
         char fin1[20],fin2[20];
         int a,b,k=0,M=0;
         if(fin<=0)fin=0;
         if(fin>=1000&&fin<1000000){fin=fin/float(1000);k=1;}
         if(fin>=1000000){fin=fin/float(1000000);M=1,k=2;}
         a=fin;
         b=(fin-a)*1000;
         ltoa(a,fin1,10);
         ltoa(b,fin2,10);
         outtextxy(x,y,txt);
         outtextxy(x+8*strlen(txt),y,fin1);
         outtextxy(x+8*(strlen(txt)+strlen(fin1)),y,".");
         outtextxy(x+8*(strlen(txt)+strlen(fin1)+1),y,fin2);
         if(k==0)outtextxy(x+8*(strlen(txt)+strlen(fin1)+4),y,"Hz");
         if(k==1)outtextxy(x+8*(strlen(txt)+strlen(fin1)+4),y,"kHz");
         if(M==1)outtextxy(x+8*(strlen(txt)+strlen(fin1)+4),y,"MHz");
}
 float edit(int x1,int y1,int x2,int y2,int cell,int bar,float din,int b)
 {
  float delta,vle,loop=0;
  int n,cent[30],k=6,yb[30],ye[30];
  void *arrow;
   int edx, edy,wx,wy;
   unsigned int size;
   wx =300;
   wy =100;
   edx = x1-350;
   edy = y1+160;
  setcolor(8);//color out bar
  delta=(y2-y1)/float(cell);
  rectangle(x1,y1-1,x2,y2+1);
  x1=x1+2;
  x2=x2-2;
  for(n=0;n<=cell;n++)
  {
  yb[n]=((y1+1)+(n-1)*delta);
  ye[n]=((y1-1)+(n)*delta);
  }
  for(n=1;n<=cell;n++)
  {
  setcolor(8); //color bar
  if(n==bar&&b!=0)setcolor(WHITE);
```

```
if(n==1)
      {
      int ln;
      setcolor(GREEN);
      outtextxy(((x2+x1)/2)-2*8,(yb[1]+ye[1])/2,"Edit");
      setcolor(8);
      }
  rectangle(x1,yb[n],x2,ye[n]);
  setcolor(BLUE);
  outtextxy(x1+8,(yb[2]+ye[2])/2,"fBegin=");
  outtextxy(x1+8,(yb[3]+ye[3])/2,"fEnd  =");
  outtextxy(x1+8,(yb[4]+ye[4])/2,"fStep =");
  outtextxy(x1+8,(yb[5]+ye[5])/2,"fScale=");
  outtextxy(x1,(yb[6]+ye[6])/2,"     Setting");
  outtextxy(x1,(yb[cell-1]+ye[cell-1])/2,"    Exit  edit");
  outtextxy(x1,(yb[cell]+ye[cell])/2,     "       Exit");
  setcolor(BROWN);
  outtextxy(x1,ye[cell]+12,"Press key");
  outtextxy(x1,ye[cell]+12*2,"Puase= any key");
  outtextxy(x1,ye[cell]+12*3,"Edit =");
  char Edi[2];Edi[0]=24;Edi[1]=25;Edi[2]=32;//up down
  outtextxy(x1+8*7,ye[cell]+12*3,Edi);
  outtextxy(x1,ye[cell]+12*4,"Exit = esc");
  setcolor(8);
  outtextxy(x1,y1+360-12*2,"Telecommunications");
  outtextxy(x1,y1+360-12,  "  Engineering'44");
  outtextxy(x1,y1+360,     "       KMIT'L");
  setcolor(BLACK);
  }
  if(din!=0&&b==13)
  {
                  if(bar==2)
                  {
                  setcolor(3);
                  size = imagesize(edx, edy, edx+wx, edy+wy);
                  arrow = malloc(size);
                  getimage(edx, edy, edx+wx, edy+wy, arrow);
                  putimage(edx, edy, arrow,1);
                  setcolor(9);
                  rectangle(edx, edy, edx+wx, edy+wy);
                  setcolor(WHITE);
                  outtextxy(edx+8,edy+wy/2,"edit code=");
                  setcolor(8);
                  rectangle(edx+2, edy+2, edx+wx-2, edy+28);
                  setcolor(WHITE);
                  outtextxy(edx+wx/2-5*8,edy+12,"Edit Code");
                  vle=inedit(edx+8,edy+wy/2,din);
                  putimage(edx, edy, arrow,0);
                  free(arrow);
                  setcolor(BLACK);
                  fvalue(x1+68,(yb[2]+ye[2])/2,"",din);
                  setcolor(3);
                  fvalue(x1+68,(yb[2]+ye[2])/2,"",vle);
                  return vle;
                  }
```

