LFSR Reseeding แบบขนานร่วมกับรีจิสเตอร์แบ่งกลุ่มสำหรับ Mixed-mode BIST

Parallel LFSR Reseeding with Selection Register for Mixed-mode BIST

ปิยนาฏ คงทิม

Piyanart Kongtim

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2553

LFSR Reseeding แบบขนานร่วมกับรีจิสเตอร์แบ่งกลุ่มสำหรับ Mixed-mode BIST

Parallel LFSR Reseeding with Selection Register for Mixed-mode BIST

ปิยนาฏ คงทิม

Piyanart Kongtim

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา

วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Computer Engineering

Prince of Songkla University

2553

ชื่อวิทยานิพนธ์ LFSR Reseeding แบบขนานร่วมกับรีจิสเตอร์แบ่งกลุ่มสำหรับ Mixed-mode BIST

ผู้เขียน นางสาวปิยนาฏ คงทิม

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

…….

(ผู้ช่วยศาสตราจารย์ ดร.ทวีศักดิ์ เรืองพีระกุล)

คณะกรรมการสอบ

…… ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

……..... กรรมการ

(รองศาสตราจารย์ ดร.มิตรชัย จงเชี่ยวชำนาญ)

……..... กรรมการ

(รองศาสตราจารย์ ดร.วัฒนพงศ์ เกิดทองมี)

……..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ทวีศักดิ์ เรืองพีระกุล)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

…...……...…….

(ศาสตราจารย์ ดร.อมรรัตน์ พงศ์ดารา)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์ LFSR Reseeding แบบขนานร่วมกับรีจิสเตอร์แบ่งกลุ่มสำหรับ Mixed-mode BIST
ผู้เขียน นางสาวปิยนาฏ คงทิม
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2553

บทคัดย่อ

งานวิจัยนี้นำเสนอวิธีการกำเนิดข้อมูลทดสอบ LFSR Reseeding แบบขนานสำหรับทดสอบวงจรอิเล็กทรอนิกส์ในระบบ Mixed-mode BIST โดยชุดข้อมูล Seed พลวัตถูกเลื่อนในรูปแบบขนานเข้าสู่วงจร LFSR วงจรเลื่อนเฟส และ Scan Chain ตามลำดับ ซึ่งงานวิจัยนี้นอกเหนือจากการนำเสนอวิธีการกำเนิดข้อมูลทดสอบ LFSR Reseeding แบบขนาน แต่ได้เพิ่มรีจิสเตอร์แบ่งกลุ่ม เพื่อทำให้ประสิทธิภาพของการเข้ารหัสชุดข้อมูลทดสอบดีขึ้น สำหรับสมการอธิบายชุดข้อมูล Seed สามารถแก้สมการได้ด้วยวิธีการ Gauss-Jordan ผลการกำเนิดข้อมูลทดสอบสำหรับวงจรมาตรฐาน ISCAS 89 เป็นตัวบ่งชี้สำคัญว่า ถึงการพัฒนาในแง่ของขนาดชุดข้อมูลทดสอบ ข้อดีหลายประการของวิธีที่นำเสนอในงานวิจัย เช่น วิธีการกำเนิดข้อมูลทดสอบ LFSR Reseeding สามารถครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลทดสอบ ลดจำนวนชุดข้อมูลทดสอบ ลดเวลาการทดสอบ อีกทั้งสามารถครอบคลุมความผิดพลาดสูงสุดเท่ากับความผิดพลาดของชุดข้อมูลทดสอบ

Thesis Title	LFSR Reseeding with Selection Register for Mixed-mode BIST
Author	Miss Piyanart Kongtim
Major Program	Computer Engineering
Academic Year	2010

ABSTRACT

This research presents a new method in order to generate test patterns generator for testing electronic circuit in Mixed-mode BIST with Parallel Linear Feedback Shift Register (LFSR) Reseeding. The dynamic seeds were injected in parallel form through the LFSR, phase shifter, and scan chains, respectively. The research not only proposed the Parallel LFSR Reseeding, but also adds the selection register for improving the efficiency of encoding test data. For seed computing, after the equations were formed, they were solved by Gauss-Jordan elimination. The experimental results for ISCAS 89 benchmark circuits indicate the significant improvement in terms of test data. There are several main advantages of proposed approach such as 100% test coverage, low test data, low test application time, high fault coverage as intended by the deterministic patterns, etc.

กิตติกรรมประกาศ

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ทวีศักดิ์ เรืองพีระกุล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้เสียสละเวลาในการให้คำปรึกษา แนวคิดในการทำวิจัย รวมถึงการช่วยเหลือแก้ไขปัญหาที่เกี่ยวกับการวิจัย ตลอดจนตรวจสอบและแก้ไขวิทยานิพนธ์ให้ดำเนินไปอย่างลุล่วงสมบูรณ์

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต ที่ได้กรุณาเสียสละเวลาเป็นประธานกรรมการสอบวิทยานิพนธ์ และให้ความช่วยเหลือในงานวิจัย ตลอดจนช่วยตรวจทานแก้ไขวิทยานิพนธ์ให้ดำเนินไปด้วยดี

ขอขอบพระคุณรองศาสตราจารย์ ดร.มิตรชัย จงเชี่ยวชำนาญ และรองศาสตราจารย์ ดร.วัฒนพงศ์ เกิดทองมี ที่ได้กรุณาให้คำปรึกษา คำแนะนำ และตรวจทานแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ขอขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ที่ให้การสนับสนุนทุนในการทำวิจัยและให้ความช่วยเหลือด้านการประสานงานต่างๆ

ขอขอบพระคุณ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ที่กรุณาให้ทุนผู้ช่วยวิจัยแก่ข้าพเจ้า

ขอขอบพระคุณ คณาจารย์ บุคลากร และนักศึกษาปริญญาโทภาควิชาวิศวกรรมคอมพิวเตอร์ทุกคนที่ได้ให้คำปรึกษา และกำลังใจในการทำงานเป็นอย่างดีเสมอมา

และสุดท้าย ข้าพเจ้าขอน้อมรำลึกถึงพระคุณของ บิดามารดา และครอบครัว ที่ส่งเสริมและสนับสนุนข้าพเจ้าในทุกๆเรื่องตลอดมาจนสำเร็จการศึกษา

ปิยนาฏ คงทิม

สารบัญ

	หน้า
สารบัญ	(6)
รายการภาพประกอบ	(12)
รายการตาราง	(14)
บทที่1 …….....	1
1.1 ความสำคัญและที่มาของวิทยานิพนธ์……...	1
1.2 วัตถุประสงค์…….....	5
1.3 ขอบเขตการวิจัย…….....	5
1.4 ประโยชน์ที่คาดว่าจะได้รับ…….....	5
1.5 ขั้นตอนการวิจัย…….....	6
บทที่ 2 …….....	7
2.1 บทนำ…….....	7
2.2 ทฤษฎีที่เกี่ยวข้อง…….....	7
2.2.1 Built-In Self-Test (BIST) ……...	7
2.2.2 วงจรกำเนิดชุดข้อมูลการทดสอบ.....	8
2.2.3 Pseudo Random Pattern Generator (PRPG)……	8
2.2.3.1. Linear Feedback Shift Register…….	9
2.2.3.2. Cellular Automata (CA).....	10
2.2.4 ข้อดีและข้อเสียการกำเนิดข้อมูลการทดสอบแบบ PRPG……	12
2.2.5 การแก้ปัญหาการกำเนิดข้อมูลการทดสอบแบบ PRPG……...	14
- วงจรเลื่อนเฟส (Phase Shifter) ……..……..……	14
2.2.6 การเปรียบเทียบเทคนิคการกำเนิดข้อมูลการทดสอบแบบ PRPG……	19
2.2.7 โครงสร้างของวงจรเซลล์ตรวจกวาดแบบ STUMPS……..……	21
2.2.8 ข้อมูล Seed…….....	22
2.2.9 ชุดข้อมูลของวงจรทดสอบ…….....	22

สารบัญ (ต่อ)

หน้า

2.2.10 Selection Register……..... 24

2.3 สรุป……..... 25

บทที่ 3 ……..... 26

3.1 บทนำ……..... 26

3.2 Static LFSR Reseeding ……..... 26

3.2.1 Original static LFSR Reseeding……..... 27

3.2.2 Multi-Polynomial LFSRs Reseeding หรือ MP-LFSRs Reseeding..... 28

3.2.3 Variable-length seeds……..... 29

3.2.4 การจัดเรียงข้อมูลทดสอบด้วยวิธี Seed ordering……..... 31

3.2.5 วิธีกำเนิดข้อมูลทดสอบด้วย LFSR Reseeding ร่วมกับวิธีการอื่น..... 33

3.2.5.1 LFSR Reseeding และ Bit-Fixing..... 33

3.2.5.2 LFSR Reseeding และ วิธีพจนานุกรม..... 35

3.2.5.3 LFSR Reseeding และ วิธีการพจนานุกรมบนพื้นฐานของซอฟแวร์ 36

3.2.6 สรุปวิธีการกำเนิดข้อมูลแบบ Static LFSR Reseeding……. 37

3.3 Dynamic LFSR Reseeding……... 39

3.3.1 Partial LFSR Reseeding…….. 39

3.2.2 Relaxation…….. 42

3.4 สรุป…….. 44

บทที่ 4 ……..... 45

4.1 บทนำ……... 45

4.2 วงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding …….. 45

4.1.1 ข้อมูล Seed…….. 47

4.2.2 กลุ่มของ AND เกต แบบ 2 อินพุท…….. 47

4.2.3 กลุ่มของ XOR เกต แบบ 2 อินพุท…….. 48

สารบัญ (ต่อ)

หน้า

4.2.4 วงจร LFSR…….. 49

4.2.5 วงจรเลื่อนเฟส……. 49

4.2.5.1 การเลือกตำแหน่งสำหรับการป้อนข้อมูล Seed ……. 51

4.2.5.2 การวนลูปของข้อมูล Seed……... 51

4.2.6 วงจรที่จะทดสอบ……. 52

4.2.7 BIST Controller หรือส่วนควบคุมการทดสอบแบบ BIST ……. 52

4.3 วงจรการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register หรือ Parallel LFSR Reseeding with Selection Register ……. 53

4.3.1 ข้อมูล Seed…….. 56

4.3.2 กลุ่มของ AND เกตแบบ 2 และ 3 อินพุท …… 56

4.3.3 ส่วนควบคุมการทดสอบแบบ BIST ……... 57

4.3.3.1 วงจรนับ ……. 57

4.3.3.2 สัญญาณ C …….. 57

4.3.3.3 Selection Register……... 57

4.4 สรุป…….. 58

บทที่ 5 ……... 59

5.1 บทนำ……... 59

5.2 ผลการกำเนิดข้อมูลการทดสอบด้วยสถาปัตยกรรม Parallel LFSR Reseeding แบบพื้นฐาน …….. 59

5.2.1 ผลการกำเนิดข้อมูลการทดสอบวงจร S5378……... 59

5.2.1.1 ข้อมูลเบื้องต้นของวงจร S5378……... 59

5.2.1.2 วงจร LFSR สำหรับทดสอบวงจร S5378…… 60

5.2.1.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S5378…… 60

5.2.1.4 ตำแหน่งการ Seed สำหรับวงจร S5378……... 61

5.2.1.5 สรุปผลการทดสอบของวงจร S5378……... 62

สารบัญ (ต่อ)

หน้า

5.2.2 ผลการกำเนิดข้อมูลการทดสอบวงจร S9234…….. 63

5.2.2.1 ข้อมูลเบื้องต้นของวงจร S9234……... 63

5.2.2.2 วงจร LFSR สำหรับทดสอบวงจร S9234…… 63

5.2.2.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S9234…… 64

5.2.2.4 ตำแหน่งการ Seed สำหรับวงจร S9234……... 66

5.2.2.5 สรุปผลการทดสอบของวงจร S9234……... 67

5.2.3 ผลการกำเนิดข้อมูลการทดสอบวงจร S13207…… 68

5.2.3.1 ข้อมูลเบื้องต้นของวงจร S13207……..…….. 68

5.2.3.1 ข้อมูลเบื้องต้นของวงจร S13207……..…….. 68

5.2.3.1 ข้อมูลเบื้องต้นของวงจร S13207……..…….. 68

5.2.3.4 ตำแหน่งการ Seed สำหรับวงจร S13207……. 70

5.2.3.5 สรุปผลการทดสอบของวงจร S13207……. 71

5.2.4 ผลการกำเนิดข้อมูลการทดสอบวงจร S15850……. 71

5.2.4.1 ข้อมูลเบื้องต้นของวงจร S15850……. 71

5.2.4.2 วงจร LFSR สำหรับทดสอบวงจร S15850…….. 72

5.2.4.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S15850……. 72

5.2.4.4 ตำแหน่งการ Seed สำหรับวงจร S15850..... 74

5.2.4.5 สรุปผลการทดสอบของวงจร S15850…… 75

5.2.5 ผลการกำเนิดข้อมูลการทดสอบวงจร S38417……. 76

5.2.5.1 ข้อมูลเบื้องต้นของวงจร S38417……. 76

5.2.5.2 วงจร LFSR สำหรับทดสอบวงจร S38417…….. 76

5.2.5.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S38417…….. 77

5.2.5.4 ตำแหน่งการ Seed สำหรับวงจร S38417……. 82

5.2.5.5 สรุปผลการทดสอบของวงจร S38417……. 84

5.2.6 ผลการกำเนิดข้อมูลการทดสอบวงจร S38584……. 84

5.2.6.1 ข้อมูลเบื้องต้นของวงจร S38584……. 84

สารบัญ (ต่อ)

หน้า

5.2.6.2 วงจร LFSR สำหรับทดสอบวงจร S38584..... 85

5.2.6.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S38584…….. 85

5.2.6.4 ตำแหน่งการ Seed สำหรับวงจร S38584..... 87

5.2.6.5 สรุปผลการทดสอบของวงจร S38584..... 89

5.3 ผลการกำเนิดข้อมูลการทดสอบด้วยสถาปัตยกรรม Parallel LFSR reseeding ร่วมกับ Selection Register…….. 89

5.3.1 ผลการกำเนิดข้อมูลการทดสอบวงจร S5378 90

5.3.1.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ…… 90

5.3.1.2 สรุปผลการทดสอบของวงจร S5378……... 91

5.3.2 ผลการกำเนิดข้อมูลการทดสอบวงจร S9234……... 92

5.3.2.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ..... 92

5.3.2.2 สรุปผลการทดสอบของวงจร S9234..... 94

5.3.3 ผลการกำเนิดข้อมูลการทดสอบวงจร S13207……. 94

5.3.3.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ..... 94

5.3.3.2 สรุปผลการทดสอบของวงจร S13207……. 95

5.3.4 ผลการกำเนิดข้อมูลการทดสอบวงจร S15850……. 96

5.3.4.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ..... 96

5.3.4.2 สรุปผลการทดสอบของวงจร S15850..... 97

5.3.5 ผลการกำเนิดข้อมูลการทดสอบวงจร S38417…… 98

5.3.5.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ..... 98

5.3.4.2 สรุปผลการทดสอบของวงจร S38417……. 99

สารบัญ (ต่อ)

หน้า

5.3.6 ผลการกำเนิดข้อมูลการทดสอบวงจร S38584……. 100

5.3.5.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ..... 100

5.3.6.2 สรุปผลการทดสอบของวงจร S38584……. 101

5.4 การเปรียบเทียบประสิทธิภาพการลดขนาดข้อมูลการทดสอบ..... 102

5.4.1 การเปรียบเทียบการลดขนาดข้อมูล Seed ด้วยวิธีการ Parallel LFSR reseeding แบบธรรมดา และ วิธีการ Parallel LFSR reseeding ร่วมกับ Selection Register…….. 102

5.4.2 การเปรียบเทียบการลดขนาดข้อมูล Seed ด้วยวิธีการ Parallel LFSR reseeding วิธีการ Parallel LFSR reseeding ร่วมกับ Selection Register กับวิธีการ Original LFSR Reseeding และวิธีการ Partial LFSR Reseeding…… 103

5.4.3 การเปรียบเทียบการลดระยะเวลาการทดสอบ..... 104

บทที่ 6 106

6.1 บทสรุป..... 106

6.2 ข้อเสนอแนะ..... 107

6.2.1 เพิ่มการแบ่งกลุ่มข้อมูล Seed เพื่อเพิ่มประสิทธิภาพการลดข้อมูล Seed……... 107

6.2.2 เพิ่มขั้นตอนวิธีเพื่อจัดการแบงจำนวนชุดข้อมูล Seed ……. 107

บรรณานุกรม 108

ประวัติผู้เขียน 114

ภาคผนวก ก 115

ภาคผนวก ข 123

ภาคผนวก ค 131

รายการภาพประกอบ

ภาพประกอบ		หน้า
รูปที่ 2-1	โครงสร้างการออกแบบวงจรด้วยวิธีการทดสอบแบบ BIST….....	8
รูปที่ 2-2	วงจร LFSR แบบภายนอก ที่มีจำนวน D Flip-Flop เท่ากับ 8 เซลล์……...	10
รูปที่ 2-3	วงจร LFSR แบบภายใน ที่มีจำนวน D Flip-Flop เท่ากับ 8 เซลล์.....	10
รูปที่ 2-4	CA กฎ 90 สำหรับ Flip-Flop ที่ X_i…….	11
รูปที่ 2-5	CA กฎ 150 สำหรับ Flip-Flop ที่ X_i ……..	11
รูปที่ 2-6	การเปรียบเทียบการเกิด Structural Dependency ของวงจร LFSR ทั้งสองแบบ และวงจร CA……	14
รูปที่ 2-7	ผลการกำเนิดข้อมูลด้วยวงจร LFSR ที่มีจำนวน D Flip-Flop เท่ากับ 8.....	15
รูปที่ 2-8	ชุดข้อมูลที่ใช้ในการสร้างวงจรวงจรเลื่อนเฟส.....	16
รูปที่ 2-9	วงจร LFSR ขนาด 8 บิตร่วมกับวงจรเลื่อนเฟสจำนวน 4 ชุด.....	17
รูปที่ 2-10	การกระจายตัวของข้อมูลเมื่อใช้ LFSR ร่วมกับวงจรเลื่อนเฟส รูป (a) วงจร LFSR แบบภายนอก และรูป (b) วงจรแบบ LFSR แบบภาย.....	18
รูปที่ 2-11	เปรียบเทียบเปอร์เซ็นต์การกำเนิดข้อมูลทดสอบด้วยวงจร LFSR และวงจร LFSR ร่วมกับวงจรเลื่อนเฟส.....	19
รูปที่ 2-12	จำนวนข้อผิดพลาดที่เกิดขึ้นในวงจรแล้วไม่สามารถตรวจพบได้ระหว่าง CA และ LFSR ของวงจร c3540……..	20
รูปที่ 2-13	โครงสร้างพื้นฐานของ STUMPS.....	22
รูปที่ 2-14	โครงสร้างวงจรยกเลิก X ของวงจรวิเคราะห์ผล.....	24
รูปที่ 3-1	Original Static LFSR reseeding.....	28
รูปที่ 3-2	โครงสร้าง Multi-Polynomial LFSR reseeding…….	29
รูปที่ 3-3	Variable-length seed…….	30
รูปที่ 3-4	วงจร LFSR ขนาด 4 บิต กำเนิดข้อมูลทดสอบ 10 เซลล์.....	31
รูปที่ 3-5	การรวมกันของ LFSR Reseeding และ Bit-Fixing ……..	34
รูปที่ 3-6	การรวมกันของ LFSR reseeding และ Dictionary…….....	36
รูปที่ 3-7	การรวมกันของ LFSR reseeding และ Dictionary base Code…….....	37

รายการภาพประกอบ (ต่อ)

ภาพประกอบ		หน้า
รูปที่ 3-8	Partial LFSR reseeding…….	40
รูปที่ 3-9	ตัวอย่างการสร้างสมการของ Partial LFSR Reseeding…….	40
รูปที่ 4-1	วงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR reseeding……..	46
รูปที่ 4-2	กลไกการกำเนิดข้อมูลทดสอบในโหมด DTPG เมื่อสัญญาณ C = “1”…….	48
รูปที่ 4-3	กลไกการกำเนิดข้อมูลทดสอบในโหมด DTPG เมื่อสัญญาณ C = “0”…….	48
รูปที่ 4-4	กลุ่มของ XOR เกตแบบ 2 อินพุท…….	49
รูปที่ 4-5	ตัวอย่างวงจร LFSR……...	49
รูปที่ 4-6	ตัวอย่างการวนลูปของข้อมูล Seed บิตที่ 3 และบิตที่ 9 ของข้อมูลลำดับที่ 16 และ 17…….	52
รูปที่ 4-7	ตัวอย่างการวนลูปของข้อมูล Seed บิตที่ 3 และบิตที่ 15 ของข้อมูลลำดับที่ 16 และ 17……..……..	52
รูปที่ 4-8	วงจรการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR reseeding ร่วมกับ Selection Register.....	55

รายการตาราง

รายการตาราง		หน้า
ตารางที่ 2-1	กฎ 90 และกฎ 150 สำหรับ CA…….	12
ตารางที่ 2-2	การเปรียบเทียบจำนวนของอุปกรณ์อิเล็กทรอนิกส์ในวงจรกำเนิดข้อมูลแบบ PRPG……	21
ตารางที่ 2-3	รายละเอียดของวงจร S5378.....	23
ตารางที่ 3-1	การเปรียบเทียบการกำเนิดข้อมูลการทดสอบด้วยวิธี Static LFSR Reseeding……..	38
ตารางที่ 3-2	การตรวจจับความผิดพลาดแบบธรรมดา และแบบ Relaxation.....	43
ตารางที่ 4-1	ตารางข้อมูลสร้างวงจรเลื่อนเฟสเพื่อทดสอบวงจร S5387.....	50
ตารางที่ 5-1	ข้อมูลพื้นฐานของวงจร S5378.....	60
ตารางที่ 5-2	ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 17 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 38 บิต…….	61
ตารางที่ 5-3	ตำแหน่งการ Seed ของวงจร S5378…….....	62
ตารางที่ 5-4	ผลการทดสอบวงจร S5378.....	63
ตารางที่ 5-5	ข้อมูลพื้นฐานของวงจร S9234…….....	63
ตารางที่ 5-6	ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 59 ชุดที่กำเนิดจากวงจร LFSR ขนาด 81 บิต.....	64
ตารางที่ 5-7	ตำแหน่งการ Seed ของวงจร S9234.....	66
ตารางที่ 5-8	ผลการทดสอบวงจร S9234.....	68
ตารางที่ 5-9	ข้อมูลพื้นฐานของวงจร S13207.....	68
ตารางที่ 5-10	ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 24 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 44 บิต.....	69
ตารางที่ 5-11	ตำแหน่งการ Seed ของวงจร S13207.....	70
ตารางที่ 5-12	ผลการทดสอบวงจร S13207.....	71
ตารางที่ 5-13	ข้อมูลพื้นฐานของวงจร S15850.....	72

รายการตาราง (ต่อ)

รายการตาราง		หน้า
ตารางที่ 5-14	ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 34 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 58 บิต	73
ตารางที่ 5-15	ตำแหน่งการ Seed ของวงจร S15850	74
ตารางที่ 5-16	ผลการทดสอบวงจร S15850	76
ตารางที่ 5-17	ข้อมูลพื้นฐานของวงจร S38417	76
ตารางที่ 5-18	ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 83 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 105 บิต	77
ตารางที่ 5-19	ตำแหน่งการ Seed ของวงจร S38417	84
ตารางที่ 5-20	ผลการทดสอบวงจร S38417	84
ตารางที่ 5-21	ข้อมูลพื้นฐานของวงจร S38584	85
ตารางที่ 5-22	ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 54 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 75 บิต	86
ตารางที่ 5-23	ตำแหน่งการ Seed ของวงจร S38584	89
ตารางที่ 5-24	ผลการทดสอบวงจร S38584	89
ตารางที่ 5-25	ผลการทดลองกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR reseeding ด้วย Selection Register สำหรับวงจร S5378	91
ตารางที่ 5-26	สรุปการกำเนิดข้อมูลทดสอบสำหรับวงจร S5378	91
ตารางที่ 5-27	ผลการกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR reseeding ด้วย Selection Register สำหรับวงจร S9234	93
ตารางที่ 5-28	ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S9234	94
ตารางที่ 5-29	ผลการกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR reseeding ด้วย Selection Register สำหรับวงจร S13207	95
ตารางที่ 5-30	ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S13207	96
ตารางที่ 5-31	ผลการกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR reseeding ด้วย Selection Register สำหรับวงจร S15850	97
ตารางที่ 5-32	ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S15850	98

รายการตาราง (ต่อ)

รายการตาราง		หน้า
ตารางที่ 5-33	ผลการทดกำเนิดข้อมูลการทดสอบวงจร S38417.....	99
ตารางที่ 5-34	ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S38417.....	100
ตารางที่ 5-35	ผลการกำเนิดข้อมูลการทดสอบวงจร S38584.....	101
ตารางที่ 5-36	ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S38584.....	102
ตารางที่ 5-37	การผลการลดขนาดข้อมูล Seed ของการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ทั้ง 2 แนวคิด.....	103
ตารางที่ 5-38	การเปรียบเทียบจำนวนข้อมูล Seed ของกำเนิดข้อมูลการทดสอบจำนวน 3 วิธี.....	104
ตารางที่ 5-39	ตารางเปรียบเทียบจำนวนสัญญาณนาฬิกา.....	105

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของวิทยานิพนธ์

ปัจจุบันเทคโนโลยีด้านอุปกรณ์อิเล็กทรอนิกส์ได้พัฒนาไปอยู่ในยุคของนาโน-เทคโนโลยี ทั้งทางด้านประสิทธิภาพ ความสามารถการทำงานและขนาดของอุปกรณ์ โดยมีการออกแบบให้ในชิปหนึ่งชิ้นประกอบไปด้วยวงจรอิเล็กทรอนิกส์มากมาย เช่น ไมโครโปรเซสเซอร์ วงจรอนาลอก หน่วยความจำ เป็นต้น ซึ่งชิปอิเล็กทรอนิกส์เหล่านี้มีขนาดเล็ก ต้องการประสิทธิภาพสูง ทั้งทางด้านความเร็ว ด้านกำลังไฟฟ้าและด้านสมรรถนะ การที่ชิปมีขนาดเล็กจะส่งผลให้อัตราการผลิตต่อหนึ่งหน่วยยูนิต (Yield) ลดลง เนื่องจากเทคโนโลยีสมัยใหม่ ที่เน้นการผลิตชิปให้มีคุณสมบัติข้างต้น เป็นผลให้มีโอกาสเกิดข้อผิดพลาดของขบวนการการผลิตมากขึ้นแล้ว ยังมีชนิดข้อผิดพลาดใหม่ๆเกิดขึ้นตามมาอีกด้วย ซึ่งหากชิปที่ถูกผลิตมีข้อผิดพลาด อาจทำให้การทำงานไม่ถูกต้องหรือไม่ทำงาน โดยผู้ผลิตอาจได้รับความเสียหายมาก จากการถูกส่งคืนของผู้รับซื้อได้

การทดสอบ (Testing) เป็นกระบวนการตรวจสอบการทำงานของชิปในหลายกระบวนการทั้งก่อนและหลังกระบวนการผลิต เช่น กระบวนการออกแบบระดับ Very Large Scale Integrate circuit การทวนสอบการออกแบบ (Design Verification) กระบวนการผลิตระบบอุปกรณ์อิเล็กทรอนิกส์ (Electronic System Manufacturing Process) และการดำเนินงานระดับระบบ (System-Level Operation) เป็นต้น การทดสอบจึงเป็นวิธีการที่จะช่วยลดระยะเวลาในกระบวนการออกแบบและกระบวนการผลิตก่อนออกสู่ท้องตลาด โดยการตรวจจับข้อผิดพลาดและวิเคราะห์ผลการทำงาน วิธีการหนึ่งที่นำมาใช้คือ การตรวจสอบโดยใช้อุปกรณ์ทดสอบอัตโนมัติแบบภายนอก (External Automatic Test Equipment หรือ External ATE) อุปกรณ์นี้ทำหน้าที่สร้างข้อมูลทดสอบจากภายนอกเพื่อทดสอบการทำงานภายในชิป แต่วิธีการดังกล่าวนี้จะต้องใช้ค่าใช้จ่ายสูงในการโปรแกรมติดตั้งอุปกรณ์ [1-2] ทั้งยังจำเป็นต้องใช้จำนวนแบนวิด (Bandwidth) จำนวนช่องสัญญาณและความถี่สูงในการทดสอบ [3] เทคนิคการออกแบบวงจรให้สามารถทดสอบได้ด้วยตัวเองจึงเกิดขึ้น ซึ่งวิธีการดังกล่าวนี้ จะแก้ปัญหาการทดสอบด้วยอุปกรณ์ทดสอบอัตโนมัติแบบภายนอกได้ทั้งสามารถลดระยะเวลาการทดสอบได้ [4-5] โดยเรียกเทคนิคนี้ว่า Design For Testable หรือ DFT [4][6]

เทคนิค DFT เป็นการออกแบบชิปให้มีฟังก์ชั่นการตรวจสอบความผิดพลาดได้ด้วยตัวเอง โดยฟังก์ชั่นการตรวจสอบความผิดพลาดถูกออกแบบเป็นฮาร์ดแวร์และฝังฟังก์ชั่นดังกล่าวลงในชิป เพื่อให้ชิปสามารถตรวจสอบความผิดพลาดได้ด้วยตัวเอง เรียกวิธีการนี้ว่า Built-In Self-Test หรือ BIST [4-5][7-11] วิธีการทดสอบแบบ BIST สามารถสร้างข้อมูลการทดสอบได้หลากหลายวิธี เช่น Exhaustive Test Patterns Generator, Pseudo Random Test Pattern Generator (PRPG) Pseudo-Exhaustive Test Patterns Generator, Deterministic Test Pattern Generator (DTPG) [5] เป็นต้น วิธี PRPG เป็นที่นิยมสำหรับการกำเนิดข้อมูลการทดสอบ เนื่องจากสามารถกำเนิดข้อมูลการทดสอบได้จำนวนมากถึง 2^n-1 ชุด เมื่อ n คือจำนวน Flip-Flop ที่ใช้ในการสร้างวงจร ซึ่งวิธีกำเนิดข้อมูลทดสอบแบบ PRPG สามารถกำเนิดข้อมูลการทดสอบได้ด้วยวงจร Linear Feedback Shift Register (LFSR) และวงจร Cellular Automata (CA) วงจรกำเนิดข้อมูลทั้งสองเป็นวงจรที่มีขนาดเล็ก ใช้ทรัพยากรในการพัฒนาน้อย เมื่อเปรียบเทียบจำนวนอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการสร้างวงจรกำเนิดข้อมูลทั้ง 2 วงจร วงจร LFSR ใช้จำนวนอุปกรณ์อิเล็กทรอนิกส์ในการสร้างวงจรน้อยกว่าวงจร CA [4] และเมื่อพิจารณาถึงจำนวนความถี่ในการกำเนิดข้อมูลการทดสอบโดยพิจารณาที่จำนวนความผิดพลาดเท่ากัน วงจร LFSR ใช้จำนวนชุดข้อมูลทดสอบที่น้อยกว่าวงจร CA [12-13] วงจร LFSR จึงเหมาะสำหรับการกำเนิดข้อมูลทดสอบแบบ PRPG แม้วงจรกำเนิดข้อมูลทดสอบด้วยวงจร LFSR จะสามารถกำเนิดข้อมูลทดสอบได้จำนวนมาก ทั้งใช้จำนวนอุปกรณ์อิเล็กทรอนิกส์ในการสร้างวงจรน้อย แต่ชุดข้อมูลทดสอบที่กำเนิดจากวงจร LFSR เกิด Structural Dependency และ Correlation ของข้อมูล ส่งผลให้ไม่สามารถทดสอบความผิดพลาดที่เป็นความผิดพลาดแบบการรวมลอจิก หรือ Combinational Logic ได้ วิธีการลดการเกิด Structural Dependency และ Correlation ของข้อมูล สามารถประยุกต์วงจรเลื่อนเฟส (Phase Shifter) [14-15] โดยวงจรเลื่อนเฟสจะทำหน้าที่กระจายชุดข้อมูล “1” และ “0” ซึ่งสามารถลดการเกิด Structural Dependency และ Correlation ของข้อมูลทดสอบได้ นอกจาก Structural Dependency และ Correlation ของข้อมูลทดสอบแล้ว การวิธีกำเนิดข้อมูลทดสอบด้วยวงจร LFSR ต้องใช้จำนวนความถี่ในการกำเนิดข้อมูลทดสอบจำนวนมาก (ในการกำเนิดข้อมูลทดสอบวงจร S5378 ต้องใช้ความถี่ในการกำเนิดข้อมูลทดสอบจำนวน 10,000 รอบ เพื่อให้สามารถความคลุม 100% ของข้อมูลทดสอบ) การกำเนิดข้อมูลทดสอบด้วยวงจร LFSR จึงใช้กำเนิดข้อมูลทดสอบเพื่อให้ครอบคลุมความผิดพลาดแบบง่าย หรือ Normal Faults ในช่วงแรกของการทดสอบ และประยุกต์เทคนิคกำเนิดข้อมูลทดสอบแบบ DTPG เพื่อกำเนิดข้อมูลทดสอบที่มีความผิดพลาดแบบสุ่มทน หรือ Random Pattern Resistant (R.P.R.) โดยเรียกการนำวิธีการทดสอบแบบ PRPG และ DTPG มาใช้กำเนิดข้อมูลทดสอบว่า Mixed-mode BIST [5]

เทคนิคการกำเนิดข้อมูลทดสอบด้วยวิธีการ DTPG สามารถแบ่งได้ 3 แบบ [16] ดังนี้

1. การปรับปรุงโครงสร้างวงจรทดสอบ (Modification circuit of test pattern) [17-19] วิธีการนี้เป็นการเพิ่มความสามารถในการตรวจจับความผิดพลาด ด้วยการเพิ่มจุดสังเกตุการณ์ตรวจจับ (Observation point) และ จุดควบคุม (Control Point) ในวงจรทดสอบ โดยจุดสังเกตุการณ์ตรวจจับจะเชื่อมต่อกับ primary input และจุดควบคุมจะเชื่อมต่อกับ primary output ของวงจรทดสอบ วิธีปรับปรุงโครงสร้างของวงจรทดสอบอาจจะส่งผลต่อประสิทธิภาพของชิป เนื่องจากต้องปรับปรุงโครงสร้างภายในของวงจร

2. การกำเนิดข้อมูลทดสอบด้วยการให้ค่าน้ำหนัก (Weight Random Pattern Generation) [20-22] เป็นการเพิ่มวงจรลอจิกเพื่อให้ข้อมูลการทดสอบเกิดการเปลี่ยนแปลงความถี่ของการปรากฏค่า “1” และ “0” ของข้อมูลการทดสอบ ซึ่งการกำเนิดข้อมูลด้วยวิธีการนี้จะต้องใช้ทรัพยากรจำนวนมาก เพราะต้องเพิ่มลอจิกพิเศษเพื่อให้เกิดการเปลี่ยนความถี่ของข้อมูลการทดสอบ

3. การรวมวงจรการทดสอบ (Mixed-mode BIST) [23-45] เป็นการสร้างวงจรที่ใช้ในการเพิ่มประสิทธิภาพตรวจจับข้อผิดพลาดที่มีรูปแบบการสุ่มทน โดยเทคนิคการกำเนิดข้อมูลทดสอบที่เรียกว่า DTPG ซึ่งการกำเนิดข้อมูลด้วยวิธีดังกล่าวนี้สามารถแบ่งได้ 3 แบบ [5] ดังนี้

- **ROM Compression** [23-25] เป็นวิธีที่พัฒนาขึ้นเพื่อช่วยลดขนาดของข้อมูลการทดสอบ โดยใช้เทคนิคการบีบอัด ซึ่งช่วยให้ขนาดของหน่วยความจำที่ใช้ในการเก็บข้อมูลลดลง แต่จะเป็นการเพิ่มวงจรขยายข้อมูลที่ถูกบีบอัดไว้

- **Linear Feedback Shift Register Reseeding หรือ LFSR Reseeding** [26-50] เป็นวงจรที่เกิดจากการนำวงจรกำเนิดข้อมูลทดสอบแบบ PRPG มากำเนิดข้อมูลทดสอบ DTPG ด้วย แต่ไม่มีการกำหนดค่าของอินพุท ใช้การคำนวณชุดข้อมูลทดสอบไว้ล่วงหน้าแล้วค่อยเลื่อนชุดข้อมูลดังกล่าวสู่วงจร LFSR โดยเรียกข้อมูลที่เกิดจากการคำนวณชุดข้อมูลทดสอบไว้ล่วงหน้าว่า “Seed” โดยข้อมูล Seed จะมีขนาดเล็กกว่าข้อมูลของวงจรทดสอบ

- **Embedding Deterministic Pattern** [51-54] เป็นวิธีการนำชุดข้อมูลการทดสอบของ PRPG ที่ไม่สามารถตรวจสอบความผิดพลาดได้มาทำการเปลี่ยนแปลงในบางบิต โดยการเพิ่มลอจิก XOR เกตในวงจร LFSR เช่นวิธี bit-Fixing, bit blipping หรือการ mapping logic

วิธีการกำเนิดข้อมูลแบบ DTPG ด้วยวงจร LFSR Reseeding สามารถลดจำนวนข้อมูลที่ใช้ในการกำเนิดข้อมูลทดสอบได้ ลดขนาดวงจรกำเนิดข้อมูลทดสอบ เนื่องจากมีการนำ

วงจรการกำเนิดข้อมูลการทดสอบด้วยวิธี PRPG คือวงจร LFSR มาใช้ในการกำเนิดข้อมูลการทดสอบแบบ DTPG ทั้งสามารถกำเนิดข้อมูลทดสอบให้ครอบคลุมความผิดพลาดที่มีรูปแบบการสุ่มทนได้ รวมไปถึงสามารถช่วยลดระยะเวลาการทดสอบและไม่มีผลต่อประสิทธิภาพการทำงานของชิป เนื่องจากไม่มีการเปลี่ยนแปลงโครงสร้างภายในของชิป

วิธีการกำเนิดข้อมูลทดสอบด้วยวิธี LFSR Reseeding เป็นการเลื่อนบิตข้อมูลพิเศษหรือข้อมูล Seed เข้าสู่วงจร LFSR เพื่อเปลี่ยนแปลงชุดข้อมูลทดสอบให้ได้ผลลัพธ์ตามที่ต้องการ การกำเนิดข้อมูลทดสอบด้วยวิธีการ LFSR Reseeding มีด้วยกัน 2 แบบ [46] คือ Static LFSR Reseeding และ Dynamic LFSR Reseeding วิธีการกำเนิดข้อมูลการทดสอบแบบ Static LFSR Reseeding ต้องใช้ข้อมูลจำนวนมากในการกำเนิดข้อมูลทดสอบ เนื่องจากใช้การเลื่อนบิตข้อมูล Seed แบบเต็ม หรือ Full Reseeding ซึ่งถูกนำเสนอใน 2 รูปแบบคือ การบีบอัดชุดข้อมูลทดสอบ [26-39] และการเพิ่มประสิทธิภาพของชุดข้อมูล Seed จำนวน 1 ชุด ให้สามารถกำเนิดข้อมูลทดสอบได้มากกว่า 1 ชุดของข้อมูลทดสอบ [40-43] ส่วนวิธีการ Dynamic LFSR Reseeding เน้นการพัฒนาให้สามารถลดการใช้จำนวนข้อมูล Seed ลง โดยยินยอมให้มีการเลื่อนบิตข้อมูล Seed เพียงบางส่วน เช่นวิธีการ Partial LFSR Reseeding [1][44] และวิธีการ Seed Relaxation ร่วมกับการวิเคราะห์ผลลัพท์ของข้อมูล Seed เพื่อลดจำนวนบิตข้อมูล Seed ในการกำเนิดข้อมูลทดสอบ [45] นอกจากนั้น วิธีการ LFSR Reseeding นอกจากจะถูกพัฒนาเพื่อให้ลดข้อมูล Seed แล้ว ยังได้มีการนำเสนอเพื่อลดจำนวนพลังงาน [46-50] อีกด้วย

รายงานวิทยานิพนธ์นี้มีเป้าหมายในการนำเสนองานวิจัยและพัฒนาวิธีการกำเนิดข้อมูลทดสอบแบบ Mixed-mode BIST ด้วยวิธี Parallel LFSR Reseeding แบบ Dynamic ที่ใช้การเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR แบบขนาน ให้มีความสามารถในการกำเนิดข้อมูลทดสอบให้ครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลการทดสอบ (Test Coverage) สอดคล้องกับการทำงานร่วมกับโครงสร้างของเซลล์ตรวจกวาดแบบ Self-Testing Using MISR and Parallel SRSG หรือ STUMPS และวงจรเลื่อนเฟส รวมถึงลดระยะเวลาการทดสอบ การกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ได้นำเสนอใน 2 แนวคิด แนวคิดแรกนำเสนอพื้นฐานการทำงานของวิธีการกำเนิดข้อมูลการทดสอบที่มีการทำงานสอดคล้องกับ STUMPS และวงจรเลื่อนเฟส ส่วนแนวคิดที่สองนำเสนอวิธีการลดข้อมูลการทดสอบโดยการเพิ่มรีจิสเตอร์แบ่งกลุ่ม หรือ Selection Register [55-56] ในวงจรควบคุมการทำงานของ BIST (BIST Controller) ในรายงานวิทยานิพนธ์ฉบับนี้ได้ใช้คำว่า Selection Register แทนคำว่า รีจิสเตอร์แบ่งกลุ่ม เพื่อให้เชื่อมโยงกับเอกสารที่ใช้อ้างอิง

1.2 วัตถุประสงค์

1.2.1 เพื่อพัฒนาวิธีการกำเนิดข้อมูลทดสอบวงจรให้สามารถครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลทดสอบ ด้วยวิธีการ Mixed-mode BIST

1.2.2 เพื่อพัฒนาวิธีการกำเนิดข้อมูลทดสอบของวงจรให้สามารถประหยัดทรัพยากรในการทดสอบ และระยะเวลาการทดสอบได้

1.3 ขอบเขตงานวิจัย

1.3.1 พัฒนาวิธีการกำเนิดข้อมูลทดสอบด้วยวิธีการ LFSR Reseeding ที่มีการเลื่อนชุดข้อมูล Seed แบบขนาน

1.3.2 พัฒนาวิธีการกำเนิดข้อมูลทดสอบของวงจรให้สามารถลดจำนวนข้อมูลการทดสอบด้วยการใช้ Selection Register

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถพัฒนาวิธีทดสอบที่ครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลการทดสอบ ด้วยวิธีการ Mixed-mode BIST ได้

1.4.2 สามารถลดการใช้ข้อมูลการทดสอบ และระยะเวลาการทดสอบได้ โดยไม่เปลี่ยนโครงสร้างของวงจรทดสอบ

1.5 ขั้นตอนการวิจัย

ขั้นที่ 1: ศึกษาแนวทาง และวิธีการดำเนินงานวิจัย

ขั้นที่ 2: ศึกษาวิธีการทดสอบวงจรด้วยวิธีการ Built-In Self-Test (BIST) และ Mixed-mode BIST

ขั้นที่ 3: ศึกษาวิธีการทดสอบวงจรด้วยวิธีการ Mixed-mode BIST แบบ LFSR Reseeding

ขั้นที่ 4: พัฒนาการทดสอบด้วยวิธีการ Mixed-mode BIST แบบ Parallel LFSR Reseeding

ขั้นที่ 5: ศึกษาวิธีการลดจำนวนข้อมูล Seed ด้วยวิธีการ Selection Register

ขั้นที่ 6: พัฒนาการทดสอบด้วยวิธีการ Mixed-mode BIST แบบ Parallel LFSR Reseeding ด้วยวิธีการ Selection Register

ขั้นที่ 7: ปรับปรุงและทดสอบระบบ

ขั้นที่ 8: ตีพิมพ์ผลงานการวิจัย

ขั้นที่ 9: สรุปผล จัดทำรายงานการวิจัยฉบับสมบูรณ์

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 บทนำ

ทฤษฎีและหลักการที่เกี่ยวข้องเป็นการนำเสนอความรู้เกี่ยวกับ การกำเนิดข้อมูลทดสอบด้วยเทคนิค BIST วงจรกำเนิดข้อมูลทดสอบ ข้อดีข้อเสียของวงจรกำเนิดข้อมูลแบบ PRPG การแก้ปัญญาการเกิด Structural Dependency และ Correlation ของข้อมูลที่ได้จากวงจรกำเนิดข้อมูลแบบ PRPR วงจรเลื่อนเฟส Selection Register รวมถึงคำนิยามของคำที่ปรากฏในรายงานวิทยานิพนธ์ฉบับนี้ ซึ่งเนื้อหาทั้งหมดเป็นเนื้อหาความรู้เบื้องต้นของการทดสอบวงจรอิเล็กทรอนิกส์ โดยเนื้อหาทั้งหมดนี้อยู่ในหัวข้อ 2.2 ส่วนหัวข้อ 2.3 เป็นการสรุปทฤษฎีและหลักการที่เกี่ยวข้องซึ่งเป็นแนวทางในการพัฒนางานวิจัย

2.2 ทฤษฎีที่เกี่ยวข้อง

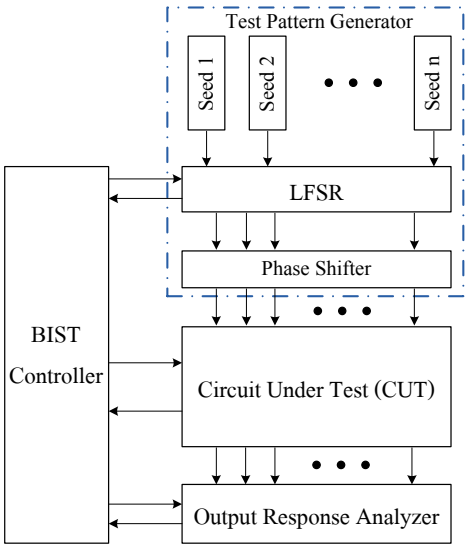
2.2.1 Built-In Self-Test (BIST)

BIST [4-5][7-11] เป็นรูปแบบพื้นฐานในการออกแบบวงจรให้สามารถทดสอบและวิเคราะห์ผลการทดสอบ เพื่อการทดสอบอุปกรณ์อิเล็กทรอนิกส์ที่มีขนาดเล็ก ให้สามารถทดสอบได้ด้วยตัวเอง อย่างมีประสิทธิภาพ และไม่จำเป็นต้องใช้ซอฟต์แวร์ที่มีราคาแพง [3] โดยใช้การออกแบบวงจรให้สามารถทดสอบได้ โครงสร้างการทำงานของ BIST แสดงดังรูปที่ 2-1

จากรูปที่ 2-1 แสดงส่วนประกอบของ BIST ซึ่งประกอบด้วย 3 ส่วน คือวงจรกำเนิดชุดข้อมูลการทดสอบ (Test Pattern Generator หรือ TPG) วงจรที่จะทดสอบ (Circuit Under Test หรือ CUT) และวงจรวิเคราะห์ผลการทดสอบ (Output Response Analyzer หรือ ORA) โดยมีวงจรควบคุมการทำงานของ BIST ทำหน้าที่ในการควบคุมและประสานจังหวะการทำงานระหว่างส่วนการกำเนิดข้อมูล ส่วนวงจรทดสอบ และส่วนวงจรวิเคราะห์ผลการทดสอบ

วงจรกำเนิดชุดข้อมูลการทดสอบที่แสดงในรูปที่ 2-1 คือวิธี LFSR Reseeding ที่ทำงานประสานกับวงจรเลื่อนเฟส ซึ่งวิธีกำเนิดข้อมูลทดสอบด้วยวิธี LFSR Reseeding ประกอบด้วย วงจร LFSR และชุดข้อมูล Seed ส่วนวงจรเลื่อนเฟสจะทำหน้าที่ในการลดการเกิด

Structural Dependency และ Correlation ของข้อมูลที่กำเนิดจากวงจร LFSR ก่อนจะถูกเลื่อนเข้าสู่วงจรที่จะทดสอบ



รูปที่ 2-1 โครงสร้างการออกแบบวงจรด้วยวิธีการทดสอบแบบ BIST

2.2.2 วงจรกำเนิดชุดข้อมูลการทดสอบ

วงจรกำเนิดชุดข้อมูลการทดสอบเป็นเทคนิคการกำเนิดข้อมูลทดสอบเพื่อตรวจจับข้อผิดพลาดของวงจรภายใต้การทดสอบ มีวัตถุประสงค์เพื่อให้สามารถทดสอบได้ใกล้เคียง 100 เปอร์เซ็นต์ของข้อผิดพลาดทั้งหมด หรือได้ค่าใกล้เคียงกับ 100 เปอร์เซ็นต์มากที่สุด ทั้งยังเป็นส่วนสำคัญในลดระยะเวลาการทดสอบและทรัพยากรที่ใช้ [5]

เทคนิคการกำเนิดข้อมูลทดสอบมีหลายเทคนิคด้วยกัน เช่น Exhaustive Test Patterns Generator, PRPG, Pseudo-Exhaustive Test Patterns Generator, Deterministic Test Patterns Generator [5] เป็นต้น แต่ที่นิยมมากที่สุดคือ PRPG เนื่องจากวงจรมีขนาดเล็ก ใช้จำนวนหน่วยความจำน้อย [4-5]

2.2.3 Pseudo Random Pattern Generator (PRPG)

PRPG เป็นเทคนิคการกำเนิดข้อมูลด้วยวิธีการสุ่มแต่สามารถคำนวณชุดข้อมูลชุดถัดไปที่จะเกิดขึ้นได้ มีวงจรขนาดเล็กและสามารถกำเนิดข้อมูลทดสอบได้ถึง 2^n-1 โดย n คือจำนวนอินพุทของวงจรกำเนิดข้อมูล ชุดข้อมูลการทดสอบที่วิธีการ PRPG ไม่สามารถกำเนิดได้คือทุกตำแหน่งมีค่าเท่ากับ “0” เมื่อการกำเนิดข้อมูลครบทั้ง 2^n-1 จะมีการวนซ้ำลำดับกำเนิดข้อมูลเดิม

เทคนิคการกำเนิดข้อมูลการทดสอบจากแบบ PRPG นี้จะอธิบายกลไกการทำงานด้วยสมการโพลิ-โนเมียล (Polynomial) ซึ่งมีด้วยกัน 2 วิธีคือ LFSR และ Cellular Automata (CA)

2.2.3.1. Linear Feedback Shift Register (LFSR) [4-5]

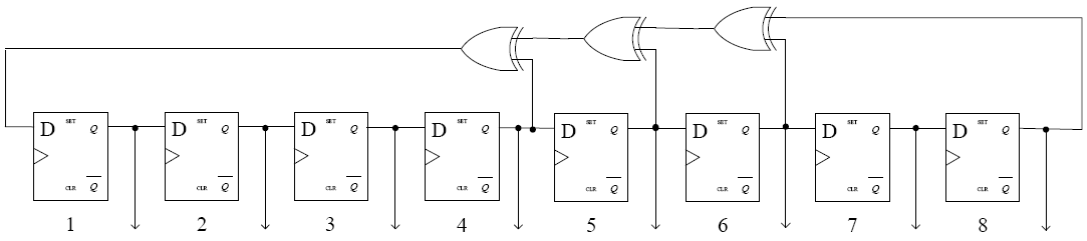
LFSR เป็นเทคนิคกำเนิดข้อมูลทดสอบที่ใช้สมการโพลิโนเมียลในการเลื่อนบิตข้อมูลเป็นลำดับๆ มีอินพุทเป็นเชิงเส้น และมีเอาท์พุทในรูปแบบอนุกรมและแบบขนาน วงจร LFSR ประกอบไปด้วย D Flip-Flop และ Exclusive–OR (XOR) โดย XOR เกตเชื่อมต่อระหว่าง D Flip-Flop เพื่อป้อนกลับข้อมูลเอาท์พุทให้กับ D Flip-Flop เซลล์ข้างเคียง สมการโพลิโนเมียลที่ใช้ในการสร้างวงจร LFSR แสดงดังสมการ (2-1) [12-13]

$$f(x) = C_nX^n + C_{n-1}X^{n-1} + C_{n-2}X^{n-2} + \ldots + C_1X^1+1 \qquad (2\text{-}1)$$

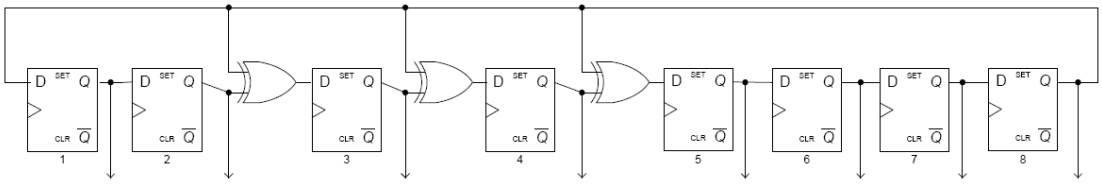
จากสมการ (2-1) ค่า C คือค่าคงที่ของสมการ ค่าคงที่นี้จะใช้ในการบอกตำแหน่งป้อนกลับของเอาท์พุท ซึ่งมีค่าเท่ากับ “0” หรือ “1” หากค่า C ในตำแหน่งใดมีค่าเท่ากับ “0” หมายถึงตำแหน่งนั้นจะไม่มีการป้อนกลับเอาท์พุทไปยังเซลล์ใกล้เคียง หากค่า C ในตำแหน่งใดมีค่าเท่ากับ “1” หมายถึงตำแหน่งนั้นจะมีการป้อนกลับเอาท์พุทไปยังเซลล์ใกล้เคียง ขณะที่ n คือจำนวนเอาท์พุทของข้อมูลในรูปแบบขนาน หรือจำนวนของ D Flip-Flop นั่นเอง ส่วน X ใช้แทนตำแหน่งของ D Flip-Flop

วงจร LFSR มีอยู่ด้วยกัน 2 ชนิด คือ วงจร LFSR แบบภายนอก (External LFSR) และวงจร LFSR แบบภายใน (Internal LFSR) โดยวงจร LFSR แบบภายนอก มีตำแหน่งของ XOR เกตอยู่ด้านนอกของ D Flip-Flop แต่วงจร LFSR แบบภายใน มีตำแหน่งของ XOR อยู่ระหว่าง D Flip-Flop สองตัว ซึ่งแสดงดังรูปที่ 2-2 และ 2-3 ตามลำดับ

รูปที่ 2-2 วงจร LFSR แบบภายนอกที่มีจำนวนอินพุทของวงจรกำเนิดข้อมูลเท่ากับ 8 หรือ D Flip-Flop จำนวน 8 เซลล์ เขียนอยู่ในรูปแบบของสมการโพลิโนเมียลคือ $f(x) = X^8+X^4+X^3+X^2+1$ ซึ่งจากรูปมีตำแหน่งการป้อนกลับของเอาท์พุทในรูปแบบขนานให้กับ D Flip-Flop เซลล์ข้างเคียงตำแหน่งที่ 4, 5, 6 และ 8 ส่วนรูปที่ 2-3 วงจร LFSR แบบภายใน มีจำนวนอินพุทของวงจรกำเนิดข้อมูลเท่ากับ 8 เขียนให้อยู่ในรูปสมการโพลิโนเมียล คือ $f(x) = X^8+X^4+X^3+X^2+1$ โดยมีตำแหน่งการป้อนกลับของเอาท์พุทตำแหน่งที่ 8, 4, 3 และ 2



รูปที่ 2-2 วงจร LFSR แบบภายนอก ที่มีจำนวน D Flip-Flop เท่ากับ 8 เซลล์



รูปที่ 2-3 วงจร LFSR แบบภายใน ที่มีจำนวน D Flip-Flop เท่ากับ 8 เซลล์

ตำแหน่งการป้อนกลับของเอาท์พุทของ D Flip-Flop สามารถอธิบายให้อยู่ในรูปแบบของเซตได้ ซึ่งเรียกว่า tap จากรูปที่ 2-2 วงจร LFSR แบบภายนอก สามารถอธิบายด้วย tap = [8, 6, 5, 4] ซึ่งจะให้ความหมายตำแหน่งการป้อนกลับของเอาท์พุทจาก D Flip-Flop ตำแหน่งที่ 8, 6, 5 และ 4 ซึ่งจะช่วยลดรูปของสมการให้สั้นลงและเข้าใจง่ายขึ้น

