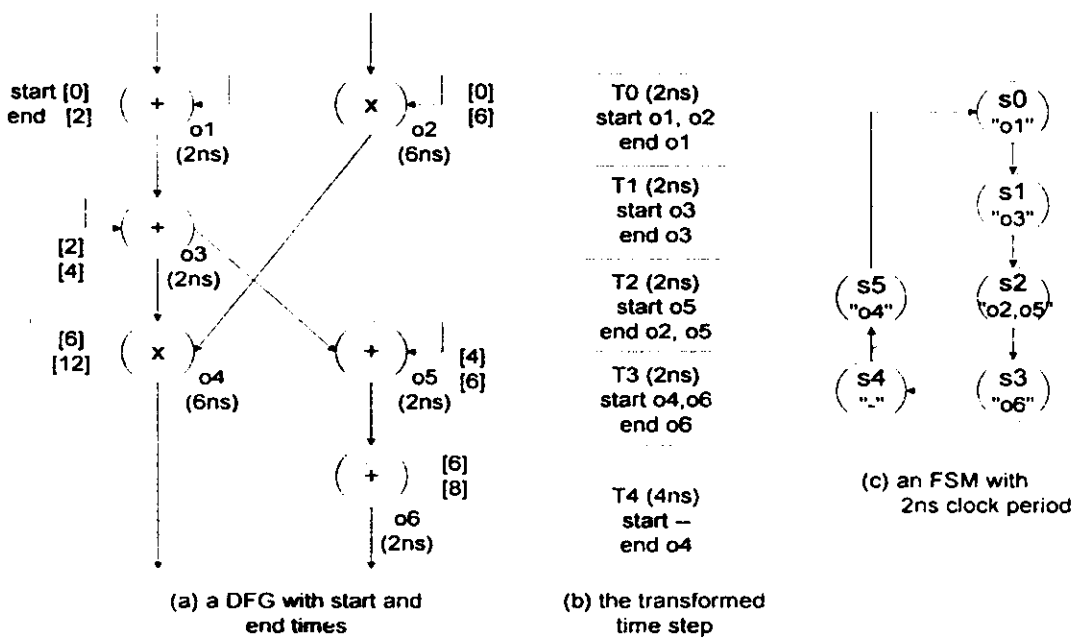
บทที่ 3
ระเบียบวิธีวิจัย

เนื้อหาในบทนี้ อธิบายถึงแนวคิดเบื้องต้น ทฤษฎีที่นำมาประยุกต์ใช้ สถาปัตยกรรมของวงจร เทคนิคการตัดสินใจจุดออกแบบที่เหมาะสม อัลกอริทึมในการสังเคราะห์ FSMs จาก DFGs

3.1 แนวคิด

โครงการวิจัยนี้มีจุดประสงค์เพื่อแก้ปัญหาข้อจำกัดของวงจรซิงโครนัสที่ได้อธิบายใน บทที่ 2 ความเร็วของวงจรที่ได้ต้องไม่ถูกจำกัดโดยเส้นทางเดินของสัญญาณที่ยาวที่สุด โครงการวิจัยนี้จะนำเสนอวิธีการสังเคราะห์ FSMs ที่ไม่ถูกจำกัดโดยเส้นทางเดินของสัญญาณที่ยาวที่สุดจาก DFG วิธีการสังเคราะห์จะเป็นลำดับขั้นตอน (systematic method) ซึ่งสามารถนำไปพัฒนาเป็นซอฟท์แวร์ช่วยออกแบบได้ FSMs ที่ได้จะต้องมีประสิทธิภาพดีที่สุด กล่าวคือ มีจำนวนสเตทน้อย และใช้คาบสัญญาณนาฬิกาที่เหมาะสม เพื่อที่จะให้การสังเคราะห์วงจรทำได้ง่ายและวงจรควบคุมที่ได้มีขนาดเล็ก และทำงานเสร็จภายในเวลาอันรวดเร็ว (มีสแลคในแต่ละโอเปอเรชันน้อย) แนวความคิดพื้นฐานแสดงดังรูปที่ 3.1



รูปที่ 3.1: The transformed DFG and its corresponding time step and FSM.

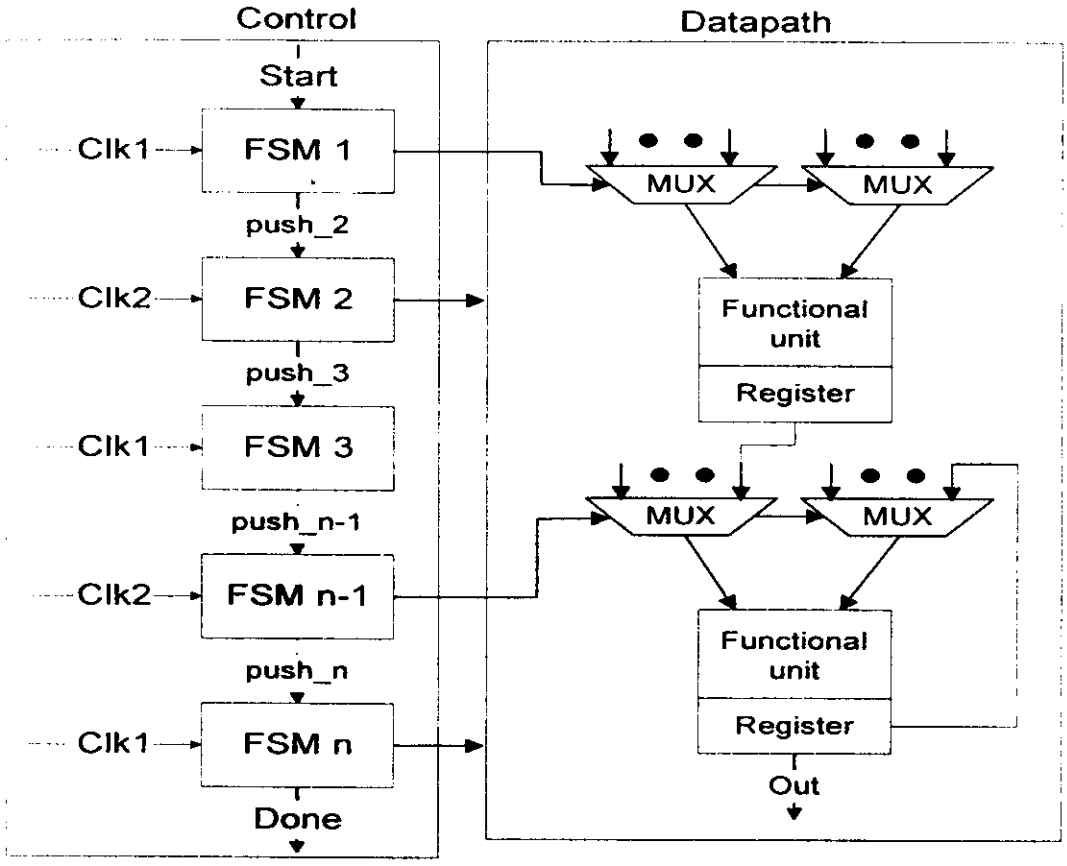
เนื่องจากในการออกแบบวงจรใดๆ ผู้ออกแบบมีข้อมูลว่าแต่ละโอเปอเรชันมีเวลาดำเนินการเป็นเท่าไร ดังนั้นเวลาที่แต่ละโอเปอเรชันเริ่มต้นดำเนินการ (start time) และเวลาที่จบการดำเนินการ (end time) ย่อมสามารถถูกคำนวณได้ รูปที่ 3.1(a) แสดงเวลาเริ่มต้นและเวลาจบการดำเนินการของแต่ละโอเปอเรชันซึ่งได้จากการคำนวณที่ไม่คำนึงถึงขั้นเวลา กล่าวคือเวลาเริ่มต้นและเวลาจบที่สามารถเริ่มต้นและจบได้เร็วที่สุดเท่าที่จะเป็นไปได้ถูกกำหนดให้แต่ละโอเปอเรชัน จากรูปที่ 3(a) ขั้นเวลาอันใหม่ถูกกำหนดได้ดังรูปที่ 3.1(b) และ FSM สำหรับขั้นเวลาอันใหม่นี้ถูกสร้างขึ้นมาใหม่ได้ดังรูปที่ 3.1(c) สังเกตได้ว่า FSM นี้มีเพียง 6 สเตท และมี cycle time เพียง 12ns เท่านั้น นอกจากนี้จะเห็นว่าโอเปอเรชันหนึ่ง ๆ ไม่จำเป็นต้องเริ่มดำนินการและจบดำเนินการภายในขั้นเวลาอันเดียวกัน ตัวอย่างเช่น โอเปอเรชัน o2 เริ่มดำเนินการที่ขั้นเวลา T0 แต่ไปจบการดำเนินการที่ ขั้นเวลา T2 และยิ่งไปกว่านี้จะสังเกตเห็นว่าโอเปอเรชันที่ตามหลัง (successor) สามารถเริ่มต้นได้ทันทีทันใดที่โอเปอเรชันที่นำหน้า (predecessor) จบการดำเนินการ ด้วยคุณสมบัติที่กล่าวมานี้เอง FSM ที่ได้จากวิธีการที่เสนอโดยโครงงานนี้ควรจะทำงานได้เร็วโดยไม่ถูกจำกัดโดยเส้นทางเดินของสัญญาณที่ยาวที่สุด

