การออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี

นายกวี วัฒนะวิรุณ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2544
ISBN 974-03-0628-4

DESIGN OF AN ACKNOWLEDGEMENT CIRCUIT FOR B-TERNARY LOGIC COMBINATIONAL CIRCUITS

Mr. Kawee Wattanaviroon

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2001
ISBN 974-03-0628-4

หัวข้อวิทยานิพนธ์	การออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี
โดย	นายกวี วัฒนะวิรุณ
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.อาทิตย์ ทองทักษ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล)

..... อาจารย์ที่ปรึกษา
(อาจารย์ ดร.อาทิตย์ ทองทักษ์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สาธิต วงศ์ประทีป)

..... กรรมการ
(ดร.อิทธิ ฤทธาภรณ์)

กวี วัฒนะวิรุณ : การออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี (DESIGN OF AN ACKNOWLEDGEMENT CIRCUIT FOR B-TERNARY LOGIC COMBINATIONAL CIRCUITS) อ.ที่ปรึกษา : อ. ดร.อาทิตย์ ทองทักษ์, 96 หน้า. ISBN 974-03-0628-4.

วิทยานิพนธ์นี้เสนอการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่สามารถป้องกันการเกิดฮาซาร์ดเนื่องจากความหน่วง ภายใต้แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน และแบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมรับเข้าส่งออก ด้วยการนำเสนออุปกรณ์หลัก 2 ชนิด ที่ออกแบบในระดับทรานซิสเตอร์ คืออุปกรณ์ตรวจสอบขั้นการทำงานและอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร โดยที่อุปกรณ์ตรวจสอบขั้นการทำงานเป็นวงจรที่ใช้ในการตรวจสอบขั้นการทำงานของสายสัญญาณในวงจรเชิงผสม และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรเป็นวงจรที่ใช้ในการป้องกันการเปลี่ยนแปลงสัญญาณเอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสม งานวิจัยนี้ได้ใช้โปรแกรมสไปซ์ในการจำลองการทำงานอุปกรณ์ตรวจสอบขั้นการทำงาน และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร และได้พัฒนาโปรแกรมสำเร็จภาษาวีเอชดีแอลเพื่อใช้ในการจำลองการทำงานของวงจรตรรกะไตรภาคชนิดบีในระดับเกต ผลการทดลองแสดงให้เห็นว่าการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่นำเสนอสามารถป้องกันการเกิดฮาซาร์ดเนื่องจากความหน่วงบนวงจรได้ โดยที่วงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีมีค่าใช้จ่ายน้อยกว่าวงจรตอบรับสำหรับวงจรเชิงผสมประเภทรางคู่ และความหน่วงของวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีมีค่าเท่ากับความหน่วงของวงจรตอบรับสำหรับวงจรรางคู่

ภาควิชา	วิศวกรรรมคอมพิวเตอร์	ลายมือชื่อนิสิต	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่ออาจารย์ที่ปรึกษา	
ปีการศึกษา	2544	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม	

4270212521 : MAJOR COMPUTER ENGINEERING

KEY WORD: ASYNCHRONOUS CIRCUITS / B-TERNARY LOGIC / ACKNOWLEDGEMENT CIRCUIT / PHASE DETECTOR ELEMENT / ASYMMETRIC TERNARY C-ELEMENT

KAWEE WATTANAVIROON : DESIGN OF AN ACKNOWLEDGEMENT CIRCUIT FOR B-TERNARY LOGIC COMBINATIONAL CIRCUITS. THESIS ADVISOR : ARTHIT THONGTAK, Ph.D., 96 pp. ISBN 974-03-0628-4.

This thesis presents a design of an acknowledgement circuit for B-ternary logic combinational circuits for preventing delay hazard under Quasi-Delay-Insensitive model and input-output mode operation. This method presents 2 main elements that were designed on transistor level. The first is a phase detector element, and the second one is an asymmetric ternary c-element. The phase detector element is a circuit that is used for detecting phase of internal wires in combinational circuits. The asymmetric ternary c-element is a circuit that is used for preventing the output signal from changing before internal signals of the combinational circuit. This research uses SPICE for simulating the phase detector and asymmetric ternary c-element, and develops a VHDL package tool for simulating B-ternary logic circuits on the gate level. From the experimental results, it has been shown that the design of the acknowledgement circuit for B-ternary logic circuits can prevent delay hazard. The hardware cost of the acknowledgement circuit for B-ternary logic circuits is less than that of the acknowledgement circuit for dual-rail circuits. The delay time of the acknowledgement circuit for B-ternary logic circuits is equal to that of the acknowledgement circuit for dual-rail circuits.

Department	Computer Engineering	Student's signature	
Field of study	Computer Engineering	Advisor's signature	
Academic year	2001	Co-advisor's signature	

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยความกรุณาอย่างดียิ่งของ อาจารย์ ดร. อาทิตย์ ทองทักษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้คำแนะนำและข้อคิดเห็นต่างๆ ในการทำวิจัยด้วยดีมาตลอด

ขอขอบคุณ อาจารย์ ดร.นัยวุฒิ วงษ์โคเมท และนายณพงศ์ ปณิธานธรรม ที่ให้คำแนะนำการออกแบบวงจรระดับทรานซิสเตอร์และการใช้โปรแกรมสไปซ์

ขอขอบคุณ นางสาวปัญจภา เรืองสินทรัพย์ ที่ให้คำแนะนำการใช้โปรแกรมโมเดลซิม

ขอขอบคุณ ห้องปฏิบัติการ Digital System Engineering Laboratory ที่เอื้อเฟื้อสถานที่ในการทำวิจัย

สุดท้ายนี้ผู้วิจัยขอขอบพระคุณบิดา มารดา ที่ส่งเสริมการศึกษา สนับสนุน และให้กำลังใจแก่ผู้ทำวิจัยเสมอมา

กวี วัฒนะวิรุณ

12 กันยายน 2544

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

บทที่ หน้า

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

ปัจจุบันการออกแบบวงจรรวมได้รับการพัฒนาให้มีความซับซ้อนมากขึ้นและทำงานที่ระดับความเร็วสูง ซึ่งส่งผลให้วงจรมีขนาดใหญ่ ใช้ความถี่ของสัญญาณนาฬิกาสูงขึ้น และความหน่วงของสายสัญญาณที่ใช้เชื่อมต่อวงจรภายในมีผลต่อความหน่วงของวงจรรวม ทำให้เกิดปัญหาการแกว่งของสัญญาณนาฬิกา (Clock Skew) [1] ที่กระจายสัญญาณนาฬิกาไปยังส่วนต่างๆ ของวงจรได้ไม่พร้อมกัน เพื่อหลีกเลี่ยงปัญหาดังกล่าวจึงทำให้การออกแบบวงจรอสมวาร (Asynchronous) ซึ่งเป็นการออกแบบวงจรโดยไม่ใช้สัญญาณนาฬิกา ได้รับความสนใจอย่างมาก เพราะนอกจากไม่เกิดปัญหาการแกว่งของสัญญาณนาฬิกาแล้ว ประสิทธิภาพรวมของวงจรยังเป็นกรณีเฉลี่ย (Average Case Performance) และมีการประหยัดพลังงานเนื่องจากไม่สูญเสียพลังงานในการเปลี่ยนระดับสัญญาณของสัญญาณนาฬิกาและใช้พลังงานเฉพาะส่วนที่ทำงานอยู่ในขณะนั้นเท่านั้น

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากวงจรอสมวารไม่มีสัญญาณนาฬิกาที่ใช้ในการกำหนดเวลาในการส่งผ่านข้อมูลอย่างเช่นวงจรสมวาร (Synchronous Circuits) ดังนั้นวงจรอสมวารจึงอาศัยการเปลี่ยนแปลงสัญญาณเป็นตัวกำหนดการเปลี่ยนแปลงข้อมูลแทนการใช้สัญญาณนาฬิกา โดยที่วงจรอสมวารจะทำงานตอบสนองการเปลี่ยนแปลงของสัญญาณทุกครั้ง เมื่อพิจารณาวงจรอสมวารที่ใช้วิธีการส่งผ่านข้อมูลโดยใช้สัญญาณร้องขอและสัญญาณตอบรับ (Request and Acknowledgement Signals) วงจรจะทำงานได้ถูกต้องก็ต่อเมื่อสัญญาณข้อมูลไปถึงวงจรปลายทางก่อนสัญญาณร้องขอเท่านั้น จึงทำให้มีข้อจำกัดด้านแบบจำลองความหน่วง (Delay Model) ที่สามารถใช้กับแบบจำลองความหน่วงชนิดมีขอบเขต (Bounded-Delay Model) [1, 2] ได้เท่านั้น แบบจำลองความหน่วงคือข้อกำหนดของความหน่วงที่ใช้การออกแบบวงจร และแบบจำลองความหน่วงชนิดมีขอบเขตกำหนดความหน่วงของวงจรโดยใช้สมมุติฐานว่ารู้ขีดจำกัดของความหน่วงของเกตและสายที่ใช้ในวงจร ซึ่งทำให้วงจรที่ได้มีประสิทธิภาพเป็นกรณีเลยร้ายสุด (Worst Case Performance) เช่นเดียวกับวงจรสมวาร ในขณะที่แบบจำลองความหน่วงอื่นๆ เช่นแบบจำลองความหน่วงแบบไม่ขึ้นต่ออัตราเร็ว (Speed Independent : SI), แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วง (Delay-Insensitive) และแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน (Quasi-Delay-Insensitive) มีประสิทธิภาพเป็นกรณีเฉลี่ย ดังนั้นการส่งผ่านข้อมูลของวงจรอสมวารจึงจำเป็นต้องมีการเข้ารหัสสัญญาณเพื่อใช้สัญญาณข้อมูลเป็นตัวประสานเวลา

(Synchronize) แทนการใช้สัญญาณร้องขอ เช่นรหัสรางคู่ชนิดกลับสู่ศูนย์ (Dual-Rail Return-to-Zero) [3] และตรรกะไตรภาคชนิดบี (B-ternary Logic) [4-6]

รหัสที่ใช้กับวงจรอสมวารที่มีสัญญาณข้อมูลเป็นตัวประสานเวลาจะต้องมีลักษณะที่ทำให้วงจรสามารถแยกอินพุตเก่าและอินพุตใหม่ออกจากกันได้ รหัสรางคู่ชนิดกลับสู่ศูนย์เป็นรหัสชนิดหนึ่งที่มีคุณสมบัติทำให้วงจรสามารถแยกอินพุตเก่าและอินพุตใหม่ออกจากกันได้ โดยใช้รหัสสองบิตแทนค่าสามค่าโดยที่ค่าสองค่าเป็นค่าตรรกะและค่าอีกหนึ่งค่าเป็นตัวแบ่งรอบการทำงาน (Spacer) เพื่อแยกอินพุตเก่าและอินพุตใหม่ออกจากกัน แต่เนื่องจากการใช้รหัสสองบิตแทนข้อมูลหนึ่งบิตส่งผลให้วงจรที่ได้นั้นใช้เกตและสายมากกว่าวงจรสมวารถึงสองเท่า ดังนั้นการออกแบบวงจรโดยใช้ตรรกะไตรภาคชนิดบี (B-ternary Logic) ที่เสนอโดย Nagata [4-6] จึงเป็นทางเลือกทางหนึ่งที่ใช้แทนรหัสรางคู่ โดยการนำตรรกะไตรภาคเข้ามาช่วยซึ่งเป็นตรรกะที่บิตข้อมูลหนึ่งบิตสามารถแทนค่าได้สามค่า แทนการใช้การเข้ารหัสสองบิตของตรรกะฐานสอง (Binary Logic) ที่บิตข้อมูลหนึ่งบิตสามารถแทนค่าได้สองค่าเพื่อลดจำนวนสายที่ใช้ลงครึ่งหนึ่งของจำนวนสายที่ใช้ในวงจรรางคู่ชนิดกลับสู่ศูนย์ นอกจากนี้วงจรตรรกะไตรภาคชนิดบีสามารถออกแบบได้ง่ายโดยใช้เกตประเภทตรรกะไตรภาคชนิดบีและวิธีการออกแบบวงจรวิธีเดียวกับการออกแบบวงจรฐานสอง (Binary Circuits)

การออกแบบวงจรอสมวารนั้นนอกจากจะต้องคำนึงถึงวิธีการส่งผ่านข้อมูลและแบบจำลองความหน่วงที่ใช้ในการออกแบบวงจรแล้ว จะต้องคำนึงถึงแบบจำลองการทำงานสิ่งแวดล้อม (Environment Operation Model) [7] ด้วย เนื่องจากแบบจำลองการทำงานสิ่งแวดล้อมมีผลต่อความถูกต้องในการทำงานของวงจร ดังนั้นเมื่อพิจารณาการออกแบบวงจรอสมวารที่มีเฉพาะส่วนตรรกะเพียงอย่างเดียว วงจรสามารถทำงานได้ถูกต้องเฉพาะภาวะแวดล้อมมูลฐาน (Fundamental Mode Environment : FM) แต่สามารถเกิดฮาซาร์ด (Hazard) ได้เมื่ออยู่ในภาวะแวดล้อมรับเข้าส่งออก (Input-Output Mode Operation : IO Mode)

เมื่อภาวะแวดล้อมของวงจรเป็นแบบภาวะแวดล้อมรับเข้าส่งออก การทำงานของสิ่งแวดล้อมจะเร็วกว่าการทำงานของวงจร ส่งผลให้การตรวจสอบการสิ้นสุดการทำงานโดยวิธีการตรวจสอบเอาต์พุตของวงจรนั้นไม่เพียงพอ วงจรสามารถเกิดฮาซาร์ดได้เนื่องจากความหน่วงของทางเดินข้อมูลค่าหนึ่งมีค่ามากกว่าเส้นทางอีกเส้นทางหนึ่ง ทำให้มีสัญญาณภายในวงจรบางสัญญาณยังไม่สิ้นสุดการเปลี่ยนแปลงสัญญาณในขณะที่เกิดการเปลี่ยนแปลงสัญญาณที่เอาต์พุต เมื่ออินพุตใหม่เข้ามายังวงจร จะทำให้เกิดการแทรกสอดของข้อมูล เป็นผลให้เกิดฮาซาร์ดที่เอาต์พุต

ของวงจร ดังนั้นการป้องกันการเปลี่ยนแปลงสัญญาณที่อินพุตของวงจรก่อนการสิ้นสุดการเปลี่ยน-แปลงระดับสัญญาณภายในวงจรจึงต้องอาศัยวงจรที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรแทนการตรวจสอบการสิ้นสุดการทำงานที่เอาต์พุตของวงจร

การวิจัยนี้ศึกษาและปรับปรุงวิธีการส่งผ่านข้อมูลแบบตรรกะไตรภาคชนิดบีด้วยการนำเสนอวิธีการออกแบบวงจรตอบรับ (Acknowledgement Circuit) ซึ่งเป็นวงจรที่ทำหน้าที่ตรวจสอบการสิ้นสุดของสัญญาณภายในของวงจรเชิงผสมแบบอสมวาร เพื่อประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน ซึ่งเป็นการป้องกันการเปลี่ยนระดับสัญญาณอินพุตก่อนเวลาอันควรสำหรับภาวะแวดล้อมรับเข้าส่งออก

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่ประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในวงจร ซึ่งเป็นการป้องกันฮาซาร์ดอันเกิดจากสาเหตุอินพุตของวงจรเปลี่ยนก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี

1.3 ขอบเขตของการวิจัย

1. ออกแบบวงจรตอบรับที่รับประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี
2. เปรียบเทียบขนาดของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีกับวงจรเชิงผสมประเภทรางคู่ชนิดกลับสู่ศูนย์ที่มีส่วนวงจรตอบรับ โดยใช้วงจรวัดเปรียบเทียบสมรรถนะ (Benchmark Circuits) LGSynth 89-93 เป็นเกณฑ์
3. ใช้โปรแกรมสไปซ์ (SPICE) ในการจำลองการทำงานของอุปกรณ์ต่างๆ ในระดับทรานซิสเตอร์ และใช้ภาษาวีเอชดีแอล (VHDL) และโปรแกรมโมเดลซิม (ModelSim) ในการจำลองการทำงานระดับเกต

1.4 ประโยชน์ที่ได้รับ

1. ได้แนวทางการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี
2. เพิ่มความถูกต้องให้แก่วงจรที่ออกแบบโดยใช้วงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี

3. ทราบความเป็นไปได้ในพัฒนาการออกแบบวงจรโดยใช้วงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีในอนาคต
4. ได้วงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่มีขนาดเล็กกว่าเมื่อเทียบกับวงจรตอบรับสำหรับวงจรเชิงผสมประเภทรางคู่ชนิดกลับสู่ศูนย์

1.5 ลำดับขั้นตอนในการเสนอผลการวิจัย

1. ศึกษาการออกแบบวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี
2. ศึกษาการออกแบบวงจรตอบรับที่ประกันการสิ้นสุดการเปลี่ยนสัญญาณภายในวงจรเชิงผสม
3. เสนอแนวทางการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี
4. ออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี
5. จำลองการทำงานของวงจรเพื่อประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสม
6. เปรียบเทียบขนาดของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีกับขนาดของวงจรเชิงผสมประเภทรางคู่ชนิดกลับสู่ศูนย์ที่มีวงจรตอบรับ โดยใช้วงจรวัดเปรียบเทียบสมรรถนะ LGSynth 89-93 เป็นเกณฑ์
7. สรุปผลการวิจัยและจัดทำวิทยานิพนธ์

1.6 โครงสร้างของวิทยานิพนธ์

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 7 บท บทที่ 1 เป็นบทนำซึ่งกล่าวถึงที่มาของปัญหาและวัตถุประสงค์ของงานวิจัย บทที่ 2 สรุปแนวคิดและทฤษฎีที่เกี่ยวข้อง บทที่ 3 อธิบายการวิเคราะห์ฮาซาร์ดและวิธีการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่สามารถป้องกันฮาซาร์ดเนื่องจากความหน่วง บทที่ 4 อธิบายรูปแบบของการจำลองการทำงาน บทที่ 5 เสนอการทดลองเพื่อทดสอบความถูกต้องในการทำงานของวงจรตรรกะไตรภาคชนิดบีที่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน บทที่ 6 เสนอผลของการวิเคราะห์ขนาดของวงจรและเวลาในการทำงานของวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีโดยเปรียบเทียบกับขนาดของวงจรและเวลาในการทำงานของวงจรตอบรับสำหรับวงจรรางคู่ และบทที่ 7 เป็นบทที่สรุปผลการวิจัยและข้อเสนอแนะ

1.7 ผลงานที่ตีพิมพ์จากงานวิจัย

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ตีพิมพ์เป็นบทความทางวิชาการ ในหัวข้อ "การออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีด้วยอุปกรณ์ตรวจสอบขั้นการทำงาน" โดย กวี วัฒนะวิรุณ และ อาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ "The Fifth National Computer Science and Engineering Conference (NCSEC'2001)" ซึ่งจะจัดโดยภาควิชาวิทยาศาสตร์คอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่ ณ โรงแรมโลตัสปางสวนแก้ว จังหวัดเชียงใหม่ ในวันที่ 7-9 พฤศจิกายน 2544

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้ตีพิมพ์เป็นบทความทางวิชาการ ในหัวข้อ "การออกแบบวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่ไร้ฮาซาร์ดเนื่องจากความหน่วง" โดย กวี วัฒนะวิรุณ และ อาทิตย์ ทองทักษ์ ในงานประชุมวิชาการ "การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 24 (24th Electrical Engineering Conference : EECON24)" ซึ่งจะจัดโดยคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ณ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในวันที่ 22-23 พฤศจิกายน 2544

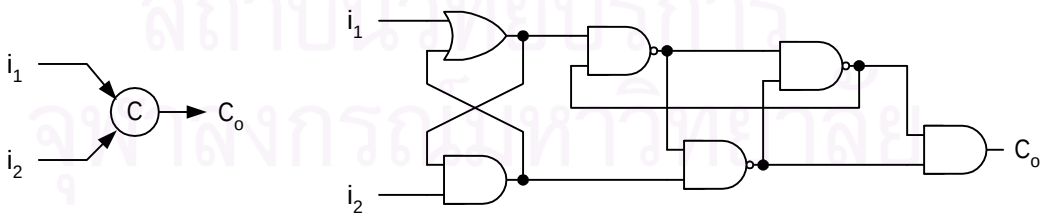
บทที่ 2
แนวคิดและทฤษฎีที่เกี่ยวข้อง

วงจรอสมวารเป็นวงจรที่ไม่มีสัญญาณนาฬิกาช่วยในการประสานเวลาในการส่งข้อมูล ดังนั้นวงจรจึงทำงานโดยตอบสนองการเปลี่ยนแปลงระดับสัญญาณทุกครั้ง เพื่อให้วงจรทำงานได้อย่างถูกต้องและมีประสิทธิภาพจึงต้องคำนึงถึงความหน่วงและการส่งข้อมูลเป็นหลัก แบบจำลองของการทำงานถูกแบ่งออกเป็น 2 แบบ คือแบบจำลองความหน่วง และแบบจำลองการทำงานสิ่งแวดล้อม แบบจำลองความหน่วงเป็นแบบจำลองที่ใช้กำหนดลักษณะความหน่วงของวงจรที่ออกแบบ ส่วนแบบจำลองการทำงานสิ่งแวดล้อมเป็นแบบจำลองที่ใช้กำหนดลักษณะความหน่วงของวงจรเทียบกับความหน่วงของสิ่งแวดล้อมที่เชื่อมต่อกับวงจร

เนื่องจากการออกแบบวงจรอสมวารมีการอ้างอิงถึงอุปกรณ์ชนิดซีในการออกแบบวงจร ดังนั้นในบทนี้จึงอธิบายถึงอุปกรณ์ชนิดซีเป็นหัวข้อแรกเพื่อใช้อ้างอิงในส่วนอื่นๆ จากนั้นจึงเป็นส่วนแนวคิดและทฤษฎีที่เกี่ยวข้องอื่นๆ ซึ่งได้แก่แบบจำลองความหน่วง, แบบจำลองการทำงานสิ่งแวดล้อม, การออกแบบวงจรด้วยรหัสรางคู่ชนิดกลับสู่ศูนย์ และการออกแบบวงจรด้วยตรรกะไตรภาคชนิดบี

2.1 **อุปกรณ์ชนิดซี** (C-element)

อุปกรณ์ชนิดซี [8, 9] เป็นอุปกรณ์ที่เอาต์พุตมีค่าเท่ากับอินพุตหลังจากอินพุตทุกอินพุตมีค่าเท่ากัน หรืออีกนัยหนึ่งคือจะให้เอาต์พุตเป็นศูนย์เมื่ออินพุตทุกอินพุตเป็นศูนย์และให้เอาต์พุตเป็นหนึ่งเมื่ออินพุตทุกอินพุตเป็นหนึ่ง ส่วนกรณีอื่นเอาต์พุตมีค่าคงเดิม อุปกรณ์ชนิดซีที่มีอินพุตสองอินพุตมีลักษณะดังรูปที่ 2.1



รูปที่ 2.1 อุปกรณ์ชนิดซีที่มีอินพุตสองอินพุต

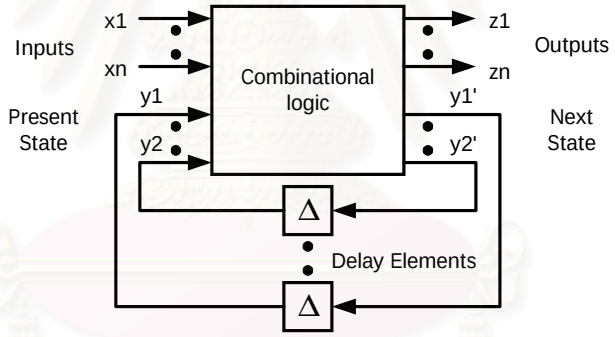
2.2 **แบบจำลองความหน่วง** (Delay Model)

แบบจำลองความหน่วงสร้างขึ้นจากสมมุติฐานที่ใช้กำหนดลักษณะความหน่วงของวงจรที่ออกแบบ และแบ่งออกเป็นสองกลุ่มหลัก คือกลุ่มแบบจำลองความหน่วงชนิดมีขอบเขต

(Bounded Delay Model) และแบบจำลองความหน่วงชนิดไม่มีขอบเขต (Unbounded Delay Model) กลุ่มแบบจำลองความหน่วงชนิดมีขอบเขตประกอบด้วยแบบจำลองความหน่วงชนิดมีขอบเขตเพียงประเภทเดียว และกลุ่มแบบจำลองความหน่วงชนิดไม่มีขอบเขตประกอบด้วย แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วง, แบบจำลองความหน่วงแบบไม่ขึ้นต่ออัตราเร็ว และแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน แบบจำลองความหน่วงแต่ละประเภทมีความหมายดังต่อไปนี้ คือเมื่อกำหนดให้ Δ เป็นความหน่วงและ α, β, ϕ, γ, ε เป็นค่าความหน่วงโดยที่ ε มีค่าน้อยมากจนอาจจะถือว่าเป็นศูนย์ได้เมื่อเทียบกับ α, β, ϕ, γ

2.2.1 แบบจำลองความหน่วงชนิดมีขอบเขต (Bounded Delay)

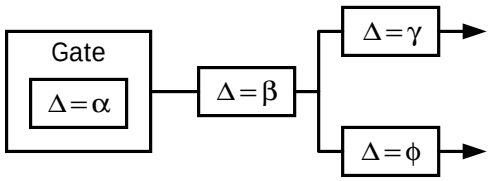
แบบจำลองความหน่วงชนิดมีขอบเขต [1] เป็นแบบจำลองความหน่วงที่ใช้สมมุติฐานว่ารู้ขอบเขตความหน่วงในเกตและสายของวงจร ทำให้การออกแบบวงจรสามารถออกแบบวงจรได้ดังรูปที่ 2.2 ดังนั้นวงจรจะทำงานได้ถูกต้องก็ต่อเมื่ออินพุตใหม่ของวงจรป้อนเข้ามาเมื่อวงจรอยู่ในสถานะสงบนิ่ง (Stable state) ทำให้วงจรที่ได้มีประสิทธิภาพเป็นกรณีเลวร้ายสุด



รูปที่ 2.2 แบบจำลองความหน่วงชนิดมีขอบเขต

2.2.2 แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วง (Delay-Insensitive : DI)

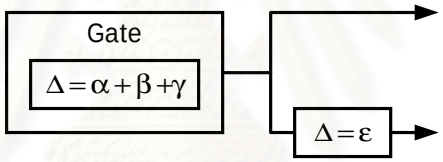
แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วง [1, 10] เป็นแบบจำลองที่ใช้สมมุติฐานว่าไม่สามารถกำหนดขอบเขตของความหน่วงของเกตและสายได้ แต่ทราบว่าค่าความหน่วงของเกตและสายมีขอบเขตที่ไม่ใช่ค่าอนันต์ดังรูปที่ 2.3 ดังนั้นวงจรที่ใช้แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงจะต้องสามารถทำงานได้โดยไม่ขึ้นอยู่กับความหน่วงของเกตและสายที่ใช้ในวงจรยกเว้นความหน่วงของเกตหรือสายมีค่าเป็นบวกอนันต์ วงจรที่จะเป็นแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงได้จะต้องมีคุณสมบัติเป็นชุดลำดับเดี่ยว (Unique-Successor-Set : USS) [10] ทำให้วงจรที่ออกแบบต้องประกอบด้วยเกตผกผันและอุปกรณ์ชนิดซีเท่านั้นซึ่งเป็นข้อจำกัดที่ทำให้ออกแบบวงจรได้ยาก



รูปที่ 2.3 แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วง

2.2.3 แบบจำลองความหน่วงแบบไม่ขึ้นต่ออัตราเร็ว (Speed Independent : SI)

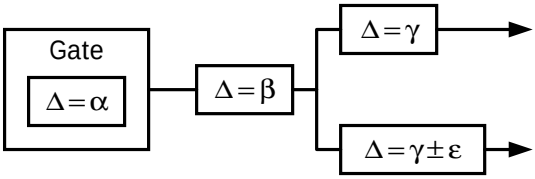
แบบจำลองความหน่วงแบบไม่ขึ้นต่ออัตราเร็ว [1] เป็นแบบจำลองที่ใช้สมมุติฐานว่าไม่สามารถกำหนดขอบเขตของความหน่วงของเกต แต่ความหน่วงของสายมีค่าน้อยมากเมื่อเทียบกับความหน่วงของเกต ทำให้สามารถละเลยความหน่วงของสายได้ หรือประมาณค่าความหน่วงของสายเท่ากับศูนย์ดังรูปที่ 2.4 สมมุติฐานดังกล่าวทำให้แบบจำลองความหน่วงแบบไม่ขึ้นต่ออัตราเร็วไม่เหมาะสมกับเทคโนโลยีในปัจจุบัน



รูปที่ 2.4 แบบจำลองความหน่วงแบบไม่ขึ้นต่ออัตราเร็ว

2.2.4 แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน (Quasi-Delay-Insensitive : QDI)

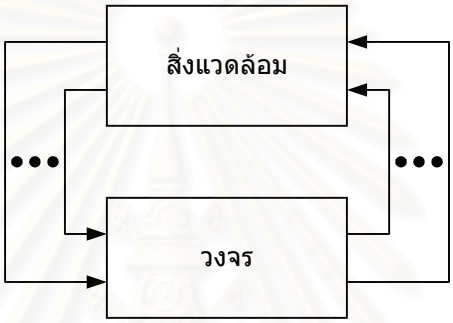
แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน [1, 7] เป็นแบบจำลองที่พัฒนามาจากแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงโดยเพิ่มสมมุติฐานกิ่งของสายเทียบเท่าตลอดช่วง (Isochronic Fork) ซึ่งเป็นสมมุติฐานที่กำหนดให้ความหน่วงของกิ่งของสาย (Fork Wire) ทุกกิ่งมีค่าเท่ากันดังรูปที่ 2.5 การเพิ่มสมมุติฐานดังกล่าวช่วยทำให้ลดข้อจำกัดในการออกแบบวงจรของแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงลง



รูปที่ 2.5 แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน

2.3 **แบบจำลองการทำงานสิ่งแวดล้อม** (Environment Operation Model)

สิ่งแวดล้อมคือส่วนที่ทำหน้าที่รับเอาต์พุตของวงจรเมื่อวงจรทำงานเสร็จและป้อนอินพุตใหม่สู่วงจรดังแสดงในรูปที่ 2.6 เมื่อกำหนดให้ d_{env} คือความหน่วงของสิ่งแวดล้อม และ $d_{circuit}$ คือความหน่วงของวงจร ความหน่วงของสิ่งแวดล้อมเมื่อเทียบกับความหน่วงของวงจรแล้วสามารถจำแนกแบบจำลองการทำงานสิ่งแวดล้อมออกเป็นสองภาวะ [3] คือ ภาวะแวดล้อมมูลฐาน และภาวะแวดล้อมรับเข้าส่งออก โดยที่ภาวะแวดล้อมมูลฐานคือกรณีที่ $d_{circuit} < d_{env}$ และภาวะแวดล้อมรับเข้าส่งออกคือกรณีที่ $d_{circuit} \geq d_{env}$



รูปที่ 2.6 การเชื่อมต่อระหว่างวงจรกับสิ่งแวดล้อม

2.3.1 ภาวะแวดล้อมมูลฐาน (Fundamental Mode Environment : FM)

ในสภาวะที่ความหน่วงของสิ่งแวดล้อมมีค่ามากกว่าความหน่วงของวงจร เราสามารถใช้ความหน่วงของสิ่งแวดล้อมประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรแทนการสร้างวงจรที่ทำการตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในได้ ดังนั้นในการออกแบบวงจรส่วนที่ตรวจสอบการสิ้นสุดการทำงานของวงจรจะอยู่ในส่วนของสิ่งแวดล้อม เพราะเมื่อสิ่งแวดล้อมตรวจสอบได้ว่าวงจรให้เอาต์พุตออกมาแล้วและส่งอินพุตใหม่ไปยังวงจร วงจรก็อยู่ในสภาวะสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในแล้ว

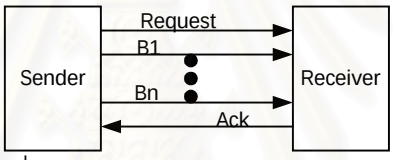
2.3.2 ภาวะแวดล้อมรับเข้าส่งออก (Input-Output Mode Operation : IO Mode)

ในสภาวะที่ความหน่วงของสิ่งแวดล้อมมีค่าน้อยกว่าหรือเท่ากับความหน่วงของวงจร เราไม่สามารถใช้ความหน่วงของสิ่งแวดล้อมประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรได้ จึงทำให้การออกแบบวงจรต้องมีส่วนที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจร เพื่อให้วงจรเปลี่ยนระดับสัญญาณเอาต์พุตเมื่อสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน ดังนั้นเอาต์พุตของวงจรที่มีส่วนตรวจสอบการสิ้นสุดการเปลี่ยนแปลงภายในจะสามารถ

ประกันความถูกต้องได้ดีกว่าการตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณที่เอาต์พุตของวงจร โดยเฉพาะเมื่อเกิดความแปรปรวนความหน่วงของวงจรหรือสิ่งแวดล้อม

2.4 การออกแบบวงจรเชิงผสมแบบอสมวาร

เทคโนโลยีวงจรรวมความจุสูงมากในปัจจุบันส่งผลให้ความหน่วงในสายมีความสำคัญต่อวงจร ทำให้การออกแบบวงจรด้วยวิธีการส่งข้อมูลรวมชุด (Bundle Data) [1, 2] ดังรูปที่ 2.7 ไม่เหมาะสมกับเทคโนโลยีในปัจจุบัน เนื่องจากการออกแบบวงจรด้วยแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วง, หรือแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน ใช้สมมุติฐานว่าไม่รู้ความหน่วงของเกตและสาย เมื่อส่งสัญญาณข้อมูลออกจากฝ่ายส่งข้อมูล (Sender) ไปยังฝ่ายรับข้อมูล (Receiver) ก่อนการส่งสัญญาณร้องขอ (Request Signal) ฝ่ายรับอาจจะได้รับสัญญาณร้องขอก่อนสัญญาณข้อมูลเนื่องจากความหน่วงของสายสัญญาณร้องขอน้อยกว่าความหน่วงของสายสัญญาณข้อมูล เป็นผลให้วงจรฝ่ายรับทำงานผิดพลาด



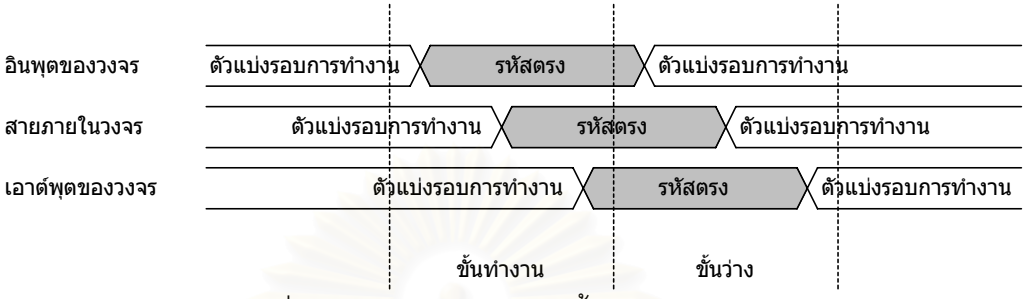
รูปที่ 2.7 วิธีส่งข้อมูลแบบข้อมูลรวมชุด

ดังนั้นการส่งข้อมูลจึงต้องใช้วิธีการส่งสัญญาณข้อมูล (Data Signaling) ซึ่งเป็นวิธีที่ใช้ข้อมูลเป็นตัวกระตุ้นฝ่ายรับข้อมูลแทนการใช้สัญญาณร้องขอ แต่เนื่องจากการส่งข้อมูลโดยใช้สายหนึ่งเส้นต่อข้อมูลหนึ่งบิตจะทำให้ฝ่ายรับข้อมูลไม่สามารถแยกอินพุตเก่าและอินพุตใหม่ออกจากกันเมื่อมีการส่งข้อมูลสองชุดที่เหมือนกันและต่อเนื่องกันไปยังฝ่ายรับข้อมูล ดังนั้นการส่งข้อมูลด้วยวิธีการส่งสัญญาณข้อมูลจำเป็นต้องมีการเข้ารหัสเพื่อให้วงจรฝ่ายรับข้อมูลสามารถแยกอินพุตเก่าและอินพุตใหม่ออกจากกันได้ เช่นการใช้รหัสรางคู่ชนิดกลับสู่ศูนย์ หรือการใช้ตรรกะไตรภาคชนิดบี เป็นต้น

2.5 วงจรรางคู่ชนิดกลับสู่ศูนย์ (Dual-Rail Return-to-Zero Circuits)

รหัสรางคู่ [3] เป็นรหัสที่ใช้ค่าตรรกะฐานสองสองบิตแทนค่าข้อมูลหนึ่งบิตโดยใช้ค่า '01' แทนค่าข้อมูล '0', ใช้ค่า '10' แทนค่าข้อมูล '1' และใช้ค่า '00' แทนตัวแบ่งรอบการทำงาน (Spacer) ส่วนค่า '11' ไม่มีการนิยามการใช้งาน ดังนั้นเมื่อข้อมูลมีจำนวน n บิตจะใช้สายสัญญาณเป็นจำนวน n คู่ โดยเป็นรหัสตรง (Codeword) เมื่อคู่สาย (d1, d0) ทุกคู่สายมีค่าเป็น (0, 1) หรือ (1, 0) เป็นตัวแบ่งรอบการทำงานเมื่อคู่สายทุกคู่สายมีค่าเป็น (0, 0) ส่วนรหัสที่เหลือ

เป็นรหัสไม่ตรง (Non-Codeword) รหัสรางคู่จะทำงานโดยใช้วิธีการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์ (2-phase return-to-zero) การทำงานหนึ่งรอบถูกแบ่งออกเป็นสองขั้น คือขั้นทำงานและขั้นว่างโดยทำงานสลับกันไปในแต่ละรอบดังรูปที่ 2.8

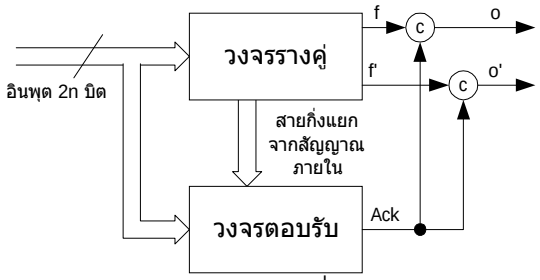


รูปที่ 2.8 การทำงานแบบสองขั้นชนิดกลับสู่ศูนย์

ขั้นทำงาน คือช่วงที่วงจรคำนวณฟังก์ชันตรรกะโดยเริ่มตั้งแต่สายภายในวงจร สายอินพุตของวงจร และสายเอาต์พุตของวงจรทุกเส้นมีค่าเป็นตัวแบ่งรอบการทำงานและได้รับอินพุตรหัสตรงจากสิ่งแวดล้อม จนกระทั่งวงจรคำนวณอินพุตที่ได้รับจากสิ่งแวดล้อมเสร็จสิ้นโดยที่สายภายในวงจรทุกเส้นอยู่ในสภาวะสงบนิ่งและให้ผลลัพธ์เป็นรหัสตรงที่เอาต์พุตของวงจร

