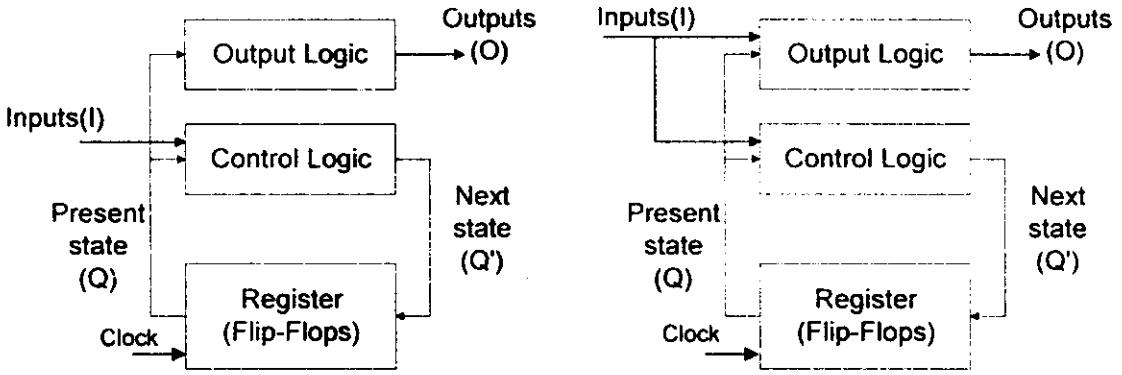
บทที่ 2
การทบทวนวรรณกรรม

เนื้อหาในบทนี้มีการอธิบาย ทฤษฎีพื้นฐานที่เกี่ยวข้องกับการสังเคราะห์วงจรควบคุม FSMs ปัญหาที่ต้องแก้ไข และงานวิจัยที่เกี่ยวข้อง

2.1 Finite State Machines (FSMs)

ในงานวิจัยนี้ Finite State Machines (FSMs) ถูกเลือกใช้เป็นวงจรควบคุม เนื่องจากสังเคราะห์ได้ง่าย รูปแบบไม่ซับซ้อน และเป็นที่นิยมใช้เป็นวงจรควบคุมในการออกแบบวงจรรวมดิจิตอลโดยทั่วไป โดย FSMs ที่นิยมใช้มี 2 รูปแบบด้วยกันคือ Moore machine และ Mealy machine ดังโครงสร้างของวงจรในรูปที่ 2.1(a) และ 2.1(b) ตามลำดับ



รูปที่ 2.1 โครงสร้างวงจร FSMs แบบ Moore และ Mealy machine

โครงสร้างวงจรของ FSMs ทั้งสองแบบประกอบด้วยวงจร 3 ส่วนดังนี้

- วงจรลอจิกขาออก (Output Logic) ทำหน้าที่สร้างสัญญาณเอาท์พุท ซึ่งเป็นสัญญาณที่ใช้ในการควบคุมและติดต่อกับวงจรส่วนอื่น
- วงจรลอจิกควบคุม (Control Logic) ทำหน้าที่ควบคุมวงจรโดยการรับสัญญาณอินพุทและสเตทปัจจุบันมาประมวลผลเพื่อตัดสินใจว่าสเตทถัดไปควรจะเป็นอะไร
- รีจิสเตอร์ (Register) ทำหน้าที่จำสเตทปัจจุบันไว้ รีจิสเตอร์นี้สามารถถูกสร้างจากฟลิป-ฟลอบ จำนวนของฟลิป-ฟลอบขึ้นกับจำนวนสเตททั้งหมดของวงจร

สัญญาณนาฬิกา (Clock) ทำหน้าที่สำคัญคือ กำหนดจังหวะการทำงานของวงจร FSMs ค่าของสเตทที่ได้จากวงจรลอจิกควบคุมจะถูกบันทึกในรีจิสเตอร์ทุกครั้งที่มีขอบสัญญาณนาฬิกา ถ้าฟลิบ-ฟลอบที่ใช้เป็นชนิดทำงานที่ขอบขาขึ้น (positive edge-triggered) ค่าของสเตทดังกล่าวจะถูกบันทึกไว้ในรีจิสเตอร์ภายในเวลาไม่กี่นาโนวินาทีหลังจากมีขอบขาขึ้นของสัญญาณนาฬิกา วงจรลักษณะนี้ถูกเรียกว่า **วงจรซิงโครนัส** (Synchronous circuits) การทำงานของวงจรต้องซิงโครไนซ์กับสัญญาณนาฬิกา