แนวคิดในโครงงานวิจัยนี้แตกต่างจากระเบียบวิธีอื่นที่มีอยู่อย่างสิ้นเชิง ระเบียบวิธีอื่นทำการคำนวณหาคาบสัญญาณนาฬิกา หรือให้นักออกแบบเป็นคนกำหนดเองก่อนทำการสังเคราะห์ที่ระดับสูงเพื่อให้ได้วงจรควบคุมออกมา ทั้งนี้เนื่องจากระเบียบวิธีเหล่านี้จำเป็นต้องใช้คาบสัญญาณนาฬิกาอ้างอิงตลอดการสังเคราะห์ ดังนั้นการสังเคราะห์จึงถูกจำกัดด้วยสัญญาณนาฬิกานี้ ในทางตรงข้ามระเบียบวิธีในโครงการวิจัยนี้ทำการสังเคราะห์วงจรที่ระดับสูงก่อนโดยไม่คำนึงถึงสัญญาณนาฬิกา เมื่อได้จุดออกแบบที่เหมาะสมจึงทำการคำนวณหาคาบสัญญาณนาฬิกา วิธีการนี้ให้วงจรที่มีประสิทธิภาพมากเมื่อเทียบกับระเบียบวิธีอื่นดังรายละเอียดที่สามารถพิสูจน์ได้ต่อไปนี้

3.2 สถาปัตยกรรมของวงจรเป้าหมาย

โครงการวิจัยนี้เสนอสถาปัตยกรรมของวงจรดังรูปที่ 3.2 วงจรประกอบด้วยสองส่วนคือ ส่วนควบคุม (control) และส่วนดาต้าพาท (datapath) ส่วนควบคุมเป็นวงจร FSM ที่ถูกสังเคราะห์โดยระเบียบวิธีที่นำเสนอในโครงการนี้ ส่วนดาต้าพาทสามารถถูกสังเคราะห์โดย CAD ทั่วไป ในที่นี้สมมติว่าแต่ละโมดูลในส่วนดาต้าพาทได้ถูกสังเคราะห์ไว้เรียบร้อยแล้ว ประกอบด้วย วงจรฟังก์ชัน (คณิตศาตร์ ลอจิก เป็นต้น) สำหรับการคำนวณ รีจิสเตอร์สำหรับเก็บข้อมูล มัลติเพลกเซอร์สำหรับในกรณีมีการใช้วงจรฟังก์ชันร่วมกันของหลายโอเปอเรเตอร์ เป็นต้น และในที่นี้เพื่อลดความยุ่งยากของอัลกอริทึม และความซับซ้อนของการต่อสาย (interconnection) แต่ละฟังก์ชันจะมีรีจิสเตอร์ของ

ตัวเอง กล่าวคือจะไม่มีมัลติเพลกเซอร์หน้ารีจิสเตอร์นั่นเอง แต่ละโมดูลมีพารามิเตอร์ เวลาทำงาน (execution time) และขนาดวงจร (area) พารามิเตอร์เหล่านี้มีความสำคัญต่อการวิเคราะห์เพื่อให้ได้วงจรควบคุมที่มีประสิทธิภาพ



รูปที่ 3.2: สถาปัตยกรรมของวงจรของโครงการวิจัยนี้

จาก DFG อันหนึ่งสามารถมี FSM ได้วงจรเดียวหรือหลายวงจรขึ้นกับผลการสังเคราะห์โดยระเบียบวิธีที่เสนอ ถ้าผลการสังเคราะห์พบว่าสามารถแบ่งขั้นเวลาได้ลงตัวด้วยคาบของสัญญาณนาฬิกาตัวเดียว ระเบียบวิธีจะสังเคราะห์ FSM เพียงตัวเดียวเพราะสามารถใช้สัญญาณนาฬิกาความถี่เดียวได้ แต่ถ้าหากการแบ่งเวลาไม่ลงตัว ผลการวิเคราะห์ให้คาบสัญญาณนาฬิกาหลายค่า ระเบียบวิธีจะสังเคราะห์ FSM ออกมาหลายตัว และการสร้างวงจรจริง (Implementation) ต้องใช้สัญญาณนาฬิกาหลายความถี่ ซึ่งอาจจะสร้างได้ยากและมี overhead มากในทางปฏิบัติ เช่น การสื่อสารระหว่าง FSM การจัดการสัญญาณนาฬิกาหลาย ๆ ตัว เป็นต้น