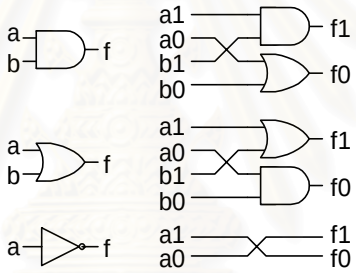
ขั้นว่าง คือช่วงที่วงจรส่งผ่านตัวแบ่งรอบการทำงานจากอินพุตไปยังเอาต์พุตโดยเริ่มตั้งแต่สายภายในวงจร สายอินพุตของวงจร และสายเอาต์พุตของวงจรมีค่าเป็นรหัสตรงอยู่ในภาวะสงบนิ่งและได้รับอินพุตเป็นตัวแบ่งรอบการทำงานจากสิ่งแวดล้อม จนกระทั่งวงจรส่งผ่านตัวแบ่งรอบการทำงานไปยังเอาต์พุตโดยที่สายภายในวงจรทั้งหมดมีค่าเป็นตัวแบ่งรอบการทำงานและอยู่ในสภาวะสงบนิ่ง

เมื่อออกแบบวงจรเชิงผสมโดยใช้แบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมรับเข้าส่งออก วงจรจะต้องประกอบด้วยส่วนประกอบสองส่วน คือส่วนที่ทำหน้าที่ประมวลผล และส่วนวงจรตอบรับซึ่งทำหน้าที่ในการตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรส่วนประมวลผลดังรูปที่ 2.9 สำหรับวงจรเชิงผสมที่ใช้รหัสรางคู่นั้นจะใช้วงจรรางคู่ในส่วนของวงจรประมวลผล วงจรรางคู่สามารถออกแบบได้สองวิธี คือการออกแบบวงจรรางคู่โดยใช้ตรรกะรางคู่ไร้ตัวผกผัน (Inverter-Free 2-Rail Logic Implementation) และการออกแบบวงจรรางคู่โดยใช้แผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ (Reduced Ordered-Binary Decision Diagram (ROBDD) Implementation) เนื่องจากวิทยานิพนธ์นี้วัดประสิทธิภาพของวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีกับวงจรตอบรับสำหรับวงจรรางคู่โดยใช้ตรรกะรางคู่ไร้ตัวผกผัน จึงอธิบายการออกแบบวงจรรางคู่เฉพาะวิธีการออกแบบวงจรรางคู่โดยใช้ตรรกะรางคู่ไร้ตัวผกผัน



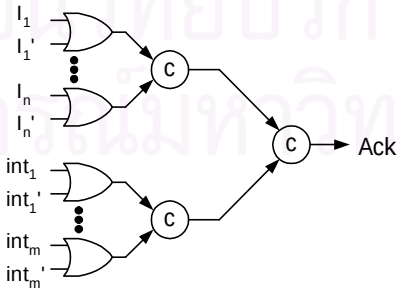
รูปที่ 2.9 วงจรเชิงผสมแบบอสมวารที่ใช้รหัสรางคู่ชนิดกลับสู่ศูนย์

การออกแบบวงจรรางคู่โดยใช้ตรรกะรางคู่ไร้ตัวผกผันจะทำโดยการใช้เกตสองตัวคือเกตแอนด์และเกตออร์อย่างละหนึ่งตัวแทนเกตแอนด์หรือเกตออร์ทั่วไปหนึ่งตัวในวงจรสมวาร และตัวผกผันสำหรับวงจรรางคู่ใช้การไขว้สายดังรูปที่ 2.10 โดยรูปด้านขวาคือเกตที่ใช้สำหรับวงจรรางคู่เมื่อเทียบกับเกตทั่วไปสำหรับวงจรสมวารด้านซ้าย ด้วยเหตุนี้การออกแบบวงจรรางคู่จึงใช้เกตและสายเป็นสองเท่าของวงจรสมวารทั่วไป

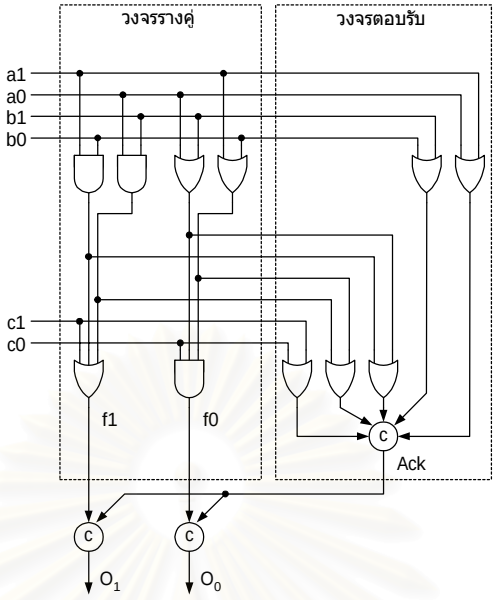


รูปที่ 2.10 เกตสำหรับวงจรสมวารทั่วไป (ซ้าย) เมื่อเทียบกับตรรกะรางคู่ไร้ตัวผกผัน (ขวา)

วงจรตอบรับนั้นใช้วิธีการตรวจสอบคู่สายสัญญาณแต่ละคู่โดยดึงคู่สายอินพุต (I_i, I_i') และคู่สายสัญญาณภายใน (int_i, int_i') แต่ละคู่มาเป็นอินพุตของเกตออร์และสร้างสัญญาณตอบรับรวมโดยใช้อุปกรณ์ชนิดซีดังรูปที่ 2.11 และวงจรรางคู่ f=ab'+a'b+c มีลักษณะดังรูปที่ 2.12



รูปที่ 2.11 วงจรตอบรับสำหรับวงจรรางคู่ที่ไร้ตัวผกผันบนแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน



รูปที่ 2.12 วงจรรางคู่ f=ab'+a'b+c

2.6 **วงจรตรรกะไตรภาคชนิดบี** (B-ternary Logic Circuits)

ตรรกะฐานสองที่ใช้กันอยู่ทั่วไปในปัจจุบันเป็นระบบพีชคณิตที่มีตัวดำเนินการ (Operator) และตัวแปรที่กระทำอยู่บนเซต V_2 เมื่อ V_2 = {0, 1} ส่วนตรรกะไตรภาคเป็นตรรกะที่มีตัวดำเนินการและตัวแปรที่กระทำอยู่บนเซต V_3 เมื่อ V_3 = {0, ½, 1} โดยที่ $0 \leq ½ \leq 1$ ตัวดำเนินการประกอบด้วย AND (·), OR (∨) และ Inverter (~) [4-6] โดยที่

$X \cdot Y = \min(X, Y)$

$X \vee Y = \max(X, Y)$

$\sim X = 1 - X$

และมีตารางค่าความจริง (Truth Table) ดังตารางที่ 2.1

ตารางที่ 2.1 ตารางค่าความจริงของตัวดำเนินการบนตรรกะไตรภาคชนิดบี

(ก) AND (·)

AND	0	½	1
0	0	0	0
½	0	½	½
1	0	½	1

(ข) OR (∨)

OR	0	½	1
0	0	½	1
½	½	½	1
1	1	1	1

(ค) Inverter (~)

INV	0	½	1
~ X	1	½	0

ผลจากนิยามและความหมายของตัวดำเนินการต่างๆ ของตรรกะไตรภาคชนิดบี ทำให้ตรรกะไตรภาคชนิดมีคุณสมบัติของกฎต่างๆ เช่นเดียวกับตรรกะฐานสอง ดังตารางที่ 2.2 ยกเว้นกฎส่วนเติมเต็ม (Complementaly laws) : $x \vee \sim x = 1,\ x \bullet \sim x = 0$

ตารางที่ 2.2 กฎการดำเนินการที่ใช้กับตรรกะไตรภาคชนิดบี

ชื่อกฎ	ตัวอย่าง
1. กฎการสลับที่ (Commutative laws)	$x \vee y = y \vee x,\ x \bullet y = y \bullet x$
2. กฎการจัดหมู่ (Associative laws)	$x \vee (y \vee z) = (x \vee y) \vee z,$ $x \bullet (y \bullet z) = (x \bullet y) \bullet z$
3. กฎการดูดซับ (Absorption laws)	$x \vee (x \bullet y) = x,\ (x \vee y) \bullet x = x$
4. กฎการกระจาย (Distributive laws)	$x \bullet (y \vee z) = (x \bullet y) \vee (x \bullet z),$ $x \vee (y \bullet z) = (x \vee y) \bullet (x \vee z)$
5. กฎค่าซ้ำ (Idempotent laws)	$x \vee x = x,\ x \bullet x = x$
6. กฎสองนิเสธ (Double negation law)	$\sim(\sim x) = x$
7. กฎของเดอมอแกน (DeMorgan's law)	$\sim(x \vee y) = \sim x \bullet \sim y,$ $\sim(x \bullet y) = \sim x \vee \sim y$
8. กฎของจำนวนมากสุดและจำนวนน้อยสุด (The least element and the greatest element)	$x \bullet 1 = x,\ x \bullet 0 = 0,$ $x \vee 1 = 1,\ x \vee 0 = x$
9. กฎของคลีน (Kleene's law)	$(x \bullet \sim x) \vee y \vee \sim y = y \vee \sim y,$ $x \bullet \sim x \bullet (y \vee \sim y) = x \bullet \sim x$
10. กฎค่ากึ่งกลาง (center)	$\sim ½ = ½$

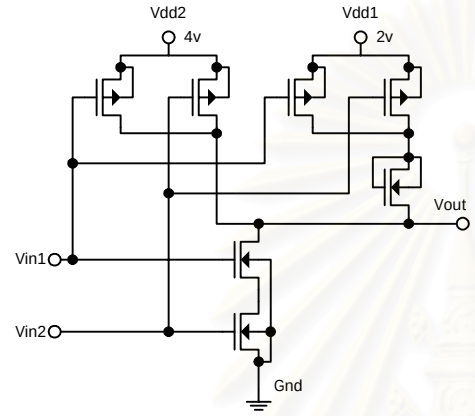
เนื่องจากคุณสมบัติของตรรกะไตรภาคชนิดบีมีตัวดำเนินการอยู่บนเซตที่ประกอบด้วยค่าสามค่า ดังนั้นเราสามารถนำมันไปใช้แทนรหัสรางคู่ซึ่งมีรหัส 4 รหัสแต่ใช้เพียง 3 รหัสได้ โดยใช้ค่า '0' แทนค่าข้อมูล '0', '1' แทนค่าข้อมูล '1' และ '½' แทนตัวแบ่งรอบการทำงาน (Spacer) ดังนั้นเมื่อข้อมูลมีจำนวน n บิต เราจะใช้สายสัญญาณเป็นจำนวน n เส้น

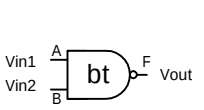
เมื่อกำหนดให้ $V_2^n = (a_1 a_2 \ldots a_n)$ โดยที่ $a_n \in V_2$

$V_3^n = (a_1 a_2 \ldots a_n)$ โดยที่ $a_n \in V_3$

$S = (1/2 \ldots 1/2)$

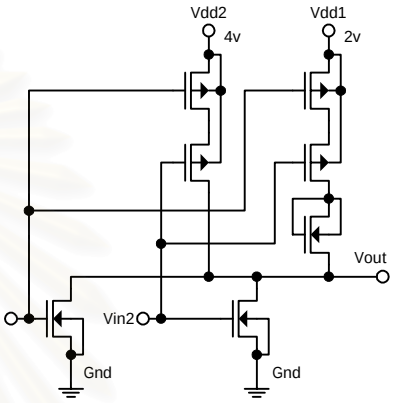
ข้อมูลขนาด n บิตจะเป็นรหัสตรงก็ต่อเมื่อสายทุกเส้นมีค่าเป็น 0 หรือ 1 หรืออยู่ในเซต V_2^n และเป็นตัวแบ่งรอบการทำงานเมื่อรหัสข้อมูลมีค่าเป็น S ส่วนเซตของรหัสที่เหลือ (V_3^n - V_2^n - S) เป็นรหัสไม่ตรง การทำงานของตรรกะไตรภาคชนิดบีใช้วิธีการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์เช่นเดียวกับการทำงานของวงจรรางคู่ดังรูปที่ 2.8

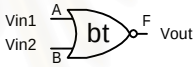




B \ A	0	1/2	1
0	1	1	1
1/2	1	1/2	1/2
1	1	1/2	0

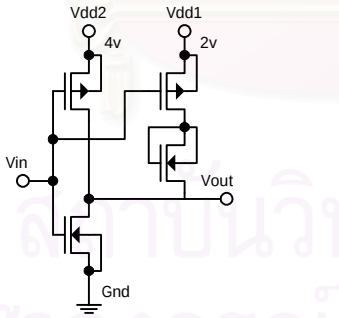
(ก) เกตแนนด์ที่ใช้ตรรกะไตรภาคชนิดบี

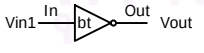




B \ A	0	1/2	1
0	1	1/2	0
1/2	1/2	1/2	0
1	0	0	0

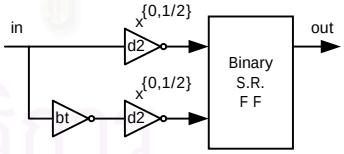
(ข) เกตนอร์ที่ใช้ตรรกะไตรภาคชนิดบี

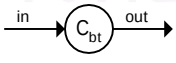




In	0	1/2	1
Out	1	1/2	0

(ค) ตัวผกผันที่ใช้ตรรกะไตรภาคชนิดบี



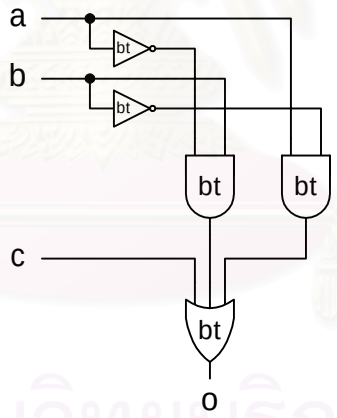


In \ Out	0	1/2	1
0	0	0	1
1	0	1	1

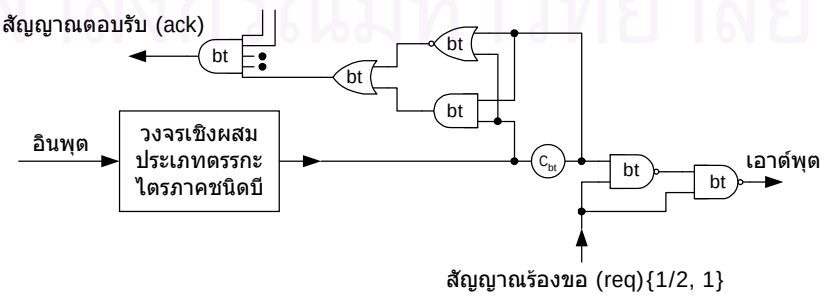
(ง) อุปกรณ์ชนิดซีที่อินพุตเป็นตรรกะไตรภาค และเอาต์พุตเป็นตรรกะฐานสอง

รูปที่ 2.13 วงจรระดับทรานซิสเตอร์ของเกตตรรกะไตรภาคชนิดบี

ผลจากลักษณะตัวดำเนินการที่สามารถใช้กับกฎต่างๆ ได้เช่นเดียวกับกฎที่ใช้กับตรรกะฐานสอง ทำให้การออกแบบวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีสามารถทำได้โดยใช้วิธีการออกแบบวงจรเชิงผสมประเภทตรรกะฐานสอง (Binary Logic) แต่เกตที่ใช้เป็นเกตสำหรับตรรกะไตรภาคชนิดบีแทนการใช้เกตทั่วไป วงจรระดับทรานซิสเตอร์ของเกตแนนด์, เกตนอร์, ตัวผกผัน และอุปกรณ์ชนิดซีสำหรับตรรกะไตรภาคชนิดบีมีลักษณะดังรูปที่ 2.13 เกตแนนด์, เกตนอร์ และตัวผกผันมีอินพุตและเอาต์พุตเป็นตรรกะไตรภาค สำหรับอุปกรณ์ชนิดซีจะรับอินพุตเป็นตรรกะไตรภาคและให้เอาต์พุตเป็นตรรกะฐานสอง สัญลักษณ์ตัวผกผันที่มี $X^{\{0,1/2\}}$ กำกับหมายถึงอุปกรณ์ที่ให้เอาต์พุตเป็นค่าตรรกะ 1 เมื่อได้รับอินพุตเป็นค่าที่อยู่ในเซตที่กำกับอยู่เหนือตัวอักษร X และให้ค่าเอาต์พุตเป็นค่าตรรกะ 0 เมื่อได้รับอินพุตค่าอื่น เช่น $X^{\{0,1/2\}}$ ให้ค่าเอาต์พุตเป็น 1 เมื่ออินพุตเป็น 0 หรือ ½ และให้เอาต์พุตเป็น 0 เมื่ออินพุตเป็น 1 จากเกตในรูปที่ 2.15 เราสามารถสร้างเป็นวงจรตรรกะไตรภาคชนิดบี f=ab'+a'b+c ได้ดังรูปที่ 2.14 และมีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการทำงานที่เอาต์พุตดังรูปที่ 2.15 วงจรตอบรับดังกล่าวไม่สามารถประกันการเกิดฮาซาร์ดเนื่องจากความหน่วงได้ สำหรับการออกแบบวงจรตอบรับที่ป้องกันการเกิดฮาซาร์ดเนื่องจากความหน่วงนั้นจะกล่าวในบทถัดไป



รูปที่ 2.14 วงจรตรรกะไตรภาคชนิดบี f=ab'+a'b+c



รูปที่ 2.15 วงจรตอบรับสำหรับตรรกะไตรภาคชนิดบีของ Nagata

บทที่ 3
การวิเคราะห์ฮาซาร์ดและการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสม ประเภทตรรกะไตรภาคชนิดบี

สิ่งสำคัญในการออกแบบวงจรเชิงเลขคือความถูกต้องในการประมวลผลข้อมูลที่ป้อนให้แก่วงจร ดังนั้นเพื่อป้องกันความผิดพลาดที่สามารถเกิดขึ้นแก่วงจร จึงต้องมีการวิเคราะห์สาเหตุที่ทำให้วงจรทำงานผิดพลาด และวิธีการป้องกันไม่ให้เกิดความผิดพลาดนั้นแก่วงจร เนื้อหาในบทนี้เสนอการวิเคราะห์ฮาซาร์ดบนตรรกะไตรภาคชนิดบี และการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะชนิดบีเพื่อป้องกันการเกิดฮาซาร์ดเนื่องจากความหน่วง

3.1 การวิเคราะห์ฮาซาร์ดบนตรรกะไตรภาคชนิดบี

ฮาซาร์ด (Hazard) [11] คือปรากฏการณ์ที่วงจรทำงานผิดพลาดโดยเกิดการสลับค่าชั่วคราวของสัญญาณที่ไม่ต้องการในเวลาชั่วขณะหนึ่ง และการเปลี่ยนแปลงนั้นส่งผลกระทบต่อเอาต์พุต พฤติกรรมการสลับค่าชั่วคราวของสัญญาณมีสาเหตุมาจากความแตกต่างของความหน่วงแพร่กระจาย (Propagation Delay) ดังนั้นการวิเคราะห์ฮาซาร์ดจึงเป็นการวิเคราะห์พฤติกรรมการเปลี่ยนแปลงสัญญาณอินพุตของเกตและวงจรที่ส่งผลกระทบต่อการเปลี่ยนแปลงสัญญาณที่เอาต์พุตของเกตและวงจร หัวข้อนี้แบ่งการวิเคราะห์ฮาซาร์ดที่เกิดบนตรรกะไตรภาคชนิดบีออกเป็นสองชนิด คือฮาซาร์ดของตรรกะ และฮาซาร์ดเนื่องจากความหน่วง

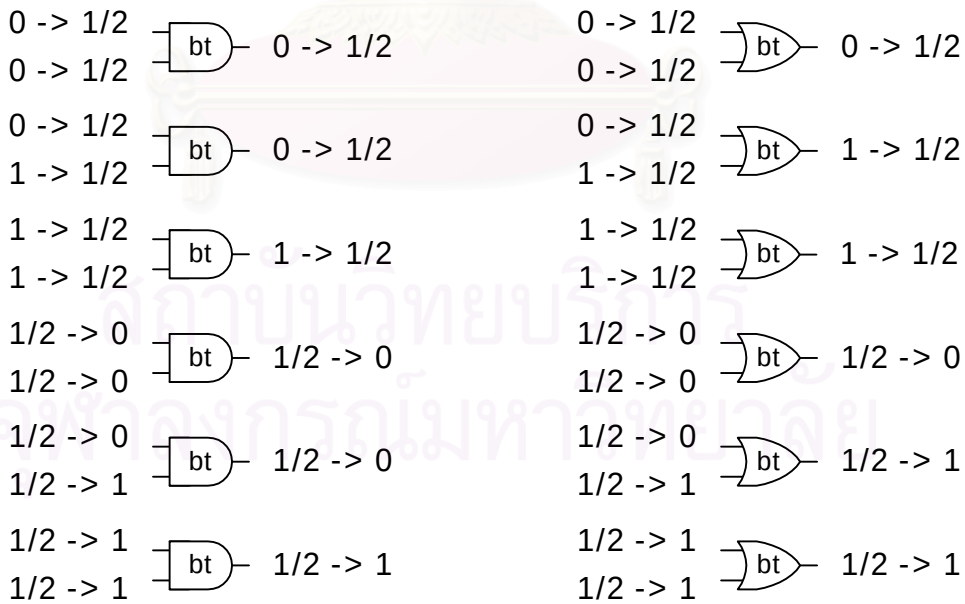
3.1.1 ฮาซาร์ดของตรรกะ (Logic Hazard)

เมื่อพิจารณาการเปลี่ยนแปลงสัญญาณที่เอาต์พุตของเกตที่เป็นตรรกะไตรภาคชนิดบีเมื่อเกิดการเปลี่ยนแปลงสัญญาณอินพุตแบบต่างๆ สำหรับเกตที่มีอินพุตจำนวนสองอินพุต เราจะพบว่าเมื่อเกิดการเปลี่ยนแปลงสัญญาณที่อินพุตหนึ่งอินพุต (Single Input Changed) เกตสามารถทำงานโดยให้เอาต์พุตของเกตเป็นค่าที่ถูกต้องได้ ในกรณีที่เกิดการเปลี่ยนแปลงสัญญาณที่อินพุตหลายอินพุต (Multiple Input Changed) เราจะพบว่าเมื่อพฤติกรรมการเปลี่ยนแปลงอินพุตเกิดการเปลี่ยนแปลงสัญญาณทุกอินพุต จากค่าข้อมูล (0 หรือ 1) เป็นตัวแบ่งรอบการทำงาน ($\frac{1}{2}$) หรือจากตัวแบ่งรอบการทำงานเป็นค่าข้อมูล เอาต์พุตของเกตจะให้เอาต์พุตที่ถูกต้องโดยไม่เกิดฮาซาร์ดดังรูปที่ 3.1 แต่เมื่ออินพุตอย่างน้อยหนึ่งอินพุตเกิดการเปลี่ยนแปลงสัญญาณจากค่าข้อมูลค่าหนึ่งเป็นค่าข้อมูลอื่น (0 เปลี่ยนเป็น 1 หรือ 1 เปลี่ยนเป็น 0) ในขณะที่อินพุตที่เหลือเกิดการเปลี่ยนแปลงจากค่าข้อมูลเป็นตัวแบ่งรอบการทำงาน จะสามารถทำให้เกิดฮาซาร์ดขึ้นได้ รูปที่ 3.2 แสดงกรณีที่อินพุตอย่างน้อยหนึ่งอินพุตเกิดการเปลี่ยนแปลงสัญญาณจากค่าข้อมูลค่าหนึ่ง

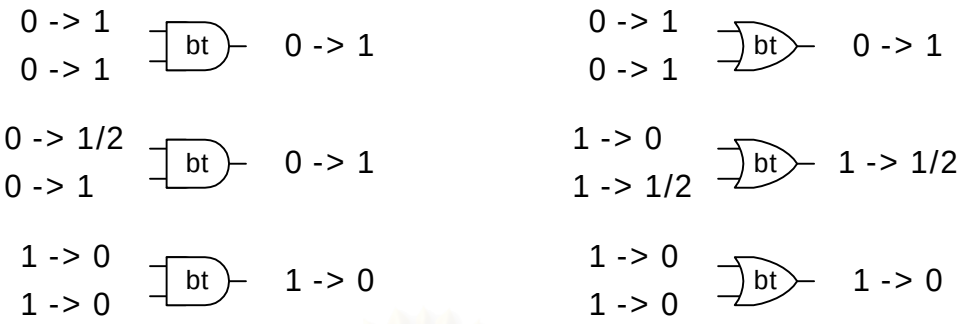
เป็นค่าข้อมูลอื่นโดยไม่เกิดฮาซาร์ด และรูปที่ 3.3 แสดงกรณีที่อินพุตอย่างน้อยหนึ่งอินพุตเกิดการเปลี่ยนแปลงสัญญาณจากค่าข้อมูลค่าหนึ่งเป็นค่าข้อมูลอื่นโดยเกิดฮาซาร์ด

สำหรับเกตที่มีอินพุตจำนวน n อินพุตโดยที่ n>2 กรณีที่อินพุตทุกอินพุตเกิดการเปลี่ยน-แปลงสัญญาณจากค่าข้อมูลเป็นตัวแบ่งรอบการทำงาน หรือเกิดการเปลี่ยนแปลงจากตัวแบ่งรอบการทำงานเป็นค่าข้อมูล วงจรจะไม่เกิดฮาซาร์ดของตรรกะ แต่ในกรณีที่อินพุตอย่างน้อยหนึ่งอินพุตเกิดการเปลี่ยนแปลงสัญญาณจากค่าข้อมูลค่าหนึ่งเป็นค่าข้อมูลอื่นในขณะที่อินพุตอื่นเกิดการเปลี่ยนแปลงสัญญาณจากค่าข้อมูลเป็นตัวแบ่งรอบการทำงาน วงจรสามารถเกิดฮาซาร์ดของตรรกะได้เช่นเดียวกับเกตที่มีจำนวนอินพุตสองอินพุต

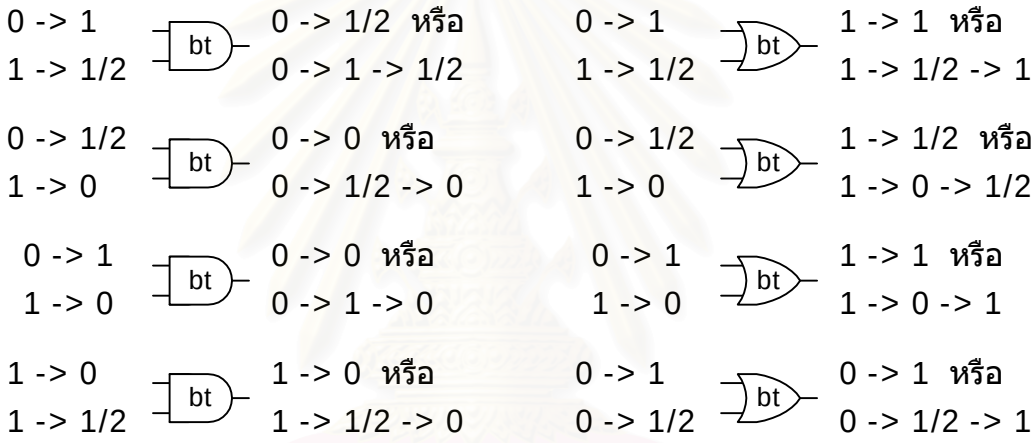
เมื่อวิเคราะห์วิธีการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์บนตรรกะไตรภาคชนิดบี เราจะพบว่าลักษณะสัญญาณอินพุตของเกตมีพฤติกรรมการเปลี่ยนแปลงสัญญาณอินพุตทุกอินพุตของเกตในวงจร จากค่าข้อมูลเป็นตัวแบ่งรอบการทำงาน หรือจากตัวแบ่งรอบการทำงานเป็นค่าข้อมูลเท่านั้น ซึ่งตรงกับกรณีของการเปลี่ยนแปลงสัญญาณอินพุตที่ไม่ก่อให้เกิดฮาซาร์ดดังรูปที่ 3.1 ดังนั้นการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์สามารถรับประกันได้ว่าไม่เกิดฮาซาร์ดของตรรกะบนตรรกะไตรภาคชนิดบี



รูปที่ 3.1 พฤติกรรมการเปลี่ยนแปลงสัญญาณเอาต์พุต กรณีสัญญาณอินพุตเปลี่ยนจากค่าข้อมูลเป็นตัวแบ่งรอบการทำงานและจากตัวแบ่งรอบการทำงานเป็นค่าข้อมูล



รูปที่ 3.2 พฤติกรรมการเปลี่ยนแปลงสัญญาณเอาต์พุต กรณีสัญญาณอินพุตอย่างน้อยหนึ่งอินพุตเปลี่ยนจากค่าข้อมูลค่าหนึ่งเป็นค่าข้อมูลอื่นแต่ไม่เกิดฮาซาร์ด



รูปที่ 3.3 พฤติกรรมการเปลี่ยนแปลงสัญญาณเอาต์พุต กรณีสัญญาณอินพุตอย่างน้อยหนึ่งอินพุตเปลี่ยนจากค่าข้อมูลค่าหนึ่งเป็นค่าข้อมูลอื่นแล้วเกิดฮาซาร์ด

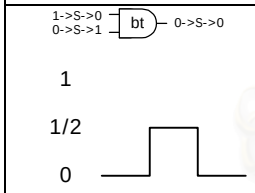
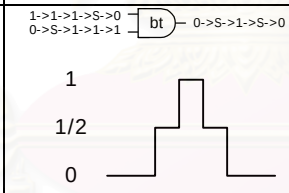
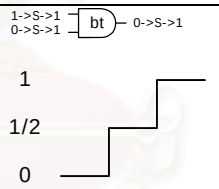
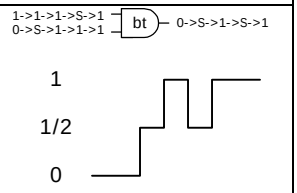
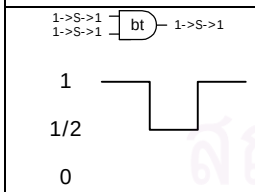
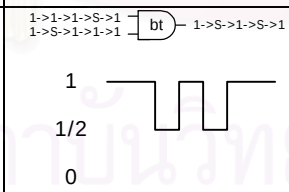
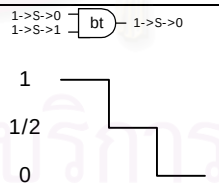
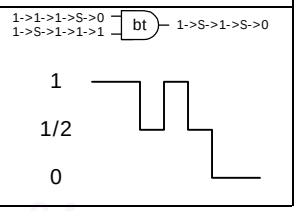
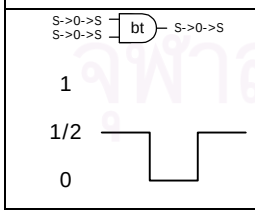
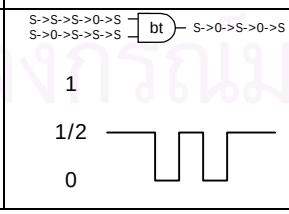
3.1.2 ฮาซาร์ดเนื่องจากความหน่วง (Delay Hazard)

เนื่องจากเส้นทางส่งผ่านข้อมูลจากอินพุตสู่เอาต์พุตของวงจรเชิงผสมแต่ละเส้นทางมีความหน่วงไม่เท่ากัน เมื่อเส้นทางส่งผ่านข้อมูลที่มีความหน่วงน้อยส่งผลทำให้เอาต์พุตเปลี่ยน เอาต์พุตของวงจรจะเกิดการเปลี่ยนแปลงค่าก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน จากพฤติกรรมของวงจรดังกล่าวเมื่อพิจารณาการทำงานของระบบที่สิ่งแวดล้อมทำหน้าที่ตรวจสอบการสิ้นสุดการทำงานของวงจรที่เอาต์พุตของวงจรเพื่อส่งอินพุตใหม่ให้วงจร ในกรณีที่ระบบมีการทำงานแบบภาวะแวดล้อมมูลฐาน สิ่งแวดล้อมจะทำงานช้ากว่าวงจรทำให้สิ่งแวดล้อมส่งอินพุตใหม่ให้วงจรเมื่อวงจรอยู่ในสภาวะสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในแล้วทำให้วงจรทำงานได้อย่างถูกต้อง แต่ในกรณีที่ระบบมีการทำงานแบบภาวะแวดล้อมรับเข้าส่งออกสิ่งแวดล้อมจะ

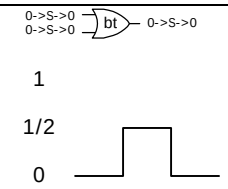
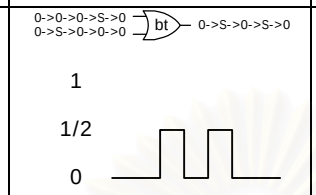
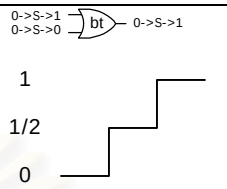
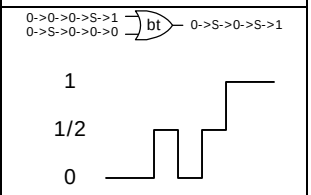
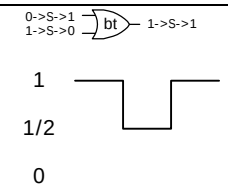
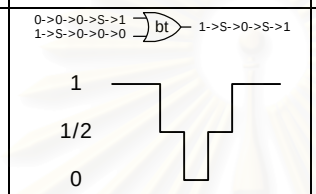
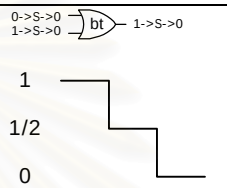
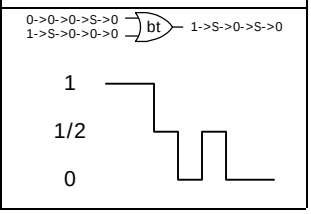
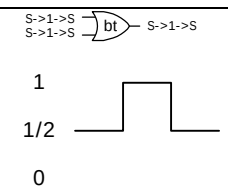
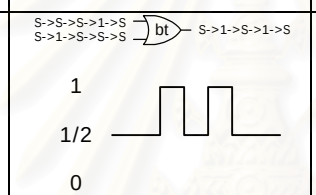
ทำงานเร็วกว่าวงจร ดังนั้นเมื่อสิ่งแวดล้อมตรวจสอบได้ว่าวงจรให้เอาต์พุตใหม่ออกมาสิ่งแวดล้อมจะส่งอินพุตใหม่ไปให้วงจร ซึ่งเป็นช่วงเวลาที่วงจรยังไม่สิ้นสุดการเปลี่ยนแปลงสัญญาณภายในจึงเป็นสาเหตุทำให้เกิดการแทรกสอดของอินพุตเก่าและอินพุตใหม่ และส่งผลทำให้วงจรทำงานผิดพลาด ฮาซาร์ดดังกล่าวเรียกว่าฮาซาร์ดเนื่องจากความหน่วง

เมื่อวิเคราะห์ความหน่วงบนตรรกะไตรภาคชนิดบี เราจะพบว่าวงจรสามารถเกิดฮาซาร์ดได้เมื่อมีอินพุตใดอินพุตหนึ่งของเกตที่มีอินพุตมากกว่าหนึ่งอินพุตเกิดการการเปลี่ยนแปลงอย่างน้อยสองครั้งในขณะที่อินพุตอื่นของเกตนั้นยังไม่เกิดการเปลี่ยนแปลงสัญญาณ ลักษณะของฮาซาร์ดที่เกิดขึ้นสามารถจำแนกออกเป็นสองลักษณะคือ ฮาซาร์ดแบบสถิต (Static Hazard) และฮาซาร์ดแบบพลวัต (Dynamic Hazard) ฮาซาร์ดแบบสถิตและฮาซาร์ดแบบพลวัตสำหรับตรรกะไตรภาคชนิดบีที่เกิดบนเกตแอนด์และเกตออร์มีลักษณะดังตารางที่ 3.1 และ 3.2 ตามลำดับ

ตารางที่ 3.1 ลักษณะการเกิดฮาซาร์ดเนื่องจากความหน่วงของเกตแอนด์สำหรับตรรกะไตรภาคชนิดบี

ฮาซาร์ดแบบสถิต		ฮาซาร์ดแบบพลวัต	
กรณีไม่เกิดฮาซาร์ด	กรณีเกิดฮาซาร์ด	กรณีไม่เกิดฮาซาร์ด	กรณีเกิดฮาซาร์ด
			
			
			

ตารางที่ 3.2 ลักษณะการเกิดฮาซาร์ดเนื่องจากความหน่วงของเกตออร์สำหรับตรรกะไตรภาคชนิดบี

ฮาซาร์ดแบบสถิต		ฮาซาร์ดแบบพลวัต	
กรณีไม่เกิดฮาซาร์ด	กรณีเกิดฮาซาร์ด	กรณีไม่เกิดฮาซาร์ด	กรณีเกิดฮาซาร์ด
			
			
			

จากวงจรตัวอย่าง f=(IN4·IN3)+(IN2·IN1)+IN0 ดังรูปที่ 3.4 โดยกำหนดให้

gd = ค่าความหน่วงของเกต

wd = ค่าความหน่วงของสาย

เมื่ออินพุตของวงจรมีพฤติกรรมการเปลี่ยนแปลงสัญญาณอินพุตดังตารางที่ 3.3 โดยกำหนดให้

U = ไม่ระบุค่าสัญญาณ (Undefined value)

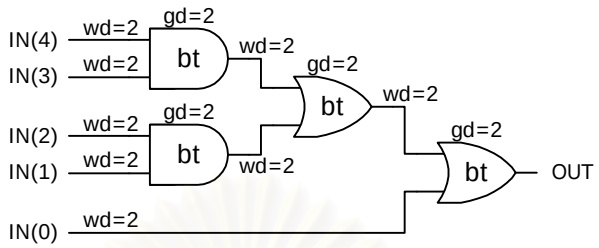
S = ตัวแบ่งรอบการทำงาน (Spacer)