ความแตกต่างของ Moore machine และ Mealy machine อยู่ที่วงจรลอจิกขาออก โดยในกรณีของ Moore machine ดังแสดงในรูปที่ 2.1(a) และสมการที่ (2.1) สัญญาณเอาท์พุทเป็นฟังก์ชันของสเตทปัจจุบันเพียงอย่างเดียว การเปลี่ยนแปลงของสัญญาณอินพุทไม่มีผลต่อสัญญาณเอาท์พุท ข้อดีคือเอาท์พุทจะมีค่าคงที่หรือสเถียรตลอดสเตท เอาท์พุทจะเปลี่ยนก็ต่อเมื่อมีการเปลี่ยนสเตทใหม่

Moore machine

$$O = f(Q)$$
$$Q' = f(Q, I) \qquad (2.1)$$

โดยที่ O คือ สัญญาณเอาท์พุท
I คือ สัญญาณอินพุท
Q คือ สัญญาณสเตทปัจจุบัน และ
Q' คือ สัญญาณสเตทถัดไป

ในทางตรงข้ามกรณีของ Mealy machine ดังแสดงในรูปที่ 2.1(b) และสมการที่ (2.2) สัญญาณเอาท์พุทเป็นฟังก์ชันของสเตทปัจจุบันและอินพุท เอาท์พุทสามารถเปลี่ยนแปลงค่าได้หากมีการเปลี่ยนแปลงของอินพุท ไม่จำเป็นต้องรอเปลี่ยนในสเตทถัดไป ข้อดีคือในงานบางลักษณะจำนวนสเตทของ Mealy machine จะน้อยกว่าจำนวนสเตทของ Moore machine

Mealy machine

$$O = f(Q, I)$$
$$Q' = f(Q, I) \qquad (2.2)$$

ในงานวิจัยนี้ Moore machine ถูกเลือกใช้เนื่องจากนำไปประยุกต์ใช้กับการควบคุมการทำงานของวงจรดาต้าพาท (Datapath) สัญญาณเอาท์พุทที่ไปควบคุมการทำงานของวงจรดาต้าพาทต้องคงที่ตลอดจนกว่าวงจรดาต้าพาทส่วนที่เกี่ยวข้องทำงานเสร็จ

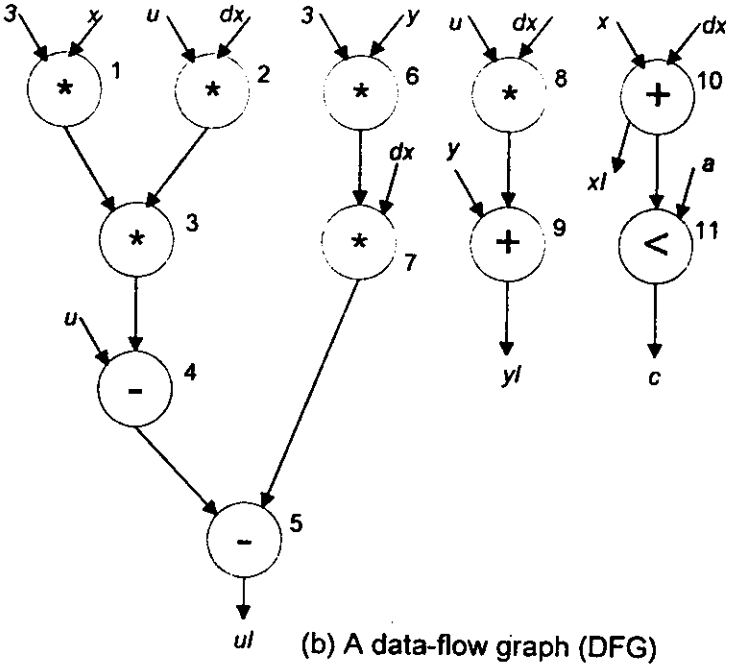
2.2 Data-Flow Graphs (DFGs)

DFGs (Data Flow Graphs) [Micheli94] เป็นกราฟตัวกลางหนึ่งซึ่งแสดงการไหลของข้อมูลผ่านตัวดำเนินการ (operator) ทางคณิตศาสตร์และลอจิก รูปที่ 2.2(b) แสดงตัวอย่างของ DFG อันหนึ่ง สัญลักษณ์ลูกศรใน DFG ถูกใช้แสดงเป็นข้อมูลอินพุทและเอาท์พุทของตัวดำเนินการซึ่งแทนด้วยวงกลมที่มีสัญลักษณ์การดำเนินการอยู่ภายใน การดำเนินการ (operation) หนึ่งการดำเนินการประกอบด้วย 3 ขั้นตอน ขั้นตอนแรกคือ การนำข้อมูลอินพุทมาพร้อมที่ด้านอินพุทของตัวดำเนินการ (fetching operands) ขั้นตอนที่สองคือ การคำนวณผลลัพธ์ของตัวดำเนินการ (execution) ขั้นตอนสุดท้ายคือ การนำผลลัพธ์ไปเก็บยังรีจิสเตอร์ที่ต้องการ (writing to destination registers)