ในโครงการวิจัยนี้จึงจำกัดให้ระเบียบวิธีวิเคราะห์คาบสัญญาณนาฬิกาออกมาไม่เกิน 2 ค่า และสังเคราะห์ FSM ออกมาหลายตัวขึ้นกับการกระจายของขั้นเวลาที่มีคาบเท่ากัน ขั้นเวลาที่มีค่า

คาบเท่ากันและต่อเนื่องกันจะถูกรวมไว้ใน FSM ตัวเดียวกัน หากขั้นเวลาตามด้วยขั้นเวลาที่มีคาบต่างกันจะต้องมี FSM ใหม่ให้กับขั้นเวลาที่ตามมาด้วย

การติดต่อสื่อสารระหว่าง FSMs จำแนกได้ 2 อย่างคือ

- ◆ การติดต่อภายใน เป็นการติดต่อกันระหว่าง FSM ที่ควบคุม DFG ตัวเดียวกัน ทำได้โดยการส่งสัญญาณ push จาก FSM ที่มาก่อนไปยัง FSM ที่มาหลัง ในลักษณะเหมือนการล้มของโดมิโน ดังแสดงในรูปที่ 3.2
- ◆ การติดต่อภายนอก เป็นการติดต่อกับ FSM อื่นภายนอก หรือระบบภายนอก ดังแสดงในรูปที่ 3.2 โดยเมื่อมีสัญญาณ start จากภายนอกมา FSM ตัวแรกจะเริ่มทำงาน จากนั้นจะผลักให้ FSM ที่ตามมาทำงานต่อ ๆ กัน และเมื่อจบการทำงานทั้งหมด FSM ตัวสุดท้ายจะสร้างสัญญาณ done ออกมาเพื่อบอกถึงการทำงานที่เสร็จเรียบร้อยแล้ว

3.3 การเลือกจุดออกแบบ

การออกแบบวงจรรวมมีข้อจำกัดหลายประการ เช่น ความเร็ว ขนาดวงจร การใช้กำลังไฟฟ้า เป็นต้น วงจรที่เหมาะสมที่สุดขึ้นอยู่กับลักษณะการนำไปใช้งาน ซอฟท์แวร์ช่วยออกแบบโดยทั่วไปจึงมักจะอนุญาตให้นักออกแบบระบุความต้องการในการออกแบบ (Design constraint) เพื่อให้วงจรที่สังเคราะห์ได้ตรงกับความต้องการของนักออกแบบมากที่สุด

โครงการวิจัยนี้เสนอเทคนิคในการเลือกจุดออกแบบที่เหมาะสม รวมถึงระเบียบวิธีในการสร้างจุดออกแบบที่เหมาะสมที่สุดออกมาหลาย ๆ จุดเพื่อเป็นทางเลือก การทำเช่นนี้เรียกว่า Design space exploration ซึ่งเป็นคุณสมบัติที่ซอฟแวร์ช่วยออกแบบควรจะมีเป็นอย่างยิ่ง

3.3.1. Pareto Points

จุดออกแบบที่เหมาะสมที่สุดเรียกว่า จุดพาริโต (Pareto point) [Micheli94] มีนิยามดังนิยามที่ 3.1 จุดออกแบบ (Design point) ในที่นี้คือ D(S, A) ใด ๆ ซึ่งเป็นโคออร์ดิเนเตอร์ของความเร็ว (S: speed) และพื้นที่ (A: area) ของวงจร เนื่องจากความเร็วและขนาดของวงจรมักเป็นแฟกเตอร์ที่ผกผันกันเสมอ วงจรที่มีความเร็วสูงมักมีขนาดที่ใหญ่ วงจรที่มีความเร็วต่ำมักมีขนาดเล็ก ดังนั้นจุดพาริโตจึงมองได้ 2 มุม คือ วงจรที่เหมาะสมที่สุดในด้านความเร็ว และวงจรที่ดีที่สุดในด้านขนาด กล่าวคือในบรรดาจุดออกแบบที่มีขนาดเท่ากัน จุดออกแบบที่มีความเร็วสูงสุดเป็นจุดพาริโต และในบรรดาจุดออกแบบที่มีความเร็วเท่ากัน จุดออกแบบที่มีขนาดเล็กที่สุดเป็นจุดพาริโต

นิยาม 3.1 **จุดพาริโต** (Pareto point) **หมายถึงจุดออกแบบ** D(S, A) **ที่ไม่มีจุดออกแบบอื่นมีเวลาการทำงาน** (S: speed **หรือ** latency) **ต่ำกว่าอีกแล้วที่มีพื้นที่** (A: area) **เท่ากัน หรือจุดออกแบบที่ไม่มีจุดออกแบบอื่นมีพื้นที่น้อยกว่าอีกแล้วที่มี** latency **เท่ากัน**

3.3.2. Making Resource-Sharing Decisions

การใช้ทรัพยากรร่วมกัน (resource-sharing) ของโอเปอเรชันต่าง ๆ มีความจำเป็นในการออกแบบวงจรในกรณีที่มีทรัพยากรจำกัด ทรัพยากร (resource) ในที่นี้คือ วงจรฟังก์ชันต่าง ๆ และรีจิสเตอร์ เป็นต้น การใช้ทรัพยากรร่วมกันทำให้ขนาดของวงจรเล็กลงแต่มักจะทำให้วงจรทำงานได้ช้าลง ถ้ามีการใช้ทรัพยากรร่วมกันมากเกินไปทำให้วงจรทำงานช้าลงมาก นอกจากนี้ขนาดและเวลาของมัลติเพลกเซอร์อาจจะใหญ่กว่าและทำงานนานกว่าตัวทรัพยากรเองด้วย ดังนั้นการสังเคราะห์ที่ให้วงจรที่เหมาะสมจึงจำเป็นต้องอาศัยเทคนิคการตัดสินใจที่ดี

ในหลายงานวิจัย เช่น [Kim94] [Memik03] [Jaschke98] [Raji97] เป็นต้น ได้เสนอระเบียบวิธีที่มีประสิทธิภาพโดยการทำ resource-sharing ในระหว่างการทำ scheduling เพื่อทำให้วงจรที่ได้มีความเร็วและขนาดที่เหมาะสม แต่อย่างไรก็ตามวิธีการเหล่านี้มีความซับซ้อนของอัลกอริทึมและใช้เวลาในการคำนวณมาก

โครงการวิจัยนี้เสนอเทคนิคในการประมาณค่า Area-speed ratio production ($A.S_{ratio}$) เพื่อช่วยให้การตัดสินใจปริมาณการทำ resource-sharing ได้ก่อนการทำ scheduling ซึ่งสามารถลดเวลาในการค้นหาจุดออกแบบที่เหมาะสมที่สุด ในที่นี้ $A.S_{ratio}$ ถูกนิยามไว้ดังสมการที่ (3.1)

$$A_{ratio} = A_{mux} / A_{resource}$$
$$S_{ratio} = S_{mux} / S_{resource}$$
$$A.S_{ratio} = A_{ratio} * S_{ratio} \qquad (3.1)$$

