การควบคุมจอแสดงผลแบบวีจีเอด้วยอุปกรณ์เอฟพีจีเอ

VGA Controlling using FPGA

โดย

นายศรัณย์ ตันติเสนีพงศ์

นายศุภการ งามเบญจกุล

นายสกนธ์ ศิลปกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ผ่านการตรวจชิ้นงานแล้ว

(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

การควบคุมแสดงผลแบบวีจีเอด้วยอุปกรณ์เอฟพีจีเอ

VGA Controlling using FPGA

โดย

นายศรัณย์ ตันติเสนีพงศ์ 45010744

นายศุภการ งามเบญจกุล 45010775

นายสกนธ์ ศิลปกุล 45010788

อาจารย์ที่ปรึกษา

อ.ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การควบคุมจอแสดงผลแบบวีจีเอด้วยอุปกรณ์เอฟพีจีเอ**

VGA Controlling using FPGA

ผู้จัดทำ

1.นายศรัณย์ตันติเสนีพงศ์ 45010744

2.นายคุภการ งามเบญจกุล 45010775

3.นายสกนธ์ ศิลปกุล 45010788

..... อาจารย์ที่ปรึกษา

(อ.ศรวัฒน์ ชิวปรีชา)

การควบคุมจอแสดงผลแบบ VGA โดยใช้ FPGA

VGA Controlling using FPGA

โดย นายศรัณย์ ตันติเสนีย์พงศ์ 45010744
นายศุภการ งามเบญจกุล 45010775
นายสกนธ์ ศิลปกุล 45010788

( )
อาจารย์ที่ปรึกษา อ.ศรวัฒน์ ชิวปรีชา

บทคัดย่อ

โครงงานนี้นำเสนอการสร้างภาพนิ่งและภาพเคลื่อนไหวบนจอแสดงผลแบบ VGA โดยได้นำอุปกรณ์ประเภท FPGA (Field Programmable Gate Array) มาใช้ในการออกแบบวงจร ซึ่งใช้ภาษา VHDL ในการอธิบายการทำงาน และสามารถใช้งานได้ทันทีเมื่อนำไปต่อกับจอภาพโดยสามารถแสดงผลได้ทั้งภาพสีและขาวดำ สามารถรับข้อมูลจาก keyboard และแสดงผลเป็นตัวหนังสือบนจอภาพได้

ABSTRACT

This project presents how to build still picture and motion picture on VGA monitor using FPGA (Field Programmable Gate Array) for circuit design. Using VHDL describe how it work. It can active immediately while connect to monitor and it can show result in gray or color images. It can be able to received input from keyboard and display text on the monitor.

สารบัญ

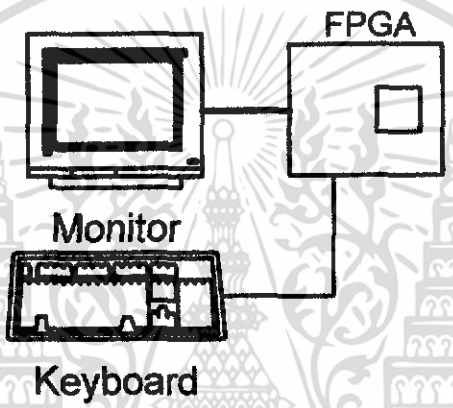
สารบัญรูปภาพ

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

ในโครงงานนี้จะเป็นการประยุกต์ ใช้อุปกรณ์ที่เรียกว่า เอฟพีจีเอ (FPGA) หรือProgrammable Gate Array Field ซึ่งจะสามารถออกแบบวงจรรวมได้ง่ายและสะดวกขึ้น เพื่อที่จะควบคุมการยิงของลำอิเล็กตรอนของหลอดภาพ และสามารถที่จะแสดงผลออกมาเป็นภาพออกทางจอภาพ โดยที่หลอดภาพโดยทั่วไปก็จะประกอบด้วยลำอิเล็กตรอนสีหลักคือ สีแดง สีเขียวและสีน้ำเงิน ซึ่งสีเหล่านี้สามารถผสมแล้วจะได้สีต่างๆ แต่ก่อนที่จะออกมาเป็นภาพที่เราเห็นได้นั้น จะต้องมีการควบคุมการยิงของลำอิเล็กตรอนให้เคลื่อนที่ไปสัมพันธ์พร้อมๆ กันกับสัญญาณที่เรียกว่า สัญญาณซิงก์โครไนซ์ เมื่อควบคุมสัญญาณของสีเหล่านี้ได้ ก็จะทำให้ได้ออกมาเป็นภาพที่แสดงบนจอภาพ



รูปที่ 1.1 โครงสร้างโดยรวมของชิ้นงาน

1.1 วัตถุประสงค์

1. ศึกษาการใช้และประยุกต์ภาษา VHDL เพื่อควบคุม FPGA
2. ศึกษาการทำงานของหลอดภาพแบบ CRT (Cathode Ray Tube)
3. ศึกษาการใช้งานประยุกต์การรับค่าจากคีย์บอร์ดได้
4. ประยุกต์การเรียกใช้งานหน่วยความจำภายในได้
5. ประยุกต์การสร้างภาพเคลื่อนไหว และอักขระต่างๆ ได้

1.2 ขอบเขตของโครงงาน

ในส่วนของเทอมนี้ จะควบคุมให้อุปกรณ์เอฟพีจีเอสามารถสร้างสัญญาณภาพเคลื่อนไหวแบบเฟรมได้ จนกระทั่งสามารถป้อนค่าผ่านทางคีย์บอร์ด แล้วนำไปแสดงผลออกทางจอภาพได้อย่างถูกต้อง

1.3 ประโยชน์ที่ได้รับ

1. สามารถประยุกต์ใช้งานอุปกรณ์ FPGA ได้อย่างกว้างขวาง
2. สามรถเขียนโปรแกรมควบคุมการทำงานของ FPGA ได้
3. สามารถควบคุมการทำงานของหลอดภาพ CRT ได้

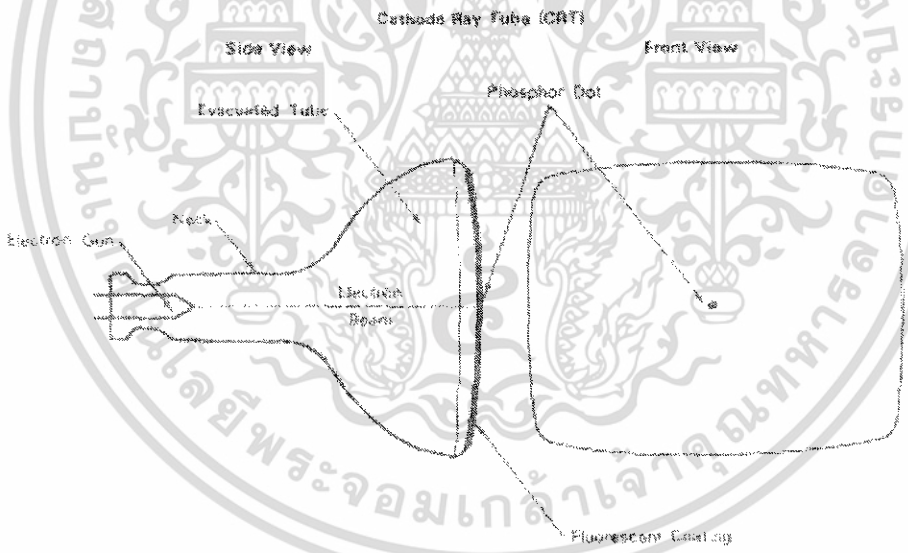
บทที่ 2
ทฤษฎีและหลักการ

2.1 การกำเนิดการแสดงผลภาพให้กับจอภาพระบบ VGA (VGA Video Display Generator)

สัญญาณที่ใช้ในระบบภาพ VGA ประกอบด้วย 5 สัญญาณด้วยกัน โดยมี 2 สัญญาณที่เกี่ยวข้องกับระดับสัญญาณลอจิก TTL นั่นคือ สัญญาณ Horizontal Sync และ Vertical Sync ซึ่งใช้สำหรับสังเคราะห์สัญญาณภาพ ส่วนอีก 3 สัญญาณนั้นจะเกี่ยวข้องสัญญาณอนาลอก โดยมีค่า $0.7 - 1.0 V_{p-p}$ จะใช้ในควบคุมสัญญาณสี ซึ่งสัญญาณสีประกอบด้วยสีแดง, สีเขียว และสีน้ำเงิน สัญญาณสีเหล่านี้เรียกรวมว่า สัญญาณ RGB

2.2 เทคโนโลยีการแสดงผลภาพจอ (Video Display Technology)

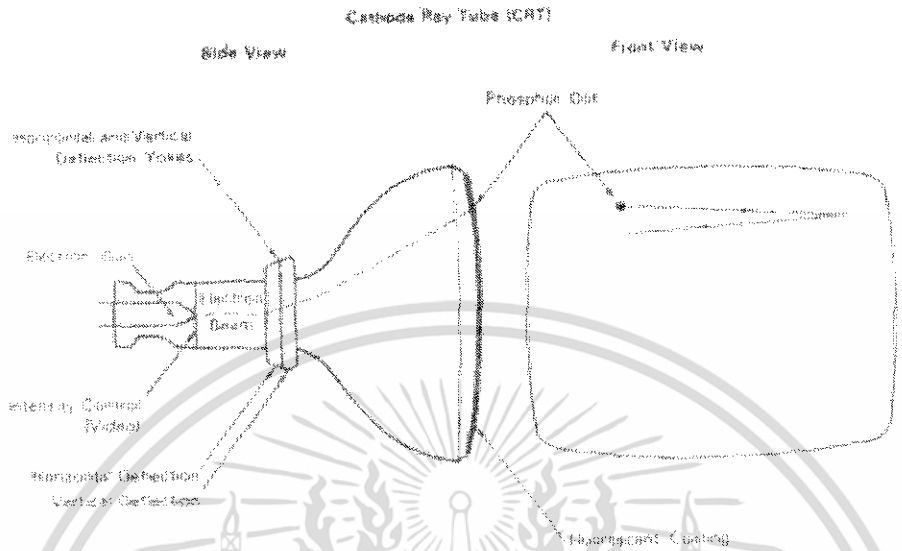
เทคโนโลยีที่ใช้โดยส่งการแสดงผลเป็นสัญญาณภาพนิ่งที่เหมือนธรรมชาติในรูปแบบของสัญญาณภาพ จะใช้หลอดภาพอิเล็กทรอนิกส์ (Cathode ray tube: CRTs) เป็นอุปกรณ์ที่ใช้กันแพร่หลายทั่วไป ส่วนมากจะมีใช้ในรูปแบบชุดโทรทัศน์บ้าง ใช้เป็นตัวเชื่อมต่อในการแสดงผลของระบบคอมพิวเตอร์ก็ดี



รูปที่ 2.1 แสดงถึงอุปกรณ์ CRT ซึ่งก็คือหลอดแก้วที่ยิงอิเล็กตรอนไปยังฉากด้านหน้า และฉากนี้เองที่พื้นผิวด้านในจะเคลือบสารฟลูเรชเซนท์อยู่ และที่ตรงส่วนท้ายของหลอดภาพจะมีปืนยิงลำอิเล็กตรอน เมื่อปืนนี้ยิงไปกระตุ้นที่ฉากก็จะทำให้เกิดจุดสว่างขึ้น

ซึ่งหลอดอิเล็กตรอนนี้จะยิงลำอิเล็กตรอน (Electron beam) กวาดสแกนบนฉากตามแนวนอนที่ต่อเนื่องกัน และเพื่อที่จะกำเนิดมาเป็นภาพ จะมีแกนเหล็กสำหรับเบี่ยงเบนลำอิเล็กตรอน (Deflection yoke) ขดลวดเลี้ยวเบนนี้จะแบ่งเป็นแกนแนวตั้งกับแนวนอน (จะอยู่ที่คอของหลอด CRT) โดยจะอาศัยสนามแม่เหล็ก (หรืออาจจะใช้สนามไฟฟ้าสถิตก็ได้ แต่ไม่เป็นที่นิยม) เบี่ยงเบนลำอิเล็กตรอนให้ไปตามตำแหน่งที่ต้องการบนฉากของจอภาพ ส่วนข้อมูลของสัญญาณ RGB ที่ออกมาเป็นสัญญาณภาพจะใช้

ควบคุมความแรงของอิเล็กตรอน และแสงที่เกิดขึ้นบนจอภาพนั้นเกิดจากการที่มีสัญญาณภาพเข้ามาซึ่งจะทำให้ลำอิเล็กตรอนทำงาน และสามารถพุ่งไปปะทะจุดสีเรืองแสงบนฉากของหลอดภาพ CRT ซึ่งบนฉากของ CRT จะเคลือบด้วยสารเรืองแสง 3 สี (สีแดง, สีเขียว และสีน้ำเงิน)



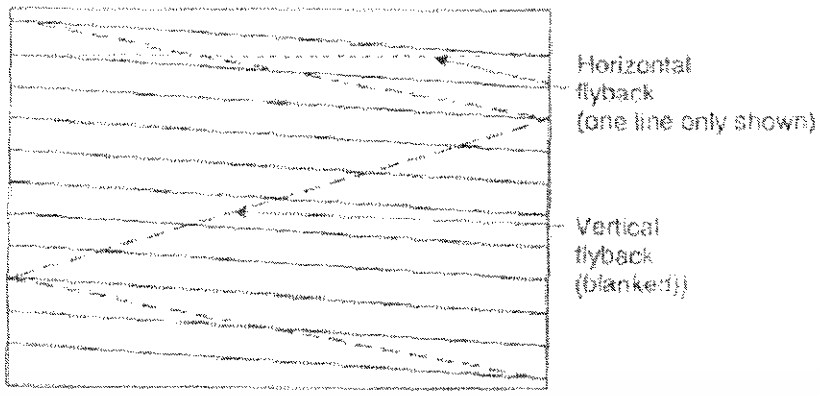
รูปที่ 2.2 การควบคุมการเคลื่อนที่ของลำอิเล็กตรอน

2.3 การกวาดตรวจภาพ (Scanning)

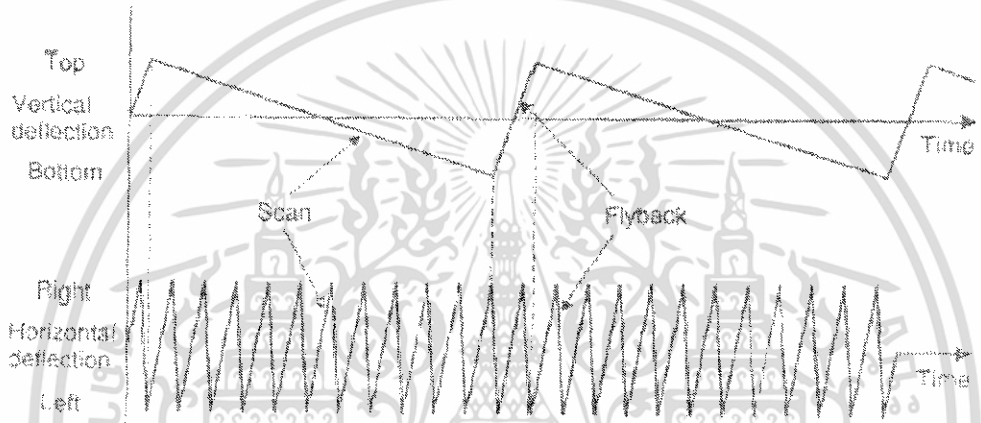
การที่จะนำภาพ 2 มิติเคลื่อนย้ายจากที่หนึ่งไปยังอีกที่หนึ่งทำได้โดยทำภาพนั้นเป็นสัญญาณไฟฟ้าซึ่งจะง่ายต่อการเคลื่อนย้าย แต่ปัญหาคือจะทำการเปลี่ยนภาพ 2 มิติไปเป็นโวลต์เวจที่เปลี่ยนตามเวลาปัญหานี้แก้ได้โดยทฤษฎีของการกราดตรวจ

วิธีการสแกนของ CRT ที่นิยมใช้กันอย่างแพร่หลายในการออกแบบทางการค้าโทรทัศน์และจอภาพแสดงผล มีชื่อเรียกว่า *การตรวจกวาดแบบราสเตอร์ (Raster scan)* ซึ่งวิธีนี้จะต้องมีการให้กำเนิดซิงก์ทั้งในแนวแกนตั้งและแนวนอนเพื่อให้ลำอิเล็กตรอนสามารถเคลื่อนที่ได้

ในการสแกนแบบนี้จะเริ่มทำจากซ้ายไปขวา และจากข้างบนลงไปข้างของจอภาพ โดยที่ลำอิเล็กตรอนจะถูกเลี้ยวเบนให้ไปอยู่ทางมุมซ้ายบนและเมื่อกวาดไปทางขวาของฉาก ลำอิเล็กตรอนก็จะสะบัดกลับโดยตอนที่กลับนั้นก็จะมีการเลื่อนตำแหน่งลงไปด้วย การสะบัดกลับนี้เรียกว่า Retrace หรือfly back ซึ่งในส่วนของการสะบัดกลับของลำอิเล็กตรอนนั้นความเข้มของลำอิเล็กตรอนจะลดลง เมื่อลดความเข้มลงอย่างเพียงพอแล้วฉากที่เคลือบด้วยฟลูเรซเซนท์ก็จะไม่ถูกกระตุ้นทำให้ไม่เกิดแสงสว่าง จึงมองไม่เห็นเส้นที่ลากกลับ ดังรูปที่ 2.3



(ก)



(ข)

รูปที่ 2.3 การสแกนภาพแบบราสเตอร์

(ก) การเคลื่อนที่ของการสแกนภาพ

(ข) กราฟแสดงความสัมพันธ์ของการเคลื่อนที่ของการสแกนกับเวลา

2.4 การซิงก์โครไนซ์ (Synchronization)

สิ่งสำคัญที่ทำให้การสแกนภาพนิ่งได้ถูกต้องนั้นเป็นหน้าที่ของสัญญาณซิงก์ ซึ่งแบ่งได้เป็นสัญญาณ Horizontal และ Vertical sync นั้นทำงานไปพร้อมๆ กันได้นั้นเพราะสัญญาณนี้จะส่งข้อมูล timing แนบไปกับสัญญาณ video

การซิงก์โครไนซ์ในแนวแกนนอนและแกนตั้ง

- Horizontal Sync จะเริ่มทำงานที่การเริ่ม retrace หรือจบการ trace จะไม่ทำตอนเริ่ม trace
- Vertical Sync เริ่มทำงานที่การเริ่ม retrace ในแนวแกนตั้ง (ก็คือจะทำงานตอน beam อยู่ที่ล่างขวาของจอภาพ)

ถ้าหากไม่มี สัญญาณ Vertical sync จะทำให้ภาพที่ถูกสร้างทวนซ้ำขึ้นมาจะไม่สามารถคงตำแหน่งในแนวแกนตั้ง ภาพจะวิ่งขึ้น-ลง แต่ถ้าหากสัญญาณ Horizontal sync ไม่ซิงก์กันจะทำให้ภาพนั้นจะเลื่อนซ้าย-ขวา และบางส่วนจะเหมือนภาพถูกแยกออก

โดยทั่วไปแล้วจอภาพส่วนใหญ่จะใช้สัญญาณอนาลอก RGB มาตรฐานที่ $0.7 V_{p-p}$ ส่วนสัญญาณซิงก์ก็จะขึ้นอยู่กับประเภทลักษณะที่ใช้ดังต่อไปนี้

1. RGB + HSYNC + VSYNC จะใช้สายในการเชื่อมต่อ 5 เส้น มาตรฐานของสัญญาณ RGB ที่ใช้คือ $0.7 V_{p-p}$ และสัญญาณซิงก์ใช้มาตรฐาน TTL หรือ $1 V_{p-p}$ ขึ้นอยู่กับระบบ ลักษณะการใช้งานแบบนี้จะง่ายและเป็นที่นิยม
2. RGB + Composite sync จะใช้สายในการเชื่อมต่อ 4 เส้น เพราะจะใช้สัญญาณซิงก์ 1 เส้น (เรียก CSYNC) ใช้มากในพวกโปรเจคเตอร์ในสมัยก่อน
3. RGB + Sync on green จะใช้สายในการเชื่อมต่อ 3 เส้น สัญญาณ R และ B ยังคงใช้ $0.7 V_{p-p}$ แต่สัญญาณ G จะส่งรวมไปกับสัญญาณ CYNC ซึ่งจะใช้งานที่ $-0.3 V_{p-p}$

2.5 อัตราการสแกนภาพ (Scan Rate)

ในมาตรฐานของระบบภาพของ USA อัตราการกวาดเส้นในแนวนอน (The horizontal scanning frequency or line rate) ของลำอิเล็กตรอนสำหรับโทรทัศน์คือ 31.5 *kHz* ซึ่ง และถ้าต้องการให้เคลื่อนที่ไปพร้อมกันในแนวตั้ง (The vertical scanning frequency or frame rate or field rate) มีอัตราการกวาดที่ 60 *Hz* ซึ่งก็จะทำให้เกิดเส้นกวาดในแนวนอนจากบนสุดจนถึงล่างสุดของฉากทั้งหมดเท่ากับ $31500 \div 60 = 525$ เส้น

จำนวนสัญลักษณ์ที่จะแสดงได้ใน 1 แถวนั้นจะขึ้นอยู่กับความถี่ของสัญญาณภาพที่ใช้ ส่วนจำนวนบรรทัดที่จะแสดงผลบนจอภาพนั้นจะขึ้นอยู่กับอัตราการสแกนภาพ ซึ่งสามารถเพิ่มได้ วิธีที่เห็นได้ชัดที่สุดก็คือการเพิ่มความถี่ในการกวาดในแนวนอน ซึ่งก็จะทำให้เกิดเส้นสแกนแนวนอนมากขึ้น ตัวอย่างเช่น

