เนื้อหา

State Machine and Systems Design

Register, Register Operation และ Register Transfer Notation

ในการออกแบบระดับ System นั้น Register และ Operation ของมัน จะใช้เป็นส่วนประกอบ และ Operation พื้นฐานของระบบ

- Register n บิต กำหนดให้เป็น Rn-1, … , R1, R0 โดย R0 เป็น LSB
- Register จะมีชื่อเป็นอักษรตัวใหญ่ ซึ่งมักเป็นอักษรตัวเดียว (A, B, หรือ C) หรือเป็นชื่อที่บอกถึงการทำงานของมัน เช่น AC คือ Accumulator, MD คือ Memory Data, หรือ MBR หมายถึง Memory Buffer Register
- X, Y จะเป็น I/P และ Z เป็น O/P
- Register Transfer Notation (RTN) ดังตารางที่ 1 จะใช้ในการอธิบายการทำงานของระบบ โดยใช้ตัวอักษรที่เหลือ

ตารางที่ 1 Operation ต่างๆ ของ Register Transfer Notation

Function	Symbolic Representation
Register Transfer Operation	
Copy (Transfer) Register B Contents to Register A	A ← B
Load word X into Register A	A ← X
Reset (clear) register A	A ← 0
Set (All bits of) Register A	A ← 1s
Increment a register's contents	Inc (A)
Decrement a register's contents	Dec (A)
Shift Register Operation	
Control S1, S0 = Hold	0, 0
Shift right	0, 1
Shift Left	1, 0
Parallel Load	1, 1

Function	Symbolic Representation
Register Arithmetic Operations	
Addition	C ← A + B
Subtraction	C ← A – B
Multiplication	C ← A × B
Division	C ← A / B
Register Logical Operations	
AND	C ← A ∧ B
OR	C ← A ∨ B
XOR	C ← A XOR B
Complement	C ← A'

One-Dimension Iterative Process

การทำงานของระบบ Digital Logic จำนวนมากที่ถูกจัดการโดยวงจร Sequential Logic ที่เรียกว่า Controller ในหลายๆ กรณี Controller (และ Control Process) สามารถจะแสดงการทำงานโดย ASM Model

Iterative Process เป็นสิ่งที่ใช้ทั้ง Combinational และ Sequential โดยการทำงานของ Combination จะถูกกระทำซ้ำอยู่จำนวนหนึ่ง (Sequential) ตัวอย่างของ Iterative Process แบบ One-Dimension และ Two-Dimension คือวงจร Addition และวงจร Multiplication ของตัวเลขหลายบิต ตามลำดับ

Iterative Process นั้น สามารถสร้างขึ้นจาก 2 วิธี คือ ใช้วงจร Combination ซึ่งจะทำให้วงจรมีขนาดใหญ่ หรือใช้วงจร Sequential ซึ่งจะทำให้วงจรใช้เวลาเพิ่มขึ้น

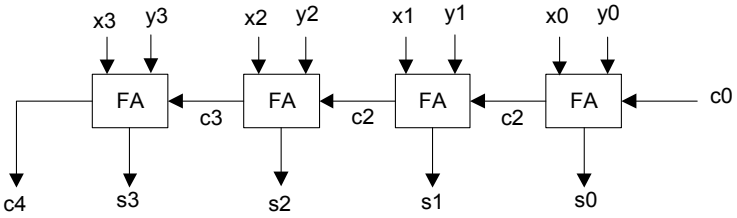
ตัวอย่างของ Iterative Process แบบ One-Dimension คือ

- การทำ 2s Complement ของตัวเลข n บิต
- การทำ Parity Check ของตัวเลข n บิต
- การเปรียบเทียบตัวเลข n บิต 2 จำนวน

Information Distribute in Space and Distribute in Time

<u>ตัวอย่างที่ 1</u>

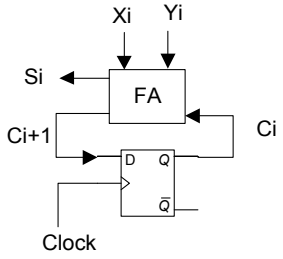
วงจร 4-bit parallel adder ดังรูปที่ 1 ซึ่งใช้วงจร 1-bit adder มาต่อพ่วงกัน 4 วงจร เป็น Process ที่มีการกระทำ Operation เดิม 4 ครั้ง ถือเป็น Iterative Process แบบ Distribute in Space



รูปที่ 1 Binary Adder ที่มี Information Distribute in Space

<u>ตัวอย่างที่ 2</u>

วงจร 4-bit adder สามารถทำจากวงจร Sequntial ดังรูปที่ 2 ซึ่งใช้ FA Module ที่มี I/P Ci, Xi, Yi และ O/P Si, Ci+1 ซึ่งสามารถทำการบวกเลข 4 บิตได้โดยทำ Operation เดิม 4 ครั้งเช่นกัน แต่ใช้เวลา 4 ช่วงสัญญาณนาฬิกา ซึ่งให้ O/P ออกมา 1 บิตต่อ 1 สัญญาณนาฬิกา ถือเป็น Iterative Process แบบ Distribute in Time



รูปที่ 2 Binary Adder ที่มี Information Distribute in Time

<u>ตัวอย่างที่ 3</u>

สามารถหาค่า 2s complement ของตัวเลข binary ได้จากการทำ 1s complement แล้วบวกด้วย 1 ดังนี้

$$\begin{aligned} Y = \ & 00001010 \\ & 11110101 \quad = \quad Y' \text{ (1s complement)} \\ + \ & \underline{\qquad 1} \\ & 11110110 \quad = \quad Y'' \text{ (2s complement)} \end{aligned}$$

ค่า 1s complement y' ของตัวเลข 1-bit สามารถหาได้จากการใช้ XOR ที่มี Control I/P w และ data I/P y

