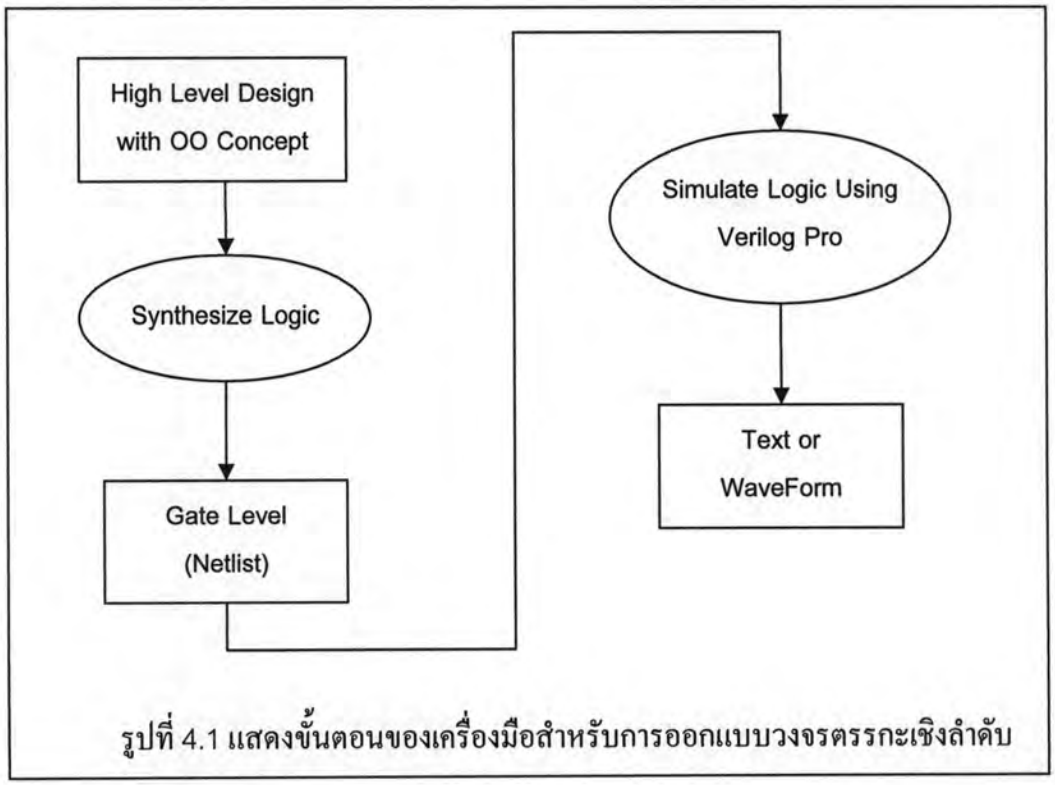
บทที่ 4
ขั้นตอนการออกแบบเครื่องมือ
สำหรับการออกแบบวงจรตรรกะเชิงลำดับโดยใช้แนวคิดเชิงวัตถุ

เนื่องจากขั้นตอนการทำงานของเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงลำดับ โดยใช้แนวคิดเชิงวัตถุนี้นั้น ได้แก่ 1) การออกแบบวงจรตรรกะเชิงลำดับแบบกราฟิกโดยใช้แนวคิดเชิงวัตถุ 2) การสังเคราะห์วงจร และ 3) การจำลองการทำงานซึ่งจะเชื่อมต่อกับเครื่องมือจำลองการทำงาน Verilogger Pro ดังแสดงขั้นตอนการทำงานของเครื่องมือออกแบบวงจรตรรกะเชิงลำดับทั้งหมดดังรูปที่ 4.1



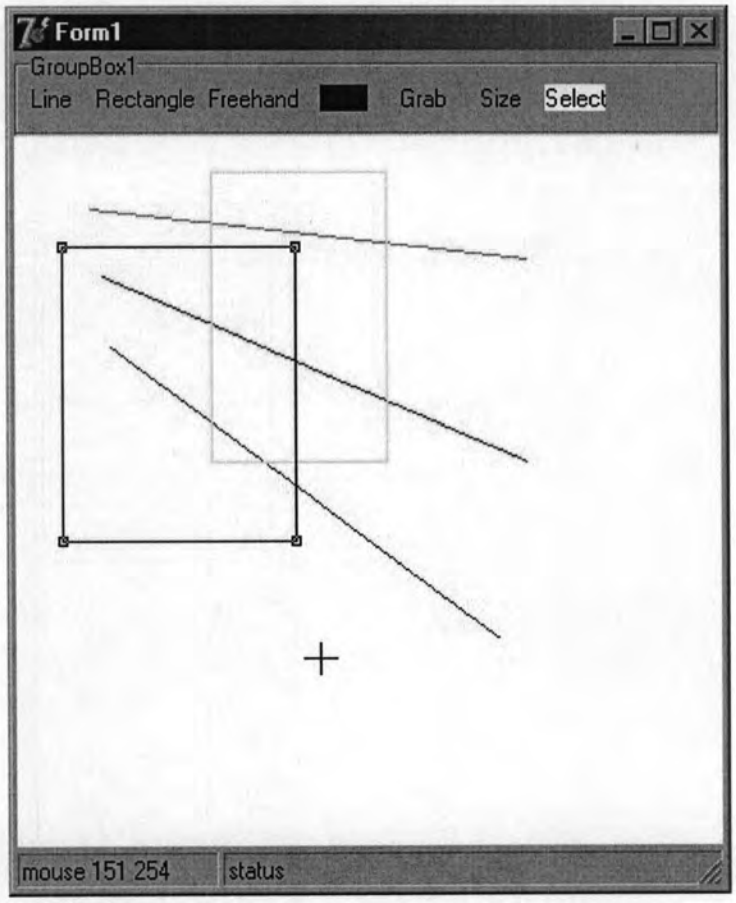
รูปที่ 4.1 แสดงขั้นตอนของเครื่องมือสำหรับการออกแบบวงจรตรรกะเชิงลำดับ

แต่ละขั้นตอนการทำงานของเครื่องมือ มีขั้นการออกแบบดังต่อไปนี้

ส่วนออกแบบวงจรตรรกะเชิงลำดับแบบกราฟิกและการสังเคราะห์วงจร

การพัฒนาเครื่องการออกแบบวงจรตรรกะเชิงลำดับ โดยใช้แนวคิดเชิงวัตถุแบบกราฟิกนี้ เนื่องจากได้ทำการคัดเลือกเครื่องมือพัฒนาโปรแกรมมาแล้ว ได้แก่ Borland Delphi 7.0 ดังนั้นผู้วิจัยจึงได้ค้นหาเครื่องมืออื่นๆ ที่ช่วยให้สามารถพัฒนาเครื่องมือออกแบบวงจรตรรกะเชิงลำดับโดยใช้แนวคิดเชิงวัตถุแบบกราฟิกได้รวดเร็วยิ่งขึ้นตามขั้นตอนดังนี้

1) ทำการค้นหาไลบรารีของ Borland Delphi 7.0 ที่ใช้ในการพัฒนาเครื่องมือดังกล่าว ที่เกี่ยวข้องกับการพัฒนาเครื่องมือแบบกราฟิก ได้แก่ Graphic Editor (Ing.Büro R.Tschaggelar, 2003) ซึ่งมีความสามารถในการจัดการกับรูปร่างทางกราฟิกต่างๆ กัน และยังใช้แนวคิดเชิงวัตถุในการพัฒนาไลบรารีดังกล่าว ข้อดีคือไลบรารีนี้ให้ซอร์สโคดมาทั้งหมด และเป็นไลบรารีที่มีขนาดเล็กที่สามารถนำไปใช้ได้เลย แต่มีข้อเสียคือ สามารถนำไปพัฒนาต่อได้ยากและใช้เวลานาน เพราะมีการพัฒนาโปรแกรมกราฟิกแบบพื้นฐาน เมื่อนำมาทดลองพัฒนาต่อแล้ว พบว่าต้องใช้เวลาพัฒนาต่อค่อนข้างนาน สำหรับตัวอย่างไลบรารีนี้จะแสดงในรูปที่ 4.2