ถ้าเพิ่มความถี่ในแนวนอนจาก 31.5 *kHz* เป็น 36 *kHz* แล้วจะทำให้เกิดเส้นกวาดแนวนอนเท่ากับ $36000 \div 60 = 600$ เส้น แต่อย่างไรก็ตามการเพิ่มความถี่การกวาดแนวนอนนั้น ก็จะทำให้ผลลัพธ์ที่ได้ไม่เป็นมาตรฐานเช่นกัน นอกจากนี้ยังต้องเพิ่มความถี่ Video เพื่อให้จำนวนสัญลักษณ์เหมาะสมกับจำนวนตัวสัญลักษณ์ต่อเส้นกวาดแนวนอน

2.6 อัตราการแสดงภาพใหม่ (Refresh rate)

คือ อัตราส่วนของจำนวนครั้งที่จุดภาพ (Pixel) ถูกฉายลงบนหน้าจอภาพ ซึ่งจะเริ่มจากซ้ายไปขวาและบนลงล่าง ในเวลา 1 วินาที ซึ่งเป็นค่าที่สำคัญเพราะว่ามีผลกระทบโดยตรงต่อการมองเห็นภาพที่แสดงบนหน้าจอภาพ

ปัญหาหลักของการเพิ่มเส้นสแกนนั้นคือ ถ้าอัตราการ Refresh แนวตั้งนั้นต่ำกว่ากับความถี่ power line ของ CRT ก็จะทำให้ลำอิเล็กตรอนเลี้ยวเบนด้วยสนามไฟฟ้าที่มีรูปแบบไม่ถูกต้อง ผลกระทบนี้เป็นเหตุให้เกิดแสงภาพของจอกระตุกสั่น การแก้ปัญหาก็คือเราควรใช้ refresh rate นั้นให้เท่าความถี่ของ power line ดังนั้นใน USA ที่ความถี่การกวาดในแนวนอนเท่ากับ 31.5 *kHz* ควรใช้ความถี่ในการ refresh rate มากกว่าหรือเท่ากับ 60 *Hz* ในขณะที่ประเทศไทยก็ใช้ตั้งแต่ความถี่ 50 *Hz* ขึ้นไป เพราะถ้าอัตราการ

Refresh น้อยไปก็จะทำให้ภาพนั้นสั่นกระพริบ เป็นอันตรายต่อสายตา ค่าอัตราการ Refresh นี้ก็จะขึ้นสายตาของแต่คน และก็ยังจะขึ้นอยู่กับขนาดของจอภาพด้วยเช่นกัน โดยที่จอภาพขนาดใหญ่จะมีการสั่นของภาพได้ง่ายกว่าจอภาพที่มีขนดเล็ก ฉะนั้นจอภาพขนาดใหญ่ควรใช้อัตราการ Refresh สูงกว่าจอภาพขนาดเล็ก

อย่างไรก็ตามยังมีวิธีการลดการกระพริบของการสแกนภาพนิ่งแต่ละภาพ ซึ่งจะนิยมใช้วิธีการสแกนไขว้กันหรือ Interlace scanning ตามรูปที่ 2.4 โดยที่ภาพหนึ่งเฟรมจะประกอบด้วยภาพนิ่ง 2 ฟิลด์ โดยที่เริ่มต้นด้วยการสแกนภาพนิ่งฟิลด์เส้นคี่ก่อน เมื่อเสร็จสิ้นถึงตำแน่งกลางของแนวเส้นสุดท้าย ลำอิเล็กตรอนก็จะกลับไปเริ่มสแกนใหม่แต่จะสแกนที่ฟิลด์ของเส้นคู่ หลังจากที่สแกนเสร็จทั้ง 2 ฟิลด์แล้วก็จะได้ภาพหนึ่งเฟรม

รูปที่ 2.4 การสแกนแบบไขว้กัน

การกวาดสแกนสัญญาณภาพหรือการเริ่มสแกนใหม่ จะเป็นไปตามกระบวนการของสัญญาณ Vertical Sync ซึ่งจะแสดงถึงการเริ่มแสดงภาพใหม่หรือ Frame ใหม่ ส่วนสัญญาณ Horizontal Sync จะแสดงถึงการเริ่มใหม่ในแถว สัญญาณ Vertical sync ก็จะทำการ reset ตำแหน่งลำอิเล็กตรอนไปที่มุมซ้ายบน (จุด (0, 0) นั่นเอง) และเริ่มทำกระบวนการเดิมไปเรื่อยๆ ในขณะที่กลับไปตำแหน่งเริ่มต้นสัญญาณ RGB จะมีค่าเป็น “000” หรือสีดำนั่นเอง แต่ถ้าสัญญาณ Sync ทั้ง 2 ค่านั้นไม่สอดคล้องกันแล้ว ส่วนใหญ่ในปัจจุบันจอภาพจะมี LED แสดงสถานะ ถ้า LED มีสีเขียวแสดงว่ามีการตรวจพบสัญญาณ Sync และถ้า LED เป็นสีเหลืองแสดงว่าตรวจไม่พบสัญญาณ Sync ในการ์ด VGA จะมีหน่วยความจำซึ่งทำหน้าที่เก็บค่าของสีทุก pixel และเวลาการแสดงผล หน่วยความจำนี้เมื่อถูกอ่านออกไปแล้วจะทำหน้าที่เหมือนเป็นตัวผลิตสัญญาณ RGB แล้วส่งให้ลำอิเล็กตรอนยิงกวาดไปบนฉาก

2.7 คุณสมบัติของแสงและสี

แสงโดยทั่วไปแบ่งเป็น 2 ประเภท คือแสงที่ตามนุษย์มองเห็น กับแสงที่ตามนุษย์มองไม่เห็น

- แสงที่ตามนุษย์มองไม่เห็น ก็เช่นแสงจำพวกรังสีแกมมา (Gamma-ray), รังสีเอ็กซ์ (X-ray), แสงเหนือม่วง (Ultraviolet ray) และแสงใต้แดง (Infrared ray) แสงเหล่านี้เป็นส่วนหนึ่งของพลังแม่เหล็กไฟฟ้า แต่มีความยาวคลื่นแตกต่างกันออกไป
- แสงที่ตามนุษย์มองเห็น เป็นคลื่นแม่เหล็กไฟฟ้าที่มีช่วงความยาวคลื่นประมาณ 380 nm จนถึงประมาณ 780 nm ซึ่งสีที่มองเห็นนี้ ตาของมนุษย์จะรู้สึกได้ 2 ประการ โดยประการแรกคือความรู้สึกว่าแสงนี้เป็นสีอะไร (sensation of color) ส่วนประการที่ 2 คือแสงสีนี้มีความสว่างมากหรือน้อย (sensation of brightness)

2.8 ความสำคัญ 3 ประการของแสงที่ตามองเห็น

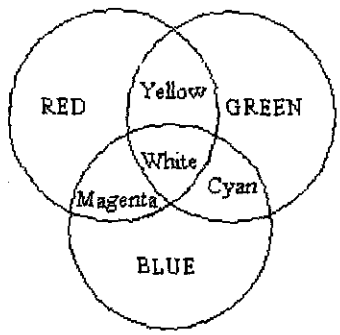
แสงที่ตามองเห็นทำให้รู้สึกที่สำคัญอยู่ 3 ประการคือ เกิดความรู้สึกในเรื่องสีของแสง (Hue), เกิดความรู้สึกในเรื่องการส่องสว่าง (Brightness), เกิดความรู้สึกในเรื่องแสงสีอิ่มตัว (Saturation)

- ความรู้สึกในเรื่องสีของแสง (Hue) จะทำให้สายตาสามารถแยกแยะออกได้ว่าแสงที่เห็นเป็นสีแดง, สีเขียว, สีน้ำเงิน, สีเหลือง เป็นต้น
- ความรู้สึกในเรื่องการส่องสว่าง (Brightness or Value) จะทำให้รู้สึกว่าแสงนี้สว่างหรือมืด
- ความรู้สึกในเรื่องแสงสีอิ่มตัว (Saturation or Chrome) จะทำให้รู้สึกถึงความบริสุทธิ์ของแสงสีได้ว่าเข้มหรือจาง

2.9 การผสมและการแยกแสงสี

การผสมแสงสีจะมีลักษณะแตกต่างจากการผสมแม่สีที่ใช้ในภาพวาด การผสมสีโดยทั่วไปแล้วในสีที่ใช้วาดเขียนนั้นเมื่อผสมแล้วจะทำให้สีที่ได้นั้นมีความเข้มข้นมากกว่าเดิม ซึ่งเป็นลักษณะการผสมแบบ Subtractive mixture ส่วนการผสมแสงสีนี้จะเป็นการผสมแม่สี (primary color) เพื่อทำให้เกิดสีต่างๆ ขึ้น ซึ่งแม่สีนี้จะต้องเป็นสีอิสระคือไม่สามารถนำสีอื่นมาผสมเป็นสีนั้นได้ แม่สีของแสงนั้นมีอยู่ด้วยกัน 3 สีคือ แดง, เขียว และน้ำเงิน นอกจากนั้นแสงสีอื่นที่มองเห็นนั้นได้มาจากการผสม ดังนี้

แสงสีเหลือง (Yellow)	ได้มาจาก แสงสีแดง + แสงสีเขียว
แสงสีฟ้าอมเขียว (Cyan)	ได้มาจาก แสงสีเขียว + แสงสีน้ำเงิน
แสงสีม่วง (Magenta)	ได้มาจาก แสงสีน้ำเงิน + แสงสีแดง
แสงสีขาว (White)	ได้มาจาก แสงสีแดง + แสงสีเขียว + แสงสีน้ำเงิน



รูปที่ 2.5 การผสมแม่สีของแสง

2.10 องค์ประกอบภาพ

หากเราตัดภาพจากหนังสือพิมพ์สักภาพหนึ่ง แล้วขยายขึ้นด้วยกล้องหรือแว่นขยาย จะพบว่าภาพมีองค์ประกอบมาจากจุดสีดำมากมายมาเรียงกันประกอบขึ้นเป็นภาพ จุดเหล่านี้เองที่เรียกว่าองค์ประกอบภาพหรือเรียกว่า Picture element หรือเรียกว่า พิกเซล (pixel)

ในทำนองเดียวกัน ภาพที่ปรากฏบนหางจอภาพก็ใช้หลักการเดียวกัน ภาพที่เกิดขึ้นบนจอภาพประกอบด้วยเส้นแนวนอนเส้นเล็กๆ เป็นจำนวนมาก แต่ละเส้นนั้นมีทั้งส่วนที่ดำสนิทและสว่างรวมกันอยู่ เส้นเหล่านี้เราได้มาจากการกวาดลำแสง ความแตกต่างกับการเส้นกวาดลำแสงหรือเส้นสแกนเหล่านี้เองเราก็จัดว่าเป็นองค์ประกอบภาพ

มาตรฐานระบบสแกน 525 เส้น แต่ละเส้นจะแบ่งได้เป็น 700 เส้น ดังนั้นจึงกล่าวได้ว่าหากจะหาองค์ประกอบภาพจะได้ $525 \times 800 = 420,000$ Pixels ใช้แบนวิดท์ 6 *MHz* ยิ่งภาพมีจำนวนเส้นสแกนมากเท่าไร รายละเอียดของภาพหรือความคมชัดก็จะมากขึ้นเท่านั้น แต่การออกแบบก็จะยากขึ้นด้วยเช่นกันเนื่องจากภาพที่มีรายละเอียดมากก็ต้องการแบนวิดท์มากขึ้นด้วย จากที่กล่าวมาว่าองค์ประกอบของภาพมีหลาย pixel แต่ในความเป็นจริงแล้วเราไม่สามารถมองเห็นได้ครบทุก pixel เนื่องจากจะมีบางส่วนหายไประหว่างการสแกนภาพ

ระบบสแกนที่ใช้ในประเทศไทยคือ 625 เส้น ซึ่งจะต้องใช้แบนวิดท์เท่ากับ 7 *MHz* ความละเอียดก็จะมากขึ้นกว่าระบบสแกน 525 เส้นด้วย องค์ประกอบภาพที่ได้จะเท่ากับ $625 \times 851 = 531,875$ pixels

ปัจจุบันสำหรับโทรทัศน์ธรรมดาเราพบว่ามีการเพิ่มเส้นภาพให้มากขึ้น และแน่นอนจำนวนพิกเซลก็มากขึ้นด้วย อย่างระบบโทรทัศน์แบบรายละเอียดสูงหรือ HDTV จะมีจำนวนเส้นสแกนเท่ากับ 725 เส้น ส่วนในโปรเจคเตอร์จะมีจำนวนเส้นสแกน 2200 เส้น

2.11 วงจรรวม (ASIC: Application Specific Integrated Circuit)

ความก้าวหน้าของอุตสาหกรรมอิเล็กทรอนิกส์ปัจจุบันทำให้เกิดการพัฒนาความสามารถของอุปกรณ์ต่างๆ ซึ่งสามารถลดค่าใช้จ่าย ลดการสิ้นเปลืองขนาดพื้นที่ รวมทั้งพลังงานที่ใช้กับอุปกรณ์ และในขณะเดียวกันยังมีการเพิ่มประสิทธิภาพและระดับความน่าเชื่อถือของวงจรที่สูงขึ้นอย่างเห็นได้ชัดจากเทคโนโลยีไมโครโปรเซสเซอร์และหน่วยความจำปัจจุบัน ในการพัฒนาเพิ่มความหนาแน่นและจำนวน

ฟังก์ชั่นที่เหมาะสม นักออกแบบอุปกรณ์ทางด้านดิจิตอลได้พิจารณาถึงการผลิตในจำนวนมากๆ และการผลิตวงจรรวม (ASIC: Application Specific Integrated Circuit) ซึ่งวงจรรวมจะสามารถแบ่งได้เป็น 2 กลุ่ม คือ Field Programmable และ Mask Programmable

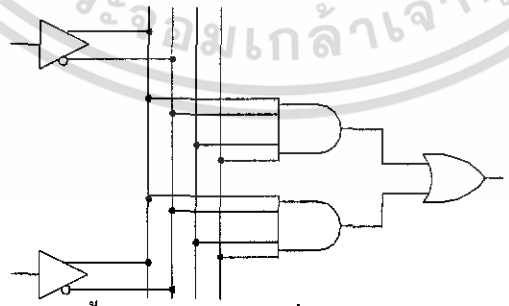
2.11.1. Field programmable

เป็นวงจรรวมเฉพาะงาน ซึ่ง ASIC แบบ field programmable มีลักษณะการสร้างหรือการกำหนดการทำงานต่างกัน กล่าวคือ ผู้ใช้สามารถออกแบบและสร้างวงจรที่ต้องการใช้ลงในอุปกรณ์ได้เองโดยไม่ต้องไปโรงงานผลิต โดยเฉพาะอย่างยิ่งในปัจจุบันนี้มีเครื่องมือช่วยในการออกแบบ และสร้างวงจรร่วนกับไมโครคอมพิวเตอร์ที่มีความสามารถสูงในการพัฒนาตั้งแต่ขั้นการออกแบบ การจำลองการทำ

งานจนถึงจัดการสร้างวงจรลงในอุปกรณ์รวมทั้งอุปกรณ์ field programmable เหล่านี้สามารถหาซื้อได้ง่ายทำให้การสร้างวงจรอิเล็กทรอนิกส์จนถึงระบบไมโครโพรเซสเซอร์หันมาใช้อุปกรณ์จำพวกนี้ เป็นอุปกรณ์ประกอบในวงจรแทนอุปกรณ์ย่อยๆ แยกชิ้น (Discrete component)

- พีแอลดี (PLD: Programmable Logic Device)

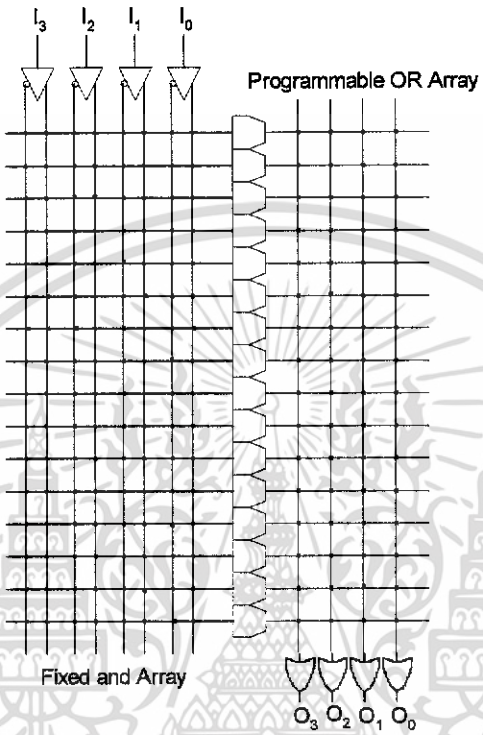
ภายในพีแอลดีจะถูกเตรียมไว้เป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่ม มีทั้งวงจรคอมบิเนชั่น (Combination) และวงจรซีเควนเชียล (Sequential) ซึ่งเทคโนโลยีของวงจรที่ใช้สร้างพีแอลดีมีทั้งทีทีแอล (TTL) อีซีแอล (ECL) และซีมอส (CMOS) ให้เลือกใช้ตามความเหมาะสมของแต่ละระบบ พีแอลดีทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมีวงจรหลักเป็นวงจรคอมบิเนชั่นที่ให้ผลเป็นผลคูณร่วมบวก (Sum of product) ประกอบด้วยชุดของแอนด์เกทที่ต่อร่วมกับออร์เกท การโปรแกรมคือการเลือกว่าจะให้มีการต่ออินพุตภายในของแอนด์เกทกับอินพุตใดบ้าง ซึ่งมีทั้งสัญญาณจากภายนอกหรือจะเป็นสัญญาณป้อนกลับจากเอาท์พุตภายในก็ได้ หรือจะต่ออินพุตของออร์เกทกับเอาท์พุตของแอนด์เกทตัวอื่นๆ เป็นต้น การเลือกโปรแกรมอินพุตต่างๆ ของตัวอุปกรณ์จะถูกต่อผ่านฟิวส์เข้ากับแหล่งกำเนิดสัญญาณ ถ้าไม่ต้องการใช้สัญญาณใดก็จะจะตัดฟิวส์นั้นให้ใช้ได้แค่ครั้งเดียว พีแอลดีบางชนิดจะใช้มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมได้ซ้ำโดยใช้กระแสไฟฟ้า และยังสามารถลบแล้วโปรแกรมใหม่ได้



รูปที่ 2.6 วงจรพื้นฐานของพีแอลดีซึ่งต่อยู่ในรูปผลคูณร่วมบวก

- พรอม (PROM: Programmable Read Only Memory)

พรอมก็คือหน่วยความจำประเภทรอม (ROM) ที่สามารถโปรแกรมได้ซึ่งนับว่าเป็นพีแอลดีชนิดหนึ่ง วงจรภายในพรอมเสมือนประกอบด้วยแถวลำดับของแอนด์เกทและออร์เกท (And – or Gate Array) ผลเอาท์พุตสามารถแสดงให้อยู่ในรูปของฟังก์ชั่นผลคูณร่วมบวก (Sum of product) ของสัญญาณอินพุตที่ขาแอดเดรส ดังรูปที่ 2.7



รูปที่ 2.7 ลักษณะวงจรภายในของพรอมขนาด 16×4 บิต

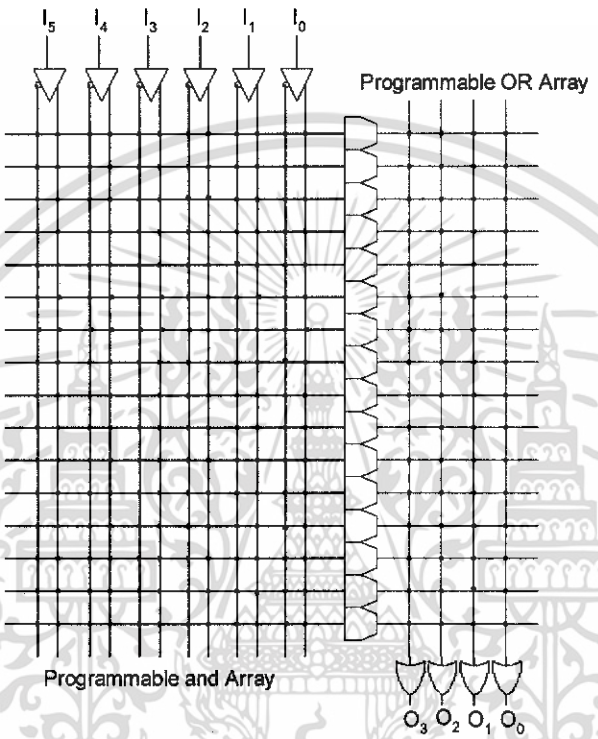
จากรูปที่ 2.7 วงจรทางด้านซ้ายเป็นแอนด์เกทที่ให้ผลเป็นผลคูณ (Product) ของกรณีอินพุตเป็น 0000 แอนด์เกทที่อยู่ถัดลงมาเป็นผลคูณของกรณีที่อินพุตเป็น 0001 และเมื่อไล่ลำดับไปเรื่อยๆ จนถึงเกทตัวล่างสุดก็จะเป็นผลคูณของกรณีอินพุตเป็น 1111 ที่อินพุตแต่ละบิตของหน่วยความจำสามารถเลือกได้ว่าจะให้เป็น 1 ในกรณีที่อินพุตจากแอดเดรสอะไรบ้างที่เหมือนกัน ซึ่งเป็นการนำเอาท์พุตจากผลคูณที่ต้องการให้เอาท์พุตแต่ละบิตเป็น 1 ไปออร์กันจึงเปรียบเสมือนว่าในพรอมมีจำนวนแอนด์เกทเท่ากับจำนวนตำแหน่งของหน่วยความจำและมีจำนวนออร์เกทเท่ากับจำนวนบิตของสัญญาณข้อมูลเอาท์พุต ส่วนอินพุตของออร์เกทสามารถที่จะต่อกับแอนด์เกทตัวใดก็ได้ทุกตัว ซึ่งอาจเรียกเป็น พีแอลดีแบบ fixed AND/programmable OR

- พีเอแอล (PAL: Programmable Array Logic)