```
if(bar==3)
{
setcolor(3);
size = imagesize(edx, edy, edx+wx, edy+wy);
arrow = malloc(size);
getimage(edx, edy, edx+wx, edy+wy, arrow);
putimage(edx, edy, arrow,1);
setcolor(9);
rectangle(edx, edy, edx+wx, edy+wy);
setcolor(WHITE);
outtextxy(edx+8,edy+wy/2,"edit code=");
setcolor(8);
rectangle(edx+2, edy+2, edx+wx-2, edy+28);
setcolor(WHITE);
outtextxy(edx+wx/2-5*8,edy+12,"Edit Code");
vle=inedit(edx+8,edy+wy/2,din);
putimage(edx, edy, arrow,0);
free(arrow);
setcolor(BLACK);
fvalue(x1+68,(yb[3]+ye[3])/2,"",din);
setcolor(3);
fvalue(x1+68,(yb[3]+ye[3])/2,"",vle);
return vle;
}
if(bar==4)
{
setcolor(3);
size = imagesize(edx, edy, edx+wx, edy+wy);
arrow = malloc(size);
getimage(edx, edy, edx+wx, edy+wy, arrow);
putimage(edx, edy, arrow,1);
setcolor(9);
rectangle(edx, edy, edx+wx, edy+wy);
setcolor(WHITE);
outtextxy(edx+8,edy+wy/2,"edit code=");
setcolor(8);
rectangle(edx+2, edy+2, edx+wx-2, edy+28);
setcolor(WHITE);
outtextxy(edx+wx/2-5*8,edy+12,"Edit Code");
vle=inedit(edx+8,edy+wy/2,din);
putimage(edx, edy, arrow,0);
free(arrow);
setcolor(BLACK);
ivalue(x1+68,(yb[4]+ye[4])/2,"",din);
setcolor(3);
ivalue(x1+68,(yb[4]+ye[4])/2,"",vle);
return vle;
}
if(bar==5)
{
setcolor(3);
size = imagesize(edx, edy, edx+wx, edy+wy);
arrow = malloc(size);
getimage(edx, edy, edx+wx, edy+wy, arrow);
putimage(edx, edy, arrow,1);
setcolor(9);
rectangle(edx, edy, edx+wx, edy+wy);
```

```
setcolor(WHITE);
outtextxy(edx+8,edy+wy/2,"edit code=");
setcolor(8);
rectangle(edx+2, edy+2, edx+wx-2, edy+28);
setcolor(WHITE);
outtextxy(edx+wx/2-5*8,edy+12,"Edit Code");
vle=inedit(edx+8,edy+wy/2,din);
putimage(edx, edy, arrow,0);
free(arrow);
setcolor(BLACK);
ivalue(x1+68,(yb[5]+ye[5])/2,"",din);
setcolor(3);
ivalue(x1+68,(yb[5]+ye[5])/2,"",vle);
return vle;
}
if(bar==6)
{
setcolor(3);
size = imagesize(edx, edy, edx+wx, edy+wy);
arrow = malloc(size);
getimage(edx, edy, edx+wx, edy+wy, arrow);
putimage(edx, edy, arrow,1);
setcolor(9);
rectangle(edx, edy, edx+wx, edy+wy);
setcolor(WHITE);
outtextxy(edx+8,edy+wy/2,"edit code=");
setcolor(8);
rectangle(edx+2, edy+2, edx+wx-2, edy+28);
setcolor(WHITE);
outtextxy(edx+wx/2-5*8,edy+12,"Edit Code");
vle=inedit(edx+8,edy+wy/2,din);
putimage(edx, edy, arrow,0);
free(arrow);
setcolor(BLACK);
ivalue(x1+68,(yb[6]+ye[6])/2,"",din);
setcolor(3);
ivalue(x1+68,(yb[6]+ye[6])/2,"",vle);
return vle;
}
if(bar==7)
{
setcolor(3);
size = imagesize(edx, edy, edx+wx, edy+wy);
arrow = malloc(size);
getimage(edx, edy, edx+wx, edy+wy, arrow);
putimage(edx, edy, arrow,1);
setcolor(9);
rectangle(edx, edy, edx+wx, edy+wy);
setcolor(WHITE);
outtextxy(edx+8,edy+wy/2,"edit code=");
setcolor(8);
rectangle(edx+2, edy+2, edx+wx-2, edy+28);
setcolor(WHITE);
outtextxy(edx+wx/2-5*8,edy+12,"Edit Code");
vle=inedit(edx+8,edy+wy/2,din);
putimage(edx, edy, arrow,0);
free(arrow);
```