รูปที่ 4.2 แสดงตัวอย่างไลบรารี Graphic Editor

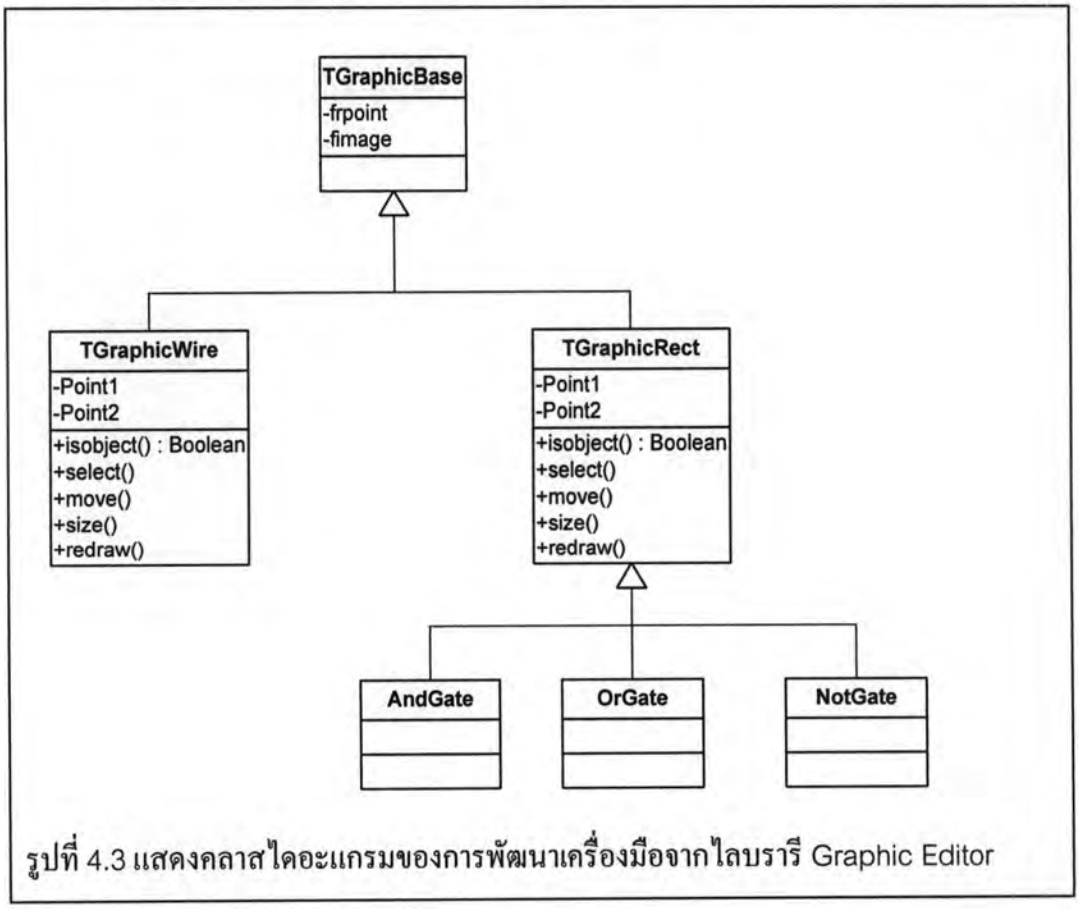
แนวคิดเชิงวัตถุในการพัฒนาโดยใช้ไลบรารี Graphic Editor สามารถอธิบายโดยใช้คลาสไดอะแกรมได้ดังนี้

1.1) คลาสตั้งต้นของทุกอุปกรณ์คือคลาส TGraphicBase ซึ่งมีข้อมูลคือจุดที่ตั้ง (frpoint) และรูปที่วาด (fimage)

1.2) ถ่ายทอดคลาส TGraphicBase ให้กับคลาส TGraphicWire ซึ่งเป็นคลาสของสายเชื่อมต่อและคลาส TGraphicRect ซึ่งเป็นคลาสตั้งต้นของเกทต่างๆ ทั้งสองคลาสนี้มีข้อมูลและพฤติกรรมคล้ายกันได้แก่ ข้อมูลจุดที่ตั้งที่หนึ่ง (Point1) และจุดที่ตั้งที่สอง(Point2) และพฤติกรรมตรวจสอบวัตถุ(isobject) เลือก(select) ย้าย(move) เปลี่ยนขนาด(size) และวาดใหม่(redraw)

1.3) ถ่ายทอดคลาส TGraphicRect ให้กับคลาสของเกทต่างๆ

คลาสต่างๆ สามารถแสดงไดอะแกรมได้ดังรูปที่ 4.3

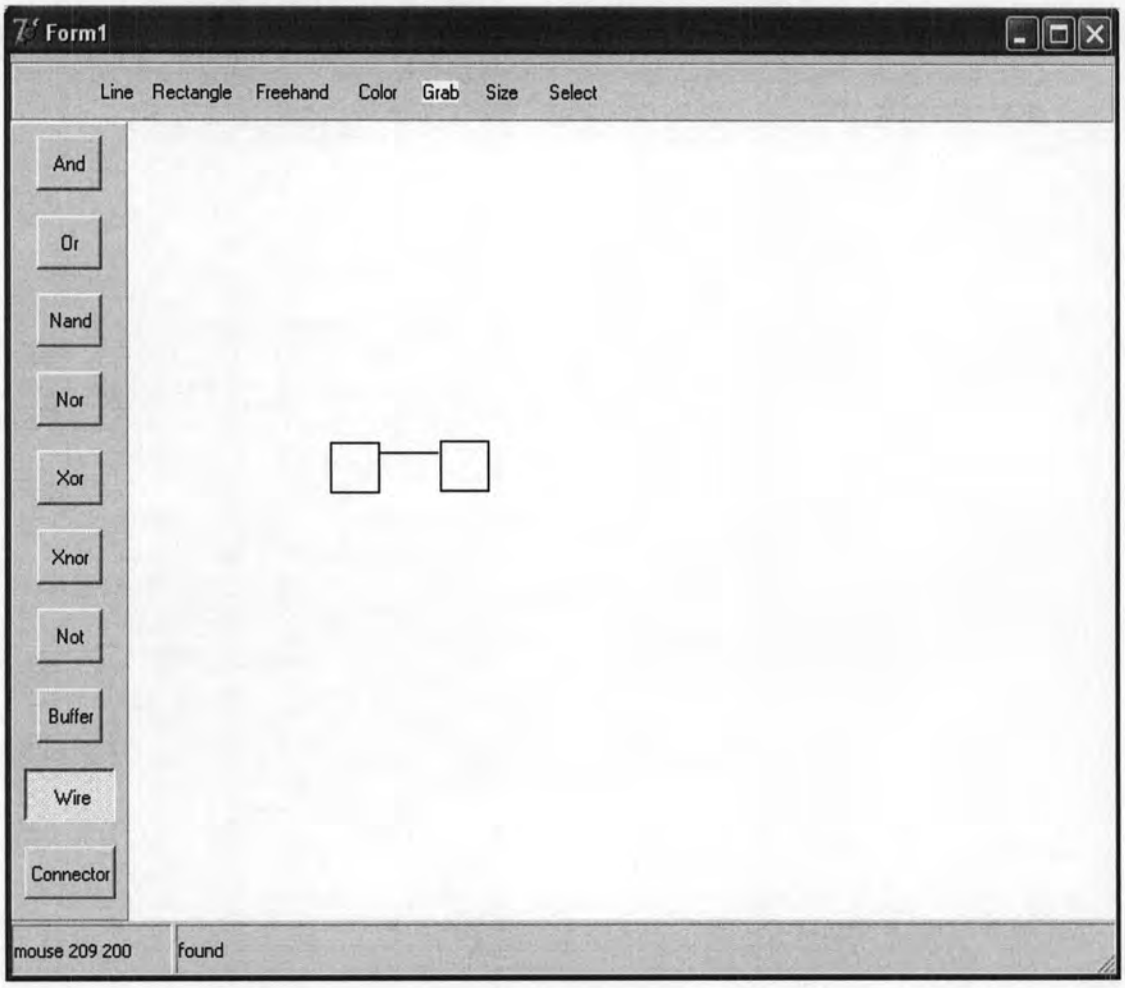


รูปที่ 4.3 แสดงคลาสไดอะแกรมของการพัฒนาเครื่องมือจากไลบรารี Graphic Editor

ตัวอย่างจากการพัฒนาไลบรารี เพื่อสร้างเครื่องมือสำหรับออกแบบวงจรตรรกะเชิงลำดับ โดยใช้แนวคิดเชิงวัตถุแบบกราฟิกจะแสดงในรูปที่ 4.4

2) ทำการค้นหาไลบรารีอื่นๆ ที่สามารถใช้งานร่วมกับ Borland Delphi 7.0 ได้ ได้แก่ LEADTOOLS® Raster Imaging ซึ่งจากการศึกษาและการทดลองใช้เครื่องมือนี้แล้วพบว่า

ไลบรารีนี้เน้นไปในการสร้างงานด้านกราฟิกขั้น Bitmap มากกว่า และค่อนข้างเน้นไปในงานกราฟิกแบบละเอียด ลักษณะบางอย่างที่อาจจะนำมาพัฒนาต่อได้ เช่น Layer ก็พบว่าเหมาะกับงานที่เน้นกราฟิกชนิดโฟโตชอปมากกว่า คำสั่งต่างๆ ล้วนใกล้เคียงกับคำสั่งที่มีอยู่ใน Borland Delphi 7.0 อยู่แล้ว รูปที่ 4.5 แสดงตัวอย่างคำสั่งที่ใช้งานของไลบรารี LEADTOOLS® VCL



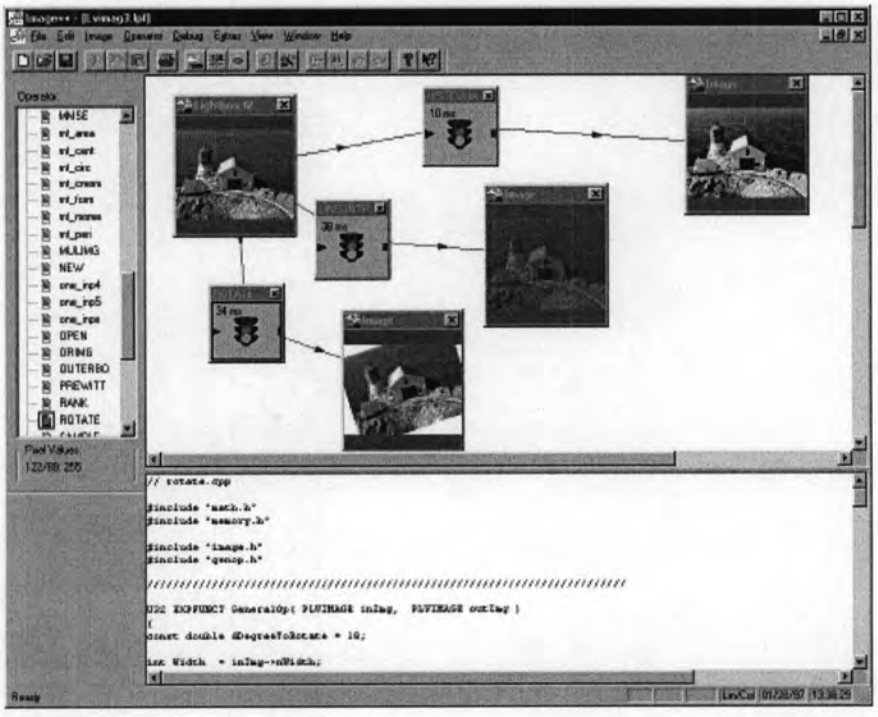
รูปที่ 4.4 แสดงตัวอย่างการพัฒนาไลบรารี Graphic Editor
เพื่อสร้างเครื่องมือออกแบบวงจรตรรกะเชิงลำดับ

