บทที่ 6

การทดลองบัสระบบ

ในบทนี้กล่าวถึงการการทดลองบัสระบบ ซึ่งประกอบด้วย วัตถุประสงค์ของการทดลอง วิธีการทดลอง การทดลอง และผลการทดลอง โดยทำการทดลองวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่ และวงจรคูณเข้ารหัสหนึ่งในสี่ภายในหน่วยดำเนินการทางคณิตศาสตร์และตรรกะ ด้านการใช้พลังงาน ขนาดวงจรที่ใช้ และความเร็วในการทำงาน สุดท้ายจะโปรแกรมวงจรบัสระบบเข้ารหัสหนึ่งในสี่ลงเอฟพีจีเอ เพื่อทดลองการทำงานของบัสระบบร่วมกับองค์ประกอบ รายละเอียดของแต่ละส่วนมีดังต่อไปนี้

6.1 วัตถุประสงค์ของการทดลอง

1. ทดลองบัสระบบเข้ารหัสหนึ่งในสี่ เพื่อตรวจสอบการทำงานให้มีความถูกต้อง และวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่เทียบกับบัสเข้ารหัสรางคู่ ในด้านของการใช้พลังงาน ขนาดวงจรที่ใช้ และความเร็วในการทำงาน

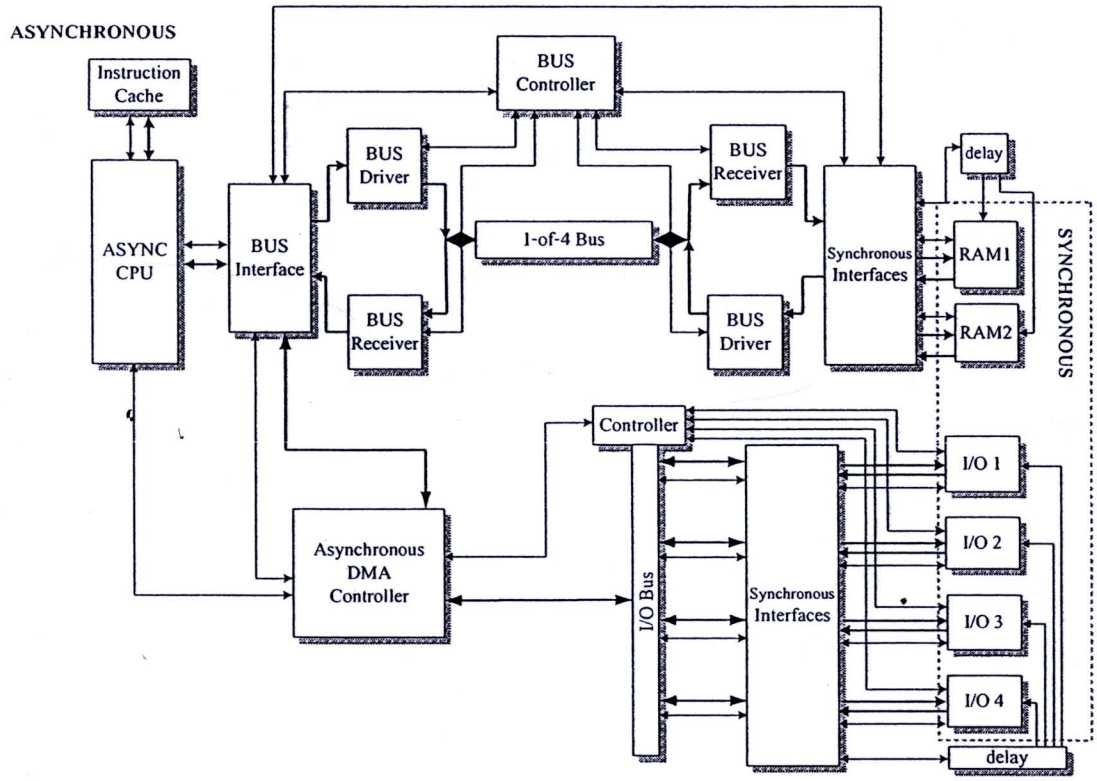
2. ทดลองวงจรคูณเข้ารหัสหนึ่งในสี่ภายในหน่วยดำเนินการทางคณิตศาสตร์และตรรกะ เพื่อตรวจสอบการทำงานให้มีความถูกต้อง และวัดประสิทธิภาพของวงจรคูณใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ เทียบกับวงจรคูณใช้วงจรฟังก์ชันเข้ารหัสรางคู่ ในด้านของการใช้พลังงาน ขนาดวงจรที่ใช้ และความเร็วในการทำงาน

3. ทดลองบัสระบบร่วมกับองค์ประกอบ เพื่อตรวจสอบการทำงานร่วมกันให้มีความถูกต้องบนเอฟพีจีเอ

6.2 วิธีการทดลอง

วิธีการทดลอง แบ่งออกเป็นการสังเคราะห์วงจรบัสระบบแบบอสมวารเข้ารหัสหนึ่งในสี่และองค์ประกอบ ซึ่งแสดงดังรูปที่ 6.1 ที่ได้ออกแบบในบทที่ 3 4 และ 5 ผ่านโปรแกรม Xilinx ISE 11.1 [21] สำหรับวงจรควบคุมที่ออกแบบไว้จะสร้างวงจรโดยใช้โปรแกรม Petrify 4.2 แล้วจึงค่อยนำมาสังเคราะห์วงจร เมื่อสังเคราะห์วงจรเสร็จสิ้น จะนำวงจรที่สังเคราะห์มาอิมพลิเมนท์ลงบนเอฟพีจีเอ โดยใช้โปรแกรม Xilinx ISE 11.1 เช่นกัน แล้วจึงดูผลการทำงานของ

วงจรด้วยการจำลองการทำงานอิงเวลาด้วยโปรแกรม ModelSim XE 6.4b [22] สุดท้ายจะนำวงจรที่ทำงานถูกต้องแล้ว ไปโปรแกรมลงบนเอฟพีจีเอโดยใช้โปรแกรม Xilinx ISE 11.1 รายละเอียดของแต่ละขั้นตอนมีดังต่อไปนี้



รูปที่ 6.1 บัสระบบเข้ารหัสหนึ่งในสี่และองค์ประกอบ

6.2.1 การสร้างวงจรโดยใช้โปรแกรม Petrify 4.2

ตัวอย่างการใช้โปรแกรม Petrify [18] ออกแบบวงจรหน่วงเวลา เป็นดังนี้

i) สร้างไฟล์นามสกุล .g ไว้ในโฟลเดอร์ bin ซึ่งอยู่ในโฟลเดอร์ของโปรแกรม Petrify และตั้งชื่อไฟล์ตามต้องการ ในตัวอย่างนี้ใช้ชื่อ delay.g

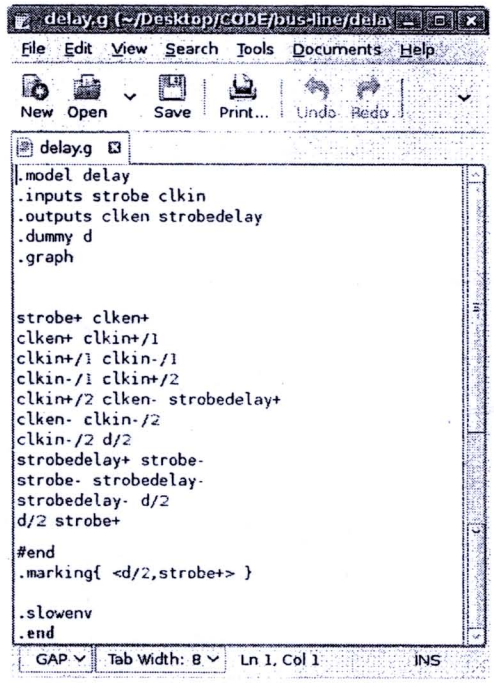
ii) เขียนภาษาอธิบายพฤติกรรมของวงจรตามกราฟ STG ไว้ในไฟล์ .g ในข้อ i. ในตัวอย่างนี้เขียนอธิบายพฤติกรรมตาม่กราฟ STG ในรูปที่ 3.9(ค) จากบทที่ 3 ไว้ที่ไฟล์ delay.g ได้ดังรูปที่ 6.2(ก)

iii) ใช้คำสั่งสร้างไฟล์ .csc และไฟล์ .eqn ผ่าน terminal โดยพิมพ์

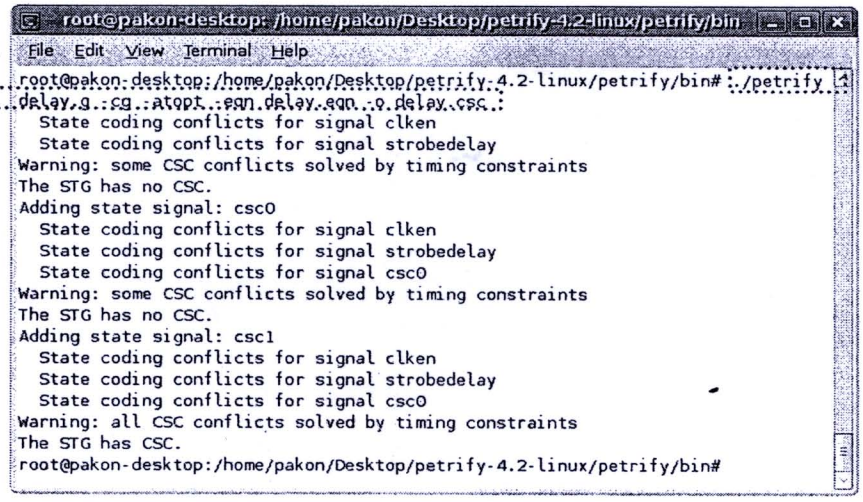
```
./petrify ชื่อไฟล์.g –cg –atopt –egn ชื่อไฟล์.eqn –o ชื่อไฟล์.csc ↵ enter
```

ใช้คำสั่งสร้างไฟล์ .ps จากไฟล์ .g และไฟล์ .csc ผ่าน terminal โดยพิมพ์

./draw_astg -nonames -noinfo -bw ชื่อไฟล์.g -o ชื่อไฟล์.g.nonam.ps ↵ enter
./draw_astg -nonames -noinfo -bw ชื่อไฟล์.csc -o ชื่อไฟล์.csc.nonam.ps ↵ enter