DFGs แสดงโอเปอเรชันและความสัมพันธ์ของข้อมูล นิยมใช้ในอัลกอริทึมสำหรับการวิเคราะห์ และการปรับปรุงประสิทธิภาพของวงจรการประมวลผลสัญญาณดิจิตอลซึ่งประกอบด้วยเซทของฟังก์ชันคณิตศาสตร์ เช่น การบวก การคูณ การลบ เป็นต้น ของสัญญาณข้อมูล ตัวอย่างเช่น เซทของการคำนวณในรูปที่ 2.2(a) แสดงถึงการทำงานของวงจรแก้สมการอนุพันธ์อันหนึ่ง เซทดังกล่าวสามารถถูกแทนอย่างตรงไปตรงมาด้วย DFG ในรูปที่ 2.2(b)

xl = x + dx;
*ul = u - (3*x*u*dx) - (3*y*dx);*
*yl = y + u*dx;*
c = xl < a;

(a) A set of computations



(b) A data-flow graph (DFG)

รูปที่ 2.2 เซทของการคำนวณ (a) และ DFG (b) ของวงจรแก้สมการอนุพันธ์อันหนึ่ง

นิยาม 2.1 DFG $G_d(V, E)$ **เป็นกราฟมีทิศทาง ประกอบด้วยเซทของโอเปอเรชัน** $V = \{v_i; i = 1, 2, 3, ..., n_{ops}\}$ **และเซทของลูกศรแสดงทิศทางของข้อมูล** $E = \{(v_i, v_j): i, j = 1, 2, 3, ..., n_{ops}\}$ **โดยที่** n_{ops} **เป็นจำนวนโอเปอเรชัน**

DFGs เป็นกราฟมีทิศทาง (Directed graph) ถูกนิยามไว้ดังนิยาม 2.1 โอเปอเรชันหนึ่งต้องการโอเปอแรนด์หนึ่งตัวหรือมากกว่า และให้ผลลัพธ์หนึ่งผลลัพธ์หรือมากกว่า ตัวอย่างเช่น การบวกจำนวนสองจำนวนให้ผลลัพธ์เป็นผลบวกและสัญญาณโอเวอร์โฟล์ว เป็นต้น ลูกศรแสดงการโอนย้ายข้อมูลจากโอเปอเรชันหนึ่งไปยังโอเปอเรชันอื่น DFGs สามารถถูกเพิ่มเติมด้วยความสัมพันธ์ชนิดอื่น ตัวอย่างเช่น เมื่อโอเปอเรชัน 2 โอเปอเรชันใช้ทรัพยากร (วงจรฟังก์ชันคณิตศาสตร์ เช่น วงจรบวก วงจรคูณ เป็นต้น) ร่วมกันอยู่ จำเป็นต้องเพิ่มลูกศรเข้าไปเพื่อแสดงให้เห็นว่า โอเปอเรชันที่ตามหลังต้องรอจนกว่าโอเปอเรชันที่นำหน้าทำงานเสร็จก่อนจึงจะเริ่มทำงานได้ ไม่เช่นนั้นข้อมูลหรือผลลัพธ์ที่ได้จากการคำนวณจะเกิดการทับซ้อนกัน ผลลัพธ์นี้เป็นข้อมูลที่ผิดพลาดนำไปใช้งานต่อไม่ได้

2.3 การสังเคราะห์วงจรที่ระดับสูง (High-Level Synthesis)

การสังเคราะห์วงจรที่ระดับสูง (High-level synthesis) มีศักยภาพสูงในการออกแบบระบบ VLSI ให้มีประสิทธิภาพสูงสุด และมีเปอเซนต์การปรับปรุงให้ระบบมีประสิทธิภาพดีขึ้นมากที่สุดเมื่อเปรียบเทียบกับการสังเคราะห์วงจรที่ระดับลอจิกเกทและระดับทรานซิสเตอร์ งานการสังเคราะห์ (synthesis task) หลักที่ระดับสูงนี้มีดังนี้