เป็นพีแอลดีชนิดใหม่โดยใช้เทคโนโลยีแบบแอลเอสไอ (LSI: Large Scale Integration) ซึ่งสามารถโปรแกรมเลือกวงจรภายใน โดยใช้ฟิวส์ที่เชื่อมต่ออยู่ระหว่างอินพุตภายนอกและจากการป้อนกลับภายในกับแอนด์เกทที่ต่อเป็นฟังก์ชั่นคูณ อยู่ในตัววงจรรวม

- พีแอลเอ (PLA: Programmable Logic Array)

ซึ่งสามารถโปรแกรมการต่อลอจิกทั้งทางด้านแอนด์เกทและออร์เกทได้และยังเลือกเอาท์พุตว่าจะให้ทำงานที่ระดับแรงดันสูงหรือต่ำได้ โดยต่อผ่านเอ็กซ์คูซีฟออร์เกท (X- OR Gate) ให้ทำหน้าที่เป็นนอนอินเวิร์ดเตอร์หรือเป็นอินเวิร์ดเตอร์ ซึ่งต่อมาได้มีการพัฒนาเป็นเอฟพีแอลเอ ที่มีรีจิสเตอร์อยู่ภายในวงจร รวมทั้งสามารถเลือกสัญญาณอินพุตที่มาจากการป้อนกลับจากรีจิสเตอร์ได้ด้วย ทำให้สามารถใช้การจำลองการทำงานของวงจรที่ออกแบบขึ้นมาได้ต่อมาได้เปลี่ยนชื่อใหม่เป็นเอฟพีแอลเอส (FPLS: Field Programmable Logic Sequencer) ซึ่งมีทั้งแบบทีทีแอลและแบบซีมอส



รูปที่ 2.8 วงจรพื้นฐานภายในของพีแอลเอ

- แอลซีเอ (LCA: Logic Cell Array)

อุปกรณ์นี้ถูกสร้างเป็นอาร์เรย์ที่ประกอบด้วยเกทจำนวน 1,200-1,800 เกท มีลักษณะสถาปัตยกรรมที่ใกล้เคียงเกทอาร์เรย์ (Gate Array) โดยมีการประมวลผลแบบซีมอสโลหะคู่ 1.6 ไมครอน ซึ่งสามารถโปรแกรมและลบได้โดยใช้กระแสไฟฟ้า (Static RAM based) ภายในเรียงเป็นเมตริกซ์ของลอจิกเซลล์ล้อมรอบภายนอกด้วยอินพุตและเอาท์พุตเซลล์ แอลซีเอนี้เองเป็นพื้นฐานของการพัฒนามาเป็นเอฟพีจีเอ (FPGA: Field Programmable Gate Array) ซึ่งจะมีประสิทธิภาพสูง

- อีพีแอลดี(EPLD: Erasable Programmable Logic Device)

เป็นอุปกรณ์ที่สามารถลบและโปรแกรมซ้ำได้เพื่อใช้ในการสร้างวงจรต้นแบบ พีแอลดีจะมีการทำงานเหมือนกับอีพรอมแบบซีมอส (CMOS EPROM) คือใช้มอสทรานซิสเตอร์เชื่อมต่อระหว่างสัญญาณอินพุตกับจุดที่ต้องการ แทนการใช้ฟิวส์ ทำให้สามารถโปรแกรมวงจรด้วยการจ่ายกระแสไฟไปยังตัวอุปกรณ์ และยังสามารถลบได้โดยใช้แสงอัลตร้าไวโอเลตฉายผ่านช่องกระจกบนตัวชิพ

2.11.2. Mask programmable

เป็นวงจรรวมที่ใช้ในเชิงพาณิชย์ ถ้าเราผลิตวงจรรวมที่มีคุณสมบัติเหมือนกับตอนที่ใช้ Field programmable ออกแบบ แต่เปลี่ยนไปใช้ Mask programmable จะช่วยในการลดต้นทุนในการผลิตได้อย่างมาก วงจรรวมประเภทนี้ หลังจากผู้ออกแบบได้ออกแบบและทดสอบการทำงานวงจรเสร็จเรียบร้อยแล้ว จะต้องส่งให้ทางผู้ผลิตทำการเจือสาร ซึ่งผู้ออกแบบจะไม่สามารถโปรแกรมได้เองเหมือน Field programmable วงจรรวมแบบ Mask programmable ในปัจจุบันได้แก่ เกทอาร์เรย์, เซลล์มาตรฐาน และฟูลคัสตัม (Full Custom)

- เกทอาร์เรย์(Gate Array)

วงจรประเภทนี้ประกอบด้วยแถวลำดับของวงจรเกท ซึ่งจะเป็นประเภทเดียวกันหรือไม่ก็ได้ ต่อกับขั้วสายไฟ (Pad) สำหรับต่อกับวงจรภายนอก ผู้ใช้มีหน้าที่ออกแบบการเชื่อมต่อทางไฟฟ้าระหว่างวงจรเกทแต่ละตัวและขั้วสายไฟเพื่อให้เกททำงานตามที่ต้องการ แล้วจึงส่งผลนี้ออกแบบไปยังโรงงานผู้ผลิตวงจรรวม หลังจากนั้นนำไปเจือสาร โดยทั่วไปการสร้างวงจรรวมจะใช้พื้นที่แค่ 25-30% ของพื้นที่ซิลิกอนสำหรับวงจรเกท ส่วนที่เหลือจะเป็นพื้นที่ซิลิกอนสำหรับวงจรพวกหน่วยความจำ

- เซลล์มาตรฐาน (Standard Cell)

เนื่องจากปัญหาการเชื่อมต่อวงจรรวม เกทอาร์เรย์จึงมีขีดจำกัดในด้านความซับซ้อนของวงจรที่ใช้ ประกอบผู้ใช้จำนวนมากต้องการจะรวมอุปกรณ์ให้อยู่ในวงจรรวมตัวเดียว ทำให้มีการสร้างเซลล์มาตรฐานขึ้นมา ซึ่งผู้ใช้สามารถเลือกกลุ่มวงจรทำหน้าที่ต่างๆได้ เช่น เกท, ฟลิปฟลอป, หน่วยความจำ เป็นต้น ซึ่งเหมาะกับวงจรที่มีความซับซ้อน แต่การผลิตวงจรรวมแบบเซลล์มาตรฐานจะมีราคาสูงกว่าแบบเกทอาร์เรย์และใช้เวลาออกแบบและเจือสารนานกว่า เซลล์มาตรฐานจึงเหมาะกับการผลิตในเชิงพาณิชย์

- ฟูลคัสตัม (Full Custom)

วงจรฟูลคัสตัมนี้ผู้ใช้จะออกแบบวงจรเองทั้งหมด แล้วส่งข้อมูลการออกแบบไปให้ผู้ผลิตเจือสารในรูปแฟ้มข้อมูลมาตรฐาน การใช้ฟูลคัสตัมเป็นที่แพร่หลายในปัจจุบัน เพราะค่าจ่ายใช้ที่ประหยัดกว่า 2 แบบแรก ใช้ในการสร้างวงจรเพื่อการศึกษาหรือวิจัย หรือจะผลิตเป็นจำนวนน้อยก็ได้ รูปแบบการเจือสารของวงจรรวมฟูลคัสตัมประกอบด้วยการออกแบบหลายวงจรรวมอยู่บนแผ่นซิลิกอนแผ่นเดียวกัน มีอยู่ 3 ประเภท ดังนี้

- เอ็มพีดับเบิ้ลยู (MPW: Multi project wafer) แผ่นเวเฟอร์นี้จะถูแบ่งเป็นส่วนๆ ที่เรียกว่า ไดซ์ (Dice) และไดซ์แต่ละชุดจะเจือสารวงจรต่างกัน
- เอ็มพีซี (MPC: Multi project chip) เป็นการเจือสารหลายๆ วงจรลงในไดซ์ชุดเดียวกันบนที่ใดที่หนึ่งบนเวเฟอร์แผ่นนั้นๆ การเจือแบบนี้จะเน้นการใช้พื้นที่ให้เกิดประโยชน์สูงสุดเป็นสำคัญ
- เอ็มพีอาร์ (MPR: Multi project reticle) เป็นการเจือสารที่ผสมผสานระหว่างฟูลคัสตัมกับเซลล์มาตรฐาน กล่าวคือจะมีการแบ่งออกเป็นส่วนๆ อย่างสม่ำเสมอ โดยแต่ละส่วนบรรจุการออกแบบแต่ละวงจร ไดซ์ชุดหนึ่งจะสามารถบรรจุวงจรที่ออกแบบได้ประมาณ 8 วงจรและไดซ์ทุกชุดในแผ่นเวเฟอร์จะมีลักษณะเหมือนกัน

2.12 FPGA (field Programmable Gate Array)

เป็นอุปกรณ์ที่ถูกสร้างเป็นอาร์เรย์ที่ประกอบด้วยเกทจำนวนมาก ซึ่งสามารถจะกำหนดฟังก์ชั่นการทำงานได้ตามความต้องการของผู้ใช้โดยผ่านการโปรแกรมของเอฟพีจีเอ ได้แก่ การออกแบบการผลิต, ลดเวลาการส่งผลิตภัณฑ์ออกตลาดซึ่งเป็นประโยชน์ต่อการผลิตวงจรเป็นอย่างมาก นักออกแบบเพียงแต่กำหนดฟังก์ชั่นการทำงานของวงจร ดังนั้นการออกแบบโดยใช้เอฟพีจีเอสามารถออกแบบ-แก้ไขวงจรและใช้เวลาทดสอบวงจรได้เพียงไม่กี่วันเท่านั้น จากประโยชน์ของเอฟพีจีเอนี้เองทำให้ประหยัดค่าใช้จ่ายเป็นอย่างมาก

2.12.1 เทคโนโลยีของ FPGA

เนื่องจากเป็นลักษณะของชิปที่สามารถโปรแกรมได้นั่นก็คือสามารถกำหนดจุดเชื่อมต่อต่างๆ ได้ เพื่อประกอบเป็นวงจรที่เราต้องการ ซึ่งแบ่งลักษณะของจุดเชื่อมต่อต่างๆ ได้ดังนี้

1. Physical Changing

1.1 Fused คือจะสามารถโปรแกรมได้เพียงครั้งเดียว หลังจากโปรแกรมจุดเชื่อมต่อขาดจากกัน

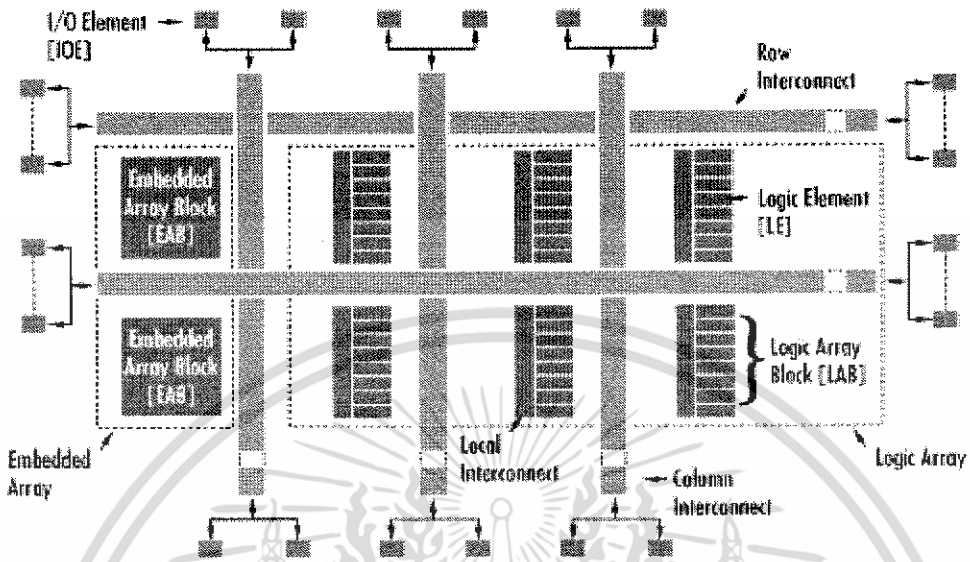
1.2 Anti Fused สามารถโปรแกรมได้เพียงครั้งเดียว หลังจากโปรแกรมจุดเชื่อมต่อเชื่อมกัน

2. Memory Base

2.1 EEPROM – Base FPGA มักจะเรียก FPGA ประเภทนี้จะมีความจุของเกทต่ำ โดยทั่วไปจะมีน้อยกว่า 20,000 เกท แต่ข้อดีคือสามารถเก็บข้อมูลที่โปแกรมลงไปได้โดยจำเป็นต้องมีไฟเลี้ยง และสามารถโปรแกรมได้ประมาณ 10,000 ครั้งมักจะมีการจัดสถาปัตยกรรมรูปแบบอาร์เรย์ใช้ AND – OR Plane ในการทำลอจิกฟังก์ชั่น

2.2 SRAM – Base FPGA ซึ่งสามารถโปรแกรมซ้ำได้ไม่จำกัดครั้ง มีความจุเกทปานกลางถึงสูงมาก (ประมาณ 10,000 จนถึง 1,000,000 เกท) จะใช้ Look-Up Table ในการทำลอจิกฟังก์ชั่น และมีการจัดทรัพยากรภายในโครงสร้างแบบอาร์เรย์ ข้อดีคือจะใช้เวลาในการโปรแกรมน้อย (ประมาณ ms) การโปรแกรมจะทำได้ไม่จำกัดจำนวนครั้ง, ง่ายสะดวก เมื่อเทียบกับการเขียน SRAM แบบทั่วไป รวมทั้งเหมาะสมกับการออกแบบวงจรที่ซับซ้อน และยังช่วยให้กระบวนการผลิตง่ายขึ้นอีกด้วย แต่ข้อเสียก็คือจะไม่สามารถเก็บโปรแกรมในสภาวะที่ไม่มีไฟเลี้ยงได้ FPGA ชนิดนี้จึงนิยมใช้คู่กับ ROM เพื่อเก็บโปรแกรมและโหลดโปรแกรมเข้าในชิปได้ทันทีเมื่อเริ่มมีการใช้งาน

อุปกรณ์ FPGA นั้นจะมีลักษณะของโครงสร้างภายในเป็นอาร์เรย์ลอจิกดังรูปที่ 2.9 ซึ่งสามารถทำการโปรแกรมได้หลายครั้ง



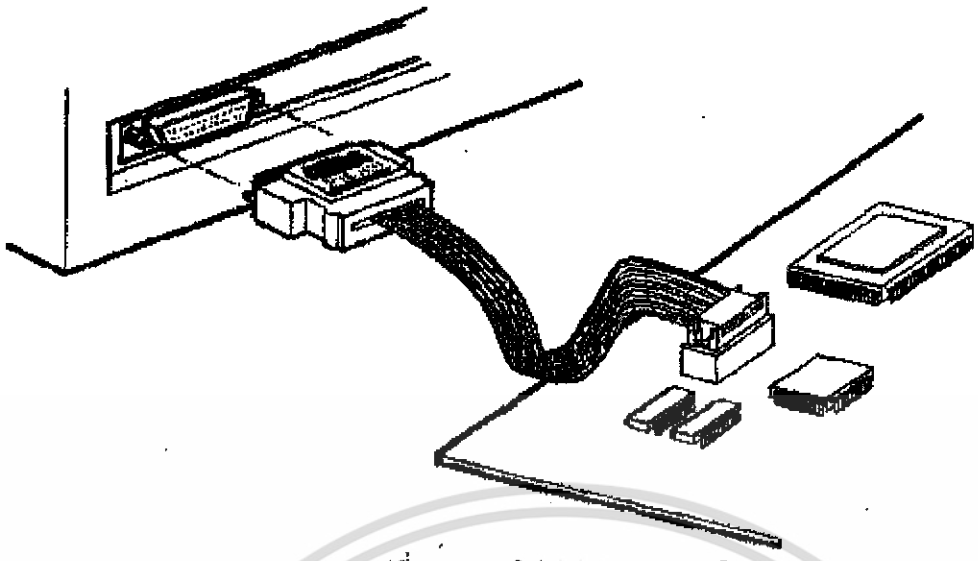
รูปที่ 2.9 โครงสร้างภายในของ FPGA ตระกูล FLEK10K

2.12.2 ทำไม FPGA ถึงได้ออกแบบได้ง่ายและสะดวกรวดเร็ว

1. ในการออกแบบ ผู้ใช้ไม่จำเป็นต้องรู้ถึงโครงสร้างภายในตัวชิป เพียงแค่รู้ขั้นตอนการออกแบบวงจรลอจิกก็พอ ซึ่งไม่เหมือนไมโครโปรเซสเซอร์ตัวอื่นที่เราต้องรู้โครงสร้างภายในของตัวชิป

2. การใช้โปรแกรมที่อธิบายการทำงานของวงจรที่เรียกว่า HDL (Hardware Description Language) จะช่วยให้ออกแบบวงจรได้ง่ายขึ้น เนื่องจากเป็นวิธีการที่มีความยืดหยุ่นสูงและทำได้รวดเร็ว นอกจากนี้เรายังไม่จำเป็นต้องรู้ว่าวงจรต่อกันอย่างไร แค่รู้กระบวนการในการทำงานแล้วโปรแกรมจะทำการสังเคราะห์และทำงานให้ตามที่ผู้ใช้ต้องการภาษาที่ใช้นี้ยังเป็นมาตรฐานเดียวกันจึงสามารถใช้ได้กับชิปของทุกบริษัทและทุกรุ่น

3. การโปรแกรมสามารถทำได้เองและยังใช้เวลาไม่นาน โดยเพียงแค่ส่งข้อมูลผ่านสายดาวน์โหลดทางพอร์ตขนานของคอมพิวเตอร์ก็สามารถโปรแกรมชิปลงตัวชิปได้ขณะที่อยู่ในระบบ โดยไม่จำเป็นต้องถอดออกมาโปแกรมข้างนอก ดังรูปที่ 2.10 และที่สำคัญคือสามารถโปรแกรมได้หลายครั้งทำให้แก้ไขพัฒนาได้ง่ายและรวดเร็ว



รูปที่ 2.10 การโปรแกรมลงบนตัวชิพ

2.12.3 การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้โปรแกรมในการสังเคราะห์วงจรทำการสังเคราะห์ขั้นตอนการทำงานของวงจรที่ได้จากการออกแบบด้วยภาพการทำงานของวงจร (Schematic) ซึ่งต้องทำหารตรวจสอบว่าสนับสนุนเทคโนโลยีของ FPGA (FPGA Library) ที่ต้องการหรือไม่ เช่น โปรแกรม MAX Plus II ซึ่งในขั้นตอนนี้โปรแกรมจะสังเคราะห์วงจร โดยทำการแปลงโค้ดภาษา VHDL และสร้างวงจรขึ้นมาตามโค้ดที่ผู้ออกแบบเขียนขึ้นมา นอกจากนี้ผู้ออกแบบยังสามารถกำหนดเงื่อนไขต่างๆ ได้ เช่นข้อบังคับเรื่องเวลา (Timing Constraints) หรือข้อบังคับในเรื่องขนาดพื้นที่ (Area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ขั้นตอนที่เรียกว่า Optimize เพื่อให้วงจรทำงานตามที่ผู้ออกแบบต้องการ ส่วนสำคัญของการ Optimize คือการเทียบ (Mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะกับโครงสร้างภายในอุปกรณ์ FPGA เมื่อทำการสังเคราะห์วงจรแล้วจะมีการรายงานผลว่าโมเดลนั้นมีการทำงานเป็นอย่างไร เช่น เวลาที่ใช้หน่วง (Delay) ใช้ทรัพยากรอะไรบ้างใน FPGA เป็นต้น

2.13 ภาษาวีเอชดีแอล (VHDL: VHSIC Hardware Description Language โดยที่ VHSIC: Very High Speed Integrated Circuit)

เป็นภาษาบรรยายอุปกรณ์ฮาร์ดแวร์(HDL: Hardware Description Language) ที่ได้รับการพัฒนาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ สำหรับมาตรฐานของภาษาที่ใช้บรรยายพฤติกรรมของวงจรหรือฮาร์ดแวร์ได้ให้ไว้ว่าสามารถสรุปได้ดั่งนี้ คือ

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถเข้าใจได้ทั้งมนุษย์และคอมพิวเตอร์โดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จัดเป็นภาษาโปรแกรมระดับสูงเช่นเดียวกับภาษาปาสคาล หรือภาษาซี

2.13.1 ข้อกำหนดสำหรับภาษาVHDL ในเดือนมกราคมปี ค.ศ. 1983 ไว้ดังนี้

1. ลักษณะทั่วไป

กำหนดให้ VHDL เป็นภาษาสำหรับการออกแบบและ บรรยายของ ฮาร์ดแวร์ซึ่งหมายถึงความสามารถในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบนซึ่งก็คือระบบจนถึงระดับเกทอีกด้วย เนื่องจากการทำงานในระบบดิจิตอลนั้นทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่จะทำงานไปพร้อมๆกันซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ก็ถือเป็นขอกำหนดที่สำคัญอย่างหนึ่งของ VHDL ด้วยเช่นกัน (สำหรับภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้นความพร้อมเพรียงจะหมายถึงทุกๆ คำสั่ง องค์ประกอบ เกทหรือวงจรลอจิกจะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไปพร้อมๆ กัน)

2.สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้น เป็นลักษณะที่สำคัญอย่างหนึ่งสำหรับการออกแบบระบบที่มีหลายๆ ระดับโดยในการออกแบบจะประกอบด้วย ส่วนการบรรยายการเชื่อมต่อ และส่วนการบรรยาย7หน้าที่การทำงานซึ่งหน้าที่การทำงานของระบบสามารถกำหนดได้ด้วยตัวเอง หรืออาจถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลงไปได้เช่นกัน แต่ที่ระดับล่างสุดองประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวของมันเอง และไม่สามารถกำหนดการทำงานโดยลักษณะแบบโครงสร้างได้

3.ไลบรารี

VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของอุปกรณ์ไว้ในระบบไลบรารี หรือจะใช้ไลบรารีกับระบบที่ได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูกต้องควรจัดเก็บไว้ในไลบรารีหลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้ว เพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไปใช้ได้ด้วย