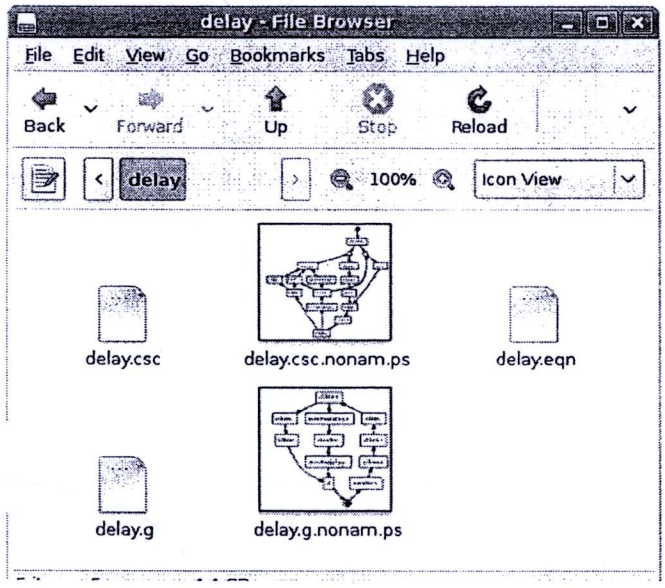


(ก) เขียนภาษาอธิบายพฤติกรรมของวงจรตามกราฟ STG

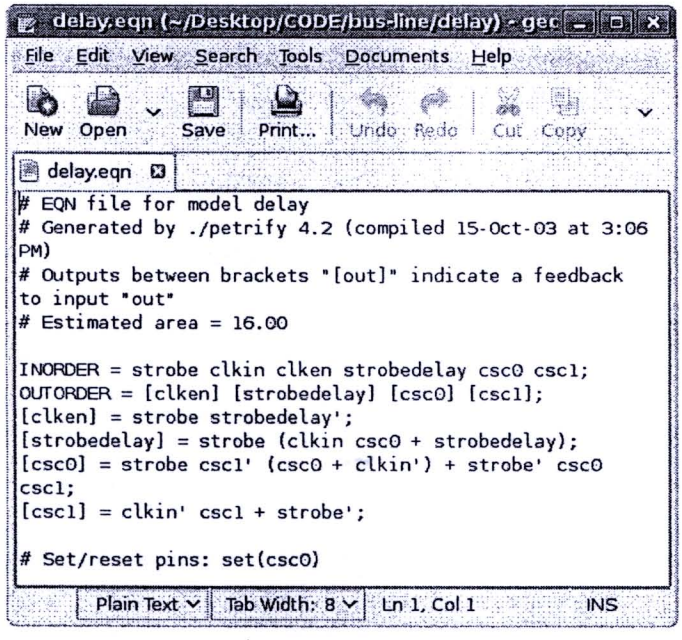


(ข) ใช้คำสั่งสร้างไฟล์วงจร

รูปที่ 6.2 การออกแบบวงจรหน่วงเวลาโดยใช้โปรแกรม Petrify

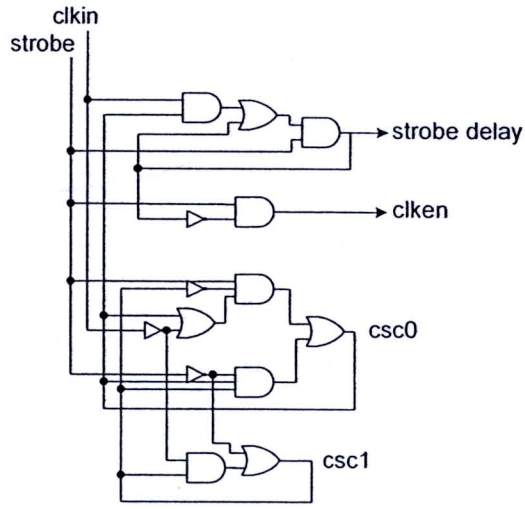


(ค) ไฟล์วงจรที่ได้จากโปรแกรม Petrify



(ง) ไฟล์สมการบูลีน (.eqn)

รูปที่ 6.2 การออกแบบวงจรหน่วงเวลาโดยใช้โปรแกรม Petrify (ต่อ)



(จ) วงจรระดับเกตซึ่งแปลงจากไฟล์สมการบูลีน

รูปที่ 6.2 การออกแบบวงจรหน่วงเวลาโดยใช้โปรแกรม Petrify (ต่อ)

ในตัวอย่างนี้นี้พิมพ์คำสั่งดังกล่าวได้เป็น

./petrify delay.g –cg –atopt –egn delay.eqn –o delay.csc ↵ enter ดังรูปที่ 6.2 (ข)

./draw_astg -nonames -noinfo -bw delay.g -o delay.g.nonam.ps ↵ enter

./draw_astg -nonames -noinfo -bw delay.csc -o delay.csc.nonam.ps ↵ enter

iv) หลังจากใช้คำสั่งในข้อ iii. จะได้ไฟล์เพิ่มขึ้นมาในโฟลเดอร์ bin ประกอบด้วย

ไฟล์ .csc คือไฟล์แก้ไขความซ้ำซ้อนของสถานะบนกราฟ STG (Complete State Coding: CSC) โดยจะสร้างสัญญาณ csc เพิ่มเพื่อให้สถานะบนกราฟ STG ไม่ซ้ำกัน

ไฟล์ .eqn คือไฟล์สมการบูลีนที่มีพฤติกรรมตามกราฟ STG

ไฟล์ delay.g.nonam.ps คือไฟล์ภาพ STG ของไฟล์ .g

ไฟล์ delay.csc.nonam.ps คือไฟล์ภาพ STG ของไฟล์ .csc

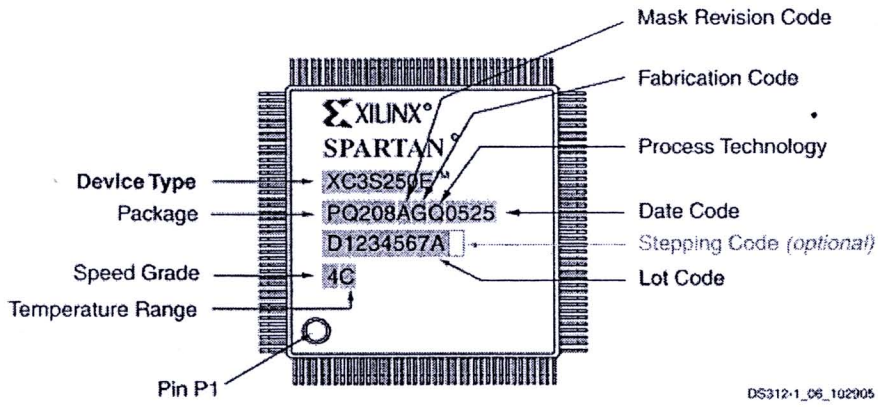
ในตัวอย่างนี้จะได้ไฟล์ในโฟลเดอร์ bin ประกอบด้วย delay.csc delay.eqn delay.g.nonam.ps delay.csc.nonam.ps ดังรูปที่ 6.2(ค)

v) เขียนวงจรจากสมการบูลีน .eqn ในตัวอย่างนี้จะเขียนวงจรจากไฟล์สมการบูลีน delay.eqn ในรูปที่ 6.2(ง) ได้เป็นวงจรหน่วงเวลาดังรูปที่ 6.2(จ) โดยอาจพิจารณาเพิ่มเติมสัญญาณ reset ไว้ในวงจรตามความเหมาะสม

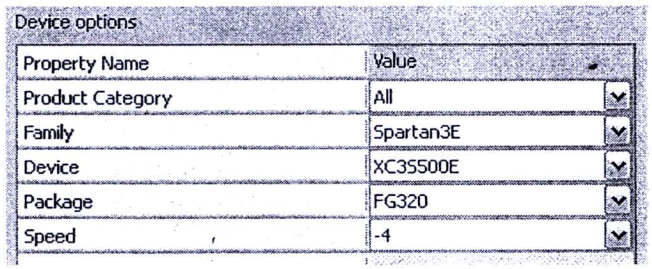
6.2.2 การสังเคราะห์วงจรด้วยโปรแกรม Xilinx ISE 11.1

1. เปิดไฟล์โปรเจควงจรที่ต้องการสังเคราะห์ ซึ่งออกแบบด้วยภาษาเวอริลอคไว้ โดยเปิดโปรแกรม Xilinx เลือกเมนู Open Project จากนั้นเลือกโปรเจคที่ต้องการเปิดแล้วกด Open

2. กำหนดอุปกรณ์เอฟพีจีเอที่ต้องการใช้สังเคราะห์และอิมพลิเมนท์วงจร โดยเลือกเมนู Project>Design Properties และทำการกำหนดคุณสมบัติของอุปกรณ์ซึ่งสามารถสังเกตได้จากชิพบนบอร์ด ตัวอย่างชิพและคุณสมบัติแสดงดังรูปที่ 6.3 [23] สำหรับงานวิจัยนี้ใช้อุปกรณ์เอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S500EFG320 ซึ่งกำหนดคุณสมบัติของอุปกรณ์ได้ดังรูปที่ 6.4

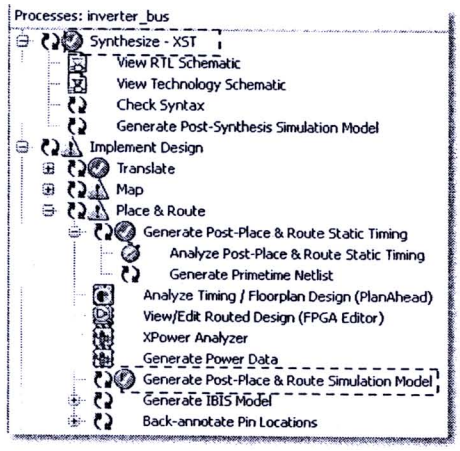


รูปที่ 6.3 ชิพบนบอร์ดเอฟพีจีเอ SPARTAN



รูปที่ 6.4 การกำหนดคุณสมบัติของอุปกรณ์เอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S500EFG320

3. สังเคราะห์วงจรที่ต้องการ โดยการคลิกเลือกไฟล์วงจรที่ต้องการสังเคราะห์ (.v) ซึ่งเรียงลำดับชั้นอยู่ภายในโปรเจคในหน้าต่าง Hierarchy ด้านซ้ายมือ แล้วดับเบิ้ลคลิกที่ Synthesis-XST ในหน้าต่าง Processes ด้านซ้ายมือ ดังแสดงในรูปที่ 6.5(ก) โปรแกรมจะเริ่มต้นสังเคราะห์วงจร และแสดงผลการสังเคราะห์ในหน้าต่าง Console หากการสังเคราะห์สมบูรณ์จะแสดงข้อความ Process "Synthesis" completed successfully



(ก) ตัวเลือกสังเคราะห์และอิมพลิเมนท์วงจร

(ข) ตัวเลือกจำลองการทำงานแบบอิงเวลา

รูปที่ 6.5 ตัวเลือกสังเคราะห์ อิมพลิเมนท์และจำลองการทำงานแบบอิงเวลา ในโปรแกรม Xlinx ISE

6.2.3 การอิมพลิเมนท์วงจรด้วยโปรแกรม Xilinx ISE 11.1

เมื่อสังเคราะห์วงจรสมบูรณ์แล้ว จึงทำการอิมพลิเมนท์โดยคลิกที่ Implement Design>Place&Route และดับเบิ้ลคลิกที่ Generate Post-Place & Route Simulation Model ในหน้าต่าง Processes ด้านซ้ายมือ ดังแสดงในรูปที่ 6.5(ก) โปรแกรมจะเริ่มต้นอิมพลิเมนท์วงจรแบบ Place and Route ซึ่งนำผลของค่าเวลาล่าช้า (Delay Time) ต่างๆในวงจรที่ออกแบบมาคำนวณร่วมกับการจัดวางเกตและเชื่อมสายบนอุปกรณ์เอฟพีจีเอ ผลการอิมพลิเมนท์จะแสดง

ในหน้าต่าง Console หากการอิมพลิเมนท์สมบูรณ์จะแสดงข้อความ Process "Generate Post-Place & Route Simulation Model" completed successfully

6.2.4 การจำลองการทำงานแบบอิงเวลาด้วยโปรแกรม ModelSim XE 6.4b

1. เมื่ออิมพลิเมนท์วงจรสมบูรณ์แล้ว จึงเลือกไฟล์ที่ใช้ทดลองการทำงานของวงจรซึ่งออกแบบด้วยภาษาเวอริลอคไว้ โดยเลือก Source for: Place & Route Simulation จากนั้นคลิกขวาในหน้าต่าง Hierarchy เลือก Add Source เลือกไฟล์ที่ใช้ทดลองที่ต้องการเปิดแล้วกด Open

2. จำลองการทำงานแบบอิงเวลา โดยการคลิกเลือกไฟล์ที่ใช้ทดลองวงจรที่เลือกในข้อ 1 ซึ่งแสดงในหน้าต่าง Hierarchy ด้านซ้ายมือ แล้วคลิกที่ ModelSim Simulator ในหน้าต่าง Processes ด้านซ้ายมือ และดับเบิ้ลคลิกที่ Simulate Post-Place & Route Model ดังแสดงในรูปที่ 6.5(ข) โปรแกรมจะเริ่มต้นจำลองการทำงานแบบอิงเวลา และเปิดโปรแกรม ModelSim ขึ้นมาโดยอัตโนมัติผลการจำลองการทำงานจะแสดงในหน้าต่าง wave ของโปรแกรม ModelSim