1 = ค่าตรรกะ 1

0 = ค่าตรรกะ 0

หลังจากวงจรอยู่ในสภาวะสงบนิ่งในขั้นว่างจนกระทั้งเวลาที่ 24 นาโนวินาที อินพุตของวงจรจะเกิดการเปลี่ยนแปลงสัญญาณเป็นขั้นทำงาน ซึ่งส่งผลให้เอาต์พุตของวงจรเปลี่ยนไปสู่ขั้นทำงานโดยที่สัญญาณภายในวงจรยังไม่สิ้นสุดการเปลี่ยนแปลงสัญญาณ เมื่อสิ่งแวดล้อมทำงานเร็วกว่าและส่งอินพุตใหม่ซึ่งเป็นอินพุตขั้นว่างมายังวงจร ณ เวลา 30 นาโนวินาที จะส่งผลให้เกิดการเปลี่ยน-แปลงเอาต์พุตของวงจรโดยที่สัญญาณภายในวงจรบางเส้นยังอยู่ในขั้นทำงาน เมื่อสิ่งแวดล้อมส่งอินพุตขั้นทำงานใหม่มายังวงจร ณ เวลา 36 นาโนวินาที จะเป็นผลให้เกิดกรณีเกตคำนวณค่าของ

อินพุตขั้นทำงานเก่ากับอินพุตขั้นทำงานใหม่ ซึ่งทำให้เกิดฮาซาร์ดที่เอาต์พุตของวงจร ณ เวลา 40 นาโนวินาที



รูปที่ 3.4 ตัวอย่างวงจรที่เกิดฮาซาร์ดเนื่องจากความหน่วง

ตารางที่ 3.3 ตัวอย่างอินพุตที่ก่อให้เกิดฮาซาร์ดสำหรับวงจรตัวอย่างในรูปที่ 3.4

เวลา (ns)	อินพุต (IN4-0)	เอาต์พุต (OUT)
0	UUUUU	U
10	SSSSS	U
22	SSSSS	S
24	00001	S
28	00001	1
30	SSSSS	1
34	SSSSS	S
36	00110	S
40	00110	0
42	SSSSS	S
44	00001	S
48	00001	1

ผลจากการวิเคราะห์ลักษณะของฮาซาร์ดทำให้สรุปได้ว่าวงจรตรรกะไตรภาคชนิดบีนั้นไม่เกิดฮาซาร์ดของตรรกะ เนื่องจากตรรกะไตรภาคชนิดบีใช้วิธีการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์ทำให้เกิดการเปลี่ยนแปลงสัญญาณข้อมูลของวงจรจากค่าข้อมูลเป็นตัวแบ่งรอบการทำงาน หรือจากตัวแบ่งรอบการทำงานเป็นค่าข้อมูลซึ่งเป็นกรณีของการเปลี่ยนแปลงสัญญาณที่ไม่ก่อให้เกิดฮาซาร์ด แต่วงจรตรรกะไตรภาคชนิดบีสามารถเกิดฮาซาร์ดเนื่องจากความหน่วงได้เมื่อระบบมีการทำงานแบบภาวะแวดล้อมรับเข้าส่งออก โดยที่สิ่งแวดล้อมทำหน้าที่ตรวจสอบการสิ้นสุดการทำงานที่เอาต์พุตของวงจร

3.2 การออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี

สาเหตุหลักของฮาซาร์ดเนื่องจากความหน่วงอยู่ที่ว่าสิ่งแวดล้อมส่งอินพุตใหม่มาให้วงจรในขณะที่วงจรยังไม่สิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน ดังนั้นการสร้างวงจรตอบรับที่สิ่งแวดล้อมเพื่อตรวจสอบการสิ้นสุดการทำงานของวงจรที่ใช้แบบจำลองการทำงานสิ่งแวดล้อม

แบบภาวะรับเข้าส่งออกจึงไม่เหมาะสม ปัญหาดังกล่าวสามารถแก้ไขได้โดยสร้างวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในวงจรแทนการตรวจสอบการสิ้นสุดการทำงานที่เอาต์พุตของวงจร ในหัวข้อนี้แบ่งการออกแบบวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีออกเป็นสี่ส่วนคือ การตรวจสอบขั้นการทำงาน, การออกแบบวงจรตอบรับโดยใช้อุปกรณ์ตรวจสอบขั้นการทำงาน การเลือกสายสัญญาณจากวงจรเชิงผสมเพื่อตรวจสอบขั้นการทำงาน และการนำสัญญาณตอบรับไปใช้งาน

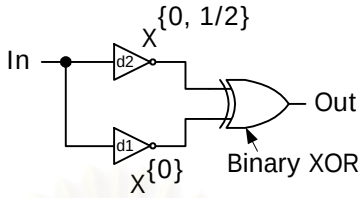
3.2.1 การตรวจสอบขั้นการทำงาน

วงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีใช้วิธีการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์ ทำให้พฤติกรรมการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสมเกิดการเปลี่ยนแปลงเป็นขั้นทำงานและขั้นว่างสลับกันในรอบการทำงานแต่ละรอบ โดยที่สายสัญญาณอินพุตทุกเส้น สายสัญญาณภายในทุกเส้น และสายสัญญาณเอาต์พุตทุกเส้นมีค่าเป็นค่าข้อมูล 0 หรือ 1 เมื่อสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของขั้นทำงาน และสายสัญญาณทุกเส้นของวงจรมีค่าเป็นตัวแบ่งรอบการทำงาน เมื่อสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของขั้นว่าง ดังนั้นเพื่อป้องกันการแทรกสอดของสัญญาณข้อมูลจึงจำเป็นต้องมีส่วนที่ทำหน้าที่ตรวจสอบสายสัญญาณภายในแต่ละเส้นว่ามีสัญญาณเป็นขั้นใด

เนื่องจากขั้นการทำงานแบ่งออกเป็นสองขั้นแต่สถานะหรือค่าตรรกะมีสามค่า วงจรหรืออุปกรณ์ที่ใช้ในการตรวจสอบขั้นการทำงานจึงต้องมีลักษณะที่ให้เอาต์พุตเป็นค่าเพียงสองค่าจากค่าสามค่าของตรรกะ โดยวงจรหรืออุปกรณ์ดังกล่าวต้องให้เอาต์พุตเป็นค่าใดค่าหนึ่งในค่าสามค่า 0, ½ และ 1 เมื่อสายสัญญาณภายในที่ตรวจสอบเป็นค่าข้อมูล (0 หรือ 1) และให้เอาต์พุตเป็นค่าใดค่าหนึ่งของค่าที่เหลือเมื่อสัญญาณภายในที่ตรวจสอบเป็นตัวแบ่งรอบการทำงาน ตัวอย่างเช่น ให้เอาต์พุตเป็น 1 เมื่อสายที่ตรวจสอบมีค่าเป็นตัวแบ่งรอบการทำงาน และให้เอาต์พุตเป็น 0 เมื่อสายที่ตรวจสอบมีค่าเป็นค่าข้อมูล

จากลักษณะของการตรวจสอบขั้นการทำงานดังกล่าวเมื่อพิจารณากรณีที่การตรวจสอบขั้นการทำงานให้เอาต์พุตเป็น 1 เมื่อสายที่ตรวจสอบเป็นตัวแบ่งรอบการทำงาน และให้เอาต์พุตเป็น 0 เมื่อสายที่ตรวจสอบเป็นค่าข้อมูล เราสามารถสร้างวงจรที่ทำหน้าที่ตรวจสอบขั้นการทำงานในระดับเกตได้ดังรูปที่ 3.5 แต่เนื่องจากวงจรดังกล่าวมีขนาดใหญ่โดยเฉพาะเมื่อเทียบกับการใช้

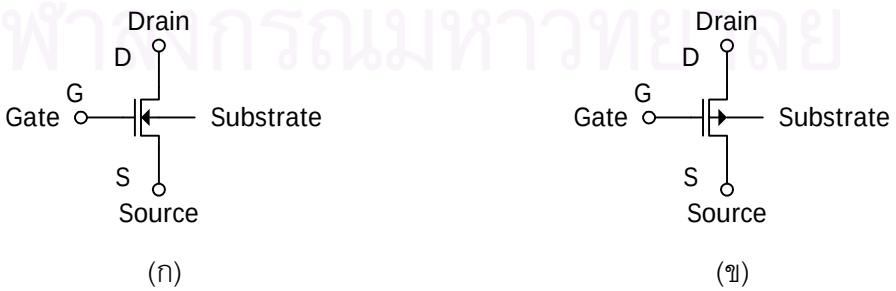
เกตออร์ที่ใช้ในการตรวจสอบขั้นการทำงานของคู่เกตแอนด์ออร์แต่ละคู่สำหรับวงจรรางคู่ ดังนั้นวงจรดังกล่าวจึงไม่เหมาะสมที่จะใช้ตรวจสอบขั้นการทำงานของวงจรตรรกะไตรภาคชนิดบี



In	0	1/2	1
Out	0	1	0

รูปที่ 3.5 วงจรส่วนตรวจสอบขั้นการทำงานในระดับเกต

การออกแบบวงจรตอบรับในส่วนตรวจสอบขั้นการทำงานของสายแต่ละเส้นภายในวงจรด้วยการใช้วงจรในรูปที่ 3.5 จะทำให้วงจรมีขนาดใหญ่เนื่องจากวงจรดังกล่าวอาศัยการเชื่อมต่อในระดับเกตของเกตพื้นฐานต่างๆ มากกว่าหนึ่งตัวเข้าด้วยกัน และวงจรมีขนาดใหญ่กว่าเกตออร์เพียงหนึ่งตัวเสมอ ดังนั้นเพื่อให้ส่วนตรวจสอบขั้นการทำงานของสายแต่ละเส้นมีขนาดเล็กกว่าเกตออร์ จึงต้องพิจารณาการออกแบบในระดับที่เป็นการเชื่อมต่อของอุปกรณ์พื้นฐานที่มีขนาดเล็กกว่าเกตออร์ ทำให้การออกแบบวงจรในระดับทรานซิสเตอร์ถูกนำขึ้นมาพิจารณาเพื่อออกแบบอุปกรณ์ตรวจสอบขั้นการทำงานโดยใช้ทรานซิสเตอร์ชนิดมอสช่องพี (*p-channel* MOS transistor : *p*MOS) และทรานซิสเตอร์ชนิดมอสช่องเอ็น (*n-channel* MOS transistor : *n*MOS) เป็นอุปกรณ์พื้นฐานในการออกแบบวงจร วิทยานิพนธ์นี้ใช้สัญลักษณ์ในรูปที่ 3.6(ก) แทนทรานซิสเตอร์ชนิดมอสช่องเอ็น และใช้สัญลักษณ์ในรูปที่ 3.6(ข) แทนทรานซิสเตอร์ชนิดมอสช่องพี ทรานซิสเตอร์ชนิดมอสช่องเอ็นและทรานซิสเตอร์ชนิดมอสช่องพีประกอบด้วยขั้ว 4 ขั้วคือเกต (Gate), ซอร์ส (Source), เดรน (Drain) และซับสเตรต (Substrate)

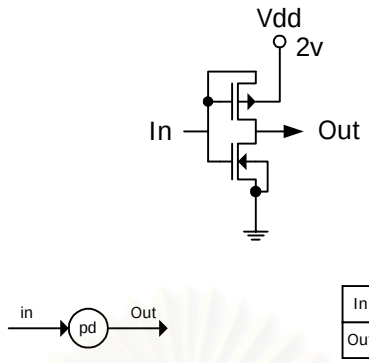


รูปที่ 3.6 ทรานซิสเตอร์ชนิดมอส

(ก) ทรานซิสเตอร์ชนิดมอสช่องเอ็น, (ข) ทรานซิสเตอร์ชนิดมอสช่องพี

เนื่องจากตรรกะไตรภาคชนิดบีเป็นตรรกะที่ประกอบด้วยค่าสามค่า และใช้แรงดันไฟฟ้าของวงจรเทียบกับค่าตรรกะ ดังนั้นวงจรจึงต้องประกอบด้วยแหล่งจ่ายแรงดัน 2 ระดับ คือ Vdd2 และ Vdd1 โดยที่แรงดัน Vdd2 แทนค่าตรรกะ 1, แรงดัน Vdd1 แทนค่าตรรกะ ½ และแรงดัน 0 โวลต์หรือกราวด์แทนค่าตรรกะ 0 วิทยานิพนธ์นี้ได้กำหนดให้ Vdd2 มีค่าเท่ากับ 4 โวลต์, Vdd1 มีค่าเท่ากับ 2 โวลต์ เมื่อพิจารณาพฤติกรรมของทรานซิสเตอร์ชนิดมอสช่องเอ็นและทรานซิสเตอร์ชนิดมอสช่องพีที่ทำงานอยู่ในย่านเชิงเส้น (Linear Region) ทรานซิสเตอร์ชนิดมอสช่องเอ็นและทรานซิสเตอร์ชนิดมอสช่องพีจะทำหน้าที่เหมือนตัวต้านทานภายใต้การควบคุมแรงดันที่เกต ทำให้ในภาวะการทำงานนี้ทรานซิสเตอร์ชนิดมอสมีลักษณะการทำงานคล้ายสวิตช์ โดยทรานซิสเตอร์ชนิดมอสช่องเอ็นจะมีคุณสมบัติเป็นสวิตช์เปิด (Open Switch) หรือไม่ให้กระแสไหลผ่าน เมื่อแรงดันที่เกตมีค่าน้อยกว่าแรงดันขีดเริ่มเปลี่ยนของทรานซิสเตอร์ชนิดมอสช่องเอ็น (*n*MOS Threshold Voltage : V_{thn}) และมีคุณสมบัติเป็นสวิตช์ปิด (Closed Switch) หรือให้กระแสไหลผ่าน เมื่อแรงดันที่เกตมีค่ามากกว่าแรงดันขีดเริ่มเปลี่ยนของทรานซิสเตอร์ชนิดมอสช่องเอ็น ทรานซิสเตอร์ชนิดมอสช่องเอ็นมีคุณสมบัติส่งผ่านค่าตรรกะ 0 ได้ดีโดยให้แรงดันที่เอาต์พุตเป็นค่า 0 โวลต์ เมื่อส่งผ่านค่าแรงดัน 0 โวลต์จากซอร์สไปยังเดรน และเมื่อส่งผ่านแรงดัน V_{dd} จะได้แรงดันที่เดรนสูงสุดเท่ากับ $V_{dd} - V_{thn}$ เมื่อกำหนดให้ V_{dd} คือค่าแรงดันไฟฟ้าค่าบวกที่มีค่ามากกว่า V_{thn} ดังนั้นเมื่อค่าแรงดันขีดเริ่มเปลี่ยนของทรานซิสเตอร์ชนิดมอสช่องเอ็นมีค่ามากกว่า 2 โวลต์และน้อยกว่า 4 โวลต์ ($2v \leq V_{thn} \leq 4v$) เอ็นมอสนั้นจะทำงานเมื่อแรงดันที่เกตเป็นค่าตรรกะ 1

สำหรับทรานซิสเตอร์ชนิดมอสช่องพีนั้น เมื่อแรงดันขีดเริ่มเปลี่ยนของทรานซิสเตอร์ชนิดมอสช่องพี (*p*MOS Threshold Voltage : V_{thp}) มีค่าเป็นบวก โดยที่ $|V_{thp}| < 2$ โวลต์ ทรานซิสเตอร์ชนิดนี้จะมีคุณสมบัติเป็นสวิตช์เปิดหรือไม่ให้กระแสไหลผ่านเมื่อแรงดันที่เกตมีค่ามากกว่า $V_{dd} + V_{thp}$ นั่นคือเมื่อเป็นค่าตรรกะ 1 และมีคุณสมบัติเป็นสวิตช์ปิดหรือให้กระแสไหลผ่านเมื่อแรงดันที่เกตมีค่าน้อยกว่า $V_{dd} + V_{thp}$ นั่นคือเมื่อเป็นค่าตรรกะ 0 หรือ ½ โดยที่ V_{dd} คือแรงดันที่ซับสเตรตของทรานซิสเตอร์ชนิดมอสช่องพี เมื่อทรานซิสเตอร์ชนิดมอสช่องพีมีคุณสมบัติเป็นสวิตช์ปิดจะมีคุณสมบัติการส่งผ่านแรงดัน V_{dd} จากซอร์สไปยังเดรนได้ดีเมื่อแรงดันที่ซอร์สเท่ากับ V_{dd} และเมื่อส่งผ่านแรงดัน 0 โวลต์ จะให้แรงดันที่เดรนต่ำสุดเท่ากับ $0 - |V_{thp}|$



รูปที่ 3.7 วงจรระดับทรานซิสเตอร์ของอุปกรณ์ตรวจสอบขั้นการทำงาน

จากพฤติกรรมของทรานซิสเตอร์ชนิดมอสช่องเอ็นและทรานซิสเตอร์ชนิดมอสช่องพี ดังกล่าวเราสามารถออกแบบวงจรส่วนตรวจสอบขั้นการทำงานของสายแต่ละเส้นในระดับทรานซิสเตอร์ได้ดังรูปที่ 3.7 โดยที่ V_{thn} = +2.5 โวลต์ และ V_{thp} = +0.5 โวลต์ ในทรานซิสเตอร์ชนิดมอสช่องพี เกตและซอร์สจะต่อกับอินพุต (In) ส่วนเดรนต่อกับเอาต์พุต (Out) และซับสเตรตต่อกับแรงดัน 2 โวลต์ (V_{dd} = 2v) สำหรับทรานซิสเตอร์ชนิดมอสช่องเอ็น เกตจะต่อกับอินพุต เดรนต่อกับเอาต์พุต ซอร์สและซับสเตรตต่อกับกราวด์หรือแรงดัน 0 โวลต์

จากรูปที่ 3.7 เมื่ออินพุตของวงจรเป็นค่าตรรกะ 1 หรือค่าแรงดัน 4 โวลต์ ทรานซิสเตอร์ชนิดมอสช่องเอ็นมีคุณสมบัติเป็นสวิตช์ปิด ทำให้แรงดันที่เดรนของทรานซิสเตอร์ชนิดมอสช่องเอ็นมีค่าเท่ากับ 0 โวลต์ ในขณะที่ทรานซิสเตอร์ชนิดมอสช่องพีมีคุณสมบัติเป็นสวิตช์เปิด ส่งผลให้เอาต์พุตของวงจรขึ้นอยู่กับค่าแรงดันที่เดรนของทรานซิสเตอร์ชนิดมอสช่องเอ็นเท่านั้น ซึ่งมีค่าเท่ากับ 0 โวลต์ หรือเป็นค่าตรรกะ 0

เมื่ออินพุตของวงจรเป็นค่าตรรกะ ½ หรือค่าแรงดัน 2 โวลต์ ทรานซิสเตอร์ชนิดมอสช่องเอ็นมีคุณสมบัติเป็นสวิตช์เปิด ในขณะที่ทรานซิสเตอร์ชนิดมอสช่องพีมีคุณสมบัติเป็นสวิตช์ปิด โดยที่ซอร์สของทรานซิสเตอร์ชนิดมอสช่องพีมีค่าแรงดันเป็น 2 โวลต์ทำให้เอาต์พุตของวงจรให้ค่าแรงดันเป็น 2 โวลต์หรือเป็นค่าตรรกะ ½

เมื่ออินพุตของวงจรเป็นค่าตรรกะ 0 หรือค่าแรงดัน 0 โวลต์ ทรานซิสเตอร์ชนิดมอสช่องเอ็นมีคุณสมบัติเป็นสวิตช์เปิด ในขณะที่ทรานซิสเตอร์ชนิดมอสช่องพีมีคุณสมบัติเป็นสวิตช์ปิด โดยที่ซอร์สของทรานซิสเตอร์ชนิดมอสช่องพีมีค่าแรงดันเป็น 0 โวลต์ ทำให้เอาต์พุตของวงจรให้ค่าแรงดันต่ำสุดเป็น 0 - | V_{thp} | หรือเป็นค่าตรรกะ 0

จากวงจรในรูปที่ 3.7 เมื่อนำไปจำลองการทำงานด้วยโปรแกรมสไปซ์ โดยกำหนดให้

ค่าตรรกะ 0 = แรงดัน 0 โวลต์

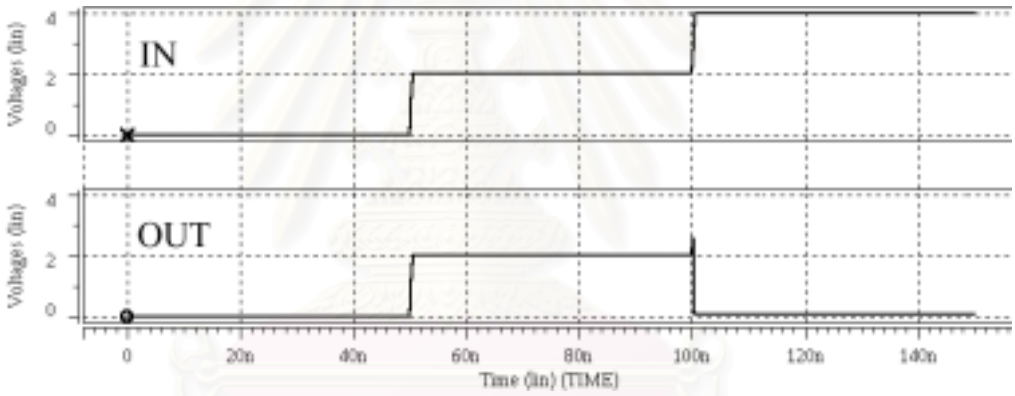
ค่าตรรกะ ½ = แรงดัน 2 โวลต์

ค่าตรรกะ 1 = แรงดัน 4 โวลต์

V_{thn} = +2.5 โวลต์

V_{thp} = +0.5 โวลต์

เราจะพบว่าผลการจำลองการทำงานเป็นดังรูปที่ 3.8 โดยวงจรให้เอาต์พุตเป็นค่าตรรกะ 0 เมื่ออินพุตเป็นค่าตรรกะ 0 หรือค่าตรรกะ 1 และให้ค่าเอาต์พุตเป็นค่าตรรกะ ½ เมื่ออินพุตเป็นค่าตรรกะ ½ เนื่องจากวงจรในรูปที่ 3.7 มีหน้าที่ตรวจสอบขั้นการทำงาน จึงเรียกวงจรดังกล่าวว่า *อุปกรณ์ตรวจสอบขั้นการทำงาน (Phase Detector Element : pd-element)*



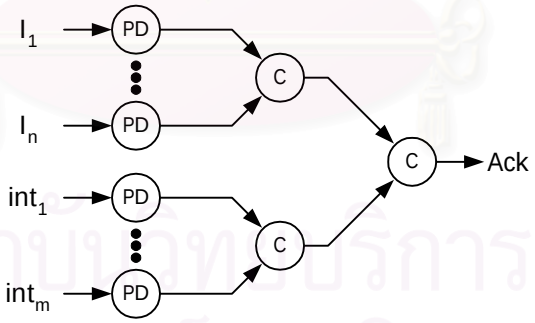
รูปที่ 3.8 ผลการจำลองการทำงานวงจรระดับทรานซิสเตอร์ของอุปกรณ์ตรวจสอบขั้นการทำงานด้วยโปรแกรมสไปซ์

3.2.2 การออกแบบวงจรตอบรับโดยใช้อุปกรณ์ตรวจสอบขั้นการทำงาน

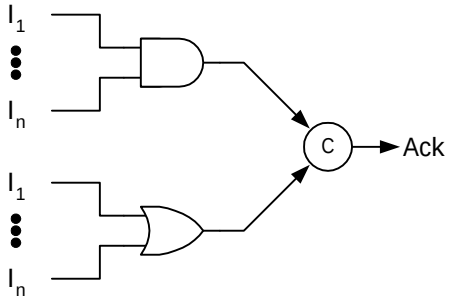
จากคุณสมบัติของอุปกรณ์ตรวจสอบขั้นการทำงานเราสามารถตรวจสอบขั้นการทำงานของสายสัญญาณได้ด้วยการแยกกิ่งของสายสัญญาณที่ต้องการตรวจสอบ แล้วนำมาเป็นอินพุตของอุปกรณ์ตรวจสอบขั้นการทำงานโดยใช้อุปกรณ์ตรวจสอบขั้นการทำงาน 1 ตัวต่อสายที่ต้องการตรวจสอบ 1 เส้น ดังนั้นวงจรตอบรับจึงต้องมีส่วนที่ทำหน้าที่รวมสัญญาณเอาต์พุตของอุปกรณ์ตรวจสอบขั้นการทำงานเพื่อสร้างสัญญาณตอบรับรวมของวงจร

เนื่องจากอุปกรณ์ตรวจสอบขั้นการทำงานให้เอาต์พุตเป็นค่าเพียงสองค่าคือ 0 และ ½ ซึ่งมีลักษณะเป็นตรรกะฐานสองโดยใช้การใช้ตรรกะ ½ เสมือนกับค่าตรรกะ 1 ของตรรกะฐานสอง

ดังนั้นวงจรส่วนที่ทำหน้ารวมสัญญาณเอาต์พุตของอุปกรณ์ตรวจสอบขั้นการทำงานเพื่อสร้างสัญญาณตอบรับรวมจึงสามารถใช้เกตหรืออุปกรณ์สำหรับวงจรตรรกะฐานสองได้ แต่เกตหรืออุปกรณ์นั้นต้องมีการปรับค่าแรงดันที่จ่ายให้วงจรและแรงดันขีดเริ่มเปลี่ยนของทรานซิสเตอร์ชนิดมอสช่องพีและทรานซิสเตอร์ชนิดมอสช่องเอ็นของเกตที่ใช้ให้เหมาะสม เพื่อให้วงจรสามารถทำงานในระดับแรงดันของค่าตรรกะ ½ แทนค่าระดับแรงดันของค่าตรรกะ 1 ได้ โดยปรับค่าแรงดัน V_{dd} = 2v และใช้ค่าแรงดันขีดเริ่มเปลี่ยนสำหรับทรานซิสเตอร์ชนิดมอสช่องเอ็นในช่วง $0v < V_{thn} < 2v$ และใช้ค่าแรงดันขีดเริ่มเปลี่ยนสำหรับทรานซิสเตอร์ชนิดมอสช่องพีในช่วง $-2v < V_{thp} < 0v$ วงจรส่วนที่ทำหน้าที่รวมสัญญาณเอาต์พุตของอุปกรณ์ตรวจสอบขั้นการทำงานจะต้องให้เอาต์พุตเป็นค่าตรรกะ ½ เมื่ออุปกรณ์ตรวจสอบขั้นการทำงานในวงจรตอบรับทุกตัวให้เอาต์พุตเป็นค่าตรรกะ ½ และให้เอาต์พุตเป็นค่าตรรกะ 0 เมื่อเอาต์พุตของอุปกรณ์ตรวจสอบขั้นการทำงานทุกตัวให้เอาต์พุตเป็นค่าตรรกะ 0 อุปกรณ์สำหรับตรรกะฐานสองที่มีคุณสมบัติดังกล่าวคืออุปกรณ์ชนิดซีของมูลเลอร์ที่กล่าวไว้ในหัวข้อที่ 2.1 โดยปรับค่าแรงดันขีดเริ่มเปลี่ยนให้ทำงานที่ค่าตรรกะ ½ แทนค่าตรรกะ 1 ซึ่งทำให้วงจรตอบรับที่ได้มีลักษณะดังรูปที่ 3.9 เมื่อ (I_1-I_n) คือสายสัญญาณอินพุตที่ 1 ถึง n ของวงจรเชิงผสมและ (int_1-int_m) คือสายสัญญาณภายในที่ 1 ถึง m ของวงจรเชิงผสม โดยอุปกรณ์ชนิดซีที่มีอินพุตจำนวน n อินพุตเมื่อ n มีค่ามากกว่า 2 สามารถสร้างได้จากเกตแอนด์และเกตออร์ดังรูปที่ 3.10 เมื่อ (I_1-I_n) คือสายสัญญาณอินพุตที่ 1 ถึง n ของอุปกรณ์ชนิดซี และ n > 2



รูปที่ 3.9 วงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี



รูปที่ 3.10 อุปกรณ์ชนิดซีที่มีจำนวนอินพุตมากกว่าสองอินพุต

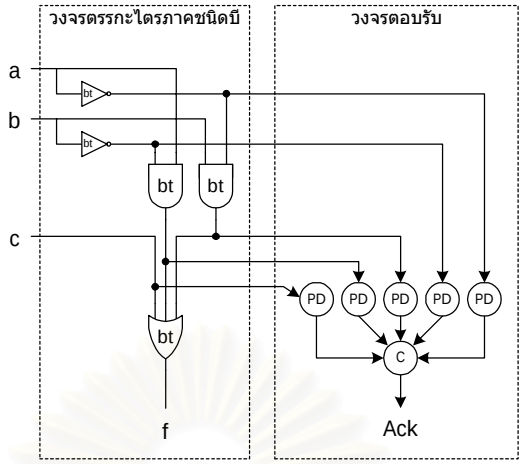
3.2.3 การเลือกสายสัญญาณจากวงจรเชิงผสมเพื่อตรวจสอบขั้นการทำงาน

เมื่อพิจารณาโครงสร้างของวงจรเชิงผสม เราจะพบว่าสำหรับเกตที่มีอินพุตหนึ่งอินพุตนั้น การเปลี่ยนแปลงระดับสัญญาณที่เอาต์พุตสามารถรับประกันการเปลี่ยนแปลงสัญญาณที่อินพุตได้ ดังนั้นการเลือกสายสำหรับเกตประเภทนี้จะทำเฉพาะสายเอาต์พุตของเกตโดยไม่เลือกสายอินพุตของเกต และเมื่อพิจารณาการเปลี่ยนแปลงสัญญาณของเกตที่มีอินพุตจำนวน n อินพุต เมื่อ $n \geq 2$ เช่นเกตแอนด์และเกตออร์ เราจะพบว่าในขั้นทำงานเอาต์พุตของวงจรนั้น ค่าตรรกะสามารถเปลี่ยนเป็น 0 หรือ 1 ได้ เมื่อมีอินพุตใดอินพุตหนึ่งเกิดการเปลี่ยนระดับสัญญาณ และในขั้นว่างเอาต์พุตของวงจรจะสามารถเปลี่ยนเป็นค่าตรรกะ ½ เมื่ออินพุตใดอินพุตหนึ่งเกิดการเปลี่ยนระดับสัญญาณเช่นกัน ดังนั้นสำหรับเกตที่มีอินพุตมากกว่าหรือเท่ากับสองอินพุต เราจะต้องเลือกสายเอาต์พุตและอินพุตทั้งหมดไปตรวจสอบขั้นการทำงาน สำหรับสายที่เป็นเอาต์พุตของวงจรนั้นเราจะไม่นำมาตรวจสอบขั้นการทำงานเนื่องจากการเปลี่ยนแปลงสัญญาณของสายที่เชื่อมต่อกับเอาต์พุตของวงจรเป็นตัวบอกสถานะของขั้นการทำงานของเอาต์พุตอยู่แล้ว

จากหลักการเลือกสายสัญญาณจากวงจรเชิงผสมเพื่อตรวจสอบขั้นการทำงาน และหลักการออกแบบวงจรตอบรับโดยใช้อุปกรณ์ตรวจสอบขั้นการทำงานที่กล่าวข้างต้น เราสามารถสรุปเป็นวิธีการสร้างวงจรตอบรับสำหรับวงจรเชิงผสมสำหรับตรรกะไตรภาคชนิดบีโดยเขียนให้อยู่ในรูปรหัสเทียม (pseudo code) ได้ดังนี้

```
for each signal in (input and output of gate)
        if (signal ≠ output of circuit) and (signal ≠ input of one input gate) then
                select signal as input of pd-element
        end if
end for
for each signal in (output signal of pd-elements in ack-path)
        select signal as input of c-element
end for
```

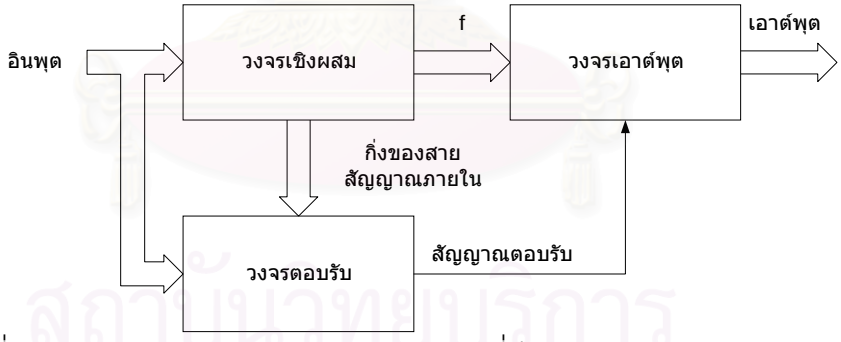
จากตัวอย่างวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี f=ab'+a'b+c เมื่อนำหลักการจากรหัสเทียมมาสร้างวงจรตอบรับเราสามารถสร้างวงจรตอบรับได้ดังรูปที่ 3.11



รูปที่ 3.11 วงจรตรรกะไตรภาคชนิดบี f=ab'+a'b+c และวงจรตอบรับ

3.2.4 การใช้งานสัญญาณตอบรับ

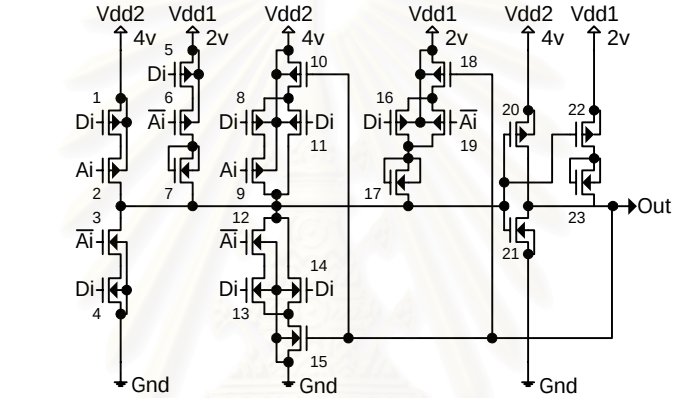
สัญญาณตอบรับที่ได้จากการตรวจสอบขั้นการทำงานของสัญญาณภายในวงจรเชิงผสมสามารถรับประกันการสิ้นสุดการทำงานของสัญญาณภายในของวงจรเชิงผสมได้ แต่วงจรยังคงสามารถเกิดการเปลี่ยนแปลงสัญญาณที่เอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในได้ เพื่อป้องกันเอาต์พุตของวงจรเกิดการเปลี่ยนแปลงสัญญาณก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจร จึงจำเป็นต้องมีวงจรเอาต์พุตดังรูปที่ 3.12



รูปที่ 3.12 โครงสร้างวงจรเชิงผสมแบบอสมวารที่มีวงจรตอบรับและวงจรเอาต์พุต

จากโครงสร้างวงจรเชิงผสมในรูปที่ 3.12 เพื่อป้องกันการเปลี่ยนแปลงสัญญาณเอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน ที่เอาต์พุตจะต้องมีลักษณะที่เกิดการเปลี่ยน-แปลงสัญญาณได้ก็ต่อเมื่อสัญญาณเอาต์พุต f ของวงจรเชิงผสมเกิดการเปลี่ยนแปลงระดับสัญญาณและเกิดการเปลี่ยนแปลงระดับสัญญาณตอบรับที่มาจากการตรวจสอบสัญญาณภายในของวงจรเชิงผสมแล้ว ซึ่งทำให้วงจรส่วนเอาต์พุตจะต้องมีคุณสมบัติดังนี้

1. ให้เอาต์พุตเป็นค่าตรรกะ 0 เมื่อเอาต์พุตของวงจรเชิงผสมเป็นค่าตรรกะ 0 และสัญญาณตอบรับเป็นค่าตรรกะ 0
2. ให้เอาต์พุตเป็นตัวแบ่งรอบการทำงาน (½) เมื่อเอาต์พุตของวงจรเชิงผสมเป็นตัวแบ่งรอบการทำงาน และสัญญาณตอบรับเป็นตัวแบ่งรอบการทำงาน
3. ให้เอาต์พุตเป็นค่าตรรกะ 1 เมื่อเอาต์พุตของวงจรเชิงผสมเป็นค่าตรรกะ 1 และสัญญาณตอบรับเป็นค่าตรรกะ 0
4. เมื่อสัญญาณเอาต์พุตของวงจรเชิงผสมและสัญญาณตอบรับเป็นกรณีอื่นๆ ให้คงค่าเอาต์พุตเดิม



Di	Ai	Out
0	0	0
0	½	Out
½	0	Out
½	½	½
1	0	1
1	½	Out

รูปที่ 3.13 อุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร

จากคุณสมบัติดังกล่าวเมื่อนำมาออกแบบวงจรในระดับทรานซิสเตอร์ เราสามารถออกแบบวงจรได้ดังรูปที่ 3.13 ซึ่งเป็นวงจรที่มีอินพุตสองอินพุตและมีเอาต์พุตหนึ่งเอาต์พุต อินพุต Di คืออินพุตที่ใช้เชื่อมต่อกับเอาต์พุตของวงจรเชิงผสม โดยที่สัญญาณที่ส่งมาที่ Di จะเป็นเซตของค่าตรรกะ {0, ½, 1} อินพุต Ai คืออินพุตที่ใช้เชื่อมต่อกับสัญญาณตอบรับของวงจร โดยที่สัญญาณที่ส่งมาที่ Ai จะเป็นเซตของค่าตรรกะ {0, ½} และเอาต์พุต Out คือเอาต์พุตของวงจรซึ่งให้สัญญาณออกมาเป็นเซตของค่าตรรกะ {0, ½, 1} โดยที่ทรานซิสเตอร์ชนิดมอสแต่ละตัวที่ใช้ในวงจรมีค่าแรงดันขีดเริ่มเปลี่ยนดังนี้

ทรานซิสเตอร์ชนิดมอสหมายเลข 1, 2, 9, 10, 11 และ 20 เป็นทรานซิสเตอร์ชนิดมอสช่องพีที่มีแรงดันขีดเริ่มเปลี่ยน V_{thp} = -2.5 โวลต์

ทรานซิสเตอร์ชนิดมอสหมายเลข 5, 6, 8, 16, 18, 19 และ 22 เป็นทรานซิสเตอร์ชนิดมอสช่องพีที่มีแรงดันขีดเริ่มเปลี่ยน V_{thp} = +0.5 โวลต์

ทรานซิสเตอร์ชนิดมอสหมายเลข 3, 4, 12, 14, 15 และ 21 เป็นทรานซิสเตอร์ชนิดมอสช่องเอ็นที่มีแรงดันขีดเริ่มเปลี่ยน V_{thn} = 2.5 โวลต์

ทรานซิสเตอร์ชนิดมอสหมายเลข 7, 17 และ 23 เป็นทรานซิสเตอร์ชนิดมอสช่องเอ็นที่มีแรงดันขีดเริ่มเปลี่ยน V_{thn} = 0.0 โวลต์