3) ทดลองใช้วิธีในการสร้างรูปกราฟิกต่างๆ จากความสามารถของ Borland Delphi 7.0 เอง เช่น การใช้รูปเกทต่างๆ จากไฟล์ แทนการใช้คำสั่งที่เกี่ยวกับกราฟิกวาดรูปเกทลงบนฟอร์มเอง และใช้กลุ่มฟังก์ชันของ BitmapList เพื่อเก็บรูปเกทต่างๆ แทนการวาดโดยใช้คำสั่ง แต่ภาพที่เกิดขึ้นจะไม่มีความสวยงาม หรือใช้ไลบรารี Image++ (Image Integration, 2006) ที่ช่วยในการพัฒนาเครื่องมือกราฟิกเชิงวัตถุ (Object-oriented graphic development tool) แต่ต้องใช้ร่วมกับ

เครื่องมือพัฒนาโปรแกรมภาษา C++ และภาพที่เกิดขึ้นจะไม่มีความสวยงามเช่นกัน รูปที่ 4.6 แสดงตัวอย่างการใช้ไลบรารี Image++ พัฒนาเครื่องมือกราฟิกเชิงวัตถุ

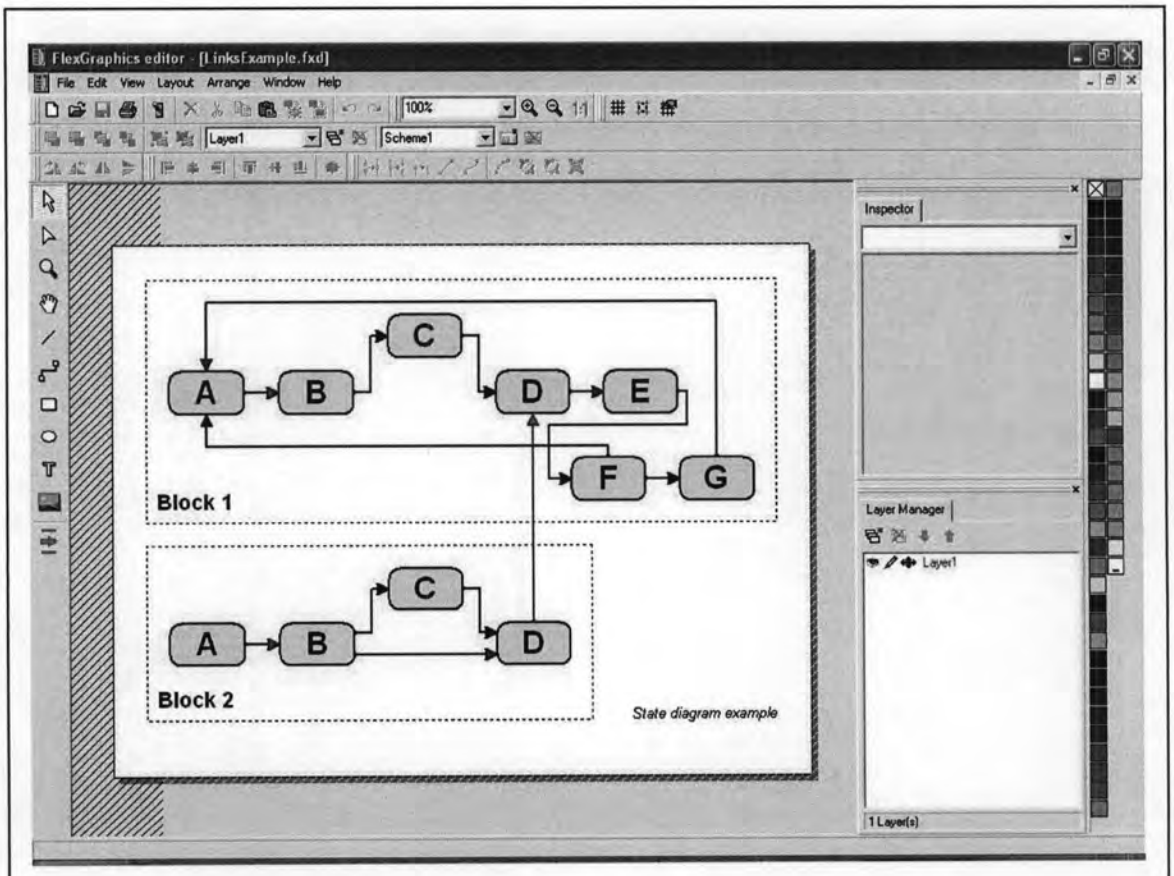
```
52  procedure TForm1.FormShow(Sender: TObject);
53  begin
54   Lead1.BorderStyle := bsSingle;
55   Lead1.BackColor := RGB(255, 255, 125);
56   Lead1.PaintDither := pdDiffusion;
57   Lead1.PaintPalette := ppAuto;
58   Lead1.AutoRepaint := True;
59   Lead1.AutoSize := False;
60   Lead1.AutoSetRects := True;
61   Lead1.PaintSizeMode := smFit;
62   DrawObject := 0;
63   Drawing := False;
64  end;
```

รูปที่ 4.5 แสดงตัวอย่างคำสั่งที่ใช้งานของไลบรารี LEADTOOLS® VCL



รูปที่ 4.6 แสดงตัวอย่างการใช้ไลบรารี Image++

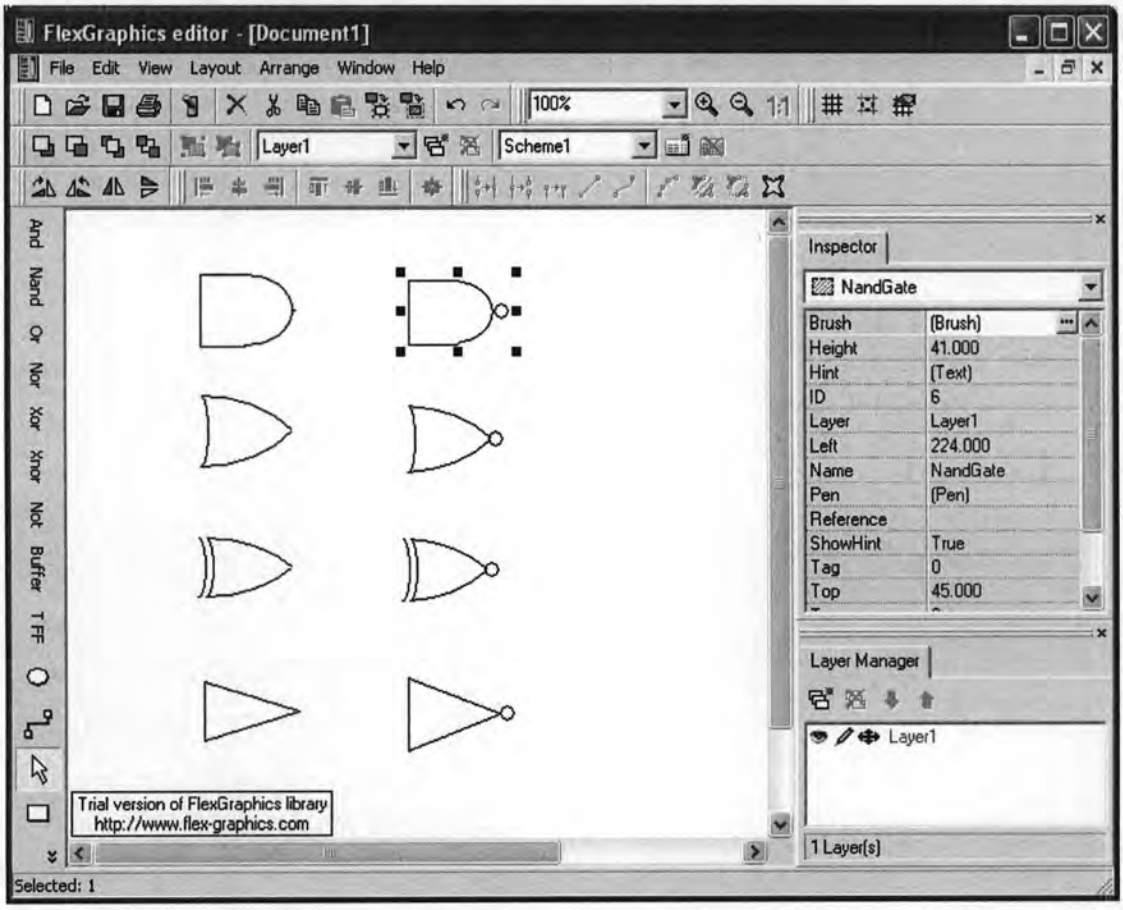
4) ค้นหาไลบรารีอื่นๆ โดยใช้คำที่ระบุเจาะจงมากขึ้นเช่น Library ,Delphi, Vector graphic ซึ่งคำว่า Vector นี่เอง ทำให้สามารถค้นพบไลบรารีที่สำคัญมาชื่อว่า FlexGraphics for Borland Delphi/C++Builder (FlexGraphics Software, 2003) ซึ่งในการใช้งานไลบรารีนี้ ในระบบต้องทำการลงไลบรารีชนิดอื่นก่อน ได้แก่ RX Libary v2.75 (with support Delphi 6, 7) (FlexGraphics Software, 2003) ,GridView Library (Roman Mochalov) และ ToolBar2000 (Jordan Russell, 2000) ไลบรารีนี้ พบว่ามีความสามารถในการจัดการรูปภาพชนิดต่างๆ แบบเวกเตอร์ได้ดี จึงนำไลบรารีนี้มาใช้ในงานวิจัย แต่ข้อเสียของไลบรารีชนิดนี้คือ จะต้องลงไลบรารีไว้ที่คอมพิวเตอร์ทุกเครื่องที่ต้องการใช้งาน ไม่สามารถรันโปรแกรมให้ทำงานแบบโดดๆ ได้ ดังเช่นโปรแกรมเดลไฟทั่วไป และนอกจากนั้นยังไม่ได้ให้ซอร์สโคดมาทั้งหมด และมีการพัฒนาไลบรารีที่ค่อนข้างซับซ้อน จึงค่อนข้างพัฒนาต่อยากพอสมควร รูปที่ 4.7 แสดงตัวอย่างการใช้ไลบรารีชนิดนี้