ผลจำลองการทำงานของวงจรย่อยอาจมีความถูกต้อง แต่เมื่อนำวงจรย่อยมาเชื่อมต่อกันเป็นวงจรขนาดใหญ่ขึ้น ผลจำลองการทำงานอาจไม่ถูกต้องตามพฤติกรรมวงจรที่ได้ออกแบบไว้ได้ เนื่องจากวงจรขนาดใหญ่จะถูกทำการสังเคราะห์และอิมพลิเมนท์วงจรใหม่ทั้งหมด กล่าวคือสังเคราะห์และอิมพลิเมนท์วงจรย่อยใหม่ทั้งหมดด้วย ซึ่งอาจทำให้วงจรย่อยมีรูปแบบการจัดวางเกตและเชื่อมสายบนอุปกรณ์เอฟพีจีเอเปลี่ยนไปจากเดิม ส่งผลกระทบถึงค่าเวลาล่าช้าในเกตและสาย ทำให้วงจรย่อยทำงานผิดพลาดได้ ดังนั้นจึงต้องแยกโครงสร้างที่ได้จากการสังเคราะห์และอิมพลิเมนท์วงจรย่อยที่มีความถูกต้องไว้ไม่ให้ถูกแก้ไข (Design Preservation) และนำโครงสร้างวงจรย่อยเหล่านั้นมาเชื่อมต่อกันเป็นวงจรขนาดใหญ่ภายหลัง

วิธีการแยกโครงสร้างดังกล่าวสามารถใช้คำสั่ง New Partition ในโปรแกรม Xilinx แยกโครงสร้างได้ โดยคลิกขวาที่ไฟล์วงจรย่อยที่ต้องการแยกแล้วเลือก New Partition จะปรากฏสัญลักษณ์  หน้าไฟล์วงจรย่อยที่ต้องการแยกโครงสร้าง และสัญลักษณ์  หน้าไฟล์วงจรขนาดใหญ่ซึ่งประกอบด้วยวงจรย่อยที่ต้องการแยกโครงสร้าง แล้วจึงเข้าสู่ขั้นตอนอิมพลิเมนท์และจำลองการทำงานอิงเวลาอีกครั้ง เพื่อตรวจสอบความถูกต้องของวงจรขนาดใหญ่ภายหลังการใช้คำสั่งแยกโครงสร้าง

6.2.5 การโปรแกรมวงจรลงเอฟพีจีเอด้วยโปรแกรม Xilinx ISE 11.1

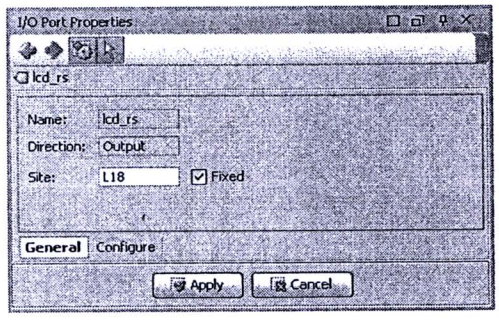
ขั้นตอนการโปรแกรมลงเอฟพีจีเอประกอบด้วย

1. อิมพลิเมนท์ไฟล์ที่ต้องการโปรแกรมลงเอฟพีจีเอแบบ Post-Place & Route ตามขั้นตอนข้อ 6.2.3 และดับเบิ้ลคลิกที่เมนู Generate Program File ในหน้าต่าง Process เพื่อสร้างไฟล์ .bit สำหรับดาวน์โหลดลงเอฟพีจีเอ

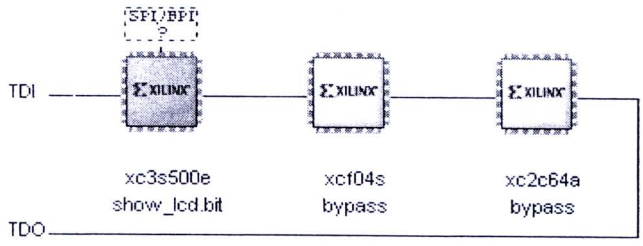
2. ดับเบิ้ลคลิกที่เมนู User Constraints>I/O Planing (PlanAhead) ในหน้าต่าง Process จะเปิดโปรแกรม Plan Ahead 11.1 ขึ้นมา ให้เชื่อมต่อพอร์ทของเอฟพีจีเอเข้ากับ พอร์ทอินพุทหรือเอาท์พุทของวงจรที่ออกแบบไว้ โดยคลิกที่พอร์ทของวงจร และแก้ไขค่าที่หน้าต่าง I/O Properties ในส่วนของ Site ให้เป็นชื่อพอร์ทของเอฟพีจีเอที่ต้องการ เช่น พอร์ทของสวิทช์ พอร์ทของจอแอลซีดี เป็นต้น แล้วกด Apply ดังรูปที่ 6.6 เมื่อกำหนดพอร์ทครบแล้ว ให้กด save แล้วปิดโปรแกรม Plan Ahead

3. คลิกที่ไฟล์ในข้อ 1. แล้วดับเบิ้ลคลิกเมนู Configure Target Device> Manage Configuration Project (iMPACT) ในหน้าต่าง Process ของโปรแกรม Xilinx จะเปิดโปรแกรม ISE iMPACT ขึ้นมา ให้ดับเบิ้ลคลิกที่ Boundary Scan และคลิกขวาที่พื้นที่ว่างสีขาวด้านซ้ายมือเลือก Initialize Chain จะปรากฏอุปกรณ์ขึ้นดังรูปที่ 6.7 ให้กด Yes จากนั้นจะมีหน้าต่าง Assign New Configuration File ขึ้นมา ให้เลือกไฟล์ .bit ที่ได้จากข้อ 1. แล้วกด OK

4. สุดท้ายคลิกขวาที่อุปกรณ์เอฟพีจีเอที่ต้องการโปรแกรมวงจร จากที่ปรากฏดังรูปที่ 6.7 แล้วเลือก Program เพื่อเริ่มการโปรแกรมวงจรลงเอฟพีจีเอ เมื่อโปรแกรมเสร็จสมบูรณ์จะแสดงข้อความว่า Program Succeeded



รูปที่ 6.6 กำหนดพอร์ทเอฟพีจีเอ



รูปที่ 6.7 โปรแกรมลงเอฟพีจีเอ

6.3 การทดลองและผลการทดลอง

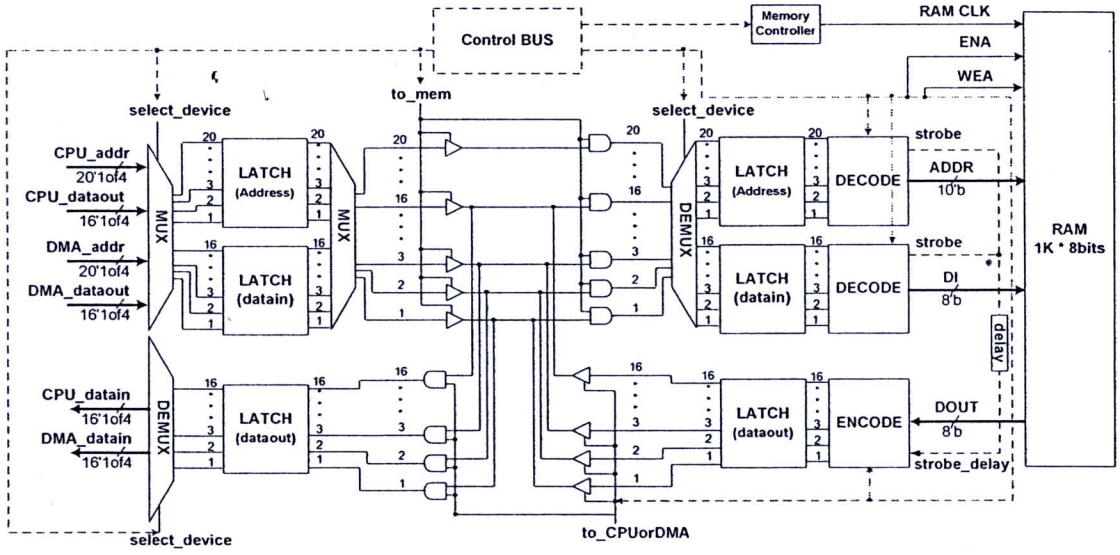
ทำการทดลองวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่ และวงจรคูณเข้ารหัสหนึ่งในสี่ภายในหน่วยดำเนินการทางคณิตศาสตร์และตรรกะ ด้านการใช้พลังงาน ขนาดวงจรที่ใช้ และค่วามเร็วในการทำงาน สุดท้ายจะโปรแกรมวงจรบัสระบบเข้ารหัสหนึ่งในสี่ลงเอฟพีจีเอ เพื่อทดลองการทำงานของบัสระบบร่วมกับองค์ประกอบ รายละเอียดของแต่ละการทดลองและผลการทดลองมีดังต่อไปนี้

6.3.1 การทดลองวัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่

วัดประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่ผ่านผลจำลองการทำงานอิงเวลา โดยเปรียบเทียบกับประสิทธิภาพของบัสเข้ารหัสรางคู่ที่ใช้โครงสร้างเดียวกันกับบัสเข้ารหัสหนึ่งในสี่เวอร์ชันปรับปรุงในบทที่ 3 แต่ปรับปรุงส่วนเข้ารหัส ส่วนถอดรหัส ส่วนพักข้อมูลให้ทำงานกับข้อมูลเข้ารหัสรางคู่ได้ บัสทั้งสองที่นำมาทดลองเชื่อมต่อกับหน่วยความจำหนึ่งตัวดังรูปที่ 6.8 วิธีการทดลองคือให้บัสระบบทั้งสองทำงานในคำสั่งเดียวกันคือ LD ST IN และOUT ซึ่งเป็นคำสั่งที่เรียกใช้งานบัสระบบทั้งสิ้น จากนั้นนำผลการทดลองการทำงานมาเปรียบเทียบในด้านของพลังงาน ขนาดวงจร และความเร็ว แล้วจึงสรุปผลการทดลอง คำสั่งที่ใช้ในการทดลองประกอบไปด้วยคำสั่งรับส่งข้อมูลผ่านบัสระบบจำนวน 8 คำสั่ง รายละเอียดของคำสั่งมีดังนี้

```
0       ORG     0000H
        LD      A,#FF       ;Reg A = FF
1       ST      A,@3FF      ; M[3FF] = Reg A
        LD      A,#7F       ; Reg A = 7F
```

```
2       ST      A,@1FF      ; M[1FF] = Reg A
3       LD      A,@3FF      ; Reg A = M[3FF]
4       LD      A,@1FF      ; Reg A = M[1FF]
5       OUT     4,@03F      ; M[3F] = I/O[4]  (I/O[4] = 3F)
6       OUT     3,@01F      ; M[1F] = I/O[3]  (I/O[3] = 1F)
7       IN      1,@03F      ; I/O[1] = M[3F]
8       IN      2,@01F      ; I/O[2] = M[1F]
        END
```



รูปที่ 6.8 บัสระบบเชื่อมต่อกับหน่วยความจำหนึ่งตัวที่ใช้ทดลอง

- **เปรียบเทียบการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

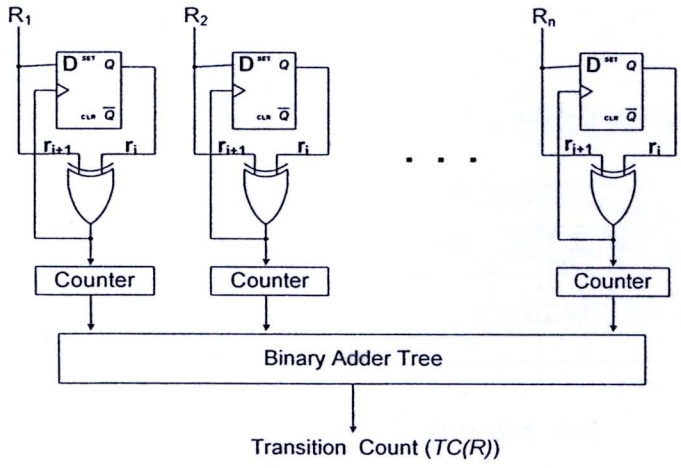
การเปรียบเทียบการเปลี่ยนสถานะสัญญาณ ทำได้โดยนับจำนวนการเปลี่ยนสถานะของสัญญาณ (Transition Count) จากสถานะ 0 เป็นสถานะ 1 หรือจากสถานะ 1 เป็นสถานะ 0 ซึ่งสามารถคำนวณได้จากสมการที่ 6.1 โดยให้ $TC(R)$ คือ จำนวนการเปลี่ยนสถานะสัญญาณของวงจร R ที่ประกอบด้วยสายสัญญาณ R_1 ไปจนถึง R_n โดย n คือจำนวนของ