จำนวนข้อมูลทดสอบสูงสุดที่วงจร LFSR สามารถกำเนิดได้เท่ากับ 2^n-1 และจะเรียกจำนวนนี้ว่า Maximum Sequence หรือ M-sequence ซึ่งเอาท์พุทที่ LFSR ไม่สามารถสร้างได้คือ ชุดข้อมูลที่ทุกตำแหน่งเป็น 0 เช่นถ้าจำนวน *n* เท่ากับ 8 ได้จำนวนการทดสอบ 2^8-1 = 255 และชุดข้อมูลที่วงจร LFSR ไม่สามารถกำเนิดได้คือ “00000000” สมการโพลิโนเมียลที่ใช้ในการสร้างวงจร LFSR ที่สามารถกำเนิดชุดข้อมูลได้สูงสุดเท่ากับ M-sequence จะเรียกสมการโพลิโนเมียลนั้นว่า โพลิโนเมียลแบบดั้งเดิม (Primitive Polynomial) จากตัวอย่าง $f(x) = X^8+X^6+X^5+X^4+1$ เป็นโพ-ลิโนเมียลแบบดั้งเดิม สมการโพลิโนเมียลแบบดั้งเดิมที่ใช้อ้างอิงจาก [59] โดยจำนวน *n*= 2 ถึง *n*= 411 สามารถดูเพิ่มเติมได้ในภาคผนวก ค

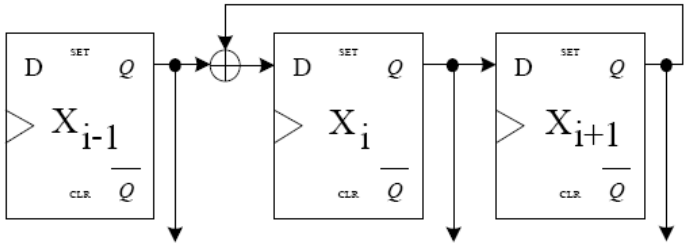
2.2.3.2. Cellular Automata (CA) [4]

การกำเนิดข้อมูลทดสอบด้วยวิธี CA มีลักษณะที่คล้ายกับการกำเนิดข้อมูลทดสอบด้วย LFSR ซึ่งเป็นการกำเนิดข้อมูลทดสอบแบบสุ่มที่สามารถคำนวณข้อมูลการกำเนิดต่อไปได้

และสามารถให้ผลลัพธ์ทั้งรูปแบบขนานและรูปแบบอนุกรมเช่นเดียวกับวงจร LFSR แตกต่างที่สถาปัตยกรรมของวงจร และสมการที่ใช้ในการอธิบายการทำงานเท่านั้น

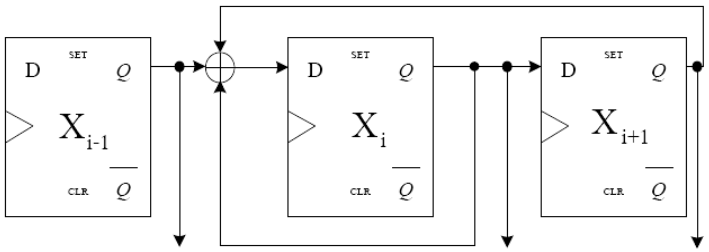
สถาปัตยกรรมของ CA เกิดจาก D Flip-Flops ในรูปแบบอนุกรมของ 2 เซลล์ถัดไปเกิดการป้อนกลับ และ XOR กับผลลัพท์ของเซลล์ก่อนหน้า เพื่อเป็นผลลัพธ์ให้กับเซลล์ถัดไป ซึ่งความสัมพันธ์ของการป้อนกลับเอาท์พุทเป็นผลให้เกิดกฎ (Rule) ขึ้น เพื่ออธิบายความสัมพันธ์ของแต่ละ Flip-Flop โดยกฎที่เป็นที่นิยมมากก็คือ กฎ 90 และ กฎ 150

กฎ 90 เกิดจากข้อมูลรูปแบบอนุกรมของเซลล์ X_{i+1} ถูกป้อนกลับเพื่อ XOR กับข้อมูลรูปแบบอนุกรมของเซลล์ X_{i-1} เพื่อเป็นอินพุทให้กับเซลล์ที่ X_i ซึ่งสามารถเขียนให้อยู่ในรูปของสมการ ณ เวลา t ได้ว่า $X_i(t+1) = X_{i-1}(t) \oplus X_{i+1}(t)$ ตัวอย่างของ CA ที่ใช้กฎ 90 ในการกำเนิดข้อมูลทดสอบแสดงดังรูปที่ 2-4



รูปที่ 2-4 CA กฎ 90 สำหรับ Flip-Flop ที่ X_i

กฎ 150 เกิดจากข้อมูลรูปแบบอนุกรมของเซลล์ X_{i+1} และ X_i ถูกป้อนกลับเพื่อ XOR กับข้อมูลรูปแบบอนุกรมของเซลล์ X_{i-1} เพื่อเป็นอินพุทให้กับเซลล์ที่ X_i ซึ่งสมการของกฎ 150 ที่ ณ เวลา t จะได้ว่า $X_i(t+1) = X_i(t) \oplus X_{i-1}(t) \oplus X_{i+1}(t)$ ตัวอย่างของ CA ที่ใช้กฎ 150 ในการกำเนิดข้อมูลทดสอบแสดงดังรูปที่ 2-5



รูปที่ 2-5 CA กฎ 150 สำหรับ Flip-Flop ที่ X_i

โดยตัวเลข 90 หรือ 150 ที่ใช้อธิบายความสัมพันธ์ของการป้อนกลับเอาท์พุท เกิดจากการนำข้อมูลที่เซลล์ปัจจุบันที่มีค่าเอาท์พุทเป็น “1” มาพิจารณาค่าตำแหน่งของเอาท์พุท ซึ่งแสดงรายละเอียดดังตารางที่ 2-1

จากตารางที่ 2-1 ตัวเลขแสดงชื่อกฎ เช่น 90 เกิดจากเอาท์พุทมีค่า “1” ของเซลล์ $X_i(t+1)$ ที่ลำดับเอาท์พุทต่างๆมารวมกัน ด้วยการนำ 2 ยกกำลังด้วยลำดับ ซึ่งมีลำดับเป็น 1, 3, 4, 6 ผลที่ได้คือ $2^1+2^3+2^4+2^6= 90$ จึงเป็นที่มาของหมายเลข 90 ส่วนกฎ 150 นั้น ตัวเลข 150 เกิดจากเอาท์พุทมีค่า “1” ของเซลล์ $X_i(t+1)$ ที่ลำดับเอาท์พุทที่ให้ค่า “1” มารวมกัน ด้วยการนำ 2 ยกกำลังด้วยลำดับนั้นๆ ซึ่งลำดับ 1, 2, 4, 7 ผลที่ได้ $2^1+2^2+2^4+2^7= 150$

ตารางที่ 2-1 กฎ 90 และกฎ 150 สำหรับ CA

	ลำดับ	7	6	5	4	3	2	1	0
	$X_{i-1}(t)\ X_i(t)\ X_{i+1}(t)$	111	110	101	100	011	010	001	000
กฎ	$X_i(t+1)$	0	1	0	1	1	0	1	0
90	value = 90		2^6		2^4	2^3		2^1	
กฎ	$X_i(t+1)$	1	0	0	1	0	1	1	0
150	value = 150	2^7			2^4		2^2	2^1	

2.2.4 ข้อดีและข้อเสียการกำเนิดข้อมูลการทดสอบแบบ PRPG

การกำเนิดข้อมูลแบบ PRPG ทั้งวิธีกำเนิดข้อมูลทดสอบด้วยวงจร LFSR และวงจร CA ซึ่งทั้งสองวงจรมีข้อดีและข้อเสียของข้อมูลทดสอบ โดยมีรายละเอียดดังนี้

ข้อดี

- **ทรัพยากรที่ใช้ในการสร้างวงจรน้อย** [12-13] พรัพยากรในการพัฒนาวงจรทดสอบแบบ PRPG มีขนาดเล็กเมื่อเปรียบเทียบกับวิธีกำเนิดข้อมูลการทดสอบด้วยวิธีอื่นๆ ที่มีความสามารถในการกำเนิดข้อมูลทดสอบในจำนวนที่เท่าๆกัน

- **วงจรมีขนาดเล็กและง่ายในการออกแบบ** [12-13] วงจรกำเนิดข้อมูลทดสอบด้วยวิธี PRPG ใช้เพียง Flip-Flop ในการออกแบบทำให้วงจรขนาดเล็กและง่ายในการออกแบบ

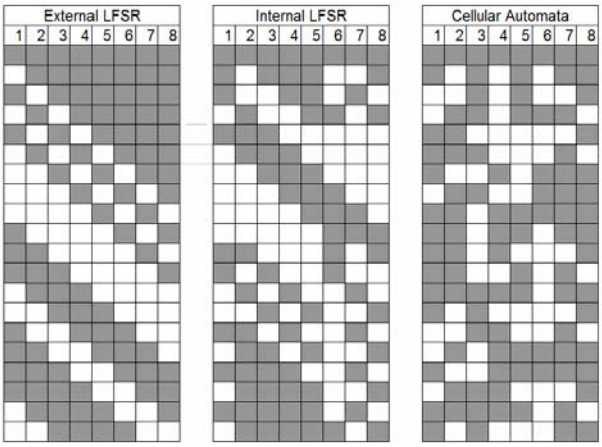
- **กำเนิดข้อมูลที่แตกต่างกันจำนวนมาก** การกำเนิดข้อมูลการทดสอบด้วยวิธี PRPG จำนวน Flip-Flop ตั้งแต่ 3 ขึ้นไป สามารถคำนวณหาสมการโพลิโนเมียลแบบดั้งเดิมได้ ซึ่งสมการโพลิโนเมียลแบบดั้งเดิมจะสามารถกำเนิดข้อมูลการทดสอบในจำนวนมากเท่ากับ M-sequence

ข้อเสีย

- **Structural Dependency** [4] คือกลุ่มข้อมูลที่ได้จากวงจรกำเนิดข้อมูลมีรูปแบบขึ้นอยู่กับโครงสร้างของวงจรที่ใช้กำเนิดข้อมูลทดสอบ ซึ่งทำให้วงจรที่จะทดสอบได้รับชุดข้อมูลทดสอบชุดเดิมซ้ำๆในหลายชุดข้อมูลทดสอบ ดังรูปที่ 2-6 วงจรกำเนิดข้อมูล LFSR แบบภายนอก, ภายใน และวงจร CA มีจำนวนของเอาท์พุทเท่ากับ 8 มีจำนวนผลการกำเนิดข้อมูลทดสอบจำนวน 23 ชุดโดยสีเข้มแทนด้วยบิตข้อมูลที่มีค่า “1” และสีอ่อนแทนบิตข้อมูลที่มีค่า “0”

จากรูปที่ 2-6 การกำเนิดข้อมูลด้วยวงจร LFSR แบบภายนอก แสดงให้เห็นถึงข้อมูลที่มีลักษณะเกิดการเลื่อนตำแหน่งของบิตข้อมูลไปเรื่อยๆ เมื่อพิจารณาถึงบิตที่ 8 จะเห็นได้ว่าสถานะของวงจรทดสอบจะได้รับชุดข้อมูลทดสอบชุดเดิมซ้ำ คือมีค่าเท่ากับ “1” ตั้งแต่ชุดทดสอบที่ 1 ถึง 8 ก่อนจะมีการเปลี่ยนสถานะเป็น “0” ในชุดทดสอบที่ 9 ซึ่งเป็นผลให้ความผิดพลาดที่เกิดจากการรวมกันของลอจิก (Combinational Logic) ไม่สามารถตรวจพบได้ แต่เมื่อพิจารณา LFSR แบบภายนอก จะมีการกระจายของกลุ่มข้อมูลมากขึ้น แต่ยังคงเกิด Structural Dependency ส่วนวงจร CA จะเกิด Structural Dependency น้อยที่สุด เมื่อเปรียบเทียบกับ LFSR แบบภายนอก และ LFSR แบบภายใน

- **Correlation** [4] คือลักษณะข้อมูลทดสอบที่กำเนิดขึ้นมีความสัมพันธ์กับข้อมูลชุดอื่น เช่น ข้อมูลในข้อมูลทดสอบมีจำนวนข้อมูล 3 ชุด 110X0, 1X011 และ 11X01 ถ้าข้อมูลในชุดที่ 2 ตำแหน่งที่ 2 ถูกเปลี่ยนให้เป็น 1 และข้อมูลในชุดที่ 3 ตำแหน่งที่ 3 ถูกเปลี่ยนให้เป็น 0 ข้อมูลทั้ง 3 ชุดมีรูปแบบเป็น Correlated ในบิตที่ 1, 2 และ 3 ส่วนในบิตที่ 4 และ 5 ไม่เกิด Correlated



รูปที่ 2-6 การเปรียบเทียบการเกิด Structural Dependency ของวงจร LFSR ทั้งสองแบบ และวงจร CA [4]

2.2.5 การแก้ปัญหาการกำเนิดข้อมูลการทดสอบแบบ PRPG

ปัญหาการเกิด Structural Dependency และ Correlation ของชุดทดสอบที่ได้จากการกำเนิดข้อมูลการทดสอบแบบ PRPG นั้น สามารถลดได้โดยใช้วงจรเลื่อนเฟส (Phase Shifter) [10-11] เพิ่มการกระจายตัวของข้อมูลทดสอบ

- วงจรเลื่อนเฟส (Phase Shifter)

วงจรเลื่อนเฟสเป็นวงจรการเลื่อนลำดับของเลขฐานสองที่ถูกกำเนิดโดย LFSR ก่อนที่ข้อมูลจะถูกส่งไปยังวงจรภายใต้การทดสอบ การสร้างวงจรเลื่อนเฟสเกิดจากการนำ XOR เกตมาเชื่อมโยงกันระหว่างเอาท์พุทของ LFSR ในแต่ละบิตข้อมูล เพื่อให้ได้กลุ่มข้อมูลที่เกิดการเลื่อนลำดับ เพื่อเพิ่มการกระจายตัวของข้อมูลและลดการเกิด Structural Dependency [14-15]

วิธีการสร้างวงจรเลื่อนเฟส [15] สามารถทำได้โดยการใช้ชุดข้อมูลที่กำเนิดด้วยวิธีการ LFSR แล้วนำชุดข้อมูลที่ได้มาสร้างเป็นวงจรเลื่อนเฟสโดยมีสิ่งที่ต้องพิจารณาดังนี้

B คือจำนวนเอาท์พุทของวงจรเลื่อนเฟสที่ต้องการ

L คือจำนวนน้อยที่สุด ที่จะได้ชุดข้อมูลชุดถัดไป

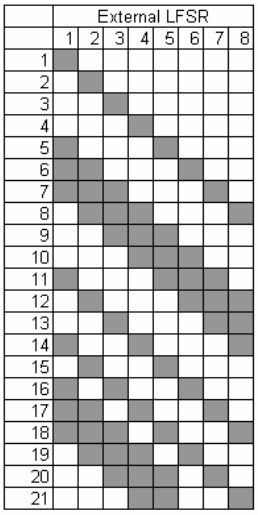
I คือจำนวนหมายเลข “1” ที่ปรากฏในชุดข้อมูล เพื่อควบคุมจำนวนลอจิก XOR เกตให้สามารถลดการใช้ทรัพยากร

จากสมการโพลิโนเมียล $f(X) = X^8+X^6+X^5+X^4+1$ มีจำนวน $n = 8$ เป็นสมการโพลิโนเมียลแบบดั้งเดิม สามารถกำเนิดชุดข้อมูลทดสอบได้สูงสุด 255 ชุด สามารถพิจารณาชุดวงจรเลื่อนเฟสจากผลลัพท์การกำเนิดข้อมูลการทดสอบของวงจร LFSR ซึ่งมีขั้นตอนดังนี้

1. กำหนดชุดข้อมูลเริ่มต้น “10000000” ให้กับวงจร LFSR จากรูปที่ 2-7 ชุดข้อมูลตั้งค่าเริ่มต้นวงจร LFSR แสดงในลำดับที่ 1 ซึ่ง 1 ชุดจะประกอบด้วย 8 บิต ตำแหน่งที่มีค่าเท่ากับ “1” คือตำแหน่งที่ 1 ของวงจร LFSR โดยบิตที่ให้ค่าเท่ากับ “1” จะแสดงด้วยสีเข้ม ส่วนบิตที่ให้ค่าเอาท์พุทเป็น “0” แสดงด้วยสีขาว

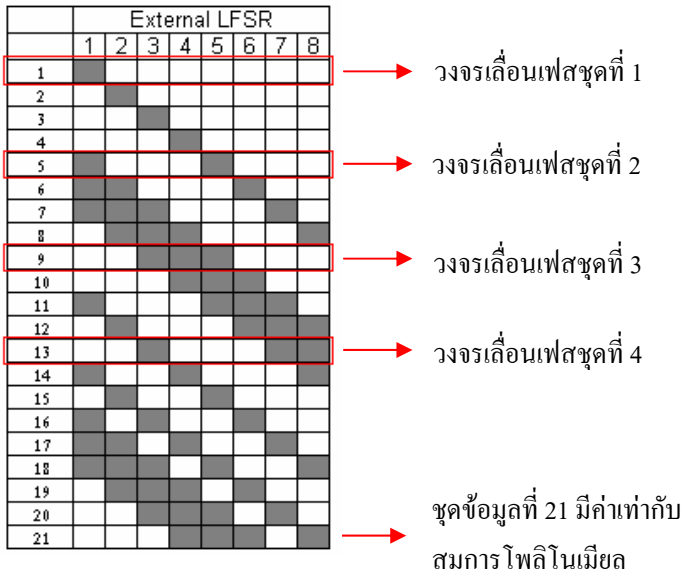
2. กำเนิดข้อมูลทั้งหมด 21 ชุด เนื่องจากชุดข้อมูลที่ 21 มีค่าเท่ากับค่าของสมการโพลีโนมียล คือตำแหน่งที่ 4, 5, 6 และ 8 มีค่าเท่ากับ 1 ผลการกำเนิดข้อมูลด้วยวงจร LFSR แบบภายนอก ที่มีจำนวน D Flip-Flop เท่ากับ 8 แสดงดังรูปที่ 2-7 ผลการกำเนิดข้อมูลด้วยวงจร LFSR ชุดข้อมูลชุดที่ 1 คือข้อมูลในการตั้งค่าเริ่มต้นให้กับวงจร LFSR ชุดข้อมูลที่ 2 ถึง 21 เกิดจากกลไกการเลื่อนบิตของวงจร LFSR และการรวมกันของลอจิก

3. กำหนดค่าตัวแปรในการพิจารณา จำนวนเอาท์พุทของวงจรเลื่อนเฟสที่ต้องการ คือตัวแปร B=4, จำนวนน้อยที่สุดที่จะได้ชุดข้อมูลชุดถัดไป คือตัวแปร L=4 และจำนวนหมายเลข “1” ที่ปรากฏในชุดข้อมูล คือตัวแปร I=3 จากนั้นพิจารณาตามค่าตัวแปร B=4, L=4 และ I=3 ชุดข้อมูลเริ่มต้นถูกเลือกเป็นชุดข้อมูลชุดแรกของวงจรเลื่อนเฟส ชุดวงจรเลื่อนเฟสชุดที่ 2 ได้จากการพิจารณาชุดข้อมูลที่กำเนิดจากวงจร LFSR ชุดถัดไป จำนวน 4 ชุด ตามค่าของตัวแปร L=4 เนื่องจากชุดข้อมูลดังกล่าวมีจำนวนหมายเลข “1” น้อยกว่าค่า I ที่กำหนดไว้ ถ้ามีจำนวนหมายเลข “1” มากกว่าค่า I จะพิจารณาข้อมูลชุดถัดไปจนกว่าจะพบตำแหน่งที่มีค่าตำแหน่งที่มีจำนวนหมายเลข“1” น้อยกว่าหรือเท่ากับค่า I และทำเช่นนี้จนครบเท่ากับจำนวนของตัวแปร B



รูปที่ 2-7 ผลการกำเนิดข้อมูลด้วยวงจร LFSR ที่มีจำนวน D Flip-Flop เท่ากับ 8

ตัวอย่างการหาชุดวงจรเลื่อนเฟสแสดงดังรูปที่ 2-8 ชุดข้อมูลวงจรเลื่อนเฟสชุดแรกที่ถูกเลือกคือ ชุดแรกของเอาท์พุทของวงจรเลื่อนเฟส ส่วนชุดที่ 2 ของวงจรเลื่อนเฟสคือเอาท์พุทของวงจรเลื่อนเฟสลำดับที่ 5 เนื่องจากเป็นค่าลำดับที่ 4 นับจากชุดข้อมูลวงจรเลื่อนเฟสชุดที่ 1 เท่ากับจำนวนค่าคงที่ของตัวแปร L และมีจำนวนหมายเลข “1” จำนวน 2 ตำแหน่ง น้อยกว่าค่าคงที่ของตัวแปร I ส่วนข้อมูลชุดที่ 3 และ 4 ของวงจรเลื่อนเฟสใช้การพิจารณาจากค่าของตัวแปรทั้ง 2 ด้วยเช่นกัน ส่วนจำนวนข้อมูลชุดที่ใช้ในการพิจาณาขึ้นอยู่กับค่าคงที่ของตัวแปร B



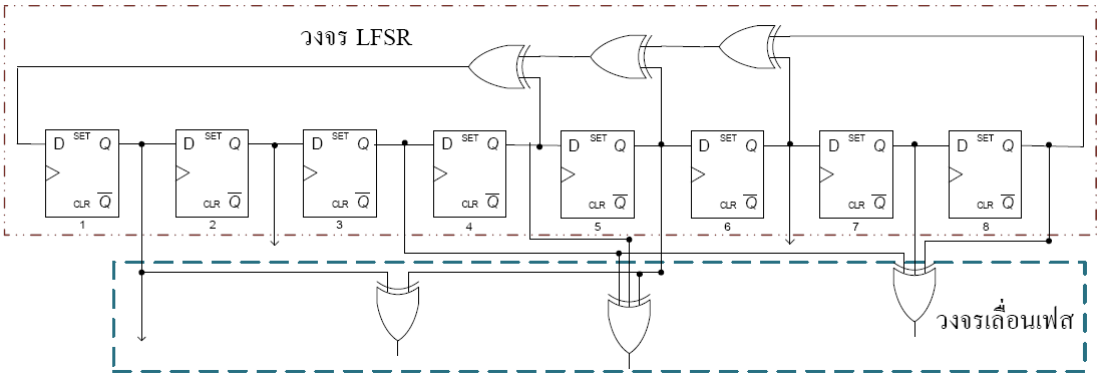
รูปที่ 2-8 ชุดข้อมูลที่ใช้ในการสร้างวงจรวงจรเลื่อนเฟส

ชุดข้อมูลวงจรเลื่อนเฟสที่ถูกเลือก เมื่อนำมาเขียนในรูปแบบของเมตริกซ์ โดยมีสดมภ์แสดงจำนวน D Flip-Flop ของวงจร LFSR และแถวแสดงจำนวนชุดของวงจรเลื่อนเฟส ซึ่งสามารถเขียนได้ดังสมการ (2-2)

$$H = \begin{bmatrix} 10000000 \\ 10001000 \\ 00011100 \\ 00100011 \end{bmatrix} \qquad (2\text{-}2)$$

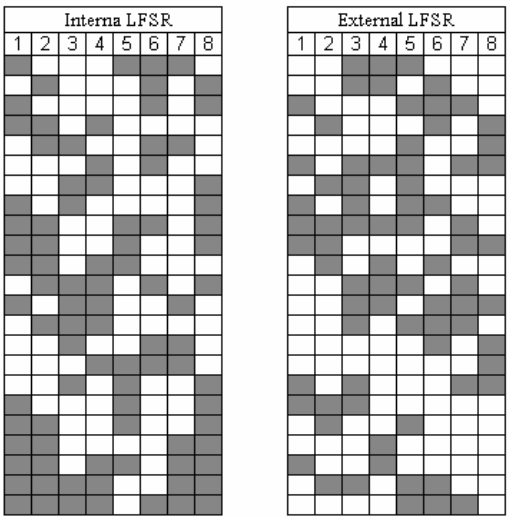
เมื่อพิจารณาจากเมตริกซ์สามารถนำมาสร้างวงจรเลื่อนเฟสได้ โดยหมายเลข “1” ในเมตริกซ์ H บอกตำแหน่งเอาท์พุทของวงจร LFSR ที่จะเป็นอินพุทของวงจรเลื่อนเฟส ผ่านตัว

ดำเนินการ XOR เพื่อเป็นเอาท์พุทของวงจรเลื่อนเฟส โดยแถวแรกของเมตริกซ์ H คือ “10000000” มีตำแหน่งเอาท์พุทตำแหน่งที่ 1 ของ LFSR เป็นอินพุทของวงจรเลื่อนเฟส แถวที่ 2 ของเมตริกซ์ H คือ “10001000” มีตำแหน่งเอาท์พุทของ LFSR ตำแหน่งที่ 1 และ 4 เป็นอินพุทของวงจรเลื่อนเฟส แถวที่ 3 ของเมตริกซ์ H คือ “00011100” มีตำแหน่งเอาท์พุทของ LFSR ตำแหน่งที่ 4, 5 และ 6 เป็นอินพุทของวงจรเลื่อนเฟส และชุดแถวที่ 4 ของเมตริกซ์ H คือ “00100011” มีตำแหน่งเอาท์พุทของ LFSR ตำแหน่งที่ 3, 7 และ 8 เป็นอินพุทของวงจรเลื่อนเฟส จากสมการของเมตริกซ์ H สามารถสร้างวงจร LFSR ร่วมกับวงจรเลื่อนเฟสได้ดังรูปที่ 2-9



รูปที่ 2-9 วงจร LFSR ขนาด 8 บิตร่วมกับวงจรเลื่อนเฟสจำนวน 4 ชุด

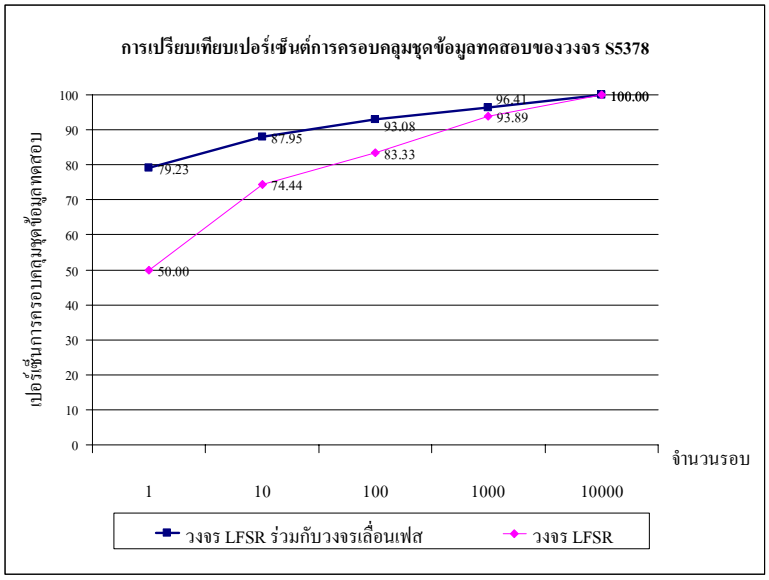
การเพิ่มวงจรเลื่อนเฟสเป็นวิธีการลดการเกิด Structural Dependency ทำให้ข้อมูลที่กำเนิดจากวงจร LFSR แบบภายในหรือภายนอก มีการกระจายตัวของตำแหน่งของข้อมูลที่มีค่าเท่ากับ “1” มากขึ้น ทั้งลดความเป็นรูปแบบของวงจรเลื่อนบิตลง ซึ่งจะเห็นได้จากรูปที่ 2-10 ที่แสดงผลการกระจายตัวของเอาท์พุทที่มีค่าเท่ากับ “1” ของวงจร LFSR ทั้งสองแบบ โดยผลการกระจายตัวของเอาท์พุทส่งผลให้สามารถเพิ่มประสิทธิภาพการตรวจจับความผิดพลาดได้เพิ่มมากขึ้น [1]



(a) (b)

รูปที่ 2-10 การกระจายตัวของข้อมูลเมื่อใช้ LFSR ร่วมกับวงจรเลื่อนเฟส
รูป (a) วงจร LFSR แบบภายนอก และรูป (b) วงจรแบบ LFSR แบบภาย [4]

การลด Structural Dependency และ Correlation ของข้อมูลด้วยวงจรเลื่อนเฟส ทำให้ความสามารถในครอบคลุมความผิดพลาดเพิ่มมากขึ้น จากรูปที่ 2-11 แสดงการเปรียบเทียบเปอร์เซ็นต์การครอบคลุมชุดข้อมูลทดสอบวงจร S5378 ด้วยวิธีกำเนิดชุดข้อมูลทดสอบด้วยวงจร LFSR ร่วมกับวงจรเลื่อนเฟส และวงจร LFSR ผลการกำเนิดข้อมูลทดสอบ จะเห็นได้ว่า ช่วง 1 ถึง 100 รอบของการกำเนิดข้อมูลทดสอบ วงจร LFSR ร่วมกับวงจรเลื่อนเฟสมีเปอร์เซ็นต์การครอบคลุมข้อมูลทดสอบสูงกว่าการกำเนิดข้อมูลทดสอบด้วยวงจร LFSR ซึ่งสามารถครอบคลุมความผิดพลาดได้สูงถึง 93 เปอร์เซ็นต์ของข้อมูลทดสอบ เมื่อต้องการให้ครอบคลุมความผิดพลาดถึง 100 เปอร์เซ็นต์ จำเป็นต้องใช้จำนวนรอบในการกำเนิดข้อมูลสูงถึง 10,000 รอบ การกำเนิดข้อมูลทดสอบด้วยวงจร LFSR ร่วมกับวงจรเลื่อนเฟสจึงถูกนำมาใช้ในการกำเนิดข้อมูลทดสอบเพียงช่วงแรกของการทดสอบเท่านั้น คือเมื่อครอบคลุมความผิดพลาดได้ประมาณ 80 เปอร์เซ็นต์ เพื่อลดระยะเวลาการทดสอบลง จากนั้นจึงใช้วิธีการกำเนิดข้อมูลทดสอบแบบ DTPG มากำเนิดข้อมูลทดสอบให้ครอบคลุมความผิดพลาดที่เหลือ



รูปที่ 2-11 การเปรียบเทียบเปอร์เซ็นต์การกำเนิดข้อมูลทดสอบของวงจร S5378 ด้วยวงจร LFSR และวงจร LFSR ร่วมกับวงจรเลื่อนเฟส

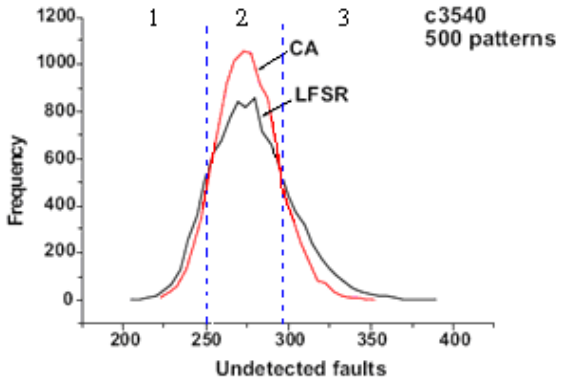
2.2.6 การเปรียบเทียบเทคนิคการกำเนิดข้อมูลการทดสอบแบบ PRPG

สถาปัตยกรรมการกำเนิดข้อมูลการทดสอบแบบ PRPG ด้วยวงจร LFSR และ CA นั้นมีความสามารถในการกำเนิดข้อมูลทดสอบได้ในจำนวนเท่ากัน ด้วยจำนวน Flip-Flop ที่เท่ากัน แต่ประสิทธิภาพในการครอบคลุมข้อผิดพลาดนั้นไม่เท่ากัน จากผลการทดลองในเอกสาร [12-13] ได้ใช้จำนวนชุดข้อมูลที่ใช้ทั้งหมด 500 ชุด ทดสอบด้วยวงจร ISCAS C3540 กราฟผลการทดลองแสดงดังรูปที่ 2-12 แกนนอนแสดงจำนวนข้อผิดพลาดที่ไม่สามารถตรวจจับได้ และแกนตั้งแสดงความถี่ที่ใช้ในการกำเนิดข้อมูลการทดสอบ จากรูปที่ 2-12 สามารถแบ่งกราฟออกเป็น 3 ช่วง

ช่วงที่ 1. ช่วงจำนวนความผิดพลาดที่ไม่สามารถตรวจจับได้ตั้งแต่ 0-250 ชุด จากกราฟจะเห็นได้ว่า เมื่อจำนวนความผิดพลาดที่ไม่สามารถตรวจจับได้เท่ากัน วิธีการกำเนิดข้อมูลทดสอบด้วยวิธี LFSR และวิธี CA มีจำนวนความถี่ในการกำเนิดข้อมูลการทดสอบในอัตราที่ใกล้เคียงกัน

ช่วงที่ 2. ช่วงจำนวนความผิดพลาดที่ไม่สามารถตรวจจับได้ตั้งแต่ 251-290 จากกราฟจะเห็นได้ว่า เมื่อจำนวนความผิดพลาดที่ไม่สามารถตรวจจับได้เท่ากัน วิธีการกำเนิดข้อมูลทดสอบด้วยวิธี CA ต้องใช้จำนวนชุดข้อมูลทดสอบที่มากกว่าวิธี LFSR มาก เช่นเมื่อจำนวนความผิดพลาดที่ไม่สามารถตรวจจับได้เท่ากับ 275 วิธีการ LFSR ต้องใช้ความถี่ในการกำเนิดข้อมูลทดสอบประมาณ 800 ชุด แต่วิธีการ CA ต้องใช้ประมาณ 1000 ชุด

และช่วงที่ 3. ช่วงจำนวนความผิดพลาดที่ไม่สามารถตรวจจับได้ตั้งแต่ 291 เป็นต้นไป จากกราฟจะเห็นได้ว่า เมื่อจำนวนความผิดพลาดที่ไม่สามารถตรวจจับได้เท่ากัน วิธีการกำเนิดข้อมูลทดสอบด้วยวิธี LFSR ต้องใช้จำนวนชุดข้อมูลทดสอบที่สูงกว่าวิธี CA อีกทั้งจำนวนความผิดพลาดที่ตรวจจับไม่ได้ของวิธีการ LFSR มีจำนวนมากกว่าอีกด้วย ซึ่งในช่วงที่ 3 ของการทำสอบจะใช้ชุดข้อมูลทดสอบที่กำเนิดด้วยวิธี DTPG ส่วนชุดข้อมูลทดสอบที่กำเนิดด้วยวิธี PRPG จะไม่ถูกนำมาใช้



รูปที่ 2-12 จำนวนข้อผิดพลาดที่เกิดขึ้นในวงจรแล้วไม่สามารถตรวจพบได้ระหว่าง CA และ LFSR ของวงจร c3540 [12-13]

จากกราฟเมื่อพิจารณาประสิทธิภาพด้านการตรวจจับความผิดพลาด วงจร LFSR สามารถตรวจจับความผิดพลาดได้ดีกว่า เนื่องจากจำนวนความถี่ในการกำเนิดข้อมูลทดสอบที่น้อยกว่า แต่ถ้าพิจารณาถึงระยะเวลาการทดสอบวงจร CA ใช้ระยะเวลาการทดสอบน้อยกว่าวิธีการ LFSR [5]

เมื่อทำการเปรียบเทียบด้านจำนวนอุปกรณ์อิเล็กทรอนิกส์ในการสร้างวงจรทดสอบของ CA และ LFSR แล้ว หากพิจารณาที่ 8 Flip-Flop เท่ากัน วงจรกำเนิดข้อมูลทดสอบด้วยวิธี CA จะต้องใช้จำนวน XOR เกต และ AND เกต ในจำนวนที่มากกว่า LFSR แต่ทั้งสองวงจรสามารถกำเนิดข้อมูลการทดสอบสูงสุดที่ 2^8-1 หรือเท่ากับ 255 ชุด ตารางการเปรียบเทียบจำนวนของอุปกรณ์อิเล็กทรอนิกส์ในวงจรกำเนิดข้อมูลแบบ PRPG แสดงดังตารางที่ 2-2

ชนิด PRPG	จำนวน Flip-Flops	จำนวน XOR เกต	จำนวน AND เกต
LFSR	8	3	8
CA	8	8	15

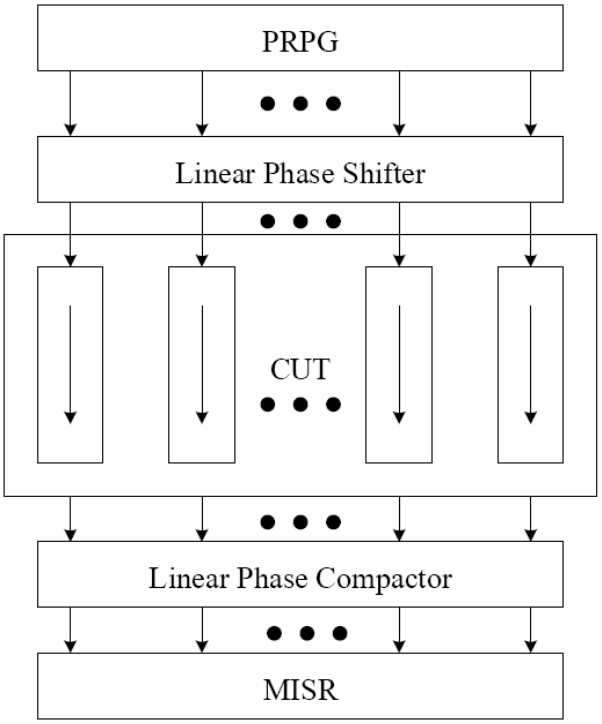
ตารางที่ 2-2 การเปรียบเทียบจำนวนของอุปกรณ์อิเล็กทรอนิกส์ในวงจรกำเนิดข้อมูลแบบ PRPG [4]

วิธีการกำเนิดข้อมูลการทดสอบด้วยวิธี PRPG ต้องใช้ระยะเวลาการทดสอบและค่าใช้จ่ายสูง การทดสอบด้วยวิธีการ PRPG จึงใช้ในการกำเนิดข้อมูลการทดสอบให้ครอบคลุมความผิดพลาดค่าหนึ่งเท่านั้น ([57-58] อธิบายค่าเหมาะสมต่อการกำเนิดข้อมูลด้วยวิธีการ PRPG) วิธีการกำเนิดข้อมูลทดสอบวงจร LFSR จึงเป็นที่นิยมในการกำเนิดข้อมูลทดสอบมากกว่าวงจร CA เนื่องจากการทดสอบในช่วงแรกของการทดสอบ วงจร LFSR ใช้จำนวนชุดข้อมูลการทดสอบน้อยกว่าวงจร CA หากพิจารณาที่การครอบคลุมความผิดพลาดที่เท่ากัน อีกทั้งวงจรกำเนิดข้อมูลทดสอบด้วยวิธี LFSR ยังใช้จำนวนอุปกรณ์อิเล็กทรอนิกส์ในการสร้างวงจรทดสอบที่น้อยกว่าวงจร CA วิทยานิพนธ์นี้จึงได้นำวิธีการกำเนิดข้อมูลทดสอบด้วยวงจร LFSR มาใช้ในการพัฒนา เพื่อกำเนิดข้อมูลการทดสอบแบบ PRPG

2.2.7 โครงสร้างของวงจรเซลล์ตรวจกวาดแบบ STUMPS

โครงสร้างของวงจรเซลล์ตรวจกวาดแบบ Self Test Using MISR and Parallel SRSG หรือ STUMPS โดย STUMPS เป็นโครงสร้างพื้นฐานของการทดสอบแบบ BIST ที่จะสามารถทำงานร่วมกับวงจร PRPG และ Scan Chain แบบหลาย หรือ Multiple Scan Chain รวมทั้งวงจรวิเคราะห์ผลการทดสอบแบบ MISR ด้วย ปัจจุบันโครงสร้างพื้นฐานแบบ STUMPS ถูกนำไปใช้อย่างกว้างขวางในโรงงานอุตสาหกรรม ในวัตถุประสงค์เพื่อลดขนาดของวงจร PRPG และวงจรวิเคราะห์ผลการทดสอบแบบ MISR รวมถึงลดการสุ่มของชุดข้อมูลทดสอบที่กำเนิดจากวงจร PRPG

รูปที่ 2-13 แสดงโครงสร้างพื้นฐานของ STUMPS ที่จะทำงานร่วมกับวงจร PRPG วงจรเลื่อนเฟส วงจรบีบอัดเฟสเชิงเส้น หรือ Linear Phase Compactor รวมทั้งวงจรวิเคราะห์ผลการทดสอบแบบ MISR ด้วย ซึ่งโครงสร้างพื้นฐานเหล่านี้คือ โครงสร้างของวงจรทดสอบแบบ BIST



รูปที่ 2-13 โครงสร้างพื้นฐานของ STUMPS [5]

2.2.8 ข้อมูล Seed

ข้อมูล Seed [5] คือ ตัวแปรอิสระที่จะถูกเลื่อนเข้าสู่วงจร LFSR เพื่อเปลี่ยนแปลงชุดข้อมูลทดสอบ ให้ได้ผลลัพธ์ตามที่ต้องการ โดยเรียกวิธีการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR เพื่อกำเนิดชุดข้อมูลทดสอบนี้เรียกว่า วิธีการ LFSR Reseeding โดยข้อมูล Seed ที่ถูกเลื่อนเข้าสู่วงจร LFSR สามารถคำนวณหาผลลัพท์ได้ด้วยการสร้างสมการเชิงเส้น และแก้สมการด้วยวิธี Gauss-Jordan Elimination

2.2.9 ชุดข้อมูลของวงจรทดสอบ

ชุดข้อมูลของวงจรทดสอบของแต่ละวงจรที่จะทดสอบจะประกอบด้วยข้อมูล 2 ประเภทคือ - Specific bit หรือที่เรียกว่าบิตเจาะจงค่า ที่จะประกอบด้วยบิตข้อมูล “0” หรือ “1”

- Don’t care bit หรือที่เรียกว่าบิตไม่สนใจ ค่า X

ในการกำเนิดข้อมูลการทดสอบ บิตที่เป็นบิตเจาะจงค่าคือ เป้าหมายในการกำเนิดข้อมูลการทดสอบเพื่ออธิบายข้อมูลในบิตให้ตรงตามค่าของบิตข้อมูลนั้น ส่วนการกำเนิดข้อมูลการทดสอบค่า X ข้อมูลที่กำเนิดมาให้ผลลัพธ์เป็น 0 หรือ 1 ก็ได้ เช่น ข้อมูลการทดสอบขนาด 8 บิต มีค่าเป็น 1X0101XX วงจรกำเนิดข้อมูลการทดสอบที่กำเนิดข้อมูลเป็น 10010111 หรือ 11010111 หรือ

10010100 หรือ 11010100 หรือ 10010101 หรือ 11010101 หรือ 10010110 หรือ 11010110 ก็สามารถกำเนิดข้อมูลทดสอบอธิบายชุดข้อมูลทดสอบดังกล่าวได้

ชุดข้อมูลของวงจรทดสอบแต่ละวงจรจะเรียกว่า Test Pattern โดย 1 Test Pattern ประกอบด้วยหลายๆ Test Cube ข้อมูลจำนวน 1 Test Cubes จะถูกแบ่งออกเป็นหลายๆ Scan chain ซึ่งใน 1 Scan Chain จะประกอบไปด้วยหลายๆ เซลล์ตรวจกวาด โดยจำนวนเซลล์ตรวจกวาด 1 Scan Chain เรียกอีกอย่างว่า Scan Length โดย Scan Chain ที่มีจำนวนของเซลล์ตรวจกวาดมากที่สุดจะใช้นับ Scan Length เช่น วงจร S5378 แสดงรายละเอียดดังตารางที่ 2-3

ตารางที่ 2-3 รายละเอียดของวงจร S5378 มีข้อมูลทั้งหมด 30 Test Cube โดย 1 Test Cube จะประกอบด้วย 214 บิตของเซลล์ตรวจกวาด หรือแบ่งออกเป็น 13 Scan Chain ทำให้ 1 Scan Chain จะมี 17 เซลล์ตรวจกวาด และ Scan Chain ที่มีจำนวนเซลล์ตรวจกวาดน้อยสุดเท่ากับ 10 เซลล์ตรวจกวาด ซึ่งขนาดของเอาท์พุทของวงจรเลื่อนเฟสวงจร S5378 มีค่าเท่ากับจำนวนความยาวของ Scan Chain

ตารางที่ 2-3 แสดงรายละเอียดของวงจร S5378

ชื่อวงจร	Test Cubes	เซลล์ตรวจกวาด	Scan Chain	ความยาวของ Scan Chain
S5378	30	214	13	17

จำนวน Scan Chain ที่ใช้ในการกำเนิดข้อมูลการทดสอบจะพิจาณาจากสมการ (2-3) [5] โดยที่จำนวนบิต Flip-Flop ของวงจร LFSR คูณกับจำนวนของชุดข้อมูลทดสอบแล้วจะต้องไม่มากกว่าจำนวน M-Sequence ของวงจร LFSR ที่สามารถกำเนิดได้

$$2^{n}\text{-}1 \geq N_{FF} * N_{TV} \qquad (2\text{-}3)$$

เมื่อ N_{FF} คือจำนวนบิต Flip-Flop ของวงจร LFSR

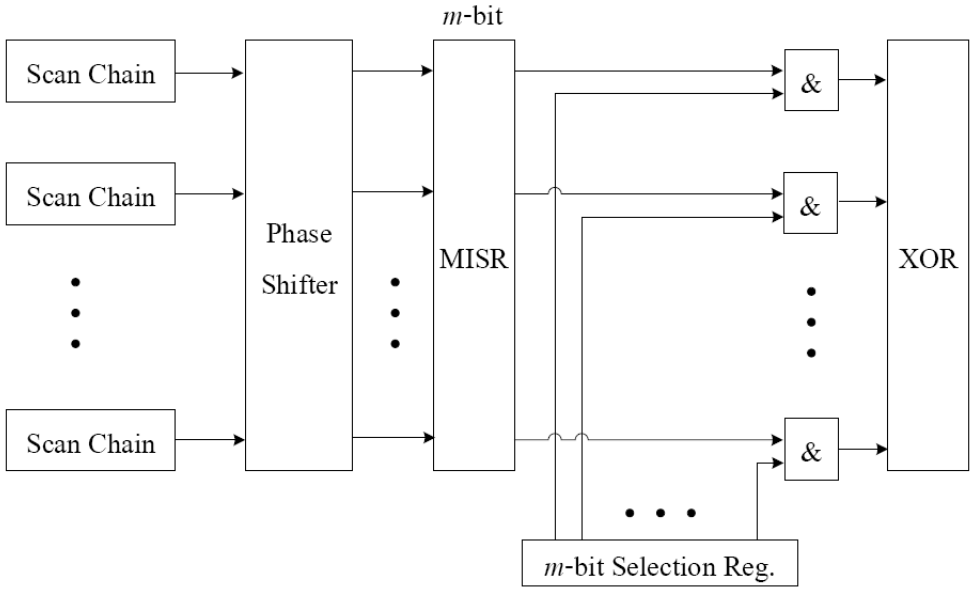
N_{TV} คือจำนวนของชุดข้อมูลทดสอบ

เมื่อจำนวนบิต Flip-Flop ของวงจร LFSR คูณกับจำนวนของชุดข้อมูลทดสอบมีค่ามากกว่าจำนวนของ M-Sequence ของวงจร LFSR จะทำให้เกิด Linear Dependency ของข้อมูล [5] ซึ่งทำให้การกำเนิดข้อมูลทดสอบไม่สามารถกำเนิดข้อมูลได้ทุกความน่าจะเป็น จึงทำให้บางความผิดพลาดไม่สามารถตรวจสอบได้

2.2.10 Selection Register [55-56]

Selection Register [55-56] เป็นกลไกลดความน่าจะเป็นในการเกิดความผิดพลาดแบบ X ด้วยวิธี การยกเลิก X หรือ X's canceling ของวงจร Multiple Input Shift Register (MISR) ซึ่งอยู่ในส่วนของการวิเคราะห์ผลการทดสอบ ในวงจรวิเคราะห์ผลจะไม่ต้องการให้เกิดเอาท์พุทที่เป็น X เนื่องจากค่า X เป็นค่าที่ไม่ทราบค่าที่แน่นอนว่าเป็น "1" หรือ "0" โดยสถาปัตยกรรมของวงจรยกเลิก X ของวงจรวิเคราะห์ผลแสดงดังรูปที่ 2-14

จากรูปที่ 2-14 โครงสร้างวงจร X-Canceling MISR ได้ประยุกต์ใช้ Selection Register สำหรับตำแหน่งที่ต้องการยกเลิก X โดยจำนวนบิตของ Selection Register ที่แทนด้วยตัวแปร *m* จะมีค่าเท่ากับความกว้างของวงจรวิเคราะห์ผล เมื่อต้องการจะยกเลิก X ที่ตำแหน่งใด ข้อมูลจาก Selection Register ก็จะให้ค่า "0" ที่ตำแหน่งนั้น และทำการ AND กับข้อมูลที่เอาท์พุท X ของ LFSR ก็จะให้ผลลัพธ์เป็น "0" ซึ่งจะเป็นการยกเลิกเอาท์พุท X จากวงจรวิเคราะห์ผล



รูปที่ 2-14 โครงสร้างวงจรยกเลิก X ของวงจรวิเคราะห์ผล [55-56]

ด้วยหลักการยกเลิกข้อมูล X ในบางตำแหน่งของเอาท์พุทจากวงจร MISR ของ Selection Register จึงสามารถนำมาประยุกต์เพื่อกำหนดกลุ่มของการเลื่อน หรือยกเลิกบิตข้อมูล Seedได้

2.3 สรุป

วิธีการแบบ Mixed-mode BIST เป็นเทคนิคการกำเนิดข้อมูลทดสอบที่ใช้การกำเนิดข้อมูลการทดสอบทั้งสองรูปแบบคือ PRPG และ DTPG โดยวิธีการกำเนิดข้อมูลทดสอบแบบ PRPG ทำหน้าที่กำเนิดข้อมูลทดสอบเพื่อทดสอบความผิดพลาดแบบธรรมดา เพื่อลดค่าใช้จ่ายจากการข้อมูลทดสอบแบบ PRPG เพียงวิธีการเดียวให้ครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลทดสอบ วงจร LFSR จึงถูกนำมาใช้กำเนิดข้อมูลทดสอบความผิดพลาดแบบธรรมดาในช่วงแรกของการทดสอบ เนื่องจากมีประสิทธิภาพในการครอบคลุมความผิดพลาดมากกว่าวงจร CA ทั้งยังใช้จำนวนอุปกรณ์อิเล็กทรอนิกส์ในการสร้างวงจรทดสอบน้อย วงจรมีขนาดเล็ก สามารถกำเนิดข้อมูลการทดสอบได้จำนวนมาก ในบางชุดข้อมูลทดสอบที่กำเนิดจากวงจร LFSR จะเกิด Structural Dependency และ Correlation ของข้อมูล ซึ่งการเกิด Structural Dependency และ Correlation ของข้อมูล จะส่งผลให้ลดประสิทธิภาพในการครอบคลุมความผิดพลาดลง กลุ่มของ XOR เกตหรือที่เรียกว่าวงจรเลื่อนเฟส ถูกนำมาใช้เพิ่มการกระจายตัวของข้อมูล “1” และ “0” ลดการเกิด Structural Dependency และ Correlation ของข้อมูลลง ทำให้ช่วงแรกของการทดสอบสามารถครอบคลุมความผิดพลาดได้เพิ่มมากขึ้น ส่วนความผิดพลาดที่มีรูปแบบสุ่มทน จะใช้การกำเนิดข้อมูลการทดสอบแบบ DTPG ซึ่งวิธีการกำเนิดข้อมูลการทดสอบด้วยวิธี LFSR Reseeding ทั้งแบบ Static และแบบ Dynamic เป็นวิธีการกำเนิดข้อมูลทดสอบแบบ DTPG ที่ไม่จำเป็นต้องเปลี่ยนแปลงวงจรทดสอบ และเน้นการลดการใช้จำนวนทรัพยากร โดยการประยุกต์ใช้วงจร LFSR มากำเนิดข้อมูลการทดสอบแบบ DTPG จึงทำให้ใช้ทรัพยากรน้อยในการสร้างวงจรทดสอบ ทั้งยังสามารถพัฒนาให้รองรับการทำงานร่วมกับวงจรเลื่อนเฟส และโครงสร้างของเซลล์ตรวจกวาดแบบ STUMPS ได้ แต่การกำเนิดข้อมูลการทดสอบด้วยวิธี LFSR Reseeding แบบ Dynamic สามารถนำมาพัฒนาให้สามารถลดจำนวนข้อมูลการทดสอบได้มากกว่าวิธีการ LFSR Reseeding แบบ Static เนื่องจากวิธีการ LFSR Reseeding แบบ Dynamic ไม่จำเป็นต้องเลื่อนบิตข้อมูล Seed แบบเต็ม สามารถเลื่อนบิตข้อมูล Seed ในบางชุดหรือบางตำแหน่งของวงจร LFSR ได้ อีกทั้งขณะทำการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR สามารถกำเนิดข้อมูลทดสอบได้ในเวลาเดียวกัน จึงสามารถพัฒนาให้สามารถลดจำนวนสัญญาณนาฬิกาในการกำเนิดข้อมูลการทดสอบได้

งานวิจัยนี้จึงได้พัฒนาวิธีการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR Reseeding แบบ Dynamic ที่เน้นการลดการใช้ทรัพยากรในการจัดเก็บข้อมูล Seed รองรับการทำงานร่วมกับวงจรเลื่อนเฟส และเซลล์ตรวจกวาดแบบ STUMPS ลดการเกิด Structural Dependency และ Correlation ของข้อมูล ครอบคลุม 100 เปอร์เซ็นต์การทดสอบ อีกทั้งลดระยะเวลาการทดสอบ

บทที่ 3
การกำเนิดข้อมูลทดสอบด้วยวิธีการ LFSR Reseeding

3.1 บทนำ

เทคนิคกำเนิดข้อมูลทดสอบแบบ Mixed-mode BIST เป็นเทคนิคกำเนิดข้อมูลการทดสอบที่รวมการกำเนิดข้อมูลแบบ PRPG และแบบ DTPG ไว้ด้วยกัน โดยการกำเนิดข้อมูลแบบ PRPG ทำหน้าที่กำเนิดข้อมูลทดสอบเพื่อให้ครอบคลุมข้อผิดพลาดแบบง่าย และการกำเนิดข้อมูลแบบ DTPG ทำหน้าที่กำเนิดข้อมูลทดสอบเพื่อให้ครอบคลุมข้อผิดพลาดแบบสุ่มทน ซึ่งวิธีการกำเนิดข้อมูลด้วยวิธีการ LFSR Reseeding แบบ Mixed-mode BIST เป็นวิธีหนึ่งของการกำเนิดข้อมูลทดสอบที่ไม่ส่งผลต่อสมรรถนะของวงจรที่จะทดสอบ ทั้งไม่มีผลต่อการเพิ่มจำนวนทรัพยากรของวงจรทดสอบ และสามารถกำเนิดข้อมูลการทดสอบให้ครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลทดสอบได้ อีกทั้งวิธี LFSR Reseeding จะนำวิธีการกำเนิดข้อมูลทดสอบแบบ PRPG มาใช้ในการกำเนิดข้อมูลทดสอบแบบ DTPG จึงสามารถประหยัดทรัพยากรที่ใช้ในการสร้างวงจรทดสอบได้