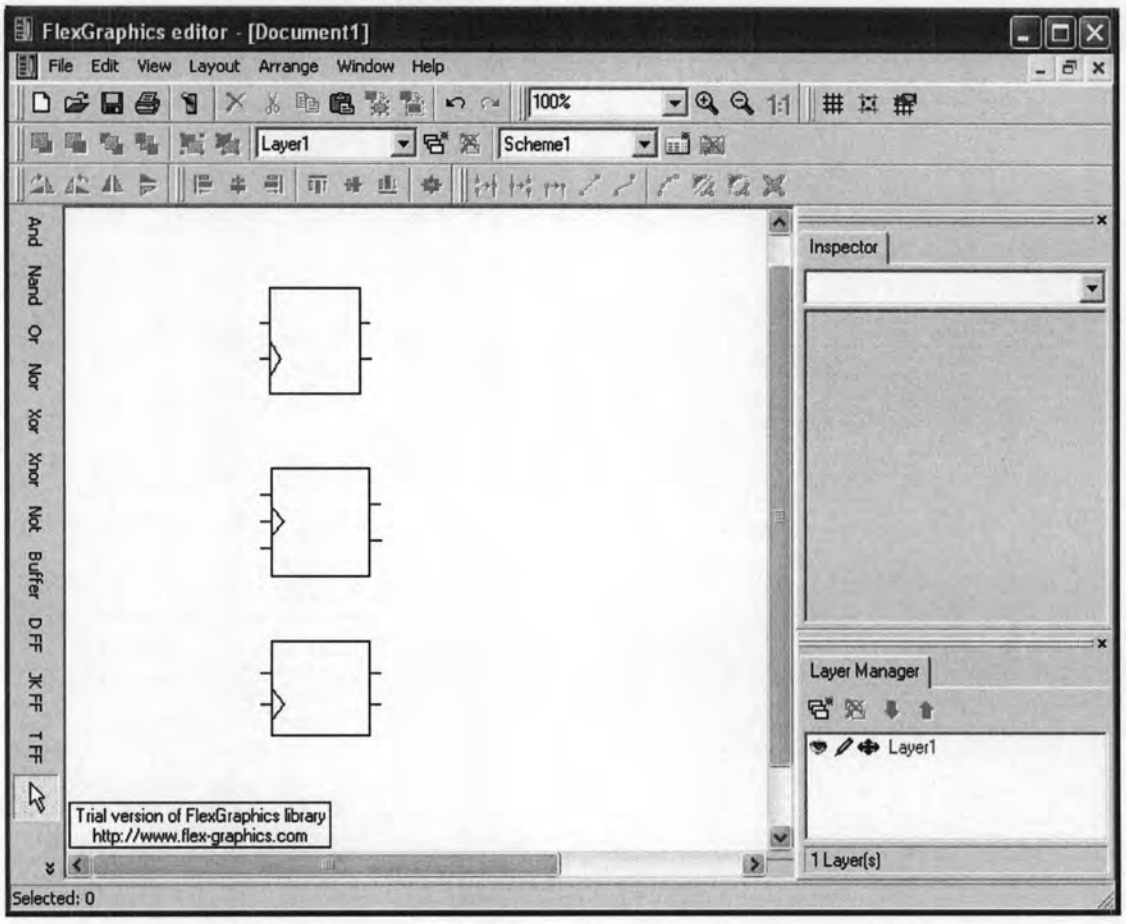
รูปที่ 4.7 แสดงตัวอย่างการใช้ไลบรารี FlexGraphics

เมื่อพบว่าไลบรารี FlexGraphics สามารถช่วยในการพัฒนาเครื่องมือออกแบบวงจรตรรกะเชิงลำดับโดยใช้แนวคิดเชิงวัตถุในส่วนกราฟิกได้เป็นอย่างดีแล้ว จึงเริ่มศึกษาวิธีการใช้ไลบรารีเบื้องต้นและทดลองการใช้การวาดรูปเกทต่างๆ แทนรูปในไลบรารี และพัฒนาเครื่องมือออกแบบวงจรตรรกะเชิงลำดับในส่วนของกราฟิกต่อไปดังนี้

4.1 สร้างการวาดกราฟิกอุปกรณ์เกทและฟลิปฟลอบต่างๆ จากคำสั่งพื้นฐานของเดลไฟและสมการคณิตศาสตร์ ได้ดังรูปที่ 4.8-4.9 วิธีการสร้างคือคลิกเลือกที่อุปกรณ์ที่ต้องการหนึ่งครั้ง และกดคลิกที่พื้นที่ส่วนวาดกราฟิกค้างไว้แล้วลากจนได้ขนาดของอุปกรณ์ตามที่ต้องการ หากต้องการเปลี่ยนขนาด ลบ ย้ายที่อุปกรณ์ต่างๆ ให้คลิกเลือกที่ลูกศรชี้ แล้วจึงมาคลิกที่มุมใดมุมหนึ่งของอุปกรณ์ หรือคลิกค้างไว้แล้วลากให้ครอบคลุมอุปกรณ์นั้นๆ แล้วจึงทำการเปลี่ยนขนาด ลบ หรือย้ายอุปกรณ์ตามความต้องการ และหากต้องการตั้งชื่อให้อุปกรณ์ใดๆ ให้ใส่ชื่ออุปกรณ์ที่ช่อง Name ที่ฟอร์ม Inspector ด้านขวา

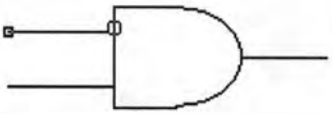


รูปที่ 4.8 แสดงการวาดเกททั้งแปดชนิด



รูปที่ 4.9 แสดงการวาดฟลิปฟลอปทั้ง 3 ชนิด

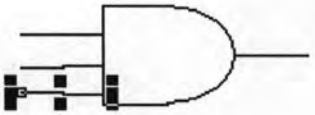
4.2 ทดลองเพิ่มการเชื่อมต่อระหว่างอุปกรณ์กับสายเชื่อมต่อ โดยหลังจากการวาดอุปกรณ์ลงบนพื้นที่กราฟิกแล้ว หากต้องการสร้างสายเชื่อมต่อระหว่างอุปกรณ์หรือสร้างสายเชื่อมต่อให้อุปกรณ์ใดๆ แล้ว ให้คลิกเลือกที่อุปกรณ์สายเชื่อมต่อ  แล้วคลิกที่จุดตั้งต้นของจุดที่จะเชื่อมต่อและลากไปยังจุดที่ต้องการเชื่อมต่อต่อไป หากต้องการลากสายสัญญาณอินพุตของอุปกรณ์ใดๆ ให้จุดตั้งต้นของสายสัญญาณอยู่ที่จุดใดๆ และลากสายสัญญาณเข้าหาด้านอินพุตของอุปกรณ์นั้นๆ จนขึ้นรูปวงกลมสีฟ้าบริเวณจุดที่ต้องการเชื่อมต่อดังรูปที่ 4.10 หากต้องการลากสายสัญญาณเอาต์พุตของอุปกรณ์ใดๆ ให้ลากจุดตั้งต้นจากด้านเอาต์พุตของอุปกรณ์นั้นๆ ซึ่งจะขึ้นรูปวงกลมสีฟ้าบริเวณจุดที่ต้องการเชื่อมต่อด้านเอาต์พุตและลากสายสัญญาณไปยังจุดใดๆ และเช่นเดียวกับอุปกรณ์อื่นๆ หากต้องการตั้งชื่อให้สายเชื่อมต่อใดๆ ก็สามารถใส่ชื่อได้ที่ช่อง Name ในฟอร์ม Inspector