สายสัญญาณ r_i คือสัญญาณก่อนหน้าของสายสัญญาณ R_n และ r_{i+1} คือสัญญาณถัดไปของสายสัญญาณ R_n

$$TC(R) = \sum_{i=1}^{n}(r_i \oplus r_{i+1}) \qquad (6.1)$$

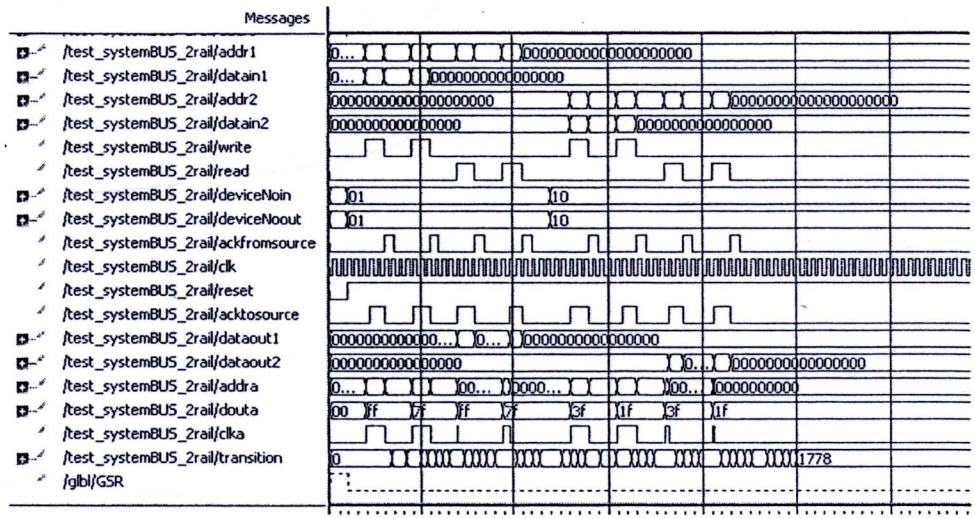
หากพิจารณาจากสายสัญญาณหนึ่งสาย จะพบว่าถ้าสัญญาณก่อนหน้าและสัญญาณถัดไปมีสถานะไม่เหมือนกัน ซึ่งหมายถึงมีการเปลี่ยนสถานะของสัญญาณในสายสัญญาณ จำนวนค่าการเปลี่ยนสถานะของสัญญาณหนึ่งสายจะถูกนับเพิ่มขึ้น 1 ค่า และเมื่อรวมจำนวนการเปลี่ยนสถานะสัญญาณของทุกสายสัญญาณในวงจร จะได้จำนวนการเปลี่ยนสถานะสัญญาณทั้งหมดของวงจร

วงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณ (Transition Count Circuit) [20] ประกอบด้วย ดีฟลิปฟล็อป (D-Flip Flop) เอ็กคลูซีฟออร์เกต (Exclusive-Or Gate) วงจรนับ (Counter) วงจรบวกแบบแผนภาพต้นไม้ (Binary Adder Tree) มีโครงสร้างดังรูปที่ 6.9



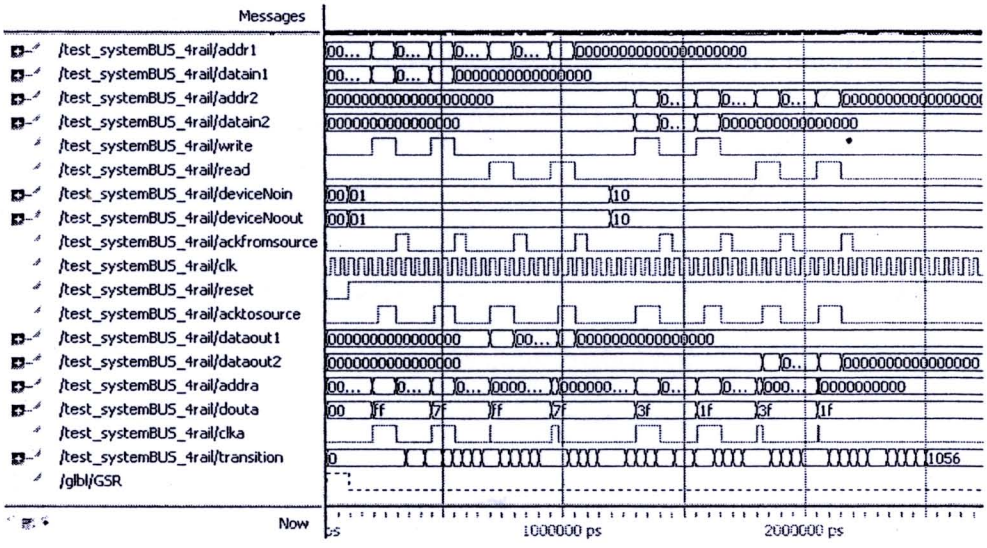
รูปที่ 6.9 วงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณ

เมื่อนำสายสัญญาณในบัสระบบทั้งหมดต่อกับวงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณ จะได้เอาท์พุทเป็นจำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบ ผลการจำลองการทำงานแบบอิงเวลาของการทดลองนี้กับ 8 คำสั่งที่ได้กล่าวไว้ข้างต้นและจำนวนการเปลี่ยนสถานะสัญญาณจากหน้าต่าง wave ของโปรแกรม ModelSim แสดงดังรูปที่ 6.10



(ก) ผลการจำลองการทำงานแบบอิงเวลาและ

จำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่



(ข) ผลการจำลองการทำงานแบบอิงเวลาและ

จำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสหนึ่งในสี่

รูปที่ 6.10 ผลการจำลองการทำงานแบบอิงเวลาของบัสระบบและจำนวนการเปลี่ยน

สถานะสัญญาณของบัสระบบ

ตารางที่ 6.1 สรุปจำนวนการเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ของการทำงานทั้ง 8 คำสั่ง โดยบัสระบบเข้ารหัสรางคู่มีการเปลี่ยนสถานะสัญญาณรวม 1,778 ครั้ง และบัสระบบเข้ารหัสหนึ่งในสี่มีการเปลี่ยนสถานะสัญญาณรวม

1,056 ครั้ง โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดการเปลี่ยนสถานะของสัญญาณลงได้ 40.61% โดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.1 การเปลี่ยนสถานะสัญญาณของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

Bus	Transition count		
	Control	Address/Data	Total
Dual-rail Bus	334	1444	1778
1-of-4 Bus	334	722	1056

- **เปรียบเทียบพลังงานที่ถูกใช้โดยประมาณในการทำงานของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

ประมาณค่าพลังงานที่ถูกใช้ในการทำงาน ของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่บนเอฟพีจีเอXlinx SPARTAN-3E เบอร์ XC3S500EFG320 ผ่านโปรแกรม XPower Analyzer 11 มีขั้นตอนการใช้โปรแกรมดังกล่าวดังนี้

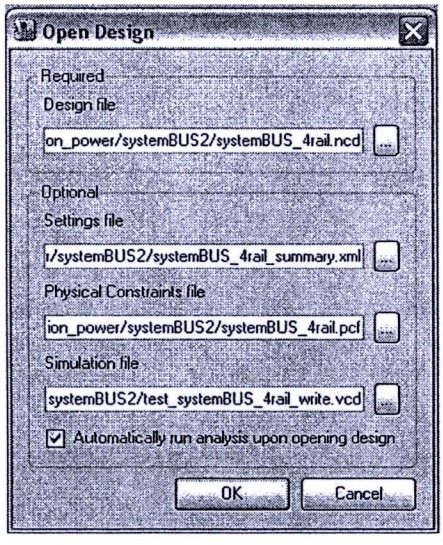
1. แก้ไขไฟล์ที่ใช้ทดลองเพื่อเพิ่มคำสั่งคำนวณพลังงาน โดยเปิดไฟล์ที่ใช้ทดลองวงจรในหัวข้อที่ 6.2.4 ด้วยโปรแกรม Xilinx จากนั้นแก้ไขไฟล์ดังกล่าวโดยเพิ่มบรรทัดโค้ดคำนวณพลังงาน แทรกไว้ระหว่างบรรทัด initial begin กับ // Initialize Inputs ได้ดังนี้

```
initial begin
$dumpfile("ชื่อไฟล์เอาท์พุท.vcd");
$dumpvars(1, ชื่อโมดูลที่ต้องการคำนวณพลังงาน.uut);
// Initialize Inputs
```

2. จำลองการทำงานแบบอิงเวลาโดยมีขั้นตอนเช่นเดียวกับที่อธิบายในหัวข้อที่ 6.2.4 คือคลิกเลือกไฟล์ที่ใช้ทดลองวงจรที่แก้ไขในข้อ 1 ซึ่งแสดงในหน้าต่าง Hierarchy ด้านซ้ายมือ แล้วคลิกที่ ModelSim Simulator ในหน้าต่าง Processes ด้านซ้ายมือ และดับเบิ้ลคลิกที่ Simulate Post-Place & Route Model โปรแกรมจะเริ่มต้นจำลองการทำงานแบบอิงเวลาและเปิดโปรแกรม ModelSim ขึ้นมาโดยอัตโนมัติ เมื่อจำลองการทำงานแบบอิงเวลาเสร็จสิ้น

ในโฟลเดอร์ที่อยู่ของไฟล์ที่ใช้ทดลองจะมีไฟล์นามสกุล .vcd ชื่อเดียวกับที่กำหนดไว้ในโค้ดข้อ 1 เพิ่มขึ้นมา

3. เปิดโปรแกรม Xpower Analyzer เลือกเมนู File>Open Design และเลือกเปิดไฟล์นามสกุล .ncd .xml .pcf และไฟล์นามสกุล .vcd ที่ได้จากข้อ 2 ใส่ลงในช่องของ Design file, Setting file, Physical Constraints file และSimulation file ตามลำดับ เพื่อใช้ในการคำนวณพลังงาน ดังแสดงในรูปที่ 6.11 จากนั้นกด OK โปรแกรมจะเริ่มคำนวณพลังงานจากไฟล์ที่เลือกไว้ เมื่อคำนวณเสร็จสิ้นจะขึ้นข้อความว่า Design 'ชื่อไฟล์.ncd' opened successfully ในหน้าต่าง Progress และสรุปผลการใช้พลังงานแบบย่อในแท็บ Table View นอกจากนี้ยังสามารถดูรายงานแบบสมบูรณ์ (Advance Report) ได้ โดยเลือกที่เมนู Tools>Generate Advance Report โปรแกรมจะสร้างไฟล์รายงานแบบสมบูรณ์นามสกุล .pwr ขึ้นมาภายในโฟลเดอร์ที่อยู่ของไฟล์ที่ใช้ทดลอง



รูปที่ 6.11 ไฟล์ที่ใช้ในการคำนวณพลังงานบนโปรแกรม XPower Analyzer

ผลรายงานแบบสมบูรณ์ที่ได้จากโปรแกรม XPower Analyzer ซึ่งประมาณค่าพลังงานที่ถูกใช้ในการทำงานทั้ง 8 คำสั่งที่ได้กล่าวไว้ข้างต้น ของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่ แสดงดังรูปที่ 6.12

Power summary		I(mA)	P(mW)
Total estimated power consumption			86.63
Total Vccint	1.20V	26.48	31.77
Total Vccaux	2.50V	18.10	45.25
Total Vcco25	2.50V	3.84	9.61
BRAM			0.00
Clocks			0.01
IO			4.97
Logic			0.12
Signals			0.47
Quiescent Vccint	1.20V	25.88	31.05
Quiescent Vccaux	2.50V	18.00	45.00
Quiescent Vcco25	2.50V	2.00	5.00
Package power limits, ambient	25C		2873.56
	250 LFM		3640.78
	500 LFM		3865.98