```
                    setcolor(BLACK);
                    ivalue(x1+68,(yb[7]+ye[7])/2,"",din);
                    setcolor(3);
                    ivalue(x1+68,(yb[7]+ye[7])/2,"",vle);
                    return vle;
                    }
  }
  return din;
 }

float inedit(int x,int y,float din)
{
   char input[]="                                                    ",
*endptr;
   double value=din;
   int i=0;
   setcolor(3);
   while(input[i-1]!=13&&input[i-1]!=27)
   {
   input[i]=getch();
   if(input[i]==8){input[i]=32;i=i-1;}
   outtextxy(x+8*11,y,input);
   setcolor(3);
   if(i==24)return din;//end txt deit
   i=i+1;
   }
   value = strtod(input, &endptr);
   if(value==0||input[i-1]==27)return din;
   return value;
   return 0;
}
 void ivalue(int x,int y,char* txt,int fin)
{
      // setcolor(WHITE);
       char fin1[20];
       if(fin<=0)fin=0;
       ltoa(fin,fin1,10);
       outtextxy(x,y,txt);
       outtextxy(x+8*strlen(txt),y,fin1);
}
void crlword(int PORT,double F_out)
{
      int i;
      unsigned long clock_in=12000000;
      double f;
      unsigned long delta_phase;
      outportb(PORT, 0x00);
            if (F_out<=0.0||F_out>clock_in)
            exit(0);
            delta_phase = 0.5+(F_out* pow(2, 28)) / clock_in;
            f= (delta_phase*1.0*clock_in)/double(pow(2,28));
      for (i = 0; i < 20; i++)
      {
```

```
            if ((delta_phase & 1) == 1)
                  {
                  outportb(PORT, 1);/* CLOCK low and DATA high - 01 */
                  outportb(PORT, 3);/* CLOCK and DATA high - 11 */
                  outportb(PORT, 0);/* CLOCK low and DATA low - 00 */
                  }
            else
                  {
                  outportb(PORT, 0);/* CLOCK low and DATA low - 00 */
                  outportb(PORT, 2);/* CLOCK high and DATA low - 10 */
                  outportb(PORT, 0);/* CLOCK low and DATA low - 00 */
                  }
                  delta_phase =delta_phase  >> 1;
      }
            outportb(PORT, 0x04);      //up date frequency
            delay(10);
            outportb(PORT, 0x00);
            setcolor(RED);
            outtextxy(10,10,"operate...");
}
float linear(float x1,float y1,float x2,float y2,float y)
{
  return (((x2-x1)*(y-y1))/float(y2-y1))+x1;
}
int det(void)
{
int
i,deca,decb,dataa[5],datab[5],bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7;
int level;
outportb(0x378,0+16);
deca=inportb(0x379)/8;
      for (i = 0; i < 5; i++)
      {
      dataa[i]=deca&1;
      deca = deca >> 1;
      }
bit7=dataa[3];bit6=!dataa[4];bit5=dataa[2];bit4=dataa[0];
outportb(0x378,8+16);
decb=inportb(0x379)/8;
      for (i = 0; i < 5; i++)
      {
      datab[i]=decb&1;
      decb = decb >> 1;
      }
bit3=datab[3];bit2=!datab[4];bit1=datab[2];bit0=datab[0];
level=bit0+bit1*2+bit2*4+bit3*8+bit4*16+bit5*32+bit6*64+bit7*128;
outportb(0x378,0);
return level;
}
```

โปรแกรมภาษาVHDL