- Resource allocation เป็นการคำนวณหาจำนวนทรัพยากร เช่น รีจิสเตอร์ บัส และวงจรฟังก์ชันคณิตศาสตร์ ที่ต้องถูกใช้ในการสร้างวงจร จุดประสงค์หลักของงานนี้คือ การลดจำนวนทรัพยากรเพื่อให้ได้วงจรที่มีขนาดเล็กที่สุดเท่าที่ทำได้ อาจมีการใช้ทรัพยากรร่วมกัน (Resource sharing) ของโอเปอเรชันประเภทเดียวกันแต่เกิดต่างเวลา
- Operation scheduling เป็นการจัดลำดับการทำงานของโอเปอเรชันทั้งหมดเพื่อให้วงจรทำงานเสร็จเร็วที่สุดเท่าที่ทำได้ งานนี้ทำการแบ่ง DFG ออกเป็นช่วงเวลาที่เรียกว่า control steps แต่ละ control step มักจะหมายถึงหนึ่งคาบของสัญญาณนาฬิกาซึ่งมักเรียกว่า clock-cycle หรือหนึ่งรอบของสัญญาณนาฬิกา โดยทั่วไปแต่ละโอเปอเรชันมักจะถูกจัดให้ทำงานเสร็จภายในหนึ่ง control step นั่นหมายความว่า คาบสัญญาณนาฬิกาจะขึ้นกับ control step ที่ยาวที่สุด หรือโอเปอเรชันที่ทำงานช้าที่สุด
- Resource binding เป็นการกำหนดตัวแปรต่าง ๆ ไปยังหน่วยความจำ และกำหนดโอเปอเรชันต่าง ๆ ไปยังหน่วยฟังก์ชัน พร้อมกับตรวจสอบเพื่อความถูกต้องว่าบัสข้อมูลได้ถูกกำหนดให้กับแต่ละการโอนย้ายข้อมูลระหว่างหน่วยความจำและหน่วยฟังก์ชัน

- Clock period selection เป็นงานหลักที่สำคัญอีกงานหนึ่ง การเลือกคาบสัญญาณนาฬิกาที่เหมาะสมส่งผลทำให้วงจรทำงานได้เร็ว โดยทั่วไปคาบสัญญาณนาฬิกาจะถูกเลือกจาก control step ที่ยาวที่สุด หรือโอเปอเรชันที่ทำงานช้าที่สุด วิธีนี้เป็นวิธีที่ง่ายที่สุดแต่วงจรทำงานได้ช้าที่สุด

ในการออกแบบวงจรรวมโดยใช้ซอฟแวร์ช่วยออกแบบ CAD (Computer Aided Design) นักออกแบบมักกำหนดคุณลักษณะที่ต้องการในวงจรที่เรียกว่า Design constraint เพื่อให้ CAD สังเคราะห์วงจรให้ใกล้เคียงกับคุณลักษณะที่ต้องการ งานการสังเคราะห์ทั้ง 4 งานข้างบนอาจจะถูกดำเนินการไปพร้อม ๆ กัน เพื่อให้ได้วงจรที่มีประสิทธิภาพมากที่สุด แต่การทำเช่นนี้ต้องใช้เวลาและหน่วยความจำในการสังเคราะห์วงจรได้ และบ่อยครั้งก็ไม่สามารถสังเคราะห์วงจรที่มีขนาดใหญ่ได้ ดังนั้นอัลกอริทึม หรือระเบียบวิธีที่ใช้ในการสังเคราะห์วงจรที่มีประสิทธิภาพนั้น นอกจากจะต้องสามารถสังเคราะห์วงจรที่มีประสิทธิภาพแล้ว จะต้องมีความง่ายไม่ซับซ้อน และสามารถสร้างทางเลือกการออกแบบ (Design space exploration) ได้หลายทางเลือกด้วย