(ก) รายงานพลังงานที่ถูกใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่

Power summary		I(mA)	P(mW)
Total estimated power consumption			85.98
Total Vccint	1.20V	26.32	31.58
Total Vccaux	2.50V	18.09	45.23
Total Vcco25	2.50V	3.67	9.17
BRAM			0.00
Clocks			0.01
IO			4.48
Logic			0.08
Signals			0.37
Quiescent Vccint	1.20V	25.87	31.04
Quiescent Vccaux	2.50V	18.00	45.00
Quiescent Vcco25	2.50V	2.00	5.00
Package power limits, ambient	25C		2873.56
	250 LFM		3640.78
	500 LFM		3865.98
	750 LFM		4032.26

(ข) รายงานพลังงานที่ถูกใช้ในการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่

รูปที่ 6.12 ผลรายงานของโปรแกรม XPower Analyzer

จากผลรายงานของโปรแกรม XPower Analyzer ในรูปที่ 6.12(ก) คำนวณพลังงานโดยประมาณที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่ได้ดังนี้

คำนวณพลังงานสแตติก :

$$P_{Static} = (VI)_{Quiescent\ Vccint} + (VI)_{Quiescent\ Vccaux} + (VI)_{Quiescent\ Vcco25}$$

$$P_{Static} = (1.20 \times 25.88) + (2.50 \times 18.00) + (2.50 \times 2.00)$$

$$P_{Static} = 31.05 + 45.00 + 5.00$$

$$P_{Static} = 81.05\ mW$$

คำนวณพลังงานไดนามิก :

$P_{Dynamic} = P_{BRAM} + P_{Clocks} + P_{IO} + P_{Logic} + P_{Signals}$

$P_{Dynamic} = 0.00 + 0.01 + 4.97 + 0.12 + 0.47$

$P_{Dynamic} \approx 5.58$ mW

คำนวณพลังงานรวม :

$P_{Total} = P_{Static} + P_{Dynamic}$

$P_{Total} = 81.05 + 5.58$

$P_{Total} = 86.63$ mW

จากผลรายงานของโปรแกรม XPower Analyzer ในรูปที่ 6.12(ข) คำนวณพลังงานโดยประมาณที่ใช้ในการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ได้ดังนี้

คำนวณพลังงานสแตติก :

$P_{Static} = (VI)_{Quiescent\ Vccint} + (VI)_{Quiescent\ Vccaux} + (VI)_{Quiescent\ Vcco25}$

$P_{Static} = (1.20 \times 25.87) + (2.50 \times 18.00) + (2.50 \times 2.00)$

$P_{Static} = 31.04 + 45.00 + 5.00$

$P_{Static} = 81.04$ mW

คำนวณพลังงานไดนามิก :

$P_{Dynamic} = P_{BRAM} + P_{Clocks} + P_{IO} + P_{Logic} + P_{Signals}$

$P_{Dynamic} = 0.00 + 0.01 + 4.48 + 0.08 + 0.37$

$P_{Dynamic} = 4.94$ mW

คำนวณพลังงานรวม :

$P_{Total} = P_{Static} + P_{Dynamic}$

$P_{Total} = 81.04 + 4.94$

$P_{Total} = 85.98$ mW

จากการคำนวณการใช้พลังงานดังแสดงข้างต้น สรุปพลังงานโดยประมาณที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ได้ดังตารางที่ 6.2 โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดการใช้พลังงานลงได้ 0.75% ต่อวินาทีโดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.2 พลังงานที่ถูกใช้โดยประมาณของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

Bus	Estimated Power Consumption (mW)		
	Static Power	Dynamic Power	Total Power
Dual-rail Bus	81.05	5.58	86.63
1-of-4 Bus	81.04	4.94	85.98

- **เปรียบเทียบขนาดวงจรของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

เปรียบเทียบขนาดวงจรของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่จากผลประมาณค่าการใช้อุปกรณ์บนเอฟพีจีเอ (Device Utilization Summary) ของวงจรในโปรแกรม Xilinx มีขั้นตอนคือ คลิกที่ไฟล์วงจรที่ต้องการประมาณค่าการใช้อุปกรณ์ จากนั้นดับเบิ้ลคลิกที่ Design Summary/Reports ในหน้าต่าง Processes ค่าประมาณของการใช้อุปกรณ์จะแสดงในแท็บ Design Summary บนตาราง Device Utilization Summary (Estimated Values) โดยประกอบไปด้วยจำนวนสไลด์ (Slice) จำนวนตารางค้นหาแบบสี่อินพุท (4 Input LUTs) จำนวนพอร์ทอินพุทเอาท์พุท (Bonded IOBs) ที่ถูกใช้ไปบนเอฟพีจีเอ

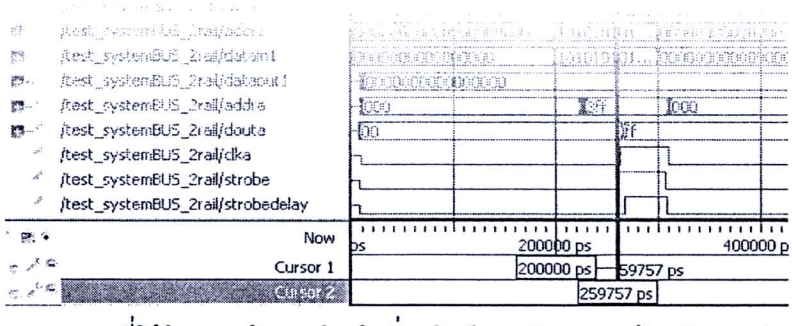
ตารางที่ 6.3 แสดงค่าการใช้อุปกรณ์ของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ โดยบัสระบบเข้ารหัสรางคู่ใช้สไลด์จำนวน 221 ตัว ตารางค้นหาแบบสี่อินพุทจำนวน 388 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 142 ตัว บัสระบบเข้ารหัสหนึ่งในสี่ใช้สไลด์จำนวน 203 ตัว ตารางค้นหาแบบสี่อินพุทจำนวน 359 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุท รวม 142 ตัว โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดขนาดวงจรลงได้ 7.72% โดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.3 การใช้อุปกรณ์ของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

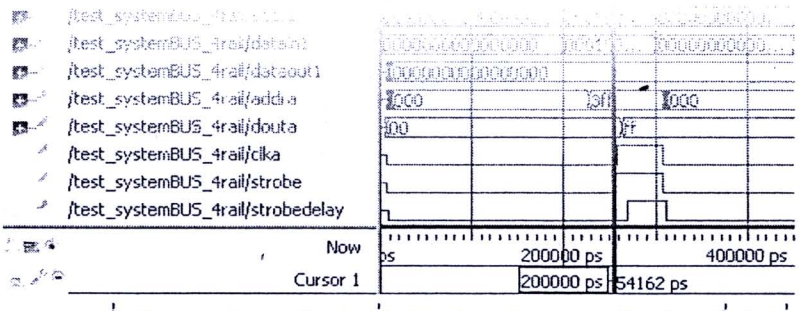
Bus	Device Utilization Summary		
	Slices	4 input LUTs	bonded IOBs
Dual-rail Bus	221	388	142
1-of-4 Bus	203	359	142

- **เปรียบเทียบความเร็วของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่**

เปรียบเทียบความเร็วของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่ โดยวัดความเร็วในการทำงานของบัสทั้งสองซึ่งแสดงบนผลการจำลองการทำงานแบบอิงเวลาที่หน้าต่าง wave ของโปรแกรม ModelSim เวลาที่ใช้ในการทำงานในคำสั่ง ST A,@3FF แบบไม่รวมเวลาอ้างอิงหน่วยความจำ ของบัสระบบเข้ารหัสรางคู่กับบัสระบบเข้ารหัสหนึ่งในสี่ แสดงดังรูปที่ 6.13



(ก) เวลาที่ใช้ในการทำงานในคำสั่งสโตร์ของบัสระบบเข้ารหัสรางคู่



(ข) เวลาที่ใช้ในการทำงานในคำสั่งสโตร์ของบัสระบบเข้ารหัสหนึ่งในสี่

รูปที่ 6.13 ผลการจำลองการทำงานแบบอิงเวลาและเวลาที่ใช้ในการทำงานในคำสั่งสโตร์ของบัสระบบ

เวลาที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ของการทำงานทั้ง 8 คำสั่งแบบไม่รวมเวลาอ้างอิงหน่วยความจำ ในหน่วยนาโนวินาที (ns) แสดงดังตารางที่ 6.4 โดยบัสระบบเข้ารหัสรางคู่ทำงานในคำสั่ง ST ใช้เวลาโดยเฉลี่ย 60 นาโนวินาที คำสั่ง LD ใช้เวลาโดยเฉลี่ย 52 นาโนวินาที คำสั่ง OUT ใช้เวลาโดยเฉลี่ย 60 นาโนวินาที คำสั่ง IN ใช้เวลาโดยเฉลี่ย 50 นาโนวินาที ในขณะที่บัสระบบเข้ารหัสหนึ่งในสี่ทำงานในคำสั่ง ST ใช้เวลาโดยเฉลี่ย 53 นาโนวินาที คำสั่ง LD ใช้เวลาโดยเฉลี่ย 50 นาโนวินาที คำสั่ง OUT ใช้เวลาโดยเฉลี่ย 53 นาโนวินาที คำสั่ง IN ใช้เวลาโดยเฉลี่ย 48 นาโนวินาที โดยเมื่อพิจารณาจะพบว่า บัสระบบเข้ารหัสหนึ่งในสี่ลดเวลาในการทำงานลงได้ 7.42% โดยประมาณ เมื่อเปรียบเทียบกับบัสระบบเข้ารหัสรางคู่

ตารางที่ 6.4 เวลาที่ใช้ในการทำงานของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

Instructions	Estimated Read/Write Cycle Time (ns)	
	Dual-rail Bus	1-of-4 Bus
ST A,@3FF	59.76	54.17
ST A,@1FF	59.20	52.71
LD A,@3FF	52.14	50.53
LD A,@1FF	50.96	49.08
OUT 4,@3F	59.18	53.10
OUT 3,@1F	61.00	53.30
IN 1,@3F	49.34	48.50
IN 2,@1F	51.17	48.50

- **สรุปผลการทดลองบัสระบบเข้ารหัสหนึ่งในสี่**

บัสระบบเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีกว่าบัสระบบเข้ารหัสรางคู่ กล่าวคือ บัสระบบเข้ารหัสหนึ่งในสี่มีจำนวนการเปลี่ยนสถานะของสัญญาณต่ำกว่า จึงใช้พลังงานในการเปลี่ยนสถานะของสัญญาณต่ำกว่า นอกจากนี้ยังใช้พลังงานรวมในการทำงานต่ำกว่า มีขนาดวงจรเล็กกว่า และใช้เวลาในการทำงานน้อยกว่าบัสระบบเข้ารหัสรางคู่ บนโครงสร้าง

เดียวกัน สรุปประสิทธิภาพได้ดังตารางที่ 6.5 จำนวนสัญลักษณ์ ✓ แสดงถึงประสิทธิภาพโดยสัญลักษณ์ ✓✓ หมายถึงประสิทธิภาพดีที่สุด

ตารางที่ 6.5 ประสิทธิภาพของบัสเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่

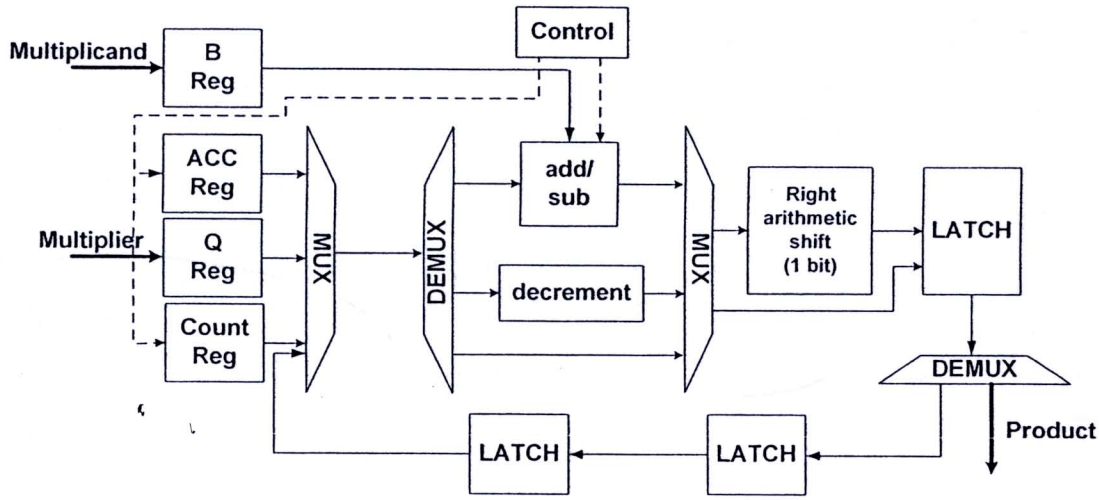
Efficiency	Bus	
	Dual-rail Bus	1-of-4 Bus
Transition Count	✓	✓✓
Power Consumption	✓	✓✓
Circuit Size	✓	✓✓
Read/Write Cycle Time	✓	✓✓