```
library ieee;
use ieee.std_logic_1164.all;
entity ddscomp is
port (tuningword1,up_fq1,clk,w_clk1 : in std_logic;
                              data : in std_logic_vector(7 downto 0);
                      clkr,rdin: in std_logic;
      rdout,reset,fsampling: out std_logic;
                          dataout: out std_logic_vector(3 downto 0);
                   fout1 : out std_logic_vector(7 downto 0));
end;
architecture dds of ddscomp is
      component deltaphase is
port ( tuningword,w_clk : in std_logic;
                dphase : out std_logic_vector(19 downto 0));
      end component;
      component sin is
port (deltaphase : in  std_logic_vector(19 downto 0);
       clk,up_fq : in  std_logic;
  reset,fsampling: out std_logic;
            fout : out std_logic_vector(7 downto 0));
      end component;
      component det is
port (data : in std_logic_vector(7 downto 0);
 clkr,rdin : in std_logic;
       rdout : out std_logic;
           q : out std_logic_vector(3 downto 0) );
      end component;
signal i1:std_logic_vector(19 downto 0);
begin
      u1:deltaphase port map(tuningword1,w_clk1,i1);
      u2:sin        port map(i1,clk,up_fq1,reset,fsampling,fout1);
      u3:det        port map(data,clkr,rdin,rdout,dataout);
end;
```

```
library ieee;
use ieee.std_logic_1164.all;
entity deltaphase is
port ( tuningword,w_clk : in std_logic;
                  dphase : out std_logic_vector(19 downto 0));
end;
architecture deltaphase of deltaphase is
begin
process(w_clk)
variable  a : std_logic_vector(19 downto 0);
variable  b : std_logic_vector(18 downto 0);
variable dphaseacc : std_logic_vector(20 downto 0);
begin
      if w_clk'event and w_clk='1' then
          a(19):=tuningword;
            dphase(19 downto 0)<=a(19 downto 0);
            b(18 downto 0):=a(19 downto 1);
            a(18 downto 0):=b(18 downto 0);
      end if;
end process;
end;
-------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity sin is
port (deltaphase : in  std_logic_vector(19 downto 0);
      clk,up_fq : in  std_logic;
  reset,fsampling: out std_logic;
            fout : out std_logic_vector(7 downto 0));
end;
architecture dds of sin is
begin
process(clk)
            variable sin      : integer range 0 to 63;
        variable table    : integer range 0 to 255;
        variable dphase   : integer range 0 to 1048575;
            variable phase    : integer range 0 to 268435455;
      variable f_out    : std_logic_vector(7 downto 0);
            variable sampling : std_logic_vector(13 downto 0);
begin
reset<=up_fq;
-
if clk'event and clk='1' then
frequency
      if up_fq='1' then
    dphase:=conv_integer(deltaphase);
    end if;
sampling
sampling:=sampling+'1';
fsampling<=sampling(13);
phase:=phase+dphase;
table:=phase/1048576;
if table>=0   and table <=63 then
          sin:=table;
end if;
```