4.ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการ โดยพร้อมเพียงกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษาเองก็มีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วยเมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้วผู้ออกแบบสามารถบรรยายหน้าที่การทำงาน ซึ่งเป็นรายละเอียดภายในของแต่ละองค์ประกอบได้ในลักษณะเดียวกันกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case, if-then-else และ loop ทั่วๆไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น อย่างก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังเป็นการทำงานแบบพร้อมเพรียงกันเช่นเดิม

5.การกำหนดคุณสมบัติ

นอกจากการกำหนดอินพุทและเอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกันโดยสิ่งนี้รวมถึงสภาพแวดล้อมละลักษณะทางกายภาพของอุปกรณ์นั้นๆด้วยซึ่งภาษาสำหรับการออกแบบที่ดีควรให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนด

ขนาด ลักษณะทางกายภาพ เวลา โหลด และเงื่อนไขทางสภาพแวดล้อมอื่นๆ ซึ่งก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

6.ชนิดของข้อมูล

สามารถกำหนดชนิดของข้อมูลเป็น Bit, Boolean, จำนวนเต็ม, จำนวนจริง, จุดทศนิยม, ลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของข้อมูลที่ผู้ออกแบบกำหนดเองก็ได้

7.โปรแกรมย่อย

ความสามารถในการใช้ฟังชั่นและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL ซึ่งผู้ออกแบบสามารถนำโปรกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำต่างๆ หรือหน้าที่อื่นๆตามที่ต้องการได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

8.การควบคุมเวลา

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการการตรวจสอบ การออกแบบเกท หรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอย เหตุการณ์ (Event) นอกจากนี้ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

9.การกำหนดแบบโครงสร้าง

การกำหนดแบบโครงสร้างขององค์ประกอบต่างๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยกำหนดโครงสร้างขององค์ประกอบร่วมที่เกิดจากองค์ประกอบย่อย ซึ่งตกต่างกัน หรือเหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของภาษา VHDL เช่นกัน

2.13.2 องค์ประกอบพื้นฐานของ VHDL

ในการเขียนรูปแบบบรรยายระบบดิจิตอลในมุมมองของการออกแบบลักษณะบนลงล่างจะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆของภาษา VHDL เสียก่อนซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วยคือ

- หน่วยการออกแบบเอนทิตี (Entity Design Unit)
- หน่วยการออกแบบสถาปัตยกรรม (Architecture Design Unit)
- หน่วยการออกแบบเพ็กเกต (Packages Design Unit)
- หน่วยการออกแบบโครงแบบ (Configuration Design Unit)

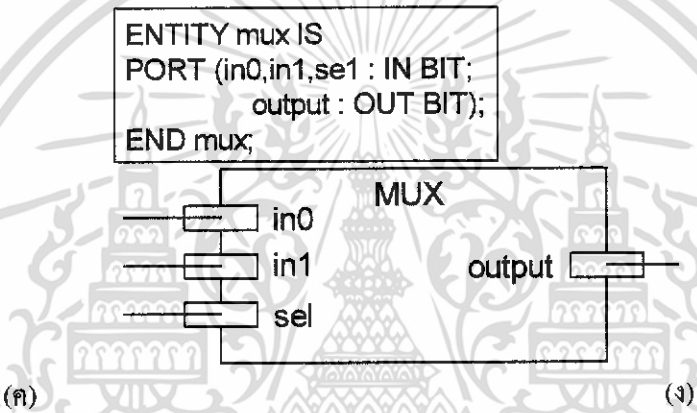
1. หน่วยการออกแบบเอนทิตี

หน่อยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างโลกภายนอกกับรูปแบบที่เขียนขึ้น ที่เรียกว่าหน่วยการออกแบบเอนทิตี ในส่วนนี้ใช้กำหนดจุดเชื่อมต่อของรูปแบบกำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสันญาณตามจุดต่างๆของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น รูปที่ 2.11 แสดงให้เห็นโครงสร้างอย่างง่ายๆของหน่วยการออกแบบเอนทิตี

```
ENTITY component_name IS
        Input and output ports Physical and
other parameters
END [component_name];
```

รูปที่ 2.11 แสดงโครงสร้างโดยทั่วไปของหน่อยการออกแบบเอนทิตี

ส่วนนี้จะขึ้นต้นด้วยคำ Entity และ IS ระหว่างคำทั้งสองเป็นส่วนสำหรับชื่อละรูปแบบที่ต้องการจะเขียน (component_name) สำหรับการตั้งชื่อนั้นต้องเป็นไปตามกฎเกณฑ์ของภาษาหลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆส่วนนี้เรียกว่าส่วนหัว (entity header) ละที่สำคัญคือหน่วยการออกแบบเอนทิตีจะต้องปิดท้ายด้วยคำว่า END และเครื่องหมายอัฒภาคเสมอ (;)



รูปที่ 2.12 แสดงรูปแบบของการมัลติเพล็กซ์

(ค) หน่วยการออกแบบเอนทิตีในรูปของ VHDL

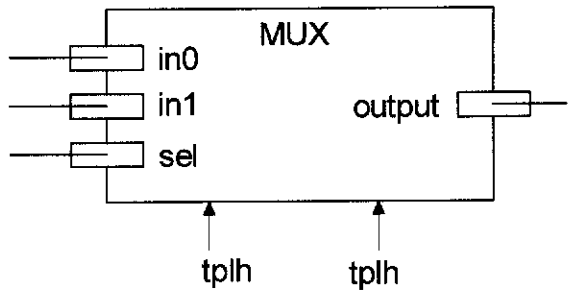
(ง) มุมมองของตัวเชื่อมประสาน (Interfacing)

ในรูปที่ 2.12 เป็นหน่วยการออกแบบเอนทิตีที่บรรยายอุปกรณ์ที่มีชื่อว่า มัลติเพล็กซ์ หรือ MUX ในส่วนหัวของ entity มีการกำหนดจัดต่อ 4 จุดภายใต้ชุดคำสั่ง PORT โดยที่ 3 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า ได้แก่ in0, in1, sel ซึ่งกำหนดทิศทางการติดต่อกับภายนอกเป็นการไหลเข้าของข้อมูล (IN) ที่แสดงด้วยรูปสี่เหลี่ยมโปร่งในรูปที่ 2.12 ส่วนจุดเอาท์พุทเป็นจุดให้ข้อมูลไหลออกซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก (OUT) ที่แสดงด้วยรูปสี่เหลี่ยมทึบในรูปที่ 2.12 ส่วนประกอบประเภทของข้อมูลที่จะไหลเข้าและออกนั้นเป็นประเภท BIT ที่สามารถมีค่าได้เพียงสองค่าคือ “0”และ“1”เท่านั้น

นอกจากนั้นผู้ออกแบบยังสามารถกำหนดค่าพารามิเตอร์ทางฟิสิกส์ที่เป็นข้อมูลเพิ่มเติมอื่นๆ ลงในส่วนหัวของเอนทิตีได้อีก เช่น ข้อมูลเกี่ยวกับความเร็วในการทำงานของอุปกรณ์อันได้แก่ ค่าเวลาหน่วงแพร่กระจาย (Propagation delay time) พารามิเตอร์เหล่านี้เรียกว่า เจนเนอริก (Generic) ที่กำหนดว่าคำสั่ง GENERIC จากตัวอย่างในรูปที่ 2.13

```
ENTITY mux IS
GENERIC (tplh : TIME := 3 ns;
         tplh : TIME := 5 ns);
PORT (in0,in1,se1 : IN BIT;
      output : OUT BIT);
END mux;
```

(จ)



(ฉ)

รูปที่ 2.13 รูปแบบมัลติเพล็กซ์ที่ประกอบด้วยข้อมูลค่าเวลาหน่วงแพร่กระจาย

(จ) หน่วยการออกแบบเอนทิตีในรูปของ VHDL

(ฉ) มุมมองของตัวเชื่อมประสาน

ในบางกรณีสามารถที่จะใช้ภาษา VHDL สร้างรูปแบบที่ปราศจากช่องทางไหลเข้าและออกของข้อมูลได้ ซึ่งส่วนใหญ่จะพบในการสร้างรูปแบบสำหรับตรวจสอบการทำงานของอีกรูปแบบหนึ่ง คือ VHDL สำหรับทดสอบเปรียบเทียบ (Test bench)

```
ENTITY test_bench
IS
END test_bench;
```

รูปที่ 2.14 หน่วยการออกแบบเอนทิตีที่ไม่มีกำหนดช่องทางต่อกับภายนอก

2. หน่วยการออกแบบสถาปัตยกรรม

คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบในมุมมองของการจำลองการทำงานพฤติกรรมต่างๆ ที่กำหนดในหน่วยการออกแบบเอนทิตี รูปที่ 2.15 แสดงให้เห็นถึงโครงสร้างอย่างง่ายๆ ของหน่วยการออกแบบสถาปัตยกรรม

```
ARCHITECTURE identifier OF
component_name IS
[declaration]
BEGIN
specification of the functionality of the
component in terms of its input lines and as
influenced bu by physical and other parameters
END [identifier];
parameters
END [identifier];
```

รูปที่ 2.15 หน่วยการออกแบบเอนทิตีที่ไม่มีการกำหนดช่องทางที่ต่อกับภายนอก

ส่วนของหน่วยการออกแบบสถาปัตยกรรมเริ่มต้นด้วยคำ ARCHITECTURE และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า ARCHITECTURE นั้นใช้บรรยายหน่วยการออกแบบเอนทิตีใดๆ (OT<entity design unit>IS) ส่วนที่อยู่ระหว่าง ARCHITECTURE และ BEGIN เป็นพื้นที่ส่วนประกาศหน่วยการออกแบบสถาปัตยกรรมกำหนด (architecture

declarative area) ที่เป็นเพียงส่วนเผื่อเลือก (option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้อาทิเช่นประเภท(type) ต่างๆ ตัวอย่างเช่น bit, bit_vector, สัญญาณ (signal), ตัวคงที่ (constant) โปรแกรมย่อย ได้แก่ function และ procedure และอุปกรณ์(component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า และไหลออกของรูปแบบ(สัญญาณที่กำหนดในชุดคำสั่ง PORT) นั้นจะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า BEGIN กับ END ของหน่วยการออกแบบสถาปัตยกรรมและนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (concurrent statement) เท่านั้น หน่วยการออกแบบสถาปัตยกรรมจะต้องปิดท้ายด้วยคำสั่ง END และชื่อของสถาปัตยกรรมนั้นๆ เป็นส่วนเผื่อเลือกโดยทั่วไปการเขียนรูปแบบระบบดิจิตอลด้วยภาษาวีเอชดีแอลสามารถเขียนได้ในลักษณะต่างๆดังนี้

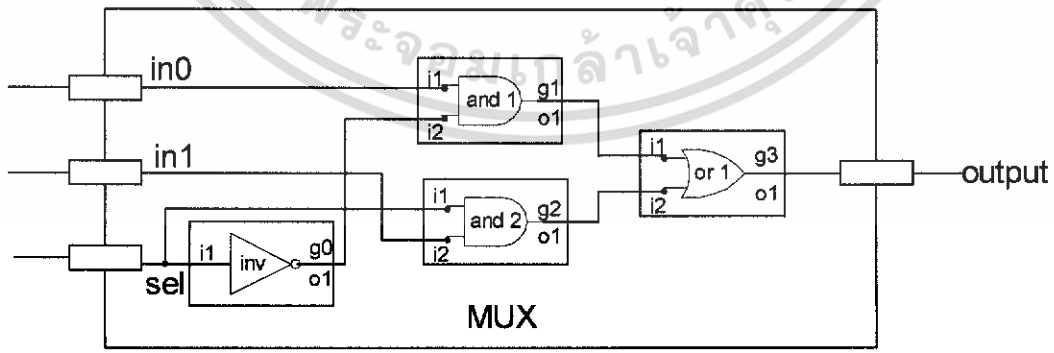
- ประเภทการไหลของข้อมูล (Dataflow description)
- ประเภทพฤติกรรม (Behavioral description)
- ประเภทโครงสร้าง (Structure description)
- ประเภทผสม (Mixed model description)

```
ARCHITECTURE data_flow OF mux IS
BEGIN
Output <= ((NOT se1)AND in0)OR(se1 AND
in1);
END data_flow;
```

รูปที่ 2.16 แสดงหน่วยการออกแบบสถาปัตยกรรมของมัลติเพล็กซ์ตามฟังชั่นบูลลีน

Output= (se1.in0) + (se1.in1)

จากรูปที่ 2.16 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลไหลเข้า *(in0, in1)* กับข้อมูลที่ไหลออก (output) ประกอบด้วยชุดคำสั่งแบบแข่งขันขนานเพียงชุดเดียวซึ่งเขียนเป็นประเภทการไหลของข้อมูลมัลติเพล็กซ์ หรือระดับการถ่ายโอนข้อมูลระหว่างรีจีสเตอร์ (RTL: Register Transfer Level)



รูปที่ 2.17 แสดงโครงสร้างภายในสถาปัตยกรรมของการมัลติเพล็กซ์

จากรูปที่ 2.17 เป็นหน่วยการออกแบบสถาปัตยกรรมของมัลติเพล็กซ์ประเภทโครงสร้างโดยใช้อินเวอร์เตอร์ (inv ที่ตำแหน่ง g0), แอนด์เกท 2 อินพุตจำนวน 2 ตัว (and2 ที่ตำแหน่ง g1 และ g2) และออร์เกท 2 อินพุต (or1 ที่ตำแหน่ง g3) มาสร้างตามฟังก์ชั่นบูลลีนของรูปที่ 2.18

```
ARCHITECTURE struc OF mux IS
COMPONENT inv
PORT (i1 : IN BIT ; o1 : OUT BIT);
COMPONENT and2
PORT (i1,i2 : IN BIT ; o1 : OUT BIT);
COMPONENT or2
PORT (i1,i2 : IN BIT ; o1 : OUT BIT);
END COMPONENT;
SIGNAL int0,int1,int2: BIT;
BEGIN
g0 : inv PORT MAP(i1 =>se1,o1 =>int0);
g1 : and2        PORT MAP(i1 =>in0,i2 =>int0,
o1 =>int1);
g3 : or2 PORT MAP(i1 =>in1,o1 =>int2, o1
=>output);
```

รูปที่ 2.18 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพล็กซ์ประเภทโครงสร้าง

```
ARCHITECTURE behav OF mux IS
BEGIN
PROCESS (in0,in1,se1)
BEGIN
        IF (se1 = '0') THEN output <=in0:
        ELSE output <= in1;
        END IF;
END PROCESS
END behav;
```

รูปที่ 2.19 หน่วยการออกแบบสถาปัตยกรรมของมัลติเพล็กซ์ประเภทพฤติกรรม

ไม่ว่าเขียนบรรยายส่วนของสถาปัตยกรรมของมัลติเพล็กซ์ในลักษณะของพฤติกรรมประเภทการไหลของข้อมูล ประเภทโครงสร้างหรือประเภทผสมที่นำเอาแต่ละประเภทมาเขียนไว้ในหน่วยของสถาปัตยกรรม ก็ตามต่างก็มีพฤติกรรมเดียวกันและจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ซึ่งก็เป็นข้อดีของภาษาวีเอชดีแอล

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของวีเอชดีแอลจะประกอบด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม(Architecture) ดังแสดงในรูปที่ 2.20 โดยในการบรรยายการเชื่อมต่อจะขึ้นต้นด้วยคำว่า Entity แล้วตามด้วยชื่อขององค์ประกอบแล้วตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ต การติดต่อ อินพุต-เอาท์พุต ขององค์ประกอบส่วนลักษณะภายนอกอื่นๆ เช่นเวลา อุณหภูมิก็สามารถเข้ารวมไปในส่วนนี้ได้เช่นกันในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้บรรยายหน้าที่การ

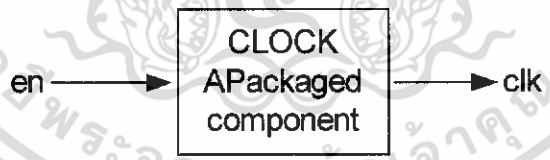
ทำงานขององค์ประกอบอื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 2.20 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจากคำว่า BEGIN เป็นต้นไป

```
ENTITY component_name IS
        Input and output ports
        Physical and other parameters
END [component_name];

ARCHITETURE identifier OF component_name IS
        [declartion]
BEGIN
        specification of the functionality of the component
        in terms of its input lines and as influenced
        by physical andother parameters
END[identifier];
```

รูปที่ 2.20 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบโดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 2.21 ซึ่งเป็นบล็อกไดอะแกรมและการบรรยายการเชื่อมต่อขององค์ประกอบ สำหรับตัวจ่ายสัญญาณนาฬิกาในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็น clock_component ตามด้วยคำว่า PORT และชื่อของพอร์ตที่อยู่ในวงเล็บ ส่วน IN และ OUT เป็นการกำหนดโหนดของสัญญาณให้เป็นอินพุตหรือเอาท์พุต และ BIT เป็นการแสดงชนิดของข้อมูล




```
ENTITY clock_component IS
        PORT (en : IN BIT; ck :OUT
BIT)
END clock_name;
```

รูปที่ 2.21 บล็อกไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_component การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายในส่วนนี้ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณเอาท์พุตในเทอมของอินพุตหรือในรูปขององค์ประกอบอื่นๆหรือทั้งสองอย่างรวมกันก็ได้ดังตัวอย่างการบรรยายของ clock_component ในรูปที่ 2.22 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุตและ ck เป็นเอาท์พุตของ PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิง

พฤติกรรมและภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1" จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (comprement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาท์พุต และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITETURE behavioral OF clock_component IS

BEGIN
        PROCESS
                VARIABLE periodic : BIT := '0';
        BEGIN
                IF en='1' THEN
                periodic := Not periodic;
                END IF;
                ck <= periodic;
                WAIT FOR 1 US;
        END PROCESS
END behavioral;
```

รูปที่ 2.22 การบรรยายเชิงพฤติกรรมของ clock_component

3. หน่วยการออกแบบแพ็กเกจ

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อยที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเกจซึ่งหน่วยการออกแบบต่างๆเช่นหน่วยการออกแบบ entity หน่วยของการออกแบบสถาปัตยกรรมหรือหน่วยการออกแบบแพ็กเกจอื่นๆสามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนี้สิ่งที่นิยมทำกันมากคือการนำรูปแบบมาตรฐานต่างๆเช่นอุปกรณ์มาตรฐาน(เช่นไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจที่ทุกคนสามารถเข้าถึงได้ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration) และส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจากแพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษาวีเอชดีแอลสามารถกระทำได้ด้วยชุดคำสั่ง USE

- การประกาศแพ็กเกจ (Package declaration)

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ ส่วนการประกาศแพ็กเกจเนื่องจากเป็นส่วนหนึ่งที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภาพในแพ็กเกจสำหรับนำไปใช้ภายนอกตัวแพ็กเกจเอง ถ้ามีการประกาศสิ่งใดๆ ในส่วนของบอดี้แพ็กเกจแต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติกรรมไม่สามารถนำไปใช้งานส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือจุดเชื่อมต่อ หรือ พอร์ตที่มีหน้าที่เชื่อมต่อกับโลกภายนอกฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นทั่วไปโดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่นใช้สำหรับประกาศชนิด (Type) หรือสัญญาณ เช่นเดียวกับส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจแต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จากรูปแบบอื่นได้

```
PACKAGE package_name IS
        Package_declartive_part
END package_name;
```

รูปที่ 2.23 โครงสร้างโดยทั่วไปของส่วนการประกาศแพ็กเกจ

- บอดี้แพ็กเกจ(Package body)

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังชั่นการทำงานของโปรแกรมย่อยทั้งหลายซึ่งชื่อนองโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้ว ในส่วนของการประกาศแพ็กเกจจุถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจทั้งนี้รวมถึงการกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูประกาศชื่อไว้ก่อน ในส่วนของการประกาศแพ็กเกจและถูกกำหนดค่าในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังรูปที่ 2.24

```
PACKAGE BODY package_name IS
        declartive_part
END package_name;
```

รูปที่ 2.24 โครงสร้างของบอดี้แพ็กเกจ

4. หน่วยการออกแบบ configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ configuration มาเพื่อกำหนดการใช้ configuration ขององค์ประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

```
CONFIGURATION identifier OF entity_name IS
        configuration_declartive_part
END;
```

รูปที่ 2.25 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

- โปรแกรมย่อย

การใช้ฟังก์ชั่นและโพซีเจอร์ในวีเอชดีแอลเปรียบได้กับการใช้โปรแกรมย่อยในกรเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไป ค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้เช่นฟังก์ชัน แทนการกระทำในสมการบูลลีนก็จะมีผลต่อวงจรลอจิกจริงๆ ในขณะที่ใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณการหน่วงเวลาแล้วก็จะไม่มีผล

ต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 2.26 แสดงการใช้โพรซีเจอร์ เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็มและรูปที่ 2.27 แสดงการใช้ฟังก์ชันโดยการกำหนดให้ X เป็นตัวแปรชนิดบิตแทนการกระทำในบูลลีน

```
TYPE type IS ARRAY(7 DOWNTO 0) OF BIT;
...
PROCEDURE byte_to_integer (ib : IN byte;oi : OUT INTEGER) IS
        VARIABLE result: INTEGER := 0;
BEGIN
        FOR : IN 0 TO 7 LOOP
                IF ib(i) = '1' THEN
                result := result + 2**i;
                END IF;
        END LOOP;
        oi := result;
END byte_to_integer
```

รูปที่ 2.26 การใช้โพรซีเจอร์

```
FUNCTION f (a,b,c:BIT) RETURN BIT IS
        VARIABLE x:BIT;
BEGIN
        x :=((NOT a)AND (NOT b)AND c);
END f;
```

รูปที่ 2.27 การใช้ฟังก์ชัน

- โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษาวีเอชดีแอลจะมีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียวกับภาษาซอฟต์แวร์ทั่งไปดังรูปที่ 2.28