6.3.2 การทดลองวัดประสิทธิภาพของหน่วยคำนวณทางคณิตศาสตร์และตรรกะเข้ารหัสหนึ่งในสี่

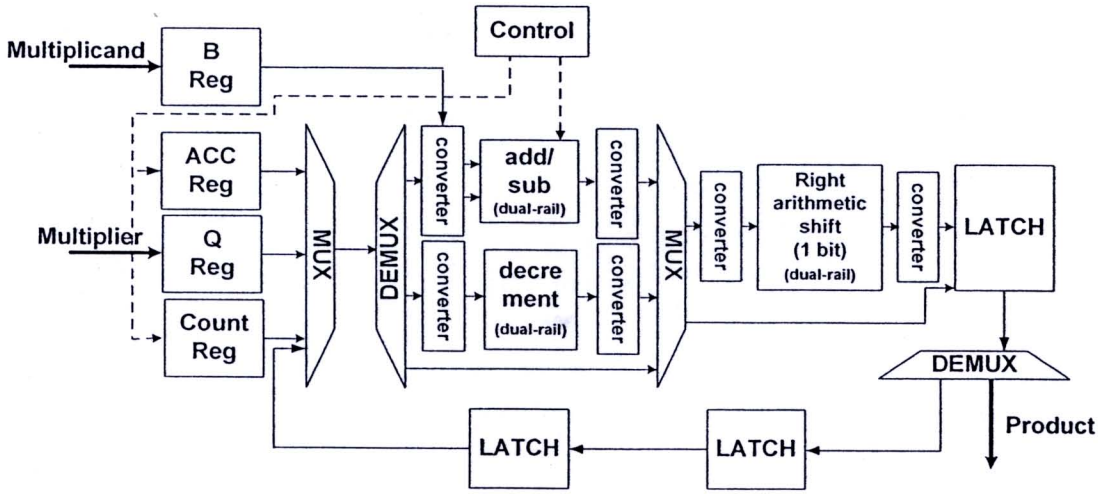
วัดประสิทธิภาพของหน่วยคำนวณทางคณิตศาสตร์และตรรกะผ่านการทดลองการทำงานของวงจรคูณบูทอัลกอลิทึมเข้ารหัสหนึ่งในสี่ ซึ่งประกอบด้วยส่วนคำนวณคือ วงจรบวก วงจรลบ วงจรลดค่า และส่วนตรรกะคือ วงจรเลื่อนบิต โดยวัดประสิทธิภาพการทำงานในด้านของการใช้พลังงาน ขนาดวงจร และความเร็ว แล้วจึงสรุปผลการทดลอง คำสั่งที่ใช้ในการทดลองมีทั้งหมด 4 คำสั่งคือ คูณค่า -34 ด้วยค่า -34 คูณค่า 125 ด้วยค่า 0 คูณค่า 15 ด้วยค่า -1 และคูณค่า 19 ด้วยค่า 30

จากการออกแบบส่วนฟังก์ชัน (Function Unit) หรือส่วนหน่วยคำนวณทางคณิตศาสตร์และตรรกะในบทที่ 5 จะพบว่าวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ที่สร้างจากแผนภาพตัดสินใจแบบทวิภาคชนิดมีการลดทอนอันดับ มีขนาดใหญ่กว่าวงจรฟังก์ชันเข้ารหัสรางคู่ ในขณะที่ประสิทธิภาพของบัสระบบเข้ารหัสหนึ่งในสี่สูงกว่าบัสระบบเข้ารหัสรางคู่ในทุกด้านดังที่อธิบายในหัวข้อที่ 6.3.4 การออกแบบวงจรแบบผสมจึงมีความน่าสนใจเนื่องจากใช้ข้อได้เปรียบของรหัสหนึ่งในสี่และรหัสรางคู่มาออกแบบ กล่าวคือ ออกแบบส่วนรับส่งข้อมูล (Data Path) ให้เข้ารหัสหนึ่งในสี่ และออกแบบส่วนฟังก์ชันให้เข้ารหัสรางคู่ การทดลองในหัวข้อนี้จึงนำวงจรแบบผสมมาร่วมทดลองด้วย โดยปรับปรุงวงจรคูณบูทอัลกอลิทึมเข้ารหัสหนึ่งในสี่ ซึ่งเดิมใช้

ฟังก์ชันเข้ารหัสหนึ่งในสี่ ให้ใช้ฟังก์ชันเข้ารหัสรางคู่แทน วงจรคูณบูทอัลกอลิทึมที่นำมาทดลองประสิทธิภาพของหน่วยคำนวณทางคณิตศาสตร์และตรรกะมีดังนี้

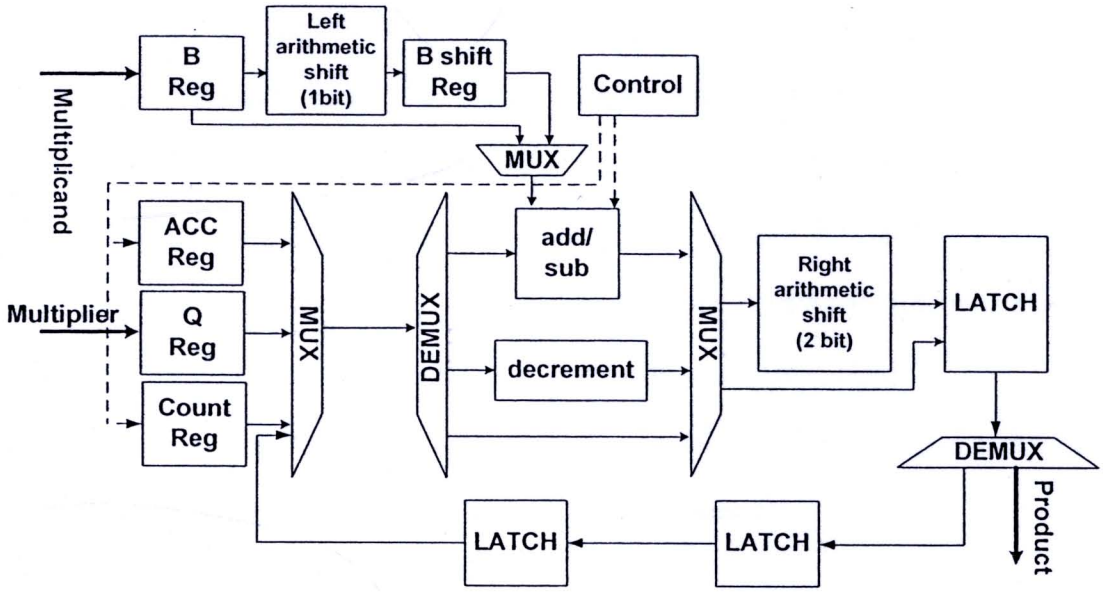


(ก) วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่

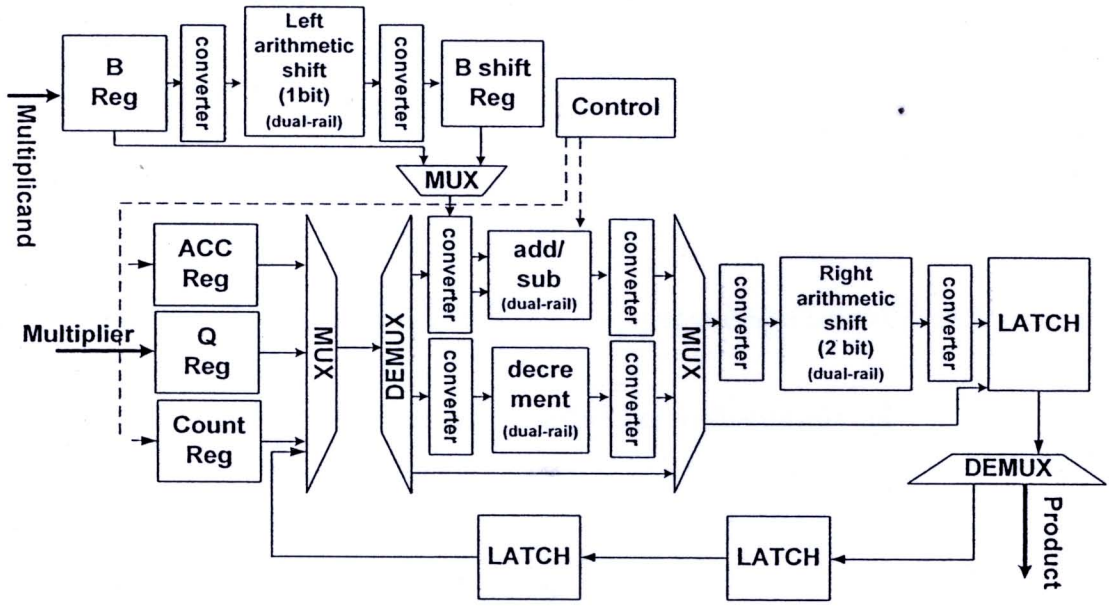


(ข) วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

รูปที่ 6.14 วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่และใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่



(ก) วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่



(ข) วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

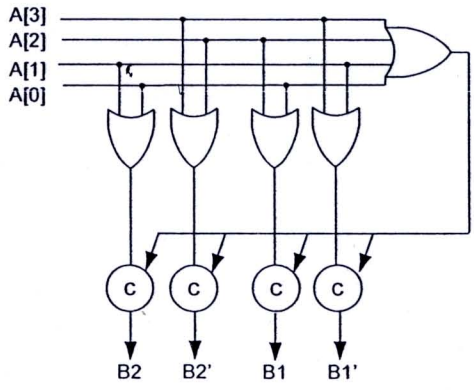
รูปที่ 6.15 วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่
และใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่

○ วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ (1-of-4 Radix-2 Booth Multiplier with 1-of-4 Function Unit) ดังรูปที่ 6.14(ก)

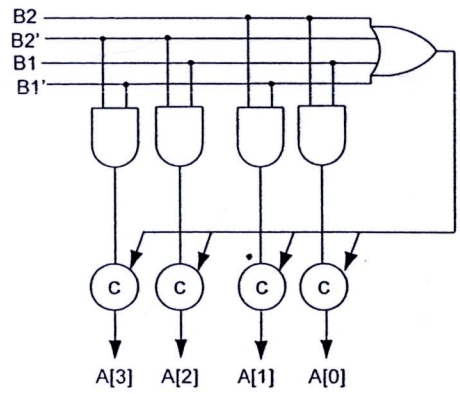
○ วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ (1-of-4 Radix-2 Booth Multiplier with Dual-rail Function Unit) ดังรูปที่ 6.14(ข)

○ วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ (1-of-4 Radix-4 Booth Multiplier with 1-of-4 Function Unit) ดังรูปที่ 6.15(ก)

○ วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ (1-of-4 Radix-4 Booth Multiplier with Dual-rail Function Unit) ดังรูปที่ 6.15(ข)