โดยที่

A_{ratio} คือ อัตราส่วนระหว่างพื้นที่ของมัลติเพลกเซอร์ (A_{mux}) และพื้นที่ของ resource $(A_{resource})$

S_{ratio} คือ อัตราส่วนระหว่างเวลาทำงานของมัลติเพลกเซอร์(S_{mux}) และเวลาทำงานของ resource $(S_{resource})$

ค่า $A.S_{ratio}$ ของแต่ละ resource จะถูกคำนวณเตรียมไว้ก่อนการประมาณค่า $A.S_{ratio}$ ของทั้งวงจร ค่าประมาณของ $A.S_{ratio}$ ของทั้งวงจรนี้คือค่าเฉลี่ยของ $A.S_{ratio}$ ของ resource ที่ถูกใช้สร้างวงจรทั้งหมด ค่า $A.S_{ratio}$ ที่เป็นไปได้ทั้งหมดของวงจรต้องถูกทำการคำนวณทุกกรณีของจุด

ออกแบบ ยกเว้นกรณีที่ A_{ratio} และ/หรือ S_{ratio} มีค่ามากกว่า 1 เพราะนั่นแสดงว่าปริมาณการใช้ทรัพยากรร่วมกันมากเกินไป ค่าประมาณที่ได้นี้อาจจะมีความคลาดเคลื่อนแต่มีแนวโน้มตรงกันกับค่า $A.S_{ratio}$ ของวงจรที่ถูกสร้างจริงดังรายละเอียดในบทความ [Nattha05-1] และ [Nattha05-2]

การตัดสินใจเลือกจุดออกแบบทำได้โดยการพิจารณาค่าประมาณ $A.S_{ratio}$ ทั้งหมดที่ถูกเตรียมไว้ดังนี้ ถ้าหากจุดออกแบบ 2 จุดใดๆ มีค่า A_{ratio} เท่ากัน จุดออกแบบที่มีค่าประมาณ $A.S_{ratio}$ น้อยกว่าจะให้วงจรที่มีความเร็วสูงกว่า ในทำนองเดียวกันถ้าหากจุดออกแบบ 2 จุดใดๆ มีค่า S_{ratio} เท่ากัน จุดออกแบบที่มีค่าประมาณ $A.S_{ratio}$ น้อยกว่าจะให้วงจรที่มีขนาดวงจรเล็กกว่า สังเกตเห็นได้ว่าการตัดสินใจดังกล่าวสอดคล้องกับคำนิยามของจุดพาริโต

3.3.3. Reduction of Resource-Constraints

ดังที่ได้อธิบายในหัวข้อ 3.3.2 เนื่องจากระเบียบวิธีต้องทำการวิเคราะห์ทุกจุดออกแบบซึ่งทำให้เสียเวลา ในหัวข้อนี้จะอธิบายถึงการลดจุดออกแบบที่ต้องนำมาวิเคราะห์ดังรายละเอียดต่อไปนี้

การทำ Design space exploration เป็นงานที่ใช้เวลามาก เนื่องจากต้องทำการวิเคราะห์ค้นหา (Search) อย่างละเอียดเพื่อหาจุดออกแบบที่เป็นจุดพาริโต งานวิจัย [Blythe00] ได้เสนอระเบียบวิธีที่สามารถค้นหาจุดพาริโตสำหรับ Time-constraints ต่าง ๆ ได้อย่างรวดเร็ว แต่ระเบียบวิธีอาจจะเพิกเฉยจุดพาริโตบางจุดไป และอาจใช้เวลานานในกรณีไม่มีจุดออกแบบที่เข้าเกณฑ์ lower-bound area ของระเบียบวิธี

โครงการวิจัยนี้นำเสนอระเบียบวิธีการค้นหาจุดพาริโตโดยการใช้ Resource-constraints แทนที่จะเป็น Time-constraints ดังระเบียบวิธีอื่น ทั้งนี้เนื่องจากระเบียบวิธีการทำ scheduling ในหัวข้อ 3.4 ไม่จำเป็นต้องอ้างอิงกับคาบสัญญาณนาฬิกา ก่อนทำ scheduling ต้องทำการคำนวณหาเซทของ Resource-constraints ออกมาก่อน แต่ละ Resource-constraint ในเซทนี้จะไม่ซ้ำกันและจะให้จุดพาริโตออกมา ระเบียบวิธีสามารถถูกเรียบเรียงได้ดังต่อไปนี้

ขั้นตอนที่ 1: จาก DFG เริ่มต้นที่ให้มา ทำการกำนดลำดับการเกิด (occurrence order: L_i) ให้กับแต่ละโอเปอเรชัน O_i ในกรณีที่โอเปอเรชันมีโอเปอเรชันนำหน้า (predecessor operation) หลายโหนด ค่าลำดับการเกิดเป็นค่าที่มากที่สุด ดังตัวอย่างที่แสดงใน DFG รูปที่ 3.3 ค่า L_i ของแต่ละ O_i แสดงอยู่ภายในวงเล็บ () สังเกตเห็นว่า O_{11} มีโอเปอเรชันนำหน้า 2 โหนดซึ่งมีค่าลำดับการเกิดเท่ากับ 2 และ 3 แสดงว่าค่า L_{11} ต้องเป็น 4

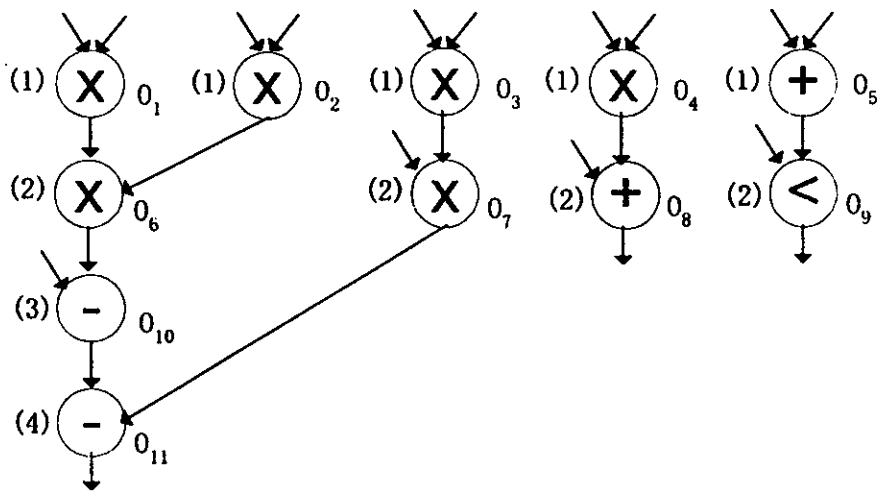
ขั้นตอนที่ 2: ทำการหาจำนวนครั้งที่เกิดพร้อมกันมากที่สุด $\max(n_k)$ ของโอเปอเรชันชนิด n_k ใด ๆ ที่เกิดพร้อมกัน (ลำดับการเกิดเท่ากัน) กล่าวคือ $\max(n_k)$ แทนจำนวนโอเปอเรชันชนิดเดียวกันที่ต้องใช้ทรัพยากรชนิดเดียวกัน จากรูปที่ 3.3 ค่าจำนวนครั้งการเกิดพร้อมกันมาก