ทรานซิสเตอร์ชนิดมอสหมายเลข 13 เป็นทรานซิสเตอร์ชนิดมอสช่องเอ็นที่มีแรงดันขีดเริ่มเปลี่ยน V_{thn} = 1.0 โวลต์

จากรูปที่ 3.13 เราสามารถแบ่งกลุ่มของทรานซิสเตอร์ได้ 7 กลุ่ม กลุ่มที่ 1 ประกอบด้วยทรานซิสเตอร์หมายเลข 1 และ 2 กลุ่มที่ 2 ประกอบด้วยทรานซิสเตอร์หมายเลข 3 และ 4 กลุ่มที่ 3 ประกอบด้วยทรานซิสเตอร์หมายเลข 5-7 กลุ่มที่ 4 ประกอบด้วยทรานซิสเตอร์หมายเลข 8-11 กลุ่มที่ 5 ประกอบด้วยทรานซิสเตอร์หมายเลข 12-15 กลุ่มที่ 6 ประกอบด้วยทรานซิสเตอร์หมายเลข 16-19 และกลุ่มที่ 7 ประกอบด้วยทรานซิสเตอร์หมายเลข 20-23 ซึ่งเป็นกลุ่มที่เป็นวงจรตัวผกผัน

กลุ่มทรานซิสเตอร์กลุ่มที่ 1 เป็นกลุ่มทรานซิสเตอร์ที่ทำให้วงจรให้เอาต์พุตเป็นค่าตรรกะ 0 เมื่ออินพุต Di และ Ai เป็นค่าตรรกะ 0 ตามคุณสมบัติข้อที่ 1 โดยทรานซิสเตอร์ทั้งสองตัวมีคุณสมบัติเป็นสวิตช์ปิดเมื่อแรงดันเกตมีค่าน้อยกว่า $Vdd2+V_{thp}$ ดังนั้นทรานซิสเตอร์ทั้งสองตัวจะส่งผ่านแรงดัน 4 โวลต์ เมื่ออินพุต Di และ Ai ทั้งสองอินพุตมีค่าเป็นตรรกะ 0 และเมื่อนำแรงดันที่ส่งผ่านออกมาเป็นอินพุตของตัวผกผันซึ่งประกอบด้วยทรานซิสเตอร์หมายเลข 20-23 จะทำให้แรงดันที่เอาต์พุตของวงจรมีค่า 0 โวลต์หรือเป็นค่าตรรกะ 0

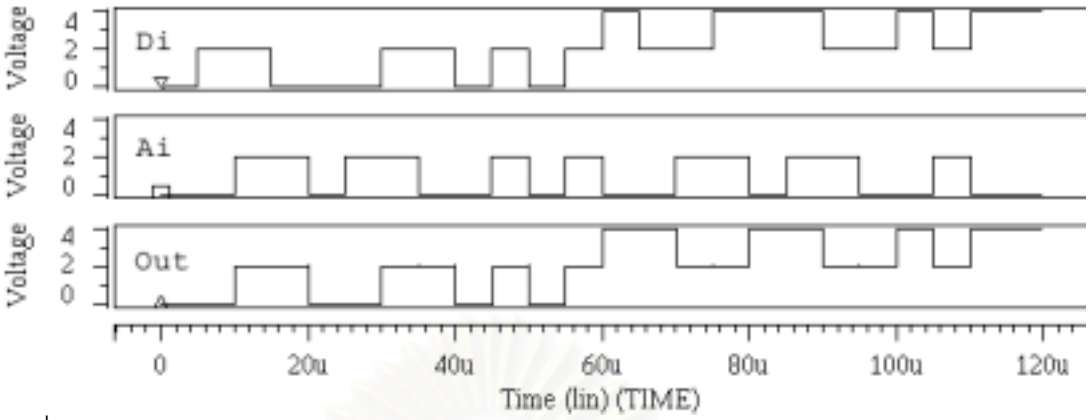
กลุ่มทรานซิสเตอร์กลุ่มที่ 2 เป็นกลุ่มทรานซิสเตอร์ที่ทำให้วงจรให้เอาต์พุตเป็นค่าตรรกะ 1 เมื่ออินพุต Di เป็นค่าตรรกะ 1 และ Ai เป็นค่าตรรกะ 0 ตามคุณสมบัติข้อที่ 3 โดยทรานซิสเตอร์หมายเลข 3 และ 4 จะมีคุณสมบัติเป็นสวิตช์ปิดเมื่อแรงดันเกตมีค่ามากกว่า V_{thn} ดังนั้นทรานซิสเตอร์กลุ่มที่ 2 จะส่งผ่านแรงดัน 0 โวลต์ เมื่ออินพุต Di เป็นค่าตรรกะ 1 และ Ai เป็นค่า

ตรรกะ 0 และเมื่อผ่านตัวผกผันในทรานซิสเตอร์กลุ่มที่ 7 จะทำให้แรงดันที่เอาต์พุตมีค่า 4 โวลต์ หรือเป็นค่าตรรกะ 1

กลุ่มทรานซิสเตอร์กลุ่มที่ 3 เป็นกลุ่มทรานซิสเตอร์ที่ทำให้วงจรให้เอาต์พุตเป็นค่าตรรกะ ½ เมื่ออินพุต Di และ Ai เป็นค่าตรรกะ ½ ตามคุณสมบัติข้อที่ 2 โดยทรานซิสเตอร์หมายเลข 5 และ 6 จะมีคุณสมบัติเป็นสวิตช์ปิดเมื่อแรงดันเกตมีค่าน้อยกว่า Vdd1 + V_{thp} ดังนั้นทรานซิสเตอร์หมายเลข 5 จะส่งผ่านแรงดัน 2 โวลต์ เมื่ออินพุต Di เป็นค่าตรรกะ 0 หรือ ½ ทรานซิสเตอร์หมายเลข 6 จะส่งผ่านแรงดันเมื่ออินพุต Ai เป็นค่าตรรกะ ½ และมีทรานซิสเตอร์หมายเลข 7 ทำหน้าที่เสมือนตัวต้านทานกันไม่ให้แรงดัน 2 โวลต์ที่ส่งผ่านออกมามีผลต่อวงจรเมื่อกลุ่มทรานซิสเตอร์อื่นส่งผ่านแรงดัน 4 โวลต์ออกมา และเมื่อแรงดัน 2 โวลต์ที่ส่งผ่านออกมาผ่านตัวผกผันในทรานซิสเตอร์กลุ่มที่ 7 จะทำให้แรงดันที่เอาต์พุตมีค่า 2 โวลต์หรือเป็นค่าตรรกะ ½

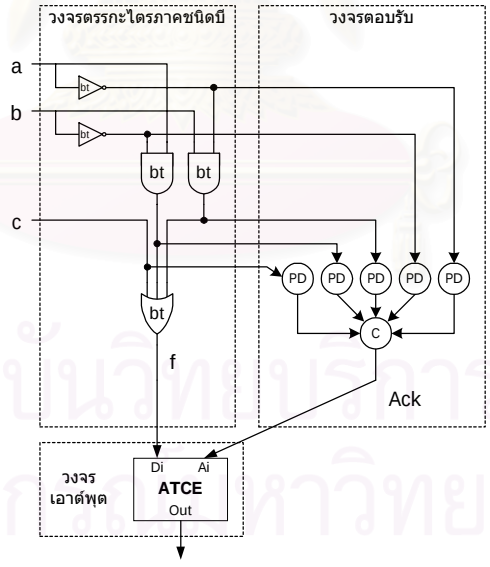
กลุ่มทรานซิสเตอร์กลุ่มที่ 4-6 เป็นกลุ่มทรานซิสเตอร์ที่ทำให้วงจรคงค่าเอาต์พุตเดิมไว้เมื่ออินพุตของวงจรเป็นกรณีอื่นๆ ตามคุณสมบัติข้อที่ 4 โดยทรานซิสเตอร์กลุ่มที่ 4 มีหน้าที่คงค่าตรรกะ 0 ที่เอาต์พุต โดยส่งผ่านแรงดัน 4 โวลต์ออกมาเป็นอินพุตของตัวผกผันเมื่อเอาต์พุตเดิมของวงจรเป็นค่าตรรกะ 0 และอินพุต Di เป็นค่าตรรกะ 0 หรือ อินพุต Di เป็นค่าตรรกะ 0 หรือ ½ และ อินพุต Ai เป็นค่าตรรกะ 0 ทรานซิสเตอร์กลุ่มที่ 5 มีหน้าที่คงค่าตรรกะ 1 ที่เอาต์พุต โดยส่งผ่านแรงดัน 0 โวลต์ออกมาเป็นอินพุตของตัวผกผันเมื่อเอาต์พุตเดิมของวงจรเป็นค่าตรรกะ 1 และอินพุต Di เป็นค่าตรรกะ 1 หรือ อินพุต Di เป็นค่าตรรกะ ½ หรือ 1 และ อินพุต Ai เป็นค่าตรรกะ 1 ทรานซิสเตอร์กลุ่มที่ 6 มีหน้าที่คงค่าตรรกะ ½ ที่เอาต์พุต โดยส่งผ่านแรงดัน 2 โวลต์ออกมาเป็นอินพุตของตัวผกผันเมื่อเอาต์พุตเดิมของวงจรเป็นค่าตรรกะ ½ และอินพุต Di หรือ Ai เป็นค่าตรรกะ 0 หรือ ½ และมีทรานซิสเตอร์หมายเลข 17 ทำหน้าที่เป็นตัวต้านทานเหมือนกับทรานซิสเตอร์ตัวที่ 7 สำหรับทรานซิสเตอร์กลุ่มที่ 3

เมื่อนำวงจรในรูปที่ 3.13 มาจำลองการทำงานด้วยโปรแกรมสไปซ์ เราได้ผลการจำลองการทำงานดังรูปที่ 3.14 เนื่องจากคุณสมบัติของวงจรในรูปที่ 3.13 มีลักษณะคล้ายอุปกรณ์ชนิดซี แต่อินพุตหนึ่งเป็นค่าตรรกะไตรภาคชนิดบี และอีกอินพุตหนึ่งเป็นค่าตรรกะฐานสองที่ใช้ค่าตรรกะ ½ เสมือนกับค่าตรรกะ 1 ดังนั้นจึงเรียกวงจรนี้ว่า *อุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร (Asymmetric Ternary C-Element : ATCE)*



รูปที่ 3.14 ผลการจำลองการทำงานอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรด้วยโปรแกรมสไปซ์

เมื่อนำอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรมาต่อกับเอาต์พุตของวงจรเชิงผสมและสัญญาณตอบรับจากวงจรตอบรับ เราจะสามารถป้องกันการเปลี่ยนแปลงเอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในได้ เนื่องจากทำให้เกิดการเปลี่ยนแปลงสัญญาณที่เอาต์พุตเมื่อเอาต์พุตของวงจรเชิงผสมเกิดการเปลี่ยนแปลงสัญญาณ และวงจรตอบรับตรวจสอบได้ว่าสัญญาณภายในวงจรสิ้นสุดการเปลี่ยนแปลงสัญญาณแล้ว วงจรเชิงผสมที่มีส่วนของวงจรตอบรับและส่วนของวงจรเอาต์พุตของวงจรตัวอย่าง f=ab'+a'b+c มีลักษณะดังรูปที่ 3.15



รูปที่ 3.15 วงจรเชิงผสม f=ab'+a'b+c ประเภทตรรกะไตรภาคชนิดบีที่มีส่วนวงจรตอบรับและส่วนวงจรเอาต์พุต

3.3 สรุป

การประมวลผลที่ถูกต้องของวงจรเชิงเลขเป็นสิ่งสำคัญที่ต้องคำนึงถึงในการออกแบบวงจร เมื่อวิเคราะห์ฮาซาร์ดที่สามารถเกิดขึ้นกับวงจรตรรกะไตรภาคชนิดบี เราสามารถสรุปได้ว่าวิธีการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์ทำให้พฤติกรรมการเปลี่ยนแปลงสัญญาณภายในวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีมีลักษณะของการเปลี่ยนแปลงสัญญาณจากค่าตรรกะ 0 หรือ 1 เป็นตัวแบ่งรอบการทำงานในขั้นว่าง และมีลักษณะการเปลี่ยนแปลงสัญญาณจากตัวแบ่งรอบการทำงานเป็นค่าตรรกะ 0 หรือ 1 ในขั้นทำงานเท่านั้น ซึ่งเป็นพฤติกรรมการเปลี่ยนแปลงสัญญาณในกรณีที่ไม่ก่อให้เกิดฮาซาร์ดของตรรกะ ดังนั้นจึงสรุปได้ว่าวงจรตรรกะไตรภาคชนิดบีไม่เกิดฮาซาร์ดของตรรกะ

เมื่อวิเคราะห์พฤติกรรมการเปลี่ยนแปลงอินพุตของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี เราสามารถสรุปได้ว่าวงจรไม่เกิดฮาซาร์ดเนื่องจากความหน่วงเมื่อใช้แบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมมูลฐานเนื่องจากความหน่วงของสิ่งแวดล้อมมากกว่าความหน่วงของวงจรทำให้สิ่งแวดล้อมส่งอินพุตใหม่มาให้วงจรเมื่อวงจรสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในแล้ว แต่วงจรที่ใช้แบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมรับเข้าส่งออกสามารถเกิดฮาซาร์ดเนื่องจากความหน่วงได้เพราะความหน่วงของสิ่งแวดล้อมมีค่าน้อยกว่าวงจรทำให้สิ่งแวดล้อมส่งอินพุตใหม่ให้วงจรในขณะที่วงจรยังไม่สิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน ดังนั้นการตรวจสอบการสิ้นสุดการทำงานที่เอาต์พุตของวงจรจึงไม่เพียงพอ เนื่องจากไม่สามารถรับประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสมได้ จึงต้องใช้วงจรตอบรับที่ใช้วิธีการตรวจสอบการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสม ซึ่งวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีนั้นแบ่งออกเป็นสองส่วนคือส่วนที่ทำการตรวจสอบขั้นการทำงานสัญญาณอินพุตและสัญญาณภายในจากวงจรเชิงผสม และส่วนที่ทำการรวมสัญญาณที่ตรวจสอบขั้นการทำงานแล้วเพื่อสร้างสัญญาณตอบรับ

ในส่วนตรวจสอบขั้นการทำงาน ผู้วิจัยได้ออกแบบวงจรที่ใช้ตรวจสอบขั้นการทำงานของสายสัญญาณแต่ละเส้น และเพื่อให้วงจรที่ใช้ตรวจสอบขั้นการทำงานของสายสัญญาณมีขนาดเล็ก จึงได้ออกแบบวงจรในระดับทรานซิสเตอร์ ทำให้วงจรที่ได้มีขนาดเล็กเมื่อเทียบกับเกตออร์ที่ใช้ตรวจสอบขั้นการทำงานของวงจรรางคู่ คือมีขนาดเพียงหนึ่งในสามของเกตออร์ เนื่องจากวงจรดังกล่าวมีหน้าที่ตรวจสอบขั้นการทำงาน จึงเรียกวงจรดังกล่าวว่าอุปกรณ์ตรวจสอบขั้นการทำงาน สำหรับส่วนที่ทำหน้าที่รวมสัญญาณที่ตรวจสอบขั้นการทำงานแล้วเพื่อสร้างสัญญาณตอบรับนั้น

ใช้อุปกรณ์ชนิดซีชนิดหลายอินพุตที่มีการปรับค่าแรงดันขีดเริ่มเปลี่ยนของทรานซิสเตอร์ชนิดมอสช่องเอ็นและทรานซิสเตอร์ชนิดมอสช่องพีภายในเพื่อให้ใช้กับอินพุตที่เป็นค่าตรรกะ 0 และค่าตรรกะ ½ ได้

เพื่อป้องกันการเปลี่ยนแปลงสัญญาณเอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในผู้วิจัยจึงได้ออกแบบอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร เพื่อใช้ในส่วนวงจรเอาต์พุต ซึ่งทำให้ระบบวงจรเชิงผสมให้เอาต์พุตเมื่อเกิดการเปลี่ยนแปลงสัญญาณเอาต์พุตของวงจรเชิงผสมและสัญญาณตอบรับแล้วทั้งคู่

บทที่ 4
การจำลองการทำงาน

การออกแบบวงจรโดยทั่วไปเป็นการออกแบบวงจรด้วยตรรกะฐานสองซึ่งเป็นตรรกะที่มีการพิสูจน์พฤติกรรมการทำงานของเกตแต่ละชนิดในระดับทรานซิสเตอร์ไว้แล้วว่าสามารถทำงานได้ถูกต้อง การออกแบบวงจรฐานสองในปัจจุบันจึงเป็นการออกแบบในระดับเกต การจำลองการทำงานของวงจรฐานสองในระดับเกตสามารถทำได้โดยใช้ภาษาชั้นสูง เช่น ภาษาวีเอชดีแอล (VHDL) และภาษาเวอริลอก (Verilog) ดังนั้นงานวิจัยนี้จึงแบ่งการจำลองการทำงานของวงจรออกเป็นสองส่วน ส่วนแรกเป็นการจำลองการทำงานของเกตและอุปกรณ์ที่ออกแบบในระดับทรานซิส-เตอร์เพื่อพิสูจน์ว่าสามารถทำงานถูกต้องเมื่อเทียบพฤติกรรมทางไฟฟ้ากับพฤติกรรมของตรรกะ ส่วนที่สองเป็นการจำลองการทำงานของวงจรขนาดใหญ่ที่ออกแบบในระดับเกตโดยใช้เกตและอุปกรณ์ที่ได้พิสูจน์ด้วยการจำลองการทำงานในส่วนแรกแล้วว่าสามารถสร้างเกตหรืออุปกรณ์ที่มีพฤติกรรมตามที่ออกแบบไว้ได้จริง

เนื่องจากวัตถุประสงค์หลักของงานวิจัยนี้คือการออกแบบวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีโดยมีวิธีการออกแบบวงจรตอบรับดังที่นำเสนอในบทที่ 3 ซึ่งได้ออกแบบอุปกรณ์เพิ่มสองตัวคือ อุปกรณ์ตรวจสอบขั้นการทำงาน และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรในระดับทรานซิสเตอร์เพื่อนำไปใช้ในการออกแบบวงจรขนาดใหญ่ ดังนั้นในการจำลองการทำงานของอุปกรณ์ทั้งสอง ผู้วิจัยจึงใช้โปรแกรมสไปซ์ในการจำลองพฤติกรรมทางไฟฟ้าเพื่อพิสูจน์ว่าสามารถทำงานได้ถูกต้องเมื่อเทียบกับพฤติกรรมของค่าตรรกะ สำหรับการจำลองการทำงานของวงจรขนาดใหญ่ที่ประกอบด้วยเกตและอุปกรณ์ที่ได้จำลองการทำงานไว้แล้วนั้นจะใช้วิธีการจำลองการทำงานด้วยภาษาชั้นสูงเช่นเดียวกับการจำลองการทำงานด้วยตรรกะฐานสอง ภาษาชั้นสูงที่เลือกใช้ก็คือภาษาวีเอชดีแอล (VHDL) เนื่องจากเป็นภาษาที่มีความยืดหยุ่นสูง ผู้เขียนโปรแกรมสามารถสร้างโปรแกรมสำเร็จ (Package) ขึ้นมาเองได้ เพื่อเพิ่มเข้าไปในคลังโปรแกรม (Library) ในขณะที่ภาษาเวอริลอกไม่สนับสนุนคุณสมบัติดังกล่าว ซึ่งเป็นคุณสมบัติที่ทำให้สามารถจำลองการทำงานของวงจรตรรกะไตรภาคชนิดบีได้

บทนี้อธิบายการจำลองการทำงานโดยแบ่งการจำลองการทำงานออกเป็นสองส่วนคือการจำลองการทำงานของอุปกรณ์ที่ออกแบบในระดับทรานซิสเตอร์ด้วยโปรแกรมสไปซ์ และการจำลองการทำงานของวงจรตรรกะไตรภาคชนิดบีในระดับเกตด้วยภาษาวีเอชดีแอล

4.1 การจำลองการทำงานของอุปกรณ์ที่ออกแบบในระดับทรานซิสเตอร์ด้วยโปรแกรมสไปซ์

การออกแบบวงจรเชิงเลขนั้นอาศัยระบบพีชคณิตเข้ามาช่วยเพื่อให้ออกแบบได้ง่าย โดยคำนึงถึงพฤติกรรมของสัญญาณในลักษณะของค่าตรรกะแทนพฤติกรรมทางไฟฟ้า เกตหรืออุปกรณ์ที่ใช้ในการออกแบบวงจรต้องเป็นเกตหรืออุปกรณ์ที่ได้มีการทดสอบการทำงานในเชิงไฟฟ้าแล้วว่าสามารถทำงานได้ถูกต้อง โดยสามารถเทียบพฤติกรรมทางไฟฟ้าเป็นค่าตรรกะของระบบพีชคณิตที่ใช้ได้ ดังนั้นการออกแบบเกตหรืออุปกรณ์ในระดับทรานซิสเตอร์จึงเป็นการออกแบบวงจรไฟฟ้าที่มีพฤติกรรมทางไฟฟ้าเทียบเป็นค่าตรรกะได้ การจำลองการทำงานของการออกแบบวงจรในระดับทรานซิสเตอร์เป็นการจำลองการทำงานเพื่อทดสอบการทำงานในเชิงไฟฟ้าว่าสามารถทำงานเทียบเป็นค่าตรรกะได้ถูกต้อง งานวิจัยนี้ได้ใช้การจำลองพฤติกรรมการทำงานทางไฟฟ้าของวงจรระดับทรานซิสเตอร์ด้วยโปรแกรมสไปซ์ โดยกำหนดให้ค่าแรงดันทางไฟฟ้าเทียบเป็นค่าตรรกะของตรรกะไตรภาคชนิดบีดังนี้

แรงดันไฟฟ้า 0 โวลต์ แทนด้วยค่าตรรกะ '0'
แรงดันไฟฟ้า 2 โวลต์ แทนด้วยค่าตรรกะ '½'
แรงดันไฟฟ้า 4 โวลต์ แทนด้วยค่าตรรกะ '1'

จากลักษณะของพฤติกรรมของอุปกรณ์ตรวจสอบขั้นการทำงานที่ระบุให้อุปกรณ์ตรวจสอบขั้นการทำงานมีอินพุตหนึ่งอินพุตและเอาต์พุตหนึ่งเอาต์พุตโดยให้เอาต์พุตเป็นค่าตรรกะ 0 เมื่ออินพุตเป็นค่าตรรกะ 0 หรือค่าตรรกะ 1 และให้เอาต์พุตเป็นค่าตรรกะ ½ เมื่ออินพุตเป็นค่าตรรกะ ½ เมื่อจำลองการทำงานด้วยโปรแกรมสไปซ์ โดยแทนค่าตรรกะด้วยแรงดันไฟฟ้า เราสามารถแปลงพฤติกรรมทางตรรกะของอุปกรณ์ตรวจสอบขั้นการทำงานเป็นพฤติกรรมทางไฟฟ้าได้ดังนี้ เมื่ออินพุตของอุปกรณ์ตรวจสอบขั้นการทำงานมีค่าแรงดันไฟฟ้า 0 โวลต์ หรือ 4 โวลต์ จะให้แรงดันไฟฟ้าที่เอาต์พุตของอุปกรณ์ตรวจสอบขั้นการทำงานเป็นค่าแรงดันไฟฟ้า 0 โวลต์ และเมื่ออินพุตของอุปกรณ์ตรวจสอบขั้นการทำงานมีค่าแรงดันไฟฟ้า 2 โวลต์ จะให้แรงดันไฟฟ้าที่เอาต์พุตของอุปกรณ์ตรวจสอบขั้นการทำงานเป็นค่าแรงดันไฟฟ้า 2 โวลต์

การจำลองการทำงานของอุปกรณ์ตรวจสอบขั้นการทำงานในรูปที่ 3.7 ด้วยโปรแกรมสไปซ์ พบว่าได้ผลการจำลองการทำงานดังรูปที่ 3.8 โดยมีพฤติกรรมทางไฟฟ้าที่ให้เอาต์พุตถูกต้องตามข้อกำหนดของตรรกะที่เทียบเป็นค่าทางไฟฟ้าแล้ว

อุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรเป็นอุปกรณ์ที่ประกอบด้วยอินพุตสองอินพุตคือ อินพุต Ai และ อินพุต Di และมีเอาต์พุตหนึ่งเอาต์พุตคือ Co โดยที่เอาต์พุต Co เป็นตรรกะ 0 เมื่ออินพุต Ai และ Di เป็นค่าตรรกะ 0 เอาต์พุต Co เป็นค่าตรรกะ ½ เมื่ออินพุต Ai และ Di เป็นค่าตรรกะ ½ เอาต์พุต Co เป็นค่าตรรกะ 1 เมื่ออินพุต Ai มีค่าตรรกะ 0 และอินพุต Di มีค่าตรรกะ 1 และเอาต์พุต Co คงค่าเดิมเมื่ออินพุต Ai และ Di เป็นกรณีอื่น เมื่อแทนค่าตรรกะด้วยแรงดันไฟฟ้า เราสามารถแปลงพฤติกรรมทางตรรกะของอุปกรณ์ตรวจสอบขั้นการทำงานเป็นพฤติกรรมทางไฟฟ้าได้ดังนี้ เมื่ออินพุต Ai และ Di มีค่าแรงดันไฟฟ้า 0 โวลต์ จะทำให้แรงดันไฟฟ้าที่เอาต์พุต Co มีค่าแรงดันไฟฟ้าเท่ากับ 0 โวลต์ เมื่ออินพุต Ai และ Di มีค่าแรงดัน 2 โวลต์ จะทำให้แรงดันไฟฟ้าที่เอาต์พุต Co มีค่าแรงดันไฟฟ้าเท่ากับ 2 โวลต์ เมื่ออินพุต Ai มีค่าแรงดัน 0 โวลต์และอินพุต Di มีค่าแรงดัน 4 โวลต์ จะทำให้แรงดันไฟฟ้าที่เอาต์พุต Co มีค่าแรงดันไฟฟ้าเท่ากับ 4 โวลต์ และแรงดันที่เอาต์พุต Co จะคงค่าเดิมเมื่ออินพุต Ai และ Di มีค่าแรงดันเป็นกรณีอื่น

จากวงจรของอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรในรูปที่ 3.13 เมื่อนำมาจำลองการทำงานด้วยโปรแกรมสไปซ์ พบว่าได้ผลของการจำลองการทำงานดังรูปที่ 3.14 โดยมีพฤติ-กรรมทางไฟฟ้าที่เทียบมาจากค่าตรรกะดังที่กล่าวข้างต้นทุกประการ

จากการจำลองการทำงานของอุปกรณ์ตรวจสอบขั้นการทำงาน และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร สามารถสรุปได้ว่าอุปกรณ์ทั้งสองชนิดที่ออกแบบสามารถทำงานได้อย่างถูกต้องตามค่าตรรกะที่ได้นิยามไว้ และสามารถนำพฤติกรรมของตรรกะไปใช้ในการออกแบบวงจรในระดับเกตได้

4.2 การจำลองการทำงานของวงจรตรรกะไตรภาคชนิดบีในระดับเกตด้วยภาษาวีเอชดี-แอล

เนื่องจากการวิเคราะห์การเกิดฮาซาร์ดบนตรรกะไตรภาคชนิดบีเป็นการวิเคราะห์ผลกระทบของความหน่วงต่อการทำงานของวงจร ดังนั้นการจำลองการทำงานของวงจรเชิงผสมขนาดใหญ่จึงเป็นการจำลองการทำงานที่เน้นการจำลองผลกระทบของความหน่วงของเกตและสายในวงจรต่อความถูกต้องของการทำงานของวงจร การจำลองผลกระทบของความหน่วงของเกตและสายในวงจรฐานสองทั่วไปสามารถใช้ภาษาวีเอชดีแอลจำลองการทำงานได้ แต่เนื่องจากตรรกะไตรภาคชนิดบีเป็นระบบพีชคณิตที่มีตัวดำเนินการและตัวแปรที่กระทำอยู่บนเซตที่ประกอบด้วยค่าสามค่าซึ่งมีลักษณะแตกต่างกับตรรกะฐานสอง ดังนั้นตัวดำเนินการหรือคำสั่งที่ใช้อยู่ในภาษาวีเอชดีแอลโดยทั่วไปจึงไม่สนับสนุนการจำลองการทำงานสำหรับวงจรตรรกะไตรภาคชนิดบี จึง

ต้องสร้างคลังโปรแกรมเพิ่มเติมเพื่อให้สนับสนุนการจำลองการทำงานของตรรกะไตรภาคชนิดบี โดยนิยามแบบชนิดข้อมูล (Data Types) ขึ้นมาใหม่ที่มีคุณลักษณะรองรับการทำงานของตรรกะไตรภาคชนิดบีและสร้างตัวดำเนินการที่สามารถดำเนินการอยู่บนแบบชนิดข้อมูลที่ได้นิยามขึ้นใหม่

งานวิจัยนี้ได้นิยามแบบชนิดข้อมูลและตัวดำเนินการใหม่เพื่อให้สนับสนุนการจำลองการทำงานของวงจรตรรกะไตรภาคชนิดบี โดยยึดหลักวิธีการออกแบบและสร้างตัวดำเนินการของโปรแกรมสำเร็จ (Package) std_logic_1164 ซึ่งเป็นมาตรฐานแบบจำลองของ IEEE 1164 ที่กำหนดมาตรฐานของแบบชนิดข้อมูลสำหรับให้ผู้ออกแบบวงจรใช้ในการอธิบายการเชื่อมโยงของข้อมูลด้วยภาษาวีเอชดีแอล แบบชนิดข้อมูลที่นิยามขึ้นในงานวิจัยนี้ได้นิยามอยู่ในรูปภาษาวีเอชดีแอลซึ่งแสดงเป็นส่วนของรหัสต้นฉบับ (Source Code) ได้ดังส่วนของรหัสต้นฉบับด้านล่าง เพื่อความสะดวกในการอธิบายรหัสต้นฉบับ ดังนั้นรหัสต้นฉบับที่อยู่ในส่วนของเนื้อหาวิทยานิพนธ์นี้จะมีหมายเลขบรรทัดด้านซ้ายมือเพิ่มขึ้นมาดังรหัสต้นฉบับด้านล่าง โดยที่หมายเลขดังกล่าว จะไม่มีอยู่ในส่วนของรหัสต้นฉบับจริง และหมายเลขบรรทัดที่ปรากฏเป็นเพียงหมายเลขที่ใช้อ้างอิงในการประกอบคำอธิบายเท่านั้นโดยไม่ใช่เลขบรรทัดจริงในรหัสต้นฉบับ เนื่องจากรหัสต้นฉบับที่นำมาประกอบเนื้อหาวิทยานิพนธ์นี้เป็นเพียงส่วนหนึ่งของรหัสต้นฉบับจริงเท่านั้น

```
1  PACKAGE BT_LOGIC_LIB IS
2      TYPE bt_ulogic IS ( 'U',  -- Uninitialized
3                          'X',  -- Forcing Unknown
4                          '0',  -- Forcing 0
5                          'S',  -- Forcing Spacer
6                          '1',  -- Forcing 1
7                          'Z',  -- High Impedance
8                          'P',  -- Forcing Unknown {0, S}
9                          'Q',  -- Forcing Unknown {S, 1}
10                         'R',  -- Forcing Unknown {0, 1}
11                         'W',  -- Weak Unknown
12                         'L',  -- Weak 0
13                         'B',  -- Weak Spacer (Below Spacer)
14                         'A',  -- Weak Spacer (Above Spacer)
15                         'H',  -- Weak 1
16                         'I',  -- Weak Unknown {0, S}
17                         'J',  -- Weak Unknown {S, 1}
18                         'K',  -- Weak Unknown {0, 1}
19                         '-'); -- Don't care
20     TYPE bt_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF bt_ulogic;
21     FUNCTION resolved ( s : bt_ulogic_vector ) RETURN bt_ulogic;
22     SUBTYPE bt_logic IS resolved bt_ulogic;
23     TYPE bt_logic_vector IS ARRAY ( NATURAL RANGE <>) OF bt_logic;
24     SUBTYPE UX0S1 IS resolved bt_ulogic RANGE 'U' TO '1';
25     FUNCTION "and" ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
26     FUNCTION "and" ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
27     FUNCTION "and" ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;
28 END BT_LOGIC_LIB;
```

บรรทัดที่ 1 เป็นการระบุว่าโปรแกรมสำเร็จที่สร้างขึ้นชื่อ BT_LOGIC_LIB บรรทัดที่ 2 ถึงบรรทัดที่ 19 เป็นการนิยามแบบชนิดข้อมูล bt_ulogic แบบชนิดข้อมูล bt_ulogic มีค่าทั้งหมด 18 ค่าคือ 'U', 'X', '0', 'S', '1', 'Z', 'P', 'Q', 'R', 'W', 'L', 'B', 'A', 'H', 'I', 'J', 'K' และ '-' โดย 'U' เป็นค่าที่ใช้ระบุสถานะที่ยังไม่มีการระบุค่าเริ่มต้น เนื่องจากค่าโดยปริยาย (Default Value) ของตัวแปรหรือสัญญาณของแบบชนิดข้อมูลใดๆ ในภาษาวีเอชดีแอลจะมีค่าเริ่มต้นเป็นค่าแรกที่ประกาศในแบบชนิดข้อมูลนั้น ดังนั้นค่าแรกของแบบชนิดข้อมูล bt_ulogic จึงใช้ค่า 'U' เพื่อใช้บอกสถานะว่าข้อมูลยังไม่ได้ถูกปรับค่า (Update) ในการจำลองการทำงาน สำหรับค่า '0', 'S', '1' ใช้แทนค่าตรรกะ 0, ตรรกะ ½ และตรรกะ 1 ตามลำดับ 'L', 'B', 'A', 'H' ใช้แทนการดึงลงตรรกะ '0' ชนิดอ่อน (weak '0' pull down), การดึงขึ้นตรรกะ '1' ชนิดอ่อน (weak '1' pull up), การดึงลงตรรกะ '½' ชนิดอ่อน (weak '½' pull down) และการดึงขึ้นตรรกะ '½' ชนิดอ่อน (weak '½' pull up) ตามลำดับ '-' ใช้แทน don't care 'Z' ใช้แทนสถานะอิมพีแดนซ์สูง (high impedance) สำหรับการเชื่อมต่อของอุปกรณ์ประเภทไทรสเตต (tristate) 'X', 'P', 'Q', 'R' ใช้แทนค่าไม่ทราบ (Unknown Value) ซึ่งเกิดจากสายสัญญาณมีการเชื่อมต่อแหล่งจ่ายสัญญาณมากกว่า 1 แหล่งที่ให้ค่าระดับสัญญาณ '0', 'S' ,'1' ที่ไม่เหมือนกัน 'W', 'I', 'J', 'K' ใช้แทนค่าไม่ทราบซึ่งเกิดจากสายสัญญาณมีการเชื่อมต่อแหล่งจ่ายสัญญาณมากกว่า 1 แหล่งที่ให้ค่าระดับสัญญาณ 'L', 'B' ,'A', 'H' ที่ไม่เหมือนกัน

บรรทัดที่ 20 เป็นการนิยามแบบชนิดข้อมูล bt_ulogic_vector ให้เป็นแถวลำดับ (Array) ของแบบชนิดข้อมูล bt_ulogic บรรทัดที่ 21 เป็นการประกาศฟังก์ชัน resolved ซึ่งเป็นฟังก์ชันที่มีหน้าที่จัดการหาผลลัพธ์ให้เมื่อสายสัญญาณมีการเชื่อมต่อแหล่งจ่ายสัญญาณมากกว่า 1 แหล่ง บรรทัดที่ 22 เป็นการนิยามแบบชนิดข้อมูลย่อย bt_logic ที่ใช้แบบชนิดข้อมูล bt_ulogic เป็นแบบชนิดข้อมูลฐานที่มีการจัดหาผลลัพธ์ให้เมื่อสายสัญญาณมีการเชื่อมต่อสัญญาณกับแหล่งจ่ายสัญญาณมากกว่าหนึ่งแหล่ง แบบชนิดข้อมูลย่อย bt_logic ประกอบด้วยค่าทั้งหมด 18 ค่าเช่นเดียวกับแบบชนิดข้อมูล bt_ulogic บรรทัดที่ 23 เป็นการนิยามแบบชนิดข้อมูล bt_logic_vector ให้เป็นแถวลำดับของแบบชนิดข้อมูลย่อย bt_logic บรรทัดที่ 24 เป็นการนิยามแบบชนิดข้อมูลย่อย UX0S1 ที่ใช้แบบชนิดข้อมูล bt_ulogic เป็นแบบชนิดข้อมูลฐานที่มีการจัดหาผลลัพธ์ให้เมื่อสายสัญญาณมีการเชื่อมต่อสัญญาณต่อกับแหล่งจ่ายสัญญาณมากกว่าหนึ่งแหล่งซึ่งประกอบด้วยค่าทั้งหมด 5 ค่าคือ 'U', 'X', '0', 'S', '1' บรรทัดที่ 25-27 เป็นการประกาศฟังก์ชัน "and" โดยมีการโอเวอร์โหลดชื่อฟังก์ชันเพื่อให้สามารถใช้ชื่อฟังก์ชันเดียวกันโดยมีพารามิเตอร์ (parameters) ต่างกัน

การจำลองการทำงานของวงจรตรรกะไตรภาคชนิดบีใช้โปรแกรมโมเดลซิม (ModelSim) จำลองการทำงานของวงจรที่เขียนเป็นภาษาวีเอชดีแอล เราสามารถเขียนรหัสต้นฉบับภาษาวีเอช-ดีแอลของวงจรตรรกะไตรภาคชนิดบี f=ab'+a'b+c ในลักษณะดังนี้

```
1  library BT;
2  use BT.BT_LOGIC_LIB.all;
3  entity example1 is
4      port ( a, b, c : in  bt_logic;
5             f       : out bt_logic);
6  end example1;
7  architecture rtl of example1 is
8      signal wo_a, wo_b, wo_c            : bt_logic;
9      signal wi_an, wi_bn, wi_abn, wi_anb : bt_logic;
10     signal wo_an, wo_bn, wo_abn, wo_anb : bt_logic;
11     signal wi_f                        : bt_logic;
12 begin
13     wo_a   <= transport a                          after 1 ns;
14     wo_b   <= transport b                          after 1 ns;
15     wo_c   <= transport c                          after 1 ns;
16     wi_an  <= transport wo_a                       after 1 ns;
17     wo_an  <= transport wi_an                      after 1 ns;
18     wi_bn  <= transport wo_b                       after 1 ns;
19     wo_bn  <= transport wi_bn                      after 1 ns;
20     wi_abn <= transport wo_a and wo_bn             after 2 ns;
21     wo_abn <= transport wo_abn                     after 1 ns;
22     wi_anb <= transport wo_an and wo_b             after 2 ns;
23     wo_anb <= transport wi_anb                     after 1 ns;
24     wi_f   <= transport wo_abn or wo_anb or wo_c after 3 ns;
25     f      <= transport wi_f                       after 1 ns;
26 end rtl;
```