การกำเนิดข้อมูลทดสอบด้วยวิธี LFSR Reseeding มีด้วยกัน 2 วิธี คือ วิธี Static LFSR Reseeding และ Dynamic LFSR Reseeding ซึ่งวิธีการกำเนิดข้อมูลทดสอบทั้งสองวิธีได้มีการนำเสนอกระบวนการเพื่อลดจำนวนข้อมูลทดสอบ หรือเพื่อเน้นประสิทธิภาพครอบคลุมความผิดพลาดแบบต่างๆ อาทิเช่น วิธีการจำนวนลดข้อมูล Seed ในรูปแบบต่างๆ วิธีบีบอัดชุดข้อมูลทดสอบ วิธีจัดเรียงข้อมูลการทดสอบเพื่อลดจำนวนพื้นที่ในการจัดเก็บ เป็นต้น

วิธีกำเนิดข้อมูลทดสอบด้วยวิธี Static LFSR Reseeding อยู่ในหัวข้อ 3.2 ในหัวข้อ 3.3 เป็นรายละเอียดเกี่ยวกับวิธี Dynamic LFSR Reseeding และหัวข้อ 3.4 เป็นการสรุปและนำเสนอแนวคิดของการวิจัย การพัฒนาวิธีการทดสอบด้วย LFSR Reseeding

3.2 Static LFSR Reseeding

การกำเนิดข้อมูลแบบ Static LFSR Reseeding เป็นการคำนวณข้อมูล Seed สำหรับแต่ละชุดของวงจรทดสอบ เพื่อให้สามารถเพิ่มประสิทธิภาพการครอบคลุมความผิดพลาดที่มากขึ้น โดยข้อมูล Seed ดังกล่าวนี้ จะถูกเลื่อนเข้าสู่วงจร LFSR มีรูปแบบเป็น Reseeding แบบเต็ม (Full

Reseeding) คือจำนวนข้อมูล Seed จะต้องเท่ากับจำนวน Flip-Flop ของวงจร LFSR เพื่อกำเนิดข้อมูลทดสอบให้กับแต่ละชุดของวงจรทดสอบ ซึ่งข้อมูล Seed ที่ได้จากการคำนวณจะถูกจัดเก็บไว้ในหน่วยความจำของวงจรควบคุมแบบ BIST เป็นผลให้ไม่จำเป็นต้องเพิ่มหน่วยความจำในการเก็บข้อมูล Seed การกำเนิดข้อมูลทดสอบแบบ Static LFSR Reseeding นั้น ส่วนใหญ่นิยมพัฒนาเพื่อบีบอัดข้อมูล Seed และการบีบอัดข้อมูลทดสอบ เช่น Original Static LFSR Reseeding, Multi-Polynomial LFSRs Reseeding หรือ MP-LFSRs Reseeding, Variable-length seeds, การจัดเรียงข้อมูลทดสอบด้วยวิธี seed ordering และวิธี LFSR Reseeding ร่วมกับพจนานุกรม

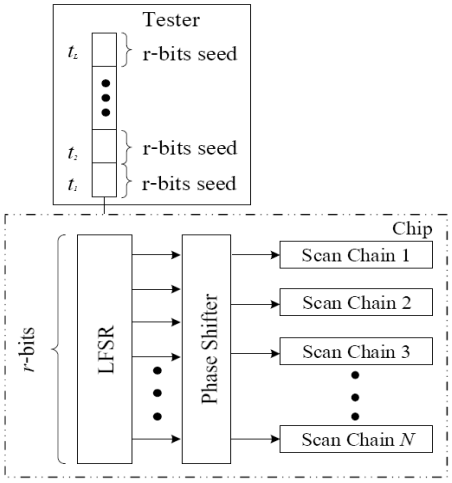
3.2.1 Original Static LFSR Reseeding [26]

วิธีการกำเนิดข้อมูลทดสอบด้วยวิธีนี้ เป็นการลดจำนวนข้อมูลทดสอบ โดยการเข้ารหัสข้อมูลทดสอบที่มีบิตข้อมูล S_{max}+20 เพื่อให้ครอบคลุมทุกความผิดพลาด โดย S_{max} คือ จำนวนบิตข้อมูลที่มีการเจาะจงค่าที่มากที่สุดของชุดข้อมูลทดสอบ โครงสร้างการทำงานของ Original Static LFSR Reseeding แสดงดังรูปที่ 3-1

จากรูปที่ 3-1 ประกอบด้วย 2 ส่วนคือ ส่วนของตัวทดสอบ (Tester) และ ชิป ซึ่งวงจรกำเนิดข้อมูลการทดสอบที่ออกแบบให้สามารถกำเนิดข้อมูลทดสอบได้ ฝังอยู่ภายในชิป ประกอบด้วยวงจรกำเนิดข้อมูลทดสอบด้วยวิธี PRPG โดย LFSR วงจรเลื่อนเฟส และ วงจรที่จะทดสอบที่ถูกแบ่งให้อยู่ในรูปของ Scan Chain

จำนวนข้อมูล Seed ที่ใช้ในการกำเนิดข้อมูลทดสอบ 1 ชุดข้อมูลทดสอบนั้น จะเท่ากับขนาดความยาวของวงจร LFSR คือ r บิตด้วย แต่จำนวนข้อมูล Seed จะมีค่าน้อยกว่าความยาวของ Scan Chain (Scan length) คือ $r <$ Scan length ถ้าจำนวนชุดข้อมูลทดสอบทั้งหมดเท่ากับ L จำนวน Seed ทั้งหมดจึงเท่ากับ $r*L$ เพื่อให้สามารถตรวจจับความผิดพลาดได้ทุกจุด

ถึงแม้ว่าวิธีนี้จะทำให้ขนาดของข้อมูลการทดสอบลดลงแต่การกำเนิดข้อมูลทดสอบด้วยวิธี Original Static LFSR Reseeding จำเป็นต้องใช้ข้อมูล Seed เป็นจำนวนมาก เมื่อวงจรการทดสอบมีขนาดใหญ่ พื้นที่ในการเก็บข้อมูล Seed จึงจำเป็นต้องมีขนาดใหญ่ตามไปด้วย เป็นผลให้ต้องใช้ทรัพยากรในการสร้างวงจรจำนวนมาก

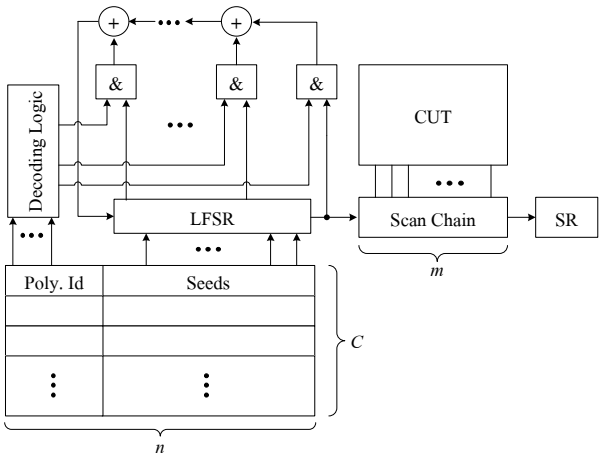


รูปที่ 3-1 Original Static LFSR Reseeding

3.2.2 MP-LFSRs Reseeding [27]

MP-LFSRs Reseeding [27] ได้นำเสนอวิธีการทดสอบด้วยวิธีการ Reseeding ที่พัฒนาวิธีการกำเนิดข้อมูลด้วยการใช้ Multiple Polynomial เพื่อลดจำนวนข้อมูล Seed ที่ใช้ในการกำเนิดข้อมูลทดสอบ ซึ่งทำให้จำนวนข้อมูล Seed ที่ใช้ในการกำเนิดข้อมูลทดสอบเพียง S_{max}+4 สถาปัตยกรรมของ MP-LFSRs Reseeding แสดงดังรูปที่ 3-2

รูปที่ 3-2 การกำเนิดข้อมูลทดสอบด้วยวิธีการ MP-LFSRs Reseeding ประกอบด้วยข้อมูล Seed ส่วน Pol.Id คือ ข้อมูลที่ได้จากการกำเนิดข้อมูลทดสอบของ LFSR ที่มีโพลิโนเมียลแบบดั้งเดิม ดีกรี k และวงจรถอดรหัส หรือ Decoding Logic โดยข้อมูล Seed จะถูกเลื่อนเข้าสู่วงจร LFSR ส่วน Pol.Id จะเลื่อนเข้าสู่วงจรถอดรหัส แล้ว Pol.Id จะถูกป้อนกลับเข้าสู่วงจร LFSR การกำเนิดชุดข้อมูลทดสอบจำนวน 1 ชุดข้อมูลทดสอบ ต้องใช้ข้อมูล n บิต ประกอบด้วย Pol.Id จำนวน q บิต และข้อมูล Seed จำนวน n-q บิต ลอจิก AND และ XOR เกต จะทำหน้าที่ป้อนกลับข้อมูล Seed เข้าสู่วงจร LFSR เพื่อกำเนิดข้อมูลทดสอบให้กับเซลล์ตรวจกวาดถัดไป เมื่อทำการเปรียบเทียบจำนวนข้อมูล Seed ที่ใช้ในการทดสอบกับวิธีการ Original Static LFSR Reseeding แล้ว จำนวนข้อมูลในการกำเนิดข้อมูลด้วยวิธีการ MP-LFSRs Reseeding ใช้จำนวนข้อมูล Seed น้อยกว่าวิธีการ Original Static LFSR Reseeding คือ $n< r$ เนื่องจากวิธีการ MP-LFSRs Reseeding ได้เพิ่มส่วนของ Pol.Id เพื่อให้ช่วยลดจำนวนข้อมูล Seed ทั้งมีการเพิ่มกลุ่ม XOR เกต และกลุ่มของ AND เกตเพื่อป้อนกลับข้อมูลทดสอบเข้าสู่วงจร LFSR จำนวนสัญญาณนาฬิกาที่ใช้ในสำหรับการกำเนิดข้อมูลทดสอบจำนวน 1 Scan Chain มีค่าเท่ากับความยาวของ Scan Chain เช่น Scan Chain มีความยาวเท่ากับ m เซลล์ จำนวนสัญญาณนาฬิกาที่ใช้ คือ m สัญญาณนาฬิกา



รูปที่ 3-2 โครงสร้าง Multi-Polynomial LFSR Reseeding [27]

การเข้ารหัสข้อมูลทดสอบด้วยวิธีการ MP-LFSRs Reseeding นี้ เป็นการปรับปรุงการกำเนิดข้อมูลการทดสอบจากวิธี Original Static LFSR Reseeding แต่ยังคงต้องใช้ข้อมูลทดสอบถึง S_{max}+4 อีกทั้งต้องใช้การเพิ่มวงจรการถอดรหัสเข้าสู่วงจรทดสอบ ซึ่งจะเป็นการเพิ่มทรัพยากรในการสร้างวงจรทดสอบ

3.2.3 Variable-length seeds [28]

วิธีการ Variable-length seed เป็นวิธีลดขนาดของข้อมูล Seed โดยการจัดระเบียบการเก็บข้อมูลทดสอบในหน่วยความจำ ซึ่งข้อมูล Seed ที่จัดเก็บจะต้องมีแนวโน้มที่จะเพิ่มสูงขึ้น การจัดรูปแบบโครงสร้างข้อมูลของ Variable-length seeds ประกอบด้วย 3 ตัวแปร คือ ขนาด (Size) จำนวนโพลิโนเมียล (Poly Id) และ ข้อมูล Seed

- ขนาด (Size) จะมีค่าเป็น “0” หรือ “1” เพื่อใช้อธิบายข้อมูล Seed ของชุดถัดไป เมื่อขนาดมีค่าเป็น 1 ข้อมูล Seed ชุดถัดไปจะมีค่าเพิ่มขึ้นเท่ากับ Δ ของข้อมูลชุดก่อนหน้า และเมื่อตัวแปรขนาดมีค่าเท่ากับ “0” จะไม่มีการเพิ่มขึ้นของข้อมูล Seed

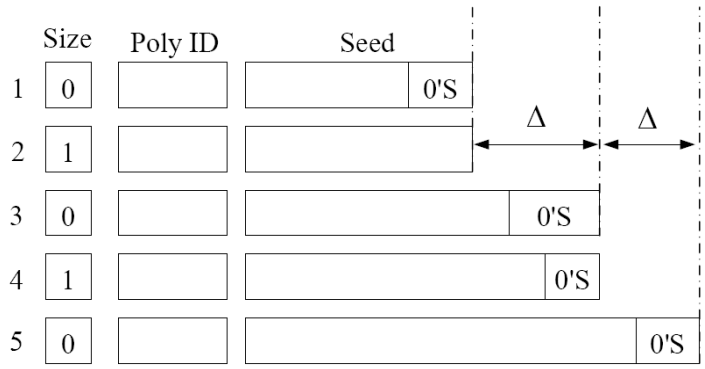
- จำนวนโพลิโนเมียล (Poly Id) จะเก็บข้อมูลจำนวนโพลิโนเมียลที่มีความสัมพันธ์กับข้อมูล Seed ซึ่งมีค่าเท่ากับ 2^q เมื่อ q คือจำนวนบิตของตัวแปรโพลิโนเมียล

- ข้อมูล Seed ที่ถูกเก็บในแต่ละชุดจะประกอบด้วย ตัวแปรพิเศษที่ใช้บอกความยาวของ Seed คือ 0’S และข้อมูล Seed เอง ซึ่งตัวแปรพิเศษ 0’S สามารถคำนวณได้ด้วยสมการ (3-1)

$$b+ i\Delta \qquad (3\text{-}1)$$

เมื่อ ตัวแปร *i* คือลำดับของข้อมูล 1, 2, 3, … ตัวแปร *b* คือจำนวนข้อมูลของ Seed ที่เล็กที่สุด ส่วน Δ คือความเปลี่ยนแปลงของจำนวนข้อมูล Seed ชุดถัดไป ซึ่งวิธีการจัดเก็บข้อมูลของ Variable-length seeds แสดงดังรูปที่ 3-3

จากรูปที่ 3-3 Variable-length seeds ประกอบด้วย ขนาด Poly Id และข้อมูล Seed ชุดข้อมูล Seed ที่ 1 มีขนาดเป็น “0” คือ ในชุดข้อมูลถัดไปจะไม่มีการจองพื้นที่เพื่อบอกการเข้ารหัสหรือ 0’S ส่วนในชุดข้อมูล Seed ที่ 2 มีขนาดเป็น 1 เพื่อบอกถึงจำนวนข้อมูล Seed มีการจองพื้นที่การเข้ารหัสของข้อมูล Seed ของชุดที่ 3 ซึ่งจะมีการเพิ่มขึ้นเท่ากับ Δ โดยข้อมูล Seed ชุดแรกจะมีขนาดเล็กที่สุดและชุดถัดไปจะมีขนาดเพิ่มขึ้นเรื่อยๆ ส่วน 0’S ที่ต่อท้ายข้อมูล Seed เป็นตัวแปรพิเศษเพื่อใช้คำนวณความยาวของข้อมูล Seed ชุดถัดไป



รูปที่ 3-3 Variable-length seed

ประสิทธิภาพของการเข้ารหัสข้อมูล Seed สามารถคำนวณได้จากสมการ

$\xi = \frac{s}{1+q+u+v}$ เมื่อ *s* คือจำนวนบิตที่มีการเจาะจงค่า

u คือจำนวนบิตที่ไม่มีการเจาะจงค่า

q คือค่าที่เกี่ยวเนื่องกับสมการโพลิโนเมียล

และ *v* คือค่าเฉลี่ยจำนวนบิตข้อมูล 0’S ต่อจำนวนชุดข้อมูลทดสอบ

ประสิทธิภาพของการเข้ารหัสข้อมูล Seed จะขึ้นอยู่กับค่าเฉลี่ยจำนวนบิตของจำนวนบิตข้อมูล 0’S ต่อจำนวนชุดข้อมูลทดสอบ เนื่องจาก จำนวนบิตที่ไม่มีการเจาะจงค่าและจำนวนบิตที่มีการเจาะจงค่าเป็นค่าคงที่ของแต่ละชุดข้อมูลทดสอบ ส่วนตัวแปร *q* จะค่าเป็น “0” เนื่องจากเป็นสมการโพลิโนเมียลแบบเดี่ยว (Single Polynomial) ดังนั้น เมื่อค่าเฉลี่ยจำนวนบิต

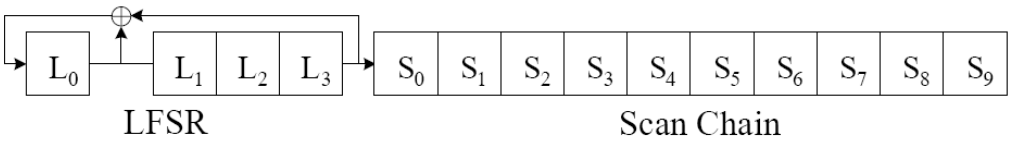
ข้อมูล 0'S ต่อจำนวนชุดข้อมูลทดสอบมีค่าสูง จะทำให้ประสิทธิภาพการเข้ารหัสข้อมูล Seed มีประสิทธิภาพน้อยลง

3.2.4 การจัดเรียงข้อมูลทดสอบด้วยวิธี Seed Ordering [29-30]

การเรียงลำดับข้อมูล Seed ด้วยวิธี Seed Ordering เป็นการเพิ่มจำนวนข้อมูลที่กำเนิดได้ด้วยข้อมูล Seed จำนวน 1 ชุด ซึ่งจะทำให้ข้อมูล Seed จำนวน 1 ชุด สามารถกำเนิดข้อมูลทดสอบได้หลายชุดของข้อมูลทดสอบ เมื่อจำนวนชุดข้อมูลทดสอบลดลง จำนวนการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR ก็ลดลงด้วยเช่นกัน ซึ่งทำให้ลดจำนวนของฮาร์ดแวร์ในการพัฒนา และจำนวนหน่วยความจำในการเก็บข้อมูล Seed

วิธีการกำเนิดข้อมูลทดสอบด้วยวิธี Seed Ordering เริ่มจาก เลื่อนบิตข้อมูล Seed เพื่อตั้งค่าเริ่มต้นให้กับวงจร LFSR โดยจำนวนบิตข้อมูล Seed ที่ใช้ในการตั้งค่าเริ่มต้นจะเท่ากับขนาดของวงจร LFSR และเพื่อกำเนิดข้อมูลทดสอบให้ครบทุกบิตของ Scan Chain จำนวนสัญญาณนาฬิกาที่ใช้จึงเท่ากับจำนวนความยาวของ Scan Chain เมื่อต้องการกำเนิดข้อมูลทดสอบของชุดข้อมูลทดสอบชุดถัดไป ก็จะไม่มีการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR แต่จะใช้ข้อมูล Seed ที่เลื่อนไปก่อนหน้ามาทำการกำเนิดข้อมูลทดสอบให้กับชุดข้อมูลทดสอบปัจจุบัน โดยการใช้ข้อมูล Seed ที่เป็นตัวแปรอิสระ (free variable) โดยที่ตัวแปรอิสระ คือ ตัวแปรของข้อมูล Seed ที่เลื่อนเข้าสู่วงจร LFSR แล้วไม่มีผลต่อสมการ กล่าวคือ เมื่อคำนวณด้วยวิธี Gausses Elimination แล้วไม่มี pivot ที่ตำแหน่งนั้น นั่นเอง

รูปที่ 3-4 การกำเนิดข้อมูลทดสอบด้วยวิธี LFSR ขนาด 4 บิต มีสมการโพลิโนเมียล เป็น $F(x) = X^4+X^3+1$ แทนด้วยตัวแปร L_0-L_3 กำหนดให้ ลำดับการกำเนิดข้อมูลการทดสอบวงจร LFSR ที่ใช้ค่าเริ่มต้นเป็น "1000" เพื่อกำเนิดข้อมูลทดสอบจำนวน 10 เซลล์ แทนด้วยตัวแปร S_0-S_9 ในการกำเนิดข้อมูลทดสอบจำนวนสัญญาณนาฬิกาที่ต้องใช้ในการเลื่อนบิตข้อมูลทดสอบเท่ากับความยาวของเซลล์ตรวจกวาด คือ 10 สัญญาณนาฬิกา สมการที่อธิบายการกำเนิดข้อมูลการทดสอบแสดงดังสมการ (3-2) ซึ่งเขียนอยู่ในรูปของ Matrix



รูปที่ 3-4 วงจร LFSR ขนาด 4 บิต กำเนิดข้อมูลทดสอบ 10 เซลล์

สมการ (3-2) Matrix อธิบายการกำเนิดข้อมูลเซลล์ตรวจกวาดที่ S_0-S_9 โดยในแนวแนวสดมภ์แสดงตัวแปร L_0-L_3 ส่วนแนวแถวแสดงสมการอธิบายสัมพันธ์ของตัวแปร L_0-L_3 ที่ใช้ในการกำเนิดข้อมูลทดสอบของเซลล์ตรวจกวาดที่ S_0-S_9 เช่น ในการกำเนิดข้อมูลทดสอบของ S_0 เกิดจาก $L_1 \oplus L_3$ เป็นต้น

$$E = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} S_0 \\ S_1 \\ S_2 \\ S_3 \\ S_4 \\ S_5 \\ S_6 \\ S_7 \\ S_8 \\ S_9 \end{matrix} \\ \begin{matrix} L_0 & L_1 & L_2 & L_3 \end{matrix} \tag{3-2}$$

จากสมการ (3-2) Matrix อธิบายการกำเนิดข้อมูลเซลล์ตรวจกวาดที่ S_0-S_9 ถ้าข้อมูลทดสอบมีค่าเป็น “X0X1X10XXX” และ “0XXX1XXXXX” การกำเนิดข้อมูลทดสอบในชุดแรกสามารถคำนวณได้จาก การนำข้อมูล Seed มาคำนวณหาค่าของตัวแปร L_0-L_3 โดยสมการที่นำมาคำนวณคือ สมการที่ต้องการการเจาะจงค่าคือตำแหน่งของเซลล์ตรวจกวาดที่ S_1, S_3, S_5 และ S_6 ซึ่ง Matrix ที่อธิบายความสัมพันธ์ระหว่างข้อมูล Seed และการกำเนิดข้อมูลเซลล์ตรวจกวาดที่ S_1, S_3, S_5 และ S_6 แสดงดังสมการ (3-3) ส่วนสมการที่ไม่มีการเจาะจงค่าของข้อมูลทดสอบชุดที่ 2 คือตำแหน่งของเซลล์ตรวจกวาดที่ S_0 และ S_4 ซึ่ง Matrix ที่อธิบายความสัมพันธ์ระหว่างข้อมูล Seed และการกำเนิดข้อมูล เซลล์ตรวจกวาดที่ S_0 และ S_4 แสดงดังสมการ (3-4)

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} L_0 \\ L_1 \\ L_2 \\ L_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{3-3}$$

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} L_0 \\ L_1 \\ L_2 \\ L_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{3-4}$$

จากสมการ (3-3) และ (3-4) Matrix อธิบายความสัมพันธ์ระหว่างข้อมูล Seed และการกำเนิดข้อมูล ที่ S_1, S_3, S_5 และ S_6 ของข้อมูลทดสอบชุดที่ 1 และเซลล์ตรวจกวาดที่ S_0 และ S_4 ของข้อมูลทดสอบชุดที่ 2 มาคำนวณด้วย Gausses elimination แล้วจะทำให้ได้คำตอบของ L_1-L_3 เป็น “1000” เพื่อกำเนิดข้อมูลทดสอบของชุดที่ 1 และ 2 โดยที่ไม่จำเป็นต้องเลื่อนบิตข้อมูล Seed เพื่อกำเนิดข้อมูลเพื่อทดสอบชุดที่ 2

วิธีการกำเนิดข้อมูลทดสอบด้วยวิธีการเรียงลำดับข้อมูล Seed ด้วยวิธี Seed ordering สามารถลดจำนวนข้อมูลทดสอบได้ แต่จำเป็นต้องใช้ระยะเวลานานในการทดสอบ เนื่องจากในการกำเนิดข้อมูลการทดสอบของชุดที่ 2 จำเป็นต้องใช้เวลาในการคำนวณถึงสองครั้ง สำหรับกรณีที่เลวร้ายที่สุด เนื่องจากในครั้งแรกจะคำนวณจากการเลื่อนข้อมูล Seed ชุดที่หนึ่ง ถ้าข้อมูล Seed ชุดที่หนึ่ง ไม่สามารถกำเนิดข้อมูลการทดสอบได้ครอบคลุม ก็จำเป็นต้องเลื่อนข้อมูล Seed ชุดที่สองเข้าสู่วงจร LFSR และก็จะทำการคำนวณอีกครั้ง ซึ่งวิธีการนี้ ทำให้จำเป็นต้องใช้เวลาในการทดสอบที่ยาวนาน

3.2.5 วิธีกำเนิดข้อมูลทดสอบด้วย LFSR Reseeding ร่วมกับวิธีการอื่น

การรวมวิธีกำเนิดข้อมูลทดสอบด้วย LFSR Reseeding และ วิธีการอื่นนั้น มีจุดประสงค์เพื่อเพิ่มประสิทธิภาพของการกำเนิดข้อมูลการทดสอบ โดยที่เน้นการลดจำนวนข้อมูล Seed และการบีบอัดข้อมูลทดสอบ เช่นการรวมวิธีทดสอบด้วย LFSR Reseeding กับ Bit-Fixing หรือ วิธีพจนานุกรม หรือ Dictionary-base ไม่ว่าจะเป็นบนพื้นฐานของฮาร์ดแวร์หรือว่าซอฟแวร์

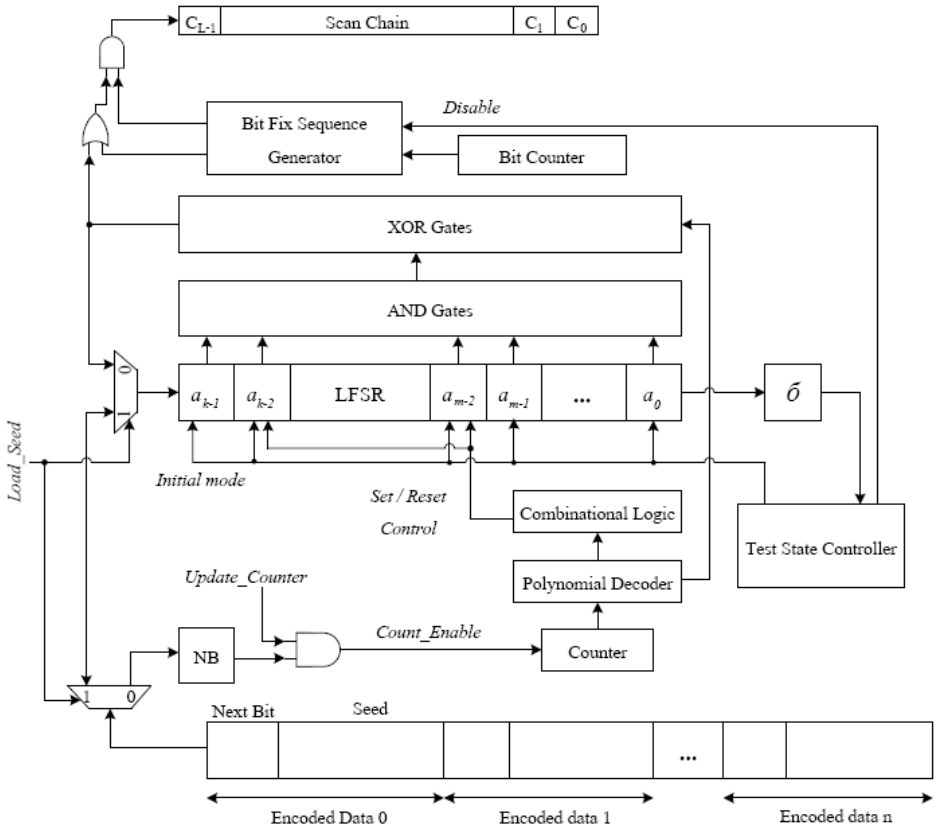
3.2.5.1 LFSR Reseeding และ Bit-Fixing [31]

การกำเนิดข้อมูลทดสอบด้วยวิธีการ LFSR Reseeding และ Bit-Fixing นั้น ใช้การประยุกต์การเปลี่ยนแปลงข้อมูลบางบิตหรือบางตำแหน่ง เพื่อให้ครอบคลุมความผิดพลาดที่ตำแหน่งนั้นๆ ร่วมกับการกำเนิดข้อมูลทดสอบด้วยวิธีการ MP-LFSR Reseeding ซึ่งแนวคิดการทำงานร่วมกันระหว่าง LFSR Reseeding และ Bit-Fixing แสดงดังรูปที่ 3-5

จากรูปที่ 3-5 สถาปัตยกรรมการรวมกันของ LFSR Reseeding และ Bit-Fixing ประกอบด้วย 4 ส่วนหลักคือ

- Variable Polynomial LFSR หรือ VD-LFSR
- Polynomial decoder
- Bit Fixing Sequence Generator

- Test State Controller หรือ TSC



รูปที่ 3-5 การรวมกันของ LFSR Reseeding และ Bit-Fixing

ทุกส่วนของวงจรกำเนิดข้อมูลทดสอบถูกควบคุมการทำงานด้วย TSC เพื่อให้ทั้งหมดทำงานประสานกัน โดยข้อมูลทดสอบจาก VD-LFSR จะถูกเปลี่ยนแปลงข้อมูลในบางตำแหน่งหรือการ Fixing Bit ซึ่งตำแหน่งที่เกิดการเปลี่ยนแปลงจะตรงกับตำแหน่งของ Bit Fixing Sequence Generator ส่วนของโพลิโนเมียลจะถูกถอดรหัสด้วยข้อมูล Seed ที่อยู่ในกลุ่มเดียวกันกับโพลิโนเมียล ในกระบวนการถอดรหัสของบิตถัดไป มีตัวชี้หรือ Pointer ระบุตำแหน่งการเปลี่ยนกลุ่มของโพลิโนเมียล ให้สามารถเลื่อนข้อมูล Seed ตำแหน่งที่ถูกต้องต่อการเปลี่ยนดีกรีของโพลิ-โนเมียลในแต่ละชุด จะส่งผลต่อความยาวของ Seed ที่ใช้ในการเข้ารหัส ส่วนจำนวนบิตของข้อมูล Seed จะส่งผลต่อจำนวนสัญญาณนาฬิกาที่ใช้ในการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR และจำนวนการป้อนกลับของโพลิโนเมียล ในส่วนของการถอดรหัสโพลิโนเมียล หรือ Polynomial Decoder ทำหน้าที่จัดเก็บบิตข้อมูล ที่มีความสำคัญมาก และบิตที่มีความสำคัญน้อย ของวงจร LFSR จำนวนอย่างละ 2 บิต เพื่อใช้ในการคำนวณจำนวนสัญญาณนาฬิกาในการเลื่อนข้อมูล Seed เข้าสู่วงจร

LFSR เช่น ถ้าค่าความสำคัญน้อยของวงจร LFSR มีค่าเป็น 1 จำนวนสัญญาณนาฬิกาที่ต้องใช้ จึงเท่ากับ m ครั้ง เพื่อเลื่อนข้อมูล Seed จำนวน m บิต ซึ่งเอาท์พุทของ polynomial decoder ที่ i เท่ากับ P_0, P_1,..., P_{m-2}, 1, 0 ,...,0 ส่วนการเก็บข้อมูลค่าความสำคัญน้อยของ polynomial decoder จะใช้สัญญาณนาฬิกาเพื่อเลื่อนข้อมูล Seed จำนวน k-m+1 ครั้ง โดยตัวแปร k คือดีกรีของสมการโพลิโนเมียล ซึ่งข้อมูลค่าความสำคัญน้อยของ LFSR จะใช้เพื่อตั้งค่าเริ่มต้นใหม่ให้กับ LFSR แต่ถ้าข้อมูลค่าความสำคัญน้อยของ LFSR มีค่าเท่ากับ m-1 บิต จะเป็นการตั้งค่าเริ่มต้นให้กับ LFSR และ δ-register

การรวมกันของการกำเนิดข้อมูลทดสอบแบบ LFSR Reseeding และ Bit-Fixing เป็นวิธีการที่เน้นการลดจำนวนข้อมูลการทดสอบ แต่เป็นการเพิ่มฮาร์ดแวร์ที่ใช้ในการเปลี่ยนแปลงบิตข้อมูลทดสอบ ซึ่งอาจจะส่งผลให้ลดประสิทธิภาพการทำงานของวงจรทดสอบ [16]

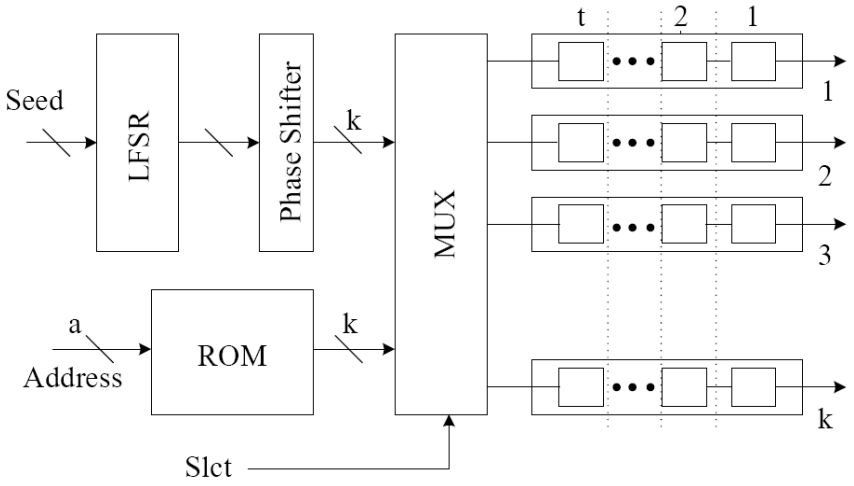
3.2.5.2 LFSR Reseeding ร่วมกับวิธีพจนานุกรม [32-33]

วิธีกำเนิดข้อมูลการทดสอบด้วย LFSR Reseeding ร่วมกับพจนานุกรมเป็นการเพิ่มประสิทธิภาพการกำเนิดข้อมูลทดสอบให้กับวงจรที่มีจำนวนบิตการเจาะจงค่าจำนวนมาก และวิธีการกำเนิดข้อมูลแบบ LFSR Reseeding ยังมีข้อจำกัดในการกำเนิดข้อมูลให้ครอบคลุมความผิดพลาด 100 เปอร์เซ็นต์ จำเป็นต้องใช้จำนวนข้อมูล Seed จำนวนมาก ในเอกสารอ้างอิงที่ 32 และ 33 จึงได้เสนอวิธีพจนานุกรม มาช่วยในการกำเนิดข้อมูลทดสอบสำหรับชุดวงจรทดสอบที่มีการเจาะจงค่าจำนวนมาก ซึ่งสถาปัตยกรรมพื้นฐานของ LFSR Reseeding และพจนานุกรมแสดงดังรูปที่ 3-6

จากรูปที่ 3-6 ประกอบด้วยสองส่วนคือ วงจรกำเนิดข้อมูลทดสอบด้วยวิธี LFSR Reseeding ที่ประกอบด้วยข้อมูล Seed วงจร LFSR และวงจรเลื่อนเฟส ส่วนหน่วยความจำในการเก็บข้อมูล (ROM) เป็นโครงสร้างในส่วนของพจนานุกรม ซึ่งการทำงานของวงจรทั้งสองส่วนจะทำงานประสานกันด้วย multiplexer หรือ MUX และใช้สัญญาณ Slct ในการเลือกวิธีกำเนิดข้อมูลทดสอบ ในการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR ผ่านวงจรเลื่อนเฟสจะต้องใช้จำนวน k ชุด เท่ากับจำนวนของเซลล์ตรวจกวาด ในส่วนของหน่วยความจำสำหรับพจนานุกรมจะจัดเก็บข้อมูลจำนวน a บิต เพื่อกำเนิดข้อมูลทดสอบจำนวน 1 ชุด โดยแต่ละชุดข้อมูลทดสอบจะต้องใช้ t สัญญาณนาฬิกาเพื่อเลื่อนบิตให้เต็มเซลล์ตรวจกวาด

การเลือกวิธีกำเนิดข้อมูลทดสอบระหว่าง LFSR Reseeding และวิธีพจนานุกรม ใช้การเลือกจากจำนวนบิตการเจาะจงค่าของแต่ละชุดข้อมูลทดสอบ เมื่อชุดข้อมูลทดสอบใดมีจำนวน

การเจาะจงค่ามากก็จะเป็น Restrict Vector จึงจะใช้พจนานุกรมช่วยในการกำเนิดทดสอบชุดข้อมูลดังกล่าว ซึ่งเรียกข้อมูลที่กำเนิดจากพจนานุกรม ว่า Dummyword



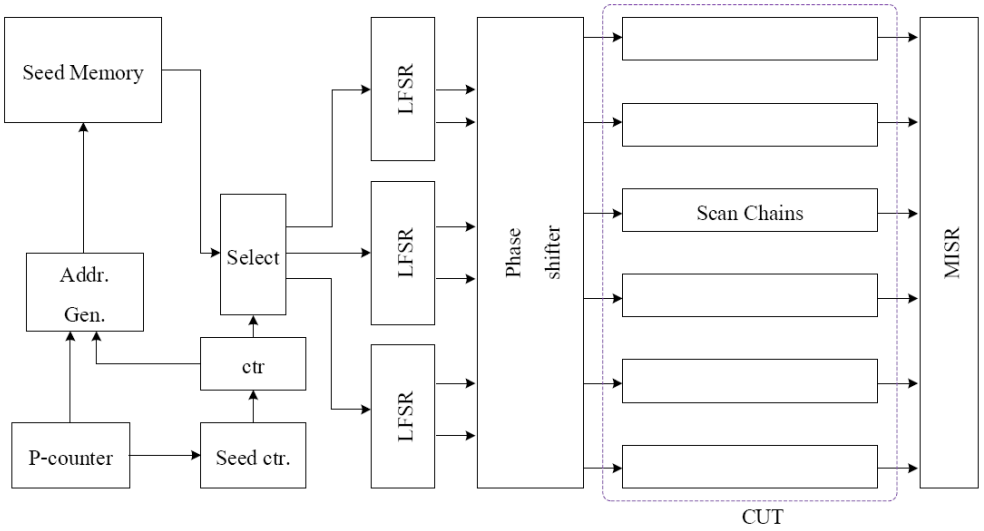
รูปที่ 3-6 การรวมกันของ LFSR Reseeding และ Dictionary

3.2.5.3 LFSR Reseeding และ วิธีการพจนานุกรมบนพื้นฐานของซอฟแวร์ [34]

วิธีการกำเนิดข้อมูลการทดสอบแบบ LFSR Reseeding ร่วมกับพจนานุกรมบนพื้นฐานของซอฟแวร์ จะไม่เน้นการเก็บข้อมูลในหน่วยความจำแบบ ROM แต่พัฒนาบนพื้นฐานของซอฟแวร์ จากรูปที่ 3-7 แสดงสถาปัตยกรรมของ LFSR Reseeding และพจนานุกรมบนพื้นฐานของซอฟแวร์ ประกอบด้วย Multiple LFSR วงจรเลื่อนเฟส และข้อมูล Seed ซึ่งเป็นส่วน LFSR Reseeding และส่วนพจนานุกรมที่ออกแบบให้อยู่ในรูปแบบของซอฟแวร์ ส่วนของวงจร LFSR Reseeding ประกอบด้วย หน่วยความจำข้อมูล Seed ที่ถูกจัดเก็บในพจนานุกรม หน่วยความจำข้อมูล Seed หรือ Memory Seed วงจรกำเนิดตำแหน่งของหน่วยความจำหรือ Addr.Gen วงจรนับข้อมูล Seed หรือ seed cnt และวงจรนับ หรือ p-counter โดย Addr.Gen ทำหน้าที่กำเนิดตำแหน่งของข้อมูล Seed เพื่อให้สอดคล้องกับการข้อมูล Seed ของแต่ละชุดข้อมูลทดสอบ วงจรนับข้อมูล Seed จะทำหน้าที่เก็บจำนวน LFSR ที่จะทำหน้าที่ใช้ในการโหลดข้อมูล Seed ในแต่ละชุดข้อมูลทดสอบ ส่วนวงจรนับทำหน้าที่เป็นตัวระบุชุดข้อมูลทดสอบปัจจุบันเพื่อให้การทำงานของ วงจรกำเนิดตำแหน่งของหน่วยความจำ วงจรนับข้อมูล Seed และ LFSR ทำงานประสานกัน

โดยวิธีการนี้ใช้การกำเนิดข้อมูลการทดสอบด้วย MP-LFSR ซึ่งจำเป็นต้องใช้ข้อมูล Seed จำนวนมาก ให้สามารถลดขนาดของข้อมูล Seed ลง การกำหนดจำนวนบิตข้อมูล Seed ที่จะถูกเข้ารหัสก็ต่อเมื่อข้อมูล Seed มีค่าเท่ากับ หรือมากกว่าค่า D คือข้อมูล Seed ที่มีค่าตั้งแต่ D,

D+1, *D*+2… เป็นต้นไป จะถูกนำไปเข้ารหัส ส่วนข้อมูล Seed ที่มีค่าน้อยกว่า *D* จะไม่ถูกจัดเก็บในหน่วยความจำ Seed เพื่อลดพื้นที่ในการการจัดเก็บข้อมูล Seed ลง



รูปที่ 3-7 การรวมกันของ LFSR Reseeding และ Dictionary base Code

วิธีการกำเนิดข้อมูลทดสอบด้วยวิธี LFSR Reseeding ร่วมกับพจนานุกรม ไม่ว่าจะบนพื้นฐานของฮาร์ดแวร์หรือซอฟแวร์ เป็นวิธีการที่เพิ่มประสิทธิภาพการกำเนิดข้อมูลทดสอบ แต่จำเป็นต้องเพิ่มพื้นที่จัดเก็บ Dummyword และวงจรกำเนิดตำแหน่งของ Dummyword ซึ่งจะเป็นการเสี่ยงที่จะใช้พรัพยากรจำนวนมาก

3.2.6 สรุปวิธีการกำเนิดข้อมูลแบบ Static LFSR Reseeding

วิธีการกำเนิดข้อมูลการทดสอบแบบ Static LFSR Reseeding ด้วยวิธีการที่ได้นำเสนอคือ วิธีการ Original LFSR Reseeding, MP-LFSRs, Variable-Length Seed, การจัดเรียงข้อมูล Seed, LFSR Reseeding ร่วมกับพจนานุกรม และ LFSR Reseeding ร่วมกับ bit fixing มาทำการวิเคราะห์ข้อดี และข้อเสียของการกำเนิดข้อมูลการทดสอบแต่ละวิธี โดยการเปรียบเทียบใน 2 ปัจจัย คือจำนวนการข้อมูลทดสอบ และการเพิ่มวงจรพิเศษต่างๆ โดยได้นำเสนอดังตารางที่ 3-1

การพิจารณาถึงจำนวนข้อมูลทดสอบจากตารางที่ 3-1 จะเห็นได้ว่าวิธีการทดสอบด้วยวิธีการ Original LFSR Reseeding และ MP-LFSRs เท่านั้น ที่นำเสนอจำนวนข้อมูล Seed ในการกำเนิดข้อมูลทดสอบ ซึ่งเท่ากับ S_{max}+20 และ S_{max}+4 ต่อ 1 ชุดข้อมูลทดสอบ ตามลำดับ ซึ่งจะเห็นได้ว่าจำเป็นต้องใช้จำนวนข้อมูลทดสอบจำนวนมาก เมื่อจำนวนเจาะจงค่าจำนวนมาก แม้

วิธีการ MP-LFSRs นำเสนอวิธีการลดจำนวนข้อมูลทดสอบจากวิธีการ Original LFSR Reseeding แต่จำเป็นต้องเพิ่มวงจร Decoder ขึ้น เพื่อใช้ในการถอดรหัสข้อมูลจากสมการโพลิโนเมียล ส่วนวิธีการ Variable-Length Seed, วิธีการจัดเรียงข้อมูล Seed, การ LFSR Reseeding ร่วมกับพจนานุกรม, LFSR Reseeding ร่วมกับ bit fixing ได้เสนอวิธีจัดการกับข้อมูล Seed ที่ถูกจัดเก็บในหน่วยความจำ ซึ่งจำเป็นต้องเพิ่มวงจรพิเศษ เช่น การเพิ่มวงจรถอดรหัส วงจรเข้ารหัส หรือวงจรกำเนิดตำแหน่ง เป็นต้น จะเป็นผลให้สิ้นเปลืองทรัพยากรในการสร้างวงจรทดสอบ วิธีการจัดเรียงข้อมูล Seed และวิธีการ Original LFSR Reseeding เป็นวิธีการกำเนิดข้อมูลที่ไม่จำเป็นต้องเพิ่มวงจรพิเศษ สำหรับวิธีการจัดเรียงข้อมูล Seed แม้ว่าจะไม่มีการเพิ่มจำนวนวงจรพิเศษ แต่จะเป็นการเพิ่มระยะเวลาการทดสอบ รวมถึงเพิ่มความยุ่งยากในการประมวลผลของหน่วยประมวลผลแบบ BIST

ตารางที่ 3-1 การเปรียบเทียบการกำเนิดข้อมูลการทดสอบด้วยวิธี Static LFSR Reseeding

วิธีการกำเนิดข้อมูล	จำนวนข้อมูลทดสอบ	การเพิ่มวงจรพิเศษ
Original LFSR Reseeding	S_{max}+20	ไม่มี
MP-LFSRs	S_{max}+4	มี
Variable-Length Seed	-	มี
การจัดเรียงข้อมูล Seed	-	ไม่มี
LFSR Reseeding ร่วมกับพจนานุกรม	-	มี
LFSR Reseeding ร่วมกับ bit fixing	-	มี

นอกจากนี้การกำเนิดข้อมูลทดสอบด้วยวิธีการ Static LFSR Reseeding ยังมีข้อเสียอีกหนึ่งอย่างก็คือ ใช้เวลานานในการทดสอบ เนื่องจากขณะการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR จะต้องหยุดกระบวนการกำเนิดข้อมูลการทดสอบ ทั้งการเลื่อนข้อมูลทดสอบเข้าสู่วงจรทดสอบในรูปแบบอนุกรม ด้วยเหตุนี้จึงได้มีการนำเสนอวิธีการกำเนิดข้อมูลการทดสอบแบบ Dynamic LFSR Reseeding เพื่อแก้ปัญหาเรื่องเวลาในการทดสอบ ทั้งลดจำนวนข้อมูล Seed ให้น้อยกว่า S_{max}+4

3.3 Dynamic LFSR Reseeding

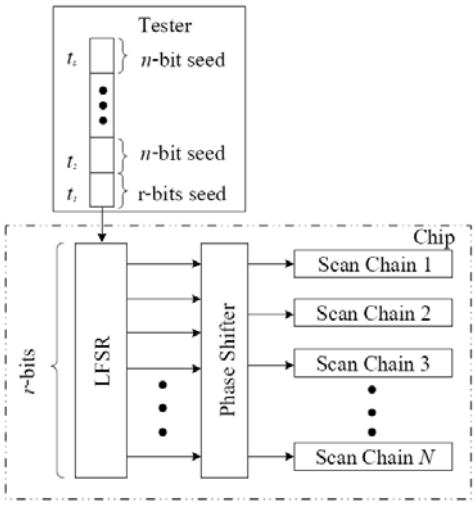
การกำเนิดข้อมูล Dynamic LFSR Reseeding เป็นการกำเนิดข้อมูลการทดสอบที่ใช้การป้อนตัวแปรอิสระเข้าสู่วงจร LFSR ผ่านวงจรเลื่อนเฟสเพื่อกำเนิดข้อมูลให้วงจรที่จะทดสอบ [15] ซึ่งวิธีการกำเนิดข้อมูลการทดสอบแบบ Dynamic LFSR Reseeding สามารถลดระยะเวลาการทดสอบได้ เนื่องจากสามารถกำเนิดข้อมูลการทดสอบได้ในขณะที่มีการเลื่อนข้อมูล Seed เข้าสู่วงจรทดสอบ อีกทั้งยังสามารถลดจำนวนข้อมูลการทดสอบได้ โดยการเลื่อนข้อมูล Seed เพียงบางส่วน และยืดหยุ่นต่อการประยุกต์ใช้กับวงจรที่จะทดสอบแบบต่างๆ การกำเนิดข้อมูลแบบ Dynamic LFSR Reseeding เช่น วิธีการ Partial LFSR Reseeding และ Relaxation เป็นต้น

3.3.1 Partial LFSR Reseeding [1][44]

การกำเนิดข้อมูลแบบ Partial LFSR Reseeding [1][44] เป็นวิธีการหนึ่งที่ใช้การลดขนาดข้อมูล Seed ในการกำเนิดข้อมูลทดสอบลง โดยขนาดของข้อมูล Seed จะขึ้นอยู่กับค่าเฉลี่ยของการเจาะจงค่า หรือ S_{avg} และเพิ่มลอจิก XOR เกตระหว่างวงจร LFSR และข้อมูล Seed ที่จะถูกเลื่อนจากผู้ทดสอบ (Tester) เพื่อให้ข้อมูล Seed ที่ถูกเลื่อนไปก่อนหน้า เกิดการป้อนกลับ มีผลกลับมาในสัญญาณนาฬิกาถัดไป การกำเนิดข้อมูลแบบ Partial LFSR Reseeding แสดงดังรูปที่ 3-8

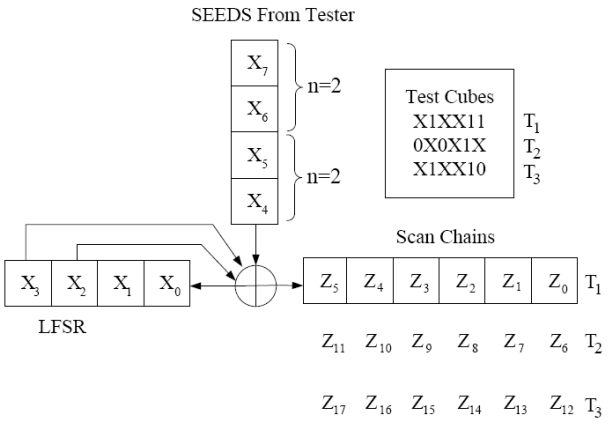
จากรูปที่ 3-8 ประกอบด้วยวงจร LFSR ขนาด r บิต, วงจรเลื่อนเฟส, XOR เกต, ชุดข้อมูล Seed ถูกจัดเก็บในผู้ทดสอบ และ Scan Chain จำนวน N ชุด โดยจำนวนชุดของข้อมูล Seed จะมีจำนวนเท่ากับจำนวนชุดของ Scan Chain คือ N ชุด ซึ่งข้อมูล Seed ชุดแรกจะมีขนาดเท่ากับขนาดของวงจร LFSR คือ r บิต เพื่อใช้ในการตั้งค่าเริ่มต้นให้กับวงจร LFSR ส่วนข้อมูล Seed ชุดถัดไปจะมีขนาดเท่ากับ n บิต โดยที่ข้อมูล Seed ที่ใช้ในการตั้งค่าเริ่มต้นจะมีขนาดใหญ่กว่าชุดข้อมูล Seed ที่จะใช้ในการกำเนิดข้อมูลทดสอบชุดอื่นๆ คือ $r>n$

วิธีการเลื่อนบิตข้อมูล Seed จากผู้ทดสอบด้วยวิธี Partial LFSR Reseeding ใช้การเลื่อนบิตข้อมูลเป็นลำดับ (Serial) กล่าวคือข้อมูลที่ใช้ในการกำเนิดข้อมูลทดสอบชุดแรก จะถูกเลื่อนเข้าสู่วงจร LFSR ที่ละบิตจนครบทั้ง r บิต เพื่อที่จะตั้งค่าเริ่มต้นให้กับวงจร LFSR และกำเนิดข้อมูลทดสอบให้กับ Scan Chain ที่ 1 ส่วนข้อมูล Seed ชุดที่ 2 จะถูกเลื่อนเข้าสู่วงจร LFSR หลังจากข้อมูลสามารถกำเนิดข้อมูลทดสอบให้กับ Scan Chain ที่ 1 ครบทุกบิต โดยข้อมูล Seed ชุดที่ 1 จะถูกป้อนกลับเพื่อกำเนิดข้อมูลให้กับ Scan Chain ที่ 2 ด้วย XOR เกตที่อยู่ระหว่างวงจร LFSR และผู้ทดสอบ ตัวอย่างการสร้างสมการการกำเนิดข้อมูลทดสอบด้วยวิธี Partial LFSR Reseeding แสดงดังรูปที่ 3-9



รูปที่ 3-8 Partial LFSR Reseeding

จากรูปที่ 3-9 ตัวอย่างการสร้างสมการของวิธีการ Partial LFSR Reseeding จากรูปประกอบไปด้วยวงจร LFSR ขนาด 4 บิต ข้อมูล Seed ที่จะถูกเลื่อนแต่ละชุดมีค่า $n = 2$ บิต คือ X_4 ถึง X_7 ข้อมูล Seed สำหรับตั้งค่าเริ่มต้นให้กับวงจร LFSR มีค่า $r = 4$ คือ X_0 ถึง X_3 เพื่อที่ใช้กำเนิดข้อมูลทดสอบจำนวน 3 ชุด T_1 ถึง T_3 ชุดละ 6 บิต คือ “X1XX11”, “0X0X1X”, “X1XX10” ข้อมูลทดสอบทั้ง 3 ชุดแทนค่าด้วยตัวแปร Z_0 ถึง Z_{17} ถูกจัดเรียงเป็น 3 Scan Chain



รูปที่ 3-9 ตัวอย่างการสร้างสมการของ Partial LFSR Reseeding

การกำเนิดข้อมูลทดสอบสำหรับ T_1 จะไม่มีการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แต่ใช้การกำเนิดข้อมูลทดสอบจากค่าตั้งต้นของวงจร คือ X_0 ถึง X_3 เพื่อกำเนิดข้อมูลทดสอบให้กับ

Z_0 ถึง Z_5 ซึ่งสมการกำเนิดข้อมูลทดสอบแสดงดังสมการ (3-5) สมการอธิบาย Z_0 ถึง Z_5 เกิดจากกลไกของวงจร LFSR คือเกิดจากการ XOR กันของค่าตั้งต้น เช่น Z_0 เกิดจากการ XOR กันของ X_2 และ X_3

$$\begin{aligned} Z_0 &= X_2 \oplus X_3 \\ Z_1 &= X_1 \oplus X_2 \\ Z_2 &= X_0 \oplus X_1 \\ Z_3 &= X_0 \oplus X_2 \oplus X_3 \\ Z_4 &= X_1 \oplus X_3 \\ Z_5 &= X_0 \oplus X_2 \end{aligned} \tag{3-5}$$

การกำเนิดข้อมูลทดสอบสำหรับ T_2 จะต้องเลื่อนบิตข้อมูล Seed คือ X_4 และ X_5 เข้าสู่วงจร LFSR โดยสมการกำเนิดข้อมูลทดสอบ Z_6 เกิดจากการ XOR กันของกำเนิดข้อมูลทดสอบสำหรับ T_1 และ X_4 ที่ถูกเลื่อนเข้าสู่วงจร LFSR สมการกำเนิดข้อมูลทดสอบ Z_7 เกิดจากการ XOR กันของกำเนิดข้อมูลทดสอบสำหรับ T_1 และ X_5 แต่สมการกำเนิดข้อมูลทดสอบ Z_8 ถึง Z_{11} จะไม่มีการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR จะใช้กลไกของวงจร LFSR เพื่อกำเนิดข้อมูลทดสอบซึ่งข้อมูลชุดดังกล่าวจะมีผลต่อเซลล์ตรวจกวาดที่ Z_6 ถึง Z_{11} ดังสมการ (3-6)

$$\begin{aligned} Z_6 &= X_1 \oplus X_2 \oplus X_3 \oplus X_4 \\ Z_7 &= X_0 \oplus X_1 \oplus X_2 \oplus X_5 \\ Z_8 &= X_0 \oplus X_1 \oplus X_2 \oplus X_3 \\ Z_9 &= X_0 \oplus X_1 \oplus X_3 \oplus X_4 \\ Z_{10} &= X_0 \oplus X_3 \oplus S_1 \oplus X_5 \\ Z_{11} &= X_3 \oplus X_5 \end{aligned} \tag{3-6}$$