2.4 ปัญหาการสังเคราะห์ FSMs จาก DFGs

วิธีการออกแบบวงจรรวมแบบซิงโครนัส (Synchronous Design) เป็นวิธีการออกแบบที่นักออกแบบนิยมนำมาใช้ในการออกแบบและสร้างวงจรรวม VLSI (Very Large Scale Integrated Circuits) วงจรที่ได้จากการออกแบบวงจรรวมแบบซิงโครนัสถูกเรียกว่า วงจรซิงโครนัส (Synchronous Circuits) การออกแบบวงจรซิงโครนัสทำได้ง่าย เนื่องจากการทำงานของวงจรถูกกำหนดจังหวะโดยสัญญาณนาฬิกา (clock) สัญญาณนาฬิกาเป็นตัวบอกว่าข้อมูลที่ได้จากการประมวลผลเป็นข้อมูลที่อยู่ในสภาวะเสถียรแล้ว พร้อมที่จะถูกนำไปประมวลผลต่อได้อย่างถูกต้อง นั่นคือสัญญาณข้อมูลจะมีนัยสำคัญเฉพาะขอบขาขึ้น (rising-edge) หรือขอบขาลง (falling-edge) เท่านั้น ในระหว่างคาบของสัญญาณนาฬิกาสัญญาณข้อมูลสามารถแกว่งอย่างไรก็ได้โดยไม่มีผลทำให้การประมวลผลของวงจรผิดพลาด ด้วยเหตุนี้การออกแบบวงจรซิงโครนัสจึงง่าย ในระหว่างการออกแบบนักออกแบบไม่จำเป็นต้องคอยหลีกเลี่ยงสัญญาณฮาซาร์ด (hazards) หรือ กริทซ์ (glitches) ซึ่งเกิดขึ้นจากการมาถึงของสัญญาณอินพุทขององค์ประกอบของวงจร (ลอจิกเกท เช่น AND-gate OR-gate เป็นต้น) ที่ไม่พร้อมกัน นอกจากการออกแบบวงจรซิงโครนัสจะง่ายดังที่กล่าวข้างต้นแล้ว ยังมีซอฟท์แวร์ช่วยออกแบบที่มีประสิทธิภาพมากมาย การออกแบบวงจรขนาดใหญ่มากจึงทำได้ง่ายและรวดเร็ว แต่ความเร็วของวงจรซิงโครนัสถูกจำกัดด้วยคุณสมบัติของตัวมันเอง ความถี่สัญญาณนาฬิกาเป็นตัวแสดงถึงความเร็วของวงจรซิงโครนัส ความเร็วสูงสุดของวงจรหรือความถี่สูงสุดของสัญญาณนาฬิกาถูกจำกัดด้วยเส้นทางเดินของสัญญาณที่ยาวที่สุด (longest delay

path or worst-case delay path) ซึ่งเส้นทางเดินดังกล่าวมักไม่ใช่เส้นทางเดินที่ถูกใช้บ่อย แต่ความยาว ของคาบสัญญาณนาฬิกาต้องครอบคลุมเส้นทางเดินดังกล่าวเพื่อเป็นการรับประกันว่าวงจรทำงานถูกต้อง

วงจรควบคุม FSM (Finite State Machine) มักถูกใช้เป็นของวงจรควบคุมสำหรับวงจรซิงโครนัส รูปที่ 2.3 แสดงการส่งข้อมูลจาก รีจิสเตอร์ Reg A ไปยัง รีจิสเตอร์ Reg B โดยผ่านวงจรประมวลผลทางคณิตศาสตร์และลอจิก (combinational circuits) กระบวนการดังกล่าวถูกควบคุมโดย FSM สัญญาณนาฬิกา (clock) ถูกใช้ในการกำหนดจังหวะการทำงานของ FSM และรีจิสเตอร์ เมื่อมีสัญญาณข้อมูลพร้อมที่อินพุทของ Reg A สัญญาณ en_A จะถูกสร้างมาเพื่อให้ Reg A นำข้อมูลมาเก็บที่เอาท์พุทของมันเมื่อมีสัญญาณนาฬิกา สัญญาณดังกล่าวจะถูกประมวลผลโดยวงจรประมวลผลทางคณิตศาสตร์และลอจิก แล้วส่งไปพร้อมอยู่ที่อินพุทของ Reg B จากนั้นสัญญาณ en_B จะถูกสร้างมาเพื่อให้ Reg B นำข้อมูลมาเก็บที่เอาท์พุทของมันเมื่อมีสัญญาณนาฬิกาอีกครั้ง คาบเวลาของสัญญาณนาฬิกาที่สั้นที่สุดที่วงจรดังกล่าวสามารถทำงานได้อย่างถูกต้องคือ