ดัง Truth Table ในตารางที่ 2

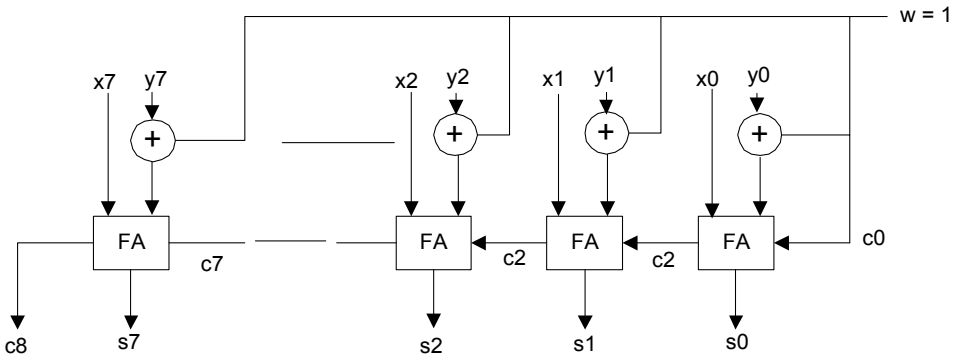
ตารางที่ 2 XOR Truth Table

W	Y	W⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

2s complement ของตัวเลข 8-bit Y หาได้จากการ complement yi ทุกบิตโดยใช้ XOR ที่ control w = 1 เพื่อหา $Y' = y_7'y_6'\ldots y_0'$ จากนั้นบวก $0 + Y' + 1$ ดังตัวอย่าง

$$\begin{aligned} Y \quad = \quad & 00001010 \\ & \underline{\oplus 11111111} \\ & 11110101 \quad = \quad Y' \text{ (1s complement)} \\ & + 00000000 \\ & \underline{+ \qquad 1} \\ = \quad & 11110110 \quad = \quad Y'' \text{ (2s complement)} \end{aligned}$$

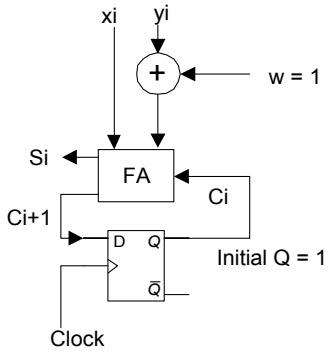
เราจึงสามารถสร้างวงจร 2s complementer แบบ combination ได้โดยการใช้ Module ของ FA 8 Module ดังรูปที่ 3 XOR จะสร้าง Y' และ FA แต่ละตัวจะบวก 0 เข้ากับ yi และ Carry-in โดยค่า carry-in ของ LSB เป็น 1 วงจร
มีการกระทำซ้ำ 8 ครั้ง จะเป็น iterative process ที่ distribute in space



รูปที่ 5 วงจร 2s complementer ในแบบ distribute in space

ตัวอย่างที่ 4

จากรูปที่ 3 จะเห็นว่าวงจรสามารถทำได้จากการต่อ module เดียวกัน cascade กัน 8 module ถ้าจะสร้าวงจรเป็นแบบอนุกรมสามารถทำได้จากการใช้วงจรเพียง 1 Module ร่วมกับ D Flip-Flop ดังรูปที่ 4โดยให้ initial State ของ D flip-flop เป็น1วงจรจะสามารถหาค่า 2 complement ได้ในเวลา 8 ช่วงของสัญญาณนาฬิกา รูปแบบนี้จะเป็น iterative process แบบ distribute in time



รูปที่ 4 วงจร 2s complementer ในแบบ distribute in time

Sequential Machines with Rudimentary Control
Design and Implementation of an Even-Parity Check Machine

ตัวอย่างที่ 5

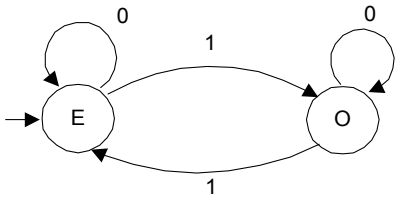
Step 1 : Problem Statement : ออกแบบ Parity Check Machine ที่มี Specification ดังนี้

a. เครื่องจะอ่านข้อมูล 8 บิต w = ABCDEFGP ประกอบด้วย Data 7 บิต A, B, C, …, G และ Parity bit P

b. เครื่องจะสร้าง O/P Z ซึ่งถ้า Z = 1 แสดงว่า Even-Parity Error (w มีจำนวนบิตของ 1 เป็นจำนวนคี่)

และถ้า Z = 0 แสดงว่า ไม่มี Error (w มีจำนวนบิตของ 1 เป็นจำนวนคู่)

Step 2 : Conceptualization : เขียน State Diagram ดังรูปที่ 5



รูปที่ 5 State Diagram ตัวอย่างที่ 5

เขียน State Table ได้ดังตารางที่ 3

ตารางที่ 3 State Table ตัวอย่างที่ 5

Present State Y	Next State Y_N Input 0 1	Output Z Input 0 1
E	E O	0 1
O	O E	1 0

Step 3 : Solution / Simplification : กำหนดให้ E = 0, O = 1 ..ใช้ State Table สร้าง NS Map และ O/P ได้ดังรูปที่ 6

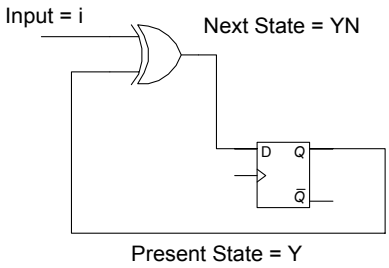
รูปที่ 6 NS Map และ O/P Map ตัวอย่างที่ 5

จะได้ NS และ O/P Function ดังนี้

$$YN = Y\bar{I} + \bar{Y}I = Y \oplus I$$

$$Z = Y\bar{I} + \bar{Y}I = Y \oplus I$$

Step 4 : Realization : สามารถสร้าง Module ของ Even-Parity Check เป็นวงจร Sequential ได้โดยใช้ clock, วงจร Combination NS Function และ O/P Function, และ D flip-flop 1 ตัว ดังรูปที่ 7