การกำเนิดข้อมูลทดสอบของ T_3 จะเป็นการเลื่อนบิตข้อมูล Seed คือ X_5 และ X_6 เข้าสู่วงจร LFSR แล้ว XOR กับข้อมูลของวงจร LFSR ซึ่งจะมีผลต่อเซลล์ตรวจกวาดที่ Z_{12} ถึง Z_{17} แสดงดังสมการ (3-7)

$$\begin{aligned} Z_{12} &= X_1 \oplus X_4 \oplus X_6 \\ Z_{13} &= X_1 \oplus X_5 \oplus X_7 \\ Z_{14} &= X_0 \oplus X_4 \\ Z_{15} &= X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6 \\ Z_{16} &= X_1 \oplus X_2 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7 \\ Z_{17} &= X_0 \oplus X_1 \oplus X_4 \oplus X_5 \oplus X_7 \end{aligned} \tag{3-7}$$

การกำเนิดข้อมูลทดสอบด้วยวิธี Partial LFSR Reseeding ใช้การเลื่อนบิตข้อมูลจากเข้าสู่วงจร LFSR แบบลำดับ ทำให้ข้อมูลที่จะถูกเลื่อนไปยัง Scan Chain เกิดการขัดแย้งกัน คือ ทำให้มีบางชุดของข้อมูลทดสอบที่ทำให้สมการไม่เป็นจริง เมื่อพิจารณาจากสมการ (3-6) โดยพิจารณาเฉพาะบิตมีค่าเป็น '1' โดย LFSR มีค่าเท่ากับ "1000" จะได้ว่า $X_1 = 1, X_2 = 0, X_2 = 0, X_3 = 0$ เพื่อให้ $Z_6 = 1$ จึงทำให้ $X_4 = 1$ เท่านั้น ที่ $Z_7 = 1$ ก็ต่อเมื่อ $X_5 = 0$ เมื่อพิจารณาที่ Z_9 จะ เกิดการขัดแย้งกันเนื่องจาก $X_0 = 1$ และ $X_4 = 1$ จึงทำให้ Z_9 มีค่าเท่ากับ "0" ไม่มีโอกาสที่จะเป็น "1" ซึ่งการขัดแย้งที่เกิดขึ้นทำให้ไม่สามารถทดสอบความผิดพลาดได้ทุกรูปแบบ อีกทั้งการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR แบบลำดับ ไม่รองรับกับโครงสร้างของวงจรเลื่อนเฟส ซึ่งอินพุทและเอาท์พุทมีรูปแบบขนาน เป็นผลให้การกำเนิดข้อมูลทดสอบด้วยวิธีการ Partial LFSR Reseeding มีโอกาสเกิด Structural Dependency และ Correlation ของข้อมูลได้ อีกทั้งการเลื่อนบิตข้อมูลทดสอบเข้าสู่วงจรที่จะทดสอบแบบลำดับ จะไม่สอดคล้องกับการทำงานร่วมกับโครงสร้างของเซลล์ตรวจกวาดแบบ STUMPS เนื่องจาก STUMPS มีรูปแบบการเลื่อนข้อมูลอินพุทและเอาท์พุทในรูปแบบขนาน

3.3.2 Relaxation [45]

Relaxation เป็นการนำเซตของข้อมูลทดสอบที่มีความสามารถครอบคลุมความผิดพลาด มาทำการคำนวณใหม่ ให้ 1 ชุดของข้อมูลทดสอบสามารถครอบคลุมความผิดพลาดได้มากขึ้น และลดจำนวนข้อมูลทดสอบลง การทำ Relaxation ของข้อมูลทดสอบสามารถลดจำนวนชุดข้อมูลทดสอบได้ ดังที่แสดงตามตารางที่ 3-2 (a) และ (b)

จากตารางที่ 3-2 (a) เซตของข้อมูลทดสอบที่ใช้ในการตั้งค่าเริ่มต้น LFSR จำนวน 5 ชุด ประกอบด้วย *t1, t2, t3, t4* และ *t5* ชุดข้อมูลทดสอบทั้ง 5 ใช้ในการทดสอบความผิดพลาดจำนวน 8 ชุด คือ *f1, f2, f3, f4, f5, f6, f7* และ *f8* โดยหมายเลข "1" ในแต่ละสดมภ์หมายถึง ชุดข้อมูลทดสอบ *t* ในสดมภ์นั้นๆ สามารถตรวจจับความผิดพลาด *f* ในแนวแถวได้ เช่น ในสดมภ์ *t1* มีหลายเลข "1" ในแถว *f1, f3* และ *f6* หมายถึงชุดข้อมูลทดสอบ *t1* สามารถตรวจจับความผิดพลาด *f1 f3* และ *f6* ได้ ส่วนตารางที่ 3-2 (b) เป็นการนำเซตของข้อมูลทดสอบเดิมคือ *t1, t2, t3, t4* และ *t5* มาคำนวณด้วยวิธี Relaxation เพื่อลดจำนวนชุดของข้อมูลทดสอบ โดยจะพิจาณาจากชุดข้อมูลทดสอบแรกสามารถตรวจจับความผิดพลาดได้ และเป็นความผิดพลาดที่ไม่สามารถตรวจจับได้ด้วยชุดข้อมูลทดสอบอื่นก็จะได้รับการพิจารณาเป็นอันดับแรก เช่นชุดข้อมูลทดสอบ *t1* สามารถตรวจจับความผิดพลาด *f1, f3* และ *f6* ได้ จากนั้นก็พิจารณาที่ชุดข้อมูลทดสอบ *t2* ชุดข้อมูลทดสอบ *t2* สามารถตรวจจับความผิดพลาด *f1, f2, f3* และ *f5* ได้ แต่เนื่องจากความผิดพลาด *f3* ถูกตรวจจับ

ด้วยชุดข้อมูลทดสอบ *t1* แล้ว จึงแทนค่าด้วยเครื่องหมายขีด (-) ซึ่งเมื่อพิจารณาครบ 4 ชุดของข้อมูลทดสอบคือ *t1, t2, t3* และ *t4* ก็สามารถตรวจจับความผิดพลาดได้ทั้งหมด จึงไม่จำเป็นต้องใช้ชุดข้อมูลทดสอบ *t5* แถวการเจาะจงค่าคือจำนวนการเจาะจงค่าที่สามารถตรวจจับได้ของแต่ละชุดข้อมูลทดสอบ ส่วนแถวสุดท้ายคือจำนวนรวมการเจาะจงค่าทั้งหมด ซึ่งจะเห็นได้ว่า เมื่อมีการใช้การคำนวนแบบ Relaxation ทำให้มีจำนวนรวมของการเจาะจงค่าทั้งหมดน้อยลง

ตารางที่ 3-2 การตรวจจับความผิดพลาดแบบธรรมดา และแบบ Relaxation

	Initial Test Set *T*				
ความผิดพลาด	*t1*	*t2*	*t3*	*t4*	*t5*
f1	1	1			
f2		1			
f3	1		1		
f4			1		1
f5		1	1		
f6	1		1	1	
f7				1	
f8				1	1
การเจาะจงค่า	23	20	29	22	19
รวม	113				

(a) แบบธรรมดา

	Initial Test Set *T*				
ความผิดพลาด	*t1*	*t2*	*t3*	*t4*	*t5*
f1	1	-			
f2		1			
f3	1		-		
f4			1		-
f5		1	-		
f6	1		-	-	
f7				1	
f8				1	-
การเจาะจงค่า	23	17	22	19	0
รวม	81				

(b) แบบ Relaxation

เมื่อนำชุดข้อมูลทดสอบดังกล่าวไปทำงานร่วมกับการกำเนิดข้อมูลทดสอบแบบ LFSR Reseeding ทั้งแบบ Static หรือ Dynamic ก็สามารถลดขนาดของข้อมูล Seed ลงได้ เนื่องจากชุดข้อมูลทดสอบมีจำนวนการเจาะจงค่าลดลง

3.4 สรุป

การกำเนิดข้อมูลการทดสอบทั้งแบบ Static LFSR Reseeding และ Dynamic LFSR Reseeding นั้น สามารถกำเนิดข้อมูลการทดสอบได้มีประสิทธิภาพใกล้เคียงกัน แต่สถาปัตยกรรมแบบ Dynamic LFSR Reseeding [39] สามารถประยุกต์ให้สามารถลดข้อมูลการทดสอบ ลดจำนวนทรัพยากรที่ใช้ และประยุกต์ใช้กับวิธีการลดจำนวนข้อมูลการทดสอบได้หลากหลายมากกว่า อีกทั้งยังสามารถพัฒนาให้สามารถลดเวลาการทดสอบได้ งานวิจัยนี้จึงมุ่งเน้นการประยุกต์ใช้สถาปัตยกรรมแบบ Dynamic LFSR Reseeding ด้วยการพัฒนาวิธีการกำเนิดข้อมูลทดสอบด้วยวิธี Partial LFSR Reseeding ให้สามารถเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR และวงจรเลื่อนเฟสในรูปแบบขนาน เพื่อลดเวลาการทดสอบ การเกิด Structural Dependency และ Correlation ของข้อมูล ทั้งยังรองรับโครงสร้างแบบ STUMPS ของเซลล์ตรวจกวาด แต่จะไม่มีการเปลี่ยนแปลงวงจรที่จะทดสอบ จึงไม่ส่งผลต่อสมรรถนะการทำงานของชิป ทั้งไม่จำเป็นต้องเพิ่มวงจรพิเศษ ที่จะส่งผลต่อจำนวนทรัพยากรในการพัฒนา โดยเรียกวิธีดังกล่าวว่า Parallel LFSR Reseeding ซึ่งการกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ได้นำความรู้เกี่ยวกับ Selection Register มาใช้ในการควบคุมกลไกการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR เพื่อช่วยลดจำนวนข้อมูล Seed ลง

บทที่ 4

Parallel LFSR Reseeding

4.1 บทนำ

วงจรกำเนิดข้อมูลการทดสอบโดยการประยุกต์ใช้สถาปัตยกรรมแบบ Dynamic LFSR Reseeding ซึ่งเน้นการพัฒนาการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR และวงจรเลื่อนเฟสแบบขนาน หรือเรียกว่า Parallel LFSR Reseeding วงจรกำเนิดข้อมูลทดสอบวิธีนี้ ได้ออกแบบให้มีประสิทธิภาพในการกำเนิดข้อมูลทดสอบ ลดระยะเวลาการทดสอบ ลดการเกิด Structural Dependency และ Correlation ของข้อมูล และรองรับโครงสร้างของเซลล์ตรวจกวาดแบบ STUMPS ทั้งยังสามารถครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลทดสอบ

วงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding ได้แบ่งออกเป็น 2 แนวคิด แนวคิดที่ 1 นำเสนอวงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding ที่รองรับการทำงานร่วมกับวงจรเลื่อนเฟส และโครงสร้างของเซลล์ตรวจกวาดแบบ STUMPS และลดเวลาการทดสอบ ซึ่งได้นำเสนอในหัวข้อ 4.2 ส่วนแนวคิดที่ 2 นำเสนอวงจรกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ที่เน้นการลดขนาดข้อมูล Seed ด้วย Selection Register หรือ Parallel LFSR Reseeding with Selection Register ซึ่งรายละเอียดแนวคิดที่ 2 ได้กล่าวถึงในหัวข้อ 4.3 ส่วนหัวข้อที่ 4.4 เป็นการสรุปวงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding ทั้งสองแนวคิด

4.2 วงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding

การกำเนิดข้อมูลด้วยวิธี Parallel LFSR Reseeding เป็นการออกแบบการกำเนิดข้อมูลทดสอบที่พัฒนาจากการกำเนิดข้อมูลแบบ Partial LFSR Reseeding โดยพัฒนาการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR และวงจรเลื่อนเฟสให้อยู่ในรูปแบบขนาน เพื่อสนับสนุนการทำงานร่วมกับเซลล์ตรวจกวาดแบบ STUMP เน้นลดระยะเวลาการทดสอบและครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลทดสอบ การกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding แสดงดังรูปที่ 4-1 ประกอบด้วย 7 ส่วน จากรูปที่ 4-1 แสดงด้วยหมายเลข 1 ถึง 7 ดังนี้

หมายเลข 1. ข้อมูล Seed

หมายเลข 2. กลุ่มของ AND เกต แบบ 2 อินพุท

หมายเลข 3. กลุ่มของ XOR เกต แบบ 2 อินพุท

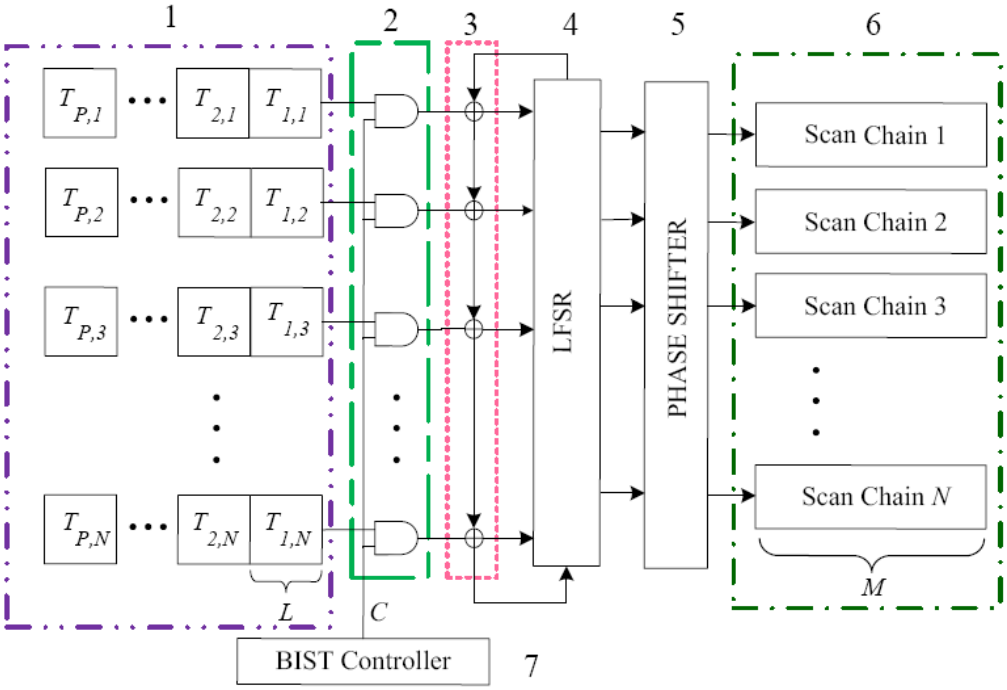
หมายเลข 4. วงจร LFSR

หมายเลข 5. วงจรเลื่อนเฟส

หมายเลข 6. วงจรที่จะทดสอบ

หมายเลข 7. BIST Controller หรือส่วนควบคุมการทดสอบแบบ BIST

รายละเอียดของส่วนต่างๆของ สถาปัตยกรรมของวงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding แบบพื้นฐาน อธิบายในหัวข้อ 4.2.1 ถึง 4.2.7



รูปที่ 4-1 วงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding

จากรูปที่ 4-1 วงจรการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ได้แบ่งการกำเนิดข้อมูลทดสอบตามแบบ Mixed-mode BIST เป็น 2 โหมดคือ DTPG และ PRPG ด้วยสัญญาณ *C* จากส่วนควบคุมการทดสอบแบบ BIST เมื่อสัญญาณ *C* มีค่าเป็น “0” เป็นการกำเนิดข้อมูลทดสอบในโหมด PRPG และเมื่อสัญญาณ *C* มีค่าเป็น “1” เป็นการกำเนิดข้อมูลทดสอบในโหมด DTPG โดยข้อมูล Seed จะถูกเลื่อนเข้าสู่วงจร LFSR ในรูปแบบขนาน มีกลุ่มของ AND เกต แบบ 2 อินพุท เป็นตัวกำหนดว่าข้อมูลดังกล่าว จะผ่านเข้าสู่วงจร LFSR หรือไม่ ส่วน

กลุ่มของ XOR เกต แบบ 2 อินพุท จะทำหน้าที่ในการป้อนกลับข้อมูล Seed เข้าสู่วงจร LFSR และผ่านเข้าสู่วงจรเลื่อนเฟส โดยวงจรเลื่อนเฟสนอกจากจะใช้เพื่อลดความเป็น Structural Dependency และ Correlation ของข้อมูลแล้ว ยังเป็นตัวกำหนดตำแหน่งการป้อนข้อมูล Seed ด้วย เพื่อควบคุมให้ข้อมูล Seed ที่ถูกเลื่อนเข้าสู่วงจร LFSR มีผลต่อวงจรที่จะทดสอบได้ตรงตำแหน่ง โดยข้อมูลการสร้างวงจรเลื่อนเฟสนั้น ได้อธิบายไว้ในเอกสาร [14-15]

ข้อมูล Seed แต่ละชุดจะใช้ในการกำเนิดข้อมูลของแต่ละ Test Cube เช่น $T_{1,1}$, $T_{1,2}$, $T_{1,3}$, ..., $T_{1,N}$ ใช้ในการกำเนิดข้อมูลให้กับ Test Cubes ที่ 1 เป็นต้น ซึ่งตัวแปร N คือจำนวนของ Scan Chain ส่วนตัวแปร L คือความยาวของข้อมูล Seed จำนวน 1 ชุด ซึ่งถ้าในกรณีที่ดีที่สุด ตัวแปร L จะมีค่าเท่ากับ 1 ในทางกลับกัน ค่าที่มากที่สุดของ L จะมีค่าเท่ากับความยาวของ Scan Chain ซึ่งแทนด้วยตัวแปร M

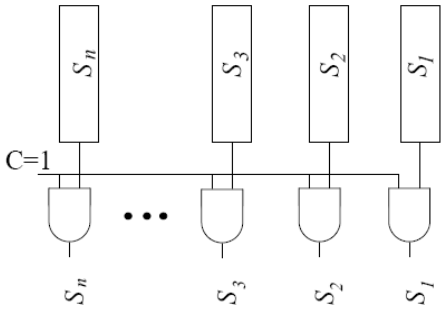
4.1.1 ข้อมูล Seed

ข้อมูล Seed เป็นข้อมูลที่ใช้ในการกำเนิดข้อมูลการทดสอบแก่วงจรทดสอบในโหมดการกำเนิดข้อมูลทดสอบแบบ DTPG โดยข้อมูล Seed จำนวน 1 ชุดจะมีขนาดเท่ากับขนาดของเอาท์พุทวงจรเลื่อนเฟสและความยาวของเซลล์ตรวจกวาด ซึ่งจากรูปที่ 4-1 คือเท่ากับ N ข้อมูล Seed จำนวน 1 ชุด คือ $T_{1,1}$, $T_{1,2}$, $T_{1,3}$, ... , $T_{1,N}$ จะถูกเลื่อนพร้อมๆกันเข้าสู่วงจร LFSR และวงจรเลื่อนเฟส ซึ่งข้อมูล Seed 1 ชุดใช้กำเนิดข้อมูลการทดสอบแก่วงจรทดสอบ 1 ชุด ดังนั้นจำนวนชุดของข้อมูล Seed จะเท่ากับจำนวนของจำนวนชุดข้อมูลทดสอบทั้งหมด หรือ P โดยความยาวของข้อมูล Seed จำนวน 1 ชุด มีค่าเท่ากับ L ในกรณีน้อยที่สุด L จะเท่ากับ 1 ส่วนกรณีมากที่สุดมีค่าเท่ากับความยาวของ Scan Chain คือ M

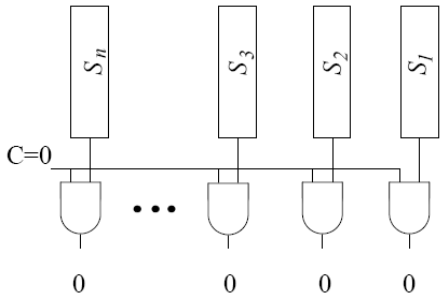
4.2.2 กลุ่มของ AND เกต แบบ 2 อินพุท

ลอจิก AND เกตแบบ 2 อินพุทในวงจรกำเนิดข้อมูลการทดสอบแบบ Parallel LFSR Reseeding ใช้ในการควบคุมการป้อนข้อมูล Seed สู่วงจร LFSR โดยอินพุทข้างหนึ่งของ AND เกต เชื่อมต่อกับข้อมูล Seed และอีกข้างหนึ่งเชื่อมต่อกับสัญญาณ C ซึ่งสัญญาณ C ใช้ควบคุมโหมดการทำงานของ ATPG เมื่อสัญญาณ C="1" จะเป็นการทำงานในโหมด DTPG คือข้อมูล Seed ก็จะถูกเลื่อนเข้าสู่วงจร LFSR เพื่อกำเนิดข้อมูลทดสอบแก่ชุดข้อมูลทดสอบที่ทดสอบยาก โดยจำนวนของลอจิก AND เกตจะเท่ากับจำนวนชุดของข้อมูล Seed เพื่อให้สามารถควบคุมข้อมูล Seed ที่จะถูกเลื่อนเข้าสู่วงจร LFSR ได้ทุกตำแหน่ง ซึ่งรูปที่ 4-2 แสดงส่วนประกอบการกำเนิดข้อมูลทดสอบในโหมด DTPG เมื่อสัญญาณ C = "1" แต่เมื่อสัญญาณ C= "0" ผลของข้อมูลที่ผ่าน AND

เกตก็จะให้เอาท์พุทเท่ากับ “0” จึงทำให้ไม่มีการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR ดังที่ได้แสดงในรูปที่ 4-3



รูปที่ 4-2 กลไกการกำเนิดข้อมูลทดสอบในโหมด DTPG เมื่อสัญญาณ C = “1”

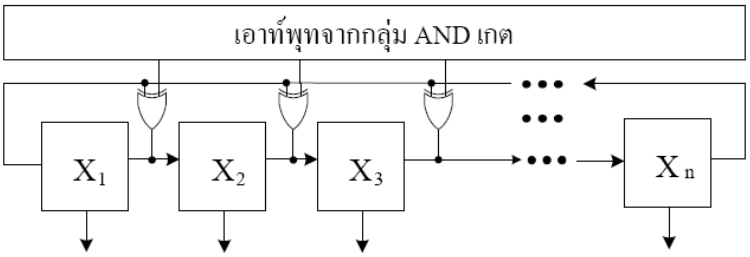


รูปที่ 4-3 กลไกการกำเนิดข้อมูลทดสอบในโหมด PRPG เมื่อสัญญาณ C = “0”

4.2.3 กลุ่มของ XOR เกต แบบ 2 อินพุท

ลอจิก XOR เกตแบบ 2 อินพุทในสถาปัตยกรรมของวงจรกำเนิดข้อมูลการทดสอบแบบ Parallel LFSR Reseeding นั้น ใช้ในการรวมข้อมูลเอาท์พุทในรูปแบบขนานของ LFSR และข้อมูล Seed จากกลุ่มของลอจิก AND เกต ซึ่งกลุ่มลอจิก XOR เกตแบบ 2 อินพุทในของวงจรกำเนิดข้อมูลการทดสอบแสดงดังรูปที่ 4-4

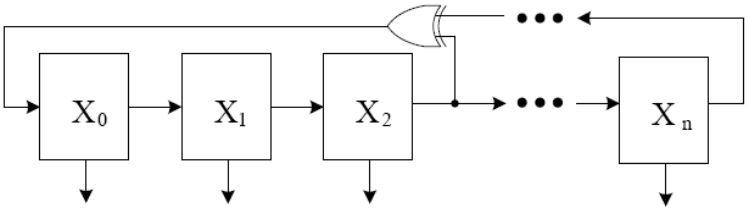
จากรูปที่ 4-4 กลุ่มของ XOR เกตแบบ 2 อินพุท โดยขาอินพุทข้างหนึ่งของ XOR เกตได้จากเอาท์พุทของกลุ่มของ AND เกตแบบ 2 อินพุท ขาอินพุทอีกข้างหนึ่งได้จากเอาท์พุทแบบขนานของวงจร LFSR ส่วนผลลัพธ์จากการ XOR จะเป็นอินพุทให้กับวงจรเลื่อนเฟส และเป็นอินพุทให้กับ Flip-Flop ที่ใกล้เคียงในวงจร LFSR โดยจำนวนของ XOR เกต จะเท่ากับจำนวนเอาท์พุทของวงจรเลื่อนเฟส



รูปที่ 4-4 กลุ่มของ XOR เกตแบบ 2 อินพุท

4.2.4 วงจร LFSR

วงจร LFSR เป็นส่วนสำคัญในการสร้างวงจรกำเนิดข้อมูลทดสอบด้วยวิธีการ LFSR Reseeding ซึ่งจะทำหน้าที่ในการเลื่อนและหมุนวนข้อมูล Seed ไปยังเซลล์ต่างๆของ เซลล์ตรวจกวาด ตัวอย่างสถาปัตยกรรมของวงจร LFSR แสดงดังรูปที่ 4-5



รูปที่ 4-5 ตัวอย่างวงจร LFSR

จากรูปที่ 4-5 แสดงตัวอย่าง LFSR ซึ่งเป็นสถาปัตยกรรมพื้นฐานวงจร LFSR ขนาด *n* บิต โดยวงจร LFSR เป็นส่วนสำคัญในโครงสร้างของวงจรการทดสอบแบบ LFSR Reseeding ซึ่งวงจรการทดสอบแต่ละชุดใช้วงจร LFSR ที่แตกต่างกัน โดยจำนวน Flip-Flop ของวงจร LFSR ที่ใช้ในการกำเนิดข้อมูลทดสอบ ได้อ้างอิงจากข้อมูลในวารสารวิชาการกำเนิดข้อมูลแบบ Partial LFSR Reseeding [1][44]

4.2.5 วงจรเลื่อนเฟส

วงจรเลื่อนเฟสเป็นวงจรการเลื่อนลำดับของเลขฐานสองที่ถูกกำเนิดโดย LFSR ก่อนที่ข้อมูลจะถูกส่งไปยังวงจรที่จะทดสอบ การสร้างวงจรเลื่อนเฟสเกิดจากการนำ XOR เกตมาเชื่อมโยงกันระหว่างเอาท์พุทของ LFSR แต่ละตำแหน่ง เพื่อลดการเกิด Structural Dependency และ

Correlation ของข้อมูล ซึ่งวงจรเลื่อนเฟสสร้างจากการนำข้อมูลทดสอบที่กำเนิดจากวิธี PRPG มาใช้ในการสร้างวงจรเลื่อนเฟส เช่นข้อมูลที่กำเนิดจากวิธี PRPG ด้วย LFSR ขนาด 38 บิต โดยค่า I=6 B=7 และ L=17 (รายละเอียดของตัวแปร I, B และ L อยู่ในบทที่ 2) เพื่อทดสอบวงจร S5387 ตัวอย่างข้อมูลเอาท์พุทของวงจร LFSR ที่นำมาใช้ในการสร้างวงจรเลื่อนเฟสแสดงดังตารางที่ 4-1

ในตารางแสดง 4-1 ข้อมูลที่ใช้ในการสร้างวงจรเลื่อนเฟส ข้อมูลในแต่ละแถวใช้อธิบาย 1 ชุดของวงจรเลื่อนเฟส โดยตัวเลข 1 จะใช้บอกตำแหน่งเอาท์พุทของ LFSR มาเป็นอินพุทของวงจรเลื่อนเฟส ส่วนตัวเลข 0 แสดงว่าไม่ปรากฏเอาท์พุทของ LFSR จากตำแหน่งนั้นๆในชุดวงจรเลื่อนเฟส เช่น ข้อมูลในลำดับที่ 1 บิตที่ 6, 14, 19, 21, 25, 29, 32 และ 37 แสดงว่า ในชุดแรกของวงจรเลื่อนเฟสเกิดจากเอาท์พุทของ LFSR บิตที่ 6, 14, 19, 21, 25, 29, 32 และ 37 ทำการ XOR กัน โดยตำแหน่งเอาท์พุทของ LFSR ที่นำมาสร้างวงจรเลื่อนเฟสมีผลต่อการวางตำแหน่งของข้อมูล Seed ทั้งมีผลต่อสมการที่ใช้อธิบายความสัมพันธ์ในการกำเนิดข้อมูลทดสอบ

ตารางที่ 4-1 ตารางข้อมูลสร้างวงจรเลื่อนเฟสเพื่อทดสอบวงจร S5387

ลำดับ	บิตที่ 1-5	บิตที่ 6-10	บิตที่ 11-15	บิตที่ 16-20	บิตที่ 21-25	บิตที่ 26-30	บิตที่ 31-35	บิตที่ 36-38
1	10000	00000	10100	01000	00000	00000	00000	001
2	00100	00100	00000	00101	00010	00000	00000	000
3	00001	00001	00001	00000	00001	01000	10000	000
4	00000	01000	00000	00000	00010	00010	00110	000
5	00100	00000	00010	00000	00000	00000	10000	100
6	00000	01100	00000	00000	00100	00000	10010	010
7	01000	00100	00000	00000	00001	00010	10000	000
8	00010	00010	00001	00000	00000	00000	01000	101
9	11000	01000	10000	10000	01000	00000	00000	000
10	10001	00000	01000	00000	00001	00001	00001	000
11	00000	10011	10000	00000	00000	00100	00001	000
12	00100	01001	00000	10000	01000	00000	00010	000
13	00000	00010	11100	00000	00000	00010	10000	000

ลำดับ	บิตที่ 1-5	บิตที่ 6-10	บิตที่ 11-15	บิตที่ 16-20	บิตที่ 21-25	บิตที่ 26-30	บิตที่ 31-35	บิตที่ 36-38
14	00000	00000	00000	10111	00000	00000	00000	101
15	00100	00000	00100	00000	00100	00100	00100	001
16	00100	10000	01100	01000	00000	00000	00001	000
17	10011	00000	00100	00000	00000	00100	00100	000

4.2.5.1 การเลือกตำแหน่งสำหรับการป้อนข้อมูล Seed

การเลือกตำแหน่งการ Seed หรือ Tap Seed เป็นการเลือกตำแหน่งที่จะเลื่อนข้อมูล Seed เข้าสู่วจร LFSR เพื่อให้สามารถควบคุมของค่าของเอาท์พุทที่จะถูกเลื่อนเข้าสู่ Scan Chain ได้ตามต้องการ โดยตำแหน่งการ Seed แต่ละตำแหน่งเลือกจากข้อมูลการสร้างวงจรเลื่อนเฟส เพื่อให้ข้อมูล Seed ในแต่ละบิตปรากฏในทุกตำแหน่งที่ต้องการของข้อมูลทดสอบ จำนวนตำแหน่งการ Seed จึงเท่ากับจำนวนชุดของวงจรเลื่อนเฟสและเท่ากับจำนวนของ Scan Chain จากตารางที่ 4-1 จำนวนชุดวงจรเลื่อนเฟสเท่ากับ 17 ชุด ดังนั้นจำนวนตำแหน่งของการ Seed จะเท่ากับ 17 เช่นกัน

จากตารางที่ 4-1 ข้อมูลลำดับที่ 16 ตำแหน่งที่จะสามารถเลือกตำแหน่งการ Seed ได้บิตที่ 3, 5, 9, 12, 15, 17, 23 และ 25 โดยตำแหน่งการเลื่อนบิต Seed ที่เลือกคือ บิตที่ 3 ของวงจร LFSR การเลือกตำแหน่งการ Seed จะต้องหลีกเลี่ยงการซ้ำกับตำแหน่งการ Seed ในชุดก่อนหน้าและการวนลูปของข้อมูล Seed ซึ่งในบิตที่ 3 ของข้อมูลลำดับที่ 17 ไม่สามารถเลือกเป็นบิตตำแหน่งการ Seed ได้ เนื่องจากบิตที่ 3 จะซ้ำกับตำแหน่งการ Seed ของข้อมูลลำดับที่ 16 ส่วนบิตที่ 9 สามารถเลือกเป็นตำแหน่งการ Seed ได้ แต่ไม่เหมาะสมที่เนื่องจากจะเกิดการวนลูปของข้อมูล Seed การวนลูบของข้อมูล Seed ทำให้ไม่สามารถควบคุมผลลัพท์ของข้อมูลการทดสอบได้ การวนลูปของข้อมูล Seed อธิบายในหัวข้อถัดไป

4.2.5.2 การวนลูปของข้อมูล Seed

การวนลูปของข้อมูล Seed คือตำแหน่งที่ถูกเลือกเป็นตำแหน่งเลื่อนข้อมูล Seed มีการสมมูลกันของตำแหน่งที่ปรากฏตัวเลข “1” เช่น ข้อมูลลำดับที่ 16 ตารางที่ 4-1 บิตที่ 3 ถูกเลือกเป็นตำแหน่งการ Seed ซึ่งทำให้ลำดับที่ 17 ไม่เหมาะสำหรับเลือกเป็นตำแหน่งการ Seed บิตที่ 9 ได้ เนื่องจากเกิดการลูปของข้อมูล Seed ในบิตที่ 9 เป็นตำแหน่งที่เกิดการสมมูลกันระหว่างข้อมูลลำดับที่ 16 และ17 คือลำดับที่ 16 และ 17 บิตที่ 3 และบิตที่ 9 มีหมายเลข “1” เหมือนกัน ข้อมูลจึงเกิดการสมมูลกัน และจะทำให้ข้อมูล Seed เกิดการวนลูป ซึ่งการวนลูปของข้อมูล Seed จะทำให้เกิดการ Dependency ของสมการที่ใช้อธิบายการกำเนิดข้อมูลทดสอบ เป็นผลให้ไม่สามารถควบคุมผลลัพท์

ของข้อมูลการทดสอบได้ในบิตดังกล่าวได้ ตัวอย่างของการวนลูปของข้อมูล Seed บิตที่ 3 และบิตที่ 9 แสดงดังรูปที่ 4-6 ส่วนรูปที่ 4-7 แสดงการวนลูปของข้อมูล Seed ของบิตที่ 3 และ 15 ของข้อมูลลำดับที่ 16 และ 17

ลำดับที่ 17 ของชุดข้อมูลวงจรเลื่อนเฟส จะไม่สามารถเลือกบิตที่ 3, 9 และ 15 ได้ ดังนั้นในชุดข้อมูลที่ 17 ตำแหน่งที่สามารถเลือกตำแหน่งการ Seed ได้ก็จะมีเพียงตำแหน่งที่ 21, 27, 28, 36 และ 38 เท่านั้น

ลำดับ	บิตที่ 1-5	บิตที่ 6-10
16	00001	00000
17	00000	00000

รูปที่ 4-6 ตัวอย่างการวนลูปของข้อมูล Seed บิตที่ 3 และบิตที่ 9 ของข้อมูลลำดับที่ 16 และ 17

ลำดับ	บิตที่ 1-5	บิตที่ 11-15
16	00001	01000
17	00000	00000

รูปที่ 4-7 ตัวอย่างการวนลูปของข้อมูล Seed บิตที่ 3 และบิตที่ 15 ของข้อมูลลำดับที่ 16 และ 17

4.2.6 วงจรที่จะทดสอบ

วงจรที่จะทดสอบเป็นการนำชุดข้อมูลทดสอบของวงจรมาตฐานที่ได้จากการ Simulation มาใช้ โดยชุดข้อมูลทดสอบของวงจรมาตราฐานแต่ละวงจรจะถูกจัดเรียงในรูปแบบของ Scan Chain โดยการคำนวณความกว้างของแต่ละเซลล์ตรวจกวาดจะคำนวณถึงจำนวนบิต Flip-Flop ของวงจร LFSR คูณกับจำนวนของชุดข้อมูลทดสอบแล้วจะต้องไม่มากกว่าจำนวน M-Sequence ของวงจร LFSR ที่สามารถกำเนิดได้ ดังที่แสดงในสมการ (2-1)

4.2.7 BIST Controller หรือส่วนควบคุมการทดสอบแบบ BIST

BIST Controller หรือส่วนควบคุมการทดสอบแบบ BIST จะทำหน้าที่ควบคุมการทำงานของส่วนต่างของวงจรทดสอบให้สามารถทำงานประสานกันได้ โดยส่วนควบคุมการ

ทดสอบแบบ BIST ของวงจรกำเนิดข้อมูลทดสอบด้วยวิธี Parallel LFSR Reseeding จะทำหน้าที่ควบคุมสัญญาณ *C* ขนาด 1 บิต ให้เปลี่ยนโหมดการทดสอบระหว่าง PRPG และ DTPG เมื่อ

- สัญญาณ *C* มีค่าเป็น “0” เป็นการกำเนิดข้อมูลทดสอบในโหมด PRPG คือจะใช้การกำเนิดข้อมูลการทดสอบด้วยกลไกการกำเนิดข้อมูลการทดสอบของวงจร LFSR เพื่อทดสอบความผิดพลาดที่ทดสอบง่าย

- สัญญาณ *C* มีค่าเป็น “1” เป็นการกำเนิดข้อมูลทดสอบในโหมด DTPG โดยสัญญาณ *C* เป็น “1” ข้อมูล Seed ก็จะสามารถเลื่อนผ่าน AND เกตได้ ซึ่งจะเป็นการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR Reseeding ซึ่งวิธีการนี้ใช้ทดสอบความผิดพลาดที่มีรูปแบบสุ่มทน

สำหรับโหมด DTPG ส่วนควบคุมการทดสอบแบบ BIST ทำหน้าที่ควบคุมวงจรนับจำนวนชุดข้อมูลทดสอบ วงจรนับจำนวนเซลล์ตรวจกวาด และวงจรนับสัญญาณ *C* โดยการควบคุมวงจรนับจำนวนเซลล์ตรวจกวาด ส่วนควบคุมแบบ BIST ทำหน้าที่นับจำนวนเซลล์ตรวจกวาดของแต่ละ Scan Chain ให้ทำงานสอดคล้องกับวงจรนับจำนวนชุดข้อมูลทดสอบและวงจรนับสัญญาณ *C* เมื่อวงจรนับเซลล์ตรวจกวาดนับจนครบ ส่วนควบคุมการทดสอบแบบ BIST จะทำหน้าที่ตั้งค่าเริ่มต้นใหม่ (Reset) ให้กับวงจรนับเซลล์ตรวจกวาด เพิ่มค่าของวงจรนับจำนวนชุดข้อมูลทดสอบ และสัญญาณ *C* ให้มีค่าเท่ากับ “1” เพื่อเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR เมื่อวงจรนับสัญญาณ *C* ครบจำนวน *L* (จำนวนบิตข้อมูล Seed) ส่วนควบคุมการทดสอบแบบ BIST จะทำหน้าที่ตั้งค่าเริ่มต้นใหม่ วงจรนับสัญญาณ *C* และกำหนดสัญญาณ *C* ให้มีค่าเท่ากับ “0” จากนั้นตั้งค่าเริ่มต้นใหม่ให้กับวงจรนับสัญญาณ *C* และทำขั้นตอนดังกล่าวซ้ำจนครบทุกชุดของข้อมูลทดสอบ

สำหรับโหมด PRPG ส่วนควบคุมการทดสอบแบบ BIST ทำหน้าที่ควบคุมวงจรนับจำนวนชุดข้อมูลทดสอบ วงจรนับจำนวนเซลล์ตรวจกวาดเท่านั้น ซึ่งมีการทำงานคล้ายกับโหมด DTPG

4.3 วงจรการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register หรือ Parallel LFSR Reseeding with Selection Register

การกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register นี้ เป็นการปรับปรุงวิธีการกำเนิดข้อมูลด้วยโครงสร้างพื้นฐานของ Parallel LFSR Reseeding เพื่อลดจำนวนข้อมูล Seed ลง แต่ยังคงครอบคลุม 100 เปอร์เซ็นต์ของข้อมูลทดสอบเช่นเดิม โดยแนวคิดที่ 2 ของวงจรการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register แสดงดังรูปที่ 4-8

วงจรกำเนิดข้อมูลการทดสอบแบบขนานด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register ประกอบด้วยที่เพิ่มขึ้นมากจาก วิธี Parallel LFSR Reseeding เดิม คือ กลุ่มของ AND เกต แบบ 2 อินพุทและ 3 อินพุท, สัญญาณ C_1, สัญญาณ C_2, Selection Register และ วงจรนับสัญญาณ C_1 และวงจรนับสัญญาณ C ในส่วนควบคุมการทดสอบแบบ BIST

จากรูปที่ 4-8 วงจรการกำเนิดข้อมูลทดสอบด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register มีวงจรควบคุมแบบ BIST ทำหน้าที่ควบคุมการทำงานให้ส่วนประกอบทั้งหมดทำงานประสานกันผ่านทาง Selection Register และ วงจรนับ (Counter 1 และ Counter 2) โดย Selection Register จะทำหน้าที่เก็บจำนวนข้อมูลของวงจรนับชุดที่ 1 หรือ Counter 1 และวงจรนับชุดที่ 2 หรือ Counter 2 เพื่อใช้นับจังหวะการ On หรือ Off ของสัญญาณ C_1 และ C_2 ตามลำดับ ซึ่งจำนวนข้อมูล *N*-bit ของ Selection Register คือจำนวนข้อมูลที่ต้องจัดเก็บ ซึ่งมีค่าเท่ากับจำนวนของ Scan chain คือ n ชุด

เมื่อ Selection Register โหลดจำนวนข้อมูลลงสู่วงจรนับชุดที่ 1 วงจรนับจะทำการนับจังหวะการ On ของสถานะสัญญาณ C_1 จนกระทั่งจำนวนนับเท่ากับจำนวนใน Selection Register สถานะของสัญญาณ C_1 จะถูกตั้งค่าเริ่มต้นใหม่เพื่อนับจังหวะการ On ของสัญญาณ C_1 ของข้อมูลทดสอบชุดถัดไป ส่วนวงจรนับชุดที่ 2 จะทำการนับจังหวะการ On ของสถานะสัญญาณ C_2 จนกระทั่งจำนวนนับเท่ากับจำนวนใน Selection Register สถานะของสัญญาณ C_2 จะถูกตั้งค่าเริ่มต้นใหม่ เมื่อทำการกำเนิดข้อมูลทดสอบชุดใหม่

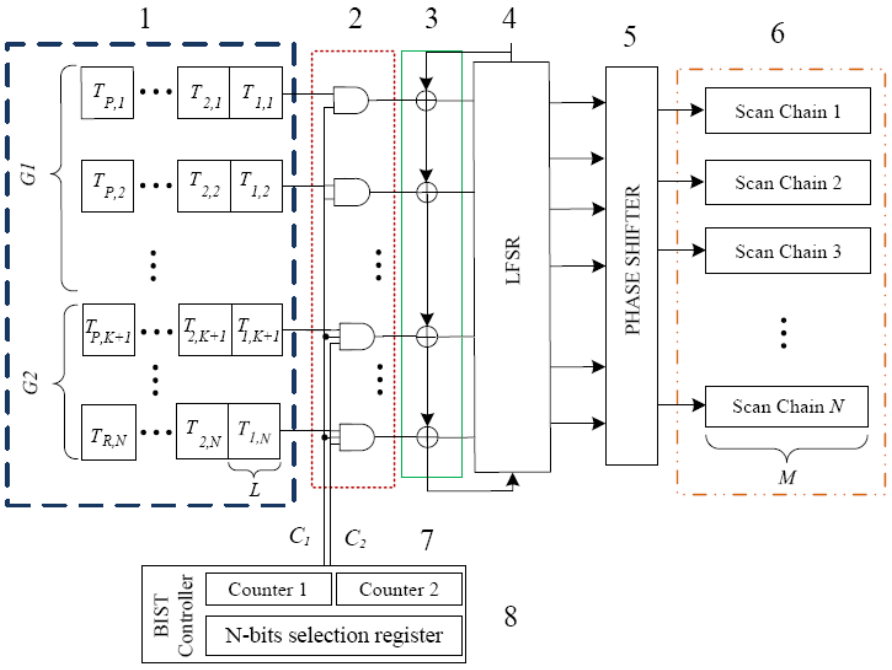
การกำหนดสถานะของสัญญาณ C_1 และ C_2 แสดงดังนี้

- เมื่อ C_1 มีค่าเป็น “0” ไม่ว่า C_2 จะมีค่าเป็น “1” หรือ “0” จะเป็นการกำเนิดข้อมูลทดสอบด้วยวิธี PRPG เพื่อทดสอบความผิดพลาดที่ทดสอบง่าย
- เมื่อ C_1 มีค่าเป็น “1” และ C_2 จะมีค่าเป็น “0” เป็นการกำเนิดข้อมูลทดสอบด้วยวิธี DTPG ซึ่งจะให้การเลื่อนข้อมูล Seed ในกลุ่ม *G1* เท่านั้น
- เมื่อ C_1 มีค่าเป็น “1” และ C_2 จะมีค่าเป็น “1” เป็นการกำเนิดข้อมูลทดสอบด้วยวิธี DTPG ซึ่งจะให้การเลื่อนข้อมูล Seed ในกลุ่ม *G1* และ *G2* เข้าสู่วงจร LFSR

ดังนั้นกลุ่มของ AND เกตแบบ 2 อินพุท ที่มีอินพุทข้างหนึ่งเชื่อมต่อกับข้อมูล Seed และ อินพุทอีกข้างจากสัญญาณ C_1 จึงเป็นส่วนหนึ่งในการช่วยควบคุมการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR ส่วนกลุ่มของ AND เกตแบบ 3 อินพุท ที่มีอินพุทข้างหนึ่งเชื่อมต่อกับข้อมูล Seed สัญญาณ C_1 และสัญญาณ C_2 จึงเป็นส่วนช่วยในการควบคุมการเลื่อนข้อมูล Seed ในกลุ่ม *G2* ของข้อมูล Seed กลุ่มซึ่งของข้อมูล Seed ที่ได้ออกแบบให้มีด้วยกัน 2 กลุ่ม เพื่อให้สามารถควบคุมการเลื่อนข้อมูลแต่ละกลุ่มเข้าสู่วงจร LFSR ได้แบบอิสระต่อกัน ในกำหนดการเลื่อนข้อมูล Seed ของ

กลุ่ม *G1* จะขึ้นอยู่กับสัญญาณ C_1 เท่านั้น ส่วนการเลื่อนข้อมูล Seed ของกลุ่ม *G2* จะขึ้นอยู่กับสัญญาณ C_1 และ C_2 วิธีการนี้จะสามารถควบคุมให้เกิดการเลื่อนข้อมูล Seed กลุ่ม *G2* เมื่อชุดข้อมูลทดสอบมีจำนวนการเจาะจงค่าสูง และไม่เลื่อนข้อมูล Seed กลุ่ม *G2* เมื่อชุดข้อมูลทดสอบมีจำนวนการเจาะจงค่าน้อย โดยการเปรียบเทียบจากจำนวนเฉลี่ยการเจาะจงค่า หรือ S_{avg} ซึ่งจะช่วยลดจำนวนข้อมูล Seed ในการกำเนิดข้อมูลการทดสอบได้

กลุ่มของ XOR เกตของแนวคิดที่ 2 กำเนิดข้อมูลการทดสอบแบบ Parallel LFSR Reseeding ร่วมกับ Selection Register นี้ มีการทำงานเหมือนกับกลุ่มของ XOR เกตของแนวคิดที่ 1 คือใช้ในการป้อนกลับข้อมูล Seed เข้าสู่วงจร LFSR และทำการรวมกับข้อมูลที่มีอยู่เดิมในวงจร ให้ข้อมูล Seed ที่ถูกเลื่อนไปก่อนหน้าส่งผลต่อการกำเนิดข้อมูลทดสอบในชุดปัจจุบัน



รูปที่ 4-8 วงจรการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register

ส่วนประกอบที่เพิ่มขึ้นของวงจรกำเนิดข้อมูลการทดสอบแบบ Parallel LFSR Reseeding ร่วมกับ Selection Register นั้น มีส่วนสำคัญที่แตกต่างจากแนวคิดที่ 1 ได้นำเสนอในหัวข้อ 4.3.1- 4.3.4

4.3.1 ข้อมูล Seed

Parallel LFSR Reseeding ร่วมกับ Selection Register ได้แบ่งข้อมูล Seed ออกเป็น 2 กลุ่มคือ *G1* และ *G2* เพื่อให้สามารถลดจำนวนข้อมูล Seed ลง และให้สอดคล้องกับการทำงานของ Selection Register รูปที่ 4-8 แสดงการแบ่งกลุ่มข้อมูล Seed โดยกลุ่ม *G1* จะมีข้อมูลเท่ากับ *K* บิต และกลุ่ม *G2* จะมีจำนวนข้อมูล Seed เท่ากับ *N-K* บิต ตัวแปร *T* แทนข้อมูล Seed และตัวแปร *L* แทนความยาวของบิตข้อมูลในการกำเนิดข้อมูลทดสอบจำนวน 1 ชุด ต่อ 1 ชุดข้อมูลทดสอบ

ข้อมูล Seed ทั้งสองกลุ่มมีกลไกในการโหลดเข้าสู่วงจร LFSR ที่แตกต่างกัน ดังนี้

- การเลื่อนข้อมูล Seed แบบครึ่ง หรือ Half Seed กลุ่มข้อมูลทดสอบ *G1* ที่จำนวนเท่ากับ *K* บิต เท่านั้นที่จะถูกเลื่อนเข้าสู่วงจร LFSR เช่น การเลื่อนข้อมูล Seed แบบครึ่งเพื่อกำเนิดข้อมูลทดสอบสำหรับชุดข้อมูลทดสอบที่ 1 ข้อมูล Seed $T_{1,1}$, $T_{1,2}$, ..., $T_{1,K}$ จะถูกเลื่อนเข้าสู่วงจร LFSR
- การเลื่อนข้อมูล Seed แบบเต็ม หรือ Full Seed กลุ่มข้อมูลทดสอบ *G1* และ *G2* จะถูกเลื่อนเข้าสู่วงจร LFSR ในการกำเนิดข้อมูลการทดสอบ จากรูปที่ 4-8 ถ้าการกำเนิดข้อมูลการทดสอบให้กับชุดข้อมูลทดสอบที่ 2 เป็นการทดสอบแบบเต็ม ข้อมูล Seed คือ $T_{2,1}$, $T_{2,2}$, ..., $T_{2,N}$ จะถูกโหลดเข้าสู่วงจร LFSR

4.3.2 กลุ่มของ AND เกตแบบ 2 และ 3 อินพุท

กลุ่มของ AND เกตแบบ 2 อินพุทในวงจรกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register เป็นตัวกำหนดการเลื่อนหรือไม่เลื่อน ของข้อมูล Seed สู่วงจร LFSR โดย AND เกตแบบ 2 อินพุท โดยอินพุทขาหนึ่งรับมาจากสัญญาณ C_1 และอีกข้างหนึ่งจะรับอินพุทจากข้อมูล Seed เพื่อกำหนดการเลื่อนบิตข้อมูล Seed กลุ่ม *G1* เข้าสู่วงจร LFSR ส่วน AND เกตแบบ 3 อินพุท ขาข้างหนึ่งจะรับอินพุทจากสัญญาณ C_1 อินพุทข้างที่สอง จะรับอินพุทจากสัญญาณ C_2 เพื่อกำหนดการเลื่อนบิตข้อมูล Seed กลุ่ม *G2* เข้าสู่วงจร LFSR และขาอินพุทสุดท้ายรับอินพุทจากข้อมูล Seed โดยกลุ่มของ AND เกตแบบ 2 และ 3 อินพุทสามารถเห็นความแตกต่างจาก Parallel LFSR Reseeding ซึ่งเป็นแนวคิดแรกได้ดังรูปที่ 4-8

4.3.3 ส่วนควบคุมการทดสอบแบบ BIST

ส่วนควบคุมการทดสอบแบบ BIST ของวิธีการกำเนิดข้อมูลทดสอบ Parallel LFSR Reseeding ด้วย Selection Register จะทำหน้าที่ควบคุมการทำงานของวงจรนับ หรือ Counter

สัญญาณ *C* และ Selection register ให้เปลี่ยนโหมดการทดสอบระหว่าง DTPG และ PRPG รวมถึงควบคุมการเลื่อนกลุ่มข้อมูล Seed เข้าสู่วงจร LFSR ระหว่างการเลื่อนบิตแบบเต็มและแบบครึ่ง

4.3.3.1 วงจรนับ

จากรูปที่ 4-8 วงจรนับจำนวน 2 ชุดคือ Counter 1 และ Counter 2 ซึ่งใช้ในการควบคุมการทำงาน (On) หรือ หยุดทำงาน (Off) ของสัญญาณ C_1 และสัญญาณ C_2 โดย เมื่อวงจรนับ Counter 1 เริ่มนับจะเป็นผลให้สัญญาณ C_1 ทำงาน และเมื่อมีการทั้งค่าเริ่มต้นใหม่ สัญญาณ C_1 ก็จะหยุดทำงานเช่นกัน ส่วนวงจรนับ Counter 2 จะควบคุมการทำงานของสัญญาณ C_2 โดยจะมีกลไกการทำงานเหมือนกับวงจรนับ Counter 1

4.3.3.2 สัญญาณ C

สัญญาณ C มีด้วยกัน 2 เส้น คือ สัญญาณ C_1 และสัญญาณ C_2 ซึ่งสัญญาณทั้งสองจะทำหน้าที่ในการควบคุมการทำงานที่แตกต่างกัน คือ

สัญญาณ C_1 จะใช้ในการควบคุมโหมดการทำงานของ Mixed-mode BIST ระหว่าง PRPG และ DTPG คือเมื่อสัญญาณ C_1 มีค่าเท่ากับ “0” จะกำเนิดข้อมูลการทดสอบในโหมด PRPG และเป็น C_1 มีค่าเท่ากับ “1” จะกำเนิดข้อมูลการทดสอบในโหมด DTPG ทั้งเป็นการควบคุมการเลื่อนบิตข้อมูล Seed ในกลุ่ม *G1* ด้วย เมื่อสัญญาณ C_1 มีค่าเท่ากับ “1” ก็จะทำให้ข้อมูล Seed สามารถเลื่อนเข้าสู่วงจร LFSR ได้

สัญญาณ C_2 จะทำหน้าที่ควบคุมการเลื่อนบิต (Enable) หรือหยุดการเลื่อนบิต (Disable) ของข้อมูล Seed กลุ่มที่ *G2* เมื่อสัญญาณ C_2 มีค่าเท่ากับ “1” ข้อมูล Seed กลุ่มที่สองก็จะถูกเลื่อนเข้าสู่วงจร LFSR ในทางกลับกัน เมื่อสัญญาณ C_2 มีค่าเท่ากับ “0” ข้อมูล Seed กลุ่มที่ *G2* ไม่สามารถเลื่อนเข้าสู่วงจร LFSR ได้ ซึ่งสัญญาณ C_2 จะทำงานสอดคล้องกับ AND เกตแบบ 3 อินพุท

4.3.3.3 Selection Register

วิธีการ Selection Register [55-56] เดิมถูกนำไปใช้ลดความน่าจะเป็นในการเกิดความผิดพลาดแบบ X ด้วยวิธี การยกเลิก X หรือ X’s canceling ของวงจร MISR สำหรับแนวคิดของการกำเนิดข้อมูลทดสอบแบบด้วยวิธี Parallel LFSR Reseeding นั้น ได้นำ Selection Register มาใช้ในการควบคุมการทำงานของวงจรนับ เพื่อให้เกิดการนับ และการตั้งค่าเริ่มต้นใหม่ของวงจรนับ วงจรนับจะเริ่มนับ เมื่อมีการกำเนิดการทดสอบให้กับข้อมูลทดสอบชุดใหม่ และจะตั้งค่าเริ่มต้นใหม่เมื่อจำนวนนับเท่ากับจำนวนใน Selection Register โดยค่าจำนวนนับนั้นเป็นค่าที่ถูกกำหนดด้วยส่วนควบคุมของ BIST หรือ BIST controller จำนวนชุดของข้อมูลใน Selection Register มีจำนวนเท่ากับจำนวนชุดข้อมูลทดสอบ