$$T_{\min} \geq T_W + T_{CMB} + T_S \qquad (2.3)$$

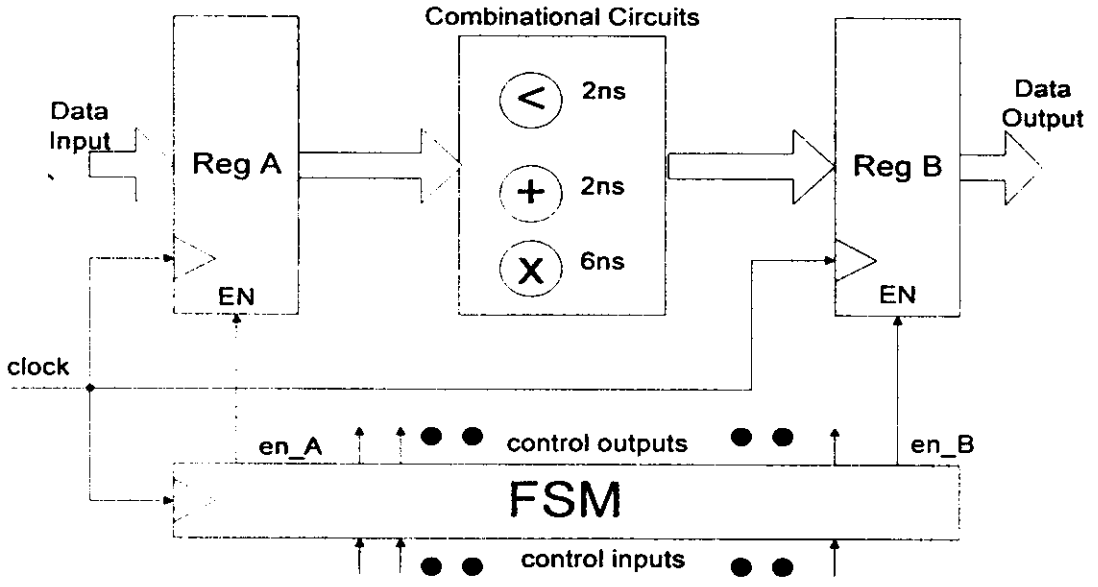
โดยที่

T_W เป็นเวลาที่มากที่สุดในการนำสัญญาณจากอินพุทมาเก็บที่เอาท์พุทของ Reg A เมื่อมีสัญญาณนาฬิกา

T_{CMB} เป็นเวลาที่นานที่สุดที่ใช้ในการประมวลผลของวงจรประมวลผลทางคณิตศาสตร์และลอจิก และ

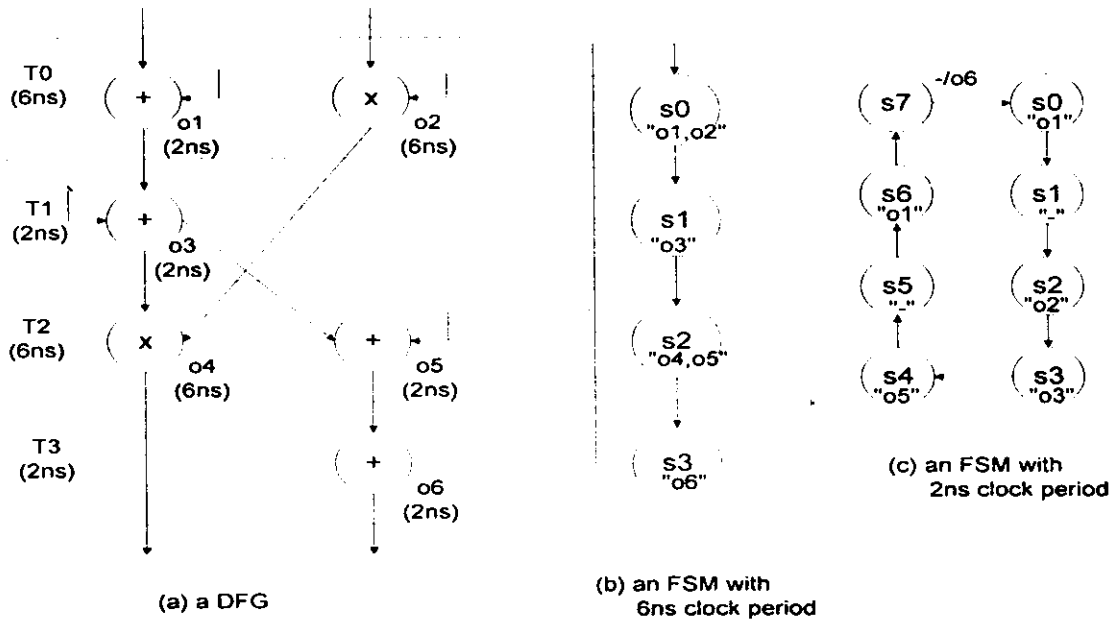
T_S เป็นเวลา setup time ของ Reg B

จากสมการที่ (2.3) พบว่าความถี่สูงสุดของสัญญาณนาฬิกาคือ $F_{\max} = 1/T_{\min}$ นั่นคือความเร็วสูงสุดของวงจรในรูปที่ 2.3 ถูกจำกัดที่ $F_{\max}$ นี้ ในรูปที่ 1 วงจรคูณใช้เวลานานที่สุด ดังนั้น $F_{\max}$ จึงถูกจำกัดด้วยวงจรคูณดังกล่าว ในการใช้งานจริงของวงจรนี้อาจจะมีการใช้การคูณน้อยมาก ตัวอย่างเช่น ในการบวกอย่างเดียว 10 ครั้ง วงจรนี้ต้องใช้เวลาถึง 60ns ในความเป็นจริงแล้ววงจรดังกล่าวใช้เวลาเพียง 20ns หากว่ามันมีเพียงวงจรบวกอย่างเดียว จะเห็นได้ว่าความเร็วของวงจรแบบซิงโครนัสถูกจำกัดด้วยเส้นทางเดินของสัญญาณที่ยาวที่สุด ทั้ง ๆ ที่มันสามารถทำงานได้เร็วกว่านี้