(ก) วงจรแปลงค่ารหัสรางคู่เป็นรหัสหนึ่งในสี่



(ข) วงจรแปลงค่ารหัสหนึ่งในสี่เป็นรหัสรางคู่

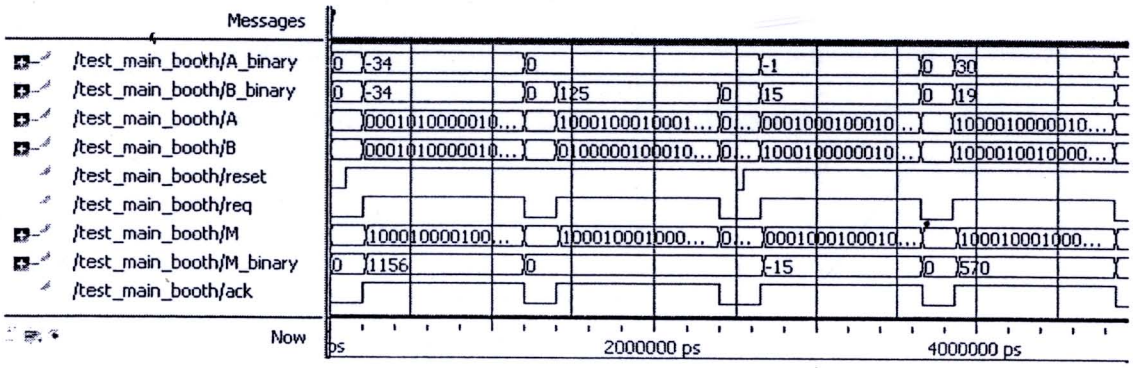
รูปที่ 6.16 วงจรแปลงค่าระหว่างรหัสรางคู่กับรหัสหนึ่งในสี่

ตารางที่ 6.6 รหัสรางคู่และรหัสหนึ่งในสี่

1-of-4 Code				Dual-rail Code			
A[3]	A[2]	A[1]	A[0]	B2	B2'	B1	B1'
1	0	0	0	0	1	0	1
0	1	0	0	0	1	1	0
0	0	1	0	1	0	0	1
0	0	0	1	1	0	1	0

วงจรแบบผสมมีส่วนรับส่งข้อมูลเข้ารหัสหนึ่งในสี่ และส่วนฟังก์ชันเข้ารหัสรางคู่ ดังนั้นจึงต้องใช้วงจรแปลงค่าดังรูปที่ 6.16 แปลงค่าระหว่างรหัสรางคู่กับรหัสหนึ่งในสี่ในตารางที่ 6.6 เพื่อให้ส่วนรับส่งข้อมูลและส่วนฟังก์ชันสามารถทำงานร่วมกันได้

ผลการจำลองการทำงานแบบอิงเวลาของวงจรคูณเข้ารหัสหนึ่งในสี่กับ 4 คำสั่งที่ได้กล่าวไว้ข้างต้น แสดงดังรูปที่ 6.17 ซึ่งผลการทำงานมีความถูกต้อง โดยมีพอร์ท B เป็นตัวตั้ง (Multiplicand) พอร์ท A เป็นตัวคูณ (Multiplier) และพอร์ท M เป็นผลลัพธ์ (Product) ซึ่งเข้ารหัสหนึ่งในสี่ทั้งหมด ส่วนพอร์ท B_binary A_binary และ M_binary จะแสดงค่าเลขฐานสิบของพอร์ท B A และ M ตามลำดับ เพื่อสะดวกต่อการอ่านค่าและตรวจสอบความถูกต้อง



รูปที่ 6.17 ผลการจำลองการทำงานแบบอิงเวลาของวงจรคูณเข้ารหัสหนึ่งในสี่

- **เปรียบเทียบการเปลี่ยนสถานะสัญญาณของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่**

เปรียบเทียบการเปลี่ยนสถานะสัญญาณของวงจรคูณ โดยใช้วงจรนับจำนวนการเปลี่ยนสถานะของสัญญาณในการทดลอง เช่นเดียวกันกับการทดลองบัสระบบ สรุปจำนวนการเปลี่ยนสถานะสัญญาณของวงจรคูณทั้ง 4 แบบกับการทำงานทั้ง 4 คำสั่ง ได้ดังตารางที่ 6.7 โดยวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่มีการเปลี่ยนสถานะสัญญาณรวม 5,444 ครั้ง วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่มีการเปลี่ยนสถานะสัญญาณรวม 8,004 ครั้ง วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่มีการเปลี่ยนสถานะสัญญาณรวม 2,990 ครั้ง วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่มีการเปลี่ยนสถานะสัญญาณรวม 4,534 ครั้ง โดยเมื่อ

พิจารณาจะพบว่า วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดการเปลี่ยนสถานะของสัญญาณลงได้ 31.98% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดการเปลี่ยนสถานะของสัญญาณลงได้ 34.05% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

ตารางที่ 6.7 การเปลี่ยนสถานะสัญญาณของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Multiplicand x Multiplier	Transition count			
	1-of-4 Radix-2 Booth Multiplier		1-of-4 Radix-4 Booth Multiplier	
	with dual-rail function unit	with 1-of-4 function unit	with dual-rail function unit	with 1-of-4 function unit
-34 x -34	2,076	1,388	1,194	770
125 x 0	1,926	1,334	1,052	716
15 x -1	1,976	1,352	1,144	752
19 x 30	2,026	1,370	1,144	752
total	8,004	5,444	4,534	2,990

- **เปรียบเทียบขนาดวงจรของของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่**

เปรียบเทียบขนาดวงจรของวงจรคูณ จากผลประมาณค่าการใช้อุปกรณ์เช่นเดียวกันกับการทดลองบัสระบบ ค่าการใช้อุปกรณ์ของบัสระบบเข้ารหัสรางคู่และบัสระบบเข้ารหัสหนึ่งในสี่ แสดงดังตารางที่ 6.8 โดยวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ใช้สไลด์จำนวน 478 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 791 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 107 ตัว วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ใช้สไลด์จำนวน 589 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 898 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 107 ตัว วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัส

หนึ่งในสี่ใช้สไลด์จำนวน 443 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 709 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 75 ตัว วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ใช้สไลด์จำนวน 671 ตัว ตารางค้นหาแบบสี่อินพุท จำนวน 1,007 ตัว และมีจำนวนพอร์ทอินพุทเอาท์พุทรวม 75 ตัว โดยเมื่อพิจารณาจะพบว่า วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดขนาดวงจรลงได้เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ โดยลดจำนวนการใช้สไลด์ลง 18.85% ลดจำนวนการใช้ตารางค้นหาแบบสี่อินพุท 11.92% และใช้จำนวนพอร์ทอินพุทเอาท์พุทเท่ากัน และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดขนาดวงจรลงได้เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ โดยลดจำนวนการใช้สไลด์ลง 33.98% ลดจำนวนการใช้ตารางค้นหาแบบสี่อินพุท 29.59% และใช้จำนวนพอร์ทอินพุทเอาท์พุทเท่ากัน

ตารางที่ 6.8 การใช้อุปกรณ์ของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Multiplier (1-of-4 encoding)	Device Utilization Summary					
	with dual-rail function unit			with 1-of-4 function unit		
	4 input LUTs	Slices	bonded IOBs	4 input LUTs	Slices	bonded IOBs
Radix-2 Booth Multiplier	898	589	107	791	478	107
Radix-4 Booth Multiplier	1,007	671	75	709	443	75

- **เปรียบเทียบความเร็วของวงจรของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่**

เปรียบเทียบความเร็วของวงจรคูณ โดยวัดความเร็วในการทำงานของวงจรคูณทั้งสี่แบบ ซึ่งแสดงบนผลการจำลองการทำงานแบบอิงเวลาเช่นเดียวกันกับการทดลองบัสระบบ เวลาที่ใช้ในการทำงานของวงจรคูณกับการทำงานทั้ง 4 คำสั่งในหน่วยนาโนวินาที (ns) แสดงดังตารางที่ 6.9 โดยวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ใช้เวลารวม

ประมาณ 742 นาโนวินาที วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ใช้เวลารวมประมาณ 668 นาโนวินาที วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ใช้เวลารวมประมาณ 397 นาโนวินาที และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ ใช้เวลารวมประมาณ 457 นาโนวินาที โดยเมื่อพิจารณาจะพบว่า วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ใช้เวลาในการทำงานเพิ่มขึ้น 10.96% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ และวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ ลดเวลาในการทำงานลงได้ 13.24% โดยประมาณ เมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่

ตารางที่ 6.9 เวลาที่ใช้ในการทำงานของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Multiplicand x Multiplier	Estimated computation time (ns)			
	1-of-4 Radix-2 Booth Multiplier		1-of-4 Radix-4 Booth Multiplier	
	with dual-rail function unit	with 1-of-4 function unit	with dual-rail function unit	with 1-of-4 function unit
-34 x -34	167.69	187.13	127.75	113.28
125 x 0	162.47	181.46	100.63	83.70
15 x -1	169.49	186.00	110.40	94.00
19 x 30	168.78	187.09	118.69	105.91
total	668.43	741.68	457.47	396.89

- **สรุปผลการทดลองของวงจรคูณเข้ารหัสหนึ่งในสี่**

วงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีที่สุดเมื่อเปรียบเทียบกับวงจรคูณครั้งละ 2 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ วงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่งในสี่ใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ และวงจรคูณครั้งละ 1 หลักเข้ารหัสหนึ่ง

ในสี่ใช้ฟังก์ชันเข้ารหัสรางคู่ กล่าวคือ ใช้พลังงานในการเปลี่ยนสถานะของสัญญาณต่ำที่สุด มีขนาดวงจรเล็กที่สุด และใช้เวลาในการทำงานน้อยที่สุด บนโครงสร้างเดียวกัน สรุปประสิทธิภาพได้ดังตารางที่ 6.10 จำนวนสัญลักษณ์ ✓ แสดงถึงประสิทธิภาพ โดยสัญลักษณ์ ✓✓✓✓ หมายถึงประสิทธิภาพดีที่สุด

ตารางที่ 6.10 ประสิทธิภาพของวงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้วงจรฟังก์ชันเข้ารหัสรางคู่กับใช้วงจรฟังก์ชันเข้ารหัสหนึ่งในสี่

Efficiency	Multiplier			
	1-of-4 Radix-2 Booth Multiplier		1-of-4 Radix-4 Booth Multiplier	
	with dual-rail function unit	with 1-of-4 function unit	with dual-rail function unit	with 1-of-4 function unit
Transition Count (Dynamic Power)	✓	✓✓	✓✓✓	✓✓✓✓
Circuit Size (Area)	✓✓	✓✓✓	✓	✓✓✓✓
Computation Time (Time)	✓✓	✓	✓✓✓	✓✓✓✓

ผลการทดลองนี้แสดงให้เห็นว่า วงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้ฟังก์ชันเข้ารหัสหนึ่งในสี่ซึ่งเป็นวงจรเข้ารหัสหนึ่งในสี่ล้วน จะมีประสิทธิภาพสูงขึ้นทั้งทางด้านพลังงานที่ใช้ในการเปลี่ยนสถานะของสัญญาณ ขนาดวงจร และความเร็ว หากเป็นการคำนวณครั้งละ 2 หลัก แต่วงจรคูณเข้ารหัสหนึ่งในสี่แบบใช้ฟังก์ชันเข้ารหัสรางคู่ซึ่งเป็นวงจรผสม จะมีประสิทธิภาพทางด้านความเร็วดีกว่าวงจรเข้ารหัสหนึ่งในสี่ล้วน หากเป็นฟังก์ชันคำนวณทีละ 1 หลัก เนื่องจากวงจรฟังก์ชันที่มีการคำนวณทีละหนึ่งหลักของรหัสหนึ่งในสี่ มีความซับซ้อนกว่าของรหัสรางคู่ อย่างไรก็ตาม วงจรผสมเป็นการออกแบบที่ไม่คุ้มค่า เนื่องจากต้องเพิ่มวงจรแปลงข้อมูลระหว่างรหัสหนึ่งในสี่และรหัสรางคู่ในระบบ ทำให้สิ้นเปลืองอุปกรณ์มากขึ้น ส่งผลให้วงจรมีขนาดใหญ่กว่าการออกแบบวงจรด้วยการเข้ารหัสหนึ่งในสี่ล้วน อีกทั้งยังใช้พลังงานในการเปลี่ยนสถานะสัญญาณสูงกว่า