การเพิ่ม Selection Register ทำให้สามารถควบคุมการโหลดของข้อมูล Seed ได้ง่ายขึ้น และใช้จำนวนน้อยลง เมื่อ Selection Register ควบคุมการทำงานของวงจรนับ ก็จะทำให้สามารถปรับกลไกการเลื่อนข้อมูล Seed ได้ โดยใช้การเลื่อนข้อมูล Seed เพียงบางส่วน เพื่อลดจำนวนของข้อมูล Seed และจำนวนการเลื่อนข้อมูลเข้าสู่วงจร LFSR

4.4 สรุป

การกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ได้นำเสนอในสองแนวคิด แนวคิดแรกนำเสนอการกำเนิดข้อมูลทดสอบวิธี LFSR Reseeding ที่มีการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แบบขนาน เพื่อลดระยะเวลาการทดสอบ สอดคล้องกับโครงสร้างของวงจรเลื่อนเฟสและโครงสร้างของเซลล์ตรวจกวาดแบบ STUMPS ซึ่งเรียกวิธีการนี้ว่า Parallel LFSR Reseeding โดยได้เพิ่มกลุ่มของ XOR เกตเพื่อใช้สำหรับการป้อนกลับข้อมูล Seed เข้าสู่วงจร LFSR และใช้กลุ่มของ AND เกต สำหรับควบคุมการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR ทั้งใช้สัญญาณ *C* สำหรับการเลือกโหมดการทดสอบระหว่าง PRPG และ DTPG เมื่อสัญญาณ *C* มีค่าเท่า “1” เป็นการกำเนิดข้อมูลทดสอบแบบ DTPG และเมื่อ *C* มีค่าเท่า “0” เป็นการกำเนิดข้อมูลทดสอบแบบ PRPG แต่วิธีการดังกล่าวนี้จำเป็นต้องใช้ข้อมูลทดสอบจำนวนมาก จึงได้นำเสนอแนวคิดที่สองการกำเนิดข้อมูล Seed ด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register โดยการประยุกต์ใช้ Selection Register ที่ใช้ในการยกเลิก X สำหรับวงจรวิเคราะห์ผลการทดสอบ เพื่อใช้ควบคุมกลไกการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR โดยการแบ่งกลุ่มข้อมูล Seed ออกเป็น 2 กลุ่ม โดยไม่จำเป็นต้องเลื่อนข้อมูล Seed ทั้ง 2 กลุ่ม เข้าสู่วงจร LFSR ทำให้สามารถเลื่อนหรือไม่เลื่อนข้อมูล Seed ในการกำเนิดข้อมูลทดสอบได้ สำหรับบางชุดข้อมูลทดสอบ ซึ่งจะช่วยลดจำนวนข้อมูล Seed ลง แต่ยังรองรับการทำงานร่วมกับวงจรเลื่อนเฟส เซลล์ตรวจกวาดแบบ STUMPS โดยไม่มีการเปลี่ยนแปลงวงจรทดสอบ

บทที่ 5
ผลการวิจัย

5.1 บทนำ

การออกแบบวงจรกำเนิดข้อมูลการทดสอบของวิทยานิพนธ์ชุดนี้ได้แบ่งออกเป็น 2 แนวคิด คือ การกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding และ การกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register โดยการทำงานของทั้ง 2 แนวคิดได้อธิบายไว้แล้วในบทที่ 4 และบทนี้ได้กล่าวถึง ผลการกำเนิดข้อมูลทดสอบเพื่อทดสอบวงจรมาตรฐาน ISCAS 89 ด้วยวิธีการ Parallel LFSR Reseeding ทั้งสองแนวคิด สำหรับแนวคิดแรก Parallel LFSR Reseeding รวมถึงข้อมูลพื้นฐานของวงจรการทดสอบ ชุดวงจร LFSR ตำแหน่งการ Seed และวงจรเลื่อนเฟสที่ใช้ในการกำเนิดข้อมูลทดสอบของแต่ละวงจรทดสอบได้นำเสนอในหัวข้อ 5.2 ส่วนแนวคิดที่สอง Parallel LFSR Reseeding ร่วมกับ Selection Register ได้นำเสนอในหัวข้อ 5.3 ผลการเปรียบเทียบการลดจำนวนข้อมูล Seed กับวิธีการ Static LFSR Reseeding [26] และวิธีการ Partial LFSR Reseeding [1][44] ได้นำเสนอในหัวข้อที่ 5.4 ส่วนในหัวข้อ 5.5 ได้นำเสนอการลดจำนวนสัญญาณนาฬิกาสำหรับการทดสอบ โดยได้ทำการเปรียบเทียบกับวิธีการจัดเรียงข้อมูล Seed หรือ Seed Ordering [29-30]

5.2 ผลการกำเนิดข้อมูลการทดสอบด้วยแนวคิด Parallel LFSR Reseeding แบบพื้นฐาน

ผลการกำเนิดข้อมูลการทดสอบด้วยแนวคิดแรก Parallel LFSR Reseeding ได้ทำการทดสอบด้วยวงจรมาตรฐาน ISCAS 89 จำนวน 6 วงจร คือ S5378, S9234, S13207, S15850, S38417 และ S38584 ซึ่งรายละเอียดผลการทดสอบของแต่ละวงจรได้แสดงรายละเอียดดังนี้

5.2.1 ผลการกำเนิดข้อมูลการทดสอบวงจร S5378

5.2.1.1 ข้อมูลเบื้องต้นของวงจร S5378

วงจร S5378 เป็นวงจรขนาดเล็กที่มีจำนวนชุดข้อมูลทดสอบทั้งหมด 30 ชุด ใน 1 ชุด มีจำนวนเซลล์ตรวจกวาดด้วยกัน 214 บิต จำนวนเซลล์ตรวจกวาดทั้งหมดจึงเท่ากับ 6,420 บิต จากจำนวนเซลล์ตรวจกวาดทั้งหมดมีจำนวนบิตการเจาะจงค่าเท่ากับ 493 บิต จากจำนวนชุดข้อมูล

ทดสอบทั้งหมด ชุดที่มีจำนวนการเจาะจงค่าสูงที่สุดหรือ S_{max} เท่ากับ 18 บิต ข้อมูลพื้นฐานของวงจร S5378 แสดงดังตารางที่ 5-1

ตารางที่ 5-1 ข้อมูลพื้นฐานของวงจร S5378

ชื่อวงจรทดสอบ	จำนวนเซลล์ตรวจกวาดต่อ 1 ชุดข้อมูลทดสอบ	จำนวนชุดข้อมูลทดสอบ	จำนวนเซลล์ตรวจกวาดทั้งหมด	จำนวนการเจาะจงค่าทั้งหมด	S_{max}
S5378	214	30	6,420	493	18

5.2.1.2 วงจร LFSR สำหรับทดสอบวงจร S5378

จำนวน Flip-Flop ของวงจร LFSR ที่ใช้ในการทดสอบวงจร S5378 อ้างอิงจากเอกสาร [1][44] มีจำนวน Flip-Flop เท่ากับ 38 บิต และมีสมการโพลิเนียลเป็น f(x) = $X^{38}+X^6+X^5+X+1$ สมการโพลิเนียลดังกล่าว สามารถกำเนิดข้อมูลทดสอบได้มีค่าสูงสุดเท่ากับค่า m-sequence คือเท่ากับ $2^{38}-1$ ชุด โดยวงจร LFSR ดังกล่าว มีตำแหน่งการป้อนกลับของเอาท์พุทแบบอนุกรมที่บิตที่ 38, 6, 5 และ 1 การค่าเริ่มต้นของ LFSR ได้ตั้งค่าเริ่มต้นให้บิตที่ 1 มีค่าเท่ากับ “1” ส่วนที่บิตอื่นๆ มีค่าเท่ากับ “0” และการเลื่อนบิตข้อมูลของวงจร LFSR ใช้การวนขวา

5.2.1.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S5378

ชุดข้อมูลที่นำมาสร้างเป็นวงจรเลื่อนเฟส ใช้เอาท์พุทจากวงจร LFSR ในรูปแบบขนาน โดยการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR จำนวน 5,000 ชุด แล้วนำข้อมูลที่กำเนิดได้ ไปคำนวณหาชุดข้อมูลวงจรเลื่อนเฟสที่เหมาะสม ซึ่งค่าตัวแปรต่างๆ ที่ใช้ในการหาชุดวงจรเลื่อนเฟส

B คือ จำนวนเอาท์พุทของวงจรเลื่อนเฟสที่ต้องการ มีค่าเท่ากับ 17 ชุด

L คือ จำนวนน้อยที่สุด ที่จะได้ชุดข้อมูลชุดถัดไป มีค่าเท่ากับ 7 ชุด

I คือ จำนวนหมายเลขหนึ่งที่ปรากฏในชุดข้อมูล มีค่าเท่ากับ 6 บิต

ผลลัพธ์ของชุดข้อมูลวงจรเลื่อนเฟสแสดงดังตารางที่ 5-2 โดยสดมภ์ที่ 1 เป็นลำดับชุดข้อมูล และสดมภ์ที่ 2 ถึง สดมภ์ที่ 8 แสดงชุดข้อมูลวงจรเลื่อนเฟสทั้ง 38 บิต ถูกจัดแบ่งออกเป็นสดมภ์ละ 5 บิต และสดมภ์ที่ 9 ถูกจัดแบ่งออกเป็น 3 บิต

ตารางที่ 5-2 ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 17 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 38 บิต

ลำดับ	บิตที่ 1-5	บิตที่ 6-10	บิตที่ 11-15	บิตที่ 16-20	บิตที่ 21-25	บิตที่ 26-30	บิตที่ 31-35	บิตที่ 36-38
1	10000	00000	10100	01000	00000	00000	00000	001
2	00100	00100	00000	00101	00010	00000	00000	000
3	00001	00001	00001	00000	00001	01000	10000	000
4	00000	01000	00000	00000	00010	00010	00110	000
5	00100	00000	00010	00000	00000	00000	10000	100
6	00000	01100	00000	00000	00100	00000	10010	010
7	01000	00100	00000	00000	00001	00010	10000	000
8	00010	00010	00001	00000	00000	00000	01000	101
9	11000	01000	10000	10000	01000	00000	00000	000
10	10001	00000	01000	00000	00001	00001	00001	000
11	00000	10011	10000	00000	00000	00100	00001	000
12	00100	01001	00000	10000	01000	00000	00010	000
13	00000	00010	11100	00000	00000	00010	10000	000
14	00000	00000	00000	10111	00000	00000	00000	101
15	00100	00000	00100	00000	00100	00100	00100	001
16	00100	10000	01100	01000	00000	00000	00001	000
17	10011	00000	00100	00000	00000	00100	00100	000

5.2.1.4 ตำแหน่งการ Seed สำหรับวงจร S5378

ตำแหน่งการ Seed หรือ tap seed จะสอดคล้องกับชุดข้อมูลของวงจรเลื่อนเฟส ซึ่งจากตารางที่ 5-2 ข้อมูลวงจรเลื่อนเฟสจำนวน 17 ชุด จึงมีจำนวนการวางตำแหน่งการ Seed เท่ากับ 17 ชุด ดังนั้น 1 ชุดวงจรเลื่อนเฟส สอดคล้องกับตำแหน่งการ Seed 1 ตำแหน่ง ซึ่งตำแหน่งการ Seed แต่ละตำแหน่งจะควบคุมข้อมูล Seed แต่ละชุด รวมทั้งหมด 17 ชุด ตำแหน่งการ Seed ที่ใช้ คือ 0, 1, 4, 11, 12, 14, 16, 17, 19, 20 ,21, 23, 26, 28, 29, 31 และ 35

ตำแหน่งการ Seed สำหรับกำเนิดข้อมูลเพื่อทดสอบวงจร S5378 ซึ่งแสดงดังตารางที่ 5-3 สดมภ์ที่ 1 และ 3 แสดงลำดับวงจรเลื่อนเฟส 17 ลำดับ และสดมภ์ที่ 2 และ 4 แสดงตำแหน่งการ seed โดยข้อมูลลำดับที่ 1 ของวงจรเลื่อนเฟสใช้ตำแหน่งการ Seed ตำแหน่งที่ 20 ซึ่งข้อมูล Seed ในลำดับที่ 1 จะถูกเลื่อนเข้าสู่วงจร LFSR ในระหว่างตำแหน่ง Flip-Flop ที่ 20 และ 21 ของวงจร LFSR

ตารางที่ 5-3 ตำแหน่งการ Seed ของวงจร S5378

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
1	20	10	19
2	21	11	26
3	14	12	31
4	17	13	4
5	16	14	23
6	28	15	0
7	12	16	29
8	35	17	1
9	11		

5.2.1.5 สรุปผลการทดสอบของวงจร S5378

ผลการกำเนิดข้อมูลทดสอบของวงจร S5378 ได้ผลลัพธ์แสดงดังตารางที่ 5-4 โดยสดมภ์ที่ 1 แสดงชื่อของวงจรทดสอบคือ S5378 สดมภ์ที่ 2 แสดงผลจำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ จำนวนบิตข้อมูลทดสอบที่ใช้คือ 17 บิตต่อ 1 ชุดข้อมูลทดสอบ สดมภ์ที่ 3 แสดงจำนวนข้อมูล Seed ทั้งหมดในการทดสอบ ส่วนสดมภ์ที่ 4 แสดงเปอร์เซ็นต์การลดขนาดข้อมูลการทดสอบ โดยการจำนวนบิต Seed ที่ใช้เปรียบเทียบกับจำนวนเซลล์ตรวจกวาดทั้งหมด ซึ่งสมการคำนวนหาเปอร์เซ็นต์การลดขนาดข้อมูล Seed แสดงดังสมการ (5-1) และสดมภ์สุดท้ายแสดงจำนวนสัญญาณนาฬิกาที่ใช้ในการกำเนิดข้อมูลทดสอบต่อ 1 ชุดข้อมูลทดสอบ เท่ากับ 13 สัญญาณนาฬิกา สมการคำนวณสัญญาณนาฬิกาที่ใช้ในการทดสอบ แสดงดังสมการ (5-2)

$$\text{เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ} = \frac{(\text{จำนวนเซลล์ตรวจกวาดทั้งหมด} - \text{จำนวนข้อมูล Seed}) \times 100}{\text{จำนวนเซลล์ตรวจกวาดทั้งหมด}} \quad (5\text{-}1)$$

จำนวนสัญญาณนาฬิกาต่อ 1 ชุดข้อมูลทดสอบ = $\frac{\text{จำนวนเซลล์ตรวจกวาด 1 ชุด}}{\text{จำนวนเอาท์พุทจากวงจรเลื่อนเฟส}}$ (5-2)

ตารางที่ 5-4 ผลการทดสอบวงจร S5378

ชื่อวงจรทดสอบ	จำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ	จำนวนบิต Seedทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ	จำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบ
S5378	17	510	92.05%	13

5.2.2 ผลการกำเนิดข้อมูลการทดสอบวงจร S9234

5.2.2.1 ข้อมูลเบื้องต้นของวงจร S9234

วงจร S9234 เป็นวงจรขนาดเล็ก มีจำนวนชุดข้อมูลทดสอบทั้งหมด 138 ชุด ใน 1 ชุดข้อมูลทดสอบมีจำนวนเซลล์ตรวจกวาดจำนวน 247 บิต จำนวนเซลล์ตรวจกวาดทั้งหมดจึงเท่ากับ 34,086 บิต จำนวนบิตการเจาะจงค่าเท่ากับ 4,674 บิต และในแต่ละชุดข้อมูลทดสอบมีจำนวนการเจาะจงค่าสูงที่สุด หรือ S_{max} เท่ากับ 61 บิต ข้อมูลพื้นฐานของวงจร S9234 แสดงดังตารางที่ 5-5

ตารางที่ 5-5 ข้อมูลพื้นฐานของวงจร S9234

ชื่อวงจรทดสอบ	จำนวนเซลล์ตรวจกวาดต่อ 1 ชุดข้อมูลทดสอบ	จำนวนชุดข้อมูลทดสอบ	จำนวนเซลล์ตรวจกวาดทั้งหมด	จำนวนการเจาะจงค่าทั้งหมด	S_{max}
S9234	247	138	34,086	4,674	61

5.2.2.2 วงจร LFSR สำหรับทดสอบวงจร S9234

วงจร LFSR ที่ใช้ในการทดสอบวงจร S9234 อ้างอิงจากเอกสาร [1][44] มีจำนวน Flip-Flop เท่ากับ 81 บิต มีสมการโพลิเนียลเป็น $f(x) = X^{81}+X^{77}+1$ ซึ่งเป็นสมการที่สามารถกำเนิดข้อมูลทดสอบได้มีค่าสูงสุด ที่มีจำนวน n เท่ากับ 81 คือเท่ากับ 2^{81}-1 ชุด โดยวงจร LFSR ดังกล่าวมี

ตำแหน่งการป้อนกลับของเอาท์พุทแบบอนุกรมที่บิตที่ 81, 77 และ 0 โดยค่าเริ่มต้นของ LFSR ใช้การตั้งค่าเริ่มต้นให้บิตที่ 1 มีค่าเท่ากับ “1” ส่วนที่บิตอื่นๆ มีค่าเท่ากับ “0”

5.2.2.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S9234

ชุดข้อมูลที่นำมาสร้างเป็นวงจรเลื่อนเฟส ใช้เอาท์พุทจากวงจร LFSR ในรูปแบบขนาน โดยการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR จำนวน 10,000 ชุด แล้วนำข้อมูลที่กำเนิดได้ไปคำนวณหาชุดข้อมูลวงจรเลื่อนเฟสที่เหมาะสม ค่าตัวแปรที่นำมาใช้ในการคำนวณชุดวงจรทดสอบคือ B มีค่าเท่ากับ 59 ชุด, L มีค่าเท่ากับ 23 ชุด และ I มีค่าเท่ากับ 15 บิต ผลลัพธ์ของชุดข้อมูลวงจรเลื่อนเฟสแสดงดังตารางที่ 5-6 โดยสดมภ์ที่ 1 เป็นลำดับชุดข้อมูล และสดมภ์ที่ 2 ถึงสดมภ์ที่ 9 คือชุดข้อมูลวงจรเลื่อนเฟสทั้ง 81 บิต ถูกจัดแบ่งออกเป็นสดมภ์ละ 10 บิต ส่วนสดมภ์ที่ 10 ถูกจัดแบ่งออกเป็น 11 บิต

ตารางที่ 5-6 ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 59 ชุดที่กำเนิดจากวงจร LFSR ขนาด 81 บิต

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-50	บิตที่ 51-60	บิตที่ 61-70	บิตที่ 71-81
1	0100010001	0001000100	0100000000	0000000000	0000000000	0000000000	0000000000	00001000000
2	0000000000	0000000000	0000000010	0010001000	1000100010	0010001000	0000000000	00000000000
3	0000000000	0000100000	0000000000	0000000000	0000000000	0000000000	0001000100	01000100010
4	0000000000	0000001000	1000100010	0000000000	0000000000	0000000000	0000010000	00010000000
5	0000000000	0000000001	0001000100	0100000000	0000000000	0100010001	0001000000	00000000000
6	0000010000	0000000000	0100000000	0000000000	0000000000	0000100010	0010001000	00000000000
7	0100000000	0001000100	0000000001	0001000000	0000010001	0000000000	0000000000	00000000100
8	0000000100	0000000000	0000000000	0010001000	0000000010	0010000000	0000100010	00000000001
9	0010000000	1000000010	0000001000	0000100000	0010000000	0000000000	0000000100	01000000000
10	1000000000	0000000000	0100000001	0000000100	0000010000	0001000000	0100000001	00000001000
11	0001000100	0100010001	0001000100	0100010000	0000000000	0000001000	0000100000	00100000000
12	0000000000	0000000000	0000000010	0000000000	0000010001	0001000100	0100010001	00010001000
13	0000000000	0000000000	0000000000	1000100010	0010000100	0000010000	0000000000	00000000000
14	1100010001	0001000000	0000000000	0000000000	0000000000	0000000000	0000010001	00010001000
15	0000000000	0000000001	0000000000	0000011000	1000100010	0000000000	0000000000	00000000000
16	0000001000	0000000000	0000000000	0000000000	0000000000	0000100000	0000000000	11000100010
17	0000000010	0010000000	0000110010	0000000000	0100000000	0000000000	0000000000	00000000000
18	0000000000	0000000000	0000000000	0000000000	0001000100	0000000001	1001000000	00000010000
19	0000010000	0001100010	0100000000	1000100000	0000000000	0000000000	0000000000	00000000100
20	0000000000	0000000000	0000000000	0010000000	1000000011	0001001000	0000010001	00000000000
21	0000000000	0000000000	0100010001	0001000110	0100011001	0000100000	0010000000	00000000000

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-50	บิตที่ 51-60	บิตที่ 61-70	บิตที่ 71-81
22	1000100110	0010001000	1000000000	0000000000	0000000000	0000001000	1000100010	00110010001
23	0000000000	1000000000	0000000100	0100010001	0011000100	0100010000	0000000000	00000000000
24	1000000000	0001000000	0000000000	0000000000	0000010000	0000000000	0010001000	10001001100
25	0010001000	0000000000	0000000000	0000100010	0000000000	0100000000	0000001000	10000000000
26	0000010001	0000000000	1000000010	0000000100	0100000000	0000000000	0000000001	00010000000
27	0000000000	0000000000	0001000000	0100000000	1000100000	0000010000	0001000000	00100010000
28	0010000001	0001000100	0100001000	0000100000	0000000000	0000000010	0000001000	00000100010
29	0000000000	0010001000	1000100001	0000000100	0000100010	0010001000	0100000001	00000000000
30	0000000000	0000110001	0001000100	0000000000	0000000100	0100010001	0000100000	00100000010
31	0100000000	0000000110	0010001000	1001000000	0000000001	1000100010	0010000000	00000000000
32	0001100100	0000000000	1000000000	0000001000	0000000000	0011000100	0100010010	00000000000
33	0000001100	1000000000	0011001000	0000000011	0010000000	0000010000	0000000000	01000000000
34	1000000001	0001000000	0000010001	0000000000	0110010000	0000000110	0100000000	00011001000
35	0000000000	1000100000	0000000000	0000000000	0000100010	0010001000	1000100010	00100010001
36	0000000000	0000000000	0001000100	0100010000	0000000000	0000000100	0100000000	00000000000
37	0000000000	0100000001	0000000000	0000000000	0000000000	0000000010	0010001000	10000000000
38	1000100010	0010000000	0000000000	0000000000	0000000100	0000000000	0001000000	00000000000
39	0100010000	0000000100	0100000000	0000010001	0001000100	0000000000	0000000000	00000000000
40	0000010000	0000000000	0000000000	0000001000	1000000000	0010001000	0000000000	10001000100
41	0000001000	0000100000	0000100000	0000000000	1000000000	0000000000	0000000000	01000100000
42	0000000000	0000000000	0000010000	0001000000	0100000001	0000000001	0000000000	00000100000
43	0001000100	0001000100	0000000001	0001000000	0000000000	0000000000	1000000010	00000010000
44	0000000000	0000100010	0010001000	1000100010	0010000010	0010000000	0000100010	00000000000
45	0001000000	0000000000	0000000000	0000010100	0000010000	0001000000	0100000000	00000000000
46	0000000000	0000000000	0000101010	1000100010	0010001000	1000100000	0000000000	00010000000
47	0000000000	0010000000	0000000100	0100000000	0000000000	0000000001	0101010001	00010001000
48	0100010000	0000001000	0000100000	0000000000	0000000010	1000001000	0000000000	00000000000
49	0000010001	0001000100	0000000000	0000000101	0101010001	0000000000	0000000000	01000100000
50	0100000100	0000000000	0000010000	0001000000	1000100010	0010000000	0000000000	00101010101
51	0000000000	1000000000	0000001010	0000001000	0010000000	0000000000	1000000010	00000100010
52	0001000000	0000000011	0010000000	0000100010	0000000000	1010101000	1000101000	10000000000
53	0000000110	0010010000	0001000000	0100000001	0100000100	0000010100	0001000000	00010000000
54	0000000000	0000000001	0000000100	0000010000	0001000001	0100000001	0000001100	10001100100
55	0000000000	0000000000	0000000100	0000010000	0000100010	0000000000	0100000000	00000000100
56	0000000000	0000001000	1000100010	0001000000	0100000000	1000100000	0000000100	00000000000
57	0000100000	0010000000	0100010000	0000000000	0000000000	0100010001	0001000010	00000010000
58	0100010000	1000000010	0000000000	0000000000	1000000000	0000001100	0100010001	00001000000
59	0001000100	0100000000	0000000001	0001000000	0000011001	0000000000	0011000100	01000100001

5.2.2.4 ตำแหน่งการ Seed สำหรับวงจร S9234

ตำแหน่งการ Seed ของวงจร S9234 จะขึ้นอยู่กับตำแหน่งข้อมูลของวงจรเลื่อนเฟส จากตารางที่ 5-6 ข้อมูลวงจรเลื่อนเฟสจำนวน 59 ชุด มีจำนวนการวางตำแหน่งการ Seed เท่ากับ 59 ชุด ทำให้ 1 ชุดวงจรเลื่อนเฟส มีความสัมพันธ์กับตำแหน่งการ Seed จำนวน 1 ตำแหน่งเพื่อทดสอบวงจร S9234 คือ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 35, 36, 37, 38, 39, 40, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 56, 57, 58, 59, 60 , 61, 62, 65, 66, 71, 75 และ 78

ตำแหน่งการ Seed เพื่อทดสอบวงจร S9234 ซึ่งแสดงดังตารางที่ 5-7 สดมภ์ที่ 1 และ 3 แสดงลำดับวงจรเลื่อนเฟส และสดมภ์ที่ 2 และ 4 แสดงตำแหน่งการ Seed โดยข้อมูลลำดับที่ 1 ของวงจรเลื่อนเฟสใช้ตำแหน่งการ Seed ตำแหน่งที่ 13 ซึ่งข้อมูล Seed ในลำดับที่ 1 จะถูกเลื่อนเข้าสู่วงจร LFSR ระหว่างตำแหน่ง Flip-Flop ที่ 13 และ 14 ของวงจร LFSR ตำแหน่งการ Seed ทุกตำแหน่งจะไม่ซ้ำกัน

ตารางที่ 5-7 ตำแหน่งการ Seed ของวงจร S9234

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
1	13	31	1
2	32	32	53
3	14	33	22
4	16	34	25
5	19	35	44
6	5	36	27
7	29	37	11
8	7	38	47
9	2	39	39
10	0	40	36
11	3	41	24
12	28	42	33
13	30	43	60
14	65	44	48

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
15	35	45	37
16	6	46	26
17	8	47	59
18	43	48	56
19	18	49	71
20	49	50	40
21	21	51	75
22	4	52	52
23	10	53	57
24	62	54	66
25	38	55	78
26	9	56	50
27	23	57	51
28	58	58	61
29	12	59	46
30	15		

5.2.2.5 สรุปผลการทดสอบของวงจร S9234

ผลการกำเนิดข้อมูลทดสอบของวงจร S9234 ได้ผลลัพธ์แสดงดังตารางที่ 5-8 โดยสดมภ์ที่ 1 แสดงชื่อของวงจรทดสอบ คือ S9234 สดมภ์ที่ 2 แสดงผลจำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ ซึ่งจำนวนข้อมูลทดสอบที่ใช้คือ 59 บิตต่อ 1 ชุดข้อมูลทดสอบ สดมภ์ที่ 3 แสดงจำนวนข้อมูล Seed ทั้งหมดในการทดสอบซึ่งเท่ากับ 8,142 บิต สดมภ์ที่ 4 แสดงเปอร์เซ็นต์การลดขนาดข้อมูลการทดสอบ โดยจำนวนบิต Seed ที่ใช้เปรียบเทียบกับจำนวนเซลล์ตรวจกวาดทั้งหมด ผลลัพธ์ที่ได้เท่ากับ 76.11% ส่วนสดมภ์ที่ 5 แสดงจำนวนสัญญาณนาฬิกาต่อ 1 ชุดข้อมูลทดสอบ มีค่าเท่ากับ 5 สัญญาณนาฬิกา โดยคำนวณด้วยสมการ (5-2)

ตารางที่ 5-8 ผลการทดสอบวงจร S9234

ชื่อวงจรทดสอบ	จำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ	จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ	จำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบ
S9234	59	8,142	76.11%	5

5.2.3 ผลการกำเนิดข้อมูลการทดสอบวงจร S13207

5.2.3.1 ข้อมูลเบื้องต้นของวงจร S13207

วงจร S13207 เป็นวงจรขนาดกลาง มีจำนวนชุดข้อมูลทดสอบทั้งหมด 157 ชุด ใน 1 ชุดข้อมูลทดสอบมีจำนวนเซลล์ตรวจกวาดจำนวน 700 บิต จำนวนเซลล์ตรวจกวาดทั้งหมดจึงเท่ากับ 109,900 บิต และจำนวนบิตการเจาะจงค่าเท่ากับ 2,824 บิต เมื่อเปรียบเทียบในแต่ละชุดข้อมูลทดสอบ จำนวนบิตการเจาะจงค่าสูงที่สุด หรือ S_{max} เท่ากับ 24 บิต ข้อมูลพื้นฐานของวงจร S13207 แสดงดังตารางที่ 5-9

ตารางที่ 5-9 ข้อมูลพื้นฐานของวงจร S13207

ชื่อวงจร	จำนวนเซลล์ตรวจกวาดต่อ 1 ชุดข้อมูลทดสอบ	จำนวนชุดข้อมูลทดสอบ	จำนวนเซลล์ตรวจกวาดทั้งหมด	จำนวนการเจาะจงค่าทั้งหมด	S_{max}
S13207	700	157	109,900	2,824	24

5.2.3.2 วงจร LFSR สำหรับทดสอบวงจร S13207

วงจร LFSR ที่ใช้ในการทดสอบวงจร S13207 อ้างอิงจากเอกสาร [1][44] ซึ่งจำนวน Flip-Flop ที่ใช้เท่ากับ 44 บิต มีสมการโพลิเนียลเป็น $f(x) = X^{44}+X^{43}+ X^{18}+ X^{17}+1$ เป็นสมการที่สามารถกำเนิดข้อมูลทดสอบได้มีค่าสูงที่สุดเท่ากับ 2^{44}-1 ชุด วงจร LFSR ที่ใช้ในการทดสอบมีตำแหน่งการป้อนกลับของเอาท์พุทแบบอนุกรมที่บิตที่ 44, 43, 18 และ17 โดยค่าเริ่มต้นของ LFSR ใช้การตั้งค่าเริ่มต้นให้บิตที่ 1 มีค่าเท่ากับ “1” ส่วนที่บิตอื่นๆ มีค่าเท่ากับ “0”

5.2.3.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S13207

ชุดข้อมูลที่นำมาสร้างเป็นวงจรเลื่อนเฟส ใช้เอาท์พุทจากวงจร LFSR ในรูปแบบขนาน โดยการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR จำนวน 5,000 ชุด แล้วนำข้อมูลที่กำเนิดได้ไปคำนวณหาชุดข้อมูลวงจรเลื่อนเฟสที่เหมาะสม โดยค่าเริ่มต้นของ LFSR ใช้การตั้งค่าเริ่มต้นให้บิตที่ 0 มีค่าเท่ากับ “1” ส่วนที่บิตอื่นๆ มีค่าเท่ากับ “0” โดยค่าตัวแปร B, L และ I มีค่าเท่ากับ 24, 7 และ 13 ตามลำดับ ผลลัพธ์ของชุดข้อมูลวงจรเลื่อนเฟสแสดงดังตารางที่ 5-10 โดยสดมภ์ที่ 1 เป็นลำดับชุดข้อมูล ส่วนสดมภ์ที่ 2 ถึง สดมภ์ที่ 6 แสดงชุดข้อมูลวงจรเลื่อนเฟสทั้ง 44 บิต ที่ถูกจัดแบ่งออกเป็นสดมภ์ละ 10 บิตและสดมภ์สุดท้าย 4 บิต

ตารางที่ 5-10 ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 24 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 44 บิต

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-44
1	1001010000	0000010100	0100000000	0000000000	0000
2	0011110010	1000000000	1010001000	0000000000	0000
3	0000000111	1001010000	0000010100	0100000000	0000
4	0110000000	0100111100	0000001001	0000100000	0010
5	0010010100	0011000010	0000010000	0010000010	0100
6	0000110010	1000000011	0000010100	0000100100	0010
7	0110000000	0010000101	0000000010	0000000110	0010
8	0001001100	0000000100	0010100000	0001000000	0011
9	0001100010	0110000000	0010000101	0000000010	0000
10	0000000011	0001001100	0000000100	0010100000	0001
11	0000001000	0000100000	0000011000	0001011100	1101
12	0011100000	0100000001	0000000000	1100000010	1110
13	1100100111	0000001000	0000100000	0000011000	0001
14	0001011001	0011100000	0100000001	0000000000	1100
15	0000100010	0001000001	0101000000	0000101010	0100
16	0000010000	0000000010	0000111100	0000010001	0111
17	1110000000	1000000000	0001000001	1110000000	1000

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-44
18	0110011100	0000010000	0000000010	0000111100	0000
19	0000100110	0110011100	0000010000	0000000010	0000
20	0010000001	0011001100	1110000000	1000000000	0001
21	0000001000	1000000011	1010011000	0000000101	0010
22	0000001001	0000001001	0010001010	0000000001	1101
23	1001000100	0000001000	0010001000	0000010001	0001
24	0101000000	0110000010	0000001000	0000000010	1100

5.2.3.4 ตำแหน่งการ Seed สำหรับวงจร S13207

ตำแหน่งการ Seed ของวงจร S13207 จะขึ้นอยู่กับการตำแหน่งข้อมูลของวงจรเลื่อนเฟส จากตารางที่ 5-11 ข้อมูลวงจรเลื่อนเฟสจำนวน 24 ชุด มีจำนวนการวางตำแหน่งการ Seed เท่ากับ 24 ชุด ทำให้ 1 ชุดวงจรเลื่อนเฟส มีความสัมพันธ์กับตำแหน่งการ Seed 1 ตำแหน่ง ซึ่งตำแหน่งการ Seed แต่ละตำแหน่งจะควบคุมข้อมูล Seed แต่ละชุด ทั้งหมด 24 ชุด ตำแหน่งของการ Seed เพื่อทดสอบวงจร S13207 คือ 0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 16, 17, 18, 19, 22, 24, 26, 30, 32, 33, 39 และ 40

จากตำแหน่งของ tap seed เพื่อทดสอบวงจร S13207 ซึ่งแสดงดังตารางที่ 5-10 สดมภ์ที่ 1 และ 3 แสดงลำดับวงจรเลื่อนเฟส และสดมภ์ที่ 2 และ 4 มาแสดงตำแหน่งการ Seed โดยข้อมูลลำดับที่ 2 ของวงจรเลื่อนเฟสใช้ตำแหน่งการ Seed ตำแหน่งที่ 17 ซึ่งข้อมูล Seed ในลำดับที่ 2 จะถูกเลื่อนเข้าสู่วงจร LFSR ในระหว่างตำแหน่ง Flip-Flop ที่ 17 และ 18 ของวงจร LFSR

ตารางที่ 5-11 ตำแหน่งการ Seed ของวงจร S13207

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
1	16	13	2
2	17	14	0
3	8	15	5
4	13	16	10
5	19	17	9
6	22	18	39

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
7	12	19	33
8	32	20	1
9	24	21	6
10	18	22	30
11	4	23	26
12	7	24	40

5.2.3.5 สรุปผลการทดสอบของวงจร S13207

ผลการกำเนิดข้อมูลทดสอบของวงจร S13207 ได้ผลลัพธ์แสดงดังตารางที่ 5-12 โดยสดมภ์ที่ 1 แสดงชื่อของวงจรทดสอบ คือ S13207 สดมภ์ที่ 2 แสดงผลจำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ ซึ่งจำนวนชุดข้อมูลทดสอบที่ใช้คือ 44 บิตต่อ 1 ชุดข้อมูลทดสอบ สดมภ์ที่ 3 แสดงจำนวนข้อมูล Seed ทั้งหมดในการทดสอบ ซึ่งเท่ากับ 8,142 บิต สดมภ์ที่ 4 แสดงเปอร์เซ็นต์การลดขนาดข้อมูลการทดสอบ โดยการจำนวนบิต Seed ที่ทั้งหมดเปรียบเทียบกับจำนวนเซลล์ตรวจกวาดทั้งหมด ผลลัพธ์ที่ได้มีค่าเท่ากับ 96.57% ส่วนสดมภ์ที่ 5 แสดงจำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบ เท่ากับ 30 สัญญาณนาฬิกา

ตารางที่ 5-12 ผลการทดสอบวงจร S13207

ชื่อวงจรทดสอบ	จำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ	จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ	จำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบ
S13207	24	3,768	96.57 %	30

5.2.4 ผลการกำเนิดข้อมูลการทดสอบวงจร S15850

5.2.4.1 ข้อมูลเบื้องต้นของวงจร S15850

วงจร S15850 เป็นวงจรขนาดกลาง เนื่องจากมีจำนวนชุดข้อมูลทดสอบทั้งหมด 167 ชุด และใน 1 ชุดข้อมูลทดสอบมีจำนวนเซลล์ตรวจกวาดจำนวน 611 บิต จำนวนเซลล์ตรวจกวาดทั้งหมดจึงเท่ากับ 102,037 บิต จำนวนบิตการเจาะจงค่าเท่ากับ 9,686 บิต เมื่อเปรียบเทียบ

จำนวนการเจาะจงค่าของชุดข้อมูลการทดสอบ จำนวนการเจาะจงค่าที่สูงที่สุดเท่ากับ 38 บิต ข้อมูลพื้นฐานของวงจร S15850 แสดงดังตารางที่ 5-13

ตารางที่ 5-13 ข้อมูลพื้นฐานของวงจร S15850

ชื่อวงจรทดสอบ	จำนวนเซลล์ตรวจกวาดต่อ 1 ชุดข้อมูลทดสอบ	จำนวนชุดข้อมูลทดสอบ	จำนวนเซลล์ตรวจกวาดทั้งหมด	จำนวนการเจาะจงค่าทั้งหมด	S_{max}
S15850	611	167	102,037	4,674	38

5.2.4.2 วงจร LFSR สำหรับทดสอบวงจร S15850

วงจร LFSR ที่ใช้ในการทดสอบวงจร S15850 อ้างอิงจากเอกสาร [1][44] ซึ่งจำนวน Flip-Flop ที่ใช้เท่ากับ 58 บิต มีสมการโพลิเนียลเป็น $f(x) = X^{58}+X^{39}+1$ ซึ่งเป็นสมการที่สามารถกำเนิดข้อมูลทดสอบได้มีค่าสูงที่สุดเท่ากับ 2^{58}-1 โดยวงจร LFSR ดังกล่าวมีตำแหน่งการป้อนกลับของเอาท์พุทแบบอนุกรมที่บิตที่ 58 และ 39 โดยค่าเริ่มต้นของ LFSR ใช้การตั้งค่าเริ่มต้นให้บิตที่ 1 มีค่าเท่ากับ “1” ส่วนที่บิตอื่นๆ มีค่าเท่ากับ “0”

5.2.4.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S15850

ชุดข้อมูลที่นำมาสร้างเป็นวงจรเลื่อนเฟส ใช้เอาท์พุทจากวงจร LFSR ในรูปแบบขนาน โดยการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR จำนวน 10,000 ชุด แล้วนำข้อมูลที่กำเนิดได้ไปคำนวณหาชุดข้อมูลวงจรเลื่อนเฟสที่เหมาะสม ค่าตัวแปรต่างๆ ที่ใช้ในการคำนวณหาชุดข้อมูลวงจรเลื่อนเฟสแสดงดังนี้

B คือ จำนวนเอาท์พุทของวงจรเลื่อนเฟสที่ต้องการ มีค่าเท่ากับ 34 ชุด

L คือ จำนวนน้อยที่สุด ที่จะได้ชุดข้อมูลชุดถัดไป มีค่าเท่ากับ 21 ชุด

I คือ จำนวนหมายเลขหนึ่งที่ปรากฏในชุดข้อมูล มีค่าเท่ากับ 15 บิต

ผลลัพธ์ของชุดข้อมูลวงจรเลื่อนเฟสแสดงดังตารางที่ 5-14 โดยสดมภ์ที่ 1 เป็นลำดับชุดข้อมูล มีจำนวนชุดข้อมูลทั้งหมดเท่ากับ 34 ชุด ขนาด 58 บิตของวงจร LFSR แสดงดังสดมภ์ที่ 2 ถึง สดมภ์ที่ 7 ถูกจัดแบ่งออกเป็นสดมภ์ละ 10 บิตและสดมภ์สุดท้าย 8 บิต

ตารางที่ 5-14 ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 34 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 58 บิต

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-50	บิตที่ 51-58
1	1101100000	0000101000	1010100000	0001111001	1000000000	00000000
2	0101000101	1000000000	1010001000	1100000000	0101000110	10000000
3	0000001010	0010110000	0000010100	0100011000	0000001010	00110100
4	1100000000	0101000101	1000000000	1010001000	1100000000	01010001
5	0000000000	1111000000	0100010101	1100000000	0101000101	10000000
6	0000000000	0000011110	0000001000	1010111000	0000001010	00101100
7	1000000000	0000000000	1111000000	0100010101	1100000000	01010001
8	0000100000	0011011011	1000000000	0000000000	1111000000	01000101
9	0010100001	0000000110	1101110000	0000000000	0000011110	00000010
10	0001000101	0000100000	0011011011	1000000000	0000000000	11110000
11	1000000010	0010100001	0000000110	1101110000	0000000000	00000111
12	0001010000	1100000000	0110000010	0001001001	1000000011	00101000
13	1100001100	0001010000	1100000000	0110000010	0001001001	10000000
14	0010100010	0011001101	0000000000	0010000000	1010010001	10001000
15	0100000101	0001000110	0110100000	0000000100	0000010100	10001100
16	0000011000	0001010001	0000110101	0000000001	0101100000	00000101
17	0010100000	1100000010	1000100001	1010100000	0000101011	00000000
18	0001010000	0010100000	1100000010	1000100001	1010100000	00001010
19	1001000001	0000100000	0000010011	0001110000	1000111000	10000000
20	0000000010	1000000100	0001100000	1010010001	1110010001	00000100
21	0000000010	0010110010	0000011010	0010000011	1000100001	00001000
22	0001100000	0000100000	0000000000	1000000010	1100101010	01101110
23	0100000110	0001000000	0101000100	1101000001	0000000100	01001001
24	0100101000	0011000010	0000001010	0010011010	0000100000	00100010
25	0110000000	0010100000	0000010001	0011001001	0000000001	10110010
26	0000001000	0111000000	1001000010	0000100000	0101111001	10000000

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-50	บิตที่ 51-58
27	0000000010	0100001000	0010100011	0000001010	0101001010	00010001
28	1010000000	0001001000	0100000101	0001100000	0101001010	01010000
29	0000010100	0000000010	0100001000	0010100011	0000001010	01010010
30	0001100000	1010000000	0001001000	0100000101	0001100000	01010010
31	0011010000	0001100000	1010000000	0001001000	0100000101	00011000
32	1110000110	1000000011	0000010100	0000000010	0100001000	00101000
33	0001001000	0000101001	1110000110	1000000011	0000010100	00000000
34	0010001110	1010001000	1000000001	0000000110	0001000010	00010001

5.2.4.4 ตำแหน่งการ Seed สำหรับวงจร S15850

ตำแหน่งการ Seed ของวงจร S15850 จะขึ้นอยู่กับตำแหน่งข้อมูลของวงจรเลื่อนเฟส จากข้อมูลในตารางที่ 5-14 วงจรเลื่อนเฟสจำนวน 34 ชุด ทำให้มีจำนวนตำแหน่งการ Seed เท่ากับ 34 ตำแหน่ง กล่าวคือ 1 ชุดของวงจรเลื่อนเฟส มีตำแหน่งของการ Seed จำนวน 1 ตำแหน่ง ซึ่งตำแหน่งการ Seed แต่ละตำแหน่งจะควบคุมข้อมูล Seed แต่ละชุด ทั้งหมด 34 ชุด ตำแหน่งการ Seed เพื่อทดสอบวงจร S15850 คือ 0, 1, 2, 3, 6, 7, 8, 14, 20, 27,19, 13, 15, 16, 18, 21, 23, 26, 28, 29, 30, 33, 35, 36, 37, 38, 39, 41, 42, 44, 45, 51, 54 และ56 ตำแหน่งการ Seed เพื่อทดสอบวงจร S15850 แสดงดังตารางที่ 5-15 สดมภ์ที่ 1 และ 3 แสดงลำดับวงจรเลื่อนเฟส และสดมภ์ที่ 2 และ 4 แสดงตำแหน่งการ Seed โดยข้อมูลลำดับที่ 10 ของวงจรเลื่อนเฟสใช้ตำแหน่งการ Seed ตำแหน่งที่ 28 ซึ่งข้อมูล Seed ในลำดับที่ 10 จะถูกเลื่อนเข้าสู่วงจร LFSR ในระหว่างตำแหน่ง Flip-Flop ที่ 28 และ 29 ของวงจร LFSR

ตารางที่ 5-15 ตำแหน่งการ Seed ของวงจร S15850

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
1	14	18	42
2	20	19	45
3	27	20	8

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
4	19	21	54
5	29	22	41
6	30	23	51
7	35	24	1
8	18	25	2
9	23	26	6
10	28	27	36
11	56	28	0
12	33	29	26
13	15	30	3
14	16	31	13
15	21	32	7
16	39	33	38
17	44	34	37

5.2.4.5 สรุปผลการทดสอบของวงจร S15850

ผลการกำเนิดข้อมูลทดสอบของวงจร S15850 ได้ผลลัพธ์แสดงดังตารางที่ 5-16 โดยสดมภ์ที่ 1 แสดงชื่อของวงจรทดสอบ คือ S15850 สดมภ์ที่ 2 แสดงผลจำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ ซึ่งจำนวนชุดข้อมูลทดสอบที่ใช้คือ 34 บิตต่อ 1 ชุดข้อมูลทดสอบ สดมภ์ที่ 3 แสดงจำนวนข้อมูล Seed ทั้งหมดในการทดสอบ ซึ่งเท่ากับ 5,678 บิต สดมภ์ที่ 4 แสดงเปอร์เซ็นต์การลดขนาดข้อมูลการทดสอบ โดยการคำนวนจากจำนวนบิต Seed ทั้งหมดเปรียบเทียบกับจำนวนเซลล์ตรวจกวาดทั้งหมด ผลลัพธ์ที่ได้เท่ากับ 94.43% ส่วนสดมภ์ที่ 5 แสดงจำนวนสัญญาณนาฬิกาที่ใช้ เท่ากับ 18 สัญญาณนาฬิกา จำนวนสัญญาณนาฬิกาที่ใช้สามารถคำนวณได้จากสมการ (5-2)

ตารางที่ 5-16 ผลการทดสอบวงจร S15850

ชื่อวงจรทดสอบ	จำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ	จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ	จำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบ
S15850	34	5,678	94.43%	18

5.2.5 ผลการกำเนิดข้อมูลการทดสอบวงจร S38417

5.2.5.1 ข้อมูลเบื้องต้นของวงจร S38417

วงจร S38417 เป็นวงจรขนาดใหญ่มาก เมื่อเทียบกับวงจรทดสอบอื่นๆ ซึ่งวงจร S38417 มีจำนวนชุดข้อมูลทดสอบทั้งหมด 340 ชุด ใน 1 ชุดข้อมูลทดสอบมีจำนวนเซลล์ตรวจสอบจำนวน 1,664 บิต จำนวนเซลล์ตรวจกวาดทั้งหมดจึงเท่ากับ 565,760 บิต จำนวนบิตการเจาะจงค่าเท่ากับ 4,674 บิต และจำนวนการเจาะจงค่าที่สูงที่สุด หรือ S_{max} เท่ากับ 85 บิต ข้อมูลพื้นฐานของวงจร S38417 แสดงดังตารางที่ 5-17

ตารางที่ 5-17 ข้อมูลพื้นฐานของวงจร S38417

ชื่อวงจรทดสอบ	จำนวนเซลล์ตรวจกวาดต่อ 1 ชุดข้อมูลทดสอบ	จำนวนชุดข้อมูลทดสอบ	จำนวนเซลล์ตรวจกวาดทั้งหมด	จำนวนการเจาะจงค่าทั้งหมด	S_{max}
S38417	1,664	340	565,760	23,984	85

5.2.5.2 วงจร LFSR สำหรับทดสอบวงจร S38417

วงจร LFSR ที่ใช้ในการทดสอบวงจร S38417 อ้างอิงจากเอกสาร [1][44] จำนวน Flip-Flop ที่ใช้เท่ากับ 105 บิต มีสมการโพลิเนียลเป็น $f(x) = X^{105}+X^{89}+1$ ซึ่งเป็นสมการที่สามารถกำเนิดข้อมูลทดสอบได้มีค่าสูงที่สุดเท่ากับ $2^{105}-1$ โดยวงจร LFSR ดังกล่าวมีตำแหน่งการป้อนกลับของเอาท์พุทแบบอนุกรมที่บิตที่ 105 และ 89 โดยค่าเริ่มต้นของ LFSR ใช้การตั้งค่าเริ่มต้นให้บิตที่ 0 มีค่าเท่ากับ “1” ส่วนที่บิตอื่นๆ มีค่าเท่ากับ “0” และการเลื่อนบิตข้อมูลของวงจร LFSR ใช้การวนขวา

5.2.5.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S38417

ชุดข้อมูลที่นำมาสร้างเป็นวงจรเลื่อนเฟส ใช้เอาท์พุทจากวงจร LFSR ในรูปแบบขนาน โดยการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR จำนวน 15,000 ชุด แล้วนำข้อมูลที่กำเนิดได้ไปคำนวณหาชุดข้อมูลวงจรเลื่อนเฟสที่เหมาะสม ค่าของตัวแปรที่ใช้ในการคำนวณ B มีค่าเท่ากับ 83 ชุด, L มีค่าเท่ากับ 30 ชุด และ I มีค่าเท่ากับ 17 บิต ผลลัพธ์ของชุดข้อมูลวงจรเลื่อนเฟสแสดงดังตารางที่ 5-18 โดยสดมภ์ที่ 1 เป็นลำดับชุดข้อมูล จำนวน 83 ชุด สดมภ์ที่ 2 ถึง สดมภ์ที่ 6 แสดงชุดข้อมูลวงจรเลื่อนเฟสทั้ง 105 บิต ถูกจัดแบ่งออกเป็นสดมภ์ละ 20 บิต และสดมภ์สุดท้ายถูกจัดแบ่งออกเป็น 5 บิต

ตารางที่ 5-18 ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 83 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 105 บิต

ลำดับ	บิตที่ 1-20	บิตที่ 21-40	บิตที่ 41-60	บิตที่ 61-80	บิตที่ 81-100	บิตที่ 101-105
1	0000000010 0000000000	0000100000 0001000000	1000000000 0000000000	0000010000 0000000000	0000000000 0000000100	00000
2	0000000001 0000000000	0000010000 0000000000	0100000000 1000000000	0000000000 0000000000	0010000000 0000000000	00000
3	0000000000 1000000000	0000001000 0000000000	0010000000 0000000000	0000000100 0000100000	0000000000 0000000001	00000
4	0010000000 0100000000	0000000100 0000000000	0001000000 0000000000	0000000000 0000010000	0000100000 0100000000	00000
5	0000000000 0010000001	0000000010 0000000000	0000100000 0000000000	1000000000 0000000000	0000000000 0010000000	01000
6	0000000000 0001000000	0000000001 0000001000	0000010000 0000000000	0100000000 0000000000	0000000000 0001000000	00000
7	0000000000 0000000000	0000000000 1000000000	0000001000 0001000000	0010000000 0000000000	0000000000 0000100000	00000
8	0010000001 0000000000	0000010000 0000000000	0100000000 0000000000	0000000000 0000010000	0000000000 0100000010	00000
9	0000000000 1000000001	0000001000 0000000000	0010000000 0000000000	0000000000 0000000000	0000000000 0010000000	00000
10	0000100000 0000000000	0000000100 0000001000	0001000000 0000000000	0000000000 0000010000	0000000000 0000000000	00000

ลำดับ	บิตที่ 1-20	บิตที่ 21-40	บิตที่ 41-60	บิตที่ 61-80	บิตที่ 81-100	บิตที่ 101-105
11	0001000000 0000000000	0100000000 0000000000	0000100000 0001000000	1000000000 0000000000	0000000000 0010000000	00000
12	0000000000 0000000000	1000000000 0000000000	0000000000 0000000000	0100000000 1000000000	0000000000 0001000000	00000
13	1000000100 0000000000	0000000000 0010000000	0000000000 0000000000	0000000000 0100000000	0000000001 0000000000	00000
14	0100000000 0000000000	0000100000 0001000000	0000000001 0000000000	0000000000 0000000000	0000000010 0000000000	00000
15	1000000000 0000000000	0000000000 0000100000	0100000000 1000000000	0000001000 0000000000	0000000000 0000000000	00000
16	0000000000 0000000100	0000000000 0001010000	0000000000 0100000010	0000000100 0000000000	0001000000 0000000000	00000
17	0000000000 0100000000	0000000000 0000100000	0000000000 1010000000	0000000010 0000010000	0000100000 0000000000	10000
18	0001000000 0000000000	0000000010 0000000000	0000000000 0100000000	0000000101 0000000000	0000010000 0010000000	01000
19	0000100000 0000000000	1000000000 0000000000	0000010000 0000000000	0000000010 0000000000	0000101000 0000000000	00100
20	0000000000 0010000000	0100000000 0000000000	0000000000 0000000000	0010000000 0000000000	0000010000 0000000000	01010
21	0000000000 0000010000	0000000001 0000000010	0000000000 0000100000	0000000000 0000000001	0000000000 0000000000	00100
22	0000000100 0000100000	0000000000 0010000000	0000001000 0000010000	0000000000 0100000000	0000000000 0000001000	00000
23	0000000010 0000000000	0000100000 0100000000	0000000001 0000000000	0001000000 0010000000	0000000010 0000000000	00000
24	0000000000 0000001000	0000010000 0000000000	0100000010 0000000000	0000001000 0000000000	1000000001 0000000000	00000
25	0100000010 0000000000	0000000000 0001000000	0010000000 0000000000	0000010000 0000000000	0001000000 0000000100	00000
26	0000000001 0100000010	0000010000 0000000000	0000000000 1000000001	0000000000 0000010000	0010000000 0000000000	10000

ลำดับ	บิตที่ 1-20	บิตที่ 21-40	บิตที่ 41-60	บิตที่ 61-80	บิตที่ 81-100	บิตที่ 101-105
27	0001000000 1010000000	0000001010 0000010000	0010000000 0000000000	0000000100 0000001000	0000000000 0010000001	00000
28	0000000000 0100000000	1000000101 0000000000	0001010000 0010000000	0000000000 0000000000	0000100000 0001000000	00000
29	0000010000 0010000000	0000000010 0000000100	0000101000 0000000000	1010000001 0000000000	0000000000 0000000000	01000
30	0000100000 0001000000	0010000001 0000000000	0000010000 0000100000	0101000000 0000000000	0000001000 0001000000	00000
31	0000000100 0000100000	0100000000 1000000000	0000001000 0000000000	0010000000 0100000000	1000000000 0000100000	00010
32	0000000000 0000010000	0000100000 0100000010	0000000100 0000001000	0001000000 0000000000	0000000010 0000010100	00000
33	1000000001 0000000000	0000000000 0010000000	0100000010 0000010000	0000100000 0001000000	1000000000 0000000000	00000
34	0100000000 0000000000	0000001000 0001000000	0000000001 0000000010	0000010000 0010000000	0100000000 1000000000	00000
35	1010000000 0000000000	0000000000 0000100000	0001000000 1000000000	0000001000 0000010000	0010000001 0000000000	00000
36	0000100000 0000000010	1000000000 0000100000	0000000000 0010000000	0000000010 0000000100	0000100000 0000000000	10000
37	0000010000 0000000000	0100000000 0000010000	0000000000 0101000000	0000000001 0000000000	0000010000 0000100000	01000
38	0000001000 0000000000	0010000000 0000000000	0000000000 0010100000	0000000010 1000000000	0000001000 0000000000	00100
39	0000000000 0000000000	0001000000 0000000000	0000000000 0000010000	0000000001 0100000000	0000010100 0000000000	00010
40	0000000000 0001000000	0000000000 0000000000	1000000000 0000000000	0000000000 0010000000	0000001010 0000000000	00101
41	1000010000 0000000000	0000000000 0000000000	0000001000 0000000000	0010000000 0000000000	0000000000 0000100000	00000
42	0000000000 0000000100	0010000000 0001000000	0000000000 0000000000	0001000000 0000000000	0000000000 0000010000	00000