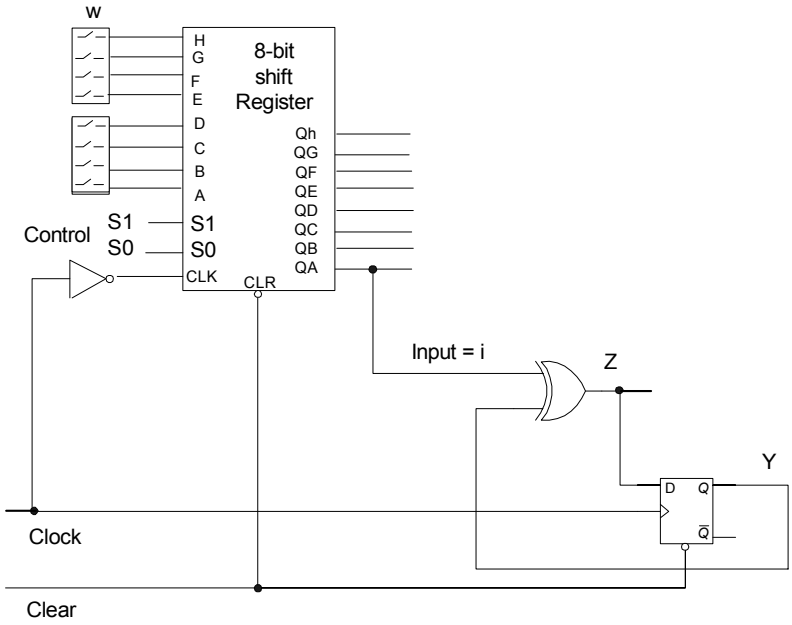


รูปที่ 7 FSM Module ของวงจร Even Parity Check

8-bit Even-Parity Check Machine ซึ่งใช้กลไกของ Rudimentary Control สามารถสร้างขึ้นจาก

- System Clock
- 8-bit Parallel Load Shift Register
- FSM Module ของวงจร Even Parity Check

ดังรูปที่ 8



รูปที่ 8 Even-Parity Check Machine

Design and Implementation of a Serial 2s Complementer

ตัวอย่างที่ 6

Step 1 : Problem Statement : ออกแบบ 2s Complementer Machine ที่มี Specification ดังนี้

a. เครื่องทำการอ่านข้อมูล 8 บิต w = ABCDEFGH

b. เครื่องสร้าง O/P Z ที่เป็น 2s Complement ของ w

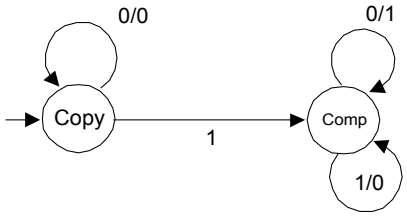
Step 2 : Conceptualization : เราจะใช้วิธี copy-complement method ในการหาค่า 2s complement ดังนี้

- Copy ทุกบิตจาก LSB จนถึงบิตแรกที่มีค่าเป็น 1
- Complement บิตที่เหลือทั้งหมด

เช่น

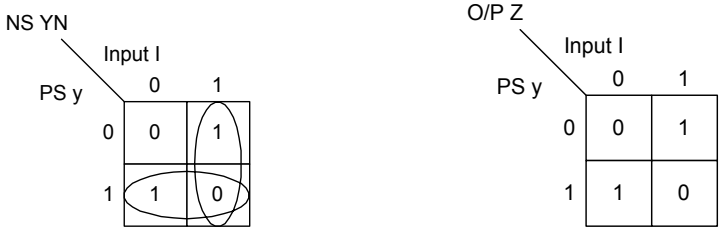
w	=	00001010		
		10		Copy ทุกบิตจาก LSB จนถึงบิตแรกที่เป็น 1
		111101		Complement ทุกบิตที่เหลือ
Z	=	11110110	=	2s complement ของ w

เขียนเป็น State Diagram ได้ดังรูปที่ 9



รูปที่ 9 State Diagram ตัวอย่างที่ 6

Step 3 : Solution/Simplification : ให้ copy state = 0 และ complement state = 1 สามารถเขียน NS Map และ O/P Map ได้ดังรูปที่ 10