```
LOGICAL OPERATORS : NOT AND OR NAND NOR XOR
        OPERAND TYPE : BIT BOOLEAN
        RESULT TYPE : BIT BOOLEAN

RELATIONAL OPERATORS : = /= < <= > >=
        OPERAND TYPE : any type
        RESULT TYPE : Boolean

ARITHMETIC OPERATORS : + - * / ** MOD REM ABS
        OPERAND TYPE : INTEGER REAL Physical
        RESULT TYPE : INTEGER REAL Physical

CONCANTENATION OPERATOR : &
        OPERAND TYPE : ARRAY of any type
        RESULT TYPE : array of any type
        RESULT TYPE : array of any type
```

รูปที่ 2.28 ตัวดำเนินการในภาษาวีเอชดีแอล

- เวลาและความพร้อมเพรียง

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องในทุกๆ เหตุการณ์ที่เกิดขึ้นเสมอ วีเอชดีแอลเป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูปแบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วนของการบรรยายสถาปัตยกรรมจะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมาการทำงานภายในเป็น แบบลำดับคำสั่งก็ตามซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆโปรเซสก็จะทำงานไปพร้อมๆ กันด้วย

- สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูล และมีเรื่องของเวลาเข้ามาเกี่ยงข้องด้วย การกำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการส่งผ่านค่าของสัญญาณ เช่น w<= a AFTER 12 NS หมายถึงการกำหนดสัญญาณ aให้กับ w หลังจากเวลาผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเหมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานแบบลำดับคำสั่งเช่นในฟังก์ชัน โพรซีเจอร์ และโปรเซสสำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์: =

2.13.3 ภาษาวีเอชดีแอลเพื่อการสังเคราะห์

ภาษาวีเอชดีแอลเป็นภาษาที่เขียนเพื่อจำลองการทำงานของวงจร ซึ่งในบางรูปแบบการเขียนไม่สามารถที่จะนำไปสังเคราะห์ได้ทั้งหมด ดังนั้นถ้าต้องการเขียนเพื่อนำไปสังเคราะห์ ควรหลีกเลี่ยงรูปแบบต่างๆ ที่ไม่สามารถที่ไม่สามารถนำไปสังเคราะห์ได้ ในที่นี้ขึ้นอยู่กับความสามารถของโปรแกรมที่ใช้สังเคราะห์แต่ละโปรแกรม โมเดลที่ใช้เขียนกันในภาษาวีเอชดีแอลแบ่งได้เป็น 2 กลุ่มคือ เกตและฟลิปฟลอป ประเภทต่างๆ

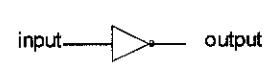
- ตัวอย่างรูปแบบการเขียนเกทพื้นฐาน

output <= a AND b;

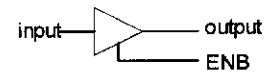
output <= a OR b;

output <= a XOR b;

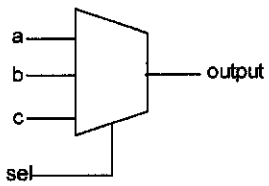
output <= a NOT b;



output <= a input WHEN enb = '1' ELSE "ZZ"



```
WITH sel SELECT
Output <= a WHEN "00" ELSE
          b WHEN "01" ELSE
          c;
```



- ตัวอย่างรูปแบบการเขียนฟลิปฟลอปพื้นฐาน

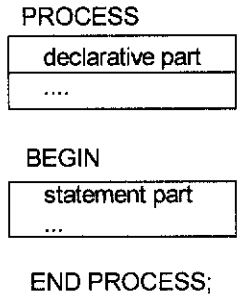
```
SIGNAL input, output: vlbit_ld (3 DOWNTO 0);
SIGNAL clock, enable, reset: vlbit;

    PROCESS BEGIN
        WAIT UNTIL PRISING (clock);
            Output <= input;
    END PROCESS;

    PROCESS BEGIN
    WAIT UNTIL PRISING (clock) OR (reset = '1');
    IF (reset = '1') THEN output <= '00';
    ELSE IF (enable = '1') THEN output <- input;
    END IF;
    END PROCESS;
```

2.13.4 การบรรยายเชิงพฤติกรรม

เป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในอัลกอริธึม สำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นเนื่องมาจากการเปลี่ยนแปลงสภาวะข้อมูล เข้ามาโดยไม่คำนึงถึงลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ภายในว่าจะเป็นอย่างไร หัวข้อนี้จะพูดถึงการบรรยายเชิงพฤติกรรมแทนการใช้โมดูลฮาร์ดแวร์ รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

โปรเซส (Process)

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ในการกำหนดให้กับสัญญาณ โปรเซสจะอยู่ในสถานะที่เตรียมไร้อมอยู่เสมอ และจะปฏิบัติคำสั่งพร้อมๆ กันกับคำสั่งอื่นๆ ที่อยู่ในสถาปัตยกรรมบรรยายเดียวกัน โดยโปรเซสนี้จะปฏิบัติตามคำสั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับสัญญาณที่อยู่ทางด้านท้ายของสัญลักษณ์กำหนดค่าให้สัญญาณ (=>) โปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบคำสั่งด้วย END PROCESS ตามรูปที่ 2.29 โดยที่ส่วนประกอบของโปรเซสจะมีส่วนของการประกาศตัวแปรและส่วนของคำสั่งปฏิบัติงาน



รูปที่ 2.29 แสดงส่วนประกอบของการบรรยายแบบโปรเซส

- การกำหนดตัวดำเนินการภายในโปรเซส

ตัวดำเนินการภายในโปรเซสมี 3 ชนิดคือ ตัวแปร (Variable), ไฟล์ (File) และค่าคงที่ (Constant) ซึ่งตัวดำเนินการทั้ง 3 นี้ หากมีการประกาศไว้ในโปรเซสใด ก็จะสามารถใช้ได้เฉพาะในโปรเซสนั้น สำหรับการติดต่อกับภายนอกหรือระหว่างโปรเซสจะสามารถทำได้โดยใช้สัญญาณหรือค่าคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในรูปที่ 2.30 แสดงการประกาศตัวกระทำภายในโปรเซสจะถูกนำมาใช้ ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และคำสั่ง BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ตอนเริ่มต้นของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายใน โปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้โปรแกรมย่อยนั้นๆ

```
PROCESS
        FILE flush : TEXT IS IN "filename.dat";
        VARIABLE var : BIT;
        CONSTANT n : INTEGER :=0;
BEGIN
        ...
END PROCESS;
```

รูปที่ 2.30 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซสการกำหนดการทำงานภายในโปรเซส

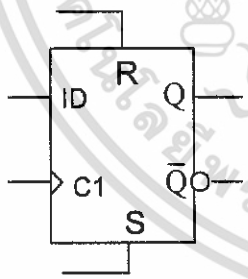
การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequence) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยคเงื่อนไขหรือการทำซ้ำได้เช่น IF – THEN – ELSE, CASE – WHEN, FOR LOOP และ WHILE LOOP ดังตัวอย่างรูปที่ 2.31

```
ARCTHITECTURE demo OF partial_process IS
....
BEGIN
        PROCESS
        ....
        BEGIN
                ...
                x <= '1';
                IF x = '1' THEN
                perform action_2
                END IF;
                ...
        END PROCESS;
END demo;
```

รูปที่ 2.31 เงื่อนไขการทำงานในโปรเซส

- การกระตุ้นและการยับยั้งการทำงานของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาวะเตรียมพร้อม และมีการทำงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์เกิดขึ้น อย่างไรก็ดีเราก็ยังสามารถกระตุ้นหรือยับยั้งการทำงานภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้องการให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนดไว้เท่านั้น ส่วนเหตุการณ์ใดที่ไม่ได้กำหนดไว้ก็จะไม่ส่งผลถึงการทำงานภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS รูปที่ 2.32(ช) แสดงตัวอย่างโมเดล และรูปที่ 2.32(ซ) เป็นตัวอย่างการบรรยาย ส่วนในรูปที่ 2.33 แสดงถึงการบรรยายเชิงพฤติกรรมการทำงานของฟลิปฟลอป โดยที่รูปที่ 2.33(ฌ) เป็นการใช้ตัวกระทำภายนอกโปรเซส และรูปที่ 2.33(ญ) เป็นการใช้ตัวกระทำงานภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการทำงานภายในโปรเซส



(ช)

```
ENTITY component_name IS
        Input and output ports
        Physical and other parameters
END pomponent_name;
```

(ซ)

รูปที่ 2.32 โมเดลดีฟลิปฟลอป (D Flip-flop)

(ช) แสดงตัวอย่างโมเดลดีฟลิปฟลอป

(ซ) เป็นตัวอย่างการบรรยายพฤติกรรมของดีฟลิปฟลอป

```
ARCHITECTURE behavioral OF d_sr_flipflop IS
        SIGNAL state : BIT := '0';
BEGIN
        dff : PROCESS (rst, set, clk)
                BEGIN
                IF set='1' THEN
                state <= '1' AFTER rq_delay;
                ELSIF rst = '1' THEN
                state <= '0' AFTER cq_delay;
                ELSIF clk = '1' AND clk ' EVENT
THEN
                state <= d AFTER cq_delay;
                END IF;
        END PROCESS dff;
        q <= state;
        qb <= NOT state;
END behavioral;
```

(ฌ)