ลำดับ	บิตที่ 1-20	บิตที่ 21-40	บิตที่ 41-60	บิตที่ 61-80	บิตที่ 81-100	บิตที่ 101-105
43	0100000000 0010000000	0000001000 0000000000	0000000000 0000000000	0000000000 0000000000	0000010000 1000000000	01000
44	0000010000 0000000000	0000000000 0000000000	0000000010 0001000000	0000000001 0000000000	1000000000 0000100000	00000
45	0000000000 1000000000	0000001000 0100000000	1010000000 0000000000	0000000000 0000000000	0000000000 0000000001	00001
46	0000000000 0100001000	0000000100 0000000000	0001000010 0000000100	0000000000 0001000000	0000000000 0000000000	00000
47	0100000000 0000000000	0000000010 0001000000	0000100000 0000000000	1000010000 0000100000	0000000000 1000000000	00000
48	0000100000 0000000010	0000000000 0000000000	0000010000 1000000000	0100000000 0000000000	0010000000 0101000000	00000
49	0000000000 0010000000	0000000010 0000000100	0000100000 0000010000	0000000001 0000000000	0000000000 0000001000	01000
50	0000000010 0001000000	0000000001 0000000000	0000010000 0000100000	0100000000 0010000000	0000001000 0000000000	00000
51	1000000000 0000100000	0000010000 1000000000	0000001000 0000000000	0010000000 0100000000	0000000001 0000000000	00010
52	0100001000 0000010000	0000000000 0100000000	0010000100 0000000000	0001000000 0000000000	0000000010 0000010000	00000
53	1000000000 0000000000	0001000000 0010100000	0000000010 0000000001	0000100000 0000000000	1000000000 0000000000	00000
54	0000000000 0000000100	0000000000 0001010000	1000000001 0100000000	0000010000 0000001000	0100000000 0000000000	00000
55	0000100000 0000000000	0000000000 0000100000	0000000000 1010000100	0000001010 0000000000	0010000000 0001000000	00000
56	0000010000 0000000000	0100000000 0000000000	0000000000 0100000000	0000000101 0000100000	0001010000 0000000000	00000
57	0010000000 0001000000	0010000000 0000000000	0000000000 0000000000	0000000010 0000000000	0000101000 0100000000	10100
58	0000000000 0000010000	1000000000 0000000000	0000000100 0000001000	0000000000 0010000000	0000000000 0000000000	00100

ลำดับ	บิตที่ 1-20	บิตที่ 21-40	บิตที่ 41-60	บิตที่ 61-80	บิตที่ 81-100	บิตที่ 101-105
59	1000000000 0000100000	0000000000 0010000100	0000000000 0001000000	0000100000 0001000000	0000000001 0000000000	00000
60	0000000000 0000000100	0000000000 0100000000	0000000001 0000100000	0000000000 1000000000	0100000000 1000000000	00000
61	0010000001 0000000000	0000000000 0000100000	0000000010 0000000000	0000001000 0100000000	0000000100 0000000010	00000
62	0000000000 1000010001	0000001000 0000000000	0000000000 0100000000	0000010000 0000000000	0001000010 0000000000	00001
63	0000010000 0000100000	0000000010 1000000000	0000101000 0100000000	0010000000 0000000000	0001000100 0000100000	00000
64	0000001000 0001000000	0000000001 0001000000	0000010001 0000001000	0000010000 0000000001	0100000000 0000010000	00100
65	0000000000 0000100000	0001000000 1000000000	0000001000 1000000000	0010001000 0001000000	0010000000 0000001000	00000
66	0100000010 0000000000	0000000000 0100000000	1000000100 0000000000	0001000100 0000000000	0001000000 1000000000	00000
67	0010000100 0000000000	0000010000 0000000000	0000000010 0000000100	0000100000 0000000000	1000100000 0000000000	10000
68	0000000000 0000000001	0000100000 0000010000	0010000000 0000000000	0000010000 0000100000	0100000000 0000000000	01000
69	0001000000 0000000000	0000000000 1000000000	0000000000 0001000010	0000000001 0000001000	0000000000 0000000001	00000
70	0000100000 0100000000	1000000000 0000000000	0000000100 0000000000	0000000000 1000010000	0000001000 0001000000	00000
71	0000000000 1000010000	0000001000 0001000010	0000000000 0000101000	0000010000 0010000001	0000000010 0000000000	00001
72	0000010000 0010000000	0001000000 0000000100	0100001000 0000000000	0010100001 0001000000	1000000100 0000000000	01000
73	0100000000 0001010000	0010000001 0000000000	1000000000 0000100000	0001000000 0000000000	0100001000 1000000000	00001
74	0010000001 0100000000	0000010100 0000100001	0100000000 0010010000	0000001000 0001010000	0000000001 0100000010	00000

ลำดับ	บิตที่ 1-20	บิตที่ 21-40	บิตที่ 41-60	บิตที่ 61-80	บิตที่ 81-100	บิตที่ 101-105
75	0000000010 0000010000	0000000000 0100001000	1000000101 0000000000	0001010010 0010000000	0000000000 0000010000	00001
76	0100000000 1001000000	0000000000 0001000000	0010000001 0000000000	0000010000 1000100000	0101000000 0000000000	00100
77	0000100000 0000001000	1000000001 0010000000	0000000010 0010000000	0100100010 0000000000	0000100000 0001000000	10000
78	0000000010 0000000000	0000100000 0000000000	1001000000 0000000000	0000000000 0010000000	0000100010 0000000100	10001
79	0000000001 0000000000	0000010000 0000000000	0100000000 0000000000	1000000000 0001000000	0000000001 0000000000	01000
80	0000010000 0000000000	0000000000 1000000000	0001000000 0000000000	0000000000 0000010000	0000000000 0100100000	00000
81	0000001000 0000000000	0010000000 0000000000	0000000100 0000000000	1000000000 0000000000	0000000000 0010000000	00000
82	0100000000 0000000000	0000000000 0001000000	0000000001 0000000010	0000010000 0100000000	0000000100 1000000000	00000
83	0000000000 0100000000	0000000000 0000000000	0000000000 0000010000	0000000010 0100000010	0000000100 0000000000	10010

5.2.5.4 ตำแหน่งการ Seed สำหรับวงจร S38417

ตำแหน่งการ Seed ของวงจร S38417 จะถูกกำหนดโดยชุดข้อมูลของวงจรเลื่อนเฟส จากตารางที่ 5-18 ข้อมูลวงจรเลื่อนเฟสจำนวน 83 ชุด มีจำนวนการวางตำแหน่งการ Seed เท่ากับ 83 ตำแหน่ง จึงทำให้ 1 ชุดของวงจรเลื่อนเฟส มีตำแหน่งของการ Seed จำนวน 1 ตำแหน่ง ซึ่งตำแหน่งการ Seed แต่ละตำแหน่งจะควบคุมข้อมูล Seed แต่ละชุด ทั้งหมด 83 ชุด ตำแหน่งของการ Seed เพื่อทดสอบวงจร S38417 คือ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 26, 28, 29, 30, 31, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 64, 65, 67, 68, 69, 70, 71, 73, 74, 76, 78, 82, 83, 85, 86, 87, 88, 89, 90, 91, 93, 95, 98, 102 และ 103

จากตำแหน่งการ Seed เพื่อทดสอบวงจร S38417 ซึ่งแสดงดังตารางที่ 5-19 สดมภ์ที่ 1, 3 และ 5 แสดงลำดับวงจรเลื่อนเฟส และสดมภ์ที่ 2, 4 และ 6 แสดงตำแหน่งการ Seed โดย

ข้อมูลลำดับที่ 1 ของวงจรเลื่อนเฟสใช้ตำแหน่งการ Seed ตำแหน่งที่ 65 ซึ่งข้อมูล Seed ในลำดับที่ 1 จะถูกเลื่อนเข้าสู่วงจร LFSR ในระหว่างตำแหน่ง Flip-Flop ที่ 65 และ 66 ของวงจร LFSR

ตารางที่ 5-19 ตำแหน่งการ Seed ของวงจร S38417

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
1	65	29	5	57	38
2	9	30	22	58	36
3	10	31	39	59	53
4	2	32	24	60	70
5	12	33	55	61	34
6	13	34	26	62	83
7	30	35	82	63	87
8	98	36	52	64	56
9	19	37	51	65	73
10	4	38	6	66	90
11	3	39	23	67	25
12	20	40	40	68	74
13	0	41	78	69	69
14	1	42	95	70	86
15	18	43	85	71	88
16	17	44	48	72	41
17	11	45	31	73	58
18	28	46	57	74	89
19	45	47	44	75	68
20	21	48	61	76	102
21	15	49	37	77	29
22	7	50	54	78	43
23	8	51	14	79	60

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
24	16	52	47	80	91
25	42	53	64	81	76
26	59	54	49	82	71
27	35	55	93	83	103
28	27	56	67		

5.2.5.5 สรุปผลการทดสอบของวงจร S38417

ผลการกำเนิดข้อมูลทดสอบของวงจร S38417 ได้ผลลัพธ์แสดงดังตารางที่ 5-20 โดย สดมภ์ที่ 1 แสดงชื่อของวงจรทดสอบ คือ S38417 สดมภ์ที่ 2 แสดงผลจำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ ซึ่งจำนวนชุดข้อมูลทดสอบที่ใช้คือ 83 บิตต่อ 1 ชุดข้อมูลทดสอบ ข้อมูลสดมภ์ที่ 3 แสดงจำนวนข้อมูล Seed ทั้งหมดในการทดสอบซึ่งเท่ากับ 28,220 บิต ข้อมูลสดมภ์ที่ 4 แสดงเปอร์เซ็นต์การลดขนาดข้อมูลการทดสอบ โดยจำนวนบิต Seed ที่ใช้เปรียบเทียบกับจำนวนเซลล์ตรวจกวาดทั้งหมด ผลลัพธ์ที่ได้เท่ากับ 95.01% ส่วนข้อมูลสดมภ์ที่ 5 แสดงจำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบเท่ากับ 21 สัญญาณนาฬิกา ซึ่งผลที่ได้คำนวณได้จากสมการ (5-2)

ตารางที่ 5-20 ผลการทดสอบวงจร S38417

ชื่อวงจรทดสอบ	จำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ	จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ	จำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบ
S38417	83	28,220	95.01%	21

5.2.6 ผลการกำเนิดข้อมูลการทดสอบวงจร S38584

5.2.6.1 ข้อมูลเบื้องต้นของวงจร S38584

วงจร S38584 เป็นวงจรขนาดกลาง เนื่องจาก 1 ชุดข้อมูลทดสอบมีจำนวนเซลล์ตรวจกวาดจำนวนมาก โดยจำนวน 1 ชุดข้อมูลทดสอบ มีจำนวนเซลล์ตรวจกวาดเท่ากับ 1,464 บิต จำนวนชุดข้อมูลทดสอบทั้งหมด 62 ชุด จำนวนเซลล์ตรวจกวาดทั้งหมดจึงเท่ากับ 90,768 บิต จาก

จำนวนเซลล์ตรวจกวาดทั้งหมด มีจำนวนบิตการเจาะจงค่าเท่ากับ 2,848 บิต มีจำนวนการเจาะจงค่าสูงที่สุด หรือ S_{max} เท่ากับ 56 บิต ข้อมูลพื้นฐานของวงจร S38584 แสดงดังตารางที่ 5-21

ตารางที่ 5-21 ข้อมูลพื้นฐานของวงจร S38584

ชื่อวงจรทดสอบ	จำนวนเซลล์ตรวจกวาดต่อ 1 ชุดข้อมูลทดสอบ	จำนวนชุดข้อมูลทดสอบ	จำนวนเซลล์ตรวจกวาดทั้งหมด	จำนวนการเจาะจงค่าทั้งหมด	S_{max}
S38584	1,464	62	90,768	2,848	56

5.2.6.2 วงจร LFSR สำหรับทดสอบวงจร S38584

วงจร LFSR ที่ใช้ในการทดสอบวงจร S38584 อ้างอิงจากเอกสาร [1][44] ซึ่งจำนวน Flip-Flop ที่ใช้เท่ากับ 75 บิต มีสมการโพลิเนียลเป็น $f(x) = X^{75}+X^{74}+X^{65}+X^{64}+1$ ซึ่งเป็นสมการที่สามารถกำเนิดข้อมูลทดสอบได้มีค่าสูง ที่มีจำนวน n เท่ากับ 75 โดยวงจร LFSR ดังกล่าวมีตำแหน่งการป้อนกลับของเอาท์พุทแบบอนุกรมที่บิตที่ 75, 74, 65 และ 64 โดยค่าเริ่มต้นของ LFSR ใช้การตั้งค่าเริ่มต้นให้บิตที่ 0 มีค่าเท่ากับ “1” ส่วนที่บิตอื่นๆ มีค่าเท่ากับ “0”

5.2.6.3 วงจรเลื่อนเฟสสำหรับทดสอบวงจร S38584

ชุดข้อมูลที่นำมาสร้างเป็นวงจรเลื่อนเฟส ใช้เอาท์พุทจากวงจร LFSR ในรูปแบบขนาน โดยการกำเนิดข้อมูลการทดสอบด้วยวิธีการ LFSR จำนวน 10,000 ชุด แล้วนำข้อมูลที่กำเนิดได้ไปคำนวณหาชุดข้อมูลวงจรเลื่อนเฟสที่เหมาะสม ค่าของตัวแปรที่ใช้ในการคำนวณ คือตัวแปร B มีค่าเท่ากับ 54 ชุด, L มีค่าเท่ากับ 23 ชุด และ I มีค่าเท่ากับ 15 บิต ผลลัพธ์ของชุดข้อมูลวงจรเลื่อนเฟสแสดงดังตารางที่ 5-22 โดยสดมภ์ที่ 1 เป็นลำดับชุดข้อมูล ทั้งหมด 54 ชุด สดมภ์ที่ 2 ถึง สดมภ์ที่ 8 ข้อมูลวงจรเลื่อนเฟสทั้ง 81 บิตที่ถูกจัดแบ่งออกเป็นสดมภ์ละ 10 บิต และสดมภ์สุดท้าย 15 บิต

ตารางที่ 5-22 ชุดข้อมูลวงจรเลื่อนเฟสจำนวน 54 ชุด ที่กำเนิดจากวงจร LFSR ขนาด 75 บิต

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-50	บิตที่ 51-60	บิตที่ 61-75
1	1000000000	1010001000	0000000000	0000000000	0000000000	0000000000	000000000000001
2	0000001100	0000000101	0001000000	0000000000	0000000000	0000000000	000000000000000
3	0000000000	0001100000	0000101000	1000000000	0000000000	0000000000	000000000000000

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-50	บิตที่ 51-60	บิตที่ 61-75
4	0000000000	0000000000	1100000000	0101000100	0000000000	0000000000	000000000000000
6	0000000000	0000000000	0000000000	0000110000	0000010100	0100000000	000000000000000
7	0000000000	0000000000	0000000000	0000000000	0110000000	0010100010	000000000000000
8	1100000000	0000000000	0000000000	0000000000	0000000011	0000000001	010001000000000
9	0111100110	0000000000	0000000000	0000000000	0000000000	0000011000	000000101000100
10	1100110011	1100110000	0000000000	0000000000	0000000000	0000000000	001100000000010
11	0000110110	0110011110	0110000000	0000000000	0000000000	0000000000	000000000110000
12	0101000000	0110110011	0011110011	0000000000	0000000000	0000000000	000000000000000
13	0000000010	1000000011	0110011001	1110011000	0000000000	0000000000	000000000000000
14	0000000000	0000010100	0000011011	0011001111	0011000000	0000000000	000000000000000
15	0000000000	0000000000	0010100000	0011011001	1001111001	1000000000	000000000000000
16	0000000000	0000000000	0000000001	0100000001	1011001100	1111001100	000000000000000
17	1000000000	0000000000	0000000000	0000001010	0000001101	1001100111	100110000000000
18	0001010100	0000000000	0000000000	0000000000	0001010000	0001101100	110011110011000
19	0000001000	1010100000	0000000000	0000000000	0000000000	1010000000	110110011001111
20	0111110000	0001000101	0100000000	0000000000	0000000000	0000000101	000000011011001
21	1000101011	1110000000	1000101010	0000000000	0000000000	0000000000	000010100000001
22	0000000100	0101011111	0000000100	0101010000	0000000000	0000000000	000000000001010
23	0001111000	0000100010	1011111000	0000100010	1010000000	0000000000	000000000000000
24	0000000000	1111000000	0100010101	1111000000	0100010101	0000000000	000000000000000
25	0000000000	0000000111	1000000010	0010101111	1000000010	0010101000	000000000000000
26	0000000000	0000000000	0000111100	0000010001	0101111100	0000010001	010100000000000
27	0100000100	0000000000	0000001100	1100000101	0100000100	0000000000	010001000001110
28	0000101010	0000100000	0000000000	0001100110	0000101010	0000100000	000000001000100
29	1100110000	0101010000	0100000000	0000000000	1100110000	0101010000	010000000000000
30	0001100110	0110000010	1010000010	0000000000	0000000110	0110000010	101000001000000
31	0000110010	1000010000	0000100010	1011100000	0000001000	0000001000	000001000010111
32	0011100000	0110010100	0010000000	0100010101	1100000000	0001000000	000100000000100
33	0000101001	1100000011	0010100001	0000000010	0010101110	0000000000	100000000010000
34	0000000000	0101001110	0000011001	0100001000	0000010001	0101110000	000000010000000
35	1000000000	0000000010	1001110000	0011001010	0001000000	0010001010	111000000000001
36	1101100000	0100000000	0000000100	0000001101	1011100000	1000001000	110000000000011
37	0000101000	0001001000	0010000000	0000101001	0110000100	0100000000	111010100101000

ลำดับ	บิตที่ 1-10	บิตที่ 11-20	บิตที่ 21-30	บิตที่ 31-40	บิตที่ 41-50	บิตที่ 51-60	บิตที่ 61-75
38	0000000100	1110100000	0000001000	0001100000	1010100000	0000001110	000000011100011
39	0110010000	0000100111	0100000000	0001000000	1100000101	0100000000	000111000000001
40	0100000010	0000000001	0000000010	0001100010	1000010010	0100001010	010000000110111
41	0000011100	0000010000	0010000000	0001000000	0010000110	0010100001	001001000010100
42	0101000000	0011100000	0010000001	0000000000	1000000001	0000110001	010000100100100
43	0001010010	1000000001	1100000001	0000001000	0000000100	0000001000	011000101000010
44	0100000001	0100101000	0110100100	0101001010	0000000111	0000000100	000010000000000
45	0001000001	0000110000	1001000000	0100100100	1001000001	0000100001	100110010100000
46	1000010100	0000000101	1000000000	0000000001	0010010000	0001000001	100010001011011
47	1011111100	0010100000	0000101100	0000000000	0000001001	0010000000	100000110001000
48	0010100000	0100000100	1000000100	1000010001	0100010001	0000011010	000000000010011
49	0001000000	0000101000	0000011000	0110000010	0000000101	0100101101	000010010010001
50	0000000110	0010000010	0001000100	1000000000	1000110000	0000010000	101110011100100
51	0001010000	0000110001	0000010000	1000100100	0000000100	0110000000	001000010111001
52	1100001110	0011000101	0000000011	0001000001	0000100010	0100000000	010001100000000
53	0100100000	0100000100	0100001000	0010000010	1000110000	1110010011	010000000000010
54	0001010011	1000000000	0000000100	0010000000	1001111011	0000000001	111000000000001

5.2.6.4 ตำแหน่งการ Seed สำหรับวงจร S38584

ตำแหน่งการ Seed ของวงจร S38584 ขึ้นอยู่กับการวางตำแหน่งข้อมูลของวงจรเลื่อนเฟส จากตารางที่ 5-22 ข้อมูลวงจรเลื่อนเฟสจำนวน 54 ชุด มีจำนวนการวางตำแหน่งการ Seed เท่ากับ 54 ชุด จึงทำให้ 1 ชุดของวงจรเลื่อนเฟส มีตำแหน่งของการ Seed จำนวน 1 ตำแหน่ง ซึ่งตำแหน่งการ Seed แต่ละตำแหน่งจะควบคุมข้อมูล Seed แต่ละชุด ทั้งหมด 54 ชุด ตำแหน่งของการ Seed เพื่อทดสอบวงจร S38584 คือ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 39, 40, 41, 42, 43, 47, 45, 44, 46, 50, 54, 57, 62, 63, 64, 65, 66, 69, 70 และ 71

จากตำแหน่งของ tap seed เพื่อทดสอบวงจร S38584 ซึ่งแสดงดังตารางที่ 5-23 สดมภ์ที่ 1 และ 3 แสดงลำดับวงจรเลื่อนเฟส และสดมภ์ที่ 2 และ 4 ตำแหน่งการ Seed โดยข้อมูลลำดับที่ 1 ของวงจรเลื่อนเฟสใช้ตำแหน่งการ Seed ตำแหน่งที่ 3 ซึ่งข้อมูล Seed ในลำดับที่ 1 จะถูกเลื่อนเข้าสู่วงจร LFSR ในระหว่างตำแหน่ง Flip-Flop ที่ 3 และ 4 ของวงจร LFSR

ตารางที่ 5-23 ตำแหน่งการ Seed ของวงจร S38584

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
1	3	28	14
2	6	29	40
3	13	30	47
4	20	31	65
5	27	32	63
6	34	33	70
7	41	34	45
8	0	35	32
9	1	36	37
10	5	37	16
11	7	38	42
12	11	39	64
13	8	40	33
14	15	41	54
15	22	42	69
16	29	43	66
17	36	44	9
18	43	45	23
19	10	46	39
20	17	47	2
21	4	48	30
22	31	49	57
23	18	50	62
24	21	51	71
25	28	52	44

ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed	ลำดับวงจรเลื่อนเฟส	ตำแหน่งการ Seed
26	24	53	50
27	26	54	46

5.2.6.5 สรุปผลการทดสอบของวงจร S38584

ผลการกำเนิดข้อมูลทดสอบของวงจร S38584 ได้ผลลัพธ์แสดงดังตารางที่ 5-24 โดย สดมภ์ที่ 1 แสดงชื่อของวงจรทดสอบ คือ S38584 สดมภ์ที่ 2 แสดงผลจำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ ซึ่งจำนวนบิต Seed ที่ใช้คือ 54 บิตต่อ 1 ชุดข้อมูลทดสอบ สดมภ์ที่ 3 แสดงจำนวนข้อมูล Seed ดังนั้นจำนวนบิต Seed ทั้งหมดของการทดสอบจึงเท่ากับ 3,348 บิต สดมภ์ที่ 4 แสดงเปอร์เซ็นต์การลดขนาดข้อมูลการทดสอบ โดยการจำนวนบิต Seed ที่ใช้เปรียบเทียบกับจำนวนเซลล์ตรวจกวาดทั้งหมด ผลลัพธ์ที่ได้เท่ากับ 96.31% ส่วนสดมภ์ที่ 5 แสดงจำนวนสัญญาณนาฬิกาต่อ 1 ชุดข้อมูลทดสอบเท่ากับ 28 สัญญาณนาฬิกา

ตารางที่ 5-24 ผลการทดสอบวงจร S38584

ชื่อวงจรทดสอบ	จำนวนบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ	จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ	จำนวนสัญญาณนาฬิกา ต่อ 1 ชุดข้อมูลทดสอบ
S38584	54	3,348	96.31%	28

5.3 ผลการกำเนิดข้อมูลการทดสอบด้วยสถาปัตยกรรม Parallel LFSR Reseeding ร่วมกับ Selection Register

การกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register นั้นใช้จำนวน Flip-Flop, จำนวนชุดและข้อมูลวงจรเลื่อนเฟส, จำนวนชุดข้อมูล Seed และการวางตำแหน่งบิต Seed เหมือนกับ การกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding แบบธรรมดา จึงไม่มีการจะไม่กล่าวถึงในหัวข้อนี้ แต่ในหัวข้อนี้ จะกล่าวถึงผลการกำเนิดข้อมูลการทดสอบด้วยการปรับปรุงกลไกการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR เพื่อลดจำนวนข้อมูล Seed ที่จะใช้กำเนิดข้อมูลทดสอบ โดยการประยุกต์ใช้ Selection Register

5.3.1 ผลการกำเนิดข้อมูลการทดสอบวงจร S5378

5.3.1.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ

การปรับปรุงกลไกการเลื่อนบิตข้อมูล Seed ด้วยการใช้ Selection Register ใช้การแบ่งกลุ่มข้อมูล Seed ออกเป็น 2 กลุ่มคือ *G1* และ *G2* และใช้ Selection Register ในการควบคุมการเลื่อนบิตข้อมูล Seed ของแต่ละกลุ่มเข้าสู่วงจร LFSR ผ่านทางวงจรนับ การเลื่อนบิต ข้อมูล Seed เพื่อกำเนิดข้อมูลทดสอบของแต่ละชุดข้อมูลทดสอบใช้ค่าเฉลี่ยของจำนวนการเจาะจงค่า หรือ S_{avg} เป็นเกณฑ์ เพื่อพิจารณาการเลื่อนข้อมูล Seed ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมากกว่าค่าเฉลี่ยของจำนวนการเจาะจงค่า จะทำการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แบบเต็ม หรือ Full คือจะเลื่อนข้อมูล Seed ทั้ง *G1* และ *G2* แต่ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมีค่าน้อยกว่าค่าเฉลี่ยของจำนวนการเจาะจงค่าจะทำการเลื่อนข้อมูล Seed แบบครึ่ง หรือ แบบ Haft คือจะเลื่อนข้อมูล Seed เฉพาะ *G1* เท่านั้น ซึ่งค่าเฉลี่ยของจำนวนการเจาะจงค่าของวงจร S5378 เท่ากับ 17 บิต จากนั้นจะพิจารณาที่ค่าบวกและลบจากค่าเฉลี่ยของจำนวนการเจาะจงค่า S_{avg}-1, S_{avg} และ S_{avg}+1 เพื่อดูแนวโน้วการเปลี่ยนแปลงของจำนวนข้อมูล Seed ที่ต้องใช้ ผลการทดลองแสดงดังตารางที่ 5-25

ตารางที่ 5-25 แสดงผลการทดลองกำเนิดข้อมูลการทดสอบวงจร S5378 ด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register โดยในสดมภ์ที่ 1 และ 2 แสดงตัวเลขที่ใช้ในการแบ่งกลุ่มชุดข้อมูล Seed ที่แสดงในรูปแบบของตัวแปร S_{avg} และตัวเลข สดมภ์ที่ 3 และ 4 แสดงจำนวนชุดข้อมูลทดสอบที่มีการเลื่อนบิตข้อมูล Seed ในแบบเต็มและแบบครึ่งตามลำดับ สดมภ์ที่ 5 และ 6 แสดงจำนวนบิตข้อมูล Seed ต่อกลุ่ม โดยแบ่งเป็น *G1* และ *G2* ตามลำดับ ส่วนสดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบ

ในการกำเนิดข้อมูลการทดสอบวงจร S5378 มีจำนวนชุดข้อมูลทดสอบทั้งหมด 30 ชุด ผลการแบ่งกลุ่ม Seed แสดงในตารางที่ 5-25 ตัวเลขที่ใช้แบ่งกลุ่มการเลื่อนบิตข้อมูล Seed และให้ผลลัพธ์ดีที่สุดคือ S_{avg}-1 หรือ 16 บิต จำนวนชุดข้อมูลที่ใช้การเลื่อนบิตข้อมูล Seed แบบเต็ม ที่จำนวน 27 ชุด และจำนวนการเลื่อนบิตข้อมูล Seed แบบครึ่งเท่ากับ 3 ชุด ซึ่งการเลื่อนบิตแบบเต็มใช้จำนวนข้อมูล Seed เท่ากับ 17 บิต และ การเลื่อนบิตข้อมูล Seed แบบครึ่ง ใช้จำนวนข้อมูล Seed เท่ากับ 11 บิต จำนวนข้อมูล Seed ทั้งหมด เท่ากับ 492 บิต

ตารางที่ 5-25 ผลการทดลองกำเนิดข้อมูลทดสอบด้วยวิธี
Parallel LFSR Reseeding ด้วย Selection Register สำหรับวงจร S5378

ตัวเลขที่ใช้แบ่งกลุ่มชุดข้อมูล Seed		จำนวนชุดข้อมูลทดสอบ		จำนวนบิตข้อมูล Seed ต่อกลุ่ม		จำนวน Seed ทั้งหมด
		เต็ม	ครึ่ง	*G1*	*G2*	
S_{avg}-1	**16**	**27**	**3**	**11**	**6**	**492**
S_{avg}	17	29	1	17	0	510
S_{avg}+1	18	30	0	17	0	510

จากผลการทดลองในตารางที่ 5-25 แสดงให้เห็นว่า ตัวเลขที่ใช้แบ่งกลุ่มชุดข้อมูล Seed ที่ใช้ค่า S_{avg} และ S_{avg}+1 ไม่สามารถลดขนาดข้อมูล Seed ด้วยการใช้ Selection Register มาแบ่งกลุ่มการเลื่อนข้อมูล Seed ได้ ถึงแม้จะมีการแบ่งกลุ่มการเลื่อนข้อมูล Seed แล้วก็ตาม ยังจำเป็นต้องใช้ข้อมูล Seed ในการกำเนิดข้อมูลทดสอบเท่ากับวิธีการ Parallel LFSR Reseeding ในแนวคิดที่ 1

5.3.1.2 สรุปผลการทดสอบของวงจร S5378

สรุปผลการทดลองในตารางที่ 5-26 สดมภ์ที่ 1 แสดงชื่อวงจรทดสอบ S5378 สดมภ์ที่ 2 และ 3 แสดงจำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบของ *G1* และ *G2* มีค่าเท่ากับ 11 และ 6 ตามลำดับ แสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบเท่ากับ 492 บิต สดมภ์สุดท้ายแสดงเปอร์เซ็นต์การลดขนาดข้อมูลทดสอบเท่ากับ 92.33 % โดยคำนวณจากสมการ (5-1) ส่วนจำนวนสัญญาณนาฬิกาที่ใช้เท่ากับวิธีการ Parallel LFSR Reseeding ในแนวคิดที่ 1

ตารางที่ 5-26 สรุปการกำเนิดข้อมูลทดสอบสำหรับวงจร S5378

ชื่อวงจรทดสอบ	จำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ		จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ
	G1	*G2*		
S5378	11	6	492	92.33%

5.3.2 ผลการกำเนิดข้อมูลการทดสอบวงจร S9234

5.3.2.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ

การปรับปรุงกลไกการเลื่อนบิตข้อมูล Seed ด้วยการใช้ Selection Register ใช้การแบ่งกลุ่มข้อมูล Seed ออกเป็น 2 กลุ่มคือ *G1* และ *G2* และใช้ Selection Register ในการควบคุมการเลื่อนบิตข้อมูล Seed ของแต่ละกลุ่มเข้าสู่วงจร LFSR ผ่านทางวงจรนับ การเลื่อนบิต ข้อมูล Seed เพื่อกำเนิดข้อมูลทดสอบของแต่ละชุดข้อมูลทดสอบใช้ค่าเฉลี่ยของจำนวนการเจาะจงค่า หรือ S_{avg} เป็นเกณฑ์ เพื่อพิจารณาการเลื่อนข้อมูล Seed ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมากกว่าจำนวนค่าเฉลี่ยของจำนวนการเจาะจงค่า จะทำการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แบบเต็ม คือจะเลื่อนข้อมูล Seed ทั้ง *G1* และ *G2* แต่ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมีค่าน้อยกว่าค่าเฉลี่ยของจำนวนการเจาะจงค่าจะทำการเลื่อนข้อมูล Seed แบบครึ่ง คือจะเลื่อนข้อมูล Seed เฉพาะ *G1* เท่านั้น ซึ่งค่า S_{avg} ของวงจร S9234 เท่ากับ 33 บิต จากนั้นจะพิจารณาค่าบวกลบของค่า S_{avg} ที่ S_{avg}-6, S_{avg}-5, S_{avg}-4, S_{avg}-3, S_{avg}-2, S_{avg}-1, S_{avg}+1, S_{avg}+2, S_{avg}+3, S_{avg}+4, S_{avg}+5 และ S_{avg}+6 เพื่อดูแนวโน้วการเปลี่ยนแปลงของจำนวนข้อมูล Seed ผลการทดลองแสดงดังตารางที่ 5-27

ตารางที่ 5-27 แสดงผลการทดลองกำเนิดข้อมูลการทดสอบวงจร S9234 ด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register โดยในสดมภ์ที่ 1 และ 2 แสดงตัวเลขที่ใช้ในการแบ่งกลุ่มชุดข้อมูล Seed ที่แสดงในรูปแบบของตัวแปร S_{avg} และตัวเลข ตามลำดับ สดมภ์ที่ 3 และ 4 แสดงจำนวนชุดข้อมูลทดสอบที่มีการเลื่อนบิตข้อมูล Seed ในแบบเต็มและแบบครึ่ง ตามลำดับ สดมภ์ที่ 5 และ 6 แสดงจำนวนบิตข้อมูล Seed ต่อกลุ่ม โดยแบ่งเป็น G1 และ G2 ตามลำดับ ส่วนสดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบ

ในการกำเนิดข้อมูลการทดสอบวงจร S9234 มีจำนวนชุดข้อมูลทดสอบทั้งหมด 138 ชุด ผลการแบ่งกลุ่ม Seed แสดงในตารางที่ 5-27 ตัวเลขที่ใช้แบ่งกลุ่มการเลื่อนบิตข้อมูล Seed และให้ผลลัพธ์ดีที่สุดคือ S_{avg}+3 หรือ 36 บิต จำนวนชุดข้อมูลที่ใช้การเลื่อนบิตข้อมูล Seed แบบเต็มที่จำนวน 63 ชุด และจำนวนการเลื่อนบิตข้อมูล Seed แบบครึ่งเท่ากับ 75 ชุด ซึ่งการเลื่อนบิตแบบเต็ม ใช้จำนวนข้อมูล Seed เท่ากับ 59 บิต และ การเลื่อนบิตข้อมูล Seed แบบครึ่ง ใช้จำนวนข้อมูล Seed เท่ากับ 27 บิต จำนวนข้อมูล Seed ทั้งหมด เท่ากับ 5,742 บิต

ตารางที่ 5-27 ผลการกำเนิดข้อมูลทดสอบด้วยวิธี
Parallel LFSR Reseeding ด้วย Selection Register สำหรับวงจร S9234

ตัวเลขที่ใช้แบ่งกลุ่ม		จำนวนชุดข้อมูลทดสอบ		จำนวนบิตข้อมูล Seed ต่อกลุ่ม		จำนวน Seed ทั้งหมด
		แบบเต็ม	แบบครึ่ง	G1	G2	
S_{avg}-6	27	51	51	33	26	6,459
S_{avg}-5	28	58	58	33	26	6,228
S_{avg}-4	29	63	63	33	26	6,063
S_{avg}-3	30	66	66	33	26	5,964
S_{avg}-2	31	68	68	33	26	5,957
S_{avg}-1	32	70	70	33	26	5,832
S_{avg}	33	71	71	34	27	5,870
S_{avg}+1	34	73	73	34	27	5,806
S_{avg}+2	35	74	74	34	27	5,774
S_{avg}+3	**36**	**75**	**75**	**34**	**27**	**5,742**
S_{avg}+4	37	79	79	32	29	5,772
S_{avg}+5	38	85	85	29	32	5,847
S_{avg}+6	39	88	88	29	32	5,766

ผลการทดลองในตารางที่ 5-27 แสดงให้เห็นว่า ตัวเลขที่ใช้แบ่งกลุ่มชุดข้อมูล Seed S_{avg}+3 ใช้จำนวนข้อมูล Seed น้อยที่สุดที่จำนวน 5,742 บิต แต่ยังคงมีค่าสูงกว่าจำนวนการเจาะจงค่าทั้งหมดที่จำนวน 4,674 บิต เนื่องจากจำนวนการเจาะจงค่าในแต่ละชุดข้อมูลทดสอบของวงจร S9234 มีความแตกต่างกันมาก โดยจำนวนการเจาะจงค่าสูงที่สุดหรือ S_{max} มีค่าเท่ากับ 61 และการเจาะจงค่าที่ต่ำสุดอยู่ที่ 11 ดังนั้นการกำเนิดข้อมูลทดสอบชุดข้อมูลทดสอบที่มีจำนวนการเจาะจงค่าสูง จะต้องใช้ข้อมูล Seed สูงมาก แต่เมื่อทุกชุดข้อมูลทดสอบต้องทำการเลื่อนข้อมูล Seed ในจำนวนที่เท่ากัน จึงทำให้ในบางชุดข้อมูลทดสอบคงเหลือข้อมูล Seed ในรูปแบบตัวแปรอิสระจำนวนมาก ซึ่งวิธีการหนึ่งที่จะช่วยลดข้อมูล Seed ได้ ก็คือการเพิ่มการแบ่งชุดข้อมูล Seed ออกเป็น 4 กลุ่ม เพื่อควบคุมกลไกการเลื่อนบิตข้อมูล Seed ซึ่งอาจทำให้สามารถลดจำนวนข้อมูล Seed ลงได้ โดยการใช้ค่าความแตกต่างของการเจาะจงค่ามาเป็นกลไกการควบคุมการแบ่งกลุ่ม

5.3.2.2 สรุปผลการทดสอบของวงจร S9234

ตารางที่ 5-28 แสดงผลการทดลอง โดยสดมภ์ที่ 1 แสดงชื่อวงจรทดสอบ S9234 สดมภ์ที่ 2 และ 3 แสดงจำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบของ *G1* และ *G2* มีค่าเท่ากับ 32 และ 27 ตามลำดับ ซึ่งเมื่อรวมจำนวนชุดข้อมูลทดสอบของ *G1* และ *G2* แล้วจะเท่ากับ 59 ชุด สดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบเท่ากับ 5,742 บิต สดมภ์สุดท้ายแสดงเปอร์เซ็นการลดขนาดข้อมูลทดสอบเท่ากับ 83.15% โดยคำนวณจากสมการ (5-1) ส่วนจำนวนสัญญาณที่ใช้ในการทดสอบเท่ากับวิธีการ Parallel LFSR Reseeding ในแนวคิดที่ 1

ตารางที่ 5-28 ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S9234

ชื่อวงจรทดสอบ	จำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ		จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ
	G1	*G2*		
S9234	32	27	5,742	83.15%

5.3.3 ผลการกำเนิดข้อมูลการทดสอบวงจร S13207

5.3.3.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ

การปรับปรุงกลไกการเลื่อนบิตข้อมูล Seed ด้วยการใช้ Selection Register ใช้การแบ่งกลุ่มข้อมูล Seed ออกเป็น 2 กลุ่มคือ *G1* และ *G2* และใช้ Selection Register ในการควบคุมการเลื่อนบิตข้อมูล Seed ของแต่ละกลุ่มเข้าสู่วงจร LFSR ผ่านทางวงจรนับ การเลื่อนบิต ข้อมูล Seed เพื่อกำเนิดข้อมูลทดสอบของแต่ละชุดข้อมูลทดสอบใช้ค่าเฉลี่ยของจำนวนการเจาะจงค่า หรือ S_{avg} เป็นเกณฑ์ เพื่อพิจารณาการเลื่อนข้อมูล Seed ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมากกว่าจำนวนค่าเฉลี่ยของจำนวนการเจาะจงค่า จะทำการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แบบเต็ม คือจะเลื่อนข้อมูล Seed ทั้ง *G1* และ *G2* แต่ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมีค่าน้อยกว่าค่าเฉลี่ยของจำนวนการเจาะจงค่าจะทำการเลื่อนข้อมูล Seed แบบครึ่ง คือจะเลื่อนข้อมูล Seed เฉพาะ *G1* เท่านั้น ซึ่ง S_{avg} ของวงจร S13207 เท่ากับ 17 บิต การกำหนดกลุ่มการเลื่อนข้อมูล Seed จากนั้นจะพิจารณาที่ค่าบวกลบของ S_{avg} คือ S_{avg}-3, S_{avg}-2, S_{avg}-1, S_{avg}+1, S_{avg}+2, และ S_{avg}+3 เพื่อดูแนวโน้วการเปลี่ยนแปลงของจำนวนข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ เพื่อให้สามารถหาจำนวน Seed ที่น้อยที่สุดที่ใช้ในการกำเนิดข้อมูลทดสอบ ผลการทดลองแสดงดังตารางที่ 5-29

ตารางที่ 5-29 แสดงผลการกำเนิดข้อมูลการทดสอบวงจร S13207 ด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register สดมภ์ที่ 1 และ 2 แสดงตัวเลขที่ใช้ในการแบ่งกลุ่มชุดข้อมูลทดสอบ ที่แสดงในรูปแบบของตัวแปร S_{avg} ส่วนสดมภ์ที่ 2 แสดงในรูปแบบของตัวเลข สดมภ์ที่ 3 และ 4 แสดงจำนวนชุดข้อมูลทดสอบที่มีการเลื่อนบิตข้อมูล Seed ในแบบเต็มและแบบครึ่ง ตามลำดับ สดมภ์ที่ 5 และ 6 จำนวนบิตข้อมูล Seed ต่อกลุ่ม โดยแบ่งเป็น *G1* และ *G2* ตามลำดับ ส่วนสดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบ

ในการกำเนิดข้อมูลการทดสอบวงจร S13207 มีจำนวนชุดข้อมูลทดสอบทั้งหมด 157 ชุด ผลจากตารางที่ 5-29 ตัวเลขที่ใช้แบ่งกลุ่มชุดข้อมูลทดสอบ ที่ให้ผลลัพธ์ดีที่สุดคือ S_{avg} หรือ 17 บิต โดยจำนวนชุดข้อมูลที่ใช้การเลื่อนบิตข้อมูล Seed แบบเต็ม ที่จำนวน 70 ชุด และจำนวนการเลื่อนบิตข้อมูล Seed แบบครึ่งเท่ากับ 87 ชุด ซึ่งการเลื่อนบิต Seed แบบเต็ม ใช้จำนวนข้อมูล Seed เท่ากับ 24 บิต และ การเลื่อนบิตข้อมูล Seed แบบครึ่ง ใช้จำนวนข้อมูล Seed เท่ากับ 14 บิต จำนวนข้อมูล Seed ทั้งหมด เท่ากับ 2,898 บิต

ตารางที่ 5-29 ผลการกำเนิดข้อมูลทดสอบด้วยวิธี
Parallel LFSR Reseeding ด้วย Selection Register สำหรับวงจร S13207

ตัวเลขที่ใช้แบ่งกลุ่ม		จำนวนชุดข้อมูลทดสอบ		จำนวนบิตข้อมูล Seed ต่อกลุ่ม		จำนวน Seed ทั้งหมด
		แบบเต็ม	แบบครึ่ง	*G1*	*G2*	
S_{avg}-3	14	127	30	5	21	3,198
S_{avg} -2	15	93	64	12	12	3,000
S_{avg} -1	16	83	74	13	11	2,954
S_{avg}	**17**	**70**	**87**	**14**	**10**	**2,898**
S_{avg} +1	18	68	89	15	9	2,967
S_{avg} +2	19	95	62	15	9	3,210
S_{avg} +3	20	108	49	17	7	3,425

5.3.3.2 สรุปผลการทดสอบของวงจร S13207

ผลการทดลองในตารางที่ 5-30 สดมภ์ที่ 1 แสดงชื่อวงจรทดสอบ S13027 สดมภ์ที่ 2 และ 3 แสดงจำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบของ *G1* และ *G2* ตามลำดับ

สดมภ์ที่ 4 แสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบเท่ากับ 5,742 บิต สดมภ์สุดท้ายแสดงเปอร์เซ็นต์การลดขนาดข้อมูลทดสอบเท่ากับ 97.36% โดยคำนวณจากสมการ (5-1) ส่วนจำนวนสัญญาณที่ใช้ในการทดสอบเท่ากับวิธีการ Parallel LFSR Reseeding ในแนวคิดที่ 1

ตารางที่ 5-30 ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S13207

ชื่อวงจรทดสอบ	จำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ		จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ
	G1	*G2*		
S13207	14	10	2,898	97.36%

5.3.4 ผลการกำเนิดข้อมูลการทดสอบวงจร S15850

5.3.4.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ

การปรับปรุงกลไกการเลื่อนบิตข้อมูล Seed ด้วยการใช้ Selection Register ใช้การแบ่งกลุ่มข้อมูล Seed ออกเป็น 2 กลุ่มคือ *G1* และ *G2* และใช้ Selection Register ในการควบคุมการเลื่อนบิตข้อมูล Seed ของแต่ละกลุ่มเข้าสู่วงจร LFSR ผ่านทางวงจรนับ การเลื่อนบิต ข้อมูล Seed เพื่อกำเนิดข้อมูลทดสอบของแต่ละชุดข้อมูลทดสอบใช้ค่าเฉลี่ยของจำนวนการเจาะจงค่า หรือ S_{avg} เป็นเกณฑ์ เพื่อพิจารณาการเลื่อนข้อมูล Seed ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมากกว่าจำนวนค่าเฉลี่ยของจำนวนการเจาะจงค่า จะทำการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แบบเต็ม คือจะเลื่อนข้อมูล Seed ทั้ง *G1* และ *G2* แต่ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมีค่าน้อยกว่าค่าเฉลี่ยของจำนวนการเจาะจงค่าจะทำการเลื่อนข้อมูล Seed แบบครึ่ง คือจะเลื่อนข้อมูล Seed เฉพาะ *G1* เท่านั้น ซึ่ง S_{avg} ของวงจร S15850 เท่ากับ 33 บิต ในการกำเนิดข้อมูลทดสอบจะพิจารณาที่ S_{avg}-4, S_{avg}-3, S_{avg}-2, S_{avg}-2, S_{avg}-1, S_{avg}+1, S_{avg}+2 และ S_{avg}+3 เพื่อดูแนวโน้วการเปลี่ยนแปลงของจำนวนข้อมูล Seed และจำนวนชุดข้อมูล Seed ที่ต้องใช้ และพิจารณาหาจำนวน Seed ที่น้อยที่สุดที่ใช้ในการกำเนิดข้อมูลทดสอบ ผลการทดลองแสดงดังตารางที่ 5-31

จากตารางที่ 5-31 แสดงผลการทดกำเนิดข้อมูลการทดสอบวงจร S15850 ด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register โดยในสดมภ์ที่ 1 และ 2 แสดงตัวเลขที่ใช้ในการแบ่งกลุ่มชุดข้อมูล Seed ที่แสดงในรูปแบบของตัวแปร S_{avg} และตัวเลข สดมภ์ที่ 3 และ 4 แสดงจำนวนชุดข้อมูลทดสอบที่มีการเลื่อนบิตข้อมูล Seed ในแบบเต็มและแบบครึ่ง สดมภ์ที่

5 และ 6 จำนวนบิตข้อมูล Seed ต่อกลุ่ม โดยแบ่งเป็น *G1* และ *G2* ตามลำดับ ส่วนสดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบ

ในการกำเนิดข้อมูลการทดสอบวงจร S15850 มีจำนวนชุดข้อมูลทดสอบทั้งหมด 167 ชุด ผลจากตารางที่ 5-31 ตัวเลขที่ใช้แบ่งกลุ่มการเลื่อนบิตข้อมูล Seed และให้ผลลัพธ์ดีที่สุดคือ S_{avg}-3 หรือ 34 บิต จำนวนชุดข้อมูลที่ใช้การเลื่อนบิตข้อมูล Seed แบบเต็ม ที่จำนวน 96 ชุด และจำนวนการเลื่อนบิตข้อมูล Seed แบบครึ่งเท่ากับ 71 ชุด ซึ่งการเลื่อนบิตแบบเต็ม มีการเลื่อนบิตข้อมูล Seed จำนวน 34 บิต และ การเลื่อนบิตข้อมูล Seed แบบครึ่ง มีการเลื่อนบิตข้อมูล Seed จำนวน 27 บิต จำนวนข้อมูล Seed ทั้งหมด เท่ากับ 5,181 บิต

ตารางที่ 5-31 ผลการกำเนิดข้อมูลทดสอบด้วยวิธี
Parallel LFSR Reseeding ด้วย Selection Register สำหรับวงจร S15850

ตัวเลขที่ใช้แบ่งกลุ่ม		จำนวนชุดข้อมูลทดสอบ		จำนวนบิตข้อมูล Seed ต่อกลุ่ม		จำนวน Seed ทั้งหมด
		แบบเต็ม	แบบครึ่ง	*G1*	*G2*	
S_{avg} -4	29	112	55	26	8	5,238
S_{avg} -3	**30**	**96**	**71**	**27**	**7**	**5,181**
S_{avg}-2	31	73	94	29	5	5,208
S_{avg}-1	32	65	102	30	4	5,270
S_{avg}	33	64	103	30	4	5,266
S_{avg}+1	34	61	106	30	4	5,254
S_{avg}+2	35	39	128	31	3	5,294
S_{avg}+3	36	28	139	32	2	5,400

5.3.4.2 สรุปผลการทดสอบของวงจร S15850

ผลสรุปการกำเนิดข้อมูลการทดสอบสำหรับวงจร S15850 แสดงดังตารางที่ 5-32 สดมภ์ที่ 1 แสดงชื่อวงจรทดสอบ S15850 สดมภ์ที่ 2 และ 3 แสดงจำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบของ *G1* และ *G2* มีค่าเท่ากับ 27 กับ 7 ตามลำดับ สดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบเท่ากับ 5,181 บิต สดมภ์สุดท้ายแสดง

เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบเท่ากับ 94.92% โดยคำนวณจากสมการ (5-1) ส่วนจำนวนสัญญาณที่ใช้ในการทดสอบเท่ากับวิธีการ Parallel LFSR Reseeding ในแนวคิดที่ 1

ตารางที่ 5-32 ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S15850

ชื่อวงจรทดสอบ	จำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ		จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ
	G1	*G2*		
S15850	27	7	5,181	94.92%

5.3.5 ผลการกำเนิดข้อมูลการทดสอบวงจร S38417

5.3.5.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ

การปรับปรุงกลไกการเลื่อนบิตข้อมูล Seed ด้วยการใช้ Selection Register ใช้การแบ่งกลุ่มข้อมูล Seed ออกเป็น 2 กลุ่มคือ *G1* และ *G2* และใช้ Selection Register ในการควบคุมการเลื่อนบิตข้อมูล Seed ของแต่ละกลุ่มเข้าสู่วงจร LFSR ผ่านทางวงจรนับ การเลื่อนบิต ข้อมูล Seed เพื่อกำเนิดข้อมูลทดสอบของแต่ละชุดข้อมูลทดสอบใช้ค่าเฉลี่ยของจำนวนการเจาะจงค่า หรือ S_{avg} เป็นเกณฑ์ เพื่อพิจารณาการเลื่อนข้อมูล Seed ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมากกว่าจำนวนค่าเฉลี่ยของจำนวนการเจาะจงค่า จะทำการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แบบเต็ม คือจะเลื่อนข้อมูล Seed ทั้ง *G1* และ *G2* แต่ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมีค่าน้อยกว่าค่าเฉลี่ยของจำนวนการเจาะจงค่าจะทำการเลื่อนข้อมูล Seed แบบครึ่ง คือจะเลื่อนข้อมูล Seed เฉพาะ *G1* เท่านั้น ซึ่ง S_{avg} ของวงจร S38417 เท่ากับ 70 บิต และพิจารณาค่าการแบ่งกลุ่มชุดข้อมูลทดสอบที่ S_{avg}-4, S_{avg}-3, S_{avg}-2, S_{avg}-1, S_{avg}+1, S_{avg}+2, S_{avg}+3, S_{avg}+4 และ S_{avg}+5 เพื่อดูแนวโน้วการเปลี่ยนแปลงของจำนวนข้อมูล Seed ที่ต้องใช้ ผลการทดลองแสดงดังตารางที่ 5-33

ตารางที่ 5-33 แสดงผลการทดกำเนิดข้อมูลการทดสอบวงจร S38417 ด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register โดยในสดมภ์ที่ 1 และ2 แสดงตัวเลขที่ใช้ในการแบ่งกลุ่มชุดข้อมูล Seed ที่แสดงในรูปแบบของตัวแปร S_{avg} ส่วนสดมภ์ที่ 2 แสดงในรูปแบบของตัวเลข สดมภ์ที่ 3 และ 4 แสดงจำนวนชุดข้อมูลทดสอบที่มีการเลื่อนบิตข้อมูล Seed ในแบบเต็มและแบบครึ่ง ตามลำดับ สดมภ์ที่ 5 และ 6 จำนวนบิตข้อมูล Seed ต่อกลุ่ม โดยแบ่งเป็น *G1* และ *G2* ตามลำดับ ส่วนสดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบ

ในการกำเนิดข้อมูลการทดสอบวงจร S38417 มีจำนวนชุดข้อมูลทดสอบทั้งหมด 340 ชุด ผลจากตารางที่ 5-33 ตัวเลขที่ใช้แบ่งกลุ่มการเลื่อนบิตข้อมูล Seed และให้ผลลัพธ์ดีที่สุดคือ S_{avg}+4 หรือ 74 บิต จำนวนชุดข้อมูลที่ใช้การเลื่อนบิตข้อมูล Seed แบบเต็ม ที่จำนวน 147 ชุด และจำนวนการเลื่อนบิตข้อมูล Seed แบบครึ่งเท่ากับ 193 ชุด ซึ่งการเลื่อนบิตแบบเต็ม มีการเลื่อนบิตข้อมูล Seed จำนวน 83 บิต และ การเลื่อนบิตข้อมูล Seed แบบครึ่ง มีการเลื่อนบิตข้อมูล Seed จำนวน 68 บิต จำนวนข้อมูล Seed ทั้งหมด เท่ากับ 25,325 บิต

ตารางที่ 5-33 ผลการทดกำเนิดข้อมูลการทดสอบวงจร S38417

ตัวเลขที่ใช้แบ่งกลุ่ม		จำนวนชุดข้อมูลทดสอบ		จำนวนบิตข้อมูล Seed ต่อกลุ่ม		จำนวน Seed ทั้งหมด
		แบบเต็ม	แบบครึ่ง	*G1*	*G2*	
S_{avg}-4	66	240	100	61	22	26,020
S_{avg} -3	67	226	114	62	23	25,826
S_{avg} -2	68	210	130	62	23	25,490
S_{avg} -1	69	194	146	64	19	25,446
S_{avg}	70	172	168	66	17	25,364
S_{avg} +1	71	162	178	67	16	25,372
S_{avg} +2	72	158	182	68	15	25,490
S_{avg} +3	73	155	185	68	15	25,445
S_{avg} +4	**74**	**147**	**193**	**68**	**15**	**25,325**
S_{avg} +5	75	135	205	69	14	25,350

5.3.4.2 สรุปผลการทดสอบของวงจร S38417

ผลการทดลองในตารางที่ 5-34 สดมภ์ที่ 1 แสดงชื่อวงจรทดสอบ S38417 สดมภ์ที่ 2 และ 3 แสดงจำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบของ *G1* และ *G2* ตามลำดับ สดมภ์ที่ 4 แสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบเท่ากับ 25,325 บิต สดมภ์สุดท้ายแสดงเปอร์เซ็นการลดขนาดข้อมูลทดสอบเท่ากับ 95.52% โดยคำนวณจากสมการ (5-1) ส่วนจำนวนสัญญาณที่ใช้ในการทดสอบเท่ากับวิธีการ Parallel LFSR Reseeding ในแนวคิดที่ 1

ตารางที่ 5-34 ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S38417

ชื่อวงจรทดสอบ	จำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ		จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ
	G1	*G2*		
S38417	68	15	25,325	95.52%