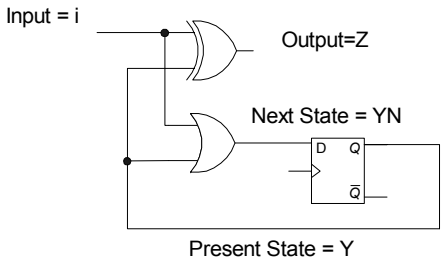


รูปที่ 10 Next State Map และ Output Map ตัวอย่างที่ 6

จะได้ NS Function และ O/P Function ดังนี้

$$YN = I + Y$$
$$Z = I\overline{Y} + \overline{Y}I = I \oplus Y$$

Step 4 : Realization : FSM Module ของ 2s complementer สามารถสร้างเป็นวงจร



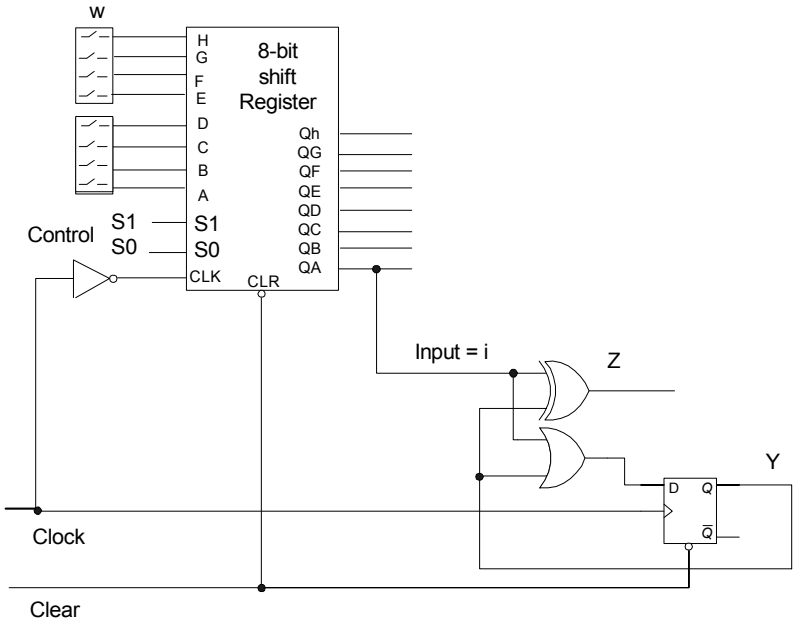
Sequential จาก clock, NS และ O/P Function, และ D Flip-Flop ดังรูปที่ 11

รูปที่ 11 FSM Module ของวงจร 2s complementer

2s complement machine ที่ใช้กลไกของ Rudimentary control สามารถสร้างขึ้นจาก

- System Clock
- 8-bit Parallel Load Shift Register
- FSM Module ของ 2s complementer

ดังรูปที่ 12



รูปที่ 12 serial 2s complementer machine

ในตอนต้น w จะถูก Load เข้าไปยัง 8-bit Shift Register จากนั้น w จะนำไปหาค่า 2s complement ทีละ 1 บิต Z ที่ได้ แต่ละบิตจะถูก Shift เข้าไปใน Register ในขณะที่ w ถูก Shift ออกไป ผลลัพธ์ที่ได้จะเป็น 2s complement ของ w เก็บอยู่ที่ Register เมื่อเวลาผ่านไป 8 สัญญาณนาฬิกา

Design and Implementation of a Serial Addition machine

ตัวอย่างที่ 7

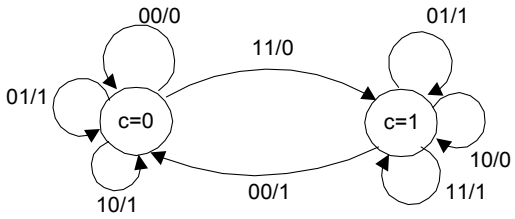
Step 1 : Problem Statement : ออกแบบ Serial Addition machine ของตัวเล 4 บิต X = $x_3x_2x_1x_0$ และ Y = $y_3y_2y_1y_0$ และ Carry-in C0

Step 2 : Conceptualization : การบวกเลขแต่ละบิต จะเป็นการบวก x_i, y_i และ c_i เข้าด้วยกัน ได้ผลลัพธ์เป็น s_i และ c_{i+1} ซึ่ง Ci+1 จะนำไปเป็น ci ใน Stage ถัดไป เราจึงทำการออกแบบโดยให้ Ci เป็น Present State และ Ci+1 เป็น Next State ดัง State table ตารางที่ 4

ตารางที่ 4 State table ตัวอย่างที่ 7

Present State	Next State C Inputs xy				Output S Input xy			
	00	01	11	10	00	01	11	10
0	0	0	1	0	0	1	0	1
1	0	1	1	1	1	0	1	0