```
ARCHITECTURE average_delay_behavioral OF d_sr_flipflop IS
BEGIN
        dff : PROCESS (rst, set, clk)
                VARIABLE state : BIT := '0';
        BEGIN
                IF set='1' THEN
                state <= '1';
                ELSIF rst = '1' THEN
                state <= '0';
                ELSIF clk = '1' AND clk ' EVENT THEN
                state <= d;
                END IF;
        q <= state AFTER (sq_delay + rq_delay + cq_delay)/3;
        qb <= NOT state AFTER (sq_delay + rq_delay + cq_delay)/3;
        END PROCESS dff;
END average_delay_behavioral;
```

(ญ)

รูปที่ 2.33 การบรรยายเชิงพฤติกรรมของดีฟลิปฟลอป

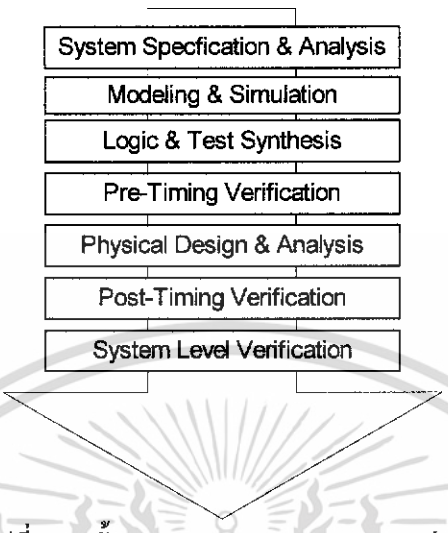
(ฌ) เป็นการใช้ตัวกระทำภายนอกโปรเซส

(ญ) เป็นการใช้ตัวกระทำงานภายในโปรเซส

2.13.5 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน ผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของบล็อกไดอะแกรมก่อนทำการวิเคราะห์ให้ถึงรายละเอียดต่อไป ซึ่งภาษาวีเอชดีแอลนั้นอนุญาตให้อธิบายและวิเคราะห์การทำงานของแต่ละบล็อก รวมถึงการการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามที่ต้องการ นอกจากนี้ยังสามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอนได้ ซึ่งหลักการนี้จะสอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top – Down Design) นั่นเอง เมื่อเทียบแบการรออกแบบจากล่างขึ้นบน (Bottom – Up Design) จะเห็นได้ว่าการออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบเข้าเป็นวงจรที่ต้องการออกแบบก่อนแล้วจึงทำการจำลองการทำงาน และตรวจสอบการทำงานความ

ถูกต้อง ดังนั้นการใช้ภาษาวีเอชดีแอล กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับผู้ออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น และยังช่วยให้ลดเวลาและค่าใช้จ่ายในการออกแบบอีกด้วย



รูปที่ 2.34 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 2.34 แสดงถึงขั้นตอนการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างบ้างเล็กน้อย เนื่องจากขั้นตอนการผลิตสามารถทำได้หลายวิธี สำหรับขั้นตอนการออกแบบจากบนลงล่างมีดังนี้

1. สร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบเพื่อหาแนวทางและหลักการในการแก้ปัญหา

2. เขียนรูปแบบของระบบที่ต้องการ เพื่อบรรยายพฤติกรรมการทำงาน พร้อมทั้งจำลองการทำงานเพื่อเปรียบเทียบความถูกต้องตามข้อกำหนด

3. หลังจากที่ได้หลักการและขั้นตอนที่ผ่านกรตรวจสอบความถูกต้องแล้ว นำสิ่งเหล่านี้ไปเพิ่มในรายละเอียด เป็นขั้นที่ 2 จนสามารถนำไปผลิตวงจรจริงได้ วงจรที่ออกแบบจะถูกกำหนดขึ้นและระบบจะช่วยออกแบบสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์หรือวงจรในระดับเกต และการเชื่อมต่อระหว่างกันระหว่างของอุปกรณ์เหล่านั้นหรือ ไม่ก็อยู่ในรูปของรายการ (Net list) ที่สามารถนำไปผลิตเป็นตัวอุปกรณ์อื่นได้

4. หลังจากการวิเคราะห์วงจรให้อยู่ในระดับเกตหรือ Net list แล้ว ข้อมูลนี้ถูกจะใช้สำหรับจำลองการทำงานในเรื่องความถูกต้องของฟังก์ชั่น พร้อมกับนำข้อมูลที่เกี่ยวข้องกับเวลาเข้ามาประกอบการพิจารณาด้วย ซึ่งตามปกติแล้วอุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation Delay Time) เสมอ ถึงแม้ว่าจะเป็นเวลาน้อยมากในระดับนาโนวินาทีก็ตาม แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกตของฟังก์ชั่นต่างๆ เป็นหมื่นๆ เกตขึ้นไป เวลาหน่วงก็จะสะสมมากขึ้น บางครั้งมากจนทำให้การทำงานของวงจรรวมผิดพลาดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูงๆ ได้

5. ผลิตวงจรจริง โดยการนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต

6. ทำการตรวจสอบการทำงานและตัวแปรการทำงานทางด้านเวลาทั้งหมด เพื่อความถูกต้องก่อนนำไปเข้ากับอุปกรณ์ชิ้นอื่นๆ เนื่องจากขั้นตอนนี้ วงจรที่ออกแบบสุดท้ายจะประกอบด้วยจุดต่อรับอินพุตและเอาท์พุต ซึ่งเป็นจุดต่อกับสัญญาณภายนอก

7. นำวงจรที่ออกแบบไว้ประกอบเข้าอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทดสอบการทำงานอีกครั้งทั้งระบบ เพื่อควบคุมและปรับปรุงคุณภาพของชิ้นงานให้ดี

2.14 การเชื่อมต่อของพอร์ต Ps/2

พอร์ต PS/2 ประกอบด้วย 6 ขา (แต่ใช้งานแค่ 4 ขา) ประกอบไปด้วย ขากราวด์, ขาไฟเลี้ยง คีย์บอร์ด, ข้อมูลที่ส่งจากคีย์บอร์ดและสายคลอกคีย์บอร์ด ซึ่งสายคล็อกคีย์บอร์ด เป็นแบบ 2 ทิศทาง และโดยปกติจะถูกควบคุมโดยคีย์บอร์ดเอง แต่มันสามารถที่จะถูกบังคับโดยระบบคอมพิวเตอร์หรือในกรณีนี้คือ FLEX chipเมื่อมันต้องการที่จะหยุดการส่งข้อมูลจากคีย์บอร์ด ทั้งคีย์บอร์ดและระบบคอมพิวเตอร์สามารถบังคับสายข้อมูลได้ สายข้อมูลเป็นแหล่งข้อมูลเพียงอันเดียวสำหรับการส่งระหว่างคอมพิวเตอร์กับคีย์บอร์ด คีย์บอร์ดและระบบสามารถแลกเปลี่ยนคำสั่งหลายๆคำสั่งและข้อความ ดังในตารางที่ 2.1

ตารางที่ 2.1 คำสั่งของ PS/2 Keyboard และค่าที่ได้

คำสั่งส่งไปยังคีย์บอร์ด	Hex Value
รีเซ็ต คีย์บอร์ด คีย์บอร์ดกลับไปเป็น AA, 00 หลังจากทดสอบตัวเอง	FF
ส่งข้อความใหม่	FE
ตั้งค่าปุ่ม typematic (autorepeat) XX เป็น scan code สำหรับปุ่ม	FB, XX
ตั้งค่าปุ่ม make และ break	FC, XX
ตั้งค่าปุ่ม make	FD, XX
ตั้งค่าทุกปุ่ม typematic, make และ break	FA
ตั้งค่าทุกปุ่ม make	F9
ตั้งค่าทุกปุ่ม make และ break	F8
ทำให้ทุกปุ่มเป็น typematic (autorepeat)	F7
ตั้งค่าเป็นค่าปกติ	F6
ตั้งค่า typematic (autorepeat) rate และ delay Bits 6 และ 5 เป็น delay (250 ms ถึง 1 sec) Bits 4 ถึง 0 เป็น rate (ทุก “0”เป็น -30x/sec ถึง ทุก “1” เป็น 2x/sec)	F3, XX
อ่านค่า คีย์บอร์ด ID คีย์บอร์ดส่ง FA, 83, AB	F2

ตั้งค่า ชุด scan code XX เป็น 01, 02, หรือ 03	F0, XX
Echo	EE
ตั้งค่า คีย์บอร์ด LED XX เป็น 00000 scroll, Num, Caps Lock bits 1 เป็น LED เปิด และ 0 เป็น LED ปิด	ED, XX
ส่งข้อความใหม่	FE
สองข้อความเสียในหนึ่งแถว	FC
คีย์บอร์ด คำสั่ง Acknowledge ส่งโดยคีย์บอร์ดหลังแต่ละ ไบท์คำสั่ง	FA
ตอบสนองต่อคำสั่ง Echo	EE
คีย์บอร์ดผ่านการทดสอบตนเอง	AA
คีย์บอร์ด buffer มากเกินไป (overflow)	00

2.15 Keyboard scan codes

คีย์บอร์ดเป็นการเข้ารหัสปกติโดยการคีย์สวิตซ์ (Key switch) ในเมตริกของแถวและคอลัมน์ ซึ่งแถวและคอลัมน์ทั้งหมดเป็นการเช็คอย่างต่อเนื่องโดยตัวเข้ารหัสของคีย์บอร์ด (Keyboard encoder) หรือ "สแกน" ที่อัตราสูงเพื่อหาทุกขั้นตอนที่มีการเปลี่ยนแปลงของปุ่ม ข้อมูลจากคีย์บอร์ดที่ผ่านเป็นอนุกรมไปยังคอมพิวเตอร์ จากคีย์บอร์ดจะใช้การสแกนรหัส แต่ละปุ่มบนคีย์บอร์ดมีรหัสเฉพาะบนพื้นฐานของเมตริกแถวและคอลัมน์แอดแดรสเพื่อที่จะระบุปุ่มที่ถูกกด

ความแตกต่างหลายๆ แบบของการสแกนรหัสที่ใช้ขึ้นอยู่กับชนิดของคีย์บอร์ดที่ใช้ PS/2 Keyboard มีชุดของรหัสอยู่ 2 ชุด โดยชุดรหัสเริ่มต้นที่จะถูกตั้งค่าขณะที่มีการใช้งาน ยกเว้นกรณีที่ระบบคอมพิวเตอร์ส่งคำสั่งคีย์บอร์ดเพื่อใช้ในการเลือกชุดคำสั่ง ปกติคอมพิวเตอร์จะส่งคำสั่งไปยังคีย์บอร์ดด้วยกำลังสูงและมันจะใช้ชุดตัวเลือกรหัส เพื่อจะประสานคีย์บอร์ด

2.16 รหัสสร้างและรหัสหยุด (Make and Break Codes)

คีย์บอร์ดจะตรวจรหัสค่าของรหัสสร้าง (Make code) และรหัสหยุด (Break code) รหัสสร้างจะถูกส่งทุกครั้งที่ปุ่มถูกกด เมื่อปุ่มถูกปล่อยรหัสหยุดก็จะถูกส่งไปทันที สำหรับปุ่มกดโดยมาก รหัสหยุดเป็นจะข้อมูลที่ขึ้นต้นด้วยรหัส F0 แล้วตามด้วยรหัสสร้าง สำหรับการคีย์แต่ละครั้งให้นึกถึงด้วยว่าเมื่อทำการพิมพ์ เป็นไปได้ว่าที่จะกดปุ่มต่อไปก่อนที่จะปล่อยปุ่มแรกที่กดอยู่ ซึ่งการใช้การตั้งค่าแบบนี้ ระบบจะสามารถบอกได้ว่าปุ่มใดถูกกดอยู่หรือไม่ได้กด และในกรณีที่มีการกดปุ่มมากกว่าหนึ่งปุ่มมันจะสามารถรู้ได้ว่าปุ่มไหนถูกปล่อยแล้ว ตัวอย่างหนึ่งของกรณีนี้คือ เมื่อกดปุ่ม Shift ค้างไว้ แล้วกดปุ่ม '3'ควรที่จะตอบสนองกลับเป็นค่า '#' แทนที่จะออกเป็น '3' สังเกตได้ว่า ถ้ากดปุ่มค้างไว้จะทำให้มีการส่งรหัสสร้าง

อย่างต่อเนื่องผ่านไทพ์เมติก (typematic rate) จนกระทั่งปุ่มนั้นถูกปล่อย ซึ่งเป็นเวลาเดียวกันกับที่รหัสหยุดถูกส่งไป

2.17 การส่งผ่านข้อมูลแบบอนุกรมผ่านพอร์ต PS/2

การสแกนรหัสที่ถูกส่งมาอย่างต่อเนื่องที่ชุดละ 11 บิตในสายข้อมูลสองทิศทาง เมื่อทั้งคีย์บอร์ดและคอมพิวเตอร์ต่างก็ไม่ต้องการที่จะส่งข้อมูล สายข้อมูลและสาย Clock จะอยู่ในสถานะสูง (ไม่ทำงาน) ดังที่เห็นในรูปที่ 10.1 การส่งข้อมูลของปุ่มๆ หนึ่งหรือค่าของคำสั่งของอุปกรณ์ดังกล่าว

1.บิตเริ่มต้น (Start bit) "0"

2. ข้อมูล 8 บิตเก็บรหัสของปุ่มในรูปของลำดับบิตต่ำไปสูง (Low to high bit order)

3.เศษ Parity bit อย่างเช่น ข้อมูล 8 บิตบวกกับ parity bit เป็นเลขเศษของหนึ่ง

4. บิตหยุด(Stop bit) "1"

กระบวนการต่อไปนี้เป็นเหตุการณ์ระหว่างการส่งคำสั่งโดยคีย์บอร์ด

1.คีย์บอร์ดเช็คเพื่อให้แน่ใจว่าทั้ง Clock และ data line อยู่ในสถานะไม่ทำงาน (inactive) ซึ่งจะหมายความว่าอยู่ในสถานะสูง ถ้าทั้งคู่อยู่ในสถานะ inactive คีย์บอร์ดจะเตรียมส่ง start bit โดนการทำให้ data line อยู่ในสถานะต่ำ

2. จากนั้นคีย์บอร์ดจะปล่อยให้ Clock line เป็น low ประมาณ 35 us.

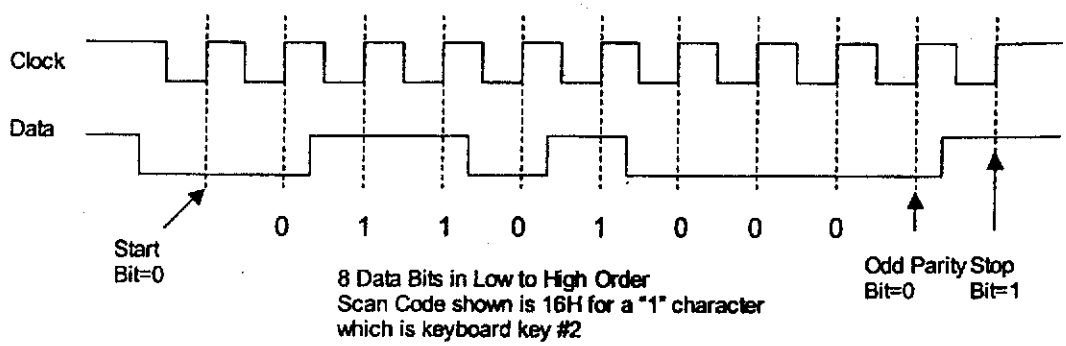
3. จากนั้นคีย์บอร์ดจะออก Clock เหลืออยู่ 10 bits ที่อัตราการส่งประมาณ 70 us. ต่อคาบเวลาของ Clock โดยคีย์บอร์ดจะทำการปรับทั้ง data line และ clock line