บรรทัดที่ 1 เป็นการประกาศชื่อคลังโปรแกรมชื่อ BT ซึ่งเป็นคลังโปรแกรมเก็บฟังก์ชันหรือเอนทิตี (Entity) ที่ถูกเรียกใช้ในเอนทิตีที่ต้องการสร้าง บรรทัดที่ 2 เป็นการประกาศชื่อโปรแกรมสำเร็จ BT_LOGIC_LIB ในคลังโปรแกรม BT ที่เก็บฟังก์ชันที่ถูกเรียกใช้โดยเอนทิตีที่ต้องการสร้าง ซึ่งส่วนของการประกาศชื่อคลังโปรแกรมในบรรทัดที่สองนี้สามารถเปลี่ยนแปลงได้ขึ้นอยู่กับการแปลโปรแกรม (Compile) งานวิจัยนี้ได้แปลโปรแกรมสำเร็จ BT_LOGIC_LIB ให้อยู่ในคลังโปรแกรม BT ซึ่งกำหนดให้เป็นคลังโปรแกรมที่เก็บโปรแกรมสำเร็จของตรรกะไตรภาคชนิดบี บรรทัดที่ 3-6 เป็นการประกาศการเชื่อมต่อของเอนทิตีชื่อ example1 โดยมีอินพุตชื่อ a, b, c และมีเอาต์พุตชื่อ f โดยเป็นสัญญาณประเภท bt_logic บรรทัดที่ 7-26 เป็นส่วนของรายละเอียดวงจร example1 โดยบรรทัดที่ 8-10 เป็นการประกาศชื่อสัญญาณภายในวงจร และบรรทัดที่ 13-25 เป็นการประกาศการเชื่อมต่อของสายสัญญาณภายในวงจร

เนื่องจากแบบจำลองความหน่วงที่ใช้ในการวิจัยเป็นแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน ซึ่งมีลักษณะของความหน่วงของเกตและสายในวงจรสามารถเป็นค่าใดๆ โดยที่ไม่เป็นค่าอนันต์และกิ่งของสายมีค่าความหน่วงเท่ากัน เพื่อทดสอบว่าวงจรสามารถทำงาน

ได้ถูกต้องตามสมมุติฐานของแบบจำลองความหน่วง ดังนั้นการจำลองการทำงานจึงอยู่ในลักษณะที่ว่าความหน่วงของเกตและสายในวงจรมีค่าความหน่วงต่างๆ กันโดยมีกิ่งของสายสัญญาณมีค่าความหน่วงเท่ากัน

สำหรับการทำงานสิ่งแวดล้อมนั้น ผู้วิจัยได้ใช้การทำงานสิ่งแวดล้อมแบบภาวะรับเข้าส่งออก และใช้การจำลองเชิงเหตุการณ์ (Event-Driven Simulation) ในการจำลองการทำงานเพื่อให้มีพฤติกรรมตามลักษณะของวงจรอสมวาร โดยกำหนดให้สิ่งแวดล้อมส่งอินพุตใหม่ไปให้วงจรหลังจากวงจรเกิดการเปลี่ยนแปลงสัญญาณที่เอาต์พุตของวงจรเป็นรหัสตรง หรือตัวแบ่งรอบการทำงานเป็นเวลาเท่ากับความหน่วงของสิ่งแวดล้อม

สำหรับการตรวจสอบความถูกต้องของวงจร ผู้วิจัยจะทำที่เอาต์พุตของวงจรทุกครั้งที่เกิดการเปลี่ยนแปลงสัญญาณที่เอาต์พุตของวงจรโดยแบ่งความผิดพลาดออกเป็นสองกรณี คือกรณีที่วงจรให้เอาต์พุตผิด และกรณีที่เกิดการเปลี่ยนแปลงสัญญาณที่ไม่ต้องการที่เอาต์พุต เช่นเอาต์พุตควรเปลี่ยนจาก S เป็น 0 แต่เกิดกรณีที่เปลี่ยนจาก S เป็น 0 และกลายเป็น S จนกระทั่งเปลี่ยนเป็น 0 อีกครั้ง

4.3 สรุป

การจำลองการทำงานในงานวิจัยนี้แบ่งออกได้เป็นสองส่วน ส่วนแรกเป็นการจำลองการทำงานของอุปกรณ์ตรวจสอบขั้นการทำงาน และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรที่ได้ออกแบบขึ้นมาในระดับทรานซิสเตอร์เพื่อใช้ในการออกแบบวงจรตรรกะไตรภาคชนิดบีในระดับเกต โดยใช้โปรแกรมสไปซ์จำลองการทำงานของอุปกรณ์ตรวจสอบขั้นการทำงาน และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรในระดับทรานซิสเตอร์ เพื่อพิสูจน์ว่าสามารถสร้างอุปกรณ์ตรวจสอบขั้นการทำงาน และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรได้จริงในระดับทรานซิสเตอร์ซึ่งผลการจำลองการทำงานที่ได้แสดงว่าวงจรที่ออกแบบทำงานได้ถูกต้องจริงตามค่าตรรกะที่ต้องการ

การจำลองการทำงานในส่วนที่สองเกี่ยวข้องกับการทำงานของวงจรตรรกะไตรภาคชนิดบีในระดับเกตที่ใช้แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน และใช้แบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมรับเข้าส่งออก โดยจำลองการทำงานด้วยการจำลองเชิงเหตุการณ์ตามพฤติกรรมของวงจรอสมวาร และใช้วิธีการตรวจสอบการเปลี่ยนแปลงสัญญาณเอาต์พุตเพื่อตรวจสอบความถูกต้องของการทำงานของวงจร

บทที่ 5
การทดลอง

เนื่องจากวัตถุประสงค์หลักของวิทยานิพนธ์ก็คือการออกแบบวงจรตอบรับที่สามารถป้องกันการเกิดฮาซาร์ดเนื่องจากความหน่วงสำหรับวงจรตรรกะไตรภาคชนิดบีได้ โดยใช้แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน และแบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมรับเข้าส่งออก ดังนั้นในส่วนของการทดลองจึงเป็นการจำลองการทำงานเพื่อทดสอบว่าวงจรตอบรับสามารถป้องกันการเกิดฮาซาร์ดได้ภายใต้สมมุติฐานของแบบจำลองความหน่วงและแบบจำลองการทำงานดังกล่าว

5.1 การทดลองการทำงานของวงจรเปรียบเทียบสมรรถนะ LGSynth 89-93

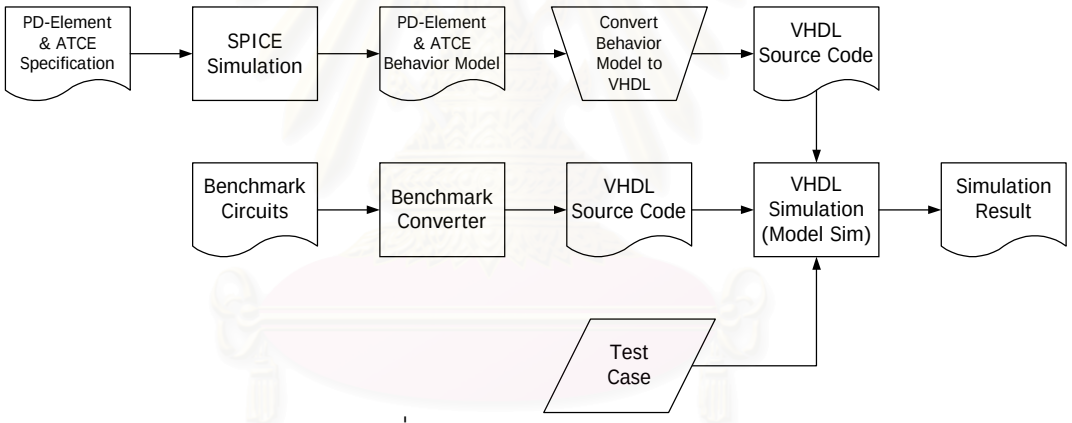
การทดลองการทำงานของวงจรเปรียบเทียบสมรรถนะ (LGSynth 89-93 [14]) เป็นการจำลองการทำงานของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีกรณีที่ไม่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน และวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีกรณีที่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน เพื่อเปรียบเทียบการเกิดฮาซาร์ดของวงจร โดยใช้แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน ด้วยการสร้างวงจรที่มีค่าความหน่วงของเกตและสายในวงจรเป็นค่าใดๆ ที่ไม่เป็นค่าอนันต์และกิ่งของสายมีค่าความหน่วงเท่ากัน โดยกำหนดให้

ความหน่วงของสายสัญญาณ	wd = 1 นาโนวินาที ถึง 20 นาโนวินาที
ความหน่วงของเกตที่มีจำนวนอินพุต 1 อินพุต	gd^1 = 1 นาโนวินาที ถึง 20 นาโนวินาที
ความหน่วงของเกตที่มีจำนวนอินพุต 2 อินพุต	gd^2 = 1 นาโนวินาที ถึง 20 นาโนวินาที
ความหน่วงของเกตที่มีจำนวนอินพุต n อินพุต	$gd^n = gd^2 * \lceil \log_2 n \rceil$

สำหรับการทำงานสิ่งแวดล้อมผู้วิจัยได้ใช้แบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมรับเข้าส่งออก และใช้การจำลองเชิงเหตุการณ์ (Event-Driven Simulation) ในการจำลองการทำงาน โดยสิ่งแวดล้อมจะส่งอินพุตใหม่ไปให้วงจรหลังจากวงจรเกิดการเปลี่ยนแปลงสัญญาณที่เอาต์พุตของวงจรเป็นรหัสตรง หรือตัวแบ่งรอบการทำงานเป็นเวลาเท่ากับความหน่วงของสิ่งแวดล้อม โดยกำหนดให้

ความหน่วงของสิ่งแวดล้อม	d_{env} = 1 นาโนวินาที

จากข้อกำหนดข้างต้นผู้วิจัยได้จำลองการทำงานของวงจรเปรียบเทียบสมรรถนะ LGSynth 89-93 ด้วยวิธีการดังรูปที่ 5.1 โดยกำหนดให้จำนวนอินพุตสูงสุดของเกตแต่ละเกตมีค่าเท่ากับ 2 อินพุต อินพุตที่ใช้ทดสอบวงจรทั้งสองประเภทใช้ชุดเดียวกันที่สร้างจากอินพุตที่ทำให้เส้นทางสั้นสุดของวงจรส่งผลต่อเอาต์พุตสลับกับค่าอินพุตอื่นที่ได้จากการสุ่ม จากการทดลองพบว่าวงจรเชิงผสมที่ไม่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในสามารถทำงานผิดพลาด โดยเกิดการเปลี่ยนแปลงสัญญาณที่ไม่ต้องการที่เอาต์พุตของวงจร ในขณะที่วงจรเชิงผสมที่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในสามารถทำงานได้ถูกต้องภายใต้เงื่อนไขที่กำหนด ผลการจำลองการทำงานของวงจรเชิงผสมมีลักษณะดังรูปที่ 5.2 และรูปที่ 5.3 โดยที่รูปที่ 5.2 เป็นตัวอย่างผลการจำลองการทำงานของวงจร con1 กรณีไม่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน และรูปที่ 5.3 เป็นตัวอย่างผลการจำลองการทำงานของวงจร con1 กรณีมีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน



รูปที่ 5.1 วิธีการจำลองการทำงาน

```
…
22971 ns: In / Out = SSSSSSS / SS
23000 ns: In / Out = 1000000 / S1
23007 ns: In / Out = 1000000 / 01
23017 ns: In / Out = SSSSSSS / S1
23035 ns: In / Out = SSSSSSS / 01
23035 ns: Incorrect output signal changed.
          Output (Expect / Actual) = S1 / 01
23037 ns: In / Out = SSSSSSS / S1
23066 ns: In / Out = SSSSSSS / SS
23096 ns: In / Out = 1111110 / 1S
23126 ns: In / Out = 1111110 / 10
TEST FAIL!
Total error is 53.
```

รูปที่ 5.2 รูปแบบผลการจำลองการทำงานด้วยการจำลองเชิงเหตุการณ์ กรณีไม่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน

```
…
54532 ns: In / Out = SSSSSSS / SS
54649 ns: In / Out = 1111101 / 10
54777 ns: In / Out = SSSSSSS / SS
54894 ns: In / Out = 1000000 / 01
55024 ns: In / Out = SSSSSSS / SS
55141 ns: In / Out = 1111110 / 10
55269 ns: In / Out = SSSSSSS / SS
55386 ns: In / Out = 1111111 / 10
55514 ns: In / Out = SSSSSSS / SS
TEST PASSED!
```

รูปที่ 5.3 รูปแบบผลการจำลองการทำงานด้วยการจำลองเชิงเหตุการณ์ กรณีมีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน

5.2 การทดลองการทำงานหน่วยคำนวณและตรรกะขนาด 32 บิต

การทดลองนี้เป็นการทดลองเพื่อทดสอบการนำวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีไปใช้แทนที่วงจรรางคู่ โดยให้สามารถทำงานได้โดยใช้วงจรควบคุมเดิม การทดลองนี้ได้ออกแบบหน่วยคำนวณและตรรกะ (ALU) ประเภทตรรกะไตรภาคชนิดบีขนาด 32 บิต ที่สามารถทำงานชุดคำสั่ง ADD, SUB, AND, OR, XOR ด้วยภาษาวีเอชดีแอล โดยใช้แบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน และแบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแล้วล้อมรับเข้าส่งออก วงจรหน่วยคำนวณและตรรกะมีลักษณะดังรูปที่ 5.4 โดยที่อินพุต Ai และ Bi เป็นตัวถูกดำเนินการ (Operand) ที่ 1 และ 2 ตามลำดับ และมีอินพุต S_2-S_0 เป็นรหัสดำเนินการ (Operation Code) ที่ไว้ใช้เลือกชุดคำสั่งของหน่วยคำนวณและตรรกะ เอาต์พุต Do เป็นผลลัพธ์ที่ได้จากการดำเนินการชุดคำสั่งของหน่วยคำนวณและตรรกะ สำหรับเอาต์พุต Co และ Vo เป็นตัวทด (Carry) และส่วนล้น (Overflow) ตามลำดับ ซึ่งรหัสดำเนินการที่ใช้กับหน่วยคำนวณและตรรกะมีความหมายดังนี้

S_2-S_0 = 000 = ADD

S_2-S_0 = 010 = SUB

S_2-S_0 = 100 = AND

S_2-S_0 = 101 = OR

S_2-S_0 = 110 = XOR

การจำลองการทำงานหน่วยคำนวณและตรรกะในรูปที่ 5.4 ประกอบด้วยอินพุตรวม 67 บิตและเอาต์พุตรวม 34 บิตโดยกำหนดให้

in_{66} - in_{64} = S_2-S_0

in_{63} - in_{32} = a_{31}-a_0

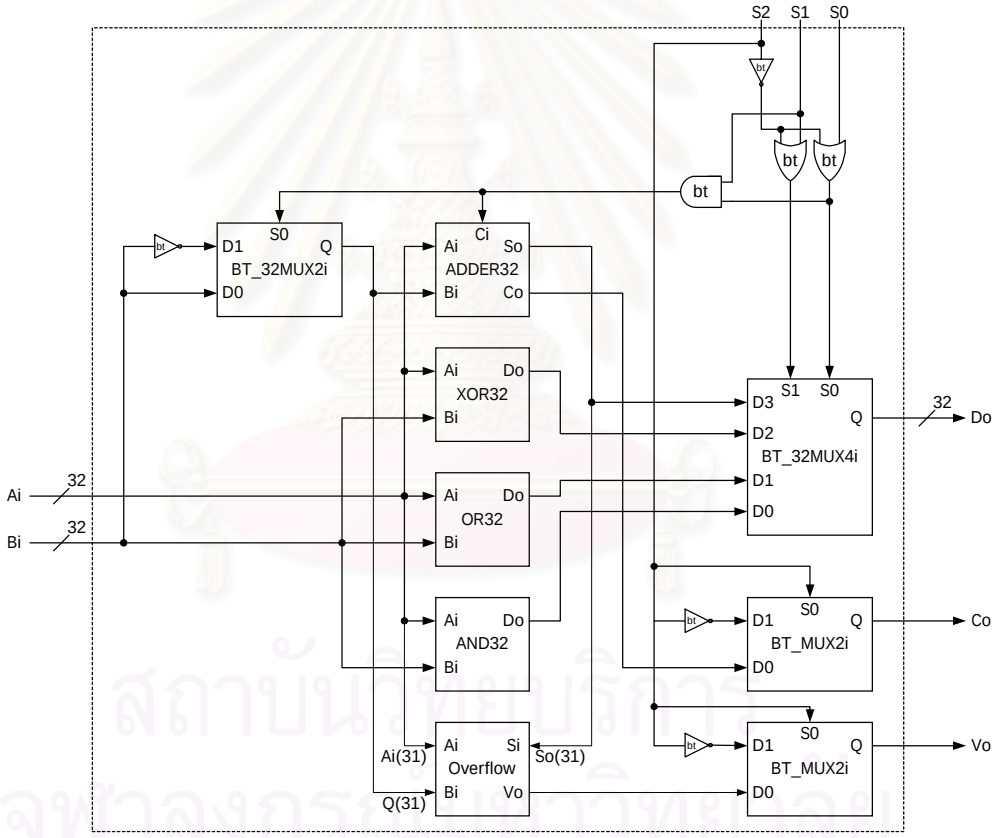
in_{31} - in_0 = a_{31}-a_0

out_{33} - out_2 = Do_{31}-Do_0

out_1 = Co

out_0 = Vo

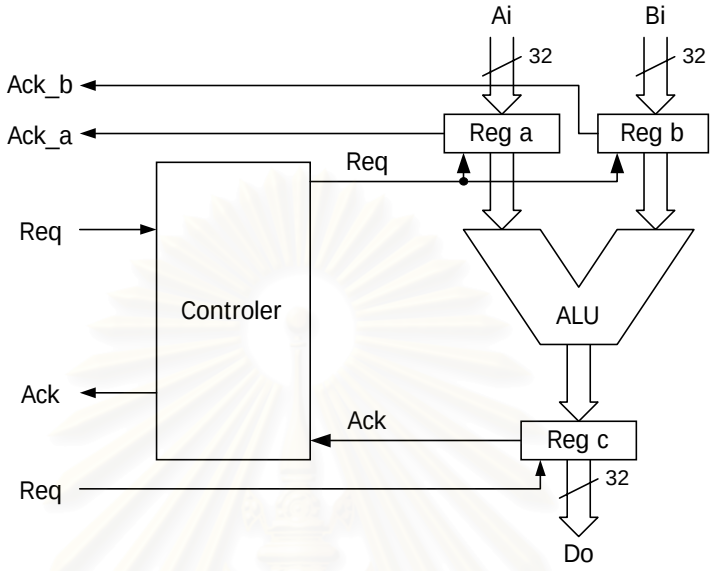
โดยที่ in_i คืออินพุตบิตที่ i และ out_i คือเอาต์พุตบิตที่ i



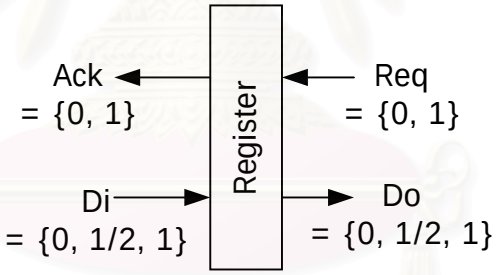
รูปที่ 5.4 หน่วยคำนวณและตรรกะขนาด 32 บิต

จากการจำลองการทำงาน พบว่าหน่วยคำนวณและตรรกะสามารถทำงานผิดพลาดได้เมื่อไม่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน และทำงานได้ถูกต้องเมื่อมีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน เมื่อพิจารณาการทำงานของหน่วยคำนวณและตรรกะในลักษณะการส่งผ่านข้อมูลจากเรจิสเตอร์ไปยังเรจิสเตอร์

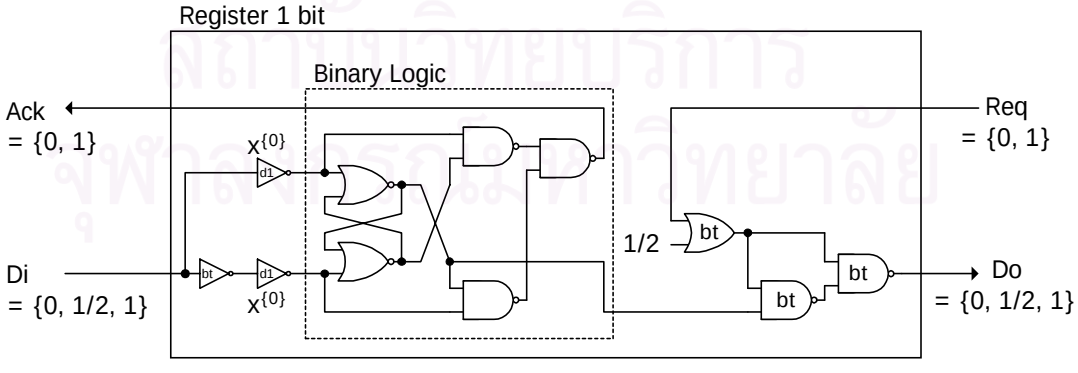
(Register to Register Transfer) ด้วยวงจรควบคุม (Controller) ที่เป็นตรรกะฐานสองดังรูปที่ 5.5 พบว่าสามารถทำได้โดยใช้เรจิสเตอร์ที่มีลักษณะดังรูปที่ 5.6



รูปที่ 5.5 การส่งผ่านข้อมูลจากเรจิสเตอร์ไปยังเรจิสเตอร์ด้วยวงจรควบคุมที่เป็นตรรกะฐานสอง



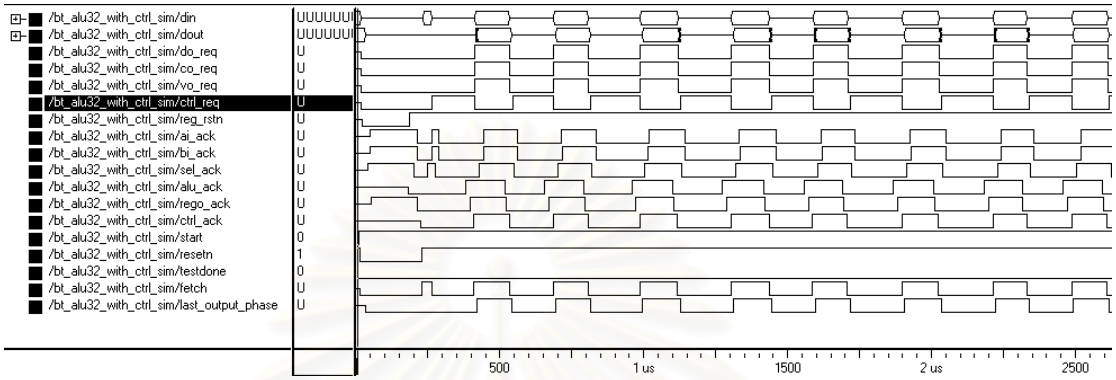
(ก)



(ข)

รูปที่ 5.6 เรจิสเตอร์ขนาด 1 บิตสำหรับตรรกะไตรภาคซนิดบีที่มีส่วนควบคุมเป็นตรรกะฐานสอง (ก) แผนภาพบล็อค, (ข) วงจร

การจำลองการทำงานวงจรดังกล่าวได้ผลการจำลองการทำงานดังรูปที่ 5.7 ซึ่งแสดงให้เห็นว่าสามารถนำวงจรตรรกะไตรภาคชนิดบีมาใช้แทนที่วงจรรางคู่ได้ โดยที่สามารถใช้วงจรควบคุมที่เป็นตรรกะฐานสองได้เหมือนเดิม



รูปที่ 5.7 ผลการจำลองการทำงานหน่วยคำนวณและตรรกะขนาด 32 บิตที่สร้างจากตรรกะไตรภาคชนิดบีและมีวงจรควบคุมเป็นตรรกะฐานสอง

5.3 สรุป

การทดลองในงานวิจัยนี้ได้จำลองการทำงานของวงจรเปรียบเทียบสมรรถนะ เพื่อเปรียบเทียบกรณีที่มีวงจรตอบรับและไม่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน โดยกรณีที่มีวงจรตอบรับวงจรสามารถทำงานให้เอาต์พุตได้ถูกต้อง แต่กรณีที่ไม่มีวงจรตอบรับวงจรสามารถเกิดการเปลี่ยนแปลงสัญญาณที่ไม่ต้องการที่เอาต์พุตของวงจร หรืออีกนัยหนึ่งคือกรณีที่มีวงจรตอบรับสามารถประกันการเกิดฮาซาร์ดเนื่องจากความหน่วงได้

สำหรับการทดลองโดยการจำลองการทำงานของหน่วยคำนวณและตรรกะในลักษณะการส่งผ่านข้อมูลจากเรจิสเตอร์ไปยังเรจิสเตอร์ด้วยวงจรควบคุมที่เป็นตรรกะฐานสองนั้น เราสามารถพิสูจน์ได้ว่าสามารถใช้วงจรตรรกะไตรภาคชนิดบีแทนที่วงจรรางคู่ได้โดยไม่ต้องเปลี่ยนวงจรควบคุม

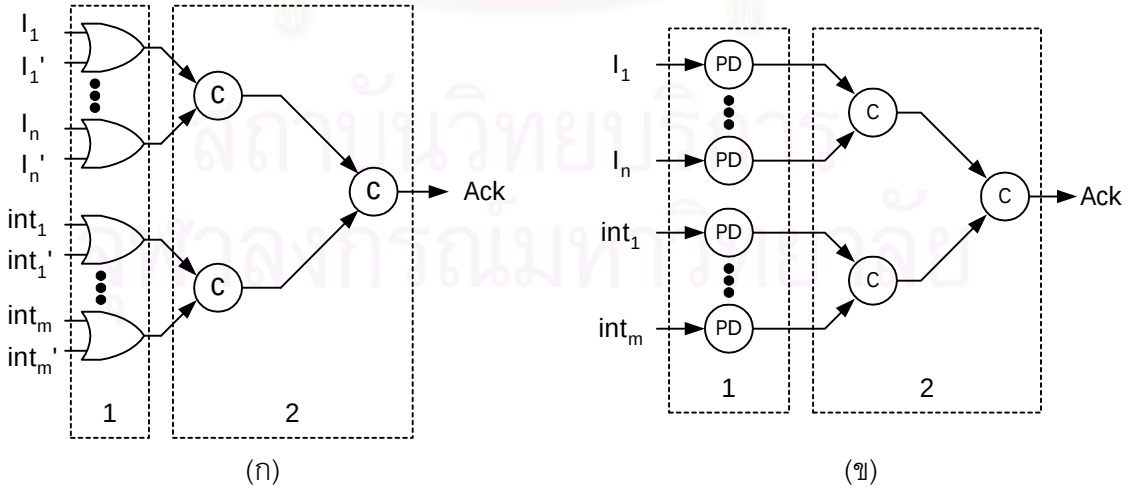
บทที่ 6
การวิเคราะห์ผลการออกแบบวงจรตอบรับ

การวิเคราะห์ผลการออกแบบวงจรในบทนี้เป็นการวิเคราะห์ผลเพื่อวัดประสิทธิภาพของวงจรตอบรับ ด้วยการวิเคราะห์ค่าใช้จ่ายวงจร (Hardware Cost) และเวลาที่ใช้ในการทำงานของวงจร โดยแบ่งการวิเคราะห์ค่าใช้จ่ายวงจรออกเป็นสองกรณี คือการวิเคราะห์ค่าใช้จ่ายวงจรของวงจรตอบรับ และการวิเคราะห์ค่าใช้จ่ายวงจรของระบบวงจรเชิงผสมที่มีวงจรตอบรับและวงจรเอาต์พุต

6.1 การวิเคราะห์ค่าใช้จ่ายวงจร

6.1.1 การวิเคราะห์ค่าใช้จ่ายวงจรของวงจรตอบรับ

จากการเปรียบเทียบลักษณะของวงจรตอบรับสำหรับวงจรรางคู่ไร้ตัวผกผันกับวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีในรูปที่ 6.1 เราพบว่าสามารถแบ่งวงจรออกเป็นสองส่วน โดยที่ส่วนที่หนึ่งกำกับด้วยหมายเลข 1 และส่วนที่สองกำกับด้วยหมายเลข 2 ส่วนที่หนึ่งของวงจรตอบรับทั้งสองประเภททำหน้าที่เหมือนกัน คือเป็นส่วนที่ใช้ตรวจสอบขั้นการทำงานของคู่สายแต่ละคู่สำหรับวงจรรางคู่ และเป็นส่วนที่ใช้ตรวจสอบขั้นการทำงานของสายสัญญาณแต่ละเส้นสำหรับวงจรตรรกะไตรภาคชนิดบี ดังนั้นจึงขอเรียกส่วนที่หนึ่งของวงจรตอบรับว่าส่วนตรวจสอบขั้นการทำงาน ส่วนที่สองของวงจรตอบรับทั้งสองประเภทเป็นส่วนที่มีลักษณะเหมือนกัน ซึ่งทำหน้าที่รวมสัญญาณที่ตรวจสอบขั้นการทำงานแล้วจากส่วนที่หนึ่งให้เป็นสัญญาณตอบรับเพียงสัญญาณเดียว จึงขอเรียกส่วนที่สองว่าส่วนสร้างสัญญาณตอบรับ



รูปที่ 6.1 วงจรตอบรับ

(ก) วงจรตอบรับสำหรับวงจรรางคู่, (ข) วงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบี

เมื่อวิเคราะห์วงจรตอบรับในส่วนตรวจสอบขั้นการทำงาน พบว่าเกตหรืออุปกรณ์ที่ใช้แตกต่างกัน วงจรตอบรับสำหรับวงจรรางคู่ใช้เกตออร์หนึ่งตัวในการตรวจสอบคู่สายสัญญาณหนึ่งคู่ ในขณะที่วงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีใช้อุปกรณ์ตรวจสอบขั้นการทำงานหนึ่งตัวต่อการตรวจสอบสายสัญญาณหนึ่งเส้น เมื่อวิเคราะห์ลักษณะการดึงสายสัญญาณของวงจรตอบรับของวงจรทั้งสองประเภทโดยกำหนดให้

x	คือจำนวนคู่สายที่ถูกเลือกเพื่อตรวจสอบขั้นการทำงานสำหรับวงจรรางคู่
y	คือจำนวนสายที่ถูกเลือกเพื่อตรวจสอบขั้นการทำงานสำหรับวงจรตรรกะไตรภาคชนิดบี

เมื่อวิเคราะห์ขนาดของเกตออร์ที่ใช้ตรวจสอบคู่สายสัญญาณและขนาดของอุปกรณ์ตรวจสอบขั้นการทำงานโดยเปรียบเทียบจำนวนทรานซิสเตอร์ที่ใช้ พบว่าจำนวนทรานซิสเตอร์ที่ใช้สำหรับเกตออร์ และอุปกรณ์ตรวจสอบขั้นการทำงานมีลักษณะดังนี้

จำนวนทรานซิสเตอร์ของเกตออร์	$size_{or}$	= 6 ตัว
จำนวนทรานซิสเตอร์ของอุปกรณ์ตรวจสอบขั้นการทำงาน	$size_{pd}$	= 2 ตัว
	$size_{or} = 3 * size_{pd}$	

จากวิธีการเลือกสายสัญญาณเพื่อตรวจสอบขั้นการทำงานในบทที่ 3 เมื่อนำมาสร้างเป็นวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบี ด้วยการทดลองออกแบบวงจรที่มีการจำกัดจำนวนอินพุตสูงสุดของเกตที่ใช้ โดยแบ่งการทดลองเป็น 7 กรณี คือกรณีที่เกตของวงจรมีจำนวนอินพุตสูงสุดเท่ากับ 2, 3, 4, 5, 6, 7 และ 8 อินพุต โดยใช้วงจรเปรียบเทียบสมรรถนะ LGSynth 89-93 ที่มีลักษณะดังตารางที่ 6.1 พบว่าได้ผลดังตารางที่ 6.2 โดยที่ dr แทนวงจรรางคู่ และ bt แทนวงจรตรรกะไตรภาคชนิดบี สดมภ์ dr คือจำนวนคู่สายที่ถูกเลือกเพื่อตรวจสอบสำหรับวงจรรางคู่ และสดมภ์ bt คือจำนวนสายที่ถูกเลือกเพื่อตรวจสอบสำหรับวงจรตรรกะไตรภาคชนิดบี จากวิธีการเลือกสายส่งผลให้จำนวนคู่สายที่ถูกเลือกสำหรับวงจรรางคู่มีจำนวนเท่ากับจำนวนสายที่ถูกเลือกสำหรับวงจรตรรกะไตรภาคชนิดบีทุกวงจรและทุกกรณีที่ตรวจสอบ ดังนั้นจึงสรุปได้ว่าอัตราส่วนจำนวนคู่สายที่ถูกเลือกเพื่อตรวจสอบขั้นการทำงานสำหรับวงจรรางคู่ (x) ต่อจำนวนสายที่ถูกเลือกเพื่อตรวจสอบขั้นการทำงานสำหรับวงจรตรรกะไตรภาคชนิดบี (y) มีค่าเท่ากับ 1 หรือ x มีค่าเท่ากับ y ซึ่งเป็นผลให้จำนวนเกตออร์ที่ใช้ในการตรวจสอบขั้นการทำงานของคู่สายของวงจรรางคู่มีค่าเท่ากับจำนวนอุปกรณ์ตรวจสอบขั้นการทำงานที่ใช้ตรวจสอบขั้นการทำงานของสายของวงจรตรรกะไตรภาคชนิดบี

ตารางที่ 6.1 วงจรเปรียบเทียบสมรรถนะ LGSynth 89-93

ชื่อวงจร	จำนวนอินพุต	จำนวนเอาต์พุต	จำนวนพจน์
5xp1	7	10	75
9sym	9	1	87
alu4	14	8	1028
apex1	45	45	206
apex2	39	3	1035
apex3	54	50	280
apex4	9	19	438
apex5	117	88	1227
b12	15	9	431
Bw	5	28	87
Clip	9	5	167
con1	7	2	9
Cordic	23	2	1206
Cps	24	109	654
duke2	22	29	87
e64	65	65	65
ex1010	10	10	1024
ex4p	128	28	620
ex5p	8	63	256
Inc	7	9	34
misex1	8	7	32
misex2	25	18	29
misex3	14	14	1848
misex3c	14	14	305
o64	130	1	65
rd53	5	3	32
rd73	7	3	141
rd84	8	4	256
sao2	10	4	58
Seq	41	35	1459
Spla	16	46	2307
sqrt8	8	4	40
squar5	5	8	32
t481	16	1	481
table3	14	14	175
table5	17	15	158
vg2	25	8	110
xor5	5	1	16
Z5xp1	7	10	128
Z9sym	9	1	420

ตารางที่ 6.2 จำนวนคู่สายที่ถูกเลือกเพื่อตรวจสอบสำหรับวงจรรางคู่ และจำนวนสายที่ถูกเลือกเพื่อตรวจสอบสำหรับวงจรตรรกะไตรภาคชนิดบีโดยใช้วงจรเปรียบเทียบสมรรถนะ LGSynth 89-93

วงจร	จำนวนอินพุตสูงสุดของเกต													
	2		3		4		5		6		7		8	
	dr	bt	dr	bt	dr	bt	Dr	bt	dr	bt	dr	bt	dr	bt
5xp1	285	285	168	168	115	115	96	96	90	90	88	88	87	87
9sym	529	529	312	312	211	211	204	204	113	113	110	110	108	108
alu4	7873	7873	4199	4199	3009	3009	2317	2317	2040	2040	1751	1751	1476	1476
Apex1	2601	2601	1367	1367	948	948	746	746	639	639	557	557	475	475
apex2	14526	14526	7505	7505	5253	5253	4078	4078	3353	3353	2863	2863	2531	2531
apex3	2979	2979	1582	1582	1139	1139	843	843	783	783	702	702	544	544
apex4	4970	4970	2553	2553	1834	1834	1299	1299	1210	1210	1114	1114	937	937
apex5	7077	7077	3950	3950	2678	2678	2325	2325	2002	2002	1906	1906	1885	1885
b12	1874	1874	1077	1077	784	784	654	654	530	530	514	514	505	505
bw	330	330	179	179	139	139	97	97	92	92	92	92	92	92
clip	887	887	485	485	365	365	285	285	224	224	201	201	197	197
con1	26	26	18	18	17	17	16	16	16	16	16	16	16	16
cordic	18388	18388	9258	9258	6876	6876	4701	4701	4573	4573	3476	3476	3448	3448
cps	7019	7019	3659	3659	2564	2564	1986	1986	1635	1635	1426	1426	1289	1289
duke2	884	884	468	468	330	330	266	266	228	228	188	188	161	161
e64	2146	2146	1122	1122	781	781	610	610	508	508	440	440	391	391
ex1010	10677	10677	5853	5853	3562	3562	3440	3440	2345	2345	2296	2296	2261	2261
ex4p	4504	4504	2556	2556	1968	1968	1376	1376	1297	1297	1282	1282	827	827
ex5p	9309	9309	4779	4779	3268	3268	2379	2379	2005	2005	1758	1758	1313	1313
inc	243	243	130	130	98	98	74	74	57	57	51	51	49	49
misex1	116	116	70	70	48	48	41	41	40	40	40	40	40	40
misex2	188	188	113	113	90	90	77	77	72	72	62	62	58	58
misex3	17957	17957	9434	9434	6571	6571	5359	5359	4091	4091	3812	3812	3650	3650
misex3c	1789	1789	983	983	708	708	563	563	482	482	425	425	390	390
o64	258	258	226	226	216	216	210	210	207	207	205	205	204	204
rd53	143	143	81	81	61	61	42	42	40	40	40	40	40	40
rd73	841	841	462	462	298	298	287	287	238	238	169	169	165	165
rd84	2204	2204	1234	1234	910	910	620	620	600	600	586	586	320	320
sao2	445	445	239	239	174	174	135	135	110	110	109	109	106	106
seq	17797	17797	9294	9294	6433	6433	5007	5007	4216	4216	3632	3632	3169	3169
spla	46849	46849	24581	24581	15661	15661	12286	12286	9470	9470	8964	8964	8574	8574
sqrt8	157	157	94	94	68	68	57	57	53	53	52	52	52	52
squar5	202	202	102	102	91	91	51	51	48	48	46	46	43	43
t481	4766	4766	2611	2611	1682	1682	1465	1465	1145	1145	993	993	965	965
table3	2457	2457	1265	1265	889	889	678	678	565	565	472	472	443	443
table5	2331	2331	1201	1201	831	831	639	639	531	531	452	452	414	414
vg2	813	813	444	444	320	320	261	261	217	217	191	191	186	186
xor5	83	83	44	44	41	41	24	24	23	23	23	23	23	23
Z5xp1	1331	1331	668	668	442	442	397	397	370	370	224	224	208	208
Z9sym	3787	3787	1898	1898	1408	1408	953	953	932	932	918	918	908	908