เมื่อพิจารณาประสิทธิภาพของรหัสหนึ่งในสี่เทียบกับการเข้ารหัสแบบอื่น จะสรุปได้ดังตารางที่ 6.11 โดยให้ n เป็นจำนวนบิตของรหัสฐานสอง ซึ่งมีค่าเป็นจำนวนนับ เมื่อพิจารณาจะพบว่าวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ ซึ่งใช้สายสัญญาณ 4 เส้นในการเข้ารหัสฐานสองจำนวน 2 บิต จึงมีประสิทธิภาพดีเมื่อใช้กับงานที่มีการคำนวณเป็นจำนวนคู่ เช่น คำนวณครั้งละ 2 บิต คำนวณครั้งละ 4 บิต คำนวณครั้งละ 8 บิต เป็นต้น โดยจะมีประสิทธิภาพดีที่สุดเมื่อใช้กับงานที่มีการคำนวณครั้งละ 2 บิต

ตารางที่ 6.11 ประสิทธิภาพของรหัสหนึ่งในสี่เทียบกับการเข้ารหัสแบบอื่น

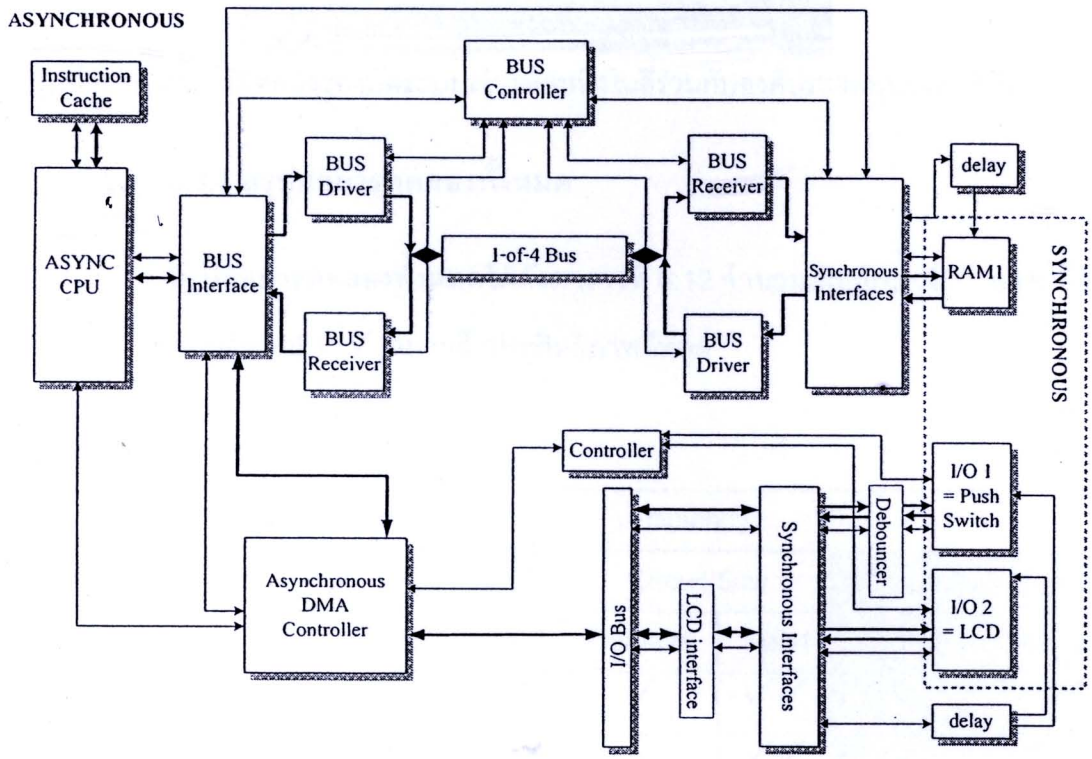
Code ($1\text{-of-}2^n$)	Wires (2^n wires)	Encoding (n bits)	Efficiency (Estimated)			
			Best	Good	Moderate	Poor
1-of-2 (dual-rail)	2	1	1 bit	n bit	-	-
1-of-4	4	2	2 bits	2n bits	2n+1 bits	<2 bits
1-of-8	8	3	3 bits	3n bits	3n+1 bits 3n+2 bits	<3 bits
1-of-16	16	4	4 bits	4n bits	4n+1 bits 4n+2 bits 4n+3 bits	<4 bits
1-of-32	32	5	5 bits	5n bits	5n+1 bits 5n+2 bits 5n+3 bits ⋮ 5n+4 bits	<5 bits
1-of-64	64	6	6 bits	6n bits	6n+1 bits 6n+2 bits 6n+3 bits ⋮ 6n+5 bits	<6 bits

6.3.3 โปรแกรมวงจรบัสระบบเข้ารหัสหนึ่งในสี่ลงเอฟพีจีเอ

ทดลองการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ร่วมกับองค์ประกอบ คือ ไมโครโพรเซสเซอร์ ดีเอ็มเอ และอุปกรณ์ต่อพ่วงคือ สวิทช์แบบกดติดปล่อยดับ และแอลซีดีบนเอฟพีจีเอ ด้วยการโปรแกรมวงจรลงเอฟพีจีเอ Xlinx SPARTAN-3E เบอร์ XC3S500EFG320 โดยเชื่อมต่อสวิทช์แบบกดติดปล่อยดับไว้กับพอร์ทอุปกรณ์ต่อพ่วงที่ 1 ของดีเอ็มเอ (I/O[1]) เชื่อมต่อแอลซีดีไว้กับพอร์ทอุปกรณ์ต่อพ่วงที่ 2 ของดีเอ็มเอ (I/O[2]) แล้วสร้างส่วนติดต่อกับสวิทช์แบบกดติดปล่อยดับคือ ตัวกันผลการเด้ง (Debouncer) และส่วนติดต่อกับแอลซีดี (LCD Interface) ดังรูปที่ 6.18 จากนั้นเขียนโปรแกรมให้เกิดการอินเตอร์รัพท์เมื่อสวิทช์แบบกดติดปล่อยดับถูกกด โดยจะอินเตอร์รัพท์ไปทำโปรแกรมย่อยคือส่งค่าตัวอักษรคำว่า 1-of-4 Bus! ออกจอแอลซีดี รายละเอียดข้องคำสั่งที่ใช้ในการทดลองมีดังนี้

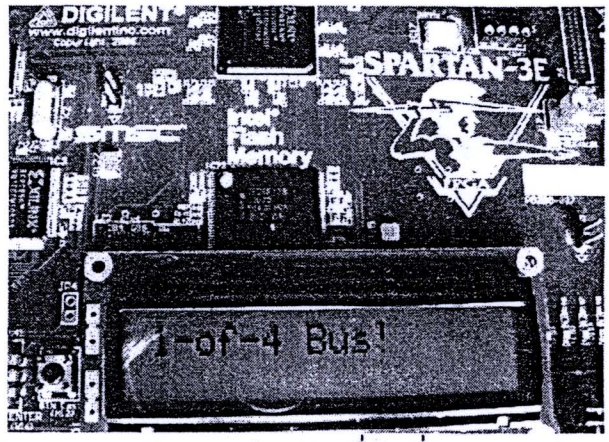
```
0       ORG   0000H
// เก็บค่าตัวอักษรไว้ในหน่วยความจำ
1       LD    A,#31    ; Reg A = 31
2       ST    A,@1     ; M[1] = Reg A
3       LD    A,#2D    ; Reg A = 2D
4       ST    A,@2     ; M[2] = Reg A
5       LD    A,#6F    ; Reg A = 6F
6       ST    A,@3     ; M[3] = Reg A
7       LD    A,#66    ; Reg A = 66
8       ST    A,@4     ; M[4] = Reg A
9       LD    A,#2D    ; Reg A = 2D
10      ST    A,@5     ; M[5] = Reg A
11      LD    A,#34    ; Reg A = 34
12      ST    A,@6     ; M[6] = Reg A
13      LD    A,#20    ; Reg A = 20
14      ST    A,@7     ; M[7] = Reg A
15      LD    A,#42    ; Reg A = 42
16      ST    A,@8     ; M[8] = Reg A
17      LD    A,#75    ; Reg A = 75
18      ST    A,@9     ; M[9] = Reg A
19      LD    A,#73    ; Reg A = 73
20      ST    A,@A     ; M[A] = Reg A
21      LD    A,#21    ; Reg A = 21
22      ST    A,@B     ; M[2] = Reg A
// รับข้อมูลจากสวิทช์กดติดปล่อยดับ
23      OUT   1,@C     ; M[C] = I/O[1]
                         (I/O[1] = 1)
// เกิดการอินเตอร์รัพท์ไปทำโปรแกรมย่อย
คือส่งค่าตัวอักษรจากหน่วยความจำออก
จอ LCD
INT:
24      IN    2,@1     ; I/O[2] = M[1]
```

```
25    IN    2,@2    ; I/O[2] = M[2]      31    IN    2,@8    ; I/O[2] = M[8]
26    IN    2,@3    ; I/O[2] = M[3]      32    IN    2,@9    ; I/O[2] = M[9]
27    IN    2,@4    ; I/O[2] = M[4]      33    IN    2,@A    ; I/O[2] = M[A]
28    IN    2,@5    ; I/O[2] = M[5]      34    IN    2,@B    ; I/O[2] = M[B]
29    IN    2,@6    ; I/O[2] = M[6]      35    END
30    IN    2,@7    ; I/O[2] = M[7]
```



รูปที่ 6.18 บัสระบบเข้ารหัสหนึ่งในสี่ องค์ประกอบ
ส่วนติดต่อกับสวิทซ์แบบกดติดปล่อยดับ และส่วนติดต่อกับแอลซีดี

ผลการทดลองการทำงานของบัสระบบเข้ารหัสหนึ่งในสี่ร่วมกับองค์ประกอบ พบว่าวงจรทำงานได้ถูกต้องบนเอฟพีจีเอ ดังแสดงในรูปที่ 6.19



รูปที่ 6.19 ผลการทดลองของบัสระบบเข้ารหัสหนึ่งในสี่ร่วมกับองค์ประกอบบนเอฟพีจีเอ

6.3.4 สรุปผลการทดลองทั้งหมด

สรุปผลการทดลองทั้งหมดได้ดังตารางที่ 6.12 จำนวนสัญลักษณ์ ✓ แสดงถึงประสิทธิภาพ โดยสัญลักษณ์ ✓✓ หมายถึงประสิทธิภาพดีที่สุด

ตารางที่ 6.12 สรุปผลการทดลองทั้งหมด

Circuits	Efficiency					
	Transition Count		Circuit Size		Computation Time	
	Dual-rail	1-of-4	Dual-rail	1-of-4	Dual-rail	1-of-4
Bus	✓	✓✓	✓	✓✓	✓	✓✓
Function Unit(compute 2-bits per time)	✓	✓✓	✓	✓✓	✓	✓✓
Function Unit(compute 1-bits per time)	✓	✓✓	✓✓	✓	✓✓	✓

จากตารางที่ 6.12 บัสระบบเข้ารหัสหนึ่งในสี่มีประสิทธิภาพดีกว่าบัสระบบเข้ารหัสรางคู่ บนโครงสร้างเดียวกัน สำหรับวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่ จะมีประสิทธิภาพสูงกว่าวงจรเข้ารหัสรางคู่ทั้งทางด้านพลังงาน ขนาดวงจร และความเร็ว หากเป็นการคำนวณครั้งละ 2 หลัก แต่หากเป็นฟังก์ชันคำนวณทีละ 1 หลัก วงจรฟังก์ชันเข้ารหัสรางคู่จะมีประสิทธิภาพทางด้านความเร็วดีกว่าวงจรเข้ารหัสหนึ่งในสี่ เนื่องจากวงจรฟังก์ชันที่มีการคำนวณทีละหนึ่งหลักของรหัส

หนึ่งในสี่ มีความซับซ้อนกว่าของรหัสรางคู่ ดังนั้นวงจรฟังก์ชันเข้ารหัสหนึ่งในสี่จึงมีประสิทธิภาพดีเมื่อใช้กับงานที่มีการคำนวณเป็นจำนวนคู่ เช่น คำนวณครั้งละ 2 บิต คำนวณครั้งละ 4 บิต คำนวณครั้งละ 8 บิต เป็นต้น โดยจะมีประสิทธิภาพดีที่สุดเมื่อใช้กับงานที่มีการคำนวณครั้งละ 2บิต