ที่สุดของโอเปอเรชันคูณ $\max(\times)=4$ และทำนองเดียวกัน $\max(+)=1$ $\max(-)=1$ และ $\max(<)=1$ ดังนั้นการสร้าง DFG นี้ต้องการทรัพยากรมากที่สุดถึง 4 วงจรคูณ 1 วงจรบวก 1 วงจรลบ และ 1 วงจรเปรียบเทียบ

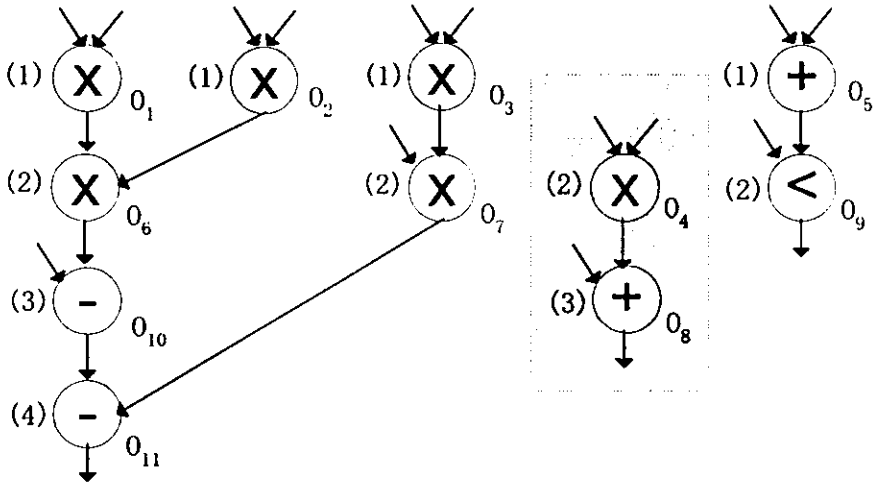
ขั้นตอนที่ 3: ทำการลดขนาดของเซท Resource-constraints โดยการพิจารณาความสัมพันธ์ของข้อมูล (data dependence) และค่า $\max(n_k)$ ที่ได้จากขั้นตอนที่ 2 สังเกตเห็นว่าโอเปอเรชันคูณมีทั้งหมด 6 ครั้ง แต่ใช้วงจรคูณเพียง 4 วงจรก็ทำให้วงจรทำงานได้เร็วเท่ากัน ทั้งนี้เนื่องจากความสัมพันธ์ของข้อมูลทำให้โอเปอเรชันคูณทั้งหมด 6 ครั้งไม่ได้เกิดพร้อมกัน รูปที่ 3.5 (a) แสดงเซทเต็มของ Resource-constraints ที่เป็นไปได้ทั้งหมด และรูปที่ 3.5(b) แสดงเซทที่ลดแล้ว ในที่นี้ constraint (6,2,2,1) หมายถึง DFG ในรูปที่ 3.3 ถูกสร้างโดยใช้ 6 วงจรคูณ 2 วงจรบวก 2 วงจรลบ และ 1 วงจรเปรียบเทียบ เป็นต้น

ขั้นตอนที่ 4: ทำการเปลี่ยนลำดับการเกิดของโอเปอเรชันถ้าการเปลี่ยนแปลงนี้ไม่เปลี่ยนแปลงเวลาทำงานของวงจร สามารถลด $\max(n_k)$ ของโอเปอเรชันลงได้อีก และไม่เพิ่มค่า $\max(n_k)$ ของโอเปอเรชันชนิดอื่น ดังตัวอย่างในรูปที่ 3.4 ลำดับของ O_4 และ O_8 สามารถเปลี่ยนได้ ทำให้ $\max(\times)=3$

ขั้นตอนที่ 5: ทำซ้ำขั้นตอนที่ 3 และ 4 จนกว่าจะได้เซท Resource-constraints สุดท้ายที่ลดขนาดไม่ได้อีกแล้ว รูปที่ 3.5(c) แสดงเซท Resource-constraints ที่ลดได้



รูปที่ 3.3: DFG ของวงจรแก้สมการอนุพันธ์อันหนึ่ง



รูปที่ 3.4: DFG **ที่มีการเปลี่ยนลำดับการเกิด** O_4 **และ** O_8 **จากรูปที่** 3.3

{ (6, 2, 2, 1), (6, 1, 2, 1), (6, 2, 1, 1), (6, 1, 1, 1),
(5, 2, 2, 1), (5, 1, 2, 1), (5, 2, 1, 1), (5, 1, 1, 1),
(4, 2, 2, 1), (4, 1, 2, 1), (4, 2, 1, 1), (4, 1, 1, 1),
(3, 2, 2, 1), (3, 1, 2, 1), (3, 2, 1, 1), (3, 1, 1, 1),
(2, 2, 2, 1), (2, 1, 2, 1), (2, 2, 1, 1), (2, 1, 1, 1),
(1, 2, 2, 1), (1, 1, 2, 1), (1, 2, 1, 1), (1, 1, 1, 1) }

(a)

{(4, 1, 1, 1), (3, 1, 1, 1),
(2, 1, 1, 1), (1, 1, 1, 1) }

(b)

{ (3, 1, 1, 1),
(2, 1, 1, 1), (1, 1, 1, 1) }

(c)

รูปที่ 3.5: **เซทของ** Resource-constraints **โดยที่** (a) **คือ เซทเต็มของ** Resource-constraints **ที่เป็นไปได้ทั้งหมด** (b) **คือ เซทที่ลดลงโดยความสัมพันธ์ของข้อมูล** (c) **คือ เซทที่ลดลงโดยการเปลี่ยนลำดับการเกิด**

จากคำอธิบายและตัวอย่างการใช้อัลกอริทึมข้างต้น จะเห็นว่าขนาดของเซท Resource-constraints ที่ลดแล้วมีขนาดเล็กว่าเซทเต็มมาก นั่นแสดงว่าระเบียบวิธีสามารถลดเวลาในการวิเคราะห์จุดออกแบบที่ไม่ใช่จุดพาริโตลงได้มาก

3.4 การจัดลำดับโอเปอเรชันแบบอะซิงโครนัส

โครงการวิจัยนี้นำหลักการของการออกแบบวงจรอะซิงโครนัสมาใช้ในการสังเคราะห์วงจรที่ระดับสูง วงจรอะซิงโครนัสเป็นวงจรที่ไม่จำเป็นต้องใช้สัญญาณนาฬิกาในการกำหนดจังหวะการทำงาน การติดต่อสื่อสารระหว่างโมดูลทำได้โดยการใช้โปรโตคอลที่มีสัญญาณร้องขอ (Request signal) ของฝ่ายส่งข้อมูล และสัญญาณตอบรับ (Acknowledge signal) ฝ่ายรับข้อมูล ดังนั้นแต่ละโมดูลสามารถทำงานได้ด้วยความเร็วของวงจร ไม่ต้องมีเวลาว่างหรือสแลคในการรอขอบสัญญาณนาฬิกาถัดไป ด้วยคุณสมบัติดังกล่าวทำให้หลาย ๆ งานวิจัยที่นำหลักการวงจรอะซิงโครนัสมา