รูปที่ 4.10 แสดงการเชื่อมต่อสายให้อุปกรณ์แอนด์เกท

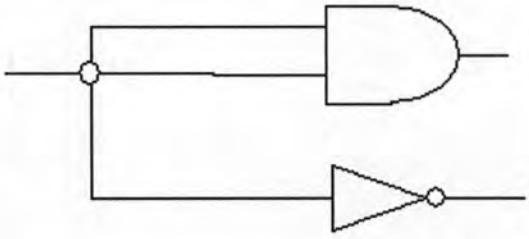
4.3 เพิ่มจุดรับอินพุตให้อุปกรณ์เกทต่างๆ ทั้งหกชนิดและบังคับให้อินพุตของฟลิปฟลอปทั้งสามชนิด และเอาต์พุตของเกทและฟลิปฟลอปทั้งหมดให้มีจำนวนที่ต้องการ

4.4 ทดลองการเพิ่มจุดรับอินพุตให้เกทต่างๆ ทั้งหกชนิดดังตัวอย่างในรูปที่ 4.11



รูปที่ 4.11 แสดงการเชื่อมต่อสายสัญญาณเพิ่มเป็นแอนด์สามอินพุต

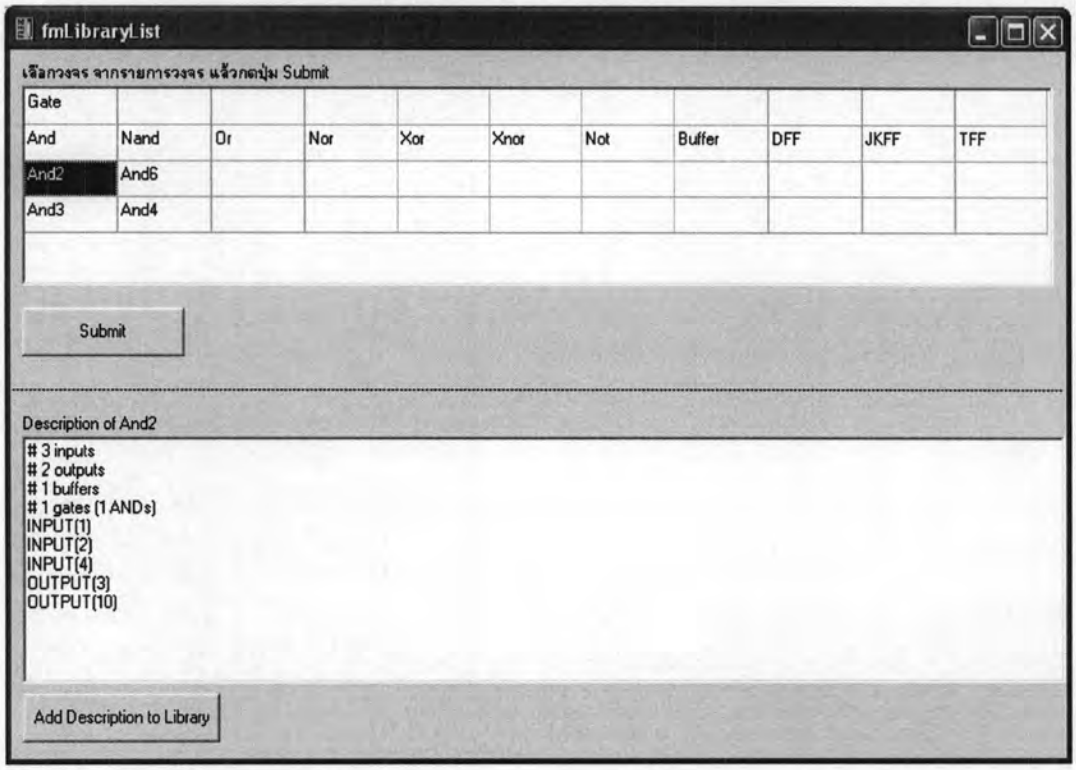
4.5 สร้างการวาดกราฟิกจุดเชื่อมต่อ โดยให้จุดเชื่อมต่อสามารถมีสายสัญญาณอินพุตได้เพียงหนึ่งสายสัญญาณ และสามารถมีสัญญาณเอาต์พุตเพื่อเชื่อมต่อกับอุปกรณ์อื่นได้จากหนึ่งถึงสามสายสัญญาณ และทดสอบการเชื่อมต่อดังรูปที่ 4.12



รูปที่ 4.12 แสดงการเชื่อมต่อสายสัญญาณ

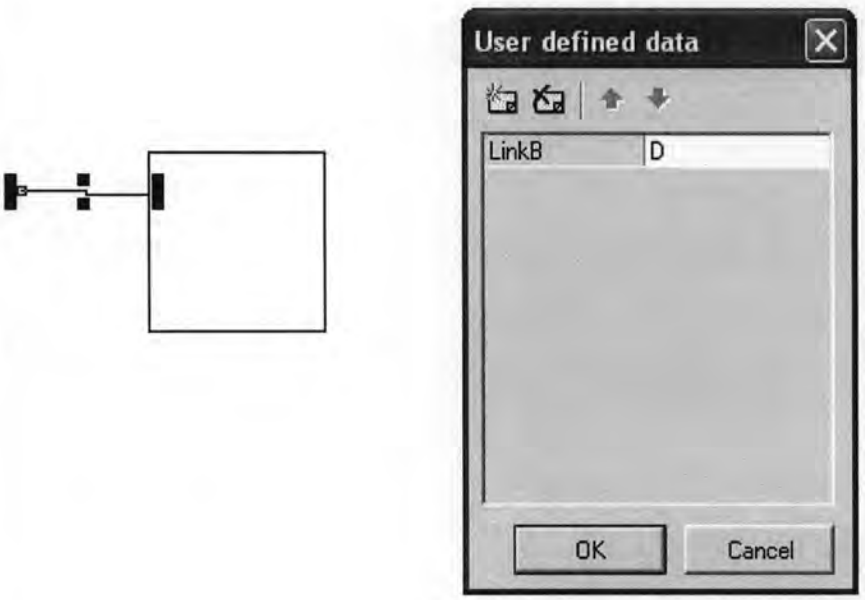
4.6 อ่านผลลัพธ์จากการสร้างอุปกรณ์ต่างๆ เพื่อนำมาสังเคราะห์วงจร และสร้างการสังเคราะห์วงจรจากวงจรง่ายๆ โดยการสร้างไฟล์เน็ตลิสต์

4.7 สร้างการทำวงจรบล็อก โดยให้เริ่มจากการเลือกวงจรบล็อกจากตารางการถ่ายทอด และวิเคราะห์อินพุต เอาต์พุตของวงจรจากไฟล์เน็ตลิสต์ ดังรูปที่ 4.13



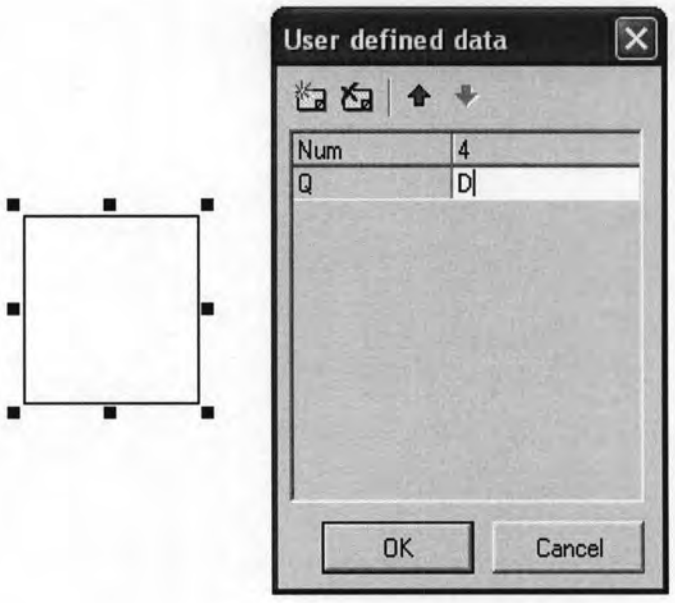
รูปที่ 4.13 แสดงการเลือกวงจรจากไลบรารีเพื่อสร้างวงจรบล็อก

4.8 เพิ่มการเชื่อมต่อสายสัญญาณจากภายนอกเข้าสู่วงจรบล็อกโดยการกำหนดที่คุณสมบัติพิเศษคือ UserData ที่อยู่ในฟอร์ม Inspector ดังแสดงในรูปที่ 4.14 วิธีการกำหนดคุณสมบัติพิเศษนี้คือ ให้คลิกที่รูป ... ด้านขวาของคุณสมบัติ UserData แล้วจะมีกล่องโต้ตอบ User defined data ขึ้นมา ให้คลิก เพื่อเพิ่มคุณสมบัติการเชื่อมต่อสาย หากจุดปลายของการเชื่อมต่อสายสัญญาณเชื่อมต่อกับด้านอินพุตกับอุปกรณ์จากไลบรารี ที่กล่อง User defined data ที่ช่องซ้ายมือให้พิมพ์คำว่า LinkB และช่องขวามือให้พิมพ์ชื่อสัญญาณอินพุตของไลบรารีที่ต้องการ หากจุดตั้งต้นการเชื่อมต่อของสายสัญญาณเชื่อมต่อกับอุปกรณ์จากไลบรารี ที่กล่อง User defined data ที่ช่องซ้ายมือให้พิมพ์คำว่า LinkA แทน และเพิ่มการวิเคราะห์วงจรบล็อกและเขียนลงในสมการเน็ตลิสต์