```
if table>=64 and table <=127 then
          sin:=127-table;
end if;
if table>=128 and table <=191 then
          sin:=table-128;
end if;
if table>=192 and table <=255 then
          sin:=255-table;
end if;
case sin is
           when 0=> f_out(7 downto 0) :="10000001" ;
           when 1=> f_out(7 downto 0) :="10000100" ;
           when 2=> f_out(7 downto 0) :="10000111" ;
           when 3=> f_out(7 downto 0) :="10001010" ;
           when 4=> f_out(7 downto 0) :="10001110" ;
           when 5=> f_out(7 downto 0) :="10010001" ;
           when 6=> f_out(7 downto 0) :="10010100" ;
           when 7=> f_out(7 downto 0) :="10010111" ;
           when 8=> f_out(7 downto 0) :="10011010" ;
           when 9=> f_out(7 downto 0) :="10011101" ;
           when 10=> f_out(7 downto 0) :="10100000" ;
           when 11=> f_out(7 downto 0) :="10100011" ;
           when 12=> f_out(7 downto 0) :="10100110" ;
           when 13=> f_out(7 downto 0) :="10101001" ;
           when 14=> f_out(7 downto 0) :="10101100" ;
           when 15=> f_out(7 downto 0) :="10101111" ;
           when 16=> f_out(7 downto 0) :="10110010" ;
           when 17=> f_out(7 downto 0) :="10110101" ;
           when 18=> f_out(7 downto 0) :="10110111" ;
           when 19=> f_out(7 downto 0) :="10111010" ;
           when 20=> f_out(7 downto 0) :="10111101" ;
           when 21=> f_out(7 downto 0) :="11000000" ;
           when 22=> f_out(7 downto 0) :="11000010" ;
           when 23=> f_out(7 downto 0) :="11000101" ;
           when 24=> f_out(7 downto 0) :="11001000" ;
           when 25=> f_out(7 downto 0) :="11001010" ;
           when 26=> f_out(7 downto 0) :="11001101" ;
           when 27=> f_out(7 downto 0) :="11001111" ;
           when 28=> f_out(7 downto 0) :="11010010" ;
           when 29=> f_out(7 downto 0) :="11010100" ;
           when 30=> f_out(7 downto 0) :="11010110" ;
           when 31=> f_out(7 downto 0) :="11011001" ;
           when 32=> f_out(7 downto 0) :="11011011" ;
           when 33=> f_out(7 downto 0) :="11011101" ;
           when 34=> f_out(7 downto 0) :="11011111" ;
           when 35=> f_out(7 downto 0) :="11100001" ;
           when 36=> f_out(7 downto 0) :="11100011" ;
           when 37=> f_out(7 downto 0) :="11100101" ;
           when 38=> f_out(7 downto 0) :="11100111" ;
           when 39=> f_out(7 downto 0) :="11101001" ;
           when 40=> f_out(7 downto 0) :="11101010" ;
           when 41=> f_out(7 downto 0) :="11101100" ;
           when 42=> f_out(7 downto 0) :="11101110" ;
           when 43=> f_out(7 downto 0) :="11101111" ;
           when 44=> f_out(7 downto 0) :="11110001" ;
           when 45=> f_out(7 downto 0) :="11110010" ;
           when 46=> f_out(7 downto 0) :="11110011" ;
```

```
            when 47=> f_out(7 downto 0) :="11110101" ;
            when 48=> f_out(7 downto 0) :="11110110" ;
            when 49=> f_out(7 downto 0) :="11110111" ;
            when 50=> f_out(7 downto 0) :="11111000" ;
            when 51=> f_out(7 downto 0) :="11111001" ;
            when 52=> f_out(7 downto 0) :="11111010" ;
            when 53=> f_out(7 downto 0) :="11111011" ;
            when 54=> f_out(7 downto 0) :="11111100" ;
            when 55=> f_out(7 downto 0) :="11111100" ;
            when 56=> f_out(7 downto 0) :="11111101" ;
            when 57=> f_out(7 downto 0) :="11111101" ;
            when 58=> f_out(7 downto 0) :="11111110" ;
            when 59=> f_out(7 downto 0) :="11111110" ;
            when 60=> f_out(7 downto 0) :="11111111" ;
            when 61=> f_out(7 downto 0) :="11111111" ;
            when 62=> f_out(7 downto 0) :="11111111" ;
            when 63=> f_out(7 downto 0) :="11111111" ;
end case;
                        if table>=0 and table <=127 then
                  fout(7 downto 0)<=f_out(7 downto 0);
                  end if;
                        if table>=128 and table <=255 then
                        fout(7 downto 0)<=not f_out(7 downto 0);
                        end if;
        end if;
          end process;
end;


-------------------------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
entity det is
      port (data : in std_logic_vector(7 downto 0);
            clkr,rdin : in std_logic;
                        rdout : out std_logic;
                  q : out std_logic_vector(3 downto 0) );
      end det;
architecture det of det is
begin
process(clkr,data(7 downto 0))
begin
      rdout<=rdin;
                  if rdin='1' then

                  if clkr='1' then
                              q(3 downto 0)<=data(3 downto 0);
                              else
                              q(3 downto 0)<=data(7 downto 4);

                        end if;
                        else
                              q(3 downto 0)<="00000000";
                  end if;
end process;
end det;
```