State Diagra



รูปที่ 13 State Diagram ของตัวอย่างที่ 7

Step 3 : Solution / Simplification : เขียน NS และ O/P Map ได้ดังรูปที่ 14

CN	xy 00	01	11	10
c 0	0	0	1	0
1	0	1	1	1

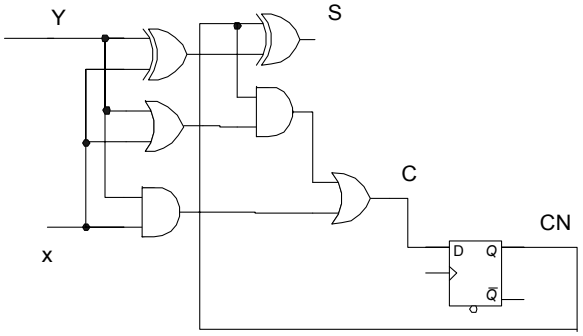
S	xy 00	01	11	10
c 0	0	1	0	0
1	1	0	1	0

รูปที่ 14 NS และ O/P Map ของตัวอย่างที่ 7

หา NS และ O/P Function ได้ดังนี้

$$CN = x\,y + c\,(x + y)$$
$$S = c \oplus x \oplus y$$

Step 4 : Realization : FSM Module ของ Serial Addition สามารถสร้างได้ดังรูปที่ 15

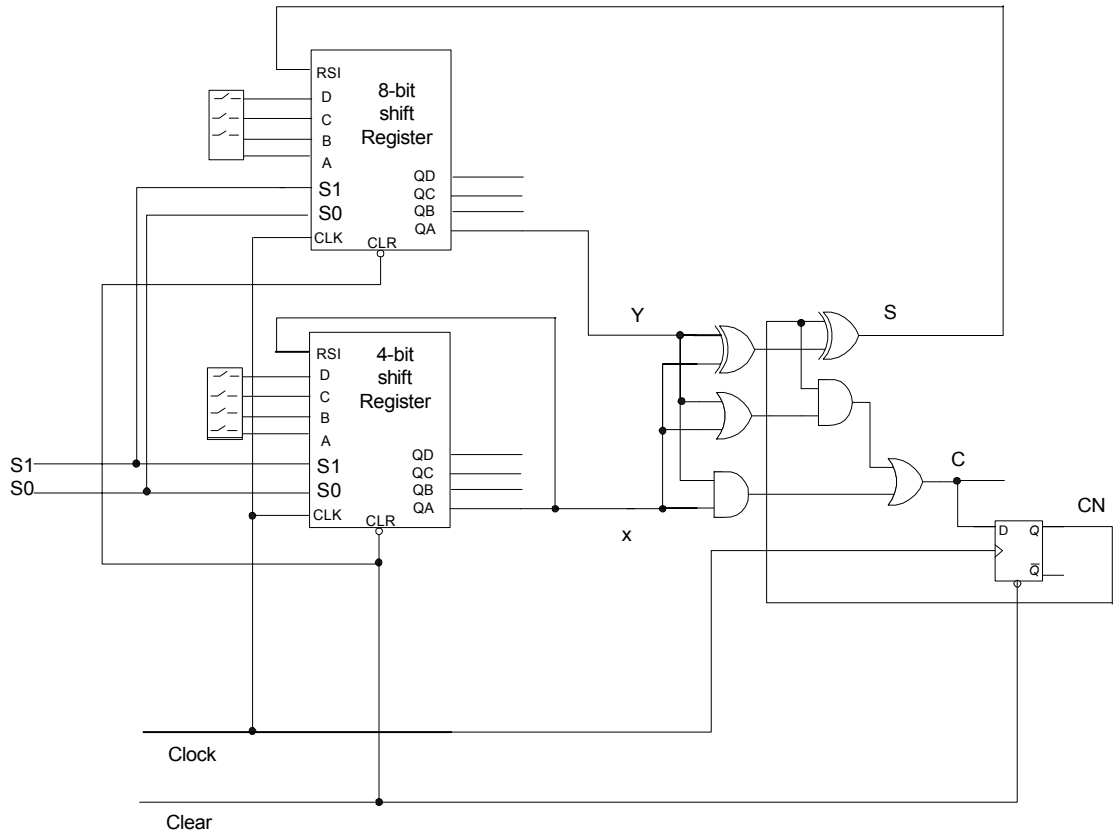


รูปที่ 15 FSM Module ของ serial Addition

เราสามารถสร้าง 4-bit Serial Addition Machine ที่ใช้กลไกของ Rudimentary Control ได้โดยใช้

- Parallel Load Shift Register 2 วงจร
- FSM Module ของ Serial Addition
- System Clock

ดังรูปที่ 16



รูปที่ 16 Serial Addition Machine ที่ใช้ Rudimentary Control

การบวกแบบอนุกรม จะสร้างค่า Sum Si และ Next State Carry Ci+1 โดยใช้ Process Algorithm ดังนี้

I/P xi, yi

Output $Si = ci \oplus xi \oplus yi$

Next State Carry Ci +1 = xi yi + ci (xi + yi) , (I = 0,1,2,3)

ขั้นตอนการควบคุมมีดังนี้

Step a :Initialize count (index I)
Set Registers' Control to hold
Key X, Y to I/P of the Registers
Step b : Set Registers' control to load
Parallel Load data into X and Y Register
Step c : Use Process Algorithm to produce Si and Ci+1
Set Registers' control to Shift right
Shift right Register X, Y
Increment I
If I < 4 then repeat Step c
Else Step a

ถ้าเรากำหนดทุก Step เป็น State ของ Control Algorithm เราก็สามารถเขียน Algorithm ในรูปของ pseudo code ดังนี้

```
State a :    Initialize index ( I := 0)
             Set Registers' Control to hold ( S1, S0 := 0, 0)
State b :    Set Registers' control to load ( S1, S0 := 1, 1)
             Parallel Load data into  X and Y Registers
State c :    Use Process Algorithm  to produce  Si and Ci+1
             Set Registers' control to Shift right   ( S1, S0 := 0, 1)
             Shift right Register X, Y
             Increment I
                  If I < 4 then { goto c }  else { goto a }
```

Algorithmic State Machines and Systems-Level Design Procedure

Procedure ในการออกแบบระดับ System มีดังนี้

Step 1 : Problem Statement : วางโครงสร้าง Process เป็น 2 ส่วน (มักจะอยู่ในรูป Peseudocode) และบอก Specification ของระบบ

a. Process
b. Process Control Algorithm

Step 2 : Conceptualization : เขียนอธิบายวงจรในลักษณะของรูปภาพ โดยใน design tools ดังนี้

a. System Structure Diagram : เป็น block diagram ของระบบที่แบ่งระบบเป็น 2 ส่วนคือ
   (1) Process section (data path)
   (2) Control section (Controller, :ซึ่งประกอบด้วย Next-State Generator และ O/P Decoder)

b. Control Flow Diagram : แสดงการทำงานของ Process Control Algorithm ซึ่งจะแสดง
   (1) Control State และเงื่อนไขของการเปลี่ยน State
   (2) การสร้างสัญญาณเพื่อควบคุมส่วนของ Data Path

Step 3 : Solution / Simplification : เขียนรายละเอียดต่อไปนี้

a. Process Data path อยู่ในรูปของ Block Diagram หรือ Logic Diagram
b. Control Unit
   (1) Next State Generator ได้จาก NS Map หรือ Table
   (2) O/P Decoder ได้จาก Logic Equation ของสัญญาณควบคุมจาก Data Path

Step 4 : Realization : สร้างวงจรของ

a. Process Data Path

b. Control Unit
(1) NS Generator
(2) O/P Decoder

จะได้ ระบบขึ้นมาจากการรวมเอาส่วนของ Process Data Path กับ Control Unit เข้าด้วยกัน

Problem Statement and System Specification

Step 1 : Problem Statement :

a. Process : การบวกตัวเลข 4 บิต 2 จำนวน คือ X = x3x2x1x0 และ Y = y3y2y1y0 และ Carry-in C0 ซึ่งสามารถทำได้โดยการบวกแบบอนุกรม ดัง Process Algorithm ต่อไปนี้