รูปที่ 4.14 แสดงการกำหนดการเชื่อมต่อสายสัญญาณกับอุปกรณ์จากไลบรารี

4.9 สร้างวงจรบล็อกชนิดที่มากกว่า 1 วงจรในขณะเดียวกัน การกำหนดจำนวนวงจรให้ใช้คุณสมบัติพิเศษ UseraData ที่อยู่ในฟอร์ม Inspector คลิกเพิ่มคุณสมบัติ ที่กล่อง User defined data ที่ช่องซ้ายมือให้พิมพ์คำว่า Num ช่องขวามือให้พิมพ์จำนวนวงจรบล็อกที่ต้องการ ส่วนการเชื่อมต่อกันระหว่างวงจรแต่ละวงจรให้กำหนดโดยใช้ UserData ที่อยู่ในฟอร์ม Inspector คลิกเพิ่มคุณสมบัติ ที่กล่อง User defined data ที่ช่องซ้ายมือให้พิมพ์ชื่อสายสัญญาณเอาต์พุตที่เชื่อมต่อภายในวงจรบล็อก ช่องขวามือให้พิมพ์ชื่อสายสัญญาณอินพุตที่จะถูกเชื่อมต่อภายในวงจรบล็อก ดังรูปที่ 4.15 และเพิ่มการวิเคราะห์วงจรบล็อกและเขียนลงในสมการเน็ตลิสต์โดยเพิ่มชื่อสายสัญญาณของวงจรบล็อกให้เท่ากับจำนวนวงจรบล็อก เช่น วงจรบล็อกมีจำนวน 3 วงจรบล็อก แต่ละวงจรบล็อกมีชื่อสายสัญญาณอินพุต 2 สายสัญญาณ ได้แก่สายสัญญาณ D และสายสัญญาณ Clock สายสัญญาณเอาต์พุต 1 สายสัญญาณได้แก่สายสัญญาณ Q การวิเคราะห์วงจรและเขียนลงสมการเน็ตลิสต์ จะทำการเพิ่มชื่อสายสัญญาณของวงจรบล็อกให้เป็น D_1, D_2, D_3, Clock_1, Clock_2, Clock_3, Q_1, Q_2 Q_3 และเพิ่มสายสัญญาณของวงจรบล็อกภายในทั้งหมด เป็นต้น ผลที่ได้ คือการเชื่อมต่อกันระหว่างวงจรบล็อกทั้งหมด 3 วงจร (การสร้างวงจรบล็อกที่มีวงจรภายในมากกว่าหนึ่งวงจรต้องมีการเชื่อมต่อกันระหว่างวงจรภายในแบบคงที่)



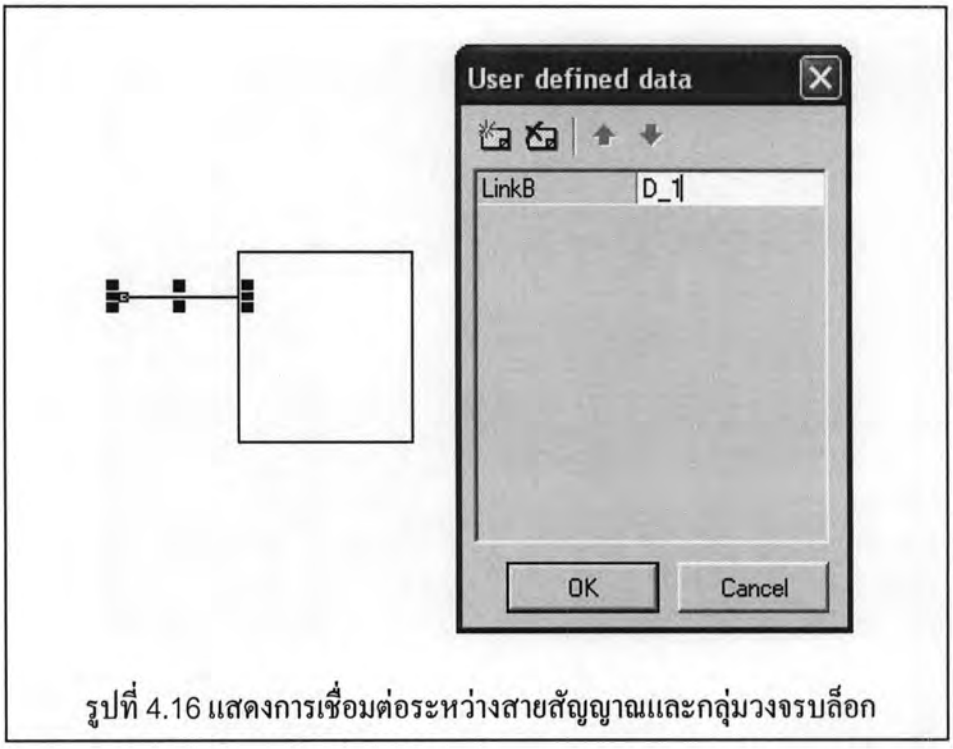
รูปที่ 4.15 แสดงการกรอกจำนวนวงจรบล็อกและการต่อกันภายในวงจรบล็อก

4.10 การเชื่อมต่อสายสัญญาณกับวงจรบล็อกที่ประกอบด้วยหลายๆ วงจรภายในให้ใช้คุณสมบัติพิเศษ UserData ที่อยู่ในฟอร์ม Inspector คลิกเพิ่มคุณสมบัติที่กล่อง User defined data ที่ช่องซ้ายมือให้พิมพ์คำว่า LinkA หากเป็นการเชื่อมต่อสายสัญญาณที่ด้านเอาต์พุตของวงจรบล็อกหรือพิมพ์คำว่า LinkB หากเป็นการเชื่อมต่อสายสัญญาณที่ด้านอินพุตของวงจรบล็อก และที่ช่องขวามือให้พิมพ์ชื่อสายสัญญาณตามรูปแบบ ชื่อสาย_ลำดับของวงจร ตามที่ต้องการดังรูปที่ 4.16 และเพิ่มการวิเคราะห์และเขียนสมการเน็ตลิสต์

4.11 สร้างการตรวจสอบวงจรแบบต่างๆ ดังนี้

4.11.1 ตรวจสอบการตั้งชื่ออุปกรณ์ซึ่งต้องไม่ซ้ำกันทุกชนิด จะยกเว้นการตรวจสอบสายสัญญาณที่ต่อออกจากด้านเอาต์พุตของจุดเชื่อมต่อ

4.11.2 จุดเชื่อมต่อใดๆ จะต้องไม่มีสายสัญญาณต่อเข้าด้านอินพุตของจุดเชื่อมต่อมากกว่า 1 สาย คือสายสัญญาณที่มีการเชื่อมต่อไปยังด้านอินพุตของจุดเชื่อมต่อใดๆ จะต้องไม่เป็นสายสัญญาณเส้นเดียวกันหรือหมายเลขเดียวกัน ดังนั้นจะไม่เกิดเหตุการณ์อุปกรณ์ต่อเอาต์พุตชนกัน



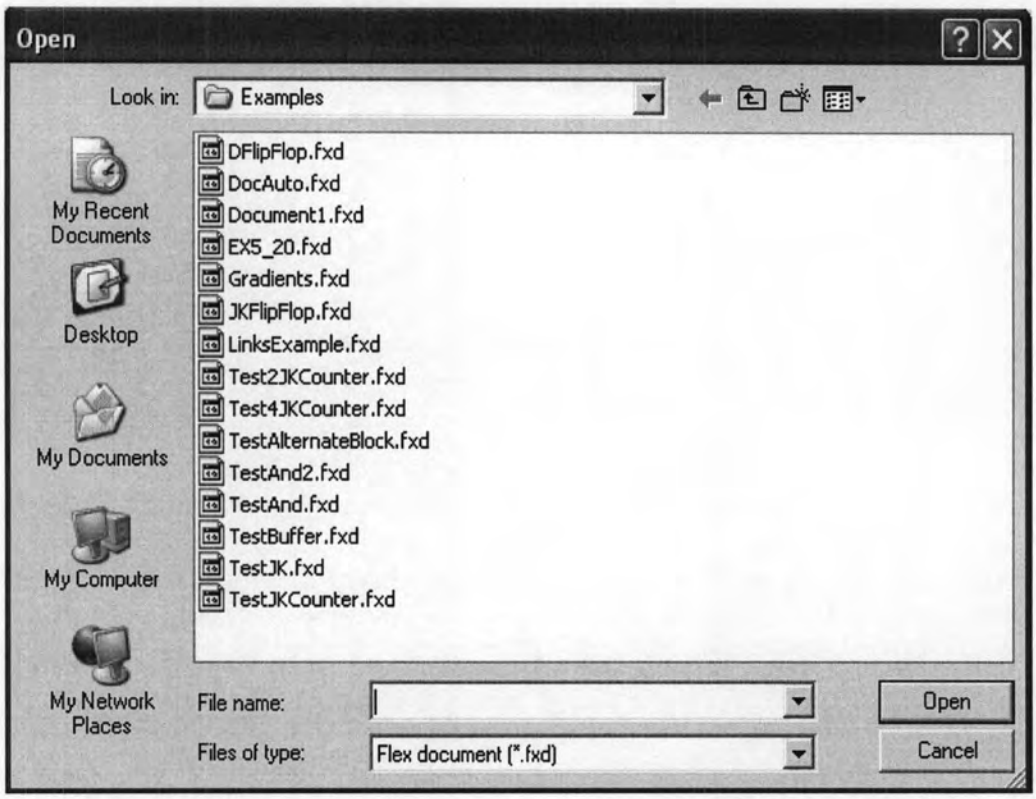
รูปที่ 4.16 แสดงการเชื่อมต่อระหว่างสายสัญญาณและกลุ่มวงจรบล็อก