4. คอมพิวเตอร์ตอบสนองต่อ Start bit และ รับชุดข้อมูล ชุดข้อมูล 8 bits ซึ่งตามด้วย parity bit และปิดท้ายด้วย high stop bit ถ้าคีย์บอร์ดต้องการที่จะส่งข้อมูลมากกว่านี้ มันจะต้องตามด้วย bit ที่ 12 ทันทีซึ่งเป็น start bit

รูปแบบนี้จะมีการทำซ้ำไปจนกว่าคีย์บอร์ดจะทำการส่งข้อมูลเสร็จสิ้น ที่จุดที่ Data lineและ colck line จะกลับสภาพเป็น high state (inactive) ในรูปที่ 10.1 คีย์บอร์ดทำการส่ง scan code ของ 16 สำหรับปุ่ม "1" และมี 0 parity bit

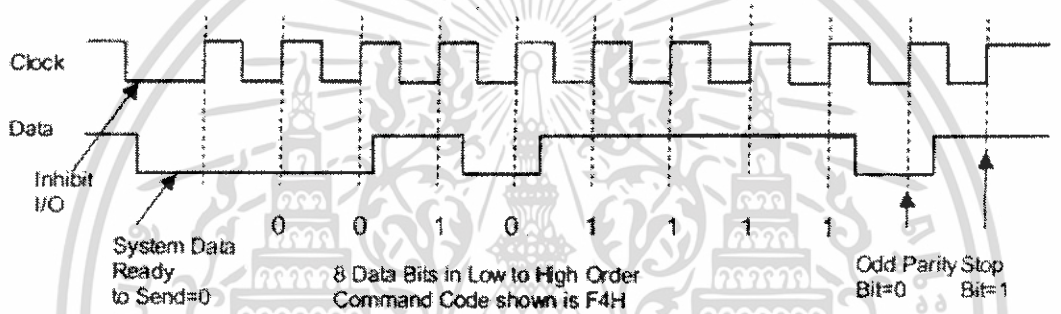
เมื่อมีการใช้โค้ด ซึ่งตัวมันจะต้องการกรองสัญญาณนาฬิกาความเร็วต่ำของคีย์บอร์ดเพื่อให้แน่ใจว่ากระบวนการเชื่อถือได้กับ Fast logic ใน FLEX chip เมื่อไรก็ตามที่ electrical pulse ถูกส่งไปบนสายสนามแม่เหล็กที่เกิดขึ้นบนสายจะมีผลให้เกิดการรบกวน และมีผลทำให้เกิดการสะท้อนกลับจากจุดสิ้นสุดของสาย บางPS/2 คีย์บอร์ดและเมาส์มีการสะท้อนกลับของ pulse บนสายเคเบิลที่แรงพอที่จะทำให้เกิด addition clock บน clock line

วิธีหนึ่งในการแก้ปัญหาการสะท้อนกลับของพัลส์ Pulse ได้โดยการป้อน สัญญาณPS/2 clock เข้าไปใน 8 bit shift register โดยใช้ clock 25 MHz และ bit ของ shift register เข้าด้วยกัน และใช้ค่าที่ได้จาก AND gate เป็นเหมือนกับ clock ที่ผ่านการกรองมาแล้ว นี่จะป้องกัน noise และ ringing บน clock line จากผลของ clock ที่เพิ่มขึ้นระหว่างการเปลี่ยน serial ไปเป็น parallel ใน FLEX chip



รูปที่ 2.35 รูปแบบการส่งข้อมูลของ Keyboard

ปัจจุบันมีคีย์บอร์ดและเมาส์ส่วนน้อยที่จะทำงานโดยปราศจากตัวกรอง clock ซึ่งโดยส่วนใหญ่แล้วทั้งคีย์บอร์ดและเมาส์นั้นจะมีการทำงานร่วมกับตัวกรอง clock และมีความสัมพันธ์อย่างง่ายๆระหว่างอุปกรณ์



รูปที่ 2.36 รูปแบบการส่งข้อมูลของระบบ ส่งคำสั่งไปยัง PS/2 คีย์บอร์ด

ดังที่เห็นในรูปที่ 2.36 รูปแบบการส่งข้อมูลของระบบคอมพิวเตอร์ หรือ FLEX chip ส่งคำสั่งไปยัง PS/2 คีย์บอร์ด

1.ระบบปรับ clock line เป็น low ประมาณ 60us เพื่อยับยั้งการส่งข้อมูลทุกอย่างของ คีย์บอร์ด clock line เป็นแบบ 2 ทิศทาง

2. ระบบจะปรับสายข้อมูลเป็น low และจากนั้นจะปล่อย clock line เพื่อรับสัญญาณที่มีข้อมูลสำหรับคีย์บอร์ด

3. คีย์บอร์ดจะสร้างสัญญาณนาฬิกาในลำดับที่ clock ที่ออกมาเป็น serial bits ในคำสั่ง

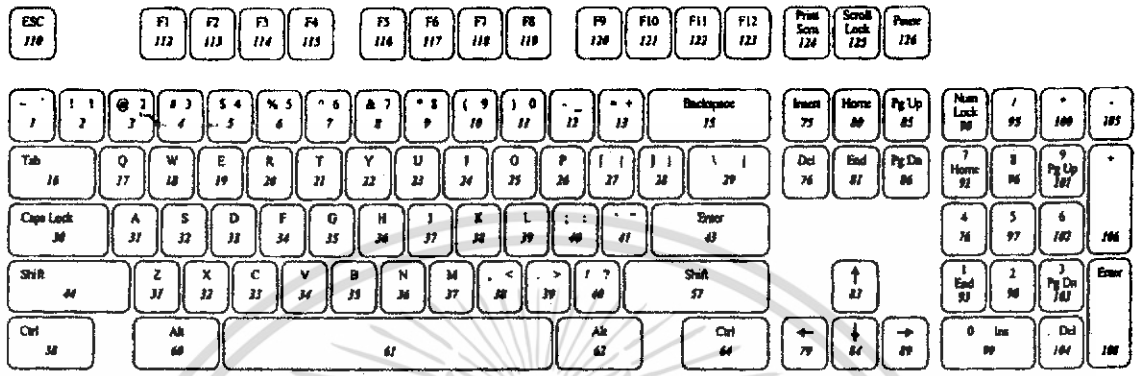
4. ระบบจะส่งคำสั่ง 8 บิต ตามด้วย parity bit และ stop bit

5. หลังจาก stop bit จะถูกปรับเป็น high สายข้อมูลถูกปล่อย

หลังจากทำคำสั่งแต่ละคำสั่งเสร็จ คีย์บอร์ดจะส่งสัญญาณ acknowledge (ACK), FA, ถ้ามันได้รับข้อมูลสมบูรณ์ ถ้าระบบไม่ปล่อย data line, คีย์บอร์ดจะทำการสร้าง clock ต่อไปจนเสร็จ, มันจะคำสั่งให้มีการส่งสัญญาณคำสั่งใหม่, FE หรือ FC, ไปยังระบบ parity error หรือ missing stop bit จะสร้างสัญญาณคำสั่งส่งใหม่

2.18 ชุด scan code สำหรับ PS/2 คีย์บอร์ด

PS/2 คีย์บอร์ดทำงานได้หลายภาษาที่ตัวอักษรต่างกันที่ถูกพิมพ์ลงบนปุ่ม กระบวนการ2 ขั้นตอนที่ต้องการเพื่อหา scan code ปุ่มหมายเลขถูกใช้เพื่อหา scan code หมายเลขปุ่มที่ต้องการสำหรับตาราง scan code นั้นแสดงในรูปที่ 2.37 สำหรับคีย์บอร์ดภาษาอังกฤษ



รูปที่ 2.37 หมายเลขปุ่มบน Keyboard เพื่อสร้าง scan code

แต่ละปุ่มส่ง make code เมื่อมีการกดและจะส่ง break code เมื่อปล่อยปุ่ม และเมื่อหลายๆ ปุ่มถูกกด make code หลายๆ อันก็จะถูกส่งก่อน break code

เมื่อคีย์บอร์ดเริ่มใช้งาน scan code นี้จะถูกใช้เป็นคำสั่งที่ต้องส่งไปยังคีย์บอร์ดเพื่อใช้ชุดของ scan code อื่นๆ ซึ่ง PC จะส่งคำสั่งเริ่มแรกเพื่อบังคับคีย์บอร์ดให้ใช้ชุด scan code อื่น

แต่ถ้าไม่มีการติดต่อกับ PC กระบวนการนี้ก็จะง่ายมาก ซึ่ง scan code ชุดปกตินี้จะถูกใช้ โดยที่ไม่จำเป็นต้องส่งคำสั่งไปยังคีย์บอร์ด ปุ่มในตารางที่ 2.2 สำหรับชุด scan code ปกติเป็นแบบ typematic (คือมันจะทำการส่ง make code ซ้ำ อัตโนมัติเมื่อมีการกดปุ่มค้าง)

ตารางที่ 2.2 scan code สำหรับ PS/2 คีย์บอร์ด

Key#	Make Code	Break Code	Key#	Make Code	Break Code	Key#	Make Code	Break Code
1	0E	F0 0E	31	1C	F0 1C	90	77	F0 77
2	16	F0 16	32	1B	F0 1B	91	6C	F0 6C
3	1E	F0 1E	33	23	F0 23	92	6B	F0 6B
4	26	F0 26	34	2B	F0 2B	93	69	F0 69
5	25	F0 25	35	34	F0 34	96	75	F0 75
6	2E	F0 2E	36	33	F0 33	97	73	F0 73
7	26	F0 26	37	3B	F0 3B	98	72	F0 72
8	2D	F0 2D	38	42	F0 42	99	70	F0 70
9	2E	F0 2E	39	4B	F0 4B	100	7C	F0 7C

10	46	F0 46	40	4C	F0 4C	101	7D	F0 7D
11	45	F0 45	41	52	F0 52	102	74	F0 74
12	4E	F0 4E	43	5A	F0 5A	103	7A	F0 7A
13	55	F0 55	44	12	F0 12	104	71	F0 71
15	66	F0 66	46	1A	F0 1A	105	7B	F0 7B
16	0D	F0 0D	47	22	F0 22	106	79	F0 79
17	15	F0 15	48	21	F0 21	110	76	F0 76
18	1D	F0 1D	49	2A	F0 2A	112	05	F0 05
19	24	F0 24	50	32	F0 32	113	06	F0 06
20	2D	F0 2D	51	31	F0 31	114	04	F0 04
21	2C	F0 2C	52	3A	F0 3A	115	0C	F0 0C
22	35	F0 35	53	41	F0 41	116	03	F0 03
23	3C	F0 3C	54	49	F0 49	117	0B	F0 0B
24	43	F0 43	55	4A	F0 4A	118	83	F0 83
25	44	F0 44	57	59	F0 59	119	0A	F0 0A
26	4D	F0 4D	58	14	F0 14	120	01	F0 01
27	54	F0 54	60	11	F0 11	121	09	F0 09
28	5B	F0 5B	61	29	F0 29	122	78	F0 78
29	5D	F0 5D	62	E0 11	E0 F0 11	123	07	F0 07
Scan code ที่เป็นฟังชั่นของปุ่ม shift, control, alt, หรือ num-lock								

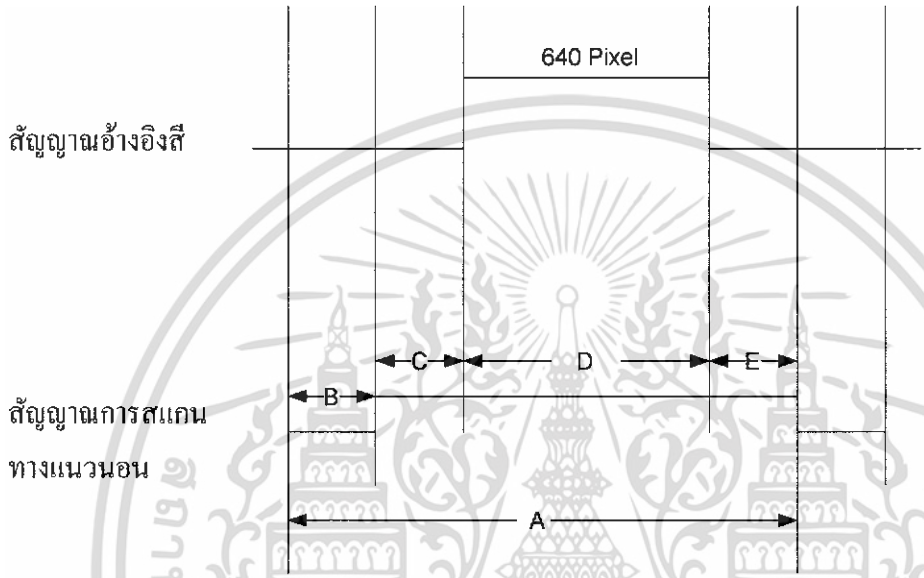
บทที่3

การคำนวณและการสร้าง

3.1 การสร้างสัญญาณควบคุม VGA

ในโครงงานนี้ได้ใช้ไอซีเบอร์ EPF 10K10 โดยรับสัญญาณ clock จาก Oscillator 25.175 MHz โดยใช้การเขียนภาษา VHDL และฟังก์ชั่น MaxplusII

3.1.1การสร้างสัญญาณสแกนทางแนวนอน(Horizontal Synchronization)



รูปที่ 3.1 รูปสัญญาณการสแกนและสัญญาณการอ้างอิงตำแหน่งทางแนวนอน

สัญญาณการสแกนทางแนวนอนจะเป็นตัวกำหนดการสแกนทางแนวนอนในแต่ละแถวจะมีค่าข้อมูล อยู่ในช่วงประมาณ 794 คือช่วง C+D+E นั่นเองโดยจะมีสัญญาณการอ้างอิงสีเป็นสัญญาณ บอกถึงต้องการสแกนในช่วง 0-639 ตามการสแกนตามมาตรฐานนั่นเอง

Parameter	A	B	C	D	E
Time	31.77 µs	3.77 µs	1.89 µs	25.17 µs	0.94 µs
Data	794.25	94.25	47.25	629.25	23.5

ตารางที่ 3.1 ตารางแสดงเวลาในช่วงต่างๆของสัญญาณการสแกนทางแนวนอน
และค่าการแปลงเวลาเป็นข้อมูลที่ใช้ในการเขียนโปรแกรม

จากตารางที่ 3.1 จะแสดงช่วงค่าของ VGA Timing ของสัญญาณการสแกนทางแนวนอน ซึ่งจะใช้เวลาต่างๆ ตามช่วงที่กำหนดโดยประมาณ นอกจากนั้นเราจะแปลงค่าของเวลาเป็นค่าของข้อมูล เพื่อการใช้ในการเขียนโปรแกรมด้วย ภาษา VHDL จากค่าเวลาต่างๆจะมีการคำนวณดังนี้

A คือเวลาทั้งหมดของการสแกนทางแนวนอน

B, G และ E คือ Guard Bands

D คือเวลาการสแกนช่วง ข้อมูล 640 Pixel/แถว

ช่วงข้อมูลจะมีการคำนวณดังนี้

ความถี่ที่ใช้งานมีค่า 25.175 MHz

การสแกนแต่ละ Pixel จะใช้เวลา

Tpixel = 1/25.175MHz = 39.721 หรือประมาณ 40 ns

เวลาทั้งหมดในการสแกนในแต่ละแถว

Trow = A = B+C+D+E = (Tpixel*640) +B+C+E = 31.77 µs

ถ้าเทียบจากเวลาเป็นข้อ มูลจะมีการคำนวณดังนี้

A = 31.77 µs /40 ns = 794.25

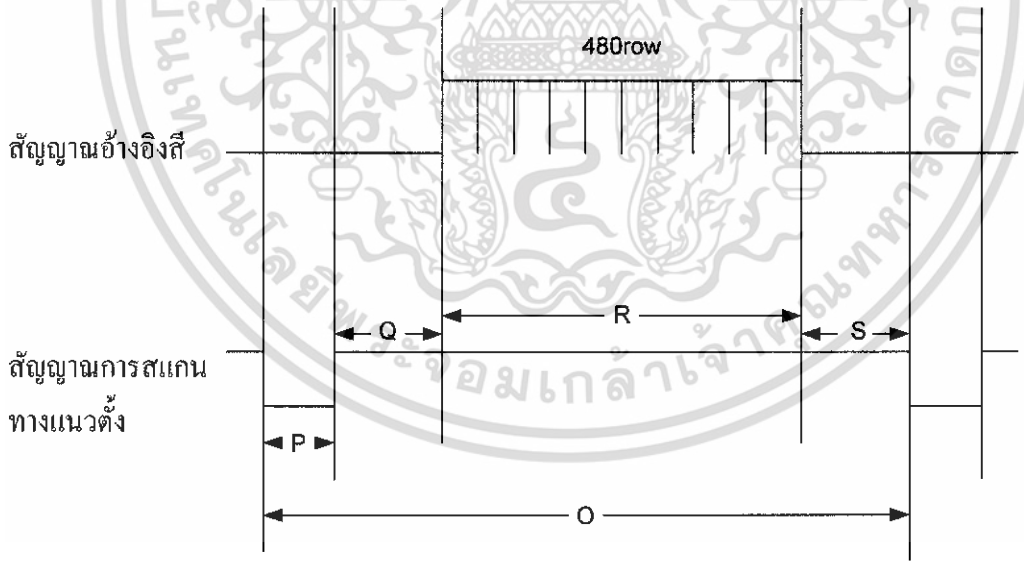
B = 3.77 µs /40 ns = 94.25

C = 1.89 µs /40 ns = 47.25

D = 25.17 µs /40 ns = 629.25

E = 0.94 µs /40 ns = 23.5

3.1.2 สัญญาณการสแกนทางแนวตั้ง (Vertical synchronization)



รูปที่ 3.2 รูปสัญญาณการสแกนและสัญญาณการอ้างอิงตำแหน่งทางแนวตั้ง

สัญญาณการสแกนทางแนวตั้งจะเป็นตัวกำหนดการสแกนทางแถวตั้งโดยถ้ามีการสแกนทางแนวนอนครบ 523 แถวจะเกิดสัญญาณการสแกนทางแนวตั้ง1ลูก คือช่วง Q+R+S นั่นเองโดยจะมีสัญญาณการอ้างอิงสีเป็นสัญญาณ บอกถึงต้องการสแกนในช่วง 0-479 ตามการสแกนมาตรฐานคือ

Parameter	O	P	Q	R	S
Time	16.6 ms	64 μs	1.02 ms	15.24 ms	0.35 ms
Data	522.505	2.014	32.105	479.697	11.016

ตารางที่ 3.2 ตารางแสดงเวลาในช่วงต่างๆของสัญญาณการสแกนทางแนวตั้ง
และค่าการแปรเวลา เป็นข้อมูลที่ใช้ในการเขียนโปรแกรม

จากตารางที่ 3.2 จะแสดงช่วงค่าของ VGA Timing ของสัญญาณการสแกนทางแนวตั้ง ซึ่งจะใช้เวลาตามช่วงที่กำหนดโดยประมาณ นอกจากนั้นเราจะแปลงค่าของเวลา เป็นค่าของข้อมูล เพื่อใช้ในการเขียนโปรแกรมด้วยภาษา VHDL จากค่าเวลาต่างๆจะมีการคำนวณดังนี้

O คือเวลาทั้งหมดของการสแกนทางแนวตั้ง

P, Q และ S คือ Guard Bands

R คือเวลาการสแกนช่วงข้อมูล 480 แถว/เฟรม คิดเป็น307, 200 พิกเซลต่อเฟรม

ช่วงข้อมูลจะมีการคำนวณและจะมีการได้สัญญาณสแกนทางแนวตั้งจะใช้เวลาดังนี้

สัญญาณการสแกนทางแนวตั้ง 1 ลูก = สัญญาณการสแกนทางแนวนอน 523 แถว

เวลาที่ใช้ในการสแกนแต่ละแถวคือ 31.77 μs

Trow = 31.77 μs

เวลาทั้งหมดในการสแกนทางแนวตั้ง

Tframe = O = P+Q+R+S = (Trow * 480rows) +P+Q+S = 16.6 ms

ถ้าเทียบจากเวลาจะได้ดังนี้

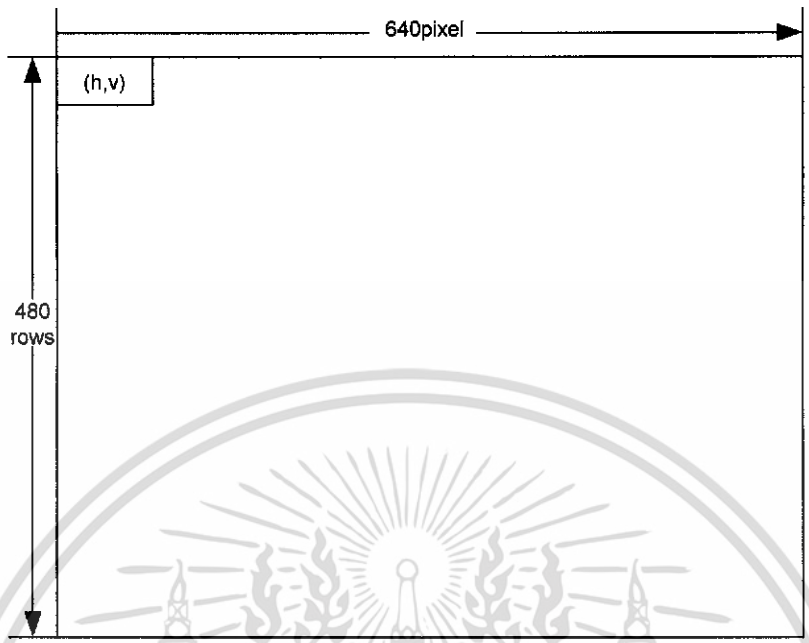
O = 16.6 ms/31.77 μs = 522.505

P = 64 μs /31.77 μs = 2.014

Q = 1.02 ms/31.77 μs = 32.105

S = 0.35 ms/31.77 μs = 11.016

3.1.3 สัญญาณอ้างอิงตำแหน่งทางแนวนอนและแนวตั้ง



รูปที่ 3.3 แสดงการอ้างอิงของจุดภาพ

จากการที่ได้กล่าวมาแล้วเราจะใช้ข้อมูลที่ได้จากการแปลงค่าจากช่วงเวลา มาสร้างสัญญาณตามที่ต้องการ ในการสร้างสัญญาณอ้างอิงตำแหน่ง จะใช้วงจรนับโดยมี Clock ตามการคำนวณของทั้งการสร้างสัญญาณทั้งทางแนวนอนและแนวตั้งคือ

ที่ ขนาดภาพ 640*480 Pixel และ refresh rate = 60 Hz

- Clock การอ้างอิงตำแหน่งทางแนวนอนใช้ความถี่ประมาณ 25.175 MHz คิดเป็นเวลาคือ ประมาณ 40 ns