Process Inputs	xi, yi	
Process Outputs	si = ci ⊕ xi ⊕ yi	(I = 0,1,2,3)
Next State Carry	Ci +1 =xi yi + ci (xi + yi) ,	

b. Process Algorithm : Procedure ของการ Initial, Starting, Loading, และ Processing [สร้าง Si และ Ci+1 (I = 0-3)] สามารถเขียนเป็น Process control Algorithm ดังนี้

State a : Initialize index (I := 0)
Set Registers' Control to hold (S1, S0 := 0, 0)
State b : Set Registers' control to load (S1, S0 := 1, 1)
Parallel Load data into X and Y Registers
State c : Use Process Algorithm to produce Si and Ci+1
Set Registers' control to Shift right (S1, S0 := 0, 1)
Shift right Register X, Y
Increment I
If I < 4 then { goto c } else { goto a }

ซึ่ง Process algorithm จะอยู่ใน Process Control Algorithm นี้ด้วย

System Specification : ออกแบบ ASM เพื่อทำการบวกตัวเลข 4 บิต 2 จำนวน X, Y และ carry-in C0 ดัง RTN นี้

$X \leftarrow X + Y + C0$

System Conceptualization

ถ้า Process Control สามารถเขียนเป็น Algorithm ได้ เราก็สามารถเขียนเป็น ASM Model ได้

ASM สามารถแบ่งออกเป็น 2 ระบบย่อย ดังนี้

1. Process Section (Data Path)ประกอบด้วยอุปกรณ์ที่ใช้เป็น Function เช่น Registers,Multiplexers, Demultiplexers, Adders, Encoders และ Decoders อุปกรณ์ใน

ส่วนของ Data Path ได้วางแบบไว้ให้ทำงานตาม Process ที่กำหนด ซึ่งมาจากสัญญาณควบคุมที่สร้างโดย Controller O/P Decoder

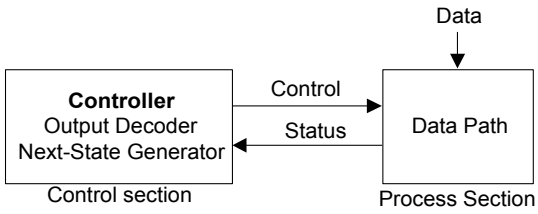
2. Control Section (Controller) ที่มีส่วนประกอบหลัก 2 ส่วน คือ

a. Next-State Generator (ทำงานตาม Control Algorithm) ที่ใช้ I/P และ PS มากำหนด NS

b. Output Decoder ซึ่งสร้างสัญญาณควบคุม Data Path ณ เวลาที่เหมาะสม เพื่อให้ Data Path กระทำตาม Function ของ Process ที่ต้องการ

Structure Diagram and Control Flow Diagram

Organization ของระบบ ASM สามารถบอก Concept ได้โดยใช้ Structure Diagram ซึ่งแบ่งเป็น Process Section (Controller ประกอบด้วย NS Generator และ O/P Decoder) ดังรูปที่ 17



รูปที่ 17 Structure Diagram ของ ASM

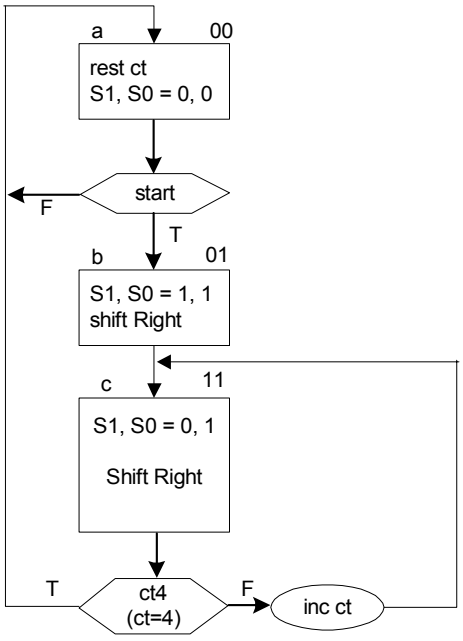
การทำงานของ Process Control Algorithm สามารถอธิบายโดยใช้ Control Flow Diagram ซึ่งจะแสดง State และ เงื่อนไขของการเปลี่ยน State และการสร้างสัญญาณควบคุมไปยังส่วนของ Data Path

Control Algorithm จะกำหนดทุก Sequence ของ State ที่เป็นไปได้ NS ของเครื่องจะถูกกำหนดโดย Control Algorithm, PS และ I/P ที่ State นั้นๆ Diagram ของการสร้าง Sequence ซึ่งกำหนดโดย Control Algorithm นี้ สามารถเขียนในรูปของ ASM Chart

Step 2 : Conceptualization

Structure Diagram : ASM ของการบวกแบบอนุกรมประกอบด้วย Controller และ Data Path ดังรูปที่ 17

Control Flow Diagram : Process Control Algorithm ของการบวกแบบอนุกรม สามารถเขียนเป็น ASM Chart ดังรูปที่ 20 System Clock จะใช้เข้าจังหวะ ASM สัญญาณ eclk เป็นสัญญาณตรงข้ามของ System Clock ซึ่ง Counter จะ Increment และ Reg X, Y จะถูก Load และ Shift บนขอบขาขึ้นของ eclk (ขาลงของ Clock)



รูปที่ 20 ASM Chart ของการบวกแบบอนุกรม

System Solution and Simplification

การทำงานของ NS Generator สามารถหาได้จาก Control Flow Diagram (ASM Chart) เราสามารถหา NS Generation Function จาก NS Table หรือ NS Map

สัญญาณควบคุมของ Data Path (สร้างโดย Output Decoder) สามารถหาได้จาก Control Flow Diagram

Step 3 : Solution/Simplification

Process data Path : ออกแบบ Process Data Path เพื่อให้ทำงานตาม Process Algorithm ต่อไปนี้

Process Inputs	xi, yi	
Process Outputs	si = $ci \oplus xi \oplus yi$	I = 0-3
Next State Carry	Ci +1 =xi yi + ci (xi + yi) ,	