4.11.3 จุดเชื่อมต่อใดๆ จะต้องไม่วนลูปเข้าหาตัวเอง คือ จุดเชื่อมต่อใดๆ สายสัญญาณด้านอินพุตของจุดเชื่อมต่อจะต้องไม่เป็นสายสัญญาณเส้นเดียวกับสายสัญญาณด้านเอาต์พุตของจุดเชื่อมต่อ

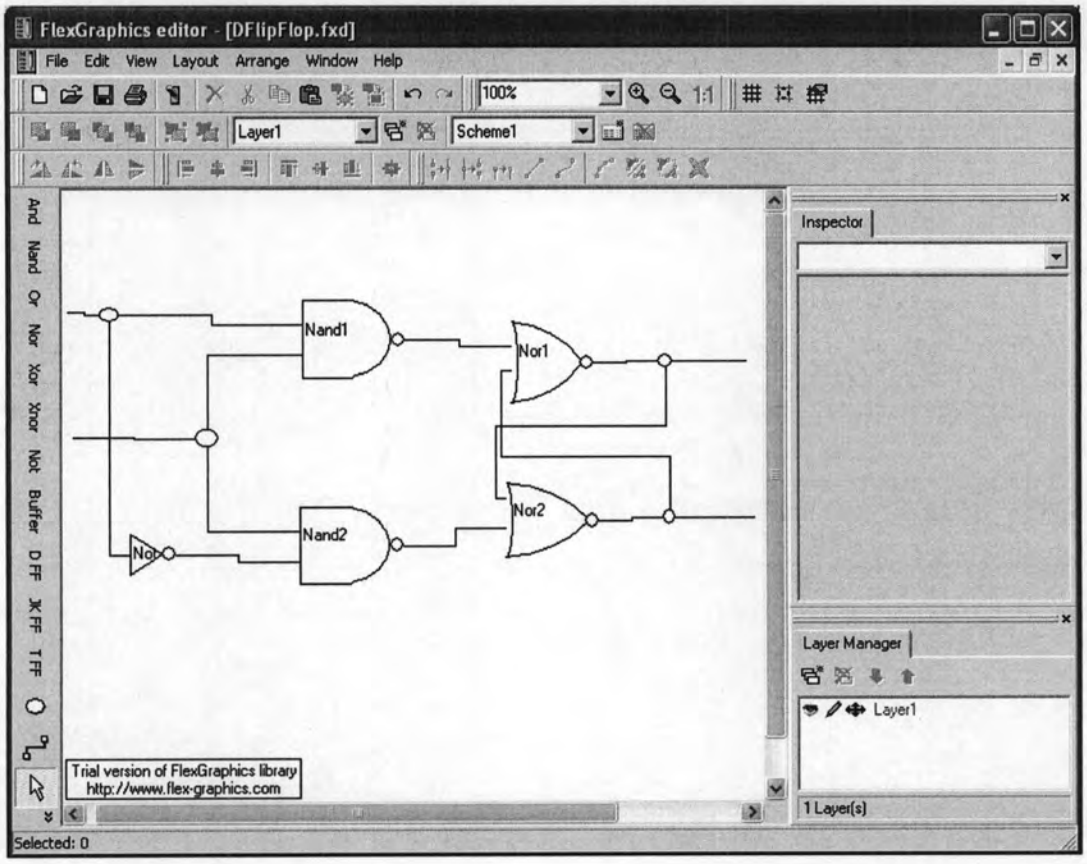
4.11.4 ทุกอุปกรณ์จะต้องมีอินพุตและเอาต์พุตเสมอ เช่น สายสัญญาณจะต้องไม่ใช่สายสัญญาณเปล่า อุปกรณ์อื่นๆ ต้องมีอินพุต 1 สัญญาณหรือขั้นต่ำ 2 สัญญาณแล้วแต่ชนิดของอุปกรณ์ และต้องมี 1 เอาต์พุต

4.12 ทดสอบการนำวงจรมาแก้ไข พบว่าสามารถแก้ไขได้เฉพาะส่วนวงจรที่สร้างขึ้นจากกลุ่มของเกทและฟลิปฟลอปเท่านั้น วงจรที่สร้างขึ้นจากวงจรบล็อกไม่สามารถนำมาแก้ไขได้ โดยการนำวงจรมาแก้ไข ให้เลือกเมนูเปิด แล้วเลือกวงจรตามต้องการ ดังรูปที่ 4.17 และหลังจากเลือกวงจรแล้วจะแสดงวงจรดังรูปที่ 4.18

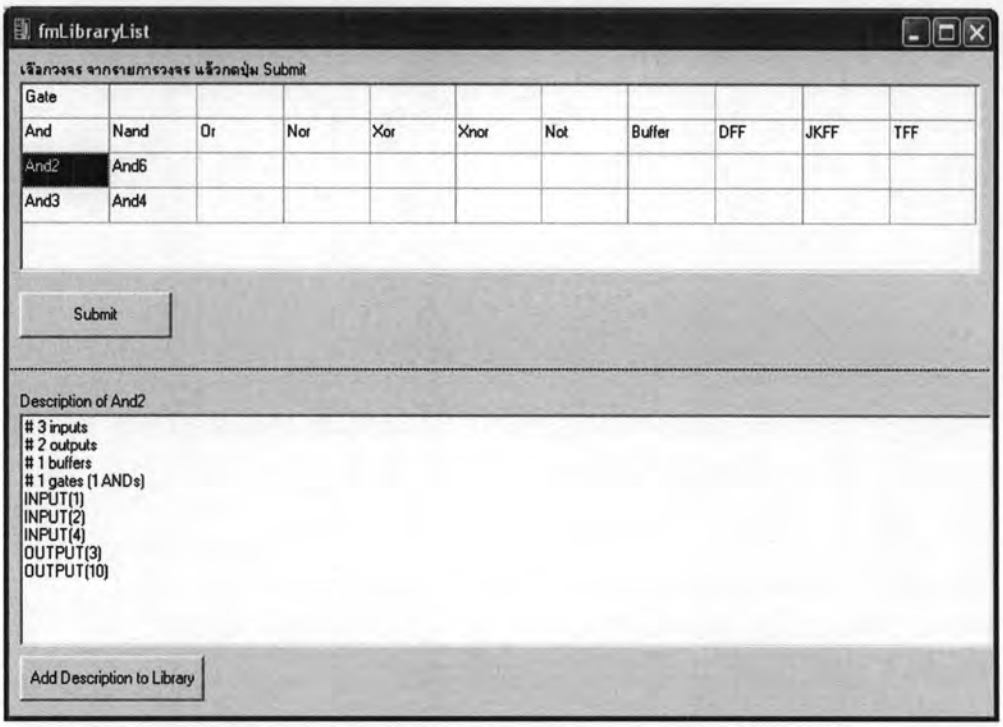
4.13 เพิ่มคุณสมบัติการถ่ายทอดและการนำกลับมาใช้ ให้สามารถใช้งานได้ในขณะออกแบบวงจร โดยสร้างฟอร์มกำหนดการถ่ายทอดและการกำหนดรายละเอียดให้เกทต่างๆ ดังรูปที่ 4.19



รูปที่ 4.17 แสดงการเลือกวงจรที่ได้สร้างไว้แล้ว นำมาแก้ไข



รูปที่ 4.18 แสดงตัวอย่างการนำวงจรที่สร้างแล้วมาแก้ไขใหม่



รูปที่ 4.19 แสดงการเพิ่มฟอร์มการถ่ายทอดและการกำหนดรายละเอียดให้เกทต่างๆ

สรุปการเปรียบเทียบข้อดีข้อเสียของไลบรารี ที่ช่วยในการพัฒนาเครื่องมือส่วนออกแบบกราฟิกได้ดังตารางที่ 1

ตารางที่ 1 แสดงการเปรียบเทียบข้อดีข้อเสียของไลบรารีช่วยในการพัฒนาเครื่องมือชนิดต่างๆ