รูปที่ 2.3 **การโอนถ่ายข้อมูล** (Data Transfer) **ระหว่างรีจิสเตอร์**

ในการออกแบบวงจรซิงโครนัส โอเปอเรชันต่าง ๆ จะถูกกำหนดให้ถูกกระทำภายในขั้นเวลา (time step) ดังแสดงในรูปที่ 2.4 เหตุการณ์ในหนึ่งขั้นเวลาถูกควบคุมหนึ่งสเตท (state) ของ FSM ดังนั้น DFG ในรูปที่ 2.4(a) ถูกควบคุมโดย FSM ในรูปที่ 2.4(b) จากรูปพบว่าคาบเวลาของสัญญาณนาฬิกาของ FSM ในรูปที่ 2.4(b) ต้องมีค่าเท่ากับ 6ns ซึ่งเป็นขั้นเวลาที่นานที่สุด นั่นคือหนึ่งรอบการทำงาน (cycle time) ของ FSM ในรูปที่ 2.4(b) ใช้เวลาถึง 24ns แต่จากการสังเกต DFG แล้วจะพบว่า มันสามารถทำงานได้เร็วกว่านี้ กล่าวคือ cycle time ของ DFG นี้น้อยกว่า 24ns ดังแสดงในรูปที่ 2.4(c) FSM ซึ่งสามารถควบคุม DFG ในรูปที่ 2.4(a) ได้ถูกต้องเช่นเดียวกันมี cycle time เพียง 16ns เท่านั้น แต่อย่างไรก็ตามจำนวนสเตทของ FSM ในรูปที่ 2.4(c) มีมากเป็น สองเท่าของจำนวนสเตท FSM ในรูปที่ 2.4(b) การเพิ่มจำนวนสเตทมาก ๆ เพื่อเพิ่มความเร็วของวงจรเป็นวิธีที่มีผลข้างเคียงมาก เนื่องจากจะเกิดปัญหาใช้เวลานานและยากต่อการสังเคราะห์วงจร นอกจากนี้วงจรที่ได้มักมีขนาดใหญ่และใช้กำลังไฟมากขึ้น



รูปที่ 2.4 Data Flow Graph (DFG) และ FSMs ที่สังเคราะห์ได้

โครงการวิจัยนี้นี้มีจุดประสงค์เพื่อแก้ปัญหาข้อจำกัดของวงจรซิงโครนัสที่ได้อธิบายในข้อความข้างต้น ความเร็วของวงจรที่ได้ต้องไม่ถูกจำกัดโดยเส้นทางเดินของสัญญาณที่ยาวที่สุด โครงการวิจัยนี้จะนำเสนอวิธีการสังเคราะห์ FSM ที่ไม่ถูกจำกัดโดยเส้นทางเดินของสัญญาณที่ยาวที่สุดจาก DFG วิธีการสังเคราะห์จะเป็นลำดับขั้นตอน (systematic method) ซึ่งสามารถนำไปพัฒนาเป็นซอฟท์แวร์ช่วยออกแบบได้ FSM ที่ได้จะต้องมีประสิทธิภาพดีที่สุด กล่าวคือ มีจำนวนสเตทน้อย เพื่อที่จะให้การสังเคราะห์วงจรทำได้ง่าย วงจรควบคุมที่ได้มีขนาดเล็ก และมี cycle time ที่น้อย เพื่อที่จะได้วงจรควบคุมที่มีความเร็วสูง

2.5 งานวิจัยที่เกี่ยวข้อง

งานหลักของโครงการวิจัยนี้ คือ การคำนวณหาความถี่ของสัญญาณนาฬิกาที่เหมาะสมที่สุด งานวิจัยหลายงานเช่น [Jung02] [Lange01] [Bhattach98] [Juan96] เป็นต้น ทำการคำนวณหาคาบเวลาของสัญญาณนาฬิกาไปพร้อม ๆ กับการทำ resource-sharing เพื่อที่จะหาคาบเวลาของสัญญาณนาฬิกาที่เหมาะสม แต่วิธีการเหล่าต้องใช้เวลาและหน่วยความจำในการสังเคราะห์มาก และอัลกอริทึมมีความซับซ้อน