- Clock การอ้างอิงตำแหน่งทางแนวตั้งจะใช้ความถี่ประมาณ 31.476 KHz คิดเป็นเวลาคือ ประมาณ 31.77 μs

ส่วนการอ้างอิงสีนั้นจะสามารถกำหนดตำแหน่งได้ให้เกิดสีปรากฏบนหน้าจอ VGA ได้โดยดูจากรูปที่ 3.3 จะมีความกว้างทางแนวนอน 640 pixel แทนด้วย h และจะมีความยาว 480 แถวแทนด้วย v ถ้าต้องการให้สีปรากฏ ที่ตำแหน่ง [h,v] เราก็จะใช่สัญญาณเป็นตัวอ้างอิงเมื่อถึงตำแหน่งที่ต้องการให้สีปรากฏออกไป ซึ่งตัวสัญญาณมีลักษณะเป็นบัส ในการอ้างอิงตำแหน่งสีซึ่งเกิดจากการนับของดีฟลิปฟลอป ซึ่งก็จะใช้สัญญาณการอ้างอิงตำแหน่งนั่นเอง แต่จะมีสัญญาณการสแกนทางแนวนอนและแนวตั้งเป็นตัวควบคุมอยู่ ดังแสดงในรูป 3.1 และ 3.2

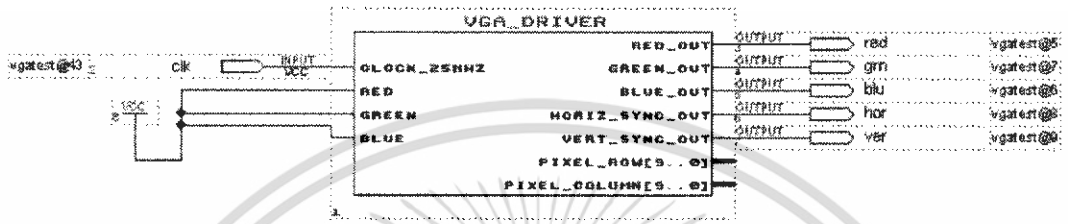
บทที่ 4

การทดลองและผลการทดลอง

ตอนที่ 4.1 ขั้นตอนการทดลอง

ตอนที่ 4.1.1 การสร้างสัญญาณซิงก์โครไนซ์และสัญญาณสี

1. ทำการทดลองโดยการจำลองการทำงานของวงจรขับการแสดงผลโดย Symbol ที่ได้จากการแปลงภาษา VHDL โดยใช้โปรแกรม Maxplus II โดยสามารถนำมาใช้งานต่อได้โดยใช้ Graphic Editor



รูปที่ 4.1 Symbol ส่วนของภาคขับสัญญาณที่ถูกเรียกใช้ใน Graphic Editor (.gdf)

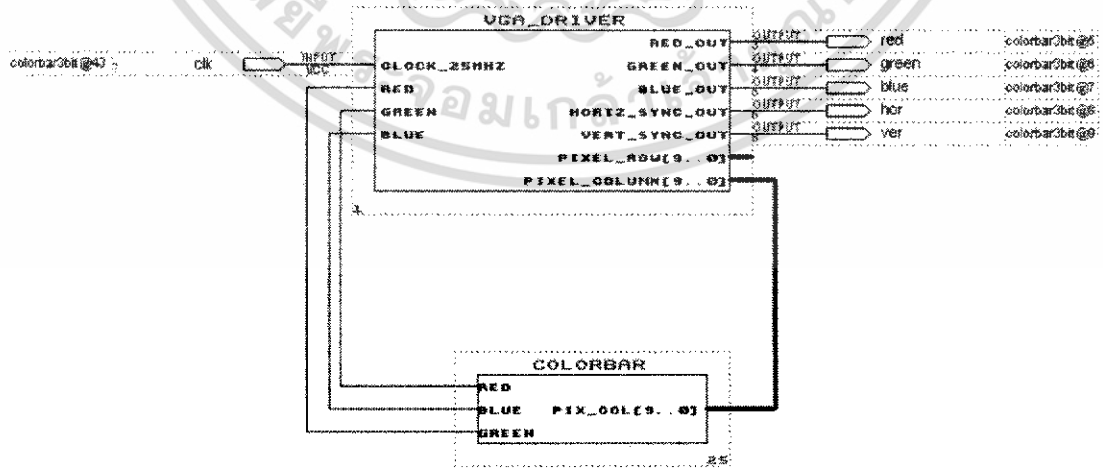
2. แล้วทำการจำลองการทำงานโดยการเรียกใช้งาน Waveform Editor (.scf) เก็บผลการจำลองของสัญญาณซิงก์และสัญญาณสี

3. นำไปโปรแกรมลงบอร์ด FPGA แล้วต่อวัดสัญญาณซิงก์และสัญญาณสี โดยใช้เครื่องออสซิลโลสโคปบันทึกผลภาพสัญญาณที่วัดออกมาได้

4. นำบอร์ดไปต่อเข้ากับคอนเนคเตอร์ภาพ แล้วต่อออกจอภาพ แล้วบันทึกผลที่ได้

ตอนที่ 4.1.2 การสร้างสัญญาณแถบสี (Color bar)

1. สร้าง color bars โดยใช้ การนับ Pixel โดย ใช้ pixel_column ในการอ้างอิงตำแหน่งสีในแนวนอน



รูปที่ 4.2 Graphic Editor (.gdf) เพื่อกำเนิด color bars บนจอแสดงผล

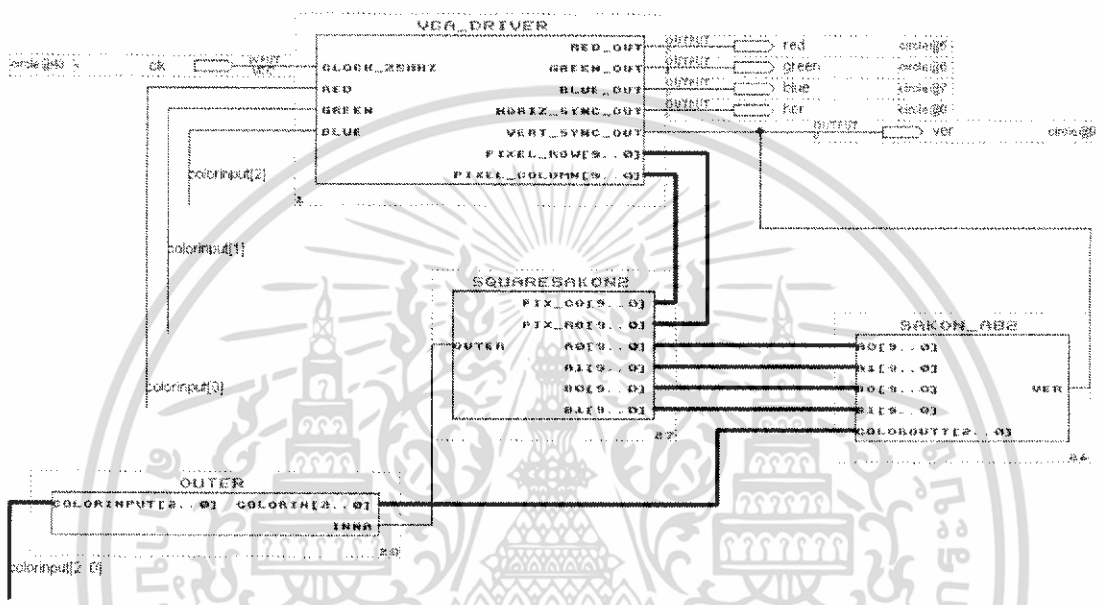
2. แล้วทำการจำลองการทำงานโดยการเรียกใช้งาน Waveform Editor (.scf) เก็บผลการจำลองของสัญญาณแถบสีที่ได้

3. นำบอร์ดไปต่อเข้ากับคอนเนคเตอร์ภาพ แล้วต่อออกจอภาพ แล้วบันทึกผลที่ได้

ตอนที่ 4.1.3 สร้างรูปสี่เหลี่ยมกระเด้งและเปลี่ยนสีเมื่อกระทบกับขอบของจอ

1. สร้างส่วนการสร้างรูปสี่เหลี่ยมและการเปลี่ยนสี

2. นำมาต่อรวมกับภาคการขับสัญญาณ

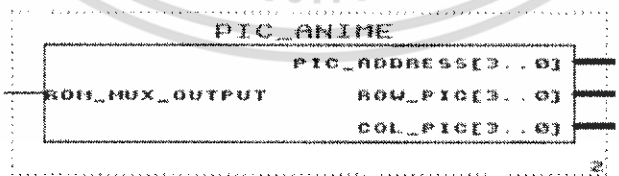


รูปที่ 4.3 Graphic Editor (.gdf) เพื่อกำเนิด สัญญาณสี่เหลี่ยมเคลื่อนที่แบบกระเด้งและเปลี่ยนสีเมื่อกระเด้งขอบจอภาพ

3. โปรแกรมลงบอร์ด FPGA บันทึกผลบนจอแสดงผล

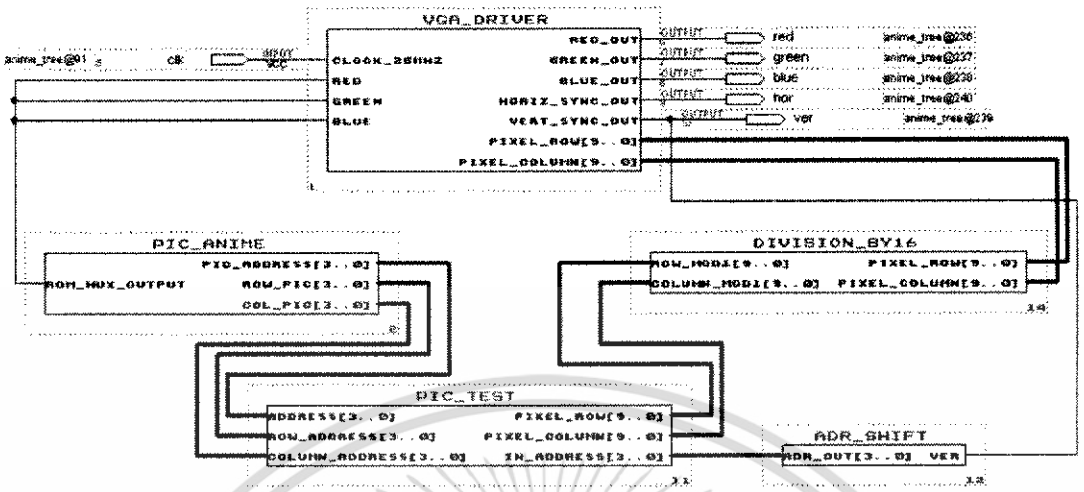
ตอนที่ 4.1.4 สร้างภาพเคลื่อนไหวแบบ Animation โดยอาศัยเมโมรี่ภายในของFPGA

1. สร้างโมดูลที่จะใช้ดึงค่าจาก EAB (Embedded Array Block) ภายในของ FPGA



รูปที่ 4.4 Symbol ของภาคการเชื่อมต่อกับ EAB ของ FPGA

2. นำมาต่อรวมกับภาคการขับสัญญาณ

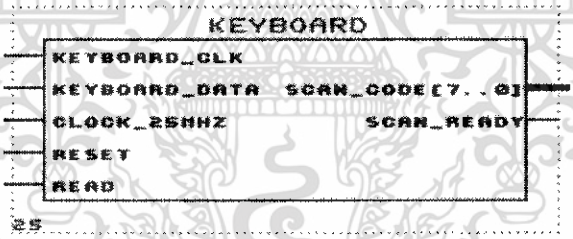


รูปที่ 4.5 Graphic Editor (.gdf) เพื่อสร้างภาพ animation บนจอ VGA

3. โปรแกรมลงบอร์ด FPGA บันทึกผลบนจอแสดงผล

ตอนที่ 4.1.5 สร้างการเชื่อมต่อกับ Keyboard โดยใช้ port PS/2

1. สร้างโมดูลการเชื่อมต่อกับ Keyboard โดยใช้พื้นฐานของการสื่อสารแบบ serial

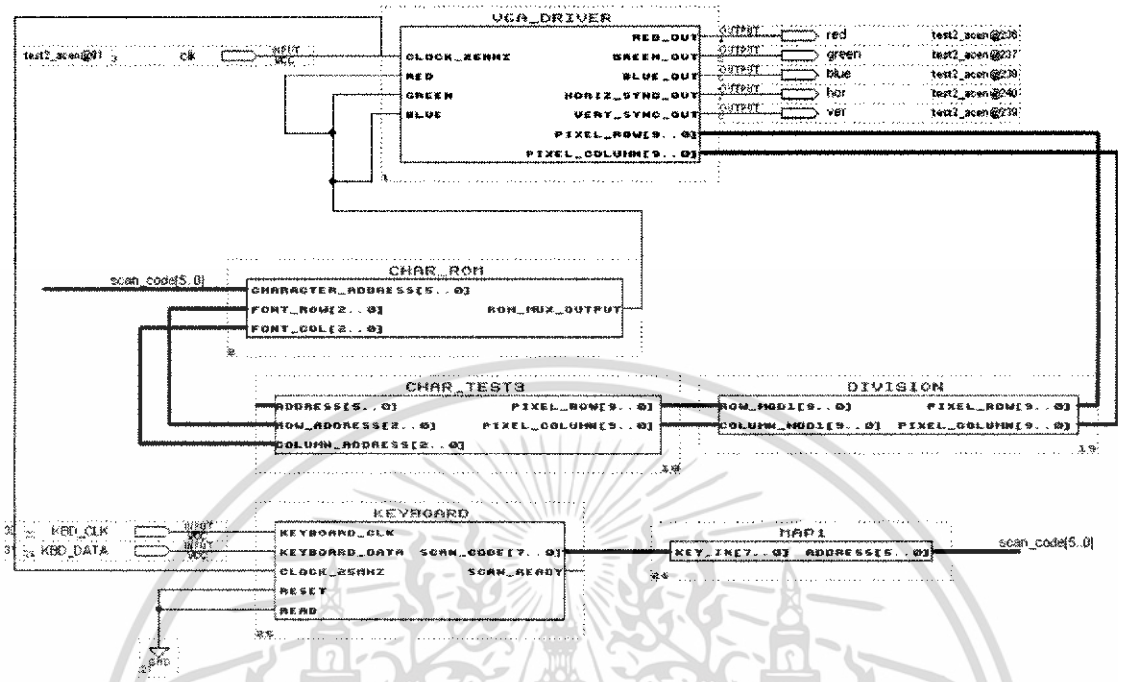


รูปที่ 4.6 โมดูลการรับค่าจาก Keyboard

2. สร้างโมดูลการ map scan_code เพื่อไปดึงค่าจาก rom ที่ออกแบบไว้แล้ว

รูปที่ 4.7 โมดูล Mapping ค่าจาก scan_code ไปดึงค่าจากรอม

3. นำมาต่อรวมกับภาคการขับสัญญาณ

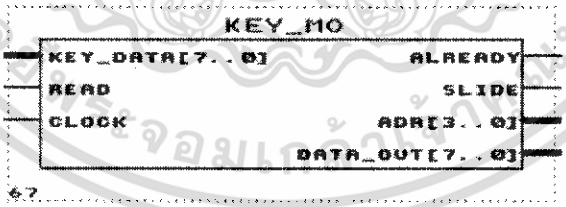


รูปที่ 4.8 Graphic Editor (.gdf) ทดลองการเชื่อมต่อ keyboard ผ่าน port PS/2

4. โปรแกรมลงบอร์ด FPGA บันทึกผลบนจอแสดงผล

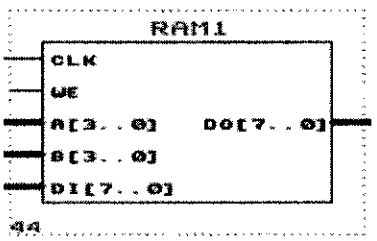
ตอนที่ 4.1.6 สร้างกลุ่มของตัวอักษรเคลื่อนไหวโดยรับค่าจาก Keyboard

1. สร้างโมดูลเพื่อวิเคราะห์ข้อมูลที่รับมาจาก Keyboard เพื่อจุดประสงค์การทำงานที่ต่างกัน (keyboard backspace or data_key)



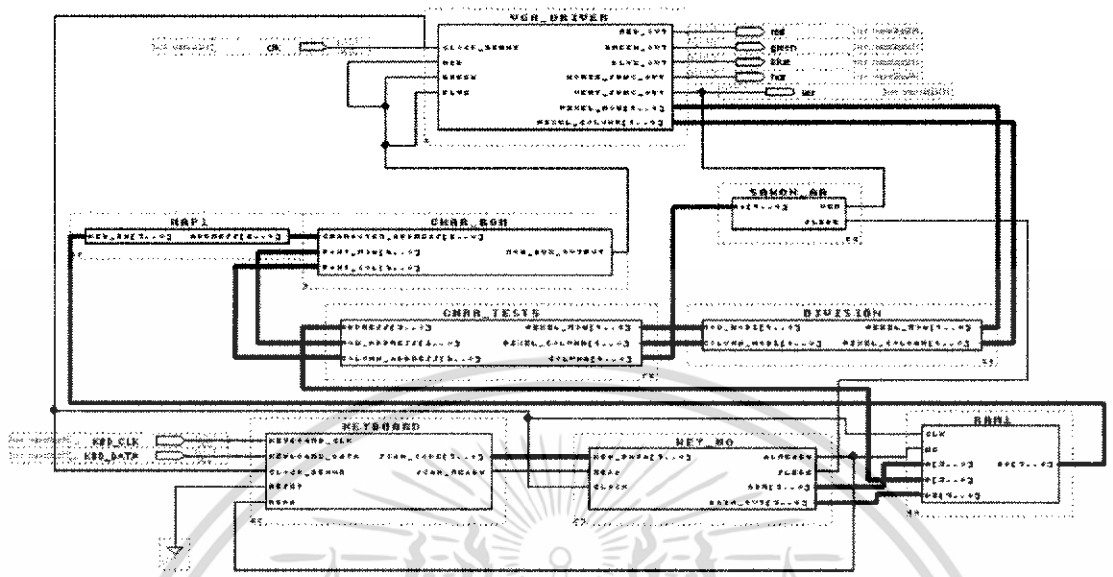
รูปที่ 4.9 โมดูลวิเคราะห์ข้อมูลจาก Keyboard

2.สร้างโมดูล RAM เพื่อเก็บค่าcharacter ที่ keyboard ส่งมาให้



รูปที่ 4.10 โมดูลเก็บค่าจาก Keyboard เพื่อรอการเรียกไปแสดงผล

3. นำมาต่อรวมกับภาคการขับสัญญาณ

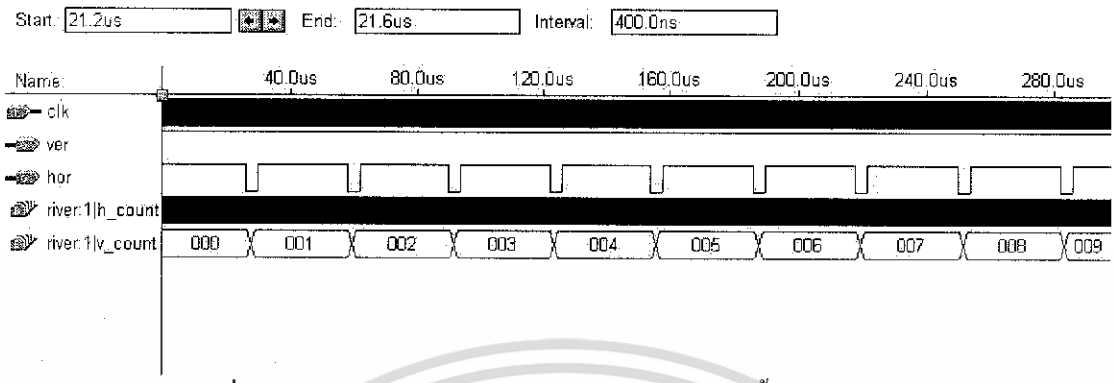


รูปที่ 4.11 Graphic Editor (.gdf) สร้างกลุ่มของตัวอักษรเคลื่อนไหวโดยรับค่าจาก keyboard

4. โปรแกรมลงบอร์ด FPGA บันทึกผลบนจอแสดงผล

ตอนที่ 4.2 ผลการทดลอง

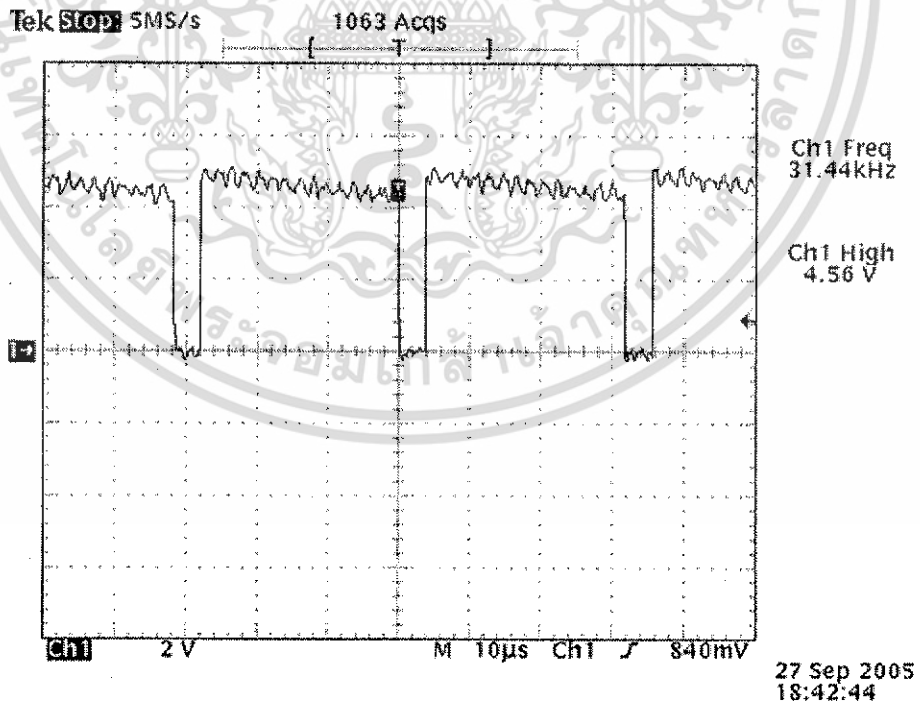
ผลการทดลองตอนที่ 4.1.1 การสร้างสัญญาณซิงก์โครไนซ์และสัญญาณสี



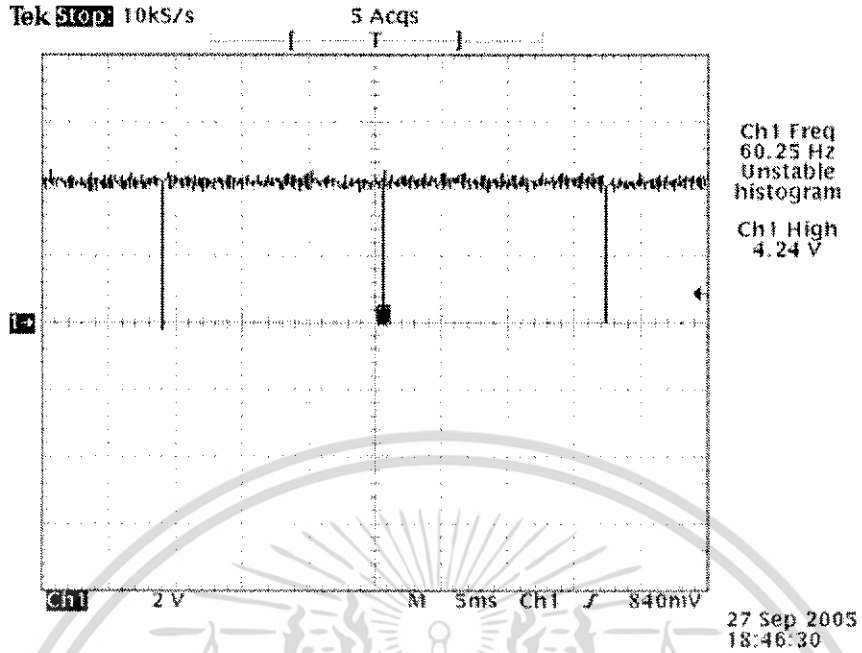
รูปที่ 4.12 แสดงผลการจำลองสัญญาณอ้างอิงทางแนวตั้งและแนวนอน

จากรูปที่ 4.12 แสดงสัญญาณโดยสัญญาณสแกนทางแนวนอนหรือ horizontal synchronization โดยให้ active low ที่ช่วงข้อมูล 659-755 และให้เป็น active high ในช่วง 0-659 และช่วง 755-799 ของ h_counter จะได้ความถี่ออกมาประมาณ 32 us ส่วนต่อไป คือ vertical synchronization โดยให้ active low ที่ช่วงข้อมูล 493-494 และให้เป็น active high ในช่วง 0-493 และช่วง 494-524 ของ v_counter

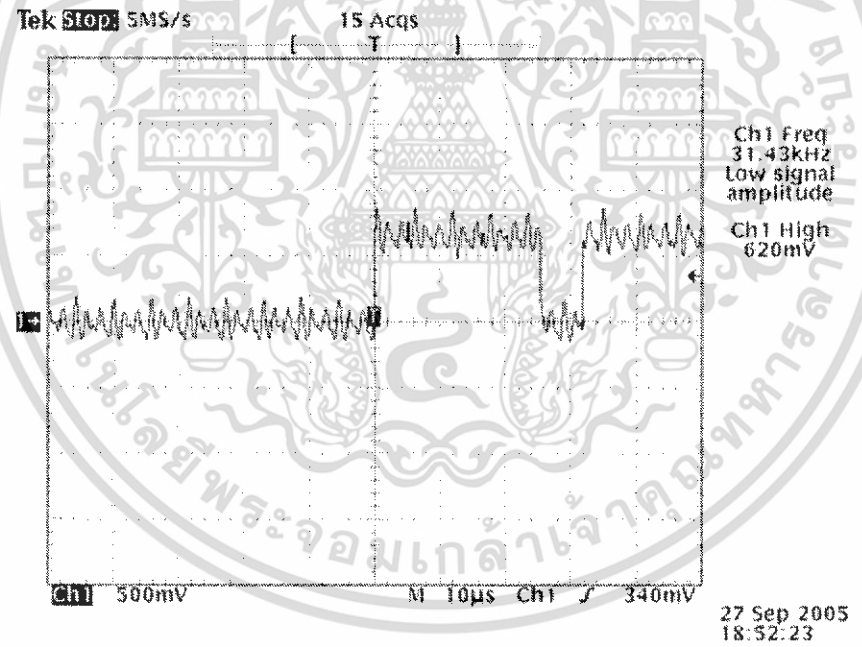
วัดสัญญาณจริงที่ได้จาก โดยใช้ Oscilloscope มาวัดสัญญาณ horizontal synchronization และสัญญาณ vertical synchronization



รูปที่ 4.13 สัญญาณ Horizontal synchronization



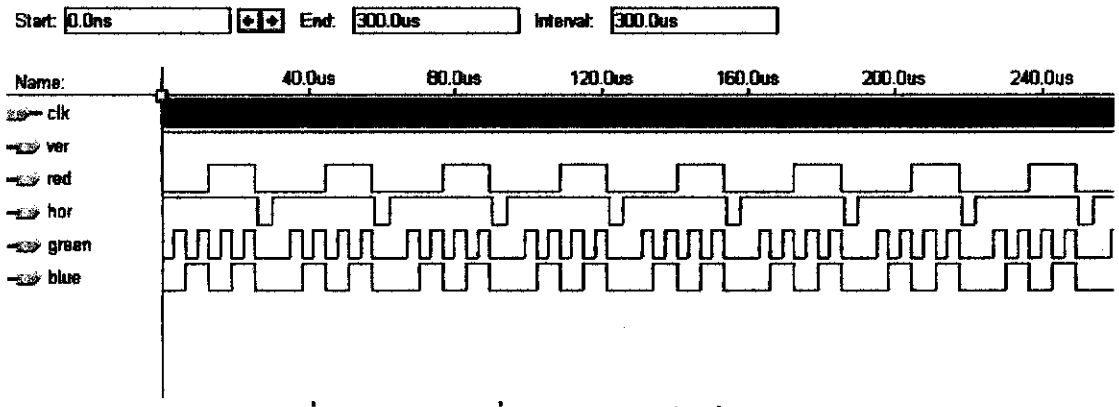
รูปที่ 4.14 สัญญาณ Vertical synchronization



รูปที่ 4.15 สัญญาณสีที่วัดได้จากออสซิลโลสโคป

<u>ผลการทดลองตอนที่ 4.1.2</u> การสร้างสัญญาณแถบสี

การขับภาพออกนั้นจะใช้การสร้างสัญญาณ Video โดยการนับช่วงข้อมูลในช่วง 0-639 จะได้เป็นจำนวน Pixel ที่เท่ากับที่เราต้องการคือ 640*480 แล้วนำมาแอนด์กับสัญญาณที่ต้องการขับออก จะได้ผลการจำลองดังนี้



รูปที่ 4.16 สัญญาณสีที่ถูกขับออกมาสร้างเป็น Color bar

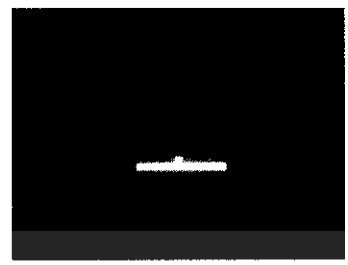
รูปที่ 4.17 หน้าจอแสดงผลเป็น Color bars (สี3 bits)

ผลการทดลองตอนที่ 4.1.3 เพิ่มการเปลี่ยนการเปลี่ยนสีเมื่อกระเด้งกับขอบของจอภาพ

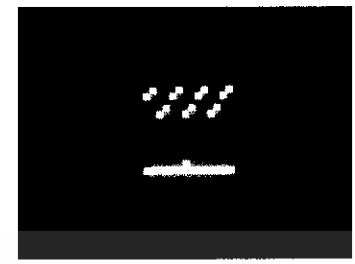
รูปที่ 4.18 การเปลี่ยนการเปลี่ยนสีเมื่อรูปสี่เหลี่ยมกระเด้งกับขอบของจอภาพ

ผลการทดลองถ่ายด้วยกล้องที่ความเร็วชัตเตอร์ 10 วินาที ในภาพเป็นรูปสี่เหลี่ยมเคลื่อนที่แบบ Bouncing มีการเปลี่ยนสีเมื่อกระเด้งกับขอบของจอ

ผลการทดลองตอนที่ 4.1.4 สร้างภาพเคลื่อนไหวแบบ Animation โดยอาศัยหน่วยความจำภายในของFPGA



(ก)



(ข)

(ค)

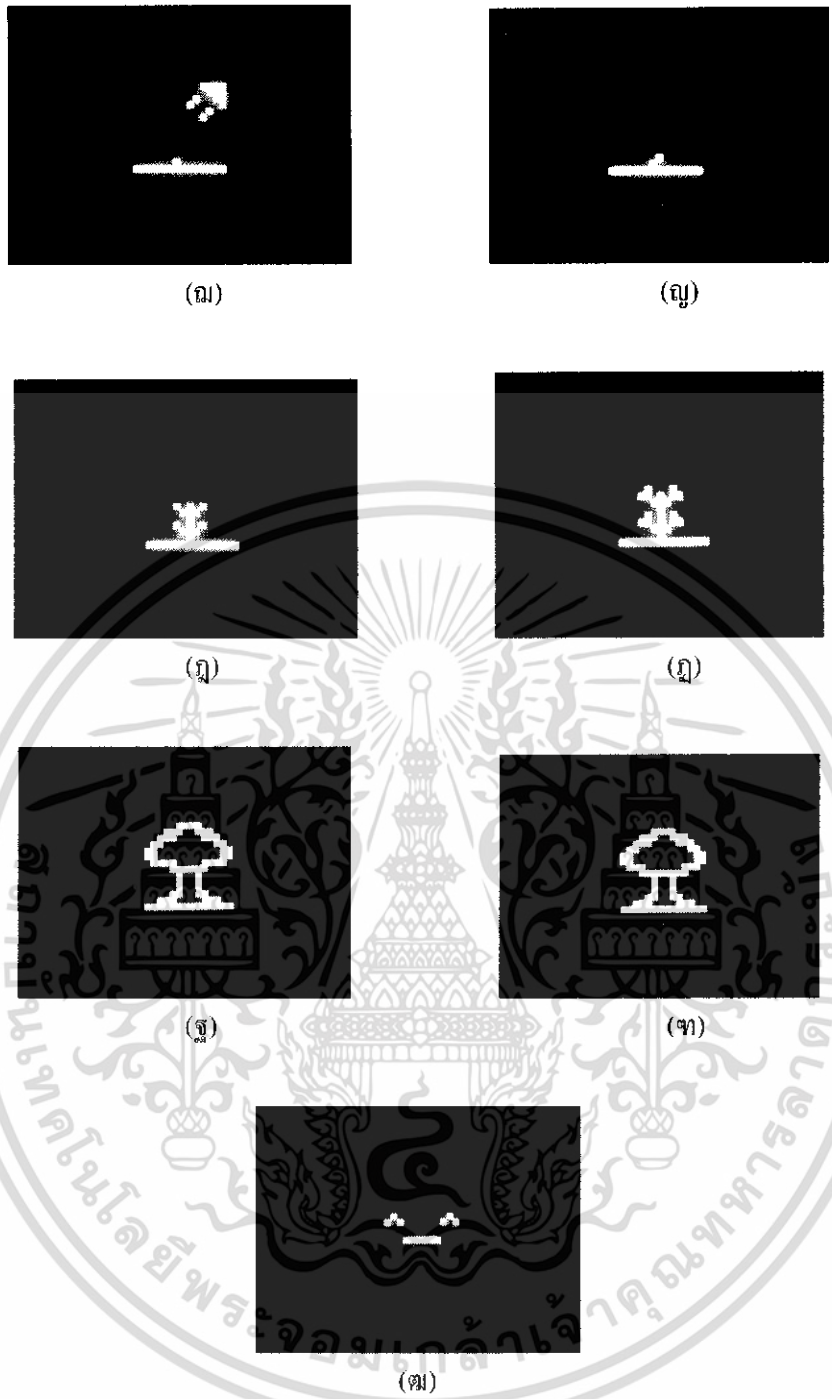
(ง)

(จ)

(ฉ)

(ช)

(ซ)



(ฌ) (ญ)

(ฎ) (ฏ)

(ฐ) (ฑ)

(ฒ)

รูปที่ 4.19 ภาพของแต่ละเฟรมที่ใช้แสดงเป็นภาพเคลื่อนไหว

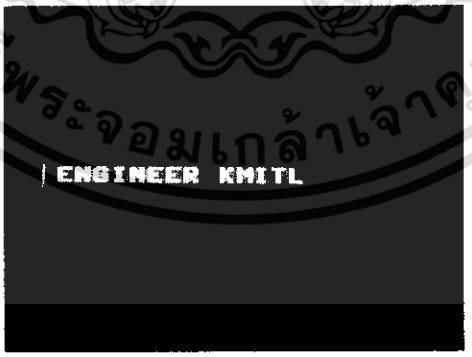
ผลการทดลองตอนที่ 4.1.5 สร้างการเชื่อมต่อกับ Keyboard โดยใช้ port PS/2

รูปที่ 4.20 ตำแหน่งเคอร์เซอร์ของตัวอักษรที่จะปรากฏขึ้น

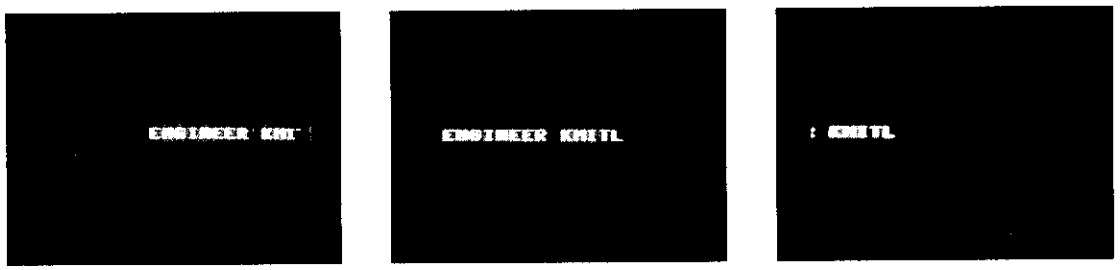
(ค) (ต)

รูปที่ 4.21 ตัวอย่างตัวอักษรที่พิมพ์
โดยที่ รูปที่ 4.21 (ค) พิมพ์ตัวอักษร A
รูปที่ 4.21 (ต) พิมพ์ตัวอักษร I

ผลการทดลองตอนที่ 4.1.6 สร้างกลุ่มของตัวอักษรเคลื่อนไหวโดยรับค่าจาก Keyboard



รูปที่ 4.22 เริ่มจากการพิมพ์ข้อความจากคีย์บอร์ดลงไป



(ต) (ถ) (ท)

รูปที่ 4.23 แสดงทิศทางข้อความที่เคลื่อนที่จากขวามือไปยังซ้ายมือ

บทที่ 5
บทวิจารณ์และบทสรุปผล

5.1 สรุปผลการทดลอง

โครงงานนี้นำเสนอการควบคุมการแสดงผลบนจอภาพ VGA โดยใช้ FPGA ซึ่งได้ศึกษาเรียนรู้การควบคุมจอภาพ และออกแบบการทดลองที่สามารถอธิบายวิธีการสร้างภาพแบบพื้นฐานและตัวอักขระต่างๆ, การกำหนดสีที่แสดงผลได้ในระดับ 3 bits, การกำหนดเฟรมการแสดงผลภาพเพื่อทำภาพเคลื่อนไหวในลักษณะเฟรมต่อเฟรม และการต่ออินพุตรับค่ามาจากคีย์บอร์ด

5.2 ปัญหาของโครงงาน

1. ในการสร้างภาพเคลื่อนไหวให้มีรายละเอียดมากขึ้นก็จำเป็นที่จะต้องการหน่วยความจำที่มากขึ้นตามด้วย สำหรับโครงงานนี้เนื่องจากการจำกัดของหน่วยความจำของอุปกรณ์ที่ใช้ จึงสร้างภาพเคลื่อนไหวได้ไม่ละเอียดและมีขนาดไม่ใหญ่มากนัก

2. ในการสร้างอักษรภาษาอื่นๆ ก็จำเป็นที่จะต้องมีหน่วยความจำโปรแกรมที่มากตามด้วยสำหรับโครงงานนี้จึงสร้างได้แต่อักษรภาษาอังกฤษและอักขระพิเศษเท่านั้น

5.3 แนวทางการพัฒนา

โครงงานนี้สามารถพัฒนาต่อไปได้อีกมากมาย นอกจากการสร้างภาพเคลื่อนไหว และโมษณาแล้ว ถ้าสามารถต่อหน่วยความจำเพิ่มได้มากพอ ก็สามารถนำไปพัฒนาเป็นเกมได้ โดยนอกจากการต่อคีย์บอร์ดแล้ว ยังสามารถต่อกับเมาส์ หรือเกมคอนโทรลเลอร์ หรืออาจจะรับภาพจากกล้องแล้วมาประมวลผลเก็บลงในหน่วยความจำ แล้วแสดงผลออกมา โดยที่เราไม่จำเป็นจะต้องมีคอมพิวเตอร์ประมวลผล และนอกจากนี้ยังสามารถพัฒนาได้ต่อด้วยการนำตัวอุปกรณ์ D/A แบบความเร็วสูงมาต่อเพื่อเพิ่มระดับสีเพื่อให้สามารถแสดงผลในระดับสีที่มากขึ้นและมีรายละเอียดมากขึ้นได้

เอกสารอ้างอิง

1. Bernard Grob; Basic Television and Video Systems 5th Edition, McGRAW HILL International Edition 1988

2. John Watkinson; Television Fundamentals 1st Edition 1996

3. Gerry Kane; CRT Controller Handbook

4. John Watkinson; An introduction to digital video 2nd Edition 2001, Genesis

5. ทฤษฎีเอฟพีจีเอและการใช้งานโปรแกรม MAX Plus II; www.astronlogic.com

6. ชำนาญ ปัญญาใส, วัชราภร หนูทอง; ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล

7. ePanorama.net