*หมายเหตุ dr ใช้แทนวงจรรางคู่

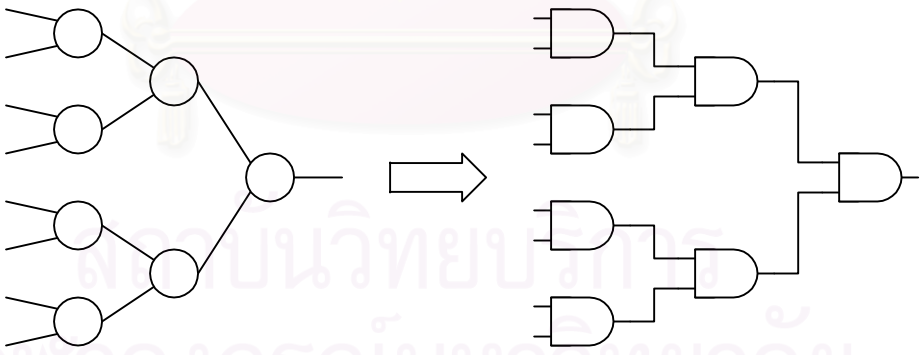
bt ใช้แทนวงจรตรรกะไตรภาคชนิดบี

เมื่อพิจารณาวงจรตอบรับส่วนสร้างสัญญาณตอบรับของวงจรทั้งสองประเภท พบว่ามีลักษณะเหมือนกันคือเป็นอุปกรณ์ชนิดซีที่มีอินพุตหลายอินพุตเหมือนกัน โดยที่ขนาดของวงจรตอบรับในส่วนนี้แปรผันตามจำนวนอินพุตของอุปกรณ์ชนิดซี จำนวนอินพุตของอุปกรณ์ชนิดซีมีค่าเท่ากับจำนวนเกตออร์ที่ใช้ในวงจรตอบรับส่วนตรวจสอบขั้นการทำงานสำหรับวงจรรางคู่ และมีค่าเท่ากับจำนวนอุปกรณ์ตรวจสอบขั้นการทำงานที่ใช้ในวงจรตอบรับส่วนตรวจสอบขั้นการทำงานสำหรับวงจรตรรกะไตรภาคชนิดบี

เนื่องจากอุปกรณ์ชนิดซีที่มีอินพุต n อินพุตสามารถสร้างได้ดังรูปที่ 3.10 ดังนั้นขนาดของอุปกรณ์ชนิดซีที่มีอินพุต n อินพุตจะมีขนาดดังนี้

ขนาดของอุปกรณ์ชนิดซีที่มีอินพุต n อินพุต =	ขนาดของเกตออร์ที่มีอินพุต n อินพุต + ขนาดของเกตแอนด์ที่มีอินพุต n อินพุต + ขนาดของอุปกรณ์ชนิดซีที่มีอินพุต 2 อินพุต

สำหรับเกตแอนด์และเกตออร์ที่มีอินพุต n อินพุตนั้นสามารถสร้างได้ในลักษณะของต้นไม้ โดยบัพแต่ละบัพ (node) ของต้นไม้แทนเกตแต่ละตัว และจำนวนลูกของบัพแต่ละบัพแทนจำนวนอินพุตของเกตแต่ละตัว เช่นตัวอย่างต้นไม้เกตแอนด์ที่มีอินพุต 8 อินพุตในรูปที่ 5.2 เมื่อจำนวนอินพุตสูงสุดของเกตแอนด์จริงเท่ากับ 2 อินพุต



รูปที่ 6.2 ตัวอย่างต้นไม้เกตแอนด์ที่มีอินพุต 8 อินพุตเมื่อจำนวนอินพุตสูงสุดของเกตแอนด์เท่ากับ 2 อินพุต

เมื่อพิจารณาต้นไม้เกตแอนด์และต้นไม้เกตออร์ที่มีอินพุต n อินพุตในกรณีที่จำนวนอินพุตสูงสุดของเกตจริงมีค่าเท่ากับ 2 อินพุต ต้นไม้เกตแอนด์และต้นไม้เกตออร์จะใช้เกตแอนด์และเกตออร์จริงเท่ากับ n-1 ตัว ดังนั้นสามารถสรุปได้ว่าอุปกรณ์ชนิดซีที่มีอินพุต n อินพุตที่สร้างจากต้นไม้เกตแอนด์และต้นไม้เกตออร์ที่เกตแอนด์และเกตออร์มีจำนวนอินพุตสูงสุดเท่ากับ 2 และ

อุปกรณ์ชนิดซีแบบสถิต (Static C-element) ที่มีอินพุตสองอินพุตซึ่งใช้จำนวนทรานซิสเตอร์เท่ากับ 6*(2(n-1))+12 ตัว เมื่ออุปกรณ์ชนิดซีแบบสถิตที่มีอินพุตสองอินพุตใช้จำนวนทรานซิสเตอร์เท่ากับ 12 ตัว

จากการวิเคราะห์ข้างต้น สามารถสรุปได้ว่าวงจรตอบรับส่วนตรวจสอบขั้นการทำงานของวงจรตรรกะไตรภาคชนิดบีมีขนาด 1/3 ของวงจรตอบรับส่วนตรวจสอบขั้นการทำงานของวงจรราง-คู่ และเมื่อวิเคราะห์ขนาดของวงจรตอบรับโดยพิจารณาจำนวนทรานซิสเตอร์ที่ใช้ทั้งหมดของวงจรตอบรับสำหรับวงจรรางคู่และวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบี สามารถสรุปได้ว่า

ขนาดของวงจรตอบรับสำหรับวงจรรางคู่	= 6n + 6*(2(n-1)) + 12
	= 18n
ขนาดของวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบี	= 2n + 6*(2(n-1)) + 12
	= 14n

ดังนั้นวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีจึงมีขนาดเล็กกว่าขนาดของวงจรตอบรับสำหรับวงจรรางคู่ 22.22% เมื่อจำนวนอินพุตสูงสุดของเกตแอนด์และเกตออร์เท่ากับ 2 อินพุต และเมื่อเปรียบเทียบอัตราการลดลงของขนาดของวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีต่อขนาดวงจรตอบรับสำหรับวงจรรางคู่ ในกรณีจำนวนอินพุตสูงสุดของเกตแอนด์มีค่าเท่ากับ 2, 3, 4, 5, 6, 7 และ 8 อินพุต ด้วยวงจรเปรียบเทียบสมรรถนะ LGSynth 89-93 พบว่าได้ผลดังตารางที่ 6.3 ซึ่งผลของอัตราส่วนของวงจรตอบรับที่ได้ของวงจรแต่ละวงจรมีค่าเหมือนกันหรือใกล้เคียงกันเมื่อจำนวนอินพุตสูงสุดของเกตมีค่าเท่ากัน โดยกรณีที่จำนวนอินพุตสูงสุดของเกตเท่ากับสองอินพุตมีอัตราการลดลงของขนาดของวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีต่อขนาดวงจรตอบรับสำหรับวงจรรางคู่น้อยที่สุด ซึ่งมีค่าเท่ากับ 22.22%

ตารางที่ 6.3 อัตราการลดลงของขนาดของวงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีต่อขนาดของวงจรตอบรับสำหรับวงจรรางคู่

วงจร	จำนวนอินพุตสูงสุดของเกต						
	2	3	4	5	6	7	8
5xp1	22.22%	28.47%	31.46%	33.10%	34.22%	34.92%	35.44%
9sym	22.22%	28.52%	31.52%	33.22%	34.19%	34.92%	35.53%
Alu4	22.22%	28.57%	31.57%	33.33%	34.47%	35.28%	35.88%
Apex1	22.22%	28.57%	31.56%	33.29%	34.44%	35.24%	35.84%
Apex2	22.22%	28.57%	31.58%	33.33%	34.47%	35.29%	35.88%
Apex3	22.22%	28.56%	31.56%	33.30%	34.44%	35.26%	35.84%
apex4	22.22%	28.57%	31.57%	33.31%	34.46%	35.26%	35.87%
apex5	22.22%	28.57%	31.57%	33.33%	34.46%	35.28%	35.87%
b12	22.22%	28.56%	31.56%	33.28%	34.44%	35.23%	35.85%
bw	22.22%	28.53%	31.48%	33.16%	34.07%	34.85%	35.66%
clip	22.22%	28.55%	31.51%	33.27%	34.36%	35.11%	35.79%
con1	22.22%	27.69%	30.09%	32.00%	33.33%	33.33%	33.33%
cordic	22.22%	28.57%	31.58%	33.33%	34.48%	35.28%	35.89%
cps	22.22%	28.57%	31.57%	33.32%	34.47%	35.27%	35.88%
duke2	22.22%	28.54%	31.52%	33.21%	34.34%	35.07%	35.74%
e64	22.22%	28.56%	31.56%	33.28%	34.42%	35.20%	35.82%
ex1010	22.22%	28.57%	31.58%	33.33%	34.47%	35.28%	35.89%
ex4p	22.22%	28.57%	31.57%	33.32%	34.45%	35.27%	35.87%
ex5p	22.22%	28.57%	31.57%	33.32%	34.47%	35.28%	35.87%
inc	22.22%	28.45%	31.31%	32.89%	33.83%	34.58%	35.38%
misex1	22.22%	28.34%	31.17%	32.93%	33.90%	34.48%	35.09%
misex2	22.22%	28.50%	31.36%	33.12%	33.96%	34.64%	35.15%
misex3	22.22%	28.57%	31.58%	33.33%	34.48%	35.28%	35.89%
misex3c	22.22%	28.56%	31.55%	33.28%	34.40%	35.23%	35.81%
o64	22.22%	28.50%	31.49%	33.18%	34.30%	35.19%	35.79%
rd53	22.22%	28.47%	31.36%	32.56%	33.90%	34.48%	35.09%
rd73	22.22%	28.54%	31.53%	33.24%	34.34%	35.17%	35.68%
rd84	22.22%	28.56%	31.56%	33.30%	34.44%	35.24%	35.79%
sao2	22.22%	28.54%	31.46%	33.13%	34.27%	35.10%	35.69%
seq	22.22%	28.57%	31.58%	33.33%	34.48%	35.28%	35.89%
spla	22.22%	28.57%	31.58%	33.33%	34.48%	35.29%	35.89%
sqrt8	22.22%	28.40%	31.19%	33.04%	33.87%	34.67%	35.14%
squar5	22.22%	28.41%	31.43%	32.80%	33.80%	34.59%	35.39%
t481	22.22%	28.57%	31.56%	33.32%	34.46%	35.26%	35.87%
table3	22.22%	28.56%	31.56%	33.28%	34.44%	35.22%	35.80%
table5	22.22%	28.56%	31.55%	33.29%	34.45%	35.20%	35.84%
vg2	22.22%	28.53%	31.50%	33.27%	34.31%	35.14%	35.70%
xor5	22.22%	28.21%	30.94%	32.43%	33.09%	34.07%	34.07%
Z5xp1	22.22%	28.55%	31.55%	33.29%	34.42%	35.11%	35.74%
Z9sym	22.22%	28.56%	31.57%	33.32%	34.44%	35.27%	35.86%
ค่าเฉลี่ย	22.22%	28.50%	31.46%	33.17%	34.26%	35.03%	35.61%

6.1.2 การวิเคราะห์ค่าใช้จ่ายวงจรของระบบวงจรเชิงผสมที่มีวงจรตอบรับและวงจรเอาต์พุต

เมื่อวิเคราะห์ระบบที่ประกอบด้วยวงจรเชิงผสม วงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน และวงจรเอาต์พุต โดยเปรียบเทียบจำนวนทรานซิสเตอร์ที่ใช้สำหรับวงจรตรรกะไตรภาคชนิดบีที่สร้างโดยใช้เกตแอนด์และเกตออร์และตัวผกผันกับวงจรรางคู่ เราจะได้ผลดังตารางที่ 6.4 ซึ่งแสดงให้เห็นว่าเมื่อนำวงจรตรรกะไตรภาคชนิดบีไปใช้แทนวงจรรางคู่จะช่วยให้ประหยัดจำนวนทรานซิสเตอร์ได้ 16.14%-20.85% และเมื่อลดขนาดของวงจรตรรกะไตรภาคชนิดบีด้วยการใช้เกตแนนด์แทนเกตแอนด์และเกตออร์จะช่วยให้ประหยัดจำนวนทรานซิสเตอร์ได้ 18.23%-27.67% ดังตารางที่ 6.5

ตารางที่ 6.4 อัตราการลดลงของขนาดของวงจรตรรกะไตรภาคชนิดบีต่อขนาดของวงจรรางคู่ เมื่อวงจรทั้งสองประเภทมีวงจรตอบรับและวงจรส่วนเอาต์พุต

วงจร	จำนวนอินพุตสูงสุดของเกต						
	2	3	4	5	6	7	8
5xp1	15.54%	17.66%	18.23%	18.43%	18.58%	18.69%	18.76%
9sym	16.45%	19.40%	20.69%	21.17%	21.87%	22.04%	22.16%
alu4	16.62%	19.83%	21.13%	21.88%	22.27%	22.59%	22.87%
apex1	17.54%	20.34%	21.06%	21.23%	21.19%	21.12%	20.98%
apex2	16.64%	19.91%	21.26%	22.00%	22.48%	22.81%	23.04%
apex3	16.09%	18.75%	19.69%	20.13%	20.36%	20.50%	20.56%
apex4	17.86%	21.29%	22.41%	23.18%	23.22%	23.27%	23.44%
apex5	16.71%	19.58%	20.68%	21.10%	21.38%	21.50%	21.59%
b12	16.47%	19.45%	20.65%	21.25%	21.67%	21.86%	22.00%
Bw	14.48%	15.58%	15.69%	15.17%	15.17%	15.26%	15.35%
Clip	16.42%	19.39%	20.51%	21.15%	21.55%	21.76%	21.90%
con1	12.37%	12.70%	13.00%	13.10%	13.35%	13.35%	13.35%
cordic	16.65%	19.96%	21.24%	22.15%	22.43%	22.89%	23.01%
cps	16.25%	19.06%	20.08%	20.58%	20.86%	21.02%	21.13%
duke2	15.61%	17.78%	18.37%	18.57%	18.64%	18.58%	18.52%
e64	15.68%	17.98%	18.66%	18.89%	18.97%	18.99%	18.98%
ex1010	16.63%	19.81%	21.32%	21.80%	22.48%	22.66%	22.79%
ex4p	16.62%	19.59%	20.67%	21.39%	21.62%	21.78%	22.15%
ex5p	16.48%	19.56%	20.79%	21.47%	21.84%	22.07%	22.34%
inc	15.43%	17.40%	17.92%	17.97%	17.85%	17.83%	17.89%
misex1	14.47%	15.77%	15.68%	15.63%	15.72%	15.81%	15.90%
misex2	13.25%	13.77%	13.54%	13.21%	13.04%	12.48%	12.24%
misex3	16.64%	19.88%	21.23%	21.91%	22.46%	22.70%	22.85%
misex3c	16.38%	19.31%	20.44%	21.02%	21.36%	21.58%	21.73%
o64	18.63%	22.95%	24.87%	25.97%	26.66%	27.20%	27.54%
rd53	15.77%	18.12%	18.88%	19.02%	19.26%	19.37%	19.48%
rd73	16.49%	19.53%	20.85%	21.32%	21.73%	22.16%	22.26%
rd84	16.58%	19.69%	20.95%	21.80%	22.09%	22.28%	22.92%
sao2	16.20%	19.01%	20.03%	20.54%	20.84%	21.01%	21.12%
seq	16.59%	19.79%	21.11%	21.83%	22.25%	22.55%	22.78%
spla	16.64%	19.87%	21.33%	22.05%	22.58%	22.79%	22.94%
sqrt8	15.69%	18.00%	18.71%	19.08%	19.21%	19.36%	19.46%
squar5	15.41%	17.31%	17.97%	17.67%	17.76%	17.82%	17.85%
t481	16.63%	19.81%	21.26%	21.87%	22.40%	22.71%	22.85%
table3	16.46%	19.52%	20.72%	21.36%	21.74%	22.01%	22.14%
table5	16.42%	19.45%	20.64%	21.26%	21.62%	21.86%	22.01%
vg2	16.19%	18.99%	20.03%	20.55%	20.84%	21.03%	21.14%
xor5	15.73%	18.07%	18.87%	19.10%	19.22%	19.42%	19.42%
Z5xp1	16.42%	19.44%	20.65%	21.14%	21.43%	21.84%	21.98%
Z9sym	16.64%	19.94%	21.21%	22.12%	22.39%	22.58%	22.71%
ค่าเฉลี่ย	16.14%	18.83%	19.83%	20.28%	20.56%	20.73%	20.85%

ตารางที่ 6.5 อัตราการลดลงของขนาดของวงจรตรรกะไตรภาคชนิดบีที่มีการลดขนาดวงจรโดยใช้เกตแนนด์แทนเกตแอนด์ออร์ต่อขนาดวงจรรางคู่ เมื่อวงจรทั้งสองประเภทมีวงจรตอบรับและวงจรส่วนเอาต์พุต

วงจร	จำนวนอินพุตสูงสุดของเกต						
	2	3	4	5	6	7	8
5xp1	19.31%	24.18%	27.03%	28.43%	29.08%	29.38%	29.58%
9sym	18.67%	23.35%	26.18%	26.91%	30.12%	30.45%	30.70%
Alu4	18.38%	23.17%	25.55%	27.19%	28.06%	28.92%	29.79%
Apex1	18.79%	22.87%	24.54%	25.41%	25.84%	26.20%	26.55%
Apex2	17.59%	21.77%	23.75%	24.98%	25.84%	26.48%	26.94%
Apex3	17.48%	21.40%	23.15%	24.36%	24.82%	25.26%	25.97%
Apex4	19.21%	24.07%	26.15%	28.03%	28.29%	28.61%	29.37%
Apex5	19.14%	24.08%	26.92%	28.06%	29.10%	29.49%	29.67%
B12	19.56%	25.01%	27.96%	29.66%	31.35%	31.79%	32.09%
Bw	18.63%	22.84%	24.46%	25.90%	26.24%	26.39%	26.55%
Clip	18.99%	24.21%	26.67%	28.52%	30.12%	30.91%	31.20%
con1	18.10%	21.57%	22.69%	23.57%	24.03%	24.03%	24.03%
cordic	17.53%	21.71%	23.49%	25.04%	25.40%	26.33%	26.47%
cps	17.59%	21.61%	23.47%	24.60%	25.37%	25.88%	26.24%
duke2	17.22%	20.80%	22.34%	23.16%	23.68%	24.15%	24.52%
e64	16.08%	18.73%	19.66%	20.08%	20.30%	20.42%	20.50%
ex1010	17.92%	22.22%	24.87%	25.50%	27.13%	27.39%	27.58%
ex4p	18.55%	23.14%	25.17%	27.27%	27.77%	28.02%	30.15%
ex5p	16.92%	20.41%	21.93%	22.88%	23.38%	23.73%	24.23%
inc	17.72%	21.62%	23.21%	24.33%	25.19%	25.61%	25.86%
misex1	18.76%	22.97%	25.31%	26.42%	26.77%	26.92%	27.08%
misex2	15.71%	17.93%	18.62%	18.93%	19.06%	19.08%	19.12%
misex3	18.02%	22.54%	24.81%	26.06%	27.35%	27.80%	28.09%
misex3c	18.74%	23.68%	26.19%	27.81%	28.87%	29.68%	30.24%
o64	22.88%	28.83%	31.51%	33.06%	34.04%	34.79%	35.26%
rd53	18.99%	23.94%	26.30%	28.53%	29.18%	29.34%	29.51%
rd73	18.77%	23.78%	26.88%	27.62%	28.87%	30.80%	31.05%
rd84	18.15%	22.57%	24.69%	26.67%	27.11%	27.41%	29.83%
sao2	18.06%	22.52%	24.61%	26.03%	27.09%	27.34%	27.58%
seq	17.71%	21.95%	24.03%	25.29%	26.11%	26.75%	27.28%
spla	17.31%	21.16%	23.17%	24.21%	25.09%	25.39%	25.60%
sqrt8	19.24%	24.16%	26.84%	28.42%	29.08%	29.44%	29.58%
squar5	17.95%	22.17%	23.44%	25.41%	25.80%	26.08%	26.42%
t481	17.98%	22.34%	24.85%	25.87%	27.06%	27.77%	28.01%
table3	17.48%	21.51%	23.37%	24.55%	25.30%	25.92%	26.19%
table5	17.41%	21.37%	23.22%	24.34%	25.07%	25.61%	25.94%
vg2	18.13%	22.62%	24.82%	26.13%	27.12%	27.83%	28.07%
xor5	18.49%	23.36%	24.70%	27.64%	28.05%	28.35%	28.35%
Z5xp1	17.79%	22.16%	24.40%	25.22%	25.72%	27.39%	27.73%
Z9sym	18.12%	22.90%	25.02%	27.05%	27.44%	27.70%	27.89%
ค่าเฉลี่ย	18.23%	22.58%	24.65%	25.98%	26.78%	27.27%	27.67%

6.2 การวิเคราะห์เวลาที่ใช้ในการทำงานของวงจร

เนื่องจากเกตที่ใช้สำหรับวงจรรางคู่และวงจรตรรกะไตรภาคชนิดบีมีลักษณะที่แตกต่างกัน ดังนั้นการวิเคราะห์เวลาที่ใช้ในการทำงานจึงใช้วิธีการวัดค่าความหน่วงของทรานซิสเตอร์แทน โดยกำหนดให้ค่าความหน่วงของทรานซิสเตอร์ชนิดมอสช่องเอ็นและทรานซิสเตอร์ชนิดมอสช่องพีมีค่าความหน่วงเท่ากัน และกำหนดให้

ความหน่วงของทรานซิสเตอร์ชนิดมอส	=	1	หน่วยเวลา

จากข้อกำหนดดังกล่าวเมื่อนำมาวิเคราะห์หาค่าความหน่วงหรือเวลาที่ใช้สำหรับเกตแต่ละประเภท โดยกำหนดให้จำนวนอินพุตสูงสุดของเกตมีค่าเท่ากับ 2 อินพุต สามารถหาค่าความหน่วงของเกตแต่ละประเภทได้ดังนี้

ค่าความหน่วงของเกตสำหรับตรรกะฐานสอง			
เกตแอนด์	=	3	หน่วยเวลา
เกตออร์	=	3	หน่วยเวลา
อุปกรณ์ชนิดซี	=	3	หน่วยเวลา

ค่าความหน่วงของเกตสำหรับตรรกะไตรภาคชนิดบี			
ตัวผกผัน	=	2	หน่วยเวลา
เกตแอนด์	=	4	หน่วยเวลา
เกตออร์	=	5	หน่วยเวลา
อุปกรณ์ตรวจสอบขั้นการทำงาน	=	1	หน่วยเวลา
อุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร	=	7	หน่วยเวลา

เมื่อพิจารณาเฉพาะวงจรตอบรับดังรูปที่ 6.1 สามารถคิดเวลาที่ใช้ในการทำงานของวงจรตอบรับสำหรับวงจรรางคู่และเวลาที่ใช้ในการทำงานของวงจรตอบรับสำหรับตรรกะไตรภาคชนิดบีได้ โดยกำหนดให้ n คือจำนวนคู่สายที่เลือกสำหรับวงจรรางคู่และจำนวนสายที่เลือกสำหรับวงจรตรรกะไตรภาคชนิดบีได้ดังนี้

ความหน่วงของอุปกรณ์ชนิดซีที่มีอินพุต n อินพุต

= max(ความหน่วงของเกตออร์ที่มีอินพุต n อินพุต, ความหน่วงของเกตแอนด์ที่มีอินพุต n อินพุต)
+ ขนาดของอุปกรณ์ชนิดซีที่มีอินพุต 2 อินพุต

<u>วงจรตอบรับสำหรับวงจรรางคู่</u>

ความหน่วงของวงจรตอบรับ = ความหน่วงของเกตออร์ที่มีอินพุต 2 อินพุต
+ ความหน่วงของอุปกรณ์ชนิดซีที่มีอินพุต n อินพุต

$= 3+\max((3*\lceil \log_2 n \rceil), (3*\lceil \log_2 n \rceil))+3$

$= (3*\lceil \log_2 n \rceil)+6$

<u>วงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบี</u>

ความหน่วงของวงจรตอบรับ = ความหน่วงของอุปกรณ์ตรวจสอบขั้นการทำงาน
+ ความหน่วงของอุปกรณ์ชนิดซีที่มีอินพุต n อินพุต

$= 1+\max((3*\lceil \log_2 n \rceil), (3*\lceil \log_2 n \rceil))+3$

$= (3*\lceil \log_2 n \rceil)+4$

ผลที่ได้จากการวิเคราะห์แสดงให้เห็นว่าเวลาที่ใช้ในการทำงานของวงจรตอบรับสำหรับวงจรรางคู่และวงจรตอบรับสำหรับตรรกะไตรภาคชนิดบีมีค่าใกล้เคียงกันโดยต่างกันเพียง 2 หน่วยเวลาโดยวงจรทั้งสองแปรผันตามค่า $\lceil \log_2 n \rceil$ เหมือนกัน

อย่างไรก็ตาม เมื่อวิเคราะห์เวลาที่ใช้ในการทำงานของระบบวงจรเชิงผสมที่มีวงจรตอบรับและวงจรเอาต์พุตของวงจรทั้งสองประเภทโดยหาค่าความหน่วงเฉลี่ยของวงจรเชิงผสมดังนี้

ค่าความหน่วงเฉลี่ยของวงจรเชิงผสม

$= (\max(\min\_delay(f_1), \ldots \min\_delay(f_m)) + \max(max\_delay(f_1), \ldots max\_delay(f_m))) / 2$

โดยกำหนดให้

m = จำนวนเอาต์พุตของวงจรเชิงผสม

$min\_delay(f_i)$ = ค่าความหน่วงต่ำสุดของเอาต์พุต i

$max\_delay(f_i)$ = ค่าความหน่วงสูงสุดของเอาต์พุต i

ผลการวิเคราะห์เวลาที่ใช้ในการทำงานของระบบวงจรเชิงผสมที่มีวงจรตอบรับและวงจรเอาต์พุตในกรณีที่จำนวนอินพุตสูงสุดของวงจรมีค่าเท่ากับ 2 อินพุตมีค่าดังตารางที่ 6.6 ผลการวิเคราะห์แสดงให้เห็นว่าเวลาที่ใช้ในการทำงานของระบบวงจรเชิงผสมที่มีวงจรตอบรับและวงจรเอาต์พุตของวงจรตรรกะไตรภาคชนิดบีมีค่ามากกว่าวงจรรางคู่ สาเหตุดังกล่าวเป็นเพราะเกตและอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรสำหรับวงจรเชิงผสมสำหรับตรรกะไตรภาคชนิดบีมีค่าความหน่วงมากกว่าเกตและอุปกรณ์ชนิดซีที่ใช้กับวงจรรางคู่

ตารางที่ 6.6 จำนวนหน่วยเวลาที่ใช้ในการทำงานของวงจรรางคู่และวงจรตรรกะไตรภาคชนิดบี ในกรณีที่จำนวนอินพุตสูงสุดของวงจรมีค่าเท่ากับ 2 อินพุต

วงจร	วงจรรางคู่		วงจรตรรกะไตรภาคชนิดบี	
	วงจรเชิงผสม	ระบบวงจรเชิงผสมที่มีวงจรตอบรับและวงจรเอาต์พุต	วงจรเชิงผสม	ระบบวงจรเชิงผสมที่มีวงจรตอบรับและวงจรเอาต์พุต
5xp1	21	54	33.5	66.5
9sym	27	63	43.5	79.5
alu4	34.5	79.5	55.5	100.5
apex1	31.5	73.5	50.5	92.5
apex2	39	87	62.5	110.5
apex3	31.5	73.5	50.5	92.5
apex4	31.5	76.5	51.5	96.5
apex5	27	72	42.5	87.5
b12	28.5	67.5	46.5	85.5
bw	15	48	23.5	56.5
clip	24	60	38.5	74.5
con1	12	33	19.5	40.5
cordic	42	93	67.5	118.5
cps	27	72	42.5	87.5
duke2	22.5	58.5	35.5	71.5
e64	19.5	61.5	27	69
ex1010	33	81	52.5	100.5
ex4p	27	72	43.5	88.5
ex5p	31.5	79.5	50.5	98.5
inc	21	51	33.5	63.5
misex1	15	42	23.5	50.5
misex2	15	45	23.5	53.5
misex3	34.5	85.5	55.5	106.5
misex3c	28.5	67.5	45.5	84.5
o64	22.5	55.5	36.5	69.5
rd53	19.5	49.5	31	61
rd73	25.5	61.5	41	77
rd84	31.5	73.5	50.5	92.5
sao2	24	57	37.5	70.5
seq	31.5	82.5	49.5	100.5
spla	42	96	67.5	121.5
sqrt8	21	51	33.5	63.5
squar5	18	48	28.5	58.5
t481	34.5	79.5	55.5	100.5
table3	28.5	70.5	46.5	88.5
table5	31.5	73.5	49.5	91.5
vg2	24	60	38.5	74.5
xor5	19.5	46.5	31	58
Z5xp1	25.5	64.5	41	80
Z9sym	36	78	57.5	99.5

6.3 สรุป

วิธีการเลือกสายสัญญาณเพื่อมาตรวจสอบการสิ้นสุดการทำงานของวงจรตรรกะไตรภาคชนิดบีทำให้ได้จำนวนสายที่เลือกเท่ากับจำนวนคู่สายที่เลือกสำหรับวงจรรางคู่ เมื่อวิเคราะห์วงจรตอบรับโดยแบ่งเป็นสองส่วน คือส่วนตรวจสอบขั้นการทำงาน และส่วนสร้างสัญญาณตอบรับ จะพบว่าส่วนตรวจสอบขั้นการทำงานของวงจรตอบรับสำหรับตรรกะไตรภาคชนิดบีมีขนาดเป็นหนึ่งในสามของขนาดวงจรส่วนตรวจสอบขั้นการทำงานของวงจรตอบรับสำหรับวงจรรางคู่ ในขณะที่ส่วนสร้างสัญญาณตอบรับของวงจรทั้งสองประเภทมีขนาดเท่ากัน ดังนั้นการลดลงของค่าใช้จ่ายวงจรของวงจรตอบรับจึงขึ้นอยู่กับส่วนตรวจสอบขั้นการทำงาน หรืออีกนัยหนึ่งคือค่าใช้จ่ายของวงจรตอบรับที่ออกแบบขึ้นอยู่กับขนาดของอุปกรณ์ตรวจสอบขั้นการทำงาน การวิเคราะห์เวลาที่ใช้ในการทำงานของระบบวงจรรางคู่และระบบวงจรตรรกะไตรภาคชนิดบีแสดงให้เห็นว่าวงจรตรรกะไตรภาคชนิดบีทำงานช้ากว่า เนื่องจากเกตที่ใช้กับวงจรตรรกะไตรภาคชนิดบีมีความหน่วงมากกว่าเกตที่ใช้สำหรับวงจรรางคู่

บทที่ 7
สรุปผลการวิจัยและข้อเสนอแนะ

7.1 สรุปผลการวิจัย

การวิจัยนี้มีจุดประสงค์เพื่อเสนอวิธีการออกแบบวงจรตอบรับสำหรับวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่สามารถประกันการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในวงจรเพื่อป้องกันฮาซาร์ดที่เกิดจากการเปลี่ยนแปลงสัญญาณอินพุตของวงจรก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี

จากการศึกษาพฤติกรรมการทำงานของเกตและวงจรตรรกะไตรภาคชนิดบีโดยวิเคราะห์ลักษณะของฮาซาร์ดที่สามารถเกิดขึ้นบนตรรกะชนิดบี พบว่าการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์ทำให้พฤติกรรมการเปลี่ยนแปลงอินพุตของเกตต่างๆ มีลักษณะการเปลี่ยนแปลงจากสัญญาณข้อมูล 0 หรือ 1 เป็นตัวแบ่งรอบการทำงาน ½ หรือจากตัวแบ่งรอบการทำงาน ½ เป็นสัญญาณข้อมูล 0 หรือ 1 การเปลี่ยนแปลงสัญญาณลักษณะดังกล่าวเป็นการเปลี่ยนแปลงในกรณีที่ไม่ทำให้เกิดฮาซาร์ดของตรรกะ ดังนั้นการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์จึงสามารถรับประกันได้ว่าตรรกะไตรภาคชนิดบีไม่เกิดฮาซาร์ดของตรรกะ อย่างไรก็ตาม เมื่อพิจารณาการทำงานของวงจรตรรกะไตรภาคชนิดบีโดยใช้แบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะรับเข้าส่งออกแล้ว จะพบว่าการตรวจสอบการสิ้นสุดการทำงานของวงจรที่เอาต์พุตไม่เพียงพอ เนื่องจากวงจรสามารถให้เอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน เมื่อสิ่งแวดล้อมตรวจสอบการสิ้นสุดการทำงานของวงจรที่สัญญาณที่เอาต์พุต สิ่งแวดล้อมจะส่งอินพุตใหม่ให้วงจรในขณะที่วงจรยังไม่สิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน ทำให้อินพุตใหม่ที่สิ่งแวดล้อมส่งให้วงจรสามารถก่อให้เกิดการแทรกสอดของการประมวลผลข้อมูลเก่าและข้อมูลใหม่ได้ซึ่งเป็นสาเหตุของฮาซาร์ดเนื่องจากความหน่วง ดังนั้นในการออกแบบวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีเมื่อใช้แบบจำลองการทำงานสิ่งแวดล้อมแบบภาวะแวดล้อมรับเข้าส่งออกจึงจำเป็นจะต้องมีวงจรตอบรับที่ทำหน้าที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน แทนการการตรวจสอบการสิ้นสุดการทำงานที่เอาต์พุตที่ของวงจร

การออกแบบวงจรตรรกะไตรภาคชนิดบีนั้นยึดหลักวิธีการส่งข้อมูลแบบสองขั้นชนิดกลับสู่ศูนย์โดยแบ่งการส่งข้อมูลแต่ละรอบออกเป็นสองขั้นคือ ขั้นทำงาน และขั้นว่าง เช่นเดียวกับวงจรรางคู่ ดังนั้นการตรวจสอบการสิ้นสุดการเปลี่ยนแปลงภายในจึงอยู่ในลักษณะการตรวจสอบสายสัญญาณภายในวงจรเชิงผสมว่าสัญญาณทุกสัญญาณอยู่ในขั้นการทำงานเดียวกันหรือไม่ โดยมีขั้นการทำงานเป็นขั้นทำงานหรือขั้นว่าง วงจรรางคู่นั้นใช้เกตออร์เป็นตัวตรวจสอบขั้นการทำงาน

ของคู่สายของวงจรเชิงผสมแต่สำหรับวงจรตรรกะไตรภาคชนิดบีนั้นไม่สามารถใช้เกตออร์เช่นเดียวกับวงจรรางคู่ การตรวจสอบขั้นการทำงานจำเป็นต้องอาศัยวงจรอื่นมาตรวจสอบขั้นการทำงานของสายแต่ละเส้น

การวิจัยนี้จึงนำเสนออุปกรณ์ที่จะใช้ตรวจสอบขั้นการทำงานของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบี และอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรที่ใช้สร้างส่วนวงจรเอาต์พุตใหม่ของวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีเพื่อป้องกันการเกิดการเปลี่ยนแปลงสัญญาณเอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน โดยจำลองการทำงานด้วยโปรแกรมสไปซ์ เพื่อพิสูจน์การทำงานของอุปกรณ์ตรวจสอบขั้นการทำงานและอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรว่าสามารถทำงานได้อย่างถูกต้อง จากผลการจำลองการทำงานของอุปกรณ์ตรวจสอบขั้นการทำงานและอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร พบว่าวงจรที่ออกแบบในระดับทรานซิสเตอร์ของอุปกรณ์ทั้งสองประเภทสามารถทำงานได้ถูกต้องโดยมีพฤติกรรมทางไฟฟ้าตามที่ต้องการ

เมื่อออกแบบวงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีโดยนำอุปกรณ์ตรวจสอบขั้นการทำงานมาออกแบบวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน และนำอุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตรมาออกแบบวงจรเอาต์พุตด้วยภาษาวีเอชดีแอล และจำลองการทำงานของวงจรดังกล่าวด้วยโปรแกรมโมเดลซิม เราพบว่าผลการจำลองการทำงานของวงจรเชิงผสมที่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในให้ผลลัพธ์ที่ถูกต้อง ไม่เกิดฮาซาร์ดที่เกิดขึ้นแก่วงจรเชิงผสมประเภทตรรกะไตรภาคชนิดบีที่ใช้วงจรตอบรับที่ตรวจสอบการสิ้นสุดการทำงานของวงจรที่เอาต์พุตของวงจรเชิงผสมเมื่อใช้อินพุตทดสอบเดียวกัน ซึ่งแสดงให้เห็นว่าวงจรตอบรับที่ออกแบบสามารถป้องกันฮาซาร์ดได้ตามทฤษฎี

จากการทดลองพบว่าวิธีการเลือกสายเพื่อตรวจสอบขั้นการทำงานของวงจรตรรกะไตรภาคชนิดบีที่นำเสนอ มีจำนวนสายที่เลือกเท่ากับจำนวนคู่สายที่เลือกเพื่อตรวจสอบขั้นการทำงานของวงจรรางคู่ อย่างไรก็ตาม วงจรรางคู่ใช้เกตออร์ในการตรวจสอบขั้นการทำงานของคู่สายสัญญาณแต่ละคู่ เกตออร์ใช้ทรานซิสเตอร์จำนวน 6 ตัวต่อเกตออร์หนึ่งตัวในขณะที่การตรวจสอบขั้นการทำงานของวงจรตรรกะไตรภาคชนิดบีใช้อุปกรณ์ตรวจสอบขั้นการทำงานซึ่งใช้ทรานซิสเตอร์จำนวน 2 ตัวต่ออุปกรณ์ตรวจสอบขั้นการทำงานหนึ่งตัว เหตุผลดังกล่าวทำให้วงจรตอบรับสำหรับวงจรตรรกะไตรภาคชนิดบีใช้ทรานซิสเตอร์น้อยกว่าวงจรตอบรับสำหรับวงจรรางคู่

22-35% โดยจำนวนทรานซิสเตอร์ที่ลดลงขึ้นอยู่กับขีดจำกัดของเกตที่ใช้ในวงจรว่าสามารถมีอินพุตสูงสุดได้กี่อินพุต

เมื่อวิเคราะห์จำนวนทรานซิสเตอร์ที่ใช้ทั้งหมดสำหรับวงจรเชิงผสมที่มีวงจรตอบรับที่ตรวจสอบการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายในและมีวงจรเอาต์พุตที่ป้องกันการเปลี่ยนแปลงสัญญาณเอาต์พุตก่อนการสิ้นสุดการเปลี่ยนแปลงสัญญาณภายใน ผลที่ได้จากการทดลองแสดงให้เห็นว่าจำนวนทรานซิสเตอร์ที่ใช้สำหรับวงจรตรรกะไตรภาคชนิดบีน้อยกว่าจำนวนทรานซิสเตอร์ที่วงจรรางคู่ใช้ อย่างไรก็ตาม การนำวงจรตรรกะไตรภาคชนิดบีไปใช้แทนวงจรรางคู่จำเป็นต้องมีการลดขนาดวงจรเชิงผสมเองด้วยเพื่อให้จำนวนทรานซิสเตอร์ที่ใช้ลดลงมากขึ้น ถึงแม้ว่าขนาดของวงจรตรรกะไตรภาคชนิดบีจะมีขนาดเล็กกว่าวงจรรางคู่ วงจรตรรกะไตรภาคชนิดบีก็ยังมีข้อด้อยที่ทำงานช้ากว่าวงจรรางคู่ เนื่องจากเกตที่ใช้กับตรรกะไตรภาคชนิดบีทำงานช้ากว่าเกตทั่วไป

7.2 **ข้อเสนอแนะ**

เนื่องจากขนาดของวงจรตอบรับแปรผันตามจำนวนสายที่ตรวจสอบ ดังนั้นการลดจำนวนสายที่ใช้ตรวจสอบขั้นการทำงานลงจะช่วยให้วงจรตอบรับมีขนาดลดลงด้วย คุณสมบัติของแบบจำลองความหน่วงแบบไม่ไวต่อความหน่วงชนิดเสมือน ที่กำหนดให้วงจรทำงานได้เมื่อความหน่วงของเกตและสายสามารถเป็นค่าใดๆ ที่ไม่ใช่ค่าอนันต์ ทำให้มีขีดจำกัดของการตรวจสอบสายสัญญาณโดยจะต้องตรวจสอบสายสัญญาณเกือบทุกเส้นยกเว้นกิ่งของสายหรืออินพุตของเกตที่มีหนึ่งอินพุต ดังนั้นหากต้องการลดจำนวนสายที่ใช้ในการตรวจสอบขั้นการทำงานลง จำเป็นจะต้องอาศัยการออกแบบโดยใช้แบบจำลองความหน่วงประเภทอื่นแทน

รายการอ้างอิง

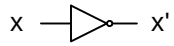
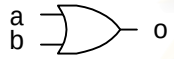
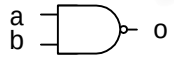
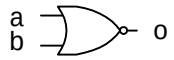
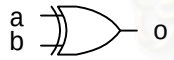
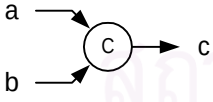
1. Hauck, S. Asynchronous design methodologies: An overview. IEEE Transaction on Computers 83, 1 (January 1995): 69-93.
2. Grass, E.; Morling, R. C. S.; and Kale, I. Activity-Monitoring Completion-Detection (AMCD): a new single rail approach to achieve self-timing. Second International Symposium on Advanced Research in Asynchronous Circuits and Systems (1996): 143-149
3. Nanya, T.; Ueno, Y.; Kagotani, H.; Kuwako, M.; and Takamura, A. TITAC: Design of a Quasi-Delay-Insensitive Microprocessor. IEEE Design & Test of Computers 11, 2 (1994): 50-63.
4. Nagata, Y.; and Mukaidono, M. Design of an Asynchronous Digital System with B-Ternary Logic. Proceedings on 27th IEEE International Symposium on Multiple-Valued Logic (1997): 265-271.
5. Nagata, Y.; Miller, D. M.; and Mukaidono, M. B-Ternary Logic Based Asynchronous Micropipeline. Proceedings on 29th IEEE International Symposium on Multiple-Valued Logic (1999): 214-219.
6. Nagata, Y.; Miller, D.M.; Mukaidono, M. Logic synthesis of controllers for B-ternary asynchronous systems. Proceedings. 30th IEEE International Symposium on Multiple-Valued Logic (2000): 402 –407.
7. Park, S. B. Synthesis of Asynchronous VLSI Circuits from Signal Transition Graph Specifications. Doctoral dissertation, Department of Engineering-Computer Science, Tokyo Institute of Technology (TIT), 1996.
8. Muller, D. E.; Bartky, W. S. A Theory of Asynchronous Circuits. Proceedings on Theory and Switching (1959): 204-243.
9. Wuu, T. Y.; and Vrudhula, S. B. K. A design of a fast and area efficient multi-input Muller C-element. IEEE Transactions on VLSI Systems 1, 2 (June 1993): 215-219.
10. Martin, A. J. The Limitations to Delay-Insensitivity in Asynchronous Circuits. Proceedings on Advanced Research in VLSI 6 (1990): 263-278.
11. Mano, M. M. Digital Design Second Edition. New Jersey: Prentice-Hall, 1991.