Data Path ที่แสดงดัง Block Diagram รูปที่ 15 สามารถทำงานตาม Process นี้ได้

Control Unit Next-State Generator : จาก ASM Chart สามารถนำมาเขียนตารางของ Next State ของแต่ละ PS ดังตารางที่ 6 (State d เป็น State ที่ไม่ใช้ ต้องเปลี่ยน State ไปเป็น Reset State ของระบบ)

ตารางที่ 6 Control Algorithm Table ของ Serial Adder

Present State AB	Name	Next State Name ANBN		Condition for Transition	Condition for AN = 1	BN = 1
00	a	a	00	$\overline{start}$	F	*start*
		b	01	*start*		
01	b	c	11	T	T	T
10	d	a	00	T	F	F
11	c	c	11	$\overline{ct4}$	$\overline{ct4}$	*ct*4
		a	00	*ct*4		

$AN = \overline{A}B + B\overline{ct4}$ $BN = \overline{A}start + \overline{A}B + B\overline{ct4}$

รูปที่ 21 NS map

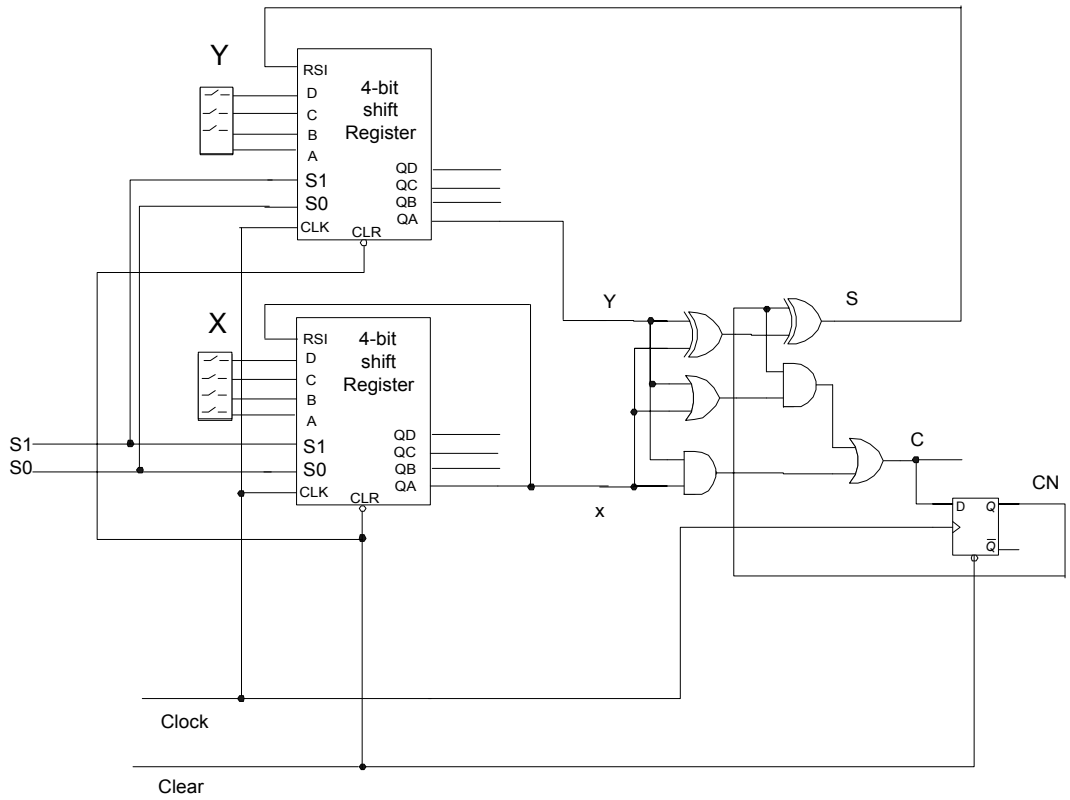
Control Unit Output Decoder : Logic equation ของสัญญาณควบคุมของ Data Path (สร้างจาก O/P decoder) หาได้จาก Control Flow Diagram ดังนี้

S1 = $\overline{AB}$
S0 = B
eClk = start . Clock
clr ctr = $\overline{AB}$
clk-X = $\left(\overline{A}B + AB\right)\cdot eclk$ = B . eclk
clk-Y = $\left(\overline{A}B + AB\right)\cdot eclk$ = B . eclk
clk-C = A B . eclk
inc ct = A B . eclk

State จะเปลี่ยนที่ขอบขาขึ้นของ System Clock ส่วน Counter จะ increment และ X,Y Registerและ Carry flip-flop C จะ Strobe ที่ขอบขาขึ้นของ eclk

System Realization
Step 4 : Realization

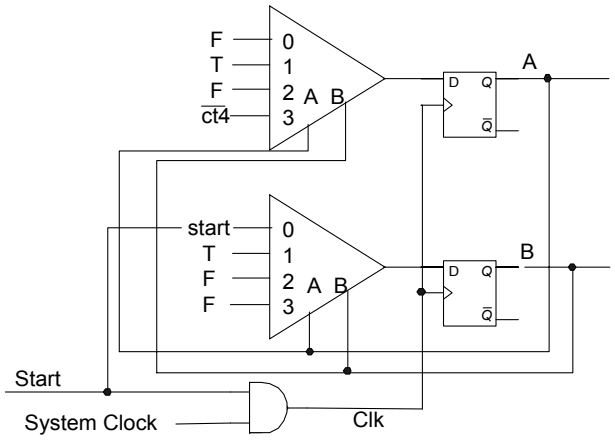
Process data Path : Block Diagram ดังรูปที่ 19 สามารถนำมาสร้าง Logic Diagram ดังรูปที่ 22