งานวิจัย [Chang96] ได้เสนอวิธีการเลือกค่าคาบสัญญาณนาฬิกาที่เหมาะสม โดยการลดสแลค (slack) ให้เหลือน้อยที่สุด สแลคในที่นี้คือ ช่วงเวลาว่าง (idle time) หลังจากที่โอเปอเรชัน

ทำงานเสร็จก่อนที่ขอบถัดไปของสัญญาณนาฬิกาจะมาถึง ตัวอย่างเช่น สมมติโอเปอเรชัน O_1 ใช้เวลาในการทำงาน 7 นาโนวินาที และ O_2 ใช้เวลา 10 นาโนวินาที ทำงานต่อเนื่องกัน ถ้าหากเลือกคาบสัญญาณนาฬิกาเป็น 10 นาโนวินาที จะเกิดสแลค 3 นาโนวินาที ในโอเปอเรชัน O_1 และสแลคเป็นศูนย์ในกรณีของ O_2 จะเห็นว่าวงจรทำงานช้ากว่าที่ควรจะเป็นไป 3 นาโนวินาที ในอีกกรณีหนึ่ง ถ้าหากเลือกคาบสัญญาณนาฬิกาเป็น 1 นาโนวินาที สแลคจะเป็นศูนย์ทั้งสองโอเปอเรชัน วงจรสามารถทำงานได้ตามเวลาที่ควรจะเป็นคือ 17 นาโนวินาที แต่อย่างไรก็ตาม วงจร FSM สำหรับกรณีหลังมีขนาดใหญ่กว่ากรณีแรกมาก เพราะต้องมีจำนวนสเตทถึง 17 สเตท ในขณะที่กรณีแรกต้องการเพียง 2 สเตท การสังเคราะห์เพื่อให้ได้วงจรลอจิกสำหรับกรณีหลังก็ยากกว่า นอกจากนี้วงจร FSM ในกรณีหลังใช้กำลังไฟมากกว่าเนื่องจากมีการเปลี่ยนแปลงของสัญญาณนาฬิกาทุก ๆ 1 นาโนวินาที วิธีการของ [Chang96] นี้จะคำนวณหาคาบสัญญาณนาฬิกาที่ทำให้เกิดสแลคน้อยที่สุด วิธีการนี้ทำได้ในเวลาอันรวดเร็วเนื่องจากสามารถคำนวณได้โดยไม่จำเป็นต้องรู้โครงสร้างและพฤติกรรมของวงจร มีการใช้เพียงข้อมูลจำนวนของโอเปอเรชันและเวลาการทำงานของโอเปอเรชันเท่านั้น แต่อย่างไรก็ตาม ดังที่ได้ยกตัวอย่างไปแล้วนั้น จะสังเกตเห็นว่าคาบสัญญาณนาฬิกาต้องเล็กมาก ๆ จึงจะทำให้สแลคมีค่าที่น้อยที่สุด สิ่งเหล่านี้ส่งผลให้วงจร FSM มีขนาดใหญ่และใช้กำลังไฟมาก

งานวิจัย [Blythe00] ได้นำเสนอระเบียบวิธีที่มีประสิทธิภาพขึ้น โดยมีการวิเคราะห์ทางเลือก (Design space exploration) ระเบียบวิธีจะสร้างจุดออกแบบ (design point) หลาย ๆ จุด จุดออกแบบ D(S, A) ใด ๆ เป็นโคออร์ดิเนเตอร์ของความเร็ว (S) และพื้นที่ (A) ของวงจร ซึ่งแต่ละจุดมีคาบสัญญาณนาฬิกาที่เหมาะสมที่สุดแล้ว จุดออกแบบเหล่านี้เรียกว่า จุดพาริโต (Pareto point) [Micheli94] ซึ่งหมายถึงจุดออกแบบที่ไม่มีจุดออกแบบอื่นมีเวลาการทำงาน (latency) ต่ำกว่าอีกแล้วที่มีพื้นที่ (area) เท่ากัน หรือจุดออกแบบที่ไม่มีจุดออกแบบอื่นมีพื้นที่น้อยกว่าอีกแล้วที่มี latency เท่ากัน ระเบียบวิธีนี้ทางเลือกที่เหมาะสมขึ้นกับลักษณะงาน แต่อย่างไรก็ตามระเบียบวิธีคำนวณหาค่าคาบสัญญาณนาฬิกาก่อนและไม่ขึ้นกับการ Scheduling หรือโครงสร้างของ DFG จึงอาจทำให้จุดออกแบบที่เหมาะสมทั้งหมดไม่ถูกสร้างขึ้นมา