ประยุกต์ใช้ในการสังเคราะห์วงจรที่ระดับสูงเป็นผลสำเร็จโดยการสังเคราะห์ทำให้ได้วงจรอะซิงโครนัสที่มีประสิทธิภาพ เช่น [Bachman99] [Saito05] [Nattha03] [Theobald01] เป็นต้น

การจัดลำดับการทำงานของโอเปอเรชันในระเบียบวิธีทางซิงโครนัสที่ผ่านมาทำโดยการแบ่ง DFG ออกเป็นช่วงเวลาที่เรียกว่า control steps แต่ละ control step มักจะหมายถึงหนึ่งคาบของสัญญาณนาฬิกาซึ่งมักเรียกว่า clock-cycle หรือหนึ่งรอบของสัญญาณนาฬิกา โดยทั่วไปแต่ละโอเปอเรชันมักจะถูกจัดให้ทำงานเสร็จภายในหนึ่ง control step นั่นหมายความว่า คาบสัญญาณนาฬิกาจะขึ้นcontrol step ที่ยาวที่สุด หรือโอเปอเรชันที่ทำงานช้าที่สุด มีงานวิจัยหลายงานได้แก้ปัญหานี้ด้วยวิธีต่าง ๆ โดยการจัดให้โอเปอเรชันที่ทำงานนานสามารถทำงานเสร็จภายในหลาย clock-cycle ได้ ดังเช่นงานวิจัยของ [Jung02] [Chang96] [Blythe00] [Jain88] [Park85] [Parker86] เป็นต้น แต่อย่างไรก็ตามระเบียบวิธีเหล่านี้ยังถูกจำกัดด้วยคาบสัญญาณนาฬิกาอยู่ดี

ถึงแม้ในโครงการวิจัยนี้มีจุดประสงค์เพื่อสังเคราะห์วงจรควบคุม FSMs ซึ่งเป็นวงจรซิงโครนัสต้องอาศัยสัญญาณนาฬิกาในการกำหนดจังหวะการทำงาน แต่การนำหลักการของวงจรอะซิงโครนัสมาประยุกต์ใช้ในการสังเคราะห์วงจรที่ระดับสูงสามารถสังเคราะห์ได้วงจรที่มีประสิทธิภาพมากทั้งในด้าน ขนาดวงจร ความเร็ว และการใช้กำลังไฟฟ้า การจัดลำดับการทำงานของโอเปอเรชันในที่นี้จึงถูกเรียกว่า การจัดลำดับโอเปอเรชันแบบอะซิงโครนัส (Asynchronous style scheduling) ซึ่งทำการจัดลำดับของโอเปอเรชันให้เริ่มทำงานได้เร็วที่สุดเท่าที่ทำได้ (As Soon As Possible หรือ อัลกอริทึม ASAP [Micheli94]) โดยไม่คำนึงถึงคาบสัญญาณนาฬิกา อัลกอริทึม ASAP จะส่งผลให้วงจรทำงานเร็วที่สุด และเมื่อรวมกับการจัดลำดับโอเปอเรชันแบบอะซิงโครนัสแล้ว วงจรที่ได้จะทำงานได้เร็วกว่าวงจรซิงโครนัสเนื่องจากไม่มีสแลค โครงงานวิจัยนี้เรียกอัลกอริทึมดังกล่าวว่า Asynchronous style ASAP scheduling รูปที่ 3.1(a) และ 3.6(a) แสดงให้เห็นถึงหลักการของอัลกอริทึมดังกล่าว จะเห็นว่าแต่ละโอเปอเรชันสามารถเริ่มทำงานได้ทันทีโอเปอเรชันที่นำหน้า (predecessor operation) ทำงานเสร็จโดยไม่ต้องรอสัญญาณนาฬิกา

ในโครงการวิจัยนี้ประยุกต์ใช้ระเบียบวิธี ASAP scheduling แบบอะซิงโครนัสนี้กับเซทของ resource-constraints ที่ลดแล้วจากหัวข้อที่ 3.3 ระเบียบวิธีจะทำการจัดลำดับโอเปอเรชันและกำหนดทรัพยากรให้กับแต่ละโอเปอเรชัน เพื่อให้วงจรที่ทำงานเร็วที่สุดภายใต้กรอบของแต่ละ resource-constraint จุดออกแบบที่ได้จะเป็นจุดพาริโตออกมาสำหรับ resource-constraint นั้น ๆ

3.5 การเลือกคาบสัญญาณนาฬิกา

ระเบียบวิธีหลายงานวิจัย เช่น [Bhattach98] [Blythe00] [Chang96] [Juan96] เป็นต้น ทำการเลือกคาบสัญญาณนาฬิกาก่อนการทำ resource-sharing และ scheduling เพื่อลดเวลาในการคำนวณ แต่การคำนวณหาคาบสัญญาณนาฬิกาโดยไม่คำนึงถึงการ scheduling หรือลักษณะของ DFG เลยอาจจะได้วงจรที่ไม่เหมาะสมที่สุด