รูปที่ 22 Logic Diagram ของ Process Data Path

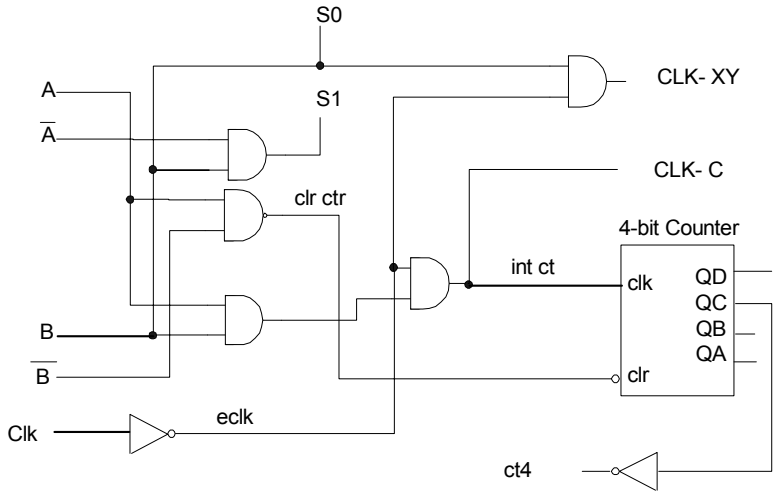
เมื่อทำการบวกเสร็จแล้ว ผลบวกขนาด 5 บิต จะอยู่ที่ carry flip-flop (c4) และ Register Y (y3y2y1y0)

Control Unit Next State Generator : จาก NS Function ที่ได้ เราสามารถสร้างวงจรของ Next State Generator ได้ดังรูปที่ 23



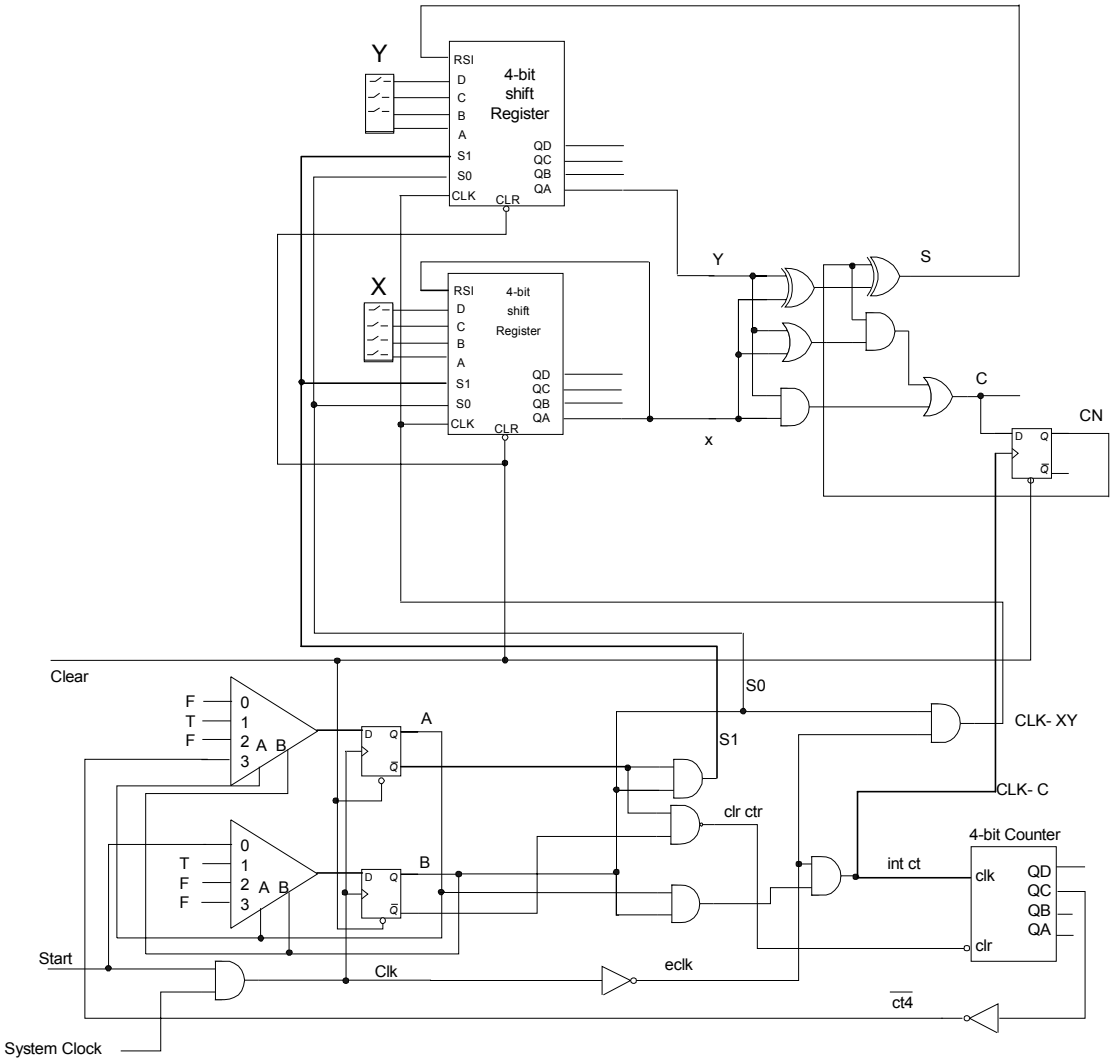
รูปที่ 23 Next State Generator สร้างจาก D flip-flop และ Multiplexer

Control Unit Output Decoder : จาก Logic Equation ของสัญญาณควบคุม Data Path สามารถนำมาสร้างวงจรในส่วนของ Output Decoder ได้ดังรูปที่ 24



รูปที่ 24 Output Decoder สร้างสัญญาณควบคุม Data Path

เมื่อนำส่วนของ Process Data Path มารวมกับ Control Unit จะได้ Logic Diagram ดังรูปที่ 25



รูปที่ 25 Serial Addition ASM