5.3.6 ผลการกำเนิดข้อมูลการทดสอบวงจร S38584

5.3.5.1 การกำหนดขนาดของ Seed ในการกำเนิดข้อมูลการทดสอบของแต่ละชุดข้อมูลทดสอบ

กลไกการเลื่อนข้อมูล Seed ด้วยการใช้ Selection Register ในการควบคุมกลุ่มข้อมูล Seed ให้มีการเลื่อนบิตหรือไม่เลื่อน ชุดข้อมูล Seed จะถูกแบ่งเป็น *G1* และ *G2* ส่วนชุดข้อมูลทดสอบแต่ละชุดใช้ค่า S_{avg} เป็นเกณฑ์ในการพิจารณากลุ่มข้อมูล Seed ที่จะถูกเลื่อนเพื่อกำเนิดข้อมูลทดสอบ ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมากกว่า S_{avg} จะทำการเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR แบบเต็ม คือเลื่อนทั้ง *G1* และ *G2* แต่ถ้าจำนวนการเจาะจงค่าของชุดข้อมูลทดสอบมีค่าน้อยกว่า S_{avg} จะทำการเลื่อนข้อมูล Seed แบบครึ่ง คือเลื่อนทั้ง *G1* เท่านั้น ซึ่ง S_{avg} ของวงจร S38584 เท่ากับ 45 บิต ในการกำเนิดข้อมูลทดสอบจะพิจารณาที่ S_{avg}-3, S_{avg}-2, S_{avg}-1, S_{avg}+1, S_{avg}+2, S_{avg}+3, S_{avg}+4, S_{avg}+5, S_{avg}+6 และ S_{avg}+7 เพื่อดูแนวโน้วการเปลี่ยนแปลงของจำนวนข้อมูล Seed และจำนวนชุดข้อมูล Seed ที่ต้องใช้ และพิจารณาหาจำนวน Seed ที่น้อยที่สุดที่ใช้ในการกำเนิดข้อมูลทดสอบ ผลการทดลองแสดงดังตารางที่ 5-35

ตารางที่ 5-35 แสดงผลการทดกำเนิดข้อมูลการทดสอบวงจร S38584 ด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register โดยในสดมภ์ที่ 1 และ2 แสดงตัวเลขที่ใช้ในการแบ่งกลุ่มชุดข้อมูล Seed ที่แสดงในรูปแบบของตัวแปร S_{avg} และ ตัวเลข สดมภ์ที่ 3 และ 4 แสดงจำนวนชุดข้อมูลทดสอบที่มีการเลื่อนบิตข้อมูล Seed ในแบบเต็มและแบบครึ่ง ตามลำดับ สดมภ์ที่ 5 และ 6 แสดงจำนวนบิตข้อมูล Seed ต่อกลุ่ม โดยแบ่งเป็น *G1* และ *G2* ส่วนสดมภ์สุดท้ายแสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบ

ในการกำเนิดข้อมูลการทดสอบวงจร S38584 มีจำนวนชุดข้อมูลทดสอบทั้งหมด 62 ชุด ผลจากตารางที่ 5-35 ตัวเลขที่ใช้แบ่งกลุ่มการเลื่อนบิตข้อมูล Seed และให้ผลลัพธ์ดีที่สุดคือ S_{avg}+7 หรือ 54 บิต จำนวนชุดข้อมูลที่ใช้การเลื่อนบิตข้อมูล Seed แบบเต็ม ที่จำนวน 45 ชุด และจำนวนการเลื่อนบิตข้อมูล Seed แบบครึ่งเท่ากับ 51 ชุด ซึ่งการเลื่อนบิตแบบเต็ม มีการเลื่อนข้อมูล

Seed จำนวน 54 บิต และ การเลื่อนบิตข้อมูล Seed แบบครึ่ง มีการเลื่อนข้อมูล Seed จำนวน 45 บิต จำนวนข้อมูล Seed ทั้งหมด เท่ากับ 2,889 บิต

ตารางที่ 5-35 ผลการกำเนิดข้อมูลการทดสอบวงจร S38584

ตัวเลขที่ใช้แบ่งกลุ่ม		จำนวนชุดข้อมูลทดสอบ		จำนวนบิตข้อมูล Seed ต่อกลุ่ม		จำนวน Seed ทั้งหมด
		แบบเต็ม	แบบครึ่ง	*G1*	*G2*	
S_{avg}-3	42	26	13	49	6	2,984
S_{avg} -2	43	32	17	45	9	2,974
S_{avg} -1	44	36	20	42	12	2,988
S_{avg}	45	39	22	40	14	3,018
S_{avg} +1	46	42	25	37	17	3,048
S_{avg} +2	47	42	30	32	22	2,988
S_{avg} +3	48	42	34	28	26	2,940
S_{avg} +4	49	43	37	25	29	2,941
S_{avg} +5	50	45	46	16	38	2,934
S_{avg} +6	51	45	50	12	42	2,898
S_{avg} +7	**52**	**45**	**51**	**11**	**43**	**2,889**

5.3.6.2 สรุปผลการทดสอบของวงจร S38584

ผลการทดลองในตารางที่ 5-36 สดมภ์ที่ 1 แสดงชื่อวงจรทดสอบ S38584 สดมภ์ที่ 2 และ 3 แสดงจำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบของ *G1* และ *G2* ตามลำดับ สดมภ์ที่ 4 แสดงจำนวน Seed ทั้งหมดที่ใช้ในการกำเนิดข้อมูลทดสอบเท่ากับ 2,889 บิต สดมภ์สุดท้ายแสดงเปอร์เซ็นการลดขนาดข้อมูลทดสอบเท่ากับ 96.81% โดยคำนวณจากสมการ (5-1) ส่วนจำนวนสัญญาณที่ใช้ในการทดสอบเท่ากับวิธีการ Parallel LFSR Reseeding ในแนวคิดที่ 1

ตารางที่ 5-36 ผลการกำเนิดข้อมูลทดสอบสำหรับวงจร S38584

ชื่อวงจรทดสอบ	จำนวนชุดบิตข้อมูล Seed ต่อจำนวนชุดข้อมูลทดสอบ		จำนวนบิต Seed ทั้งหมด	เปอร์เซ็นต์การลดขนาดข้อมูลทดสอบ
	G1	*G2*		
S38584	45	9	2,889	96.81%

5.4 การเปรียบเทียบประสิทธิภาพการลดขนาดข้อมูลการทดสอบ

การเปรียบเทียบประสิทธิภาพการลดขนาดข้อมูลการทดสอบของการกำเนิดข้อมูลทดสอบแบบ LFSR Reseeding ได้ทำการเปรียบเทียบจำนวนข้อมูลทดสอบที่ใช้กับวิธีการ Partial LFSR Reseeding [1], วิธีการ Static LFSR Reseeding [26], และวิธีการที่ได้นำเสนอ Parallel LFSR Reseeding ทั้งสองรูปแบบ โดยผลการเปรียบเทียบได้นำเสนอในหัวข้อ 5.3.1 – 5.3.2

5.4.1 การเปรียบเทียบการลดขนาดข้อมูล Seed ด้วยวิธีการ Parallel LFSR Reseeding และ วิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register

การเปรียบเทียบการลดขนาดข้อมูล Seed ของการกำเนิดข้อมูลการทดสอบของทั้งสองสถาปัตยกรรมแสดงดังตารางที่ 5-37 ซึ่งสดมภ์แรกแสดงชื่อวงจรการทดสอบมาตรฐาน ISCAS 89 จำนวน 6 วงจร สดมภ์ที่ 2 และ 3 แสดงผลของจำนวนข้อมูลการทดสอบของสถาปัตยกรรม Parallel LFSR Reseeding แบบธรรมดา ในสดมภ์ที่ 2 คือจำนวนข้อมูล Seed ต่อ 1 ชุดข้อมูลทดสอบ สดมภ์ที่ 3 คือ จำนวนข้อมูล Seed ทั้งหมด ส่วนผลของจำนวนข้อมูลการทดสอบของสถาปัตยกรรม Parallel LFSR Reseeding ร่วมกับ Selection Register แสดงในสดมภ์ที่ 4, 5 และ 6 โดยในสดมภ์ที่ 4 และ 5 แสดงจำนวนข้อมูล Seed ต่อ 1 ชุดข้อมูลทดสอบในกลุ่ม *G1* และ *G2* ตามลำดับ ส่วนสดมภ์ที่ 6 แสดงจำนวนข้อมูล Seed ทั้งหมด

ตารางที่ 5-37 การผลการลดขนาดข้อมูล Seed ของการกำเนิดข้อมูลการทดสอบด้วย
วิธี Parallel LFSR Reseeding ทั้ง 2 แนวคิด

ชื่อวงจรทดสอบ	Parallel LFSR Reseeding		Parallel LFSR Reseeding ร่วมกับ Selection Register		
	จำนวนข้อมูล Seed ต่อ 1 ชุดข้อมูลทดสอบ	จำนวนข้อมูล Seed ทั้งหมด	จำนวนข้อมูล Seed ต่อ 1 ชุดข้อมูลทดสอบ		จำนวนข้อมูล Seed ทั้งหมด
			G1	*G2*	
S5378	17	510	11	6	**492**
S9234	59	8,142	27	32	**5,742**
S13207	24	3,768	14	10	**2,898**
S15850	34	5,678	27	7	**5,181**
S38417	83	28,220	68	15	**25,325**
S38584	54	3,348	45	9	**2,889**

จากตารางจะเห็นได้ว่า แนวคิดที่ 2 การกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register สามารถลดจำนวนข้อมูล Seed ที่ใช้ในการกำเนิดข้อมูลการทดสอบได้ดีกว่าแนวคิดที่ 1 การกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ทุกวงจรทดสอบ

5.4.2 การเปรียบเทียบการลดขนาดข้อมูล Seed ด้วยวิธีการ Parallel LFSR Reseeding Parallel LFSR Reseeding ร่วมกับ Selection Register กับวิธีการ Original LFSR Reseeding และวิธีการ Partial LFSR Reseeding

การเปรียบเทียบการลดขนาดข้อมูล Seed ของการกำเนิดข้อมูลการทดสอบของ 4 แนวคิด คือ วิธีการ Parallel LFSR Reseeding วิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register กับวิธีการ Original LFSR Reseeding และวิธีการ Partial LFSR Reseeding แสดงดังตารางที่ 5-38 ซึ่งสดมภ์แรกแสดงชื่อวงจรการทดสอบมาตรฐาน ISCAS 89 จำนวน 6 วงจร สดมภ์ที่ 2 แสดงผลของจำนวนข้อมูลการทดสอบของสถาปัตยกรรม Static LFSR Reseeding สดมภ์ที่ 3 แสดงผลของจำนวนข้อมูลการทดสอบของสถาปัตยกรรม Partial LFSR Reseeding และสดมภ์ที่ 4

แสดงผลของจำนวนข้อมูล Seed ในการกำเนิดข้อมูลทดสอบด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register

ตารางที่ 5-38 การเปรียบเทียบจำนวนข้อมูล Seed ของกำเนิดข้อมูลการทดสอบจำนวน 3 วิธี

ชื่อวงจรทดสอบ	Static LFSR Reseeding [26]	Partial LFSR Reseeding [1]	Parallel LFSR with Selection Register
S5378	1,140	502	**492**
S9234	11,178	**5,013**	5,742
S13207	6,908	3,008	**2,898**
S15850	9,686	5,204	**5,181**
S38417	35,700	**24,513**	25,325
S38584	4,650	2,942	**2,889**

จากตารางจะเห็นได้ว่า การกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register สามารถลดจำนวนข้อมูล Seed ได้ดีกว่าวิธีการ Static LFSR Reseeding แต่สำหรับวิธีการ Partial LFSR Reseeding นั้น การกำเนิดข้อมูลทดสอบด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register สามารถลดจำนวนข้อมูล Seed ได้ดีกว่าในวงจร S5378, S13207, S15850 และวงจร S38584 ส่วนวงจร S9234 และวงจร S38417 วิธีการกำเนิดข้อมูลทดสอบด้วยวิธีการ Partial Reseeding ใช้จำนวนข้อมูล Seed ที่น้อยกว่า ทั้งวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register และ Static LFSR Reseeding

5.4.3 การเปรียบเทียบการลดระยะเวลาการทดสอบ

การกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ทั้งสองรูปแบบสามารถลดจำนวนข้อมูล Seed ที่ใช้ในการกำเนิดข้อมูลทดสอบความผิดพลาดแบบทดสอบยากแล้ว ยังสามารถลดระยะเวลาการทดสอบได้อีกด้วย เนื่องจากการกำเนิดข้อมูลทดสอบด้วยวิธีการ Parallel LFSR Reseeding ทั้ง 2 แนวคิด ใช้การเลื่อนข้อมูล Seed เข้าสู่วงจร LFSR และเอาท์พุทเลื่อนเอาท์พุทของวงจรเลื่อนเฟส เข้าสู่เซลล์ตรวจกวาดในรูปแบบขนาน จึงสามารถลดระยะเวลาการทดสอบได้ดีกว่าวิธีการที่ใช้การเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR แบบอนุกรม

การเปรียบเทียบการลดระยะเวลาการทดสอบได้นำเสนอในตารางที่ 5-39 โดยได้นำจำนวนสัญญาณนาฬิกาที่ใช้ของวิธี Parallel LFSR Reseeding มาเปรียบเทียบกับวิธีการ LFSR Reseeding with Seed ordering [30] ในสดมภ์แรกของตารางเป็นการแสดงชื่อวงจรการทดสอบมาตรฐาน ISCAS 89 จำนวน 6 วงจร สดมภ์ที่ 2 แสดงจำนวนนาฬิกาที่ใช้ในการทดสอบของวิธีการ LFSR Reseeding with Seed ordering สดมภ์ที่ 3 แสดงจำนวนนาฬิกาที่ใช้ในการทดสอบของวิธีการ Parallel LFSR Reseeding เปรียบเทียบผลลัพธ์

ตารางที่ 5-39 ตารางเปรียบเทียบจำนวนสัญญาณนาฬิกา

ชื่อวงจรทดสอบ	จำนวนสัญญาณนาฬิกา	
	LFSR Reseeding with Seed ordering [30]	Parallel LFSR Reseeding
S5378	1920	390
S9234	4992	690
S13207	2688	4710
S15850	3904	3006
S38417	N/A	7140
S38584	2624	1736

บทที่ 6
บทสรุปและข้อเสนอแนะ

6.1 บทสรุป

งานวิจัยนี้เป็นการนำเสนอวิธีการกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ซึ่งนำเสนอใน 2 แนวคิด แนวคิดแรกเป็นการนำเสนอวิธีการกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ที่สามารถกำเนิดข้อมูลการทดสอบที่ลดการเกิด Structural Dependency และ Correlation รองรับการทำงานร่วมกับโครงสร้างของเซลล์ตรวจกวาดแบบ STUMPS และสามารถครอบคลุม 100% ของข้อมูลการทดสอบได้ แต่แนวคิดดังกล่าวนี้ ยังมีข้อจำกัดในเรื่องจำนวนข้อมูล Seed ที่ต้องใช้มีจำนวนมากเมื่อเปรียบเทียบกับวิธีการกำเนิดข้อมูลการทดสอบ Partial LFSR Reseeding แต่น้อยกว่าวิธี Static LFSR Reseeding ด้วยข้อจำกัดของทรัพยากรที่ใช้ในการพัฒนา จึงได้นำแนวคิด Selection Register มาใช้ควบคุมกลไกการเลื่อนบิตข้อมูล Seed เข้าสู่วงจร LFSR เพื่อให้สามารถเลือกกลุ่มของการเลื่อนข้อมูล Seed ได้ ซึ่งได้เรียกวิธีการทดสอบวิธีนี้ว่า Parallel LFSR Reseeding ร่วมกับ Selection Register วิธีการ Selection Register ในวงจรกำเนิดข้อมูลทดสอบแบบ Parallel LFSR Reseeding นี้ ทำหน้าที่ควบคุมกลุ่มข้อมูล Seed ที่ถูกแบ่งออกเป็น 2 กลุ่ม คือ *G1* และ *G2* ให้สามารถเลื่อนข้อมูล Seed แบบเต็ม (G1 และ G2) หรือเลื่อนแค่เพียงกลุ่มเดียว (G1) ซึ่งทำให้ลดการเลื่อนบิตข้อมูล Seed ลงในบางชุดข้อมูลทดสอบ

ผลการกำเนิดข้อมูลทดสอบด้วยวงจรมาตราฐาน ISCAS 89 จำนวน 6 วงจร เมื่อเปรียบเทียบจำนวนข้อมูล Seed ที่ใช้ทั้งหมดของวิธีการกำเนิดข้อมูลการทดสอบด้วยวิธี Parallel LFSR Reseeding ร่วมกับ Selection Register (แนวคิดที่ 2) กับวิธี Parallel LFSR Reseeding (แนวคิดที่ 1), Static LFSR Reseeding และ Partial LFSR Reseeding แล้ว วิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register ให้ผลลัพธ์ที่ดีกว่าวิธีการ Parallel LFSR Reseeding (แนวคิดที่ 1) และ Static LFSR Reseeding ในทุกวงจรทดสอบ คือใช้จำนวนข้อมูล Seed ที่น้อยกว่า แต่สำหรับวิธีการ Partial LFSR Reseeding วิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register ให้ผลลัพธ์ที่ดีกว่าจำนวน 4 วงจร นอกจากสามารถลดจำนวนข้อมูล Seed ได้แล้ว วิธีการ Parallel LFSR Reseeding ทั้ง 2 แนวคิด ยังสามารถลดจำนวนสัญญาณนาฬิกาของการทดสอบได้อีก

ด้วย เมื่อทำการเปรียบเทียบจำนวนสัญญาณนาฬิกาที่ใช้กับวิธีการ LFSR Reseeding with Seed Ordering ที่เน้นการลดจำนวนสัญญาณนาฬิกาการทดสอบ ซึ่งผลที่ได้ วิธีการ Parallel LFSR Reseeding ทั้ง 2 แนวคิดให้ผลลัพธ์ที่ดีกว่าวิธีการ LFSR Reseeding with Seed Ordering ถึง 5 วงจร

6.2 ข้อเสนอแนะ

6.2.1 เพิ่มการแบ่งกลุ่มข้อมูล Seed เพื่อเพิ่มประสิทธิภาพการลดข้อมูล Seed

การกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register สามารถลดจำนวนข้อมูล Seed ได้ดีกว่าวิธีการ Static LFSR Reseeding แต่สำหรับวิธีการ Partial LFSR Reseeding นั้น การกำเนิดข้อมูลทดสอบด้วยวิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register สามารถลดจำนวนข้อมูล Seed ได้ดีกว่าในวงจร S5378, S13207, S15850 และวงจร S38584 ส่วนวงจร S9234 และวงจร S38417 วิธีการกำเนิดข้อมูลทดสอบด้วยวิธีการ Partial Reseeding ใช้จำนวนข้อมูล Seed ที่น้อยกว่าทั้ง วิธีการ Parallel LFSR Reseeding ร่วมกับ Selection Register และ Static LFSR Reseeding ซึ่งวิธีการหนึ่งที่จะช่วยเพิ่มประสิทธิภาพการลดขนาดข้อมูล Seed ได้ ก็คือการเพิ่มการแบ่งชุดข้อมูล Seed ออกเป็น 4 กลุ่ม เพื่อควบคุมกลไกการเลื่อนบิตข้อมูล Seed เป็นวิธีการหนึ่งที่อาจจะช่วยลดจำนวนข้อมูล Seed ลงได้ แต่จะเป็นการเพิ่มความยุ่งยากภายวงจรควบคุมแบบ BIST แม้จะสามารถลดข้อมูล Seed ได้จริง แต่ไม่คุ้มกับจำนวนพื้นที่ในการพัฒนาที่ต้องเพิ่มมากขึ้น

6.2.2 เพิ่มขั้นตอนวิธีเพื่อจัดการแบงจำนวนชุดข้อมูล Seed

แนวคิดที่สองของการกำเนิดข้อมูลการทดสอบด้วยวิธีการ Parallel LFSR Reseeding ได้ใช้วิธีการแบ่งกลุ่มข้อมูล Seed ที่จะเลื่อนเข้าสู่วงจร LFSR ด้วยการวิเคราะห์จากจำนวนข้อมูล Seed ทั้งหมดที่ใช้ แล้วทำการ Simulate โดยการปรับจำนวนชุดของข้อมูล Seed ของกลุ่ม *G1* และ *G2* ไปเรื่อยๆ จากนั้นพิจารณาแนวโน้วการเพิ่มขึ้นและลดลงของข้อมูล Seed ที่ใช้วิธีการดังกล่าวนี้ เป็นการแบ่งข้อมูล Seed ที่ต้องใช้เวลาในการ Simulate เพื่อลดเวลาในการ Simulate จึงควรหาขั้นตอนวิธีในการจัดแบ่งจำนวนชุดของข้อมูล Seed ใน *G1* และ *G2* ที่เหมาะสม

บรรณานุกรม

[1] C. V. Krishna, A. Jas and Nur A. Touba, "**Test Vector Encoding Using Partial LFSR Reseeding**," ACM Transactions on Design Automation of Electronic Systems (TODAES), pp.500-516, 2004.

[2] Automated Test Equipment, **วิธีสืบค้นวัสดุสารสนเทศ**. [ออนไลน์]. เข้าถึงได้จาก : http://www.globalspec.com/LearnMore/Labware_Test_ /Lab_Test_Equipment/Automated_Test_Equipment, (วันที่ค้นข้อมูล : 16 กันยายน 2553).

[3] C.V. Krishna and N.A. Touba, "**Hybrid BIST Using an Incrementally Guided LFSR**", *Proceeding of IEEE Symposium on Defect and Fault Tolerance,*pp. 217-224, 2003.

[4] C.E. Stroud**,** (2002)**,** "**A Designer's Guide to Built-in Self-Test**," Vol. 19: Springer

[5] Laung-Terng Wang, Cheng-Wen Wu and Xiaoqing Wen, (2006), **"VLSI Test Principles and Architectures design for testability**," page 308-309, San Francisco: Elsevier

[6] Design For Test, **วิธีสืบค้นวัสดุสารสนเทศ**. [ออนไลน์]. เข้าถึงได้จาก : http://en. wikipedia.org/wiki/Design_For_Test, (วันที่ค้นข้อมูล : 16 กันยายน 2553).

[7] Built-In Self-Test, **วิธีสืบค้นวัสดุสารสนเทศ**. [ออนไลน์]. เข้าถึงได้จาก : http://en. wikipedia.org/wiki/Built-in_self-test, (วันที่ค้นข้อมูล : 16 กันยายน 2553).

[8] BIST, **วิธีสืบค้นวัสดุสารสนเทศ**. [ออนไลน์]. เข้าถึงได้จาก : http://www.siliconfareast.com/bist.htm, (วันที่ค้นข้อมูล : 16 กันยายน 2553).

[9] A.R. Mohamed, Built-In Self-Test (BIST), **วิธีสืบค้นวัสดุสารสนเทศ**. [ออนไลน์]. เข้าถึงได้จาก : http://iroi.seu.edu.cn/books/asics/Book2/CH14 /CH14.7.htm, (วันที่ค้นข้อมูล : 16 กันยายน 2553).

[10] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "**A tutorial on built-in self-test, Part 1: Principles**", *IEEE Design Test Computation*, vol. 10, pp. 1993.

[11] N.A. Touba, and E. J. McCluskey, "**Transformed Pseudo-Random Patterns for BIST**," *IEEE VLSI Test Symposium*, pp. 410-416, 1995.

[12] P. Fiser, **"Pseudo-Random Pattern Generator Design for Column-Matching BIST**", *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, page 657-663, 2007.

[13] P. Fišer, and H. Kubatova, "**Pseudorandom Testing – A Study of the Effect of the Generator Type**," *Czech Technical University in Prague*, 2005.

[14] J. Rajski, N. Tamarapalli, and J. Tyszer, **"Automated synthesis of large phase shifters for built-in self-test**," *Proceedings of International Test Conference*, pp. 1047–1056, 1998.

[15] D. Kagaris, **"A unified method for phase shifter computation"**, *ACM Transactions on Design Automation of Electronic Systems*, pp. 157–167, 2005.

[16] T. Reungpeerakul, D. Kay, and S. Mourad, "**Partial-Matching Technique in Mixed-Mode BIST Environment**," *IEEE Transactions on Instrumentation and Measurement*, pp. 970-977, 2010.

[17] T. Hayashi, N. Hiraiwa, T. Shinogi, H. Takase, and H. Kita, "**Test Modification and Compression Technique for Reducing Total Test Volume with Dictionary Data**," *International Conference on ASIC*, pp. 639-644, 2005.

[18] K.J. Balakrishnan, and F. Lei, "**RTL Test Point Insertion to Reduce Delay Test Volume**," *Proceedings of IEEE VLSI Test Symposium*, pp. 325-332, 2007.

[19] U. Snehal, and K. Dimitri, "**Minimizing Observation Points for Fault Location**," *IEEE International Symposium on Defect and Fault Tolerance in VLSI System*, pp. 263-267, 2009.

[20] A. Jas, C. V. Krishna, and N. A. Touba, "**Weighted Pseudorandom Hybrid BIST**," *IEEE Transaction of VLSI Systems*, pp. 1277-1283, 2004.

[21] Y. Chaowen, S.M. Reddy, and I. Pomeranz, "**Weighted Pseudo-Random BIST for N-detection of Single Stuck-at Faults**," *Proceedings of IEEE Asian Test Symposium*, pp. 178-183, 2004.

[22] Y. Chaowen, S.M. Reddy, and I. Pomeranz, "**Circuit independent Weighted Pseudo-Random BIST Pattern Generator**," *Proceedings of IEEE Asian Test Symposium*, pp. 132-137, 2005.

[23] V.K. Agrawal, and E. Cerny, "**Store and generate built-in testing approach**," *Fault-Tolerant Computing Symposium*, pp.35-40, 1981.

[24] R. Dandapani, J. Patel, and J. Abraham, "**Design of Test Pattern Generators for built-in test**," *International Test Conference*, pp.315-319, 1984.

[25] G. Edirisooriya, and J.P. Robinson, "**Design of Low cost ROM best test generators**," *IEEE VLSI Test Symposium*, pp.61-66, 1992.

[26] S. Hellebrand, J. Rasjki, S. Venkataraman, and B. Countois, "**Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers**," *Proceedings of International Test Conference*, pp. 120-129, 1992.

[27] J. Rajski, J. Tyszer and N. Zacharia, "**Test Data Decompression for Multiple Scan Designs with Boundary Sca**n," *IEEE Transactions on Computers*, pp. 1188-1200, 1998.

[28] H.S. Kim, Y. Kim, and S. Kang, "**Test-Compression Mechanism, Using a Variable-Length Multiple-Polynomial LFSR**," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, pp.687-690, 2003.

[29] A.A Al-Yamani, S. Mitra, and E.J. McCluskey**, "BIST Reseeding with very few seeds,"** *IEEE VLST Test Symposium***,** pp. 69-74, 2003.

[30] A.A Al-Yamani, S. Mitra, and E.J. McCluskey, "**Optimized Reseeding by Seed Ordering and Encoding**," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* pp. 264-270, 2005.

[31] K. Hong-Sik, and K. Sungho, "**Increasing encoding efficiency of LFSR reseeding-base test compression**," *IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems*, pp. 913-917, 2006.

[32] X. Sun, L. Kinney, and B. Vinnakota, "**Combining dictionary coding and LFSR reseeding for test data compression**," *Proceedings of Design Automation Conference*, pp. 944-947, 2004.

[33] A. W. Hakmi, S. Holst, H. J. Wunderlisch, J. Schloffel, F. Hapke, and A. Glowatz, "**Restrict Encoding for Mixed-Mode BIST**," *IEEE VLSI Test Symposium*, pp.179-184, 2009.

[34] K. J. Balakrishnan, "**Efficient Scan-Base BIST using Multiple LFSRs and Dictionary Coding**," *International Conference on VLSI design*, pp. 345-350, 2007.

[35] S. Udar, and D. Kagaris, "**LFSR Reseeding with Irreducible Polynomials**," *IEEE International On-Line Testing Symposium*, pp. 293-298, 2007.

[36] S. Wang, W. Wei, and S.T. Chakradhar, "**A High Compression and Short Test Sequence Test Compression Technique to Enhance Compressions of LFSR Reseeding**," *Proceedings of IEEE Asian Test Symposium*, pp. 79-86, 2007.

[37] Z. Wang, K. Chakrabarty, and M. Bienek, "**A Seed-Selection Method to Increase Defect Coverag for LFSR-Reseeding-Base Test Compression**," *IEEE European Test Symposium*, pp.125-130, 2007.

[38] E.H. Volkerink, and S. Mitra, "**Efficient Seed Utilization for Reseeding based Compression," IEEE VLSI Test Symposium**, *IEEE VLSI Test Symposium*, pp.232-237, 2007.

[39] S. Wang, W. Wei, and Z. Wang, "**A Low Overhead High Test Compression Technique Using Pattern Clustering With n-Detection Test Support**," *IEEE Transactions on Very Large Scale Integration (VLSI) System*, pp. 1-14, 2009.

[40] N. Oh, R. Kapur, and T.W. Willians, "**Fast Seed Computation for Reseeding Shift Register in Test Pattern Compression**," *IEEE/ACM International Conference on Computer Aided Design*, pp.76-81, 2002.

[41] M. Yilmaz, and K. Chakrabarty, "**Seed Selection in LFSR Reseeding-Based Test Compression for the Detection of Small-Delay Defects**," *Design, Automation & Test in Europe Conference & Exhibition*, pp. 1488-1493, 2009.

[42] M. Arai, H. Kurokawa, K. Ichino, S. Fukumoto, and K. Iwasaki, "**Seed Selection procedure for LFSR –based BIST with multiple scan chains and phase shifters**," *Proceedings of IEEE Asian Test Symposium*, pp. 190-195, 2004.

[43] H. Fagn, K. Chakrabarty, and R. Parekhji, "**Bit-Operation-Based Seed Augmentation for LFSR Reseeding with High Defect Coverage**," *Proceedings of IEEE Asian Test Symposium*, pp.331-336, 2009.

[44] C.V. Krishna, and N.A. Touba, "**Hybrid BIST Using an Incrementally Guided LFSR**," *Proceedings of IEEE Symposium on Defect and Fault Tolerance*, pp. 217-224, 2003.

[45] S.N. Neophytou, and M.K. Michael, "**Test Set Generation with a Large Number of Unspecified Bits Using Static and Dynamic Techniques**," *IEEE Transactions on Computers*, pp. 301-316, 2010.

[46] M.H. Yang, Y. Kim, Y. Park, D. Lee, and S. Kang, "**Deterministic built-in self-test using split linear feedback shift register reseeding for low-power testing**,", *Computer and Digital Techniques*, pp. 369-376, 2007.

[47] S. Wang, Z. Wang, W. Wei, and S.T. Chakradhar, "**A Low Cost Test Data Compression Technique for High n-Detection Fault Coverage**," *IEEE International Test Conference*, pp. 1-10, 2007.

[48] M.H. Yang, Y. Kim, Y. Park, D. Lee, and S. Kang, "**Deterministic built-in self-test using split linear feedback shift register reseeding for low-power testing**," *Proceedings of IEEE Asian Test Symposium*, pp. 369 - 376, 2009.

[49] Z. Bin, Y. Yi-zheng, W. Xin-chun, and L. Zhao-lin, "**Reduction of Test Power and Data volume in BIST Scheme Based on Scan Slice Overlapping**," *IEEE International Symposium on Circuits and Systems*, pp. 2737-2740, 2009.

[50] W.D. Tseng, L.J. Lee, and R.B. Lin, "**Deterministic Built-in Self-Test Using Multiple Linear Feedback Shift Registers for test power and test volume reduction**," *Computer and Digital Techniques*, pp 317-324, 2010.

[51] A. Jas, and N.A. Touba, "**Deterministic Test Vector Compression/ Decompression for Systems-on-a-Chip Using and Embedded Procession**," *Journal of Electronic Testing: Theory and Application*, pp.503-514 , 2002.

[52] J. Rahski, M. Kassab, N. Mukherjee, and N. Tamarapalli, "**Embedded Deterministic Test for Low-Cost Manufacturing,**" *IEEE Design & Test of Computer*, pp.58-66, 2003.

[53] Adam B. Kinsman, and Nicola Nicolici, "**Embedded Deterministic Test Exploiting Care Bit Clustering and Seed Borrowing**," *International Symposium on Quality Electronic Design*, pp.832-837, 2008.

[54] D. Czysz, G. Mrugalski, J. Rajski, and J. Tyszer, "**Low Power Embedded Deterministic test**," *IEEE VLSI Test Symposium*, pp. 75-83, 2007.

[55] N.A. Touba, "**X-Canceling MISR-An X Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR**," *International Test Conferenc*e, pp. 1-10, 2007.

[56] R. Garg, R. Putman, and N.A. Touba, "**Increasing Output Compaction in Presence of Unknowns Using an X-Canceling MISR with Deterministic Observation**," *IEEE VLSI Test Symposium,* pp. 35-42, 2008.

[57] G. Jervan, E. Orasson, R. Ubar, and H. Kruus, "**Hybrid BIST optimization using reseeding and test set compaction**," *Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pp. 596-603, 2008.

[58] R. Ubar, H. Kruus, Z. Peng, and G. Jervan, "**Using Tabu Search Method for optimization the Cost of Hybrid BIST**," *International Biennial Baltic Electronics Conference (BEC 2008)*, pp. 155-158, 2008.

[59] M. Zivkovic, "**A Table of Primitive binary Polynomials**", Mathematics Subject Classification, pp.1-22, 1980.

ประวัติผู้เขียน

ชื่อ สกุล นางสาวปิยนาฏ คงทิม

รหัสประจำตัวนักศึกษา 5110120025

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยวลัยลักษณ์	2550

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

ทุนผู้ช่วยวิจัย คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ประจำปีการศึกษา 2551

การตีพิมพ์เผยแพร่ผลงาน

P. Kongtim, T. Reungpeerakul, “Parallel LFSR Reseeding for Mixed-mode BIST,” *International conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pp. 198-202, 2010.

P. Kongtim, T. Reungpeerakul, “Parallel LFSR Reseeding with Selection Register for Mixed-mode BIST,” *Proceeding of IEEE Asian Test Symposium*, pp. 153-138, 2010.

ภาคผนวก ก

บทความทางวิชาการที่นำเสนอใน

International Conference on Electrical/Electronics, Computer,

Telecommunications and Information Technology (ECTI-CON 2010)

19-21 พฤษภาคม 2553 จ. เชียงใหม่

ECTI-CON 2010

The 2010 ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology

Empress Convention Centre
Chiang Mai, Thailand
19-21 May 2010

IEEE Catalog Number: CFP1006E-CDR
ISBN: 978-974-672-491-3
Organized by ...

ECTI Association
IEEE THAILAND SECTION

TPM1-1-4 (1314) : Parallel LFSR Reseeding for Mixed-Mode BIST

P. Kongtim* and Asst. Prof. Dr. T. Reungpeerakul
Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University,
Hat Yai, Songkhla 90112
*E-mail: s5110120025@psu.ac.th**

Abstract

In this paper, a novel parallel LFSR reseeding technique for mixed-mode BIST that is suitable and applicable to a multiple scan chain design. This approach can be applied to generate test cubes that detect Random Pattern Resistant (RPR) faults. A multiple test vector is used to guide the LFSR in order to generate target test cube at the application time. The encoded test seed is solved by using a system linear equation. Experimental results have been discussed by performing the largest ISCAS 89 benchmark circuits. Advantages: 100% test coverage, reduction of test application, reduction of test data storage, requiring few additional hardware, high fault coverage as intended by the deterministic test, and capability to generate any deterministic test cubes without proportional to the largest number of specified bits.

Parallel LFSR Reseeding for Mixed-mode BIST

P. Kongtim* and Asst. Prof. Dr. T. Reungpeerakul
Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University,
Hat Yai, Songkhla 90112
E-mail: s5110120025@psu.ac.th*

***Abstract*- In this paper, a novel parallel LFSR reseeding technique for mixed-mode BIST that is suitable and applicable to a multiple scan chain design. This approach can be applied to generate test cubes that detect Random Pattern Resistant (RPR) faults. A multiple test vector is used to guide the LFSR in order to generate target test cube at the application time. The encoded test seed is solved by using a system linear equation. Experimental results have been discussed by performing the largest ISCAS 89 benchmark circuits. Advantages: 100% test coverage, reduction of test application, reduction of test data storage, requiring few additional hardware, high fault coverage as intended by the deterministic test, and capability to generate any deterministic test cubes without proportional to the largest number of specified bits.**

I. INTRODUCTION

Most of the improvement trends have resulted principally from the industry's ability to exponentially decrease the minimum feature sizes used to fabricate integrated circuits. The most frequently cited trend is in integration level, which is usually expressed as Moore's Law [1]. Continued scaling feature sizes has made the integration of several cores in a single monolithic integrated circuit possible, called system on a chip (SOC). As the number of cores integrated in a SOC increased rapidly, both the test data storage requirements on the tester and the test bandwidth requirements between the tester and the chip have grown dramatically [2]. One of the most effective methods is to use a Built-In Self-Test (BIST), whose main purpose is to reduce the complexity, the test application time, and the number of automatic test equipment (ATE) channels. The most economical BIST schemes are based on pseudorandom pattern generation which usually yields low fault coverage due to the presence of Random Pattern Resistant (RPR) faults.

A large body of research has been dedicated to increasing the probability of randomly generated test pattern detecting RPR faults. This includes three main categories:

1) Modification of the Circuit-Under-Test (CUT) [3-5] in order to improve fault coverage by redesigning or by inserting test points. This approach degrades system performance, causes elongation of the design cycle, but is not always possible because of performance restriction or intellectual property reasons.

2) Use of Weighted Random Pattern Generation [6-8] to increase the probability for test patterns to detect faults. The weighted random pattern generation is biased by adding extra logic circuits. This can lead to large area overhead.

3) Mixed-mode on test pattern generation [9-26] to use STUMPS architecture [27]. There are two phrases of test pattern generation: pseudorandom patterns to detect the random pattern testable faults, and deterministic scan vectors to detect RPR faults.

For the third category, several techniques for improving the efficiency of the test pattern generation have been proposed to compress the amount of test data stored on the tester. These include using run length code [9], applying fixed-length code word [10-11], encoding test data based on Huffman coding [12], translating the original test data into symbols related to the frequency of occurrence [13-15], applying dictionary coding [16-17], adding mapping scheme [18], fixing logic method [19], fixing and flipping logic approach [20], and developing LFSR reseeding techniques [21-26].

In this paper, we focus on parallel LFSR reseeding in a mixed-mode BIST. The reseeding technique proposed in this paper is a dynamic reseeding method in which the parallel seeds are modified incrementally while the test generation proceeds. It does not only reduce test data and test application time, but it is also realistic and applicable. The efficiency of encoded test data relies on the number of specifies bits and the number of test patterns. This technique guarantees 100% test coverage while keeping the fault coverage as high as intended by the deterministic test.

The rest of this paper is organized as follows. Section 2 describes previous work. In Section 3, we present the overview and architecture of proposed method. Section 4 discusses forming and solving linear equations for parallel reseeding. The experimental results are provided in Section 5, with the conclusion following in Section 6.

II. PREVIOUS WORK

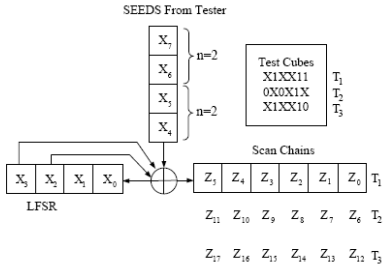
The original LFSR reseeding method has been proposed in [21]. The improved methods have been developed in several ways such as multiple-polynomial LFSR [22], variable-length seeds [23] [24], etc. Note that all of mentioned approaches have involved static reseeding. An alternative reseeding approach is a dynamic reseeding, called partial LFSR reseeding [25] [26]. The results of the partial LFSR reseeding seem to offer good features, but the assumption and calculation of seeds are not realistic and applicable in the test application based on STUMPS architecture. The clarification will be presented through the rest of this section.

The sentence of "Note that for simplicity, the example shows an LFSR feeding a single scan chain, however without loss of generality, the same procedure would apply for an LFSR feeding multiple scan chains" has been specified in [25].

In other words, seeds are loaded in serial form which is in contrast with multiple scan chains and phase shifter structure. In the real test, seeds have to be calculated in such a way of association with both the positions of scan cells and shift cycles that feed through multiple taps of the LFSR and phase shifter in parallel form.

As an illustration, the architecture of the partial LFSR reseeding and forming equations proposed in [25] are shown in Fig. 1. It is unrealistic in several cases:

- All seeds are fed in serial form through only one tap, X_0 in Fig.1.



Test Cube T_1	$Z_0 = X_3 \oplus X_3$ $Z_1 = X_1 \oplus X_2$ $Z_2 = X_3 \oplus X_1$ $Z_3 = X_0 \oplus X_2 \oplus X_3$ $Z_4 = X_1 \oplus X_3$ $Z_5 = X_0 \oplus X_2$
Test Cube T_2	$Z_6 = X_1 \oplus X_2 \oplus X_3 \oplus X_4$ $Z_7 = X_0 \oplus X_1 \oplus X_2 \oplus X_5$ $Z_8 = X_0 \oplus X_1 \oplus X_3 \oplus X_3$ $Z_9 = X_0 \oplus X_1 \oplus X_3 \oplus X_4$ $Z_{10} = X_0 \oplus X_3 \oplus X_4 \oplus X_5$ $Z_{11} = X_3 \oplus X_5$
Test Cube T_3	$Z_{12} = X_2 \oplus X_5 \oplus X_6$ $Z_{13} = X_1 \oplus X_6 \oplus X_7$ $Z_{14} = X_0 \oplus X_6$ $Z_{15} = X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6$ $Z_{16} = X_1 \oplus X_2 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7$ $Z_{17} = X_0 \oplus X_1 \oplus X_4 \oplus X_5 \oplus X_7$

Figure 1. Example of Forming Equations for Partial Reseeding in [25]

- Some bits of a test cube are formed by consisting more than one bit of new seeds, for example Z_{10} contained both X_4 and X_5. These bits cannot be assigned to a position related with the first shift cycle because one bit of new seeds is supposed to shift at once per cycle, otherwise new seeds have to be fed in parallel form. In the multiple scans design, these scan cells could be assigned to the position related with the first shift cycle which will make this method does not applicable.

- If the number of bits in new seeds is less than the number of scan chains, no information on how to deal with additional scan chains, otherwise extra bits of seeds are supposed to add. For example in test cube t_2, if 6 scan cells are assigned into 3 scan chains, two bits of the new seed are fed into a couple scan chains, but it is unclear on how to deal with the seed related to another scan chain. In the worst case, if the scan cell in that scan chain lied in the first cycle is a specified bit, there is no way to control in order to meet the target value, for example Z_8 and Z_{14}. It is not controllable because it is not contained any bit of new seeds.

For above reasons, the assumption and calculation of partial LFSR reseeding [25] are simplicity. In the real test application based on multiple scans architecture, it is much more complicated. We propose a novel method to address the stated problems with parallel LFSR reseeding in mixed-mode BIST detailed in Section 3 and 4.

III. OVERVIEW OF PROPOSED METHOD

The architecture of parallel LFSR reseeding is illustrated in Fig. 2. Extra XOR gates are added in the LFSR for feeding seeds. AND gates are inserted between the LFSR and seeds. One input of each AND gate is connected to a sub-section of seeds. Another input of AND gate is a control signal, C, which is used to either enable or disable data from seeds. New seeds can be fed into the LFSR by controlling C to be "1", otherwise the contents of scan cells are received from the combination of the LFSR through the phase shifter without feeding any data from new seeds. Seed length, L, is proportional to the average number of specified bits per test cube, S_{avg}. In the worst case, L equals to scan length, M, which means the test cubes are not able to be encoded into a smaller data. Nevertheless, it guarantees to generate deterministic test patterns with 100% test coverage. The number of feeding points of seeds is the same number of scan chains, but it is usually less than the number of the LFSR sizes. The minimum value of test vectors, $T_{P,N}$, associated with scan chain number are solved by the system of linear equations. Note that P is the number of test patterns and N is the number of scan chains. For example, $T_{1,2}$ is an encoded test vector for targeting test pattern 1 located in scan chain 2. Tap selection of the phase shifter has been presented in [28].

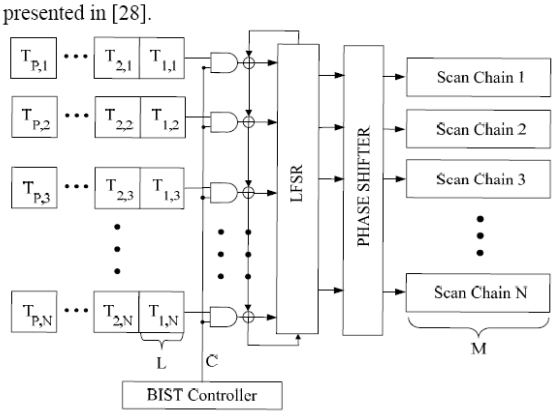


Figure 2. Architecture of Parallel LFSR Reseeding

IV. FORMING AND SOLVING EQUATION FOR PARALLEL RESEEDING

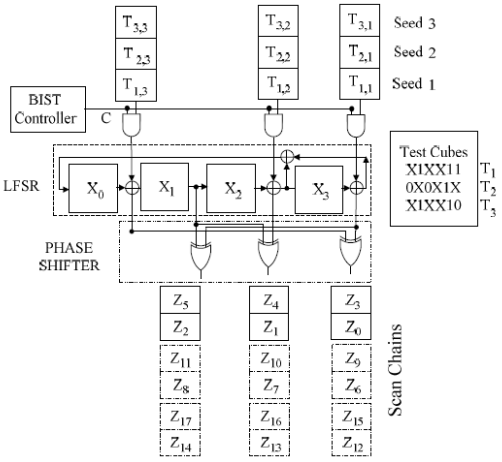
After the BIST controller allows pseudorandom patterns generated to detect the random pattern testable faults, parallel LFSR reseeding performs the generation of test cubes that target RPR faults. We propose a new method to inject parallel seeds through LFSR and phase shifter in order to guide them to supply the desired test cubes in a multiple scan design. The seed sequence of test vectors, $T = (T_{1,1}, T_{1,2}, \ldots, T_{P,N})$, is fed by controlling C signal. The C signal is forced to logic '1' in same period as the number of cycles associated with L. Otherwise, the remaining of test cubes is fed by LFSR without getting data from seeds which the C signal will hold logic '0' as M-L cycles.

We illustrate the parallel LFSR reseeding method using the example of Fig. 3 and assume that there are only three scan chains injected seeds through three LFSR taps. The outputs of phase shifter are expressed in terms of the LFSR taps as follows: outputs for scan chain 1, 2, and 3 are represented by $X_0 \oplus X_3$, $X_1 \oplus X_2$, and $X_1 \oplus X_3$, respectively. For the next step, the tap positions of LFSR needed to support for seeds injection are calculated and selected in such a way that each test vector fully controls particular part of test cubes of individual scan chain. For this particular example, the outputs of phase shifter for scan chain 1, 2, and 3 are chosen from LFSR tap 0, 2, and 3, respectively.

Each group of seed is contained test vectors as the same number of scan chains. For example, seed 1 consists of three test vectors: $T_{1,1}$, $T_{1,2}$, and $T_{1,3}$. Seed 2 consists of three test vectors: $T_{2,1}$, $T_{2,2}$, and $T_{2,3}$ and so on. The encoded seed 1 is injected through LFSR and phase shifter for targeting test cube 1. The length of encoded seed is normally in between the maximum number of specified bits, S_{max}, and S_{avg}. For this example, the minimum seed length is 3 because S_{max} equals to S_{avg}, thus all equations related to three test cubes are formed in Fig. 3.

The variables of X_0, X_1, X_2, and X_3 are actually constant values resulting from last cycle of pseudorandom pattern generation. They do not affect the determination of solution. After the linear equations have been formed, they can then be solved by using Gauss-Jordan elimination. For example in Fig. 3, test cubes T_1, T_2, and T_3 are X1XX11, 0X0X1X, and X1XX10, respectively. A solution is obtained by considering only specified bits of test cubes. One solution is $T_{1,1} = 0$, $T_{1,2} = 1$, $T_{1,3} = 1$, $T_{2,1} = 0$, $T_{2,2} = 1$, $T_{2,3} = 1$, $T_{3,1} = 0$, $T_{3,2} = 1$, and $T_{3,3} = 0$.

The experimental results for parallel LFSR reseeding are shown in Section 5.



Test Cube T_1	Test Cube T_2	Test Cube T_3
$Z_0 = T_{1,1} \oplus T_{1,3}$	$Z_6 = T_{1,2} \oplus T_{2,1} \oplus T_{2,3}$	$Z_{12} = T_{1,3} \oplus T_{2,2} \oplus T_{3,1} \oplus T_{3,3}$
$Z_1 = T_{1,2}$	$Z_7 = T_{1,1} \oplus T_{1,2} \oplus T_{1,3} \oplus T_{2,2}$	$Z_{13} = T_{1,2} \oplus T_{1,3} \oplus T_{2,1} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{3,2}$
$Z_2 = T_{1,1}$	$Z_8 = T_{1,1} \oplus T_{1,2} \oplus T_{2,1}$	$Z_{14} = T_{1,1} \oplus T_{1,2} \oplus T_{1,3} \oplus T_{2,1} \oplus T_{2,2} \oplus T_{3,1}$
$Z_3 = T_{1,1}$	$Z_9 = T_{2,1}$	$Z_{15} = T_{1,2} \oplus T_{1,3} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{3,1} \oplus T_{3,2} \oplus T_{3,3}$
$Z_4 = T_{1,3}$	$Z_{10} = T_{1,1} \oplus T_{2,3}$	$Z_{16} = T_{1,1} \oplus T_{1,2} \oplus T_{2,1} \oplus T_{3,3}$
$Z_5 = T_{1,2} \oplus T_{1,3}$	$Z_{11} = T_{1,2} \oplus T_{1,3} \oplus T_{2,2} \oplus T_{2,3}$	$Z_{17} = T_{1,1} \oplus T_{1,3} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{3,2} \oplus T_{3,3}$

Figure 3. Example of Forming Equations for Parallel Reseeding

V. Experimental Results

The largest ISCAS 89 benchmark circuits [29] have been performed to demonstrate the effectiveness of the proposed technique. The set of test cubes targeted the RPR faults is the same as applied in [25].

In table 1, the first column lists the circuit used in the experiment. The second column gives the number of the scan elements. The third column lists the number of test cubes. The last column shows the LFSR size.

TABLE 1. ISCAS 89 benchmark circuit

Circuit Name	Scan Elements	# Test Cubes	LFSR Size
S5378	214	30	38
S9234	247	138	81
S13207	700	157	44
S15850	611	167	58
S38417	1,664	340	105
S38584	1,464	62	75

Table 2 shows a comparison of parallel LFSR reseeding with standard LFSR reseeding [21] and partial LFSR reseeding [25]. For a fair comparison, the exact same set of test cubes and LFSR sizes are used to encode. For each reseeding scheme, three main points are shown: the number of bits per seed, the total number of encoded bits, and the overall test reduction (*TR*). In the third, the sixth, and the ninth columns of the table, the overall *TR* is compared between three schemes. The overall *TR* is defined by:

$$TR = \frac{total\ test\ bits - total\ encoded\ bits}{total\ test\ bits} \times 100.$$

As can be seen, the proposed parallel LFSR reseeding provides much more *TR* than standard LFSR reseeding [21]. Even through the *TR* for parallel reseeding is slightly less than the partial reseeding [25], the proposed scheme is clearly realistic and applicable based on multiple scan chains architecture.

VI. Conclusion

Parallel LFSR reseeding is a novel approach for encoding test data in order to apply in mixed-mode BIST. The advantages of our techniques include:

- Applicable and suitable for a multiple scan chain configuration
- 100% test coverage
- The capability to handle with any deterministic test pattern without proportional to the largest number of specified bits, S_{max}, in any test cube
- High fault coverage as intended by the deterministic test
- A reduction in the test application time and the test data storage
- Few additional hardware required beyond the need of STUMPS architecture (only one 2-input AND and XOR gates per scan chain at most)

The test application mode decompresses the encoded test data to guide the LFSR in order to generate the desired test patterns instead of directly applying the whole deterministic patterns. The test reduction (*TR*) over the standard LFSR reseeding [21] is quite substantial, but slightly under the partial reseeding [25], however, it is suitable for a multiple scan chain architecture. Moreover, this approach guarantees 100% test coverage.

One drawback of parallel LFSR reseeding is that the seed length is fixed for every test cube in one test set. The dynamic variable-length reseeding in parallel form might be one of good approaches to reduce more test storage requirements, but it might face with the complexity on both seed computation and mechanism of seed injection. This leads to our future works.

TABLE 2. Comparison of three schemes

Circuit Name	Standard LFSR Reseeding [21]		Partial LFSR Reseeding [25]			Proposed Parallel LFSR Reseeding		
	Total Bits	TR (%)	Bits per Seed	Total Bits	TR (%)	Bits per Seed	Total Bits	TR (%)
S5378	1,140	82.42	16	502	92.18	17	510	92.05
S9234	11,178	67.15	36	5,013	85.29	59	8,142	76.11
S13207	6,908	93.71	19	3,008	97.26	24	3,768	96.57
S15850	9,686	90.50	31	5,204	94.89	34	5,678	94.43
S38417	35,700	93.68	72	24,513	95.66	83	28,220	95.01
S38584	4,650	94.87	47	2,942	96.75	54	3,348	96.31

VII. REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS), http://www.itrs.net/Links/2009ITRS/Home2009.htm Last visited: Jan 29, 2010.

[2] Khoche, A., and J. Rivoir, "I/O Bandwidth Bottleneck for Test: Is it Right?," *Proc. of International Workshop on Test Resource Partitioning*, 2000.

[3] C.H. Chiang, and S.K. Gupta, "Random Pattern Testable Logic Synthesis," *Proc. of ICCAD*, pp. 125-128, 1994.

[4] M. Chatterjee, D.K. Pradhan, and W. Kunz, "LOT: Logic Optimization with Testability- New Transformations using Recursive Learning," *Proc. of ICCAD*, pp. 318-325, 1995.

[5] N.A. Touba, and E.J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VTS*, pp. 2-8, 1996.

[6] B. Pouya, and N.A. Touba, "Modifying User-defined Logic for Test Access to Embedded Cores," *Proc. of ITC*, pp. 60-68, 1997.

[7] D.J. Neebel, and C.R. Kime, "Cellular Automata for Weighted Random Pattern Generation," *Trans. on Computers*, pp. 1219-1229, 1997.

[8] S. Ghos, E. MacDonald, S. Basu, N.A. Touba, "Testing: Low-Power Weighted Pseudo-Random BIST Using Special Scan Cells," *Proc. of GLSVLSI*, pp. 86-91, 2004.

[9] A. Jas and N.A. Touba, "Test Vector decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs," *Proc. Int'l Test Conf. (ITC 98)*, pp.458-464, 1998.

[10] A. Chandra and K. Chakrabarty, "System-on-a-Chip Test-Data Compression and Decompression Architectures Based on Golomb Codes," *IEEE Trans. Computer-Aided Design*, vol. 20, no. 3, pp. 355-368, 2001.

[11] P.T. Gonciari, B.M. Al-Hashimi, and N. Nicolici, "Variable-Length Input Huffman Coding for System-on-a-Chip Test," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 6, pp. 783-796, 2003.

[12] A. Jas et al., "An Efficient Test Vector Compression Scheme Using Selective Huffman Coding," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 6, pp. 797-806, 2003.

[13] P.T. Gonciari, B.M. Al-Hashimi, and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-Chip Test Data Compression/Decompression," *Proc. Automation and Test in Europe Conference and Test in Europe Conference and Exhibition*, 2002

[14] S.M. Reddy et al., "On Test Data Volume Reduction for Multiple Scan Chain Designs," *Proc. VLSI Test Symposium*, pp. 103-108, 2002.

[15] Armin Würtenberger, Christofer S. Tautermann and Sybille Hellebrand, "A hybrid coding strategy for optimized test data compression," *Proceedings IEEE Test Conference*, pp.944 – 947, 2003.