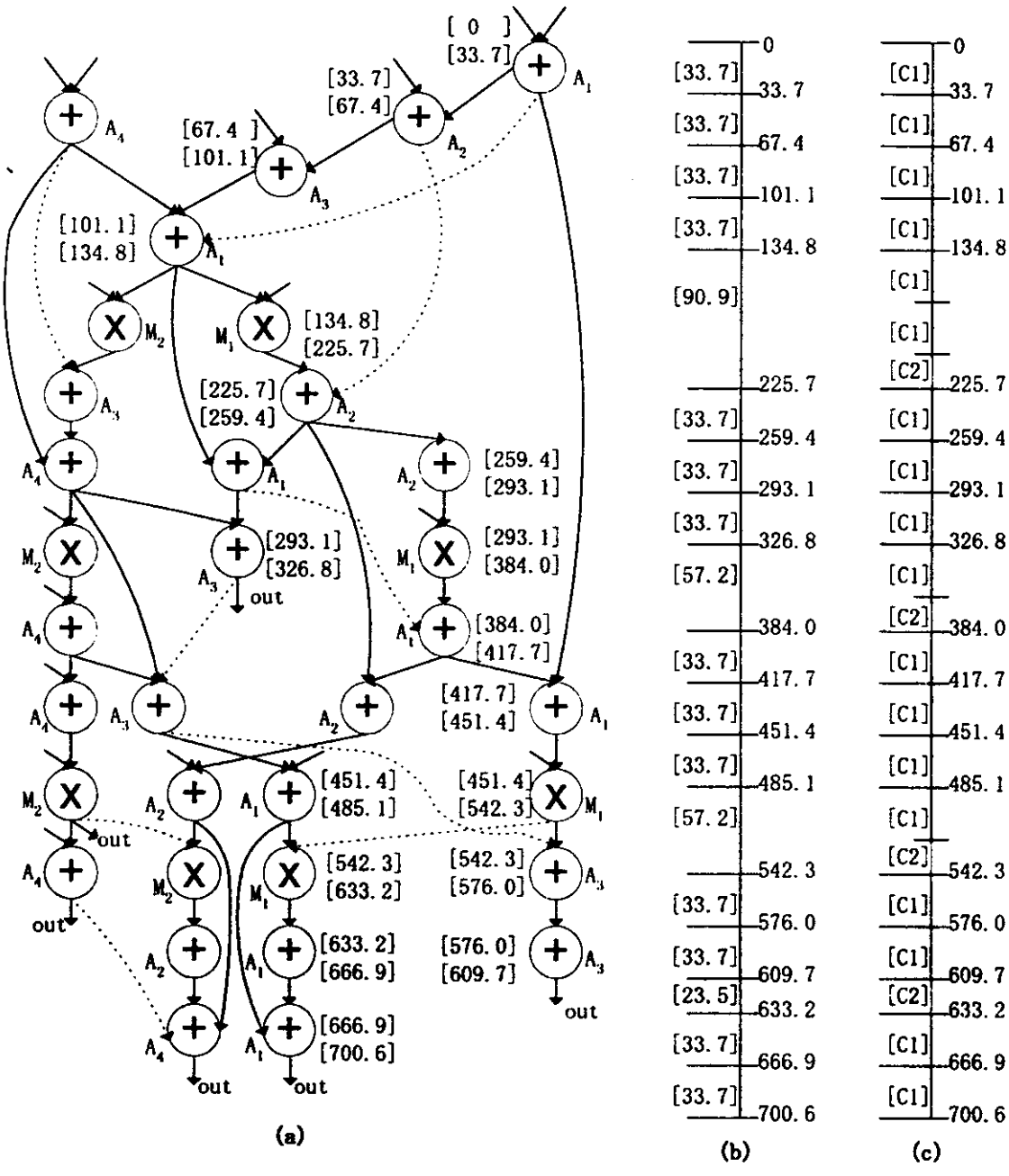
ในทางตรงกันข้ามโครงการวิจัยนี้ทำการคำนวณค่าคาบสัญญาณนาฬิกาหลังจากการทำ ASAP scheduling แบบอะซิงโครนัสภายใต้ resource-constraint หนึ่ง ๆ ระเบียบวิธีการคำนวณค่าคาบสัญญาณนาฬิกามีรายละเอียดดังต่อไปนี้ ในที่นี้สมมติว่า เวลาทำงานของโอเปอเรชันบวกมีค่าเท่ากับ33.7ns และเวลาทำงานของโอเปอเรชันคูณมีค่าเท่ากับ 90.9ns ซึ่งนำมาจาก [Chnag96]

ขั้นตอนที่ 1: คำนวณเวลาเริ่ม (start time) และเวลาจบ (end time)ให้กับแต่ละโอเปอเรชัน start time คือเวลาที่เร็วที่สุดที่โอเปอเรชันหนึ่ง ๆ สามารถเริ่มได้ end time คือเวลา start time บวกกับเวลาการทำงาน (execution time) ของโอเปอเรชันนั้น ๆ ดังแสดงในรูปที่ 3.6(a) ที่แต่ละโอเปอเรชัน start time แสดงไว้ในวงเล็บ [-] ด้านบน ส่วน end time แสดงไว้ในวงเล็บ [-] ด้านล่าง สังเกตว่า start time และ end time ไม่ได้แสดงไว้ทุกโอเปอเรชัน เนื่องจากเพื่อลดความซับซ้อนในรูปภาพ

ขั้นตอนที่ 2: ทำการฉายเวลา start time และ end time ทั้งหมดลงไปบนขั้นเวลาที่อนุกรมกันอย่างต่อเนื่อง ในที่นี้ ขั้นเวลา (time step) หมายถึง control step หรือ 1 สเตท ของ FSM นั่นเอง การฉายเวลาของ DFG ในรูปที่ 3.6(a) ได้ผลดังรูปที่ 3.6(b)

ขั้นตอนที่ 3: ทำการแยกกลุ่ม time step ที่มีค่าดีเลย์เท่ากัน จากรูปที่ 3.6(b) สามารถแยกได้ 3 กลุ่ม ดังนี้ 22.5ns 33.7ns 57.2ns และ 90.9ns

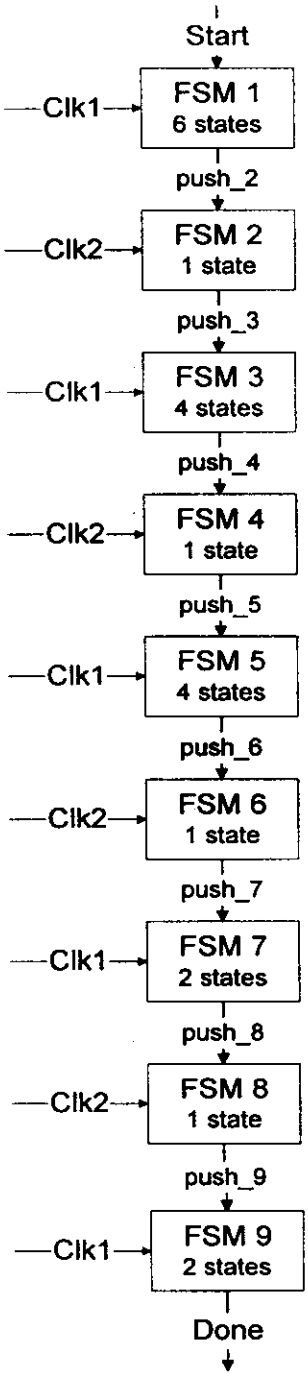
ขั้นตอนที่ 4: ทำการลดกลุ่ม time step ถ้ามันสามารถแทนได้ด้วย time step อื่นที่มีอยู่ตั้งแต่สองขั้นขึ้นไป เนื่อง time step หมายถึง คาบสัญญาณนาฬิกา นั่นแสดงว่าเราจำเป็นต้องลดกลุ่มของ time step ให้เหลือน้อยที่สุดเพื่อลดความยุ่งยากในการสร้างวงจรจริง ดังแสดงในรูปที่ 3.6(c) time step 90.9ns ถูกแทนด้วย time step 3 ขั้นคือ 33.7ns+33.7ns+ 22.5ns เป็นต้น ในขั้นตอนสุดท้ายนี้ time step ลดวงเหลือ 2 กลุ่มคือ 33.7ns และ 22.5ns นั่นคือในการสร้างวงจรต้องมีวงจรกำเนิดสัญญาณนาฬิกา 2 ความถี่นั่นเอง