12. MUKHERJEE, A. Introduction to nMOS and CMOS VLSI Systems Design. New Jersey: Prentice-Hall, 1986.
13. Chang, K. C. Digital Design and Modeling with VHDL and Synthesis. Los Alamitos, CA: IEEE Computer Society Press, 1997.
14. Yang, S. Logic synthesis and optimization benchmarks userguide version 3.0 [Online]. Research Triangle Park, NC: Microelectronics Center of North Carolina, 1991. Available from: http://www.cbl.ncsu.edu/ [2000, May 12]

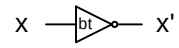
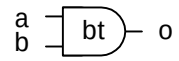
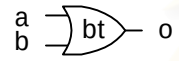
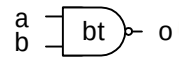
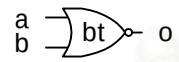
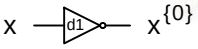
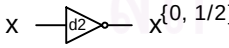
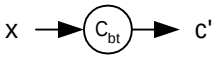
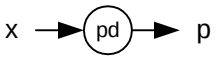
ภาคผนวก

ภาคผนวก ก
สัญลักษณ์และความหมายของเกตและอุปกรณ์ที่ใช้ในการออกแบบวงจร

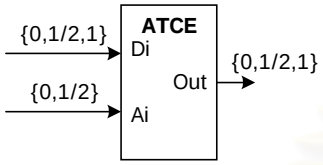
ตารางที่ ก.1 สัญลักษณ์ ชื่อ และตารางค่าความจริงของเกตและอุปกรณ์ที่ใช้กับวงจรฐานสอง

สัญลักษณ์	ชื่อ	ตารางค่าความจริง
	เกตตัวผกผัน (Inverter gate)	x | 0 | 1 x' | 1 | 1
	เกตแอนด์ (AND gate)	a\b | 0 | 1 0 | 0 | 0 1 | 0 | 1
	เกตออร์ (OR gate)	a\b | 0 | 1 0 | 0 | 1 1 | 1 | 1
	เกตแนนด์ (NAND gate)	a\b | 0 | 1 0 | 1 | 1 1 | 1 | 0
	เกตนอร์ (NOR gate)	a\b | 0 | 1 0 | 1 | 0 1 | 0 | 0
	เกตออร์เฉพาะ (XOR gate)	a\b | 0 | 1 0 | 0 | 1 1 | 1 | 0
	อุปกรณ์ชนิดซีของมูลเลอร์ (Muller's C-element)	c\ab | 00 | 01 | 11 | 10 0 | 0 | 0 | 1 | 0 1 | 0 | 1 | 1 | 1

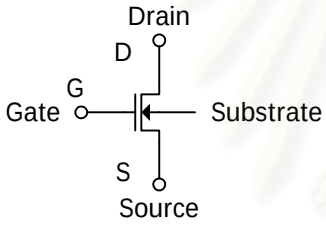
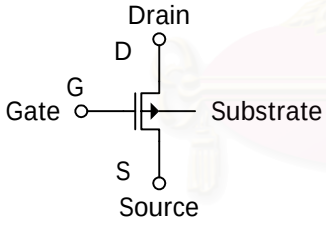
ตารางที่ ก.2 สัญลักษณ์ ชื่อ และตารางค่าความจริงของเกตและอุปกรณ์ที่ใช้กับวงจรตรรกะไตรภาคชนิดบี

สัญลักษณ์	ชื่อ	ตารางค่าความจริง
	เกตตัวผกผัน (Inverter gate)	x | 0 | 1/2 | 1 x' | 1 | 1/2 | 0
	เกตแอนด์ (AND gate)	a\b | 0 | 1/2 | 1 0 | 0 | 0 | 0 1/2 | 0 | 1/2 | 1/2 1 | 0 | 1/2 | 1
	เกตออร์ (OR gate)	a\b | 0 | 1/2 | 1 0 | 0 | 1/2 | 1 1/2 | 1/2 | 1/2 | 1 1 | 1 | 1 | 1
	เกตแนนด์ (NAND gate)	a\b | 0 | 1/2 | 1 0 | 1 | 1 | 1 1/2 | 1 | 1/2 | 1/2 1 | 1 | 1/2 | 0
	เกตนอร์ (NOR gate)	a\b | 0 | 1/2 | 1 0 | 1 | 1/2 | 0 1/2 | 1/2 | 1/2 | 0 1 | 0 | 0 | 0
	สัญพจน์ล่างชนิดดีหนึ่ง (down-literal d1)	x | 0 | 1/2 | 1 x{0} | 1 | 0 | 0
	สัญพจน์ล่างชนิดดีสอง (down-literal d2)	x | 0 | 1/2 | 1 x{0, 1/2} | 1 | 1 | 0
	อุปกรณ์ชนิดซีประเภทเข้าแบบไตรภาคออกแบบฐานสอง (Ternary-in binary-out C-element)	c\x | 0 | 1/2 | 1 0 | 0 | 0 | 1 1 | 0 | 1 | 1
	อุปกรณ์ตรวจสอบขั้นการทำงาน (Phase detector)	x | 0 | 1/2 | 1 p | 0 | 1/2 | 0

ตารางที่ ก.2 สัญลักษณ์ ชื่อ และตารางค่าความจริงของเกตและอุปกรณ์ที่ใช้กับวงจรตรรกะไตรภาคชนิดบี (ต่อ)

สัญลักษณ์	ชื่อ	ตารางค่าความจริง
	อุปกรณ์ชนิดซีแบบไตรภาคชนิดอสมมาตร (Asymmetric Ternary C-element)	AiDi / Out: 00, 0½, 01, ½0, ½½, ½1 0: 0, 0, 1, 0, 1/2, 0 1/2: 0, 1/2, 1, 1/2, 1/2, 1/2 1: 0, 1, 1, 1, 1/2, 1

ตารางที่ ก.3 สัญลักษณ์ และชื่อทรานซิสเตอร์

สัญลักษณ์	ชื่อ
	ทรานซิสเตอร์ชนิดมอสช่องเอ็น (*n-channel* MOS transistor : *n*MOS)
	ทรานซิสเตอร์ชนิดมอสช่องพี (*p-channel* MOS transistor : *p*MOS)