[16] L. Li, K. Chakrabarty, and N.A. Touba, "Test Data Compression Using Dictionaries with Selective Entries and Fixed-Length Indices," *ACM Trans. Design Automation Electrical Systems*, vol. 8, no. 4, pp. 470-490, 2003.

[17] A. Würtenberger, C.S. Tautermann, and S. Hellebrand, "Data Compression for Multiple Scan Chains Using Dictionaries with Corrections," *Proc. Int'l Test Conf.*, pp. 926-935, 2004.

[18] N.A. Touba and E.J. McCluskey, "Synthesis of Mapping Logic for Generating Transformed Pseudo-Random Patterns for BIST," *Proc. of IEEE International Test Conference*, pp. 674-682, 1995.

[19] N.A. Touba and E.J. McCluskey, "Bit-Fixing in Pseudo-Random Sequences for Scan BIST," *Transactions on Computer-Aided Design*, Vol. 20, No. 4, pp.545-555, 2001.

[20] Gang Zeng and Hideo Ito, "Hybrid BIST for System-on-a-Chip Using an Embedded FPGA Core," *Proc. of IEEE VLSI Test Symposium*, 2004.

[21] B. Konemann, "LFSR-Coded Test Patterns for Scan Designs," *Proc. of European Test Conference*, pp.237-242, 1991.

[22] S. Hellebrand, J. Rasjki, S. Venkataraman and B. Countois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *Proc. of International Test Conference*, pp. 120-129, 1992.

[23] N. Zacharia , J. Rajski, J. Tyszer, and J. Waicukauski "Two Dimensional Test Data Decompressor for Multiple Scan Designs," *Proc. of International Test Conference*, pp. 186-194, 1996.

[24] J. Rajski, J. Tyszer, and N. Zacharia, "Test Data Decompression for Multiple Scan Designs with Boundary Scan," *IEEE Transactions on Computers*, Vol. 47, No. 11, pp. 1188-1200, Nov. 1998.

[25] C.V. Krishna, A. Jas, and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," *Proc. of IEEE International Test Conference*, pp. 885-893, 2001.

[26] C.V. Krishna and N.A. Touba, "Hybrid BIST Using an Incrementally Guided LFSR," *Proc. of IEEE Symposium on Defect and Fault Tolerance*, pp. 217-224, 2003.

[27] P. H. Bardell andW. H. McAnney, "Self-testing of multichip logic modules," *Proc. Int. Test Conf.*, pp. 200–204, 1982.

[28] D. Kagaris, "A unified method for phase shifter computation," ACM *Transactions on Design Automation of Electronic Systems*, pp. 157–167, 2005.

[29] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. Of International Symposium on Circuits and Systems*, pp. 1929–1934, 1989.

ภาคผนวก ข

บทความทางวิชาการที่นำเสนอใน

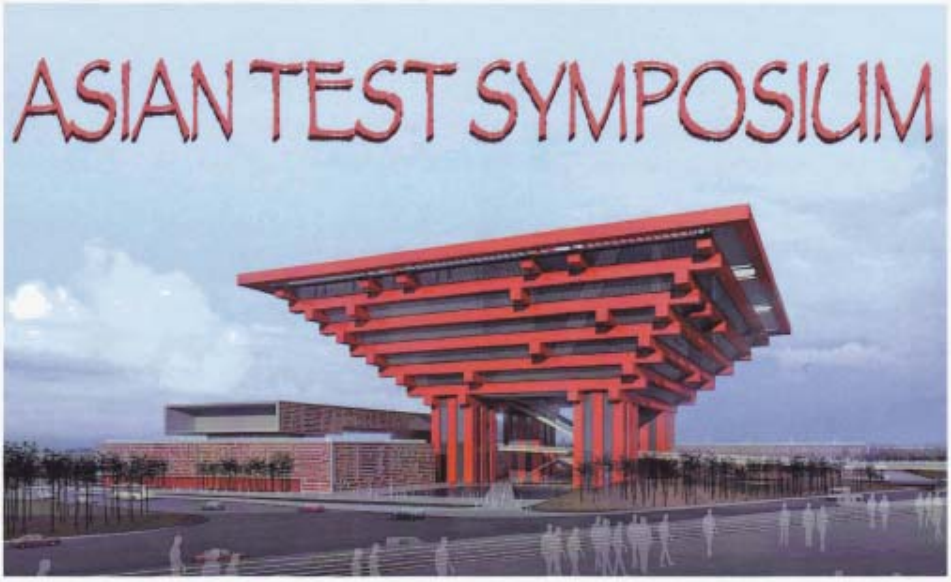
The 19th International Conference on IEEE Asian Test Conference (ATS 2010)

1-4 ธันวาคม 255 เซี่ยงไฮ้ ประเทศจีน

ATS

PROCEEDINGS OF THE NINETEENTH

ASIAN TEST SYMPOSIUM



1–4 DECEMBER 2010 • SHANGHAI, CHINA

IEEE computer society

ttc™

IEEE

Sponsored by
IEEE Computer Society TTTC
Shanghai University

In cooperation with
Shanghai Normal University
National Natural Science Foundation of China (NSFC)
Fault Tolerant Computing Technical Committee of CCF
Shanghai Key Lab. of Computer Software Testing and Evaluating

2010 19th IEEE Asian Test Symposium

PARALLEL LFSR RESEEDING WITH SELECTION REGISTER FOR MIXED-MODE BIST

Piyanart Kongtim [1] and Taweesak Reungpeerakul [2]
[1] E-mail: s5110120025@psu.ac.th , [2] rtaweesak@coe.psu.ac.th
Department of Computer Engineering, Faculty of Engineering,
Prince of Songkla University, Hat Yai, Songkhla 90112

Abstract—**This paper presents a new parallel LFSR reseeding with selection register for mixed-mode BIST in order to reduce the number of test data. The dynamic seeds were injected in parallel form through the LFSR, phase shifter, and scan chains, respectively. The selection register was added to improve the efficiency of encoding test data. For seed computing, after the equations were formed, they were solved by Gauss-Jordan elimination. The experimental results for ISCAS 89 benchmark circuits indicate the significant improvement in terms of test data. There are several main advantages of proposed approach such as 100% test coverage, low test data, low test application time, high fault coverage as intended by the deterministic patterns, etc.**

Keywords: *Parallel LFSR Reseeding, Mixed-mode BIST, Selection Register*

I. INTRODUCTION

The complexity of integrated circuits is increasing rapidly with lower feature size and higher components. The components are integrated in a chip such as processor, I/O interface, memory, etc, called System on Chip (SoC). The complexity and the large amount of transistors of SoC is becoming a major problem in order to gain higher yield of the product, thus attractive test strategies are essential. One of efficient test strategies is to use Built-in Self-Test (BIST) embedded in SoC. BIST has several advantages such as low cost, ease of implementation and low test data storage [1], [2].

One of the BIST schemes is to use pseudo random patterns for achieving low hardware overhead. However, it usually yields low fault coverage due to the presence of random pattern resistant (r.p.r.) faults. Several researches have been proposed to coverage r.p.r. faults described into 3 categories [3].

1. Modification of the Circuit under Test [4-7] has been proposed to increase fault coverage by inserting control and observation points of the circuit under test. These include using ATPG heuristics [4], describing a method of identification and insertion register transfer level [5], presenting reordering scan cells [6], and applying technique to insert observation points before and after running automatic test pattern [7]. However, this category might reduce performance of the SoC by system restriction and intellectual property seasons.

2. Weighted Random Pattern Generation (WRPG) [8-11] has been focused on increasing the frequency of occurrence of "1" and "0" logic values of Linear Feedback Shift Register (LFSR) output by adding weighted logic. The WRPG is used in several ways: the reduction of test data storage on tester [8], the detection single struck at fault [9], bit-flipping [10], and column-matching and redesign LFSR with weighted logic [11]. WRPG approach may increase probability of detecting fault, but risk for area overhead by adding extra logic.

3. Mixed-mode BIST architecture [12-21] has been used to generate both pseudo random patterns and deterministic test patterns. Pseudo random patterns are generated to detect normal fault and deterministic test patterns are developed to detect the r.p.r. faults. The application to reduce test data storage was proposed dictionary based to compress test data [12-14], applied four techniques to compact test data: selection of first vector, Hamming distance base reordering, columnwise bit stuffing, and difference vector [15], discussed bit-fixing [16-18] and described bit-flipping [19]. Dynamic LFSR Reseeding has been discussed in [20-21]. The main advantages of this technique are to reduce area overhead and not deteriorate competency of SoC and cover a certain portion of r.p.r. faults.

The flexibility LFSR reseeding strategy can be described in two types; static and dynamic. The static LFSR reseeding proposed to store seeds instead of the full test cubes in order to generate test patterns. The published papers of static LFSR reseeding proposed methods for conductively of test data compression such as the original of LFSR reseeding [22], Multi-polynomial LFSR [23-24], mapping load data [25], LFSR reseeding based on fixing logic [26], pattern independent design independent seed compression technique [27], and seed ordering [28]. For the second type, the dynamic LFSR reseeding [20-21] is an essential part of test generation procedure that injects free variables from tester to LFSR through phase shifter for gathering or scattering output to scan chains.

In this paper, we focus on parallel LFSR reseeding with selection register for Mixed-mode BIST. The parallel LFSR reseeding technique proposed in this paper is a dynamic reseeding method in which the parallel seeds are modified incrementally while the test generation proceeds. This approach guarantees 100% test coverage while keeping the fault coverage as high as intended by the deterministic test. It provides low test data, low test application time, and applicable with multiple scan design.

1081-7735/10 $26.00 © 2010 IEEE
DOI 10.1109/ATS.2010.35

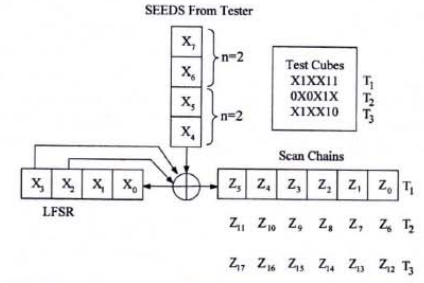
The rest of this paper is organized as follows. In section II, we give the previous work of LFSR reseeding. In section III discusses the overview of proposed method. In section IV, we describe forming and solving linear equation. In Section V, the experimental results with benchmark ISCAS 89 circuit are presented and compared with previous work. Finally, we conclude the paper in section VI.

II. PREVIOUS WORK

This section describes the publications of LFSR reseeding for mixed-mode BIST with dynamic design. In [20] and [21] presented dynamic reseeding and partial reseeding scheme, respectively. The partial reseeding has been proposed encoding test cubes with S_{avg}, where S_{avg} is the average number of specific bits of test patterns. The assumption and calculation of seeds in partial LFSR approach are not realistic and applicable in test application base on STUMPS architecture. From [20], the sentence of "Note that for simplicity, the example shows an LFSR feeding a single scan chain, however without loss of generality, the same procedure would apply for an LFSR feeding multiple scan chains" can be implied that seeds are loaded in serial form which is in contrast with multiple scan chains and phase shifter structure. In the real test, seeds which are fed through multiple taps of the LFSR and phase shifter in parallel form are calculated in association with both the positions of scan cells and shift cycles. The clarification of confliction in partial reseeding scheme can be described in 3 reasons:

1. *Unsuitable with STUMPS architecture:* All seeds are fed in serial form to scan cell through only one tap, X_0 in Fig. 1, which is unsuitable for multiple scan design of STUMPS.
2. *Structural dependency in LFSR*: The Structural dependency in output of LFSR is the fact that the patterns coming from each stage of the LFSR are identical to those of the previous stage but delayed by one clock cycle. The structural dependency of LFSR is reduced by using XOR network called phase shifter. In Fig. 1, partial LFSR reseeding scheme [20] uses output of LFSR in serial mode, which is not suitable with XOR network that shifts in parallel form.
3. *Conflict of equations for STUMPS*: Some bits of a test cube are formed by consisting more than one bit of new seeds, for example Z_{10} contained both X_4 and X_5 in Fig. 1. These bits cannot be assigned to a position related with the first shift cycle because one bit of new seeds is supposed to shift at one per cycle, otherwise new seeds must be fed in parallel form. In the STUMPS architecture, these scan cells could be assigned to the position related with the first shift cycle which will make this method not applicable.

From above reasons, a novel parallel LFSR reseeding method in order to reduce the number of test data was proposed in this paper. It is not only suitable for multiple scan chains design based on STUMPS, but also profitable low test application time. The details of parallel LFSR reseeding with selection register are described in section III and IV.



Test Cube T_1	$Z_0 = X_2 \oplus X_3$ $Z_1 = X_1 \oplus X_2$ $Z_2 = X_0 \oplus X_1$ $Z_3 = X_0 \oplus X_2 \oplus X_3$ $Z_4 = X_1 \oplus X_3$ $Z_5 = X_0 \oplus X_2$
Test Cube T_2	$Z_6 = X_1 \oplus X_2 \oplus X_3 \oplus X_4$ $Z_7 = X_0 \oplus X_1 \oplus X_2 \oplus X_5$ $Z_8 = X_0 \oplus X_1 \oplus X_2 \oplus X_3$ $Z_9 = X_0 \oplus X_1 \oplus X_3 \oplus X_4$ $Z_{10} = X_0 \oplus X_3 \oplus X_4 \oplus X_5$ $Z_{11} = X_3 \oplus X_5$
Test Cube T_3	$Z_{12} = X_2 \oplus X_5 \oplus X_6$ $Z_{13} = X_1 \oplus X_6 \oplus X_7$ $Z_{14} = X_0 \oplus X_6$ $Z_{15} = X_2 \oplus X_3 \oplus X_4 \oplus X_5 \oplus X_6$ $Z_{16} = X_1 \oplus X_2 \oplus X_4 \oplus X_5 \oplus X_6 \oplus X_7$ $Z_{17} = X_0 \oplus X_1 \oplus X_4 \oplus X_5 \oplus X_7$

Fig. 1 Partial Reseeding [20]

III. OVERVIEW OF PROPOSED METHOD

The original parallel reseeding was proposed in [29]. The selection register has been added to the original parallel reseeding in order to further reduce test data.

Original parallel reseeding: The architecture of the original scheme is illustrated in Fig. 2. The state of signal C is determined by BIST controller Extra 2-input AND gates are added for switching between Pseudo-Random Pattern Generator (PRPG) and Deterministic Test Pattern Generator (DTPG). One input of AND gates from BIST controller is a control signal, *C*, which is used to control between enable mode and disable mode seed injection. For enable mode, seeds are fed into LFSR through phase shifter to scan chains. For disable mode, data of scan cells are received from combination of LFSR through phase shifter. Extra XOR gates are added in LFSR for feeding seeds in DTPG mode. Position of tap seeds depends on the tap of phase shifter. The selection tap of phase shifter is described in

[30]. Variable N is the number of scan chains. Seed length, L, is proportional to the average number of specified bits per test cube, S_{avg}. In the worst case, L equals to scan length, M, which means the test cubes are not able to be encoded into a smaller data. Variable P is the number of patterns.

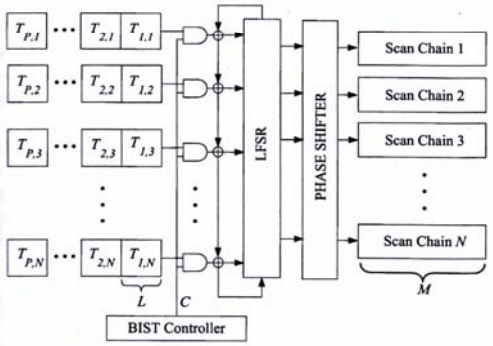


Fig. 2 Original parallel LFSR reseeding [29]

Parallel reseeding with selection register: The idea of selection register was described in [31] providing high probability of error coverage by X's canceling in Multiple Input Shift Register. We apply this idea to the parallel LFSR reseeding to improve the efficiency of encoded test data. The architecture of parallel LFSR reseeding with selection register is shown in Fig. 3. Main parts of the BIST controller are both N-bits selection register and two counters. The counter 1 and counter 2 are used to control signal *C1* and signal *C2*, respectively. Signal *C1* is used to determine the testing mode between PRPG and DTPG. PRPG is activated when signal *C1* is logic "1" but DTPG is activated when *C1* is logic "0". Signal *C2* is used to control groups of seed feeding into LFSR. There are 2 groups of seeds: group 1 (*G1*) is associated with test cubes containing a lot of numbers of specified bits and group 2 (*G2*) is related to the rest of test cubes.

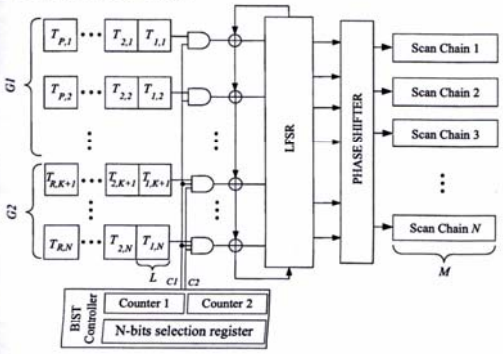


Fig. 3 Parallel LFSR reseeding with selection register

The combination 3-input AND gates is used to enable and disable seeds in *G2*. The subset of extra XOR gates is used to support feeding mechanism in DTPG mode. Seed length, L, is 1 in the best case and equals to M in the worst case. The signal *C2* is forced to "1" for full feeding ($T_{1,1}$, $T_{1,2}$, … $T_{1,N}$) to LFSR, otherwise *C2* is set to "0" for partial feeding ($T_{1,1}$, $T_{1,2}$, $T_{1,3}$, … $T_{1,K}$). The number of seeds in *G2*, R, is a number of seeds for encoding test cubes when R is less than P. The number of encoding bits is calculated by $K*L*P+(N-K)*L*R$.

IV. Forming and Solving Linear Equations

In this work, a new design for injection seeds to LFSR and phase shifter in order to target the r.p.r. faults is proposed in a parallel LFSR reseeding.

This section describes how to form and solve the equations for selection register and how to calculate group size of seeds. The number of encoding bits to store on tester is calculated by *K*L*P+(N-K)*L*R*.

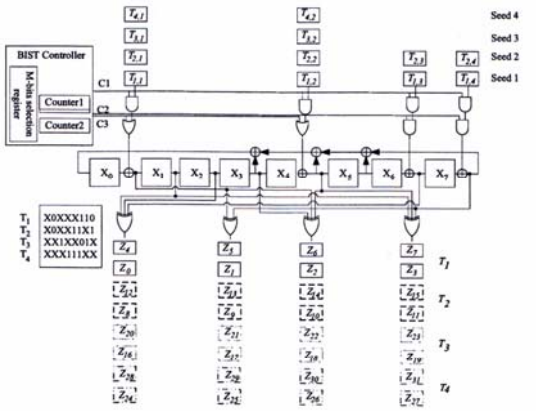


Fig. 4 Example of forming equations for parallel reseeding with selection register

The example of forming equations for parallel reseeding with selection register is illustrated in Fig. 4. The 8-bit flip-flop of LFSR is used to generate four test cubes (T_1, T_2, T_3 and T_4). With the seed condition of L=1, K=2, N=4, P=4 and R=2, gives the number of encoding bits of 12. And the four test cubes are encoded with twelve bits of data represented from *seed1* through *seed4*. The linear equations of each scan cells are described by Z_0, Z_1, Z_2, Z_3, ..., Z_{31}. These linear equations are calculated by the Boolean matrix of tap seed, *matrix(T)* of the LFSR, *matrix(PS)* of phase shifter and variables of seeds, *seed1* to *seed4*. Forming linear equations is defined by two modes: enable and disable mode. Seeds are loaded to LFSR circuit in enable mode.

To describe the mechanism of LFSR with polynomial term of $X^7 \oplus X^5 \oplus X^4 \oplus X^3+1$ in Fig. 4, the Boolean matrix, *matrix(T)*, shown in Fig. 5 (a) is used. Values of each column in *matrix(T)* are associated with variables from X_0 to X_7. The Boolean matrix of phase shifter, *matrix(PS)*, is shown in Fig. 5 (b), where each column of matrix shows the result of tap $X_6 \oplus X_2 \oplus X_1$, $X_7 \oplus X_0$, $X_4 \oplus X_3 \oplus X_0$ and $X_7 \oplus X_4 \oplus X_1$, respectively.

$$T = \begin{bmatrix} 00000001 \\ 10000000 \\ 01000001 \\ 00100001 \\ 00010001 \\ 00001000 \\ 00000100 \\ 00000010 \end{bmatrix} \qquad PS = \begin{bmatrix} 0101 \\ 1000 \\ 0000 \\ 0011 \\ 0010 \\ 1000 \\ 1001 \\ 0110 \end{bmatrix}$$

(a) (b)

Fig. 5 (a) Boolean matrix of LFSR (b) Boolean matrix of phase shifter

Fig. 6 (a) shows the Boolean matrix of seed variables, *seed1*, *seed2*, *seed3*, and *seed4*. *seed1* and *seed2* are 4-bit seed which is described by variables from $T_{1,1}$ to $T_{1,4}$ and $T_{2,1}$ to $T_{2,4}$, respectively. *seed3* and *seed4* are 2-bit seed from $T_{3,1}$ to $T_{3,2}$ and $T_{4,1}$ to $T_{4,2}$. Tap position for seed feeding is shown in Fig. 6 (b). The number "1" in each row is represented the position for seed feeding that is X_6, X_0, X_4 and X_7. Tap positions of X_6, X_0, X_4 and X_7 are depended on taps of phase shifter such as X_6 is depended on $X_6 \oplus X_2 \oplus X_1$.

$$seed1 = [T_{1,4} \;\; T_{1,3} \;\; T_{1,2} \;\; T_{1,1}]$$
$$seed2 = [T_{2,4} \;\; T_{2,3} \;\; T_{2,2} \;\; T_{2,1}]$$
$$seed3 = [0 \;\; 0 \;\; T_{3,2} \;\; T_{3,1}]$$
$$seed4 = [0 \;\; 0 \;\; T_{4,2} \;\; T_{4,1}]$$

(a)

$$tab\,seed = \begin{bmatrix} 01000000 \\ 00000001 \\ 00010000 \\ 10000000 \end{bmatrix}$$

(b)

Fig. 6 (a) Matrix of seed variables (b) Boolean matrix of tap position for seed feeding

For enable mode, equations are from the dot product of the matrix of LFSR, seed variables, tap position for seed and last state of output matrix, O(t-1); Z_n = [*seed* * *tap seed* * O(t-1) * *PS*]. For disable mode, equations are computed by the operation of matrix of LFSR, phase shifter and last state of output matrix; Z_n = [O(t-1)* *T* * *PS*].

The X_0, X_1, X_2, X_3, X_4, X_5, X_6 and X_7 are actually constant values of last cycle in PRPG mode, which do not effect to the determination of solution in the first state output Boolean output matrix, O(1). The LFSR matrix is not applied in O(1) matrix because the set of seed is added in between LFSR flip-flop. So, the O(1) is generated between *seed1* and Boolean output matrix of tap seed. The equation for describing scan cells Z_3 to Z_0 is from the dot product of O(1) and *PS*. The second test clock cycle of output Boolean matrix calculation, O(2) is computed by the operation between the first clock cycle of O(1) and *T*. The equations of Z_7 to Z_4 are computed by the operation between the O(2) and *PS*. The complete matrix is achieved by running 8 cycles as shown in Fig. 7.

test clock 1	$O(1) = seed1 * tab\,seed$	$Z_3 - Z_0 = O(1) * PS$
test clock 2	$O(2) = O(1) * T$	$Z_7 - Z_4 = O(2) * PS$
test clock 3	$O(3) = seed2 * tab\,seed * O(2)$	$Z_{11} - Z_8 = O(3) * PS$
test clock 4	$O(4) = O(3) * T$	$Z_{15} - Z_{12} = O(4) * PS$
⋮	⋮	⋮
test clock 8	$O(8) = O(7) * T$	$Z_{31} - Z_{27} = O(8) * PS$

Fig. 7 The linear equation of each scan cells

The linear equations shown in Fig. 8 are solved by using Gauss-Jordan elimination technique. Only specified bits of test cubes are considered to be solved. One example of solution is shown in Fig. 9.

Test Cube 1	Test Cube 2	Test Cube 3	Test Cube 4
$Z_0 = T_{1,3}$	$Z_8 = T_{1,1} \oplus T_{1,4} \oplus T_{2,3}$	$Z_{16} = T_{1,2} \oplus T_{1,3} \oplus T_{2,1} \oplus T_{2,4}$	$Z_{24} = T_{1,1} \oplus T_{1,4} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{3,1}$
$Z_1 = T_{1,1} \oplus T_{1,4}$	$Z_9 = T_{1,2} \oplus T_{1,3} \oplus T_{2,1} \oplus T_{2,4}$	$Z_{17} = T_{1,1} \oplus T_{1,2} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{3,1}$	$Z_{25} = T_{1,1} \oplus T_{1,3} \oplus T_{1,4} \oplus T_{2,1} \oplus T_{2,2} \oplus T_{3,2} \oplus T_{4,1}$
$Z_2 = T_{1,1} \oplus T_{1,2}$	$Z_{10} = T_{1,2} \oplus T_{1,3} \oplus T_{2,1} \oplus T_{2,2}$	$Z_{18} = T_{1,4} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{3,1} \oplus T_{3,2}$	$Z_{26} = T_{1,1} \oplus T_{1,2} \oplus T_{1,4} \oplus T_{2,4} \oplus T_{3,2} \oplus T_{4,1} \oplus T_{4,2}$
$Z_3 = T_{1,2} \oplus T_{1,4}$	$Z_{11} = T_{1,2} \oplus T_{1,4} \oplus T_{2,2} \oplus T_{2,4}$	$Z_{19} = T_{1,1} \oplus T_{2,2} \oplus T_{2,4} \oplus T_{3,2}$	$Z_{27} = T_{1,1} \oplus T_{1,3} \oplus T_{1,4} \oplus T_{2,1} \oplus T_{3,2} \oplus T_{4,2}$
$Z_4 = T_{1,1}$	$Z_{12} = T_{1,3} \oplus T_{1,4} \oplus T_{2,1}$	$Z_{20} = T_{1,1} \oplus T_{1,2} \oplus T_{2,3} \oplus T_{2,4} \oplus T_{3,1}$	$Z_{28} = T_{1,3} \oplus T_{1,4} \oplus T_{2,1} \oplus T_{2,2} \oplus T_{4,1}$
$Z_5 = T_{1,2} \oplus T_{1,3} \oplus T_{1,4}$	$Z_{13} = T_{1,2} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{2,4}$	$Z_{21} = T_{1,1} \oplus T_{1,2} \oplus T_{1,4} \oplus T_{2,2} \oplus T_{3,2}$	$Z_{29} = T_{1,1} \oplus T_{1,3} \oplus T_{1,4} \oplus T_{2,1} \oplus T_{2,2} \oplus T_{2,4} \oplus T_{3,2} \oplus T_{4,2}$
$Z_6 = T_{1,2} \oplus T_{1,4}$	$Z_{14} = T_{1,1} \oplus T_{2,2} \oplus T_{2,4}$	$Z_{22} = T_{1,1} \oplus T_{1,2} \oplus T_{1,3} \oplus T_{2,1} \oplus T_{3,2}$	$Z_{30} = T_{1,1} \oplus T_{1,2} \oplus T_{1,3} \oplus T_{1,4} \oplus T_{2,1} \oplus T_{2,2} \oplus T_{2,3} \oplus T_{3,1} \oplus T_{4,2}$
$Z_7 = T_{1,1} \oplus T_{1,3}$	$Z_{15} = T_{1,3} \oplus T_{2,1} \oplus T_{2,3}$	$Z_{23} = T_{1,1} \oplus T_{1,4} \oplus T_{2,3} \oplus T_{3,1}$	$Z_{31} = T_{1,3} \oplus T_{1,4} \oplus T_{2,1} \oplus T_{2,4} \oplus T_{4,1}$

Fig. 8 The summary of equation for describing scan cells Z_0-Z_{31}

$$T_{1,1} = 1 \quad T_{1,4} = 0 \quad T_{2,3} = 0 \quad T_{3,2} = 0$$
$$T_{1,2} = 0 \quad T_{2,1} = 1 \quad T_{2,4} = 1 \quad T_{4,1} = 0$$
$$T_{1,3} = 0 \quad T_{2,2} = 0 \quad T_{3,1} = 0 \quad T_{4,2} = 1$$

Fig. 9 One of solution of equation.

V. EXPERIMENTAL RESULTS

The experimental results of parallel reseeding were performed with ISCAS 89 benchmark circuits. The information details of ISCAS 89 benchmark circuits are shown in Table 1. First column lists the circuits used in the experiment. The second column shows the number of scan cells. The third column lists the number of test cubes. The last column gives the LFSR size. The LFSR size is relied on last published of partial LFSR reseeding in [20-21].

Table 1. ISCAS 89 benchmark circuit

Circuit Name	Scan Cells	Test Cubes	LFSR Size
S5378	214	30	38
S9234	247	138	81
S13207	700	157	44
S15850	611	167	58
S38417	1,664	340	105
S38584	1,464	62	75

Table 2. shows the experimental results of parallel LFSR reseeding. The first column gives the circuit name. The second column shows total specific bits. In the result of original parallel reseeding [29], bits per seed are shown in the third column and total encoding bits are shown in the fourth column. The results of parallel reseeding with selection register give in fifth, sixth, and seventh column. For the fifth column, *G1* is the total number of bits in the first group. In the sixth column, *G2* is the total number of bits in the second group. The seventh column shows the total number of seeds. The parallel LFSR reseeding with selection register achieves higher efficiency for encoding than the original design.

Table 2. Experimental result of propose method

Circuit Name	Total Specific bits	Original Parallel LFSR Reseeding [29]		Proposed Parallel LFSR with Selection Register		
		Bits per Seed	Total Bits	Bits per Seeds		Total Bits
				G1	*G2*	
S5378	493	17	510	11	6	**492**
S9234	4,674	59	8,142	27	32	**5,742**
S13207	2,824	24	3,768	14	10	**2,898**
S15850	5,092	34	5,678	27	7	**5,181**
S38417	23,984	83	28,220	68	15	**25,325**
S38584	2,848	54	3,348	45	9	**2,889**

Table 3. shows comparison of three schemes of previous LFSR reseeding: original LFSR reseeding [22], partial reseeding [20], and parallel LFSR reseeding with selection register. From three schemes, the parallel LFSR reseeding with selection register achieves highest efficiency than others in most of circuits except S9234 and S38417.

Table 3. Encoded test data comparison

Circuit Name	Original LFSR Reseeding [22]	Partial LFSR Reseeding [20]	Proposed Parallel LFSR with Selection Register
S5378	1,140	502	**492**
S9234	11,178	**5,013**	5,742
S13207	6,908	3,008	**2,898**
S15850	9,686	5,204	**5,181**
S38417	35,700	**24,513**	25,325
S38584	4,650	2,942	**2,889**

The comparisons of test clock cycles are shown in Table 4. First column gives circuit name. Second column shows the number of test clock cycles in LFSR reseeding with seed ordering [28]. In the last column shows the total test clock cycles of proposed method. Our approach provides lower test clock cycles than LFSR reseeding with seed ordering in all circuits, but S13207.

Table 4. Test clock cycles comparison

Circuit Name	Number of Test Clock Cycles	
	LFSR reseeding with Seed ordering [28]	Proposed Parallel LFSR Reseeding with Selection Register
S5378	1920	**390**
S9234	4992	**690**
S13207	**2688**	4710
S15850	3904	**3006**
S38417	N/A	**7140**
S38584	2624	**1736**

VI. CONCLUSION

The experimental results for ISCAS 89 benchmark circuits indicate the significant improvement in terms of test data. Moreover, there are several advantages of our proposed technique:

- 100% test coverage
- High fault coverage as intended by the deterministic test
- Low test data
- Suitable for multi scan design or STUMPS
- Low structural dependency
- Low test application time
- High efficiency for encoding

Therefore, the design of parallel reseeding with selection register is an attractive approach for encoding test data in order to apply for mixed-mode BIST.

REFERENCES

[1] C. E. Stroud, "A designer's Guide to Built-In Self-Test," Massachusetts: *Kluwer Academic publishers*, 2002.

[2] G. Dimitris, "Advances in electronic testing: challenges and methodologies," *Dordrecht:Springer US*, 2006.

[3] T.Reungpeerakul, D. Kay, S. Mourad, "Partial-Matching Technique in Mixed-Mode BIST Environment," *IEEE Transactions on Instrumentation and Measurement*, pp. 970-977, 2010.

[4] T. Hayashi, N. Hiraiwa, T. Shinogi, H. Takase, H, Kita, "Test Modification and Compression Technique for Reducing Total Test Volume with Dictionary Data," *ASIC International Conference*, pp. 639-644, 2005.

[5] K.J. Balakrishnan, F. Lei, "RTL Test Point Insertion to Reduce Delay Test Volume," *VLSI Test Symposium*, pp. 325-332, 2007.

[6] W. Seongmoon, W. Wenlong, "Cost Efficient Methods to Improve Performance of Broadcast Scan," *Asia Test Symposium*, pp. 163-169, 2008.

[7] U. Snehal, K. Dimitri, "Minimizing Observation Points for Fault Location," *IEEE International Symposium on Defect and Fault Tolerance in VLSI System*, pp. 263-267, 2009.

[8] A. Jas, C. V. Krishna, and N. A. Touba, "Weighted Pseudorandom Hybrid BIST," *IEEE Trans. VLSI Systems*, pp. 1277-1283, 2004.

[9] Y. Chaowen, S.M. Reddy, I. Pomeranz, "Weighted Pseudo-Random BIST for N-detection of Single Stuck-at Faults," *Asian Test Symposium*, pp. 178-183, 2004.

[10] Y. Chaowen, S.M. Reddy, I. Pomeranz, "Circuit independent Weighted Pseudo-Random BIST Pattern Generator," *Asian Test Symposium*, pp. 132-137, 2005.

[11] P. Fiser "Pseudo-Random Pattern Generator Design for column-matching BIST," *Euromicro Conference*, pp. 657 – 663, 2007.

[12] D. Das, R. Kumar, P. P. CHakrabarti, "Dictionary based code compression for variable length instruction encodings", *International Conference on VLSI Design*, pp. 545-550, 2005.

[13] K. Basu, P. Mishra, "Test data Compression Using Efficient Bitmask and Dictionary Selection Methods," *IEEE VLSI Systems*, 2009.

[14] A. W. Hakmi, S. Holst, H. J. Wunderlisch, J. Schloffel, F. Hapke, A. Glowatz, "Restrict Encoding for Mixed-Mode BIST," *IEEE VLSI Test Symposium*, pp.179-184, 2009.

[15] U.S. Mehta, K.S. Dasgupta, N.M. Devashrayee, "Hamming Distance Based Reordering and Columnwise Bit Stuffing with Difference Vector: A Better Scheme for Test data compression with Run Length Based Codes," *VLST design*, pp. 33-38, 2010.

[16] N.A. Touba, E.J. McCluskey, "Bit-fixing in pseudorandom sequences for scan BIST," *IEEE Transaction on Computer-Aided Design of Integrated Circuit and Systems*, pp. 545-555, 2001.

[17] L. Wei, Y. Chaowen, S.M. Reddy, I. Pomeranz, "A scan BIST generation method using a Markov source and partial bit-fixing," *Proceedings of Design Automation Conference*, pp. 554-559, 2003.

[18] A. Gabizon, R. Raz, R. Shaltiel, "Deterministic extractors for bit-fixing source by obtaining an independent seed," *IEEE Symposium on Foundations of Computer Science*, pp. 394-403, 2004.

[19] H.-J., Wunderlich, G. Kiefer, "Bit-Flipping BIST," *IEEE Computer-Aided Design*, pp. 337-343, 1996.

[20] C.V. Krishna, A. Jas, and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," *Proc. of IEEE International Test Conference*, pp. 885-893, 2001.

[21] C.V. Krishna and N.A. Touba, "Hybrid BIST Using an Incrementally Guided LFSR," *Proc. of IEEE Symposium on Defect and Fault Tolerance*, pp. 217-224, 2003.

[22] B. Konemann, "LFSR-Coded Test Patterns for Scan Designs," *Proceeding of European Test Conference*, pp. 237-242, 1991.

[23] S. Hellebrand, J. Rasjki, S. Venkataraman and B. Countois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *Proc. of International Test Conference*, pp. 120-129, 1992.

[24] J. Rajski, J. Tyszer, N. Zacharia, "Test Data Decompression for Multiple Scan Designs with Boundary Scan," *IEEE Transactions on Computers*, pp. 1188-1200, 1998.

[25] P. Wohl, J.A. Waicukauski, S. Patel, F. DaSilva, T.W. Williams, R. Kapur, "Efficient Compression of Deterministic Patterns into Multiple PRPG Seeds," *Proceeding of International Test Conference*, pp. 916-925, 2005.

[26] K. Hong-Sik, K. Sungho, "Increasing encoding efficiency of LFSR reseeding-base test compression," *IEEE Trans. on Computer-Aided Design of Integrated Circuit and Systems*, pp. 913-917, 2006.

[27] K.J. Balakrishnan, W. Seongmoon, S.T. Chakradhar, "PIDISC: pattern independent design independent seed compression technique," *International Conference on VLSI design*, 2006.

[28] A.A Al-Yamani, S. Mitra, E.J. McCluskey, "Optimized Reseeding by Seed Ordering and Encoding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 264-270, 2005.

[29] P. Kongtim, T. Reungpeerakul, "Parallel LFSR reseeding for Mixed-mode BIST," *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pp. 198-202, 2010.

[30] D. Kagaris, "A unified method for phase shifter computation," *ACM Transactions on Design Automation of Electronic Systems*, pp. 157–167, 2005.

[31] N.A. Touba, "X-Canceling MISR-An X Tolerant Methodology for Compacting Output Responses with Unknowns Using a MISR," *International Test Conference*, pp. 1-10, 2007.

ภาคผนวก ค

ตารางสรุปสมการโพลิโนเมียลแบบดั้งเดิมที่มีค่า n มีค่าระหว่าง 2-411

k / n	3	5			7				
2	1								
3	1								
4	1								
5	2	1	2	3					
6	1	1	4	5					
7	1	2	3	4	1	2	3	4	5
8		1	2	7	2	4	5	6	7
9	4	3	5	6	2	3	6	7	8
10	3	2	3	8	1	2	5	6	7
11	2	1	8	10	1	2	5	7	9
12		1	2	10	2	6	8	9	10
13		3	5	8	1	2	5	10	12
14		1	11	12	1	3	4	5	11
15	1	3	4	12	5	6	7	8	13
16		10	12	15	2	9	12	13	14
17	3	4	12	16	2	5	6	8	13
18	7	4	11	16	1	4	7	8	10
19		3	9	10	6	12	13	16	18
20	3	2	7	13	1	10	14	16	18
21	2	3	4	9	6	8	14	18	19
22	1	3	7	12	2	4	9	14	21
23	5	4	8	15	5	11	12	13	17
24		2	5	11	3	6	7	16	23
25	3	7	12	13	7	10	13	15	23
26		13	15	23	1	6	15	17	24
27		17	22	23	6	11	17	18	19
28	3	5	8	24	5	11	21	24	27
29	2	2	6	16	3	11	15	16	22
30		9	10	27	11	12	24	28	29
31	3	8	23	25	1	8	10	14	16
32		2	7	16	1	3	12	17	30
33	13	11	16	26	1	8	17	19	32
34		8	12	17	4	7	14	20	31
35	2	9	17	27	2	21	23	31	32
36	11	7	12	33	6	17	25	26	28
37		2	14	22	3	21	30	31	33
38		5	6	27	6	9	11	20	36
39	4	16	23	35	2	13	15	36	38
40		23	27	29	6	7	18	28	36
41	3	27	31	32	11	12	20	32	40
42		30	31	34	1	8	14	24	27
43		5	22	27	8	25	30	32	35
44		18	35	39	5	16	25	40	43
45		4	28	39	14	15	23	27	33
46		18	31	40	21	23	24	40	44
47	5	11	24	32	5	17	19	32	42
48		1	9	19	5	12	27	29	43
49	9	16	18	24	8	39	41	42	45
50		17	31	34	5	6	16	21	36
51		15	24	46	12	15	22	24	25

k / n	3	5			7				
52	3	17	18	22	1	2	16	25	50
53		20	41	50	4	18	29	37	51
54		29	49	53	9	10	23	24	34
55	24	19	38	50	16	23	44	45	51
56		29	39	41	5	20	28	38	45
57	7	1	16	42	4	5	31	40	50
58	19	4	37	52	23	32	37	54	55
59		26	46	54	21	22	34	45	53
60	1	27	28	34	12	13	19	31	48
61		15	19	44	33	38	47	52	59
62		3	26	57	2	9	16	18	48
63	1	20	44	54	5	8	18	22	60
64		9	34	61	23	28	31	56	61
65	18	10	18	38	8	10	15	43	60
66		39	48	55	4	7	8	23	50
67		3	33	61	25	26	28	44	64
68	9	29	47	62	14	29	39	41	63
69		20	27	63	21	22	39	44	50
70		3	57	69	30	34	43	58	63
71	6	48	53	59	21	30	34	45	49
72		2	14	23	6	10	11	14	22
73	25	11	50	58	2	12	35	48	66
74		7	43	68	4	17	23	28	69
75		14	18	33	2	21	29	60	72
76		14	29	52	1	17	27	28	34
77		2	36	52	13	25	62	68	74
78		16	20	47	5	29	40	53	73
79	9	24	28	44	28	33	39	56	57
80		17	27	75	10	37	50	51	70
81	4	9	34	43	1	27	28	48	63
82		27	41	68	43	44	53	66	79
83		16	33	55	25	27	42	47	67
84	13	45	51	59	15	30	49	62	82
85		11	36	50	17	22	27	44	78
86		7	10	80	32	47	56	65	78
87	13	21	53	56	24	52	65	68	85
88		15	53	86	33	46	51	54	86
89	38	34	67	77	18	21	31	68	81
90		10	58	71	45	62	64	74	82
91		29	31	50	1	44	58	78	83
92		13	24	32	42	47	65	74	76
93	2	67	77	88	12	66	73	80	83
94	21	18	29	80	2	14	18	28	43
95	11	11	77	83	5	17	40	90	92
96		15	17	81	4	10	11	14	57
97	6	17	44	93	5	6	28	53	82
98	11	26	85	87	5	34	35	41	75
99		11	38	68	4	9	28	43	84
100	37	36	60	81	16	22	34	77	83
101		26	74	83	33	45	57	86	92

k / n	3	5			7				
102		15	19	27	38	50	52	65	88
103	9	60	80	83	22	35	43	67	69
104		6	49	89	33	43	80	81	102
105	16	70	87	96	7	15	21	40	101
106	15	19	86	96	12	17	34	78	86
107		39	54	59	23	29	40	84	89
108	31	3	24	59	36	43	46	62	68
109		25	58	102	3	69	74	95	100
110		21	55	97	7	17	30	70	72
111	10	5	67	77	19	54	71	101	102
112		2	19	68	63	71	87	109	111
113	9	25	80	96	13	38	48	92	109
114		54	72	103	2	38	62	74	79
115		8	20	30	17	21	47	58	98
116		24	27	95	4	11	12	43	105
117		64	73	74	4	53	70	74	104
118	33	50	106	117	29	37	45	59	109
119	8	36	52	82	20	43	92	111	116
120		9	46	88	70	71	77	82	87
121	18	33	42	43	8	25	105	115	116
122		35	39	54	93	98	100	109	119
123	2	23	51	113	4	14	18	21	121
124	37	15	31	43	48	60	72	74	107
125		65	90	103	9	24	39	57	108
126		10	70	117	51	64	70	78	81
127	1	13	45	54	31	38	67	68	97
128		11	35	77	36	38	45	57	95
129	5	2	5	10	41	43	100	110	114
130	3	19	70	97	20	46	84	110	123
131		17	28	85	32	85	87	89	104
132	29	22	43	70	5	9	83	91	93
133		28	44	50	14	21	69	101	120
134	57	34	40	71	12	18	25	69	74
135	11	10	93	109	13	17	80	88	134
136		109	114	134	9	18	39	67	106
137	21	42	56	98	1	24	44	51	99
138		26	47	103	19	24	105	109	117
139		23	60	85	35	77	91	112	118
140	29	63	97	112	3	6	39	42	69
141		7	67	125	57	64	68	81	115
142	21	67	96	137	80	85	90	104	118
143		110	118	142	10	13	17	112	136
144		54	65	129	51	53	56	66	71
145	52	23	133	138	19	55	111	124	139
146		78	101	115	22	38	46	105	116
147		43	89	110	50	118	122	141	142
148	27	27	57	124	55	98	121	129	145
149		27	60	132	33	34	53	71	148
150	53	17	62	136	69	83	87	89	94
151	3	25	27	117	7	11	33	53	64

k / n	3	5			7				
152		35	120	145	36	89	90	101	143
153	1	12	72	137	8	40	54	74	91
154		119	128	151	35	51	96	102	122
155		77	129	152	19	29	42	116	151
156		10	50	143	46	52	63	65	116
157		42	47	110	27	69	79	84	85
158		19	35	151	48	52	75	107	108
159	31	7	20	100	87	92	98	107	137
160		30	56	101	28	58	87	88	136
161	18	25	109	134	33	35	52	67	69
162		123	127	150	2	14	116	133	155
163		64	115	133	4	40	59	88	153
164		8	111	140	8	48	72	75	117
165		12	75	137	32	55	70	110	152
166		26	77	157	8	18	32	93	118
167	6	4	9	103	51	72	84	102	125
168		29	32	127	13	21	102	104	106
169	34	29	100	131	21	65	93	103	129
170	23	92	105	145	29	44	54	98	121
171		70	106	114	25	105	109	142	150
172	7	22	27	95	80	97	103	136	156
173		10	13	123	32	44	102	151	169
174	13	41	56	78	12	30	46	67	90
175	6	37	146	173	57	85	90	135	143
176		57	119	129	35	103	105	128	137
177	8	122	151	170	14	24	50	72	170
178	87	34	159	160	84	87	88	117	165
179		39	129	152	26	53	123	154	157
180		14	98	149	68	73	148	155	178
181		63	133	164	9	22	38	47	58
182		59	111	155	26	48	115	120	175
183	56	11	73	148	19	24	96	113	181
184		11	148	174	1	81	109	152	182
185	24	9	33	120	14	39	121	130	134
186		62	63	146	47	52	65	124	128
187		17	65	88	56	100	105	160	178
188		81	87	170	67	69	113	141	142
189		86	120	171	36	45	65	147	180
190		109	145	187	32	58	125	159	163
191	9	3	78	188	30	66	99	119	166
192		17	103	142	59	94	113	143	181
193	15	19	39	61	22	65	113	159	173
194	87	51	56	182	20	21	47	64	161
195		28	41	68	84	105	106	108	154
196		69	152	191	28	65	72	133	148
197		11	44	114	17	26	77	79	124
198	65	97	144	154	82	103	107	108	143
199	34	48	84	106	27	38	44	104	184
200		69	134	135	13	17	57	106	132
201	14	133	164	200	64	119	125	147	156

k / n	3	5			7				
202	55	22	63	83	90	105	117	189	195
203		121	123	167	59	85	124	133	142
204		59	95	108	97	121	140	143	162
205		147	169	197	12	103	124	174	190
206		125	129	155	3	4	88	104	199
207	43	114	126	206	28	65	129	136	167
208		63	77	97	58	64	124	159	201
209	6	78	143	204	50	63	66	98	155
210		38	47	155	13	51	62	110	190
211		52	153	155	51	63	89	114	136
212	105	29	36	176	83	92	127	158	181
213		55	84	112	15	26	64	134	135
214		88	100	133	103	176	189	191	207
215	23	41	96	124	46	74	106	125	141
216		98	103	109	31	96	133	190	207
217	45	31	51	144	12	25	81	87	144
218	11	11	95	128	27	163	165	180	212
219		55	58	143	3	37	134	190	201
220		23	121	168	39	100	134	160	190
221		142	156	211	32	69	114	154	202
222		45	46	106	44	140	157	171	180
223	33	30	64	72	44	57	85	124	169
224		2	39	116	9	132	135	203	217
225	32	57	103	205	72	93	147	178	180
226		107	128	162	65	81	96	108	137
227		11	43	142	65	100	104	189	224
228		20	100	125	44	74	127	181	220
229		4	66	189	17	55	62	112	157
230		195	212	222	24	49	96	170	201
231	26	99	137	224	13	21	118	138	174
232		35	71	169	80	150	155	180	222
233	74	41	149	189	58	65	148	185	230
234	31	37	80	113	19	113	124	146	155
235		22	37	124	20	122	160	189	234
236	5	110	117	224	73	78	86	127	141
237		54	64	211	26	31	89	144	186
238		7	44	155	72	84	93	140	178
239	36	10	56	66	12	61	207	216	226
240		226	235	238	25	31	138	150	160
241	70	28	32	170	26	100	214	217	219
242		83	91	216	29	46	66	143	170
243		97	181	191	51	94	199	203	236
244		157	190	220	18	75	119	127	210
245		193	206	243	17	107	126	137	197
246		25	147	231	55	109	184	214	226
247	82	40	96	214	12	107	151	193	220
248		53	189	199	102	107	152	178	221
249	86	40	116	146	9	65	82	113	163
250	103	28	107	180	127	139	170	175	216
251		61	75	178	110	124	199	235	249

k / n	3	5			7				
252	67	58	67	167	11	48	145	169	236
253		19	50	222	5	27	82	100	158
254		16	131	189	14	41	133	164	186
255	52	4	107	184	50	82	116	153	166
256		121	178	241	12	48	115	133	213
257	12	61	181	195	59	110	151	199	227
258	83	115	119	170	28	46	58	146	167
259		17	40	221	66	134	190	191	223
260		63	211	218	69	86	91	163	179
261		6	37	150	23	61	191	203	223
262		22	117	247	81	123	171	172	182
263	93	30	34	181	110	122	137	145	154
264		76	175	217	59	159	168	206	241
265	42	43	148	243	18	36	89	129	239
266	47	44	133	198	21	24	36	136	146
267		100	150	165	75	80	90	154	250
268	25	23	109	207	17	24	39	69	187
269		116	133	166	49	114	149	164	259
270	53	10	196	205	41	142	198	215	235
271	58	9	161	187	97	109	111	136	231
272		150	197	221	88	115	137	141	150
273	23	96	187	220	9	65	105	130	193
274	67	40	201	237	16	52	149	199	267
275		1	234	250	6	42	106	148	188
276		15	37	61	18	130	145	149	195
277		108	207	216	12	56	89	130	139
278	5	71	153	242	90	163	217	236	247
279	5	90	220	265	150	160	187	228	238
280		175	234	238	19	49	163	246	274
281	93	104	129	134	51	103	105	264	280
282	35	16	80	199	40	122	138	161	270
283		2	82	255	60	130	161	186	234
284	119	114	211	247	29	71	147	230	265
285		73	127	146	129	188	222	255	269
286	69	99	141	189	3	115	152	165	171
287	71	121	155	157	36	70	108	222	259
288		74	101	159	13	127	166	175	285
289	21	176	228	250	14	72	169	197	279
290		69	149	266	11	20	81	146	195
291		218	253	287	48	54	116	228	270
292	97	156	195	255	35	87	143	147	160
293		93	106	205	74	114	205	231	268
294	61	139	159	187	84	186	191	241	244
295	48	98	122	283	65	102	150	182	210
296		10	198	235	31	76	80	195	222
297	5	43	160	292	4	14	19	134	260
298		114	196	251	74	100	167	168	255
299		80	113	149	35	69	133	254	280
300	7	89	122	220	49	107	158	163	295
301		181	209	215	33	196	210	222	277

k / n	3	5			7				
302	41	186	189	281	61	114	182	206	277
303		43	217	274	17	77	119	215	244
304		114	145	198	56	59	74	228	235
305	102	33	63	209	140	161	230	245	300
306		119	133	244	20	52	71	86	254
307		229	237	273	33	62	81	119	306
308		51	163	229	65	126	237	282	286
309		241	286	289	6	22	146	220	300
310		84	171	211	7	113	147	251	262
312		181	238	265	171	186	195	225	283
313	79	103	133	180	66	86	119	187	262
314	15	48	64	251	39	60	116	169	207
315		21	166	259	64	113	145	185	263
316	135	42	232	267	18	20	86	174	265
317		60	227	232	76	139	166	174	227
318		35	98	201	44	101	188	303	315
319	36	44	50	144	36	135	152	233	283
320		169	293	319	9	57	233	280	295
321	31	27	70	198	78	126	149	246	299
322	67	31	234	309	48	213	233	251	321
323		7	32	106	101	202	234	247	313
324		58	169	279	56	155	158	281	321
325		20	178	245	56	71	75	239	322
326		66	107	289	88	225	258	260	301
327	34	100	154	208	9	33	243	244	301
328		10	214	289	119	134	166	213	270
329	50	219	232	301	66	151	173	175	293
330		92	247	292	52	63	195	258	267
331		50	219	298	3	25	76	130	292
332	123	227	258	281	103	120	185	205	263
333	2	40	43	110	13	21	154	255	257
334		287	325	332	132	232	269	296	331
335		193	266	307	130	166	177	213	231
336		193	235	330	2	4	19	149	274
337	55	54	137	229	21	102	112	118	258
338		203	250	303	44	97	120	126	171
339		212	237	246	69	125	219	234	236
340		222	290	317	94	183	194	267	338
341		22	49	179	103	109	234	299	333
342	125	148	152	253	240	273	281	310	316
343	75	28	68	303	29	32	228	305	340
344		29	153	211	57	92	131	145	160
345	22	241	252	279	113	129	161	230	333
346		7	40	274	27	76	138	247	325
347		37	267	334	31	64	162	209	236
348		12	122	161	72	109	123	169	298
350	53	13	238	248	103	184	237	265	278
351	34	100	147	183	49	159	221	283	308
352		134	153	313	152	168	241	285	326
354		156	183	188	11	142	222	231	308

k / n	3	5			7				
355		58	59	80	34	143	185	212	248
356		71	144	303	125	144	215	230	311
357		197	302	354	14	79	181	247	262
358		115	120	283	20	77	80	235	299
359	68	66	201	249	91	99	155	226	296
360		38	171	290	35	61	76	125	197
362	63	9	37	290	41	191	288	324	353
363		183	255	262	44	188	201	219	335
364	67	148	241	349	28	181	233	247	255
365		111	220	253	24	93	138	283	313
366	29	8	183	299	32	188	270	349	357
368		121	293	355	103	124	162	187	247
369	91	41	218	344	33	59	71	204	340
370	139	12	350	359	16	52	173	174	263
371		287	310	343	30	161	293	322	338
372		126	141	248	41	211	212	227	359
373		7	106	147	33	160	201	233	318
374		105	181	266	40	49	96	194	253
375	16	122	200	304	1	29	77	185	242
376		23	24	85	19	99	254	309	365
377	41	7	159	209	53	66	108	155	312
378	43	36	63	352	25	123	149	243	323
379		10	186	303	99	132	173	249	323
380	47	37	70	80	103	151	281	282	376
381		190	259	327	10	71	161	190	351
382	81	88	193	375	6	79	110	192	194
384		23	51	381	4	18	222	274	375
385	6	215	218	352	144	175	222	238	379
386	83	79	175	299	119	128	232	288	346
387		13	106	156	115	190	203	303	365
388		92	97	278	66	138	261	293	314
389		250	326	381	6	12	33	211	284
390	89	5	176	291	150	164	171	247	269
392		207	222	365	44	61	82	144	373
393	7	68	204	254	88	201	251	309	373
394	135	18	142	219	76	129	151	174	234
395		106	154	237	29	83	109	148	177
396	25	83	147	348	105	142	237	277	345
398		90	305	331	15	134	221	264	308
399	86	337	357	375	46	196	226	345	364
400		8	293	295	13	88	118	156	233
402		87	320	336	77	150	217	243	383
403		12	75	298	28	252	301	306	346
404	189	110	114	315	16	116	160	330	337
405		124	272	307	9	15	181	246	369
406	157	25	255	397	84	130	229	292	366
408		127	174	345	129	176	206	235	243
409	87	127	157	207	80	99	148	247	401
410		166	227	372	67	70	107	165	244
411		3	202	228	31	60	190	195	245