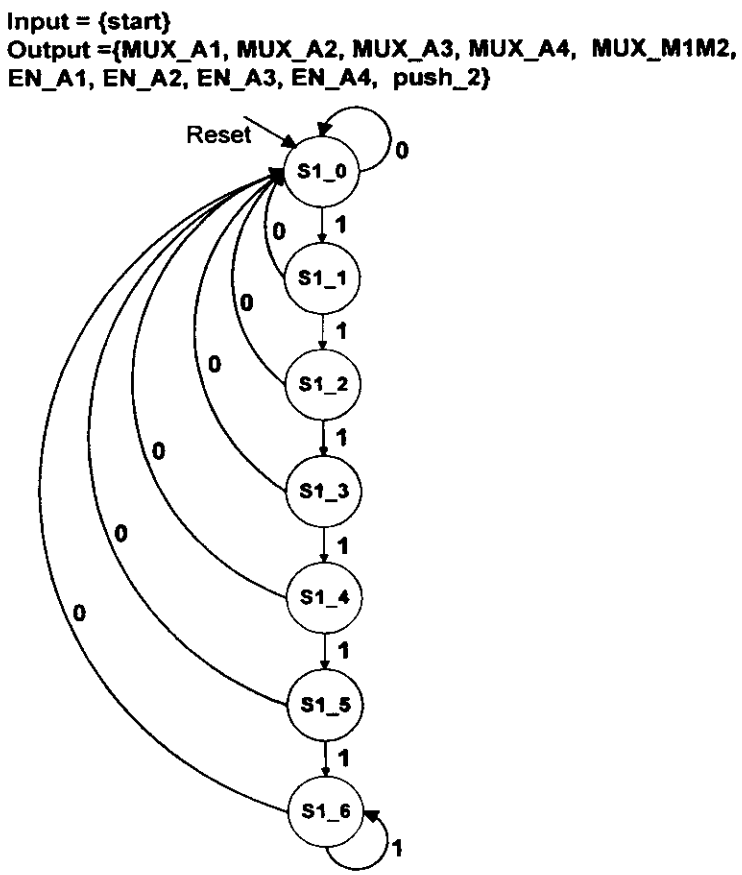
รูปที่ 3.6: An elliptical wave filter (EWF) taken from [Bachman99]: (a) a DFG with asynchronous style ASAP scheduling, (b)time steps, (c) clock assignment.

3.6 การสังเคราะห์ FSMs

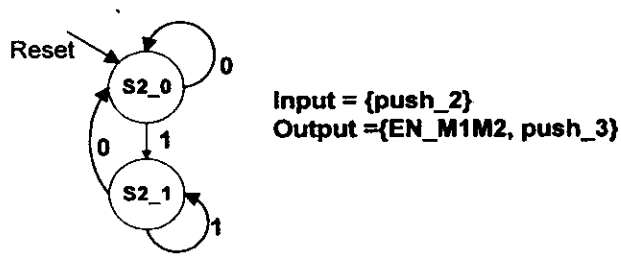
จากสถาปัตยกรรมของวงจรเป้าหมายที่ได้อธิบายในหัวข้อที่ 3.2 และระเบียบวิธีที่ได้อธิบายในหัวข้อที่ 3.5 สามารถสังเคราะห์ FSMs สำหรับ DFG ในรูปที่ 3.6 ได้ดังแสดงในรูปที่ 3.7 ซึ่งประกอบด้วย FSMs ย่อยทั้งหมด 9 วงจรด้วยกัน มีการใช้สัญญาณนาฬิกา 2 ความถี่



(a) An FSM of an EWF



(b) The state diagram of FSM 1



(b) The state diagram of FSM 2

รูปที่ 3.7: FSM ที่สังเคราะห์ได้จาก DFGในรูปที่ 3.6

การสังเคราะห์ FSM ออกมาหลายตัวขึ้นกับการกระจายของขั้นเวลาที่มีคาบเท่ากัน ขั้นเวลาที่มีค่าคาบเท่ากันและต่อเนื่องกันจะถูกรวมไว้ใน FSM ตัวเดียวกัน หากขั้นเวลาตามด้วยขั้นเวลาที่มีคาบต่างกันจะต้องมี FSM ใหม่ให้กับขั้นเวลาที่ตามมาด้วย ดังแสดงในรูปที่ 3.6(c) ขั้นเวลาที่ 1 ถึงขั้นเวลาที่ 6 มีค่าดีเลย์ 33.7ns เท่ากันอย่างต่อเนื่องจึงถูกรวมอยู่ในที่เดียวกันคือ FSM 1 ในรูปที่ 3.7 ส่วนขั้นเวลาที่ 7 มีดีเลย์ 22.5ns ต่างออกไปจึงถูกจัดไว้ใน FSM อีกอันหนึ่งคือ FSM 2 ในรูปที่ 3.7 เป็นต้น

State transition diagram ของ FSM 1แสดงดังในรูปที่ 3.7(b) FSM 1 ต้องควบคุม 6 เวลา ต้องมี 7 สเตท โดยสเตท 1-6 ทำหน้าที่ควบคุมขั้นเวลาที่ 1-6 และสเตท 0 มีไว้เพื่อเตรียมพร้อมรอการเรียกใช้จากวงจร FSM ส่วนอื่น ในทำนองเดียวกัน FSM 2 ซึ่งควบคุมขั้นเวลาที่ 7 เพียงขั้นเดียวมี State transition diagram ดังในรูปที่ 3.7(c) ยกเว้น สเตท 0 ในสเตทอื่น FSM ต้องสร้างสัญญาณควบคุมดาต้าพาทที่เหมาะสม ในรูปแสดงเพียงแต่สัญญาณอินพุทและเอาท์พุทของแต่ละ FSM เท่านั้น การเปลี่ยนแปลงของสัญญาณเหล่านี้ในแต่ละสเตทได้ละไว้เพื่อความง่ายในการแสดงภาพ

สังเกตเห็นว่าขั้นเวลาที่มีดีเลย์เท่ากันจะไม่ถูกรวมไว้ใน FSM เดียวกัน เนื่องจากขั้นเวลาเหล่านี้ไม่ต่อเนื่องกัน ถ้ารวมทั้งหมดเข้าด้วยจะทำให้ FSM มีความซับซ้อนมาก มีปัญหาของการซิงโครไนซ์กัน และไม่ได้วงจรที่เร็วตามที่วิเคราะห์ไว้

ข้อควรระวังในการสร้างวงจรจริงมีดังนี้ การสร้างวงจร FSM ที่มีสัญญาณนาฬิกาความถี่เดียวทำได้ยาก แต่การสร้างวงจร FSM ที่มีสัญญาณนาฬิกา 2 ความถี่ขึ้นไปทำได้ยาก โดยเฉพาะในสถาปัตมกรรมที่นำเสนอนี้ ผู้ออกแบบต้องสามารถควบคุมสัญญาณนาฬิกาให้มีลักษณะเริ่มและหยุด (pausesible clocks) ได้ดังต้องการ ซึ่งเป็นงานวิจัยที่ท้าทายที่น่าจะดำเนินต่อไป