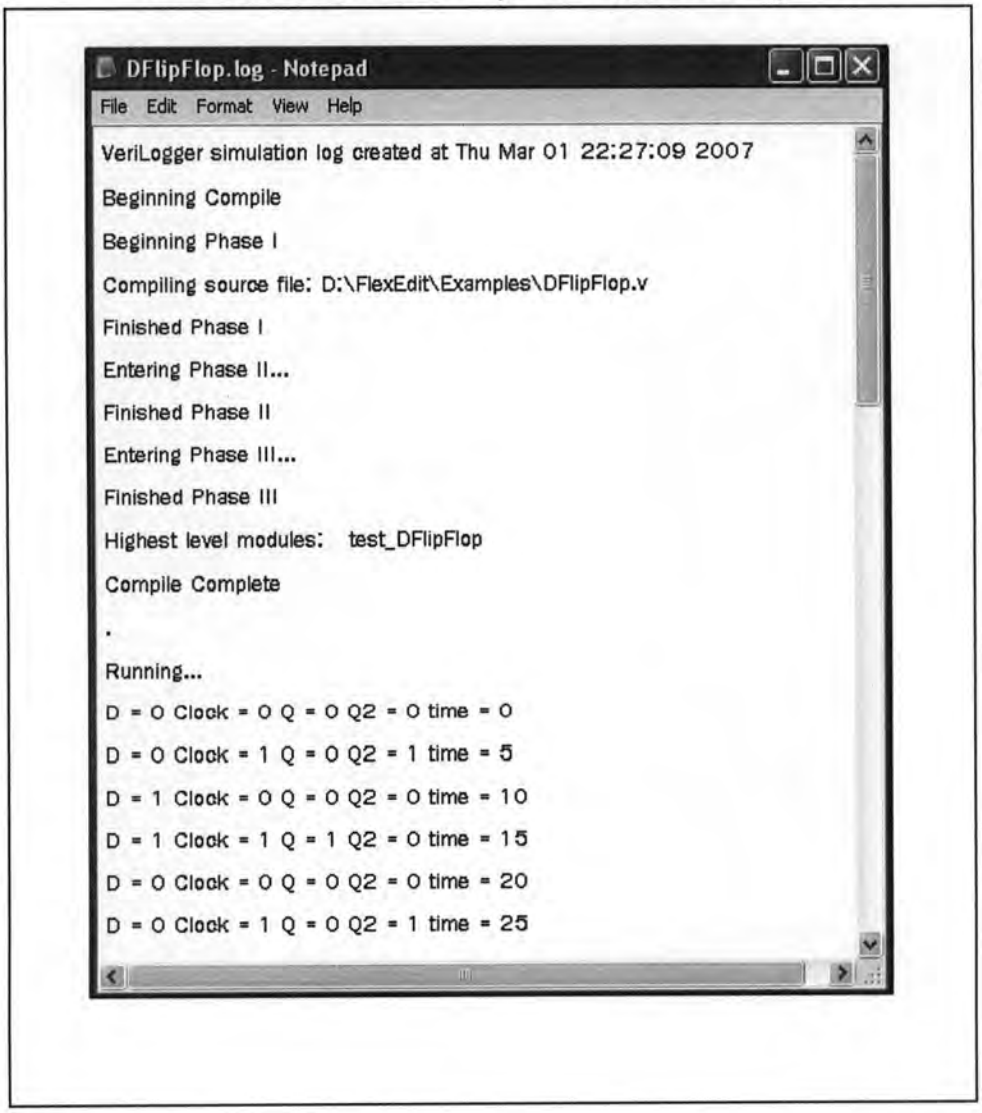
	ข้อดี	ข้อเสีย
Graphic Editor (Ing.Büro R.Tschaggelar, 2003)	-จัดการกราฟิกได้ดี -ใช้แนวคิดเชิงวัตถุพัฒนา -ให้ซอร์สโคดมาทั้งหมด - มีขนาดเล็ก	-พัฒนาโปรแกรมกราฟิกแบบพื้นฐาน ยากต่อการนำไปช่วยในการพัฒนาเครื่องมือ
LEADTOOLS® Raster Imaging	- เน้นกราฟิกแบบละเอียด	- ใกล้เคียงกับเดลไฟอยู่แล้ว
Image++ (Image Integration, 2006)	- ช่วยพัฒนาเครื่องมือกราฟิกเชิงวัตถุ (Object-oriented graphic development tool)	- ต้องใช้ร่วมกับเครื่องมือพัฒนาโปรแกรมภาษา C++
FlexGraphics for Borland Delphi/C++Builder (FlexGraphics Software,	- จัดการรูปภาพชนิดต่างๆ แบบเวกเตอร์ได้ดี ตรงกับวัตถุประสงค์	- ทำให้ไม่สามารถรันโปรแกรมเครื่องมือแบบโดดๆ ได้ - ไม่ให้ซอร์สโคดมาทั้งหมด

2003)		และพัฒนาไลบรารีซับซ้อน

ส่วนเครื่องมือจำลองการทำงาน

เครื่องมือจำลองการทำงานที่จะนำมาเชื่อมต่อกับเครื่องมือที่สร้างขึ้นในวิทยานิพนธ์นี้ เดิมเนื่องจาก Analog Insydes ไม่สนับสนุนการจำลองการทำงานในระดับเกท จึงค้นหาเครื่องมือจำลองการทำงาน โดยเฉพาะเครื่องมือจำลองการทำงานที่สามารถรับไฟล์จากการสังเคราะห์การทำงานหรือเน็ตลิสต์ประเภทต่างๆ ได้ พบว่าโปรแกรม Verilogger Pro สามารถรับไฟล์จากการสังเคราะห์การทำงานซึ่งเป็นภาษาเวอริล็อกได้ และยังสามารถสั่งให้ทำงานโดยใช้คำสั่งได้ ดังนั้นจึงต้องทำการแปลงไฟล์ที่สังเคราะห์ได้จากเน็ตลิสต์ให้เป็นไฟล์เวอริล็อกก่อน จึงนำไปใช้งานสร้างการจำลองการทำงานได้ วิธีการจำลองการทำงานสามารถทำได้ 2 รูปแบบได้แก่

1. หลังจากการสังเคราะห์วงจรในเครื่องมือสร้างวงจรแบบกราฟิก และทำการ Save เรียบร้อยแล้ว ให้เลือกปุ่ม Simulator เลือกชื่อวงจรที่อยู่ในนามสกุล *.v แล้วจะแสดงผลการจำลองการทำงานในรูปแบบของไฟล์ข้อความซึ่งเป็นไฟล์ที่มีชื่อเป็นชื่อวงจรและมีนามสกุลเป็น *.log ดังแสดงในรูป 4.20



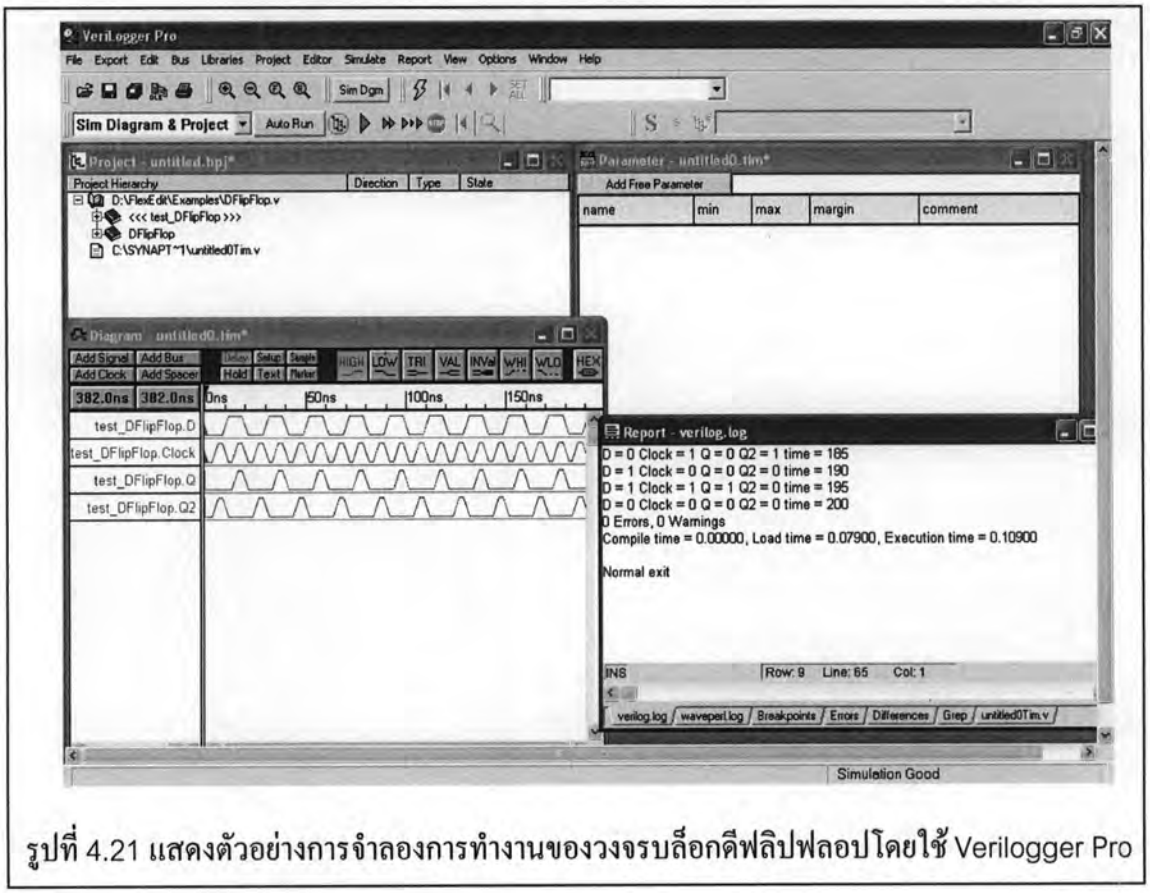
รูปที่ 4.20 แสดงตัวอย่างการจำลองการทำงานในรูปไฟล์ข้อความ

2. หลังจากการสังเคราะห์วงจรในเครื่องมือสร้างวงจรแบบกราฟิกแล้ว ทำการจำลองการทำงานแบบกราฟิกโดยจำลองการทำงานในรูปแบบ waveform และจำลองการทำงานในรูปแบบไฟล์ข้อความโดยใช้โปรแกรม Verilogger Pro ต่างหาก โดยเปิดโปรแกรมขึ้นมาก่อน แล้วเลือกไฟล์ของวงจรที่ต้องการ โดยไฟล์ที่เลือกจะต้องเป็นชื่อวงจรและมีนามสกุลเป็น .v โดยทำตามลำดับดังนี้

2.1 เลือก Project-> Add File(s)

2.2 กดปุ่ม  เพื่อคอมไพล์

2.3 กดปุ่ม  เพื่อรันการจำลองการทำงาน จะได้ผลการทำงานในรูปแบบกราฟิกหรือ waveform และในรูปแบบไฟล์ข้อความได้ดังรูปที่ 4.21



รูปที่ 4.21 แสดงตัวอย่างการจำลองการทำงานของวงจรบล็อกดีฟลิปฟลอปโดยใช้ Verilogger Pro