ภาคผนวก ข

รหัสต้นฉบับของโปรแกรมสำเร็จ BT_LOGIC_LIB

```
-- ---------------------------------------------------------------------
--
--   Title     :  B-ternary_Logic multi-value system
--   Library   :  This package shall be compiled into a library
--             :  symbolically named BT.
--             :
--   Developers:  Kawee Wattanaviroon
--   Purpose   :  This packages defines a B-Ternary logic standard for
--             :  designers to use in describing the interconnection data
--             :  types used in vhdl modeling.
--             :
--   Limitation:  The logic system defined in this package may
--             :  be insufficient for modeling switched transistors,
--             :  since such a requirement is out of the scope of this
--             :  effort. Furthermore, mathematics, primitives,
--             :  timing standards, etc. are considered orthogonal
--             :  issues as it relates to this package and are therefore
--             :  beyond the scope of this effort.
--             :
--   Note      :  No declarations or definitions shall be included in,
--             :  or excluded from this package. The "package declaration"
--             :  defines the types, subtypes and declarations of
--             :  BT_Logic. The BT_Logic package body shall be
--             :  considered the formal definition of the semantics of
--             :  this package. Tool developers may choose to implement
--             :  the package body in the most efficient manner available
--             :  to them.
--             :
-- ---------------------------------------------------------------------
--   modification history :
-- ---------------------------------------------------------------------
--  mod. date:   | Changed:
--  DEC 11, 2000 | 1. Add functions :
--               |    resolved, bt_and(), bt_nand(), bt_or(), bt_nor(),
--               |    bt_xor(), bt_xnor(), bt_inv(), bt_ltr0s(), bt_ltr0(),
--               |    To_bit(), To_bitvector(),
--               |    To_stdUlogic(), To_stdLogicVector(), To_stdULogicVector(),
--               |    To_btULogic(), To_btLogicVector(), To_btULogicvector(),
--               |    To_X0S1(), To_X0S1Z(), To_UX0S1(),
--               |    plus_edge(), minus_edge(), return_edge(), Is_X()
--  JUL 20, 2001 | 1. Add bt_pd() function(s).
--  SEP 09, 2001 | 1. Add overload function(s) "and", "nand", "or", "nor", "xor",
--               |    "xnor", "not"
-- ---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

PACKAGE BT_LOGIC_LIB IS
    ---------------------------------------------------------------------
    -- logic state system  (unresolved)
    ---------------------------------------------------------------------
    TYPE bt_ulogic IS (  'U',  -- Uninitialized
                         'X',  -- Forcing  Unknown
                         '0',  -- Forcing  0
                         'S',  -- Forcing  Spacer
                         '1',  -- Forcing  1
                         'Z',  -- High Impedance
                         'P',  -- Forcing  Unknown {0, S}
                         'Q',  -- Forcing  Unknown {S, 1}
                         'R',  -- Forcing  Unknown {0, 1}
                         'W',  -- Weak     Unknown
                         'L',  -- Weak     0
                         'B',  -- Weak Spacer (Below Spacer)
                         'A',  -- Weak Spacer (Above Spacer)
                         'H',  -- Weak     1
                         'I',  -- Weak     Unknown {0, S}
                         'J',  -- Weak     Unknown {S, 1}
                         'K',  -- Weak     Unknown {0, 1}
                         '-'   -- Don't care
                       );
    ---------------------------------------------------------------------
    -- unconstrained array of bt_ulogic for use with the resolution function
    ---------------------------------------------------------------------
    TYPE bt_ulogic_vector IS ARRAY ( NATURAL RANGE <> ) OF bt_ulogic;

    ---------------------------------------------------------------------
    -- resolution function
    ---------------------------------------------------------------------
    FUNCTION resolved ( s : bt_ulogic_vector ) RETURN bt_ulogic;


    ---------------------------------------------------------------------
    -- *** industry standard logic type ***
    ---------------------------------------------------------------------
    SUBTYPE bt_logic IS resolved bt_ulogic;

    ---------------------------------------------------------------------
    -- unconstrained array of bt_logic for use in declaring signal arrays
    ---------------------------------------------------------------------
    TYPE bt_logic_vector IS ARRAY ( NATURAL RANGE <>) OF bt_logic;

    ---------------------------------------------------------------------
    -- common subtypes
    ---------------------------------------------------------------------
    SUBTYPE X0S1    IS resolved bt_ulogic RANGE 'X' TO '1'; -- ('X','0','S','1')
    SUBTYPE X0S1Z   IS resolved bt_ulogic RANGE 'X' TO 'Z'; -- ('X','0','S','1','Z')
```

```
SUBTYPE UX0S1   IS resolved bt_ulogic RANGE 'U' TO '1'; -- ('U','X','0','S','1')
SUBTYPE UX0S1Z  IS resolved bt_ulogic RANGE 'U' TO 'Z'; -- ('U','X','0','S','1','Z')

-------------------------------------------------------------------
-- overloaded logical function
-------------------------------------------------------------------

FUNCTION "and"    ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION "nand"   ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION "or"     ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION "nor"    ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION "xor"    ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
function "xnor"   ( l : bt_ulogic; r : bt_ulogic ) return UX0S1;
FUNCTION "not"    ( l : bt_ulogic                ) RETURN UX0S1;
FUNCTION bt_and   ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION bt_nand  ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION bt_or    ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION bt_nor   ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
FUNCTION bt_xor   ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1;
function bt_xnor  ( l : bt_ulogic; r : bt_ulogic ) return UX0S1;
FUNCTION bt_inv   ( l : bt_ulogic                ) RETURN UX0S1;
FUNCTION bt_ltr0s ( l : bt_ulogic                ) RETURN UX0S1; --literals function {0, S}
FUNCTION bt_ltr0  ( l : bt_ulogic                ) RETURN UX0S1; --literals function {0}
FUNCTION bt_pd    ( l : bt_ulogic                ) RETURN UX0S1; --phase detector

-------------------------------------------------------------------
-- vectorized overloaded logical operators
-------------------------------------------------------------------
FUNCTION "and"   ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION "and"   ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION "nand"  ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION "nand"  ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION "or"    ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION "or"    ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION "nor"   ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION "nor"   ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION "xor"   ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION "xor"   ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

function "xnor"  ( l, r : bt_logic_vector  ) return bt_logic_vector;
function "xnor"  ( l, r : bt_ulogic_vector ) return bt_ulogic_vector;

FUNCTION "not"   ( l : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION "not"   ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION bt_and   ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION bt_and   ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION bt_nand  ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION bt_nand  ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION bt_or    ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION bt_or    ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION bt_nor   ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION bt_nor   ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION bt_xor   ( l, r : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION bt_xor   ( l, r : bt_ulogic_vector ) RETURN bt_ulogic_vector;

function bt_xnor  ( l, r : bt_logic_vector  ) return bt_logic_vector;
function bt_xnor  ( l, r : bt_ulogic_vector ) return bt_ulogic_vector;

FUNCTION bt_inv   ( l : bt_logic_vector  ) RETURN bt_logic_vector;
FUNCTION bt_inv   ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector;

FUNCTION bt_ltr0s ( l : bt_logic_vector  ) RETURN bt_logic_vector;  --literals function {0, S}
FUNCTION bt_ltr0s ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector; --literals function {0, S}

FUNCTION bt_ltr0  ( l : bt_logic_vector  ) RETURN bt_logic_vector;  --literals function {0}
FUNCTION bt_ltr0  ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector; --literals function {0}

FUNCTION bt_pd    ( l : bt_logic_vector  ) RETURN bt_logic_vector;  --phase detector
FUNCTION bt_pd    ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector; --phase detector

-------------------------------------------------------------------
-- conversion functions
-------------------------------------------------------------------
FUNCTION To_bit         ( t : bt_ulogic        ; xmap : BIT := '0') RETURN BIT;
FUNCTION To_bitvector   ( t : bt_logic_vector  ; xmap : BIT := '0') RETURN BIT_VECTOR;
FUNCTION To_bitvector   ( t : bt_ulogic_vector ; xmap : BIT := '0') RETURN BIT_VECTOR;

FUNCTION To_stdUlogic       ( t : bt_ulogic         ; xmap : std_ulogic := 'X') RETURN std_ulogic;
FUNCTION To_stdLogicVector  ( t : bt_ulogic_vector  ; xmap : std_logic  := 'X') RETURN std_logic_vector;
FUNCTION To_stdLogicVector  ( t : bt_logic_vector   ; xmap : std_logic  := 'X') RETURN std_logic_vector;
FUNCTION To_stdULogicVector ( t : bt_ulogic_vector  ; xmap : std_ulogic := 'X') RETURN std_ulogic_vector;
FUNCTION To_stdULogicVector ( t : bt_logic_vector   ; xmap : std_ulogic := 'X') RETURN std_ulogic_vector;

FUNCTION To_btULogic       ( b : BIT               ) RETURN bt_ulogic;
FUNCTION To_btULogic       ( s : std_ulogic        ) RETURN bt_ulogic;
FUNCTION To_btLogicVector  ( b : BIT_VECTOR        ) RETURN bt_logic_vector;
FUNCTION To_btLogicVector  ( s : std_ulogic_vector ) RETURN bt_logic_vector;
FUNCTION To_btLogicVector  ( s : std_logic_vector  ) RETURN bt_logic_vector;
FUNCTION To_btLogicVector  ( t : bt_ulogic_vector  ) RETURN bt_logic_vector;
FUNCTION To_btULogicVector ( b : BIT_VECTOR        ) RETURN bt_ulogic_vector;
```

```
    FUNCTION To_btULogicVector ( s : std_ulogic_vector ) RETURN bt_ulogic_vector;
    FUNCTION To_btULogicVector ( s : std_logic_vector  ) RETURN bt_ulogic_vector;
    FUNCTION To_btULogicVector ( t : bt_logic_vector   ) RETURN bt_ulogic_vector;

    ---------------------------------------------------------------------
    -- strength strippers and type convertors
    ---------------------------------------------------------------------
    FUNCTION To_X0S1  ( t : bt_logic_vector   ) RETURN  bt_logic_vector;
    FUNCTION To_X0S1  ( t : bt_ulogic_vector  ) RETURN  bt_ulogic_vector;
    FUNCTION To_X0S1  ( t : bt_ulogic         ) RETURN  X0S1;
    FUNCTION To_X0S1  ( s : std_logic_vector  ) RETURN  bt_logic_vector;
    FUNCTION To_X0S1  ( s : std_ulogic_vector ) RETURN  bt_ulogic_vector;
    FUNCTION To_X0S1  ( s : std_ulogic        ) RETURN  X0S1;
    FUNCTION To_X0S1  ( b : BIT_VECTOR        ) RETURN  bt_logic_vector;
    FUNCTION To_X0S1  ( b : BIT_VECTOR        ) RETURN  bt_ulogic_vector;
    FUNCTION To_X0S1  ( b : BIT               ) RETURN  X0S1;

    FUNCTION To_X0S1Z ( t : bt_logic_vector   ) RETURN  bt_logic_vector;
    FUNCTION To_X0S1Z ( t : bt_ulogic_vector  ) RETURN  bt_ulogic_vector;
    FUNCTION To_X0S1Z ( t : bt_ulogic         ) RETURN  X0S1Z;
    FUNCTION To_X0S1Z ( s : std_logic_vector  ) RETURN  bt_logic_vector;
    FUNCTION To_X0S1Z ( s : std_ulogic_vector ) RETURN  bt_ulogic_vector;
    FUNCTION To_X0S1Z ( s : std_ulogic        ) RETURN  X0S1Z;
    FUNCTION To_X0S1Z ( b : BIT_VECTOR        ) RETURN  bt_logic_vector;
    FUNCTION To_X0S1Z ( b : BIT_VECTOR        ) RETURN  bt_ulogic_vector;
    FUNCTION To_X0S1Z ( b : BIT               ) RETURN  X0S1Z;

    FUNCTION To_UX0S1 ( t : bt_logic_vector   ) RETURN  bt_logic_vector;
    FUNCTION To_UX0S1 ( t : bt_ulogic_vector  ) RETURN  bt_ulogic_vector;
    FUNCTION To_UX0S1 ( t : bt_ulogic         ) RETURN  UX0S1;
    FUNCTION To_UX0S1 ( s : std_logic_vector  ) RETURN  bt_logic_vector;
    FUNCTION To_UX0S1 ( s : std_ulogic_vector ) RETURN  bt_ulogic_vector;
    FUNCTION To_UX0S1 ( s : std_ulogic        ) RETURN  UX0S1;
    FUNCTION To_UX0S1 ( b : BIT_VECTOR        ) RETURN  bt_logic_vector;
    FUNCTION To_UX0S1 ( b : BIT_VECTOR        ) RETURN  bt_ulogic_vector;
    FUNCTION To_UX0S1 ( b : BIT               ) RETURN  UX0S1;

    ---------------------------------------------------------------------
    -- edge detection
    ---------------------------------------------------------------------
    FUNCTION plus_edge   (SIGNAL t : bt_ulogic) RETURN BOOLEAN;
    FUNCTION minus_edge  (SIGNAL t : bt_ulogic) RETURN BOOLEAN;
    FUNCTION return_edge (SIGNAL t : bt_ulogic) RETURN BOOLEAN;

    ---------------------------------------------------------------------
    -- object contains an unknown
    ---------------------------------------------------------------------
    FUNCTION Is_X ( t : bt_ulogic_vector ) RETURN  BOOLEAN;
    FUNCTION Is_X ( t : bt_logic_vector  ) RETURN  BOOLEAN;
    FUNCTION Is_X ( t : bt_ulogic        ) RETURN  BOOLEAN;

END BT_LOGIC_LIB;

PACKAGE BODY BT_LOGIC_LIB IS
    ---------------------------------------------------------------------
    -- local types
    ---------------------------------------------------------------------
    TYPE btlogic_1d IS ARRAY (bt_ulogic) OF bt_ulogic;
    TYPE btlogic_table IS ARRAY(bt_ulogic, bt_ulogic) OF bt_ulogic;

    ---------------------------------------------------------------------
    -- resolution function
    ---------------------------------------------------------------------
    CONSTANT resolution_table : btlogic_table := (
    --      ---------------------------------------------------------------------------------------------------
    --      |  U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -        |   |
    --      ---------------------------------------------------------------------------------------------------
            ( 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ), -- | U |
            ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | X |
            ( 'U', 'X', '0', 'P', 'R', '0', 'P', 'X', 'R', '0', '0', '0', '0', '0', '0', '0', 'X' ), -- | 0 |
            ( 'U', 'X', 'P', 'S', 'Q', 'S', 'P', 'Q', 'X', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'X' ), -- | S |
            ( 'U', 'X', 'R', 'Q', '1', '1', 'X', 'Q', 'R', '1', '1', '1', '1', '1', '1', '1', 'X' ), -- | 1 |
            ( 'U', 'X', '0', 'S', '1', 'Z', 'P', 'Q', 'R', 'W', 'L', 'B', 'A', 'H', 'I', 'J', 'K', 'X' ), -- | Z |
            ( 'U', 'X', 'P', 'P', 'X', 'P', 'P', 'X', 'X', 'P', 'P', 'P', 'P', 'P', 'P', 'P', 'P', 'X' ), -- | P |
            ( 'U', 'X', 'X', 'Q', 'Q', 'Q', 'X', 'Q', 'X', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', 'X' ), -- | Q |
            ( 'U', 'X', 'R', 'X', 'R', 'R', 'X', 'X', 'R', 'R', 'R', 'R', 'R', 'R', 'R', 'R', 'X' ), -- | R |
            ( 'U', 'X', '0', 'S', '1', 'W', 'P', 'Q', 'R', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'X' ), -- | W |
            ( 'U', 'X', '0', 'S', '1', 'L', 'P', 'Q', 'R', 'W', 'L', 'I', 'I', 'K', 'I', 'W', 'K', 'X' ), -- | L |
            ( 'U', 'X', '0', 'S', '1', 'B', 'P', 'Q', 'R', 'W', 'I', 'B', 'S', 'J', 'I', 'J', 'W', 'X' ), -- | B |
            ( 'U', 'X', '0', 'S', '1', 'A', 'P', 'Q', 'R', 'W', 'I', 'S', 'A', 'J', 'I', 'J', 'W', 'X' ), -- | A |
            ( 'U', 'X', '0', 'S', '1', 'H', 'P', 'Q', 'R', 'W', 'K', 'J', 'J', 'H', 'W', 'J', 'K', 'X' ), -- | H |
            ( 'U', 'X', '0', 'S', '1', 'I', 'P', 'Q', 'R', 'W', 'I', 'I', 'I', 'W', 'I', 'W', 'W', 'X' ), -- | I |
            ( 'U', 'X', '0', 'S', '1', 'J', 'P', 'Q', 'R', 'W', 'W', 'J', 'J', 'J', 'W', 'J', 'W', 'X' ), -- | J |
            ( 'U', 'X', '0', 'S', '1', 'K', 'P', 'Q', 'R', 'W', 'K', 'W', 'W', 'K', 'W', 'W', 'K', 'X' ), -- | K |
            ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' )  -- | - |
        );

    FUNCTION resolved ( s : bt_ulogic_vector ) RETURN bt_ulogic IS
        VARIABLE result : bt_ulogic := 'Z';  -- weakest state default
    BEGIN
        -- the test for a single driver is essential otherwise the
        -- loop would return 'X' for a single driver of '-' and that
        -- would conflict with the value of a single driver unresolved
        -- signal.
        IF    (s'LENGTH = 1) THEN    RETURN s(s'LOW);
        ELSE
            FOR i IN s'RANGE LOOP
                result := resolution_table(result, s(i));
            END LOOP;
```

```
    END IF;
    RETURN result;
END resolved;

--------------------------------------------------------------------
-- tables for logical operations
--------------------------------------------------------------------

-- truth table for bt_and function
CONSTANT bt_and_table : btlogic_table := (
--      ---------------------------------------------------------------------------------------------
--      |  U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -    |   |
--      ---------------------------------------------------------------------------------------------
        ( 'U', 'U', '0', 'U', 'U', 'U', 'U', 'U', 'U', 'U', '0', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ),  -- | U |
        ( 'U', 'X', '0', 'P', 'X', 'X', 'P', 'X', 'X', 'X', '0', 'P', 'P', 'X', 'P', 'X', 'X', 'X' ),  -- | X |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  -- | 0 |
        ( 'U', 'P', '0', 'S', 'S', 'P', 'P', 'S', 'P', 'P', '0', 'S', 'S', 'S', 'P', 'S', 'P', 'P' ),  -- | S |
        ( 'U', 'X', '0', 'S', '1', 'X', 'P', 'Q', 'R', 'X', '0', 'S', 'S', '1', 'P', 'Q', 'R', 'X' ),  -- | 1 |
        ( 'U', 'X', '0', 'P', 'X', 'X', 'P', 'X', 'X', 'X', '0', 'P', 'P', 'X', 'P', 'X', 'X', 'X' ),  -- | Z |
        ( 'U', 'P', '0', 'P', 'P', 'P', 'P', 'P', 'P', 'P', '0', 'P', 'P', 'P', 'P', 'P', 'P', 'P' ),  -- | P |
        ( 'U', 'X', '0', 'S', 'Q', 'X', 'P', 'Q', 'X', 'X', '0', 'S', 'S', 'Q', 'P', 'Q', 'X', 'X' ),  -- | Q |
        ( 'U', 'X', '0', 'P', 'R', 'X', 'P', 'X', 'R', 'X', '0', 'P', 'P', 'R', 'P', 'X', 'R', 'X' ),  -- | R |
        ( 'U', 'X', '0', 'P', 'X', 'X', 'P', 'X', 'X', 'X', '0', 'P', 'P', 'X', 'P', 'X', 'X', 'X' ),  -- | W |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  -- | L |
        ( 'U', 'P', '0', 'S', 'S', 'P', 'P', 'S', 'P', 'P', '0', 'S', 'S', 'S', 'P', 'S', 'P', 'P' ),  -- | B |
        ( 'U', 'P', '0', 'S', 'S', 'P', 'P', 'S', 'P', 'P', '0', 'S', 'S', 'S', 'P', 'S', 'P', 'P' ),  -- | A |
        ( 'U', 'X', '0', 'S', '1', 'X', 'P', 'Q', 'R', 'X', '0', 'S', 'S', '1', 'P', 'Q', 'R', 'X' ),  -- | H |
        ( 'U', 'P', '0', 'P', 'P', 'P', 'P', 'P', 'P', 'P', '0', 'P', 'P', 'P', 'P', 'P', 'P', 'P' ),  -- | I |
        ( 'U', 'X', '0', 'S', 'Q', 'X', 'P', 'Q', 'X', 'X', '0', 'S', 'S', 'Q', 'P', 'Q', 'X', 'X' ),  -- | J |
        ( 'U', 'X', '0', 'P', 'R', 'X', 'P', 'X', 'R', 'X', '0', 'P', 'P', 'R', 'P', 'X', 'R', 'X' ),  -- | K |
        ( 'U', 'X', '0', 'P', 'X', 'X', 'P', 'X', 'X', 'X', '0', 'P', 'P', 'X', 'P', 'X', 'X', 'X' )   -- | - |
);

-- truth table for bt_or function
CONSTANT bt_or_table : btlogic_table := (
--      ---------------------------------------------------------------------------------------------
--      |  U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -    |   |
--      ---------------------------------------------------------------------------------------------
        ( 'U', 'U', 'U', 'U', '1', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', '1', 'U', 'U', 'U', 'U' ),  -- | U |
        ( 'U', 'X', 'X', 'Q', '1', 'X', 'X', 'Q', 'X', 'X', 'X', 'Q', 'Q', '1', 'X', 'Q', 'X', 'X' ),  -- | X |
        ( 'U', 'X', '0', 'S', '1', 'X', 'P', 'Q', 'R', 'X', '0', 'S', 'S', '1', 'P', 'Q', 'R', 'X' ),  -- | 0 |
        ( 'U', 'Q', 'S', 'S', '1', 'Q', 'S', 'Q', 'Q', 'Q', 'S', 'S', 'S', '1', 'S', 'Q', 'Q', 'Q' ),  -- | S |
        ( '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1' ),  -- | 1 |
        ( 'U', 'X', 'X', 'Q', '1', 'X', 'X', 'Q', 'X', 'X', 'X', 'Q', 'Q', '1', 'X', 'Q', 'X', 'X' ),  -- | Z |
        ( 'U', 'X', 'P', 'S', '1', 'X', 'P', 'Q', 'X', 'X', 'P', 'S', 'S', '1', 'P', 'Q', 'X', 'X' ),  -- | P |
        ( 'U', 'Q', 'Q', 'Q', '1', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', '1', 'Q', 'Q', 'Q', 'Q' ),  -- | Q |
        ( 'U', 'X', 'R', 'Q', '1', 'X', 'X', 'Q', 'R', 'X', 'R', 'Q', 'Q', '1', 'X', 'Q', 'R', 'X' ),  -- | R |
        ( 'U', 'X', 'X', 'Q', '1', 'X', 'X', 'Q', 'X', 'X', 'X', 'Q', 'Q', '1', 'X', 'Q', 'X', 'X' ),  -- | W |
        ( 'U', 'X', '0', 'S', '1', 'X', 'P', 'Q', 'R', 'X', '0', 'S', 'S', '1', 'P', 'Q', 'R', 'X' ),  -- | L |
        ( 'U', 'Q', 'S', 'S', '1', 'Q', 'S', 'Q', 'Q', 'Q', 'S', 'S', 'S', '1', 'S', 'Q', 'Q', 'Q' ),  -- | B |
        ( 'U', 'Q', 'S', 'S', '1', 'Q', 'S', 'Q', 'Q', 'Q', 'S', 'S', 'S', '1', 'S', 'Q', 'Q', 'Q' ),  -- | A |
        ( '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1', '1' ),  -- | H |
        ( 'U', 'X', 'P', 'S', '1', 'X', 'P', 'Q', 'X', 'X', 'P', 'S', 'S', '1', 'P', 'Q', 'X', 'X' ),  -- | I |
        ( 'U', 'Q', 'Q', 'Q', '1', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', 'Q', '1', 'Q', 'Q', 'Q', 'Q' ),  -- | J |
        ( 'U', 'X', 'R', 'Q', '1', 'X', 'X', 'Q', 'R', 'X', 'R', 'Q', 'Q', '1', 'X', 'Q', 'R', 'X' ),  -- | K |
        ( 'U', 'X', 'X', 'Q', '1', 'X', 'X', 'Q', 'X', 'X', 'X', 'Q', 'Q', '1', 'X', 'Q', 'X', 'X' )   -- | - |
);

-- truth table for bt_xor function
CONSTANT bt_xor_table : btlogic_table := (
--      ---------------------------------------------------------------------------------------------
--      |  U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -    |   |
--      ---------------------------------------------------------------------------------------------
        ( 'U', 'U', 'U', 'S', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'S', 'S', 'U', 'U', 'U', 'U', 'U' ),  -- | U |
        ( 'U', 'X', 'X', 'S', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'S', 'S', 'X', 'X', 'X', 'X', 'X' ),  -- | X |
        ( 'U', 'X', '0', 'S', '1', 'X', 'P', 'Q', 'R', 'X', '0', 'S', 'S', '1', 'P', 'Q', 'R', 'X' ),  -- | 0 |
        ( 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S' ),  -- | S |
        ( 'U', 'X', '1', 'S', '0', 'X', 'Q', 'P', 'R', 'X', '1', 'S', 'S', '0', 'Q', 'P', 'R', 'X' ),  -- | 1 |
        ( 'U', 'X', 'X', 'S', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'S', 'S', 'X', 'X', 'X', 'X', 'X' ),  -- | Z |
        ( 'U', 'X', 'P', 'S', 'Q', 'X', 'P', 'Q', 'X', 'X', 'P', 'S', 'S', 'Q', 'P', 'Q', 'X', 'X' ),  -- | P |
        ( 'U', 'X', 'Q', 'S', 'P', 'X', 'Q', 'P', 'X', 'X', 'Q', 'S', 'S', 'P', 'Q', 'P', 'X', 'X' ),  -- | Q |
        ( 'U', 'X', 'R', 'S', 'R', 'X', 'X', 'X', 'R', 'X', 'R', 'S', 'S', 'R', 'X', 'X', 'R', 'X' ),  -- | R |
        ( 'U', 'X', 'X', 'S', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'S', 'S', 'X', 'X', 'X', 'X', 'X' ),  -- | W |
        ( 'U', 'X', '0', 'S', '1', 'X', 'P', 'Q', 'R', 'X', '0', 'S', 'S', '1', 'P', 'Q', 'R', 'X' ),  -- | L |
        ( 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S' ),  -- | B |
        ( 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S', 'S' ),  -- | A |
        ( 'U', 'X', '1', 'S', '0', 'X', 'Q', 'P', 'R', 'X', '1', 'S', 'S', '0', 'Q', 'P', 'R', 'X' ),  -- | H |
        ( 'U', 'X', 'P', 'S', 'Q', 'X', 'P', 'Q', 'X', 'X', 'P', 'S', 'S', 'Q', 'P', 'Q', 'X', 'X' ),  -- | I |
        ( 'U', 'X', 'Q', 'S', 'P', 'X', 'Q', 'P', 'X', 'X', 'Q', 'S', 'S', 'P', 'Q', 'P', 'X', 'X' ),  -- | J |
        ( 'U', 'X', 'R', 'S', 'R', 'X', 'X', 'X', 'R', 'X', 'R', 'S', 'S', 'R', 'X', 'X', 'R', 'X' ),  -- | K |
        ( 'U', 'X', 'X', 'S', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'S', 'S', 'X', 'X', 'X', 'X', 'X' )   -- | - |
);

-- truth table for bt_inv function
CONSTANT bt_inv_table: btlogic_1d :=
--  ---------------------------------------------------------------------------------------------
--  |  U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -   |
--  ---------------------------------------------------------------------------------------------
     ( 'U', 'X', '1', 'S', '0', 'X', 'Q', 'P', 'R', 'X', '1', 'S', 'S', '0', 'Q', 'P', 'R', 'X' );

-- truth table for bt_ltr0s function
CONSTANT bt_ltr0s_table: btlogic_1d :=
--  ---------------------------------------------------------------------------------------------
--  |  U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -   |
--  ---------------------------------------------------------------------------------------------
     ( 'U', 'X', '1', '1', '0', 'X', 'Q', 'P', 'R', 'X', '1', 'S', 'S', '0', 'Q', 'P', 'R', 'X' );

-- truth table for bt_ltr0 function
CONSTANT bt_ltr0_table: btlogic_1d :=
```

```
--  ------------------------------------------------------------------------------------------------
--  |   U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -   |
--  ------------------------------------------------------------------------------------------------
     ( 'U', 'X', '1', '0', '0', 'X', 'Q', 'P', 'R', 'X', '1', 'S', 'S', '0', 'Q', 'P', 'R', 'X' );

-- truth table for BT_PD_S function
CONSTANT bt_pd_table: btlogic_1d :=
--  ------------------------------------------------------------------------------------------------
--  |   U    X    0    S    1    Z    P    Q    R    W    L    B    A    H    I    J    K    -   |
--  ------------------------------------------------------------------------------------------------
     ( 'U', 'X', '0', 'S', '0', 'X', 'P', 'P', 'P', 'X', '0', 'S', 'S', '0', 'P', 'P', 'P', 'X' );


---------------------------------------------------------------------
-- overloaded logical operators ( with optimizing hints )
---------------------------------------------------------------------

FUNCTION "and"  ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_and_table(l, r);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END "and";

FUNCTION "nand" ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_inv_table ( bt_and_table(l, r));
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' =>
            RETURN rtn;
        when others =>
            RETURN 'X';
    end case;
END "nand";

FUNCTION "or"   ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_or_table(l, r);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END "or";

FUNCTION "nor"  ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_inv_table ( bt_or_table( l, r ));
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END "nor";

FUNCTION "xor"  ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_xor_table(l, r);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END "xor";

function "xnor"  ( l : bt_ulogic; r : bt_ulogic ) return ux0s1 is
    VARIABLE rtn : bt_logic;
begin
    rtn := bt_inv_table(bt_xor_table(l, r));
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
end "xnor";

FUNCTION "not"  ( l : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_inv_table(l);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END "not";

FUNCTION bt_and  ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_and_table(l, r);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_and;
```

```
FUNCTION bt_nand ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_inv_table ( bt_and_table(l, r));
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' =>
            RETURN rtn;
        when others =>
            RETURN 'X';
    end case;
END bt_nand;

FUNCTION bt_or   ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_or_table(l, r);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_or;

FUNCTION bt_nor  ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_inv_table ( bt_or_table( l, r ));
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_nor;

FUNCTION bt_xor  ( l : bt_ulogic; r : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_xor_table(l, r);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_xor;

function bt_xnor  ( l : bt_ulogic; r : bt_ulogic ) return ux0s1 is
    VARIABLE rtn : bt_logic;
begin
    rtn := bt_inv_table(bt_xor_table(l, r));
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
end bt_xnor;

FUNCTION bt_inv  ( l : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_inv_table(l);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_inv;

FUNCTION bt_ltr0s ( l : bt_ulogic  ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_ltr0s_table(l);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_ltr0s;

FUNCTION bt_ltr0  ( l : bt_ulogic  ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_ltr0_table(l);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_ltr0;

FUNCTION bt_pd ( l : bt_ulogic ) RETURN UX0S1 IS
    VARIABLE rtn : bt_logic;
BEGIN
    rtn := bt_pd_table(l);
    case rtn is
        when 'U' | 'X' | '0' | 'S' | '1' => RETURN rtn;
        when others => RETURN 'X';
    end case;
END bt_pd;

-----------------------------------------------------------------------
-- and
-----------------------------------------------------------------------
FUNCTION "and"  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
```

```
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_and' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_and_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "and";
-----------------------------------------------------------------------
FUNCTION "and"  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_and' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_and_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "and";
-----------------------------------------------------------------------
-- nand
-----------------------------------------------------------------------
FUNCTION "nand"  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nand' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_and_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "nand";
-----------------------------------------------------------------------
FUNCTION "nand"  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nand' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_and_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "nand";
-----------------------------------------------------------------------
-- or
-----------------------------------------------------------------------
FUNCTION "or"  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_or' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_or_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
```

```
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "or";
-------------------------------------------------------------------
FUNCTION "or"  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_or' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_or_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "or";
-------------------------------------------------------------------
-- nor
-------------------------------------------------------------------
FUNCTION "nor"  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nor' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_or_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "nor";
-------------------------------------------------------------------
FUNCTION "nor"  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nor' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_or_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "nor";
-------------------------------------------------------------------
-- xor
-------------------------------------------------------------------
FUNCTION "xor"  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_xor' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_xor_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "xor";
-------------------------------------------------------------------
FUNCTION "xor"  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
```

```
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_xor' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_xor_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END "xor";
-------------------------------------------------------------------
-- xnor
-------------------------------------------------------------------
FUNCTION "xnor"  ( l,r : bt_logic_vector ) return bt_logic_vector is
    alias lv : bt_logic_vector ( 1 to l'length ) is l;
    alias rv : bt_logic_vector ( 1 to r'length ) is r;
    VARIABLE result : bt_logic_vector ( 1 to l'length );
begin
    if ( l'length /= r'length ) then
        assert false
        report "arguments of overloaded 'bt_xnor' operator are not of the same length"
        severity failure;
    else
        for i in result'range loop
            result(i) := bt_inv_table(bt_xor_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        end loop;
    end if;
    return result;
END "xnor";
-------------------------------------------------------------------
FUNCTION "xnor"  ( l,r : bt_ulogic_vector ) return bt_ulogic_vector is
    alias lv : bt_ulogic_vector ( 1 to l'length ) is l;
    alias rv : bt_ulogic_vector ( 1 to r'length ) is r;
    VARIABLE result : bt_ulogic_vector ( 1 to l'length );
begin
    if ( l'length /= r'length ) then
        assert false
        report "arguments of overloaded 'bt_xnor' operator are not of the same length"
        severity failure;
    else
        for i in result'range loop
            result(i) := bt_inv_table(bt_xor_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        end loop;
    end if;
    return result;
END "xnor";
-------------------------------------------------------------------
-- not
-------------------------------------------------------------------
FUNCTION "not"  ( l : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_inv_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION "not"  ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_inv_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
-- bt_and
-------------------------------------------------------------------
FUNCTION bt_and  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
```

```
        REPORT "arguments of overloaded 'bt_and' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_and_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_and;
---------------------------------------------------------------------
FUNCTION bt_and  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_and' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_and_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_and;
---------------------------------------------------------------------
-- bt_nand
---------------------------------------------------------------------
FUNCTION bt_nand  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nand' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_and_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_nand;
---------------------------------------------------------------------
FUNCTION bt_nand  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nand' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_and_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_nand;
---------------------------------------------------------------------
-- bt_or
---------------------------------------------------------------------
FUNCTION bt_or  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_or' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_or_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
```

```
    RETURN result;
END bt_or;
-------------------------------------------------------------------
FUNCTION bt_or  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_or' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_or_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_or;
-------------------------------------------------------------------
-- bt_nor
-------------------------------------------------------------------
FUNCTION bt_nor  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nor' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_or_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_nor;
-------------------------------------------------------------------
FUNCTION bt_nor  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_nor' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_inv_table(bt_or_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_nor;
-------------------------------------------------------------------
-- bt_xor
-------------------------------------------------------------------
FUNCTION bt_xor  ( l,r : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_logic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_xor' operator are not of the same length"
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_xor_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_xor;
-------------------------------------------------------------------
FUNCTION bt_xor  ( l,r : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    ALIAS rv : bt_ulogic_vector ( 1 TO r'LENGTH ) IS r;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH );
BEGIN
    IF ( l'LENGTH /= r'LENGTH ) THEN
        ASSERT FALSE
        REPORT "arguments of overloaded 'bt_xor' operator are not of the same length"
```

```
        SEVERITY FAILURE;
    ELSE
        FOR i IN result'RANGE LOOP
            result(i) := bt_xor_table (lv(i), rv(i));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        END LOOP;
    END IF;
    RETURN result;
END bt_xor;
---------------------------------------------------------------------
-- bt_xnor
---------------------------------------------------------------------
FUNCTION bt_xnor  ( l,r : bt_logic_vector ) return bt_logic_vector is
    alias lv : bt_logic_vector ( 1 to l'length ) is l;
    alias rv : bt_logic_vector ( 1 to r'length ) is r;
    VARIABLE result : bt_logic_vector ( 1 to l'length );
begin
    if ( l'length /= r'length ) then
        assert false
        report "arguments of overloaded 'bt_xnor' operator are not of the same length"
        severity failure;
    else
        for i in result'range loop
            result(i) := bt_inv_table(bt_xor_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        end loop;
    end if;
    return result;
END bt_xnor;
---------------------------------------------------------------------
FUNCTION bt_xnor  ( l,r : bt_ulogic_vector ) return bt_ulogic_vector is
    alias lv : bt_ulogic_vector ( 1 to l'length ) is l;
    alias rv : bt_ulogic_vector ( 1 to r'length ) is r;
    VARIABLE result : bt_ulogic_vector ( 1 to l'length );
begin
    if ( l'length /= r'length ) then
        assert false
        report "arguments of overloaded 'bt_xnor' operator are not of the same length"
        severity failure;
    else
        for i in result'range loop
            result(i) := bt_inv_table(bt_xor_table (lv(i), rv(i)));
            case result(i) is
                when 'U' | 'X' | '0' | 'S' | '1' => NULL;
                when others => result(i) := 'X';
            end case;
        end loop;
    end if;
    return result;
END bt_xnor;
---------------------------------------------------------------------
-- bt_inv
---------------------------------------------------------------------
FUNCTION bt_inv  ( l : bt_logic_vector ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_inv_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END;
---------------------------------------------------------------------
FUNCTION bt_inv  ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_inv_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END;
---------------------------------------------------------------------
-- bt_ltr0s
---------------------------------------------------------------------
FUNCTION bt_ltr0s ( l : bt_logic_vector  ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector (1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_ltr0s_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
```

```
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION bt_ltr0s ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector (1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_ltr0s_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
-- bt_ltr0
-------------------------------------------------------------------
FUNCTION bt_ltr0  ( l : bt_logic_vector  ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector (1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_logic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_ltr0_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION bt_ltr0  ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector (1 TO l'LENGTH ) IS l;
    VARIABLE result : bt_ulogic_vector ( 1 TO l'LENGTH ) := (OTHERS => 'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_ltr0_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
-- bt_pd
-------------------------------------------------------------------
FUNCTION bt_pd ( l : bt_logic_vector  ) RETURN bt_logic_vector IS
    ALIAS lv : bt_logic_vector ( 1 TO l'LENGTH ) is l;
    VARIABLE result : bt_logic_vector (1 TO l'LENGTH ) := (OTHERS =>'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_pd_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END bt_pd;
-------------------------------------------------------------------
FUNCTION bt_pd ( l : bt_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS lv : bt_ulogic_vector ( 1 TO l'LENGTH ) is l;
    VARIABLE result : bt_ulogic_vector (1 TO l'LENGTH ) := (OTHERS =>'X');
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := bt_pd_table( lv(i) );
        case result(i) is
            when 'U' | 'X' | '0' | 'S' | '1' => NULL;
            when others => result(i) := 'X';
        end case;
    END LOOP;
    RETURN result;
END bt_pd;

-------------------------------------------------------------------
-- conversion tables
-------------------------------------------------------------------
TYPE logic_x0s1_table  IS ARRAY (bt_ulogic'LOW TO bt_ulogic'HIGH) OF X0S1;
TYPE logic_x0s1z_table IS ARRAY (bt_ulogic'LOW TO bt_ulogic'HIGH) OF X0S1Z;
TYPE logic_ux0s1_table IS ARRAY (bt_ulogic'LOW TO bt_ulogic'HIGH) OF UX0S1;
TYPE logic_std_to_x0s1_table  IS ARRAY (std_ulogic'LOW TO std_ulogic'HIGH) OF X0S1;
TYPE logic_std_to_x0s1z_table IS ARRAY (std_ulogic'LOW TO std_ulogic'HIGH) OF X0S1Z;
TYPE logic_std_to_ux0s1_table IS ARRAY (std_ulogic'LOW TO std_ulogic'HIGH) OF UX0S1;
-----------------------------------------------------------
-- table name : cvt_to_x0s1
--
-- parameters :
--        in  :  bt_ulogic   -- some logic value
-- returns    :  x0s1        -- state value of logic value
-- purpose    :  to convert state-strength to state only
--
-- example    : if (cvt_to_x0s1 (input_signal) = '1' ) then ...
--
-----------------------------------------------------------
CONSTANT cvt_to_x0s1 : logic_x0s1_table := (
```

```
                    'X',  -- 'U'
                    'X',  -- 'X'
                    '0',  -- '0'
                    'S',  -- 'S'
                    '1',  -- '1'
                    'X',  -- 'Z'
                    'X',  -- 'P'
                    'X',  -- 'Q'
                    'X',  -- 'R'
                    'X',  -- 'W'
                    '0',  -- 'L'
                    'S',  -- 'B'
                    'S',  -- 'A'
                    '1',  -- 'H'
                    'X',  -- 'I'
                    'X',  -- 'J'
                    'X',  -- 'K'
                    'X'   -- '-'
                   );

------------------------------------------------------------
-- table name : cvt_to_x0s1z
--
-- parameters :
--        in  :  bt_ulogic  -- some logic value
-- returns    :  x0s1z       -- state value of logic value
-- purpose    :  to convert state-strength to state only
--
-- example    : if (cvt_to_x0s1z (input_signal) = '1' ) then ...
--
------------------------------------------------------------
CONSTANT cvt_to_x0s1z : logic_x0s1z_table := (
                    'X',  -- 'U'
                    'X',  -- 'X'
                    '0',  -- '0'
                    'S',  -- 'S'
                    '1',  -- '1'
                    'Z',  -- 'Z'
                    'X',  -- 'P'
                    'X',  -- 'Q'
                    'X',  -- 'R'
                    'X',  -- 'W'
                    '0',  -- 'L'
                    'S',  -- 'B'
                    'S',  -- 'A'
                    '1',  -- 'H'
                    'X',  -- 'I'
                    'X',  -- 'J'
                    'X',  -- 'K'
                    'X'   -- '-'
                   );

------------------------------------------------------------
-- table name : cvt_to_ux0s1
--
-- parameters :
--        in  :  bt_ulogic  -- some logic value
-- returns    :  ux0s1       -- state value of logic value
-- purpose    :  to convert state-strength to state only
--
-- example    : if (cvt_to_ux0s1 (input_signal) = '1' ) then ...
--
------------------------------------------------------------
CONSTANT cvt_to_ux0s1 : logic_ux0s1_table := (
                    'U',  -- 'U'
                    'X',  -- 'X'
                    '0',  -- '0'
                    'S',  -- 'S'
                    '1',  -- '1'
                    'X',  -- 'Z'
                    'X',  -- 'P'
                    'X',  -- 'Q'
                    'X',  -- 'R'
                    'X',  -- 'W'
                    '0',  -- 'L'
                    'S',  -- 'B'
                    'S',  -- 'A'
                    '1',  -- 'H'
                    'X',  -- 'I'
                    'X',  -- 'J'
                    'X',  -- 'K'
                    'X'   -- '-'
                   );
CONSTANT cvt_std_to_x0s1 : logic_std_to_x0s1_table := (
                    'X',  -- 'U'
                    'X',  -- 'X'
                    '0',  -- '0'
                    '1',  -- '1'
                    'X',  -- 'Z'
                    'X',  -- 'W'
                    '0',  -- 'L'
                    '1',  -- 'H'
                    'X'   -- '-'
                   );
CONSTANT cvt_std_to_x0s1z : logic_std_to_x0s1z_table := (
                    'X',  -- 'U'
                    'X',  -- 'X'
                    '0',  -- '0'
                    '1',  -- '1'
```

```
                    'Z',  -- 'Z'
                    'X',  -- 'W'
                    '0',  -- 'L'
                    '1',  -- 'H'
                    'X'   -- '-'
                   );
CONSTANT cvt_std_to_ux0s1 : logic_std_to_ux0s1_table := (
                    'U',  -- 'U'
                    'X',  -- 'X'
                    '0',  -- '0'
                    '1',  -- '1'
                    'X',  -- 'Z'
                    'X',  -- 'W'
                    '0',  -- 'L'
                    '1',  -- 'H'
                    'X'   -- '-'
                   );

-------------------------------------------------------------------
-- conversion functions
-------------------------------------------------------------------
FUNCTION To_bit       ( t : bt_ulogic;         xmap : BIT := '0') RETURN BIT IS
BEGIN
    CASE t IS
        WHEN '0' | 'L' => RETURN ('0');
        WHEN '1' | 'H' => RETURN ('1');
        WHEN OTHERS => RETURN xmap;
    END CASE;
END;
-------------------------------------------------------------------
FUNCTION To_bitvector ( t : bt_logic_vector ; xmap : BIT := '0') RETURN BIT_VECTOR IS
    ALIAS tv : bt_logic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : BIT_VECTOR ( t'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE tv(i) IS
            WHEN '0' | 'L' => result(i) := '0';
            WHEN '1' | 'H' => result(i) := '1';
            WHEN OTHERS => result(i) := xmap;
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_bitvector ( t : bt_ulogic_vector; xmap : BIT := '0') RETURN BIT_VECTOR IS
    ALIAS tv : bt_ulogic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : BIT_VECTOR ( t'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE tv(i) IS
            WHEN '0' | 'L' => result(i) := '0';
            WHEN '1' | 'H' => result(i) := '1';
            WHEN OTHERS => result(i) := xmap;
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
-------------------------------------------------------------------
FUNCTION To_stdUlogic       ( t : bt_ulogic         ; xmap : std_ulogic := 'X') RETURN std_ulogic IS
BEGIN
    CASE t IS
        WHEN 'U' => return 'U';
        WHEN 'X' => return 'X';
        WHEN '0' => return '0';
        WHEN '1' => return '1';
        WHEN 'Z' => return 'Z';
        WHEN 'W' => return 'W';
        WHEN 'L' => return 'L';
        WHEN 'H' => return 'H';
        WHEN '-' => return '-';
        WHEN OTHERS => return xmap;
    END CASE;
END;
-------------------------------------------------------------------
FUNCTION To_stdLogicVector  ( t : bt_ulogic_vector  ; xmap : std_logic  := 'X') RETURN std_logic_vector IS
    ALIAS tv : bt_ulogic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : std_logic_vector ( t'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE tv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := xmap;
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_stdLogicVector  ( t : bt_logic_vector   ; xmap : std_logic  := 'X') RETURN std_logic_vector IS
    ALIAS tv : bt_logic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : std_logic_vector ( t'LENGTH-1 DOWNTO 0 );
```

```
BEGIN
    FOR i IN result'RANGE LOOP
        CASE tv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := xmap;
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_stdULogicVector ( t : bt_ulogic_vector  ; xmap : std_ulogic := 'X') RETURN std_ulogic_vector IS
    ALIAS tv : bt_ulogic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : std_ulogic_vector ( t'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE tv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := xmap;
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_stdULogicVector ( t : bt_logic_vector   ; xmap : std_ulogic := 'X') RETURN std_ulogic_vector IS
    ALIAS tv : bt_logic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : std_ulogic_vector ( t'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE tv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := xmap;
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
--------------------------------------------------------------------
FUNCTION To_btULogic       ( b : BIT               ) RETURN bt_ulogic IS
BEGIN
    CASE b IS
        WHEN '0' => RETURN '0';
        WHEN '1' => RETURN '1';
    END CASE;
END;
--------------------------------------------------------------------
FUNCTION To_btULogic       ( s : std_ulogic        ) RETURN bt_ulogic IS
BEGIN
    CASE s IS
        WHEN 'U' => return 'U';
        WHEN 'X' => return 'X';
        WHEN '0' => return '0';
        WHEN '1' => return '1';
        WHEN 'Z' => return 'Z';
        WHEN 'W' => return 'W';
        WHEN 'L' => return 'L';
        WHEN 'H' => return 'H';
        WHEN '-' => return '-';
        WHEN OTHERS => return 'X';
    END CASE;
END;
--------------------------------------------------------------------
FUNCTION To_btLogicVector  ( b : BIT_VECTOR        ) RETURN bt_logic_vector IS
    ALIAS bv : BIT_VECTOR ( b'LENGTH-1 DOWNTO 0 ) IS b;
    VARIABLE result : bt_logic_vector ( b'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_btLogicVector  ( s : std_ulogic_vector ) RETURN bt_logic_vector IS
```

```
    ALIAS sv : std_ulogic_vector ( s'LENGTH-1 DOWNTO 0 ) IS s;
    VARIABLE result : bt_logic_vector ( s'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE sv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := 'X';
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_btLogicVector  ( s : std_logic_vector  ) RETURN bt_logic_vector IS
    ALIAS sv : std_logic_vector ( s'LENGTH-1 DOWNTO 0 ) IS s;
    VARIABLE result : bt_logic_vector ( s'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE sv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := 'X';
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_btLogicVector  ( t : bt_ulogic_vector  ) RETURN bt_logic_vector IS
    ALIAS tv : bt_ulogic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : bt_logic_vector ( t'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := tv(i);
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_btULogicVector ( b : BIT_VECTOR        ) RETURN bt_ulogic_vector IS
    ALIAS bv : BIT_VECTOR ( b'LENGTH-1 DOWNTO 0 ) IS b;
    VARIABLE result : bt_ulogic_vector ( b'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_btULogicVector ( s : std_ulogic_vector ) RETURN bt_ulogic_vector IS
    ALIAS sv : std_ulogic_vector ( s'LENGTH-1 DOWNTO 0 ) IS s;
    VARIABLE result : bt_ulogic_vector ( s'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE sv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := 'X';
        END CASE;
    END LOOP;
    RETURN result;
END;
--------------------------------------------------------------------
FUNCTION To_btULogicVector ( s : std_logic_vector  ) RETURN bt_ulogic_vector IS
    ALIAS sv : std_logic_vector ( s'LENGTH-1 DOWNTO 0 ) IS s;
    VARIABLE result : bt_ulogic_vector ( s'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE sv(i) IS
            WHEN 'U' => result(i) := 'U';
            WHEN 'X' => result(i) := 'X';
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
            WHEN 'Z' => result(i) := 'Z';
            WHEN 'W' => result(i) := 'W';
            WHEN 'L' => result(i) := 'L';
            WHEN 'H' => result(i) := 'H';
```

```
            WHEN '-' => result(i) := '-';
            WHEN OTHERS => result(i) := 'X';
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_btULogicVector ( t : bt_logic_vector   ) RETURN bt_ulogic_vector IS
    ALIAS tv : bt_logic_vector ( t'LENGTH-1 DOWNTO 0 ) IS t;
    VARIABLE result : bt_ulogic_vector ( t'LENGTH-1 DOWNTO 0 );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := tv(i);
    END LOOP;
    RETURN result;
END;

-------------------------------------------------------------------
-- strength strippers and type convertors
-------------------------------------------------------------------
-- to_x0s1
-------------------------------------------------------------------
FUNCTION To_X0S1  ( t : bt_logic_vector ) RETURN  bt_logic_vector IS
    ALIAS tv : bt_logic_vector ( 1 TO t'LENGTH ) IS t;
    VARIABLE result : bt_logic_vector ( 1 TO t'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_to_x0s1 (tv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( t : bt_ulogic_vector ) RETURN  bt_ulogic_vector IS
    ALIAS tv : bt_ulogic_vector ( 1 TO t'LENGTH ) IS t;
    VARIABLE result : bt_ulogic_vector ( 1 TO t'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_to_x0s1 (tv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( t : bt_ulogic ) RETURN  X0S1 IS
BEGIN
    RETURN (cvt_to_x0s1(t));
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( s : std_logic_vector ) RETURN  bt_logic_vector IS
    ALIAS sv : std_logic_vector ( 1 TO s'LENGTH ) IS s;
    VARIABLE result : bt_logic_vector ( 1 TO s'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_std_to_x0s1 (sv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( s : std_ulogic_vector ) RETURN  bt_ulogic_vector IS
    ALIAS sv : std_ulogic_vector ( 1 TO s'LENGTH ) IS s;
    VARIABLE result : bt_ulogic_vector ( 1 TO s'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_std_to_x0s1 (sv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( s : std_ulogic ) RETURN  X0S1 IS
BEGIN
    RETURN (cvt_std_to_x0s1(s));
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( b : BIT_VECTOR ) RETURN  bt_logic_vector IS
    ALIAS bv : BIT_VECTOR ( 1 TO b'LENGTH ) IS b;
    VARIABLE result : bt_logic_vector ( 1 TO b'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( b : BIT_VECTOR ) RETURN  bt_ulogic_vector IS
    ALIAS bv : BIT_VECTOR ( 1 TO b'LENGTH ) IS b;
    VARIABLE result : bt_ulogic_vector ( 1 TO b'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1  ( b : BIT ) RETURN  X0S1 IS
```

```
BEGIN
        CASE b IS
            WHEN '0' => RETURN('0');
            WHEN '1' => RETURN('1');
        END CASE;
END;

-------------------------------------------------------------------
-- to_x0s1z
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( t : bt_logic_vector ) RETURN  bt_logic_vector IS
    ALIAS tv : bt_logic_vector ( 1 TO t'LENGTH ) IS t;
    VARIABLE result : bt_logic_vector ( 1 TO t'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_to_x0s1z (tv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( t : bt_ulogic_vector ) RETURN  bt_ulogic_vector IS
    ALIAS tv : bt_ulogic_vector ( 1 TO t'LENGTH ) IS t;
    VARIABLE result : bt_ulogic_vector ( 1 TO t'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_to_x0s1z (tv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( t : bt_ulogic ) RETURN  X0S1Z IS
BEGIN
    RETURN (cvt_to_x0s1z(t));
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( s : std_logic_vector ) RETURN  bt_logic_vector IS
    ALIAS sv : std_logic_vector ( 1 TO s'LENGTH ) IS s;
    VARIABLE result : bt_logic_vector ( 1 TO s'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_std_to_x0s1z (sv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( s : std_ulogic_vector ) RETURN  bt_ulogic_vector IS
    ALIAS sv : std_ulogic_vector ( 1 TO s'LENGTH ) IS s;
    VARIABLE result : bt_ulogic_vector ( 1 TO s'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_std_to_x0s1z (sv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( s : std_ulogic ) RETURN  X0S1Z IS
BEGIN
    RETURN (cvt_std_to_x0s1z(s));
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( b : BIT_VECTOR ) RETURN  bt_logic_vector IS
    ALIAS bv : BIT_VECTOR ( 1 TO b'LENGTH ) IS b;
    VARIABLE result : bt_logic_vector ( 1 TO b'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( b : BIT_VECTOR ) RETURN  bt_ulogic_vector IS
    ALIAS bv : BIT_VECTOR ( 1 TO b'LENGTH ) IS b;
    VARIABLE result : bt_ulogic_vector ( 1 TO b'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_X0S1Z  ( b : BIT ) RETURN  X0S1Z IS
BEGIN
        CASE b IS
            WHEN '0' => RETURN('0');
            WHEN '1' => RETURN('1');
        END CASE;
END;

-------------------------------------------------------------------
-- to_ux0s1
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( t : bt_logic_vector ) RETURN  bt_logic_vector IS
    ALIAS tv : bt_logic_vector ( 1 TO t'LENGTH ) IS t;
```

```
    VARIABLE result : bt_logic_vector ( 1 TO t'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_to_ux0s1 (tv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( t : bt_ulogic_vector ) RETURN  bt_ulogic_vector IS
    ALIAS tv : bt_ulogic_vector ( 1 TO t'LENGTH ) IS t;
    VARIABLE result : bt_ulogic_vector ( 1 TO t'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_to_ux0s1 (tv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( t : bt_ulogic ) RETURN  UX0S1 IS
BEGIN
    RETURN (cvt_to_ux0s1(t));
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( s : std_logic_vector ) RETURN  bt_logic_vector IS
    ALIAS sv : std_logic_vector ( 1 TO s'LENGTH ) IS s;
    VARIABLE result : bt_logic_vector ( 1 TO s'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_std_to_ux0s1 (sv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( s : std_ulogic_vector ) RETURN  bt_ulogic_vector IS
    ALIAS sv : std_ulogic_vector ( 1 TO s'LENGTH ) IS s;
    VARIABLE result : bt_ulogic_vector ( 1 TO s'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        result(i) := cvt_std_to_ux0s1 (sv(i));
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( s : std_ulogic ) RETURN  UX0S1 IS
BEGIN
    RETURN (cvt_std_to_ux0s1(s));
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( b : BIT_VECTOR ) RETURN  bt_logic_vector IS
    ALIAS bv : BIT_VECTOR ( 1 TO b'LENGTH ) IS b;
    VARIABLE result : bt_logic_vector ( 1 TO b'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( b : BIT_VECTOR ) RETURN  bt_ulogic_vector IS
    ALIAS bv : BIT_VECTOR ( 1 TO b'LENGTH ) IS b;
    VARIABLE result : bt_ulogic_vector ( 1 TO b'LENGTH );
BEGIN
    FOR i IN result'RANGE LOOP
        CASE bv(i) IS
            WHEN '0' => result(i) := '0';
            WHEN '1' => result(i) := '1';
        END CASE;
    END LOOP;
    RETURN result;
END;
-------------------------------------------------------------------
FUNCTION To_UX0S1  ( b : BIT ) RETURN  UX0S1 IS
BEGIN
        CASE b IS
            WHEN '0' => RETURN('0');
            WHEN '1' => RETURN('1');
        END CASE;
END;

-------------------------------------------------------------------
-- edge detection
-------------------------------------------------------------------
FUNCTION plus_edge  (SIGNAL t : bt_ulogic) RETURN BOOLEAN IS
BEGIN
    RETURN (t'EVENT AND (To_X0S1(t) = '1') AND
                        (To_X0S1(t'LAST_VALUE) = 'S'));
END;
-------------------------------------------------------------------
FUNCTION minus_edge (SIGNAL t : bt_ulogic) RETURN BOOLEAN IS
BEGIN
    RETURN (t'EVENT AND (To_X0S1(t) = '0') AND
                        (To_X0S1(t'LAST_VALUE) = 'S'));
END;
-------------------------------------------------------------------
FUNCTION return_edge (SIGNAL t : bt_ulogic) RETURN BOOLEAN IS
BEGIN
```

```
        RETURN (t'EVENT AND (To_X0S1(t) = 'S') AND
                            ((To_X0S1(t'LAST_VALUE) = '0') OR
                            (To_X0S1(t'LAST_VALUE) = '1')));
    END;

    ---------------------------------------------------------------------
    -- object contains an unknown
    ---------------------------------------------------------------------
    FUNCTION Is_X ( t : bt_ulogic_vector ) RETURN  BOOLEAN IS
    BEGIN
        FOR i IN t'RANGE LOOP
            CASE t(i) IS
                WHEN 'U' | 'X' | 'Z' | 'P' | 'Q' | 'R' | 'W' | 'I' | 'J' | 'K' | '-' => RETURN TRUE;
                WHEN OTHERS => NULL;
            END CASE;
        END LOOP;
        RETURN FALSE;
    END;
    ---------------------------------------------------------------------
    FUNCTION Is_X ( t : bt_logic_vector  ) RETURN  BOOLEAN IS
    BEGIN
        FOR i IN t'RANGE LOOP
            CASE t(i) IS
                WHEN 'U' | 'X' | 'Z' | 'P' | 'Q' | 'R' | 'W' | 'I' | 'J' | 'K' | '-' => RETURN TRUE;
                WHEN OTHERS => NULL;
            END CASE;
        END LOOP;
        RETURN FALSE;
    END;
    ---------------------------------------------------------------------
    FUNCTION Is_X ( t : bt_ulogic        ) RETURN  BOOLEAN IS
    BEGIN
        CASE t IS
            WHEN 'U' | 'X' | 'Z' | 'P' | 'Q' | 'R' | 'W' | 'I' | 'J' | 'K' | '-' => RETURN TRUE;
            WHEN OTHERS => NULL;
        END CASE;
        RETURN FALSE;
    END;
END BT_LOGIC_LIB;
```

ภาคผนวก ค
รูปแบบแฟ้มข้อมูลนำเข้า

รูปแบบแฟ้มข้อมูลนำเข้า ESPRESSO เป็นรูปแบบแฟ้มข้อมูลนำเข้าที่ใช้อธิบายฟังก์ชันแบบบูล โดยอธิบายอยู่ในรูปเมทริกซ์ (Matrix) อักขระ และมีคำหลัก (Keyword) ที่ใช้ระบุขนาดของเมทริกซ์และรูปแบบของตรรกะของฟังก์ชันอินพุต โดยสามารถใช้เครื่องหมาย # ที่อักขระแรกของบรรทัดเพื่อระบุว่าบรรทัดนั้นเป็นหมายเหตุ (Comment) และสมมุติว่าแต่ละแถวของพีแอลเอ (PLA) คือหนึ่งบรรทัดของแฟ้มข้อมูลนำเข้า ซึ่งคำหลักที่ใช้ในแฟ้มข้อมูลนำเข้า ESPRESSO มีลักษณะดังต่อไปนี้ โดยกำหนดให้ [d] แทนเลขฐานสิบ และ [s] แทนสายอักขระ

.i [d] ระบุจำนวนตัวแปรอินพุต

.o [d] ระบุจำนวนฟังก์ชันเอาต์พุต

.type [s] กำหนดการแปลความหมายทางตรรกะของเมทริกซ์อักขระ ซึ่งสายอักขระ [s] จะถูกระบุด้วยสายอักขระ f, r, fd, fr, dr หรือ fdr อันใดอันหนึ่ง

.phase [s] สายอักขระ [s] ระบุจำนวน 0 หรือ 1 ของฟังก์ชันเอาต์พุตเพื่อประโยชน์ในการลดขนาด

.p [d] ระบุจำนวนผลคูณ (Product Terms)

.e (.end) กำหนดจุดสิ้นสุดของส่วนอธิบายพีแอลเอ

รูปแบบของเมทริกซ์พีแอลเอแต่ละบรรทัดหมายถึงผลคูณหนึ่งพจน์ โดยประกอบด้วยส่วนประกอบสองส่วน คือส่วนซ้ายและส่วนขวาซึ่งคั่นด้วยช่องว่าง ส่วนซ้ายหมายถึงอินพุต และส่วนขวาหมายถึงฟังก์ชันเอาต์พุต ตำแหน่งแต่ละตำแหน่งของส่วนอินพุตหมายถึงตัวแปรอินพุตแต่ละตัวโดยที่อักขระ 0 หมายถึงสัญพจน์ของตัวแปรอินพุตในรูปแบบส่วนเติมเต็ม (Complement) ในผลคูณ อักขระ 1 หมายถึงสัญพจน์ของตัวแปรอินพุตในรูปแบบปกติที่ไม่ใช่รูปแบบส่วนเติมเต็มในผลคูณ และอักขระ – หมายถึงไม่มีตัวแปรนั้นปรากฏในผลคูณ

เนื่องจากความหมายของฟังก์ชันตรรกะแบบบูลสามารถอธิบายได้โดยใช้การแบ่งกลุ่มของผลคูณของอินพุต ซึ่งสามารถแบ่งออกเป็นสามกลุ่มคือ เซตเปิด (ON-set), เซตปิด (OFF-set) และเซตไม่สนใจค่า (DC-set) เซตเปิดของตรรกะแบบบูลหมายถึงเซตของผลคูณที่ทำให้ฟังก์ชันเอาต์พุตให้ผลลัพธ์เป็นค่าตรรกะ 1 เซตปิดหมายถึงเซตของผลคูณที่ทำให้ฟังก์ชันเอาต์พุตให้

ผลลัพธ์เป็นค่าตรรกะ 0 และเซตไม่สนใจค่าหมายถึงเซตผลคูณที่ไม่ระบุผลลัพธ์ของฟังก์ชันเอาต์พุต ดังนั้นการตีความหมายของส่วนฟังก์ชันเอาต์พุตจะมีลักษณะการตีความหมายดังนี้

ตรรกะประเภท f สำหรับฟังก์ชันเอาต์พุตแต่ละฟังก์ชัน อักขระ 1 หมายถึงว่าผลคูณนั้นเป็นสมาชิกของเซตเปิด และอักขระ 0 หรือ – หมายถึงผลว่าคูณนั้นไม่มีความหมายต่อค่าของฟังก์ชัน ฟังก์ชันที่ได้จากตรรกะประเภทนี้จะเป็นฟังก์ชันที่ได้จากเซตเปิดเพียงอย่างเดียว

ตรรกะประเภท fd สำหรับฟังก์ชันเอาต์พุตแต่ละฟังก์ชัน อักขระ 1 หมายถึงว่าผลคูณนั้นเป็นสมาชิกของเซตเปิด อักขระ 0 หมายถึงว่าผลคูณนั้นไม่มีความหมายต่อค่าของฟังก์ชัน และอักขระ – หมายถึงว่าผลคูณนั้นเป็นสมาชิกของเซตไม่สนใจค่า

ตรรกะประเภท fr สำหรับฟังก์ชันเอาต์พุตแต่ละฟังก์ชัน อักขระ 1 หมายถึงว่าผลคูณนั้นเป็นสมาชิกของเซตเปิด อักขระ 0 หมายถึงว่าผลคูณนั้นเป็นสมาชิกของเซตปิด และอักขระ – หมายถึงว่าผลคูณนั้นไม่มีความหมายต่อค่าของฟังก์ชัน

ตรรกะประเภท fdr สำหรับฟังชันเอาต์พุตแต่ละฟังก์ชัน อักขระ 1 หมายถึงว่าผลคูณนั้นเป็นสมาชิกของเซตเปิด อักขระ 0 หมายถึงว่าผลคูณนั้นเป็นสมาชิกของเซตปิด อักขระ – หมาย-ถึงว่าผลคูณนั้นเป็นสมาชิกของกลุ่มที่ไม่สนใจค่า และอักขระ ~ หมายถึงว่าผลคูณนั้นไม่มีความ-หมายต่อค่าของฟังก์ชัน

ตัวอย่าง แฟ้มข้อมูลนำเข้าวงจร con1.pla

```
.i 7
.o 2
.p 9
-1--1-- 10
1-11--- 10
-001--- 10
01---1- 10
-0--0-- 01
1---0-- 01
0-----0 01
01--1-- 01
10-0--- 01
.e
```

ประวัติผู้เขียนวิทยานิพนธ์

นายกวี วัฒนะวิรุณ เกิดเมื่อวันที่ 14 ตุลาคม พ.ศ. 2520 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2541 และศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปีการศึกษา 2542