State Machine and Systems Design
Register, Register Operation และ Register Transfer Notation

System ประกอบด้วย

- Register
- Operation

System ประกอบด้วย

- **Register n บิต**
  - กำหนดให้เป็น Rn-1, … , R1, R0 โดย R0 เป็น LSB
- **Register**
  - ชื่อเป็นอักษรตัวใหญ่ เป็นอักษรตัวเดียว (A, B, หรือ C) หรือเป็นชื่อที่บอกถึงการทำงาน 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)**
  - ใช้ในการอธิบายการทำงานของระบบ โดยใช้ตัวอักษรที่เหลือ

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′

Iterative Process

- วงจร Sequential Logic ที่เรียกว่า Controller สามารถจะแสดงการทำงานโดย ASM Model
- Iterative Process เป็นสิ่งที่ใช้ทั้ง Combinational และ Sequential โดยการทำงานของ Combination จะถูกกระทำซ้ำอยู่จำนวนหนึ่ง (Sequential)

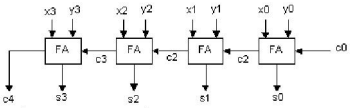
ตัวอย่างของ Iterative Process

- One-Dimension
- Two-Dimension
  - คือวงจร Addition และวงจร Multiplication ของตัวเลขหลายบิต ตามลำดับ
- Iterative Process สร้างได้ 2 วิธี
  - Combination วงจรมีขนาดใหญ่
  - Sequential ซึ่งจะทำให้วงจรใช้เวลาเพิ่มขึ้น

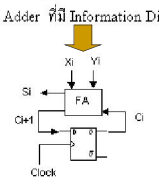
ตัวอย่างของ Iterative Process แบบ One-Dimension

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

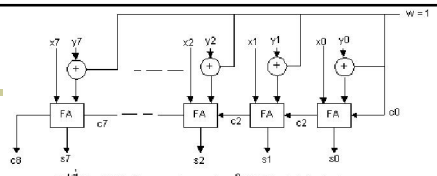
Information Distribute in Space and Distribute in Time



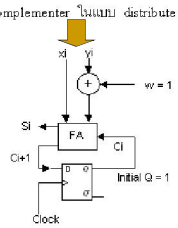
รูปที่ 1 Binary Adder ที่มี Information Distribute in Space



รูปที่ 2 Binary Adder ที่มี Information Distribute in Time



รูปที่ 5 วงจร 2s complementer ในแบบ distribute in space



รูปที่ 4 วงจร 2s complementer ในแบบ distribute in time

ออกแบบวงจร Even-Parity Check Machine

Step 1 : Problem Statement :

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

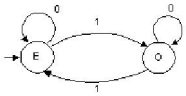
b. เครื่องจะสร้าง O/P Z ซึ่งถ้า Z = 1 แสดงว่า Even-Parity Error (w มีจำนวนบิตของ 1 เป็นจำนวนคี่)
และถ้า Z = 0 แสดงว่า ไม่มี Error (w มีจำนวนบิตของ 1 เป็นจำนวนคู่)

ออกแบบวงจร Even-Parity Check Machine

Step 2 : Conceptualization :

เขียน State Diagram

เขียน State Table



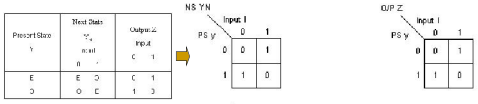
Present State Y	Next State Y_n Input 0	Next State Y_n Input 1	Output Z Input 0	Output Z Input 1
E	E	O	0	1
O	O	E	1	0

ออกแบบวงจร Even-Parity Check Machine

Step 3 : Solution / Simplification :

หา NS Function และ O/P Fcuntion

กำหนดให้ E = 0, O = 1



รูปที่ 6 NS Map และ O/P Map ตัวอย่างที่ 5

จะได้ NS และ O/P Function ดังนี้

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

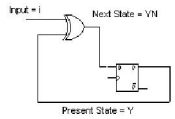
$$Z = Y\bar{I} + \bar{Y}I = Y \oplus I$$

ออกแบบวงจร

Even-Parity Check Machine

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

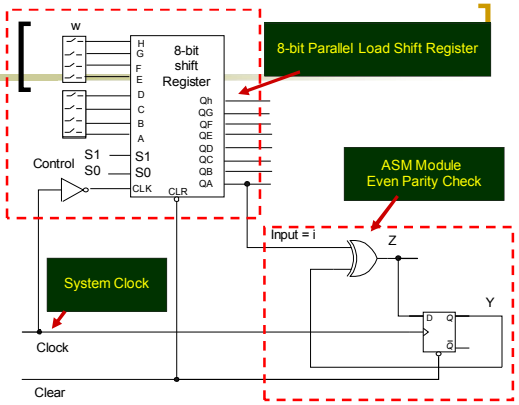


รูปที่ 7 ASM Module ของวงจร Even Parity Check

8-bit Even-Parity Check Machine โดย Rudimentary Control

ประกอบด้วย

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



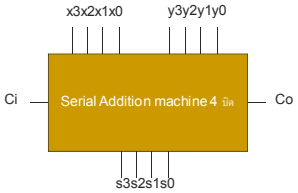
ออกแบบวงจร

Serial Addition machine

Step 1 : Problem Statement :

Serial Addition machine 4 บิต

X = x3x2x1x0
Y = y3y2y1y0
S = s3s2s1s0
Ci ตัวทดเข้า
C0 ตัวทดออก

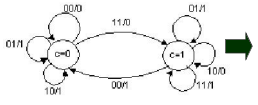


ออกแบบวงจร

Serial Addition machine

Step 2 : Conceptualization :

การบวกเลขแต่ละบิต จะเป็นการบวก xi, yi และ ci เข้าด้วยกัน ได้ผลลัพธ์เป็น si และ ci+1 ซึ่ง Ci+1 จะนำไปเป็น ci ใน Stage ถัดไป เราจึงทำการออกแบบโดยให้ Ci เป็น Present State และ Ci+1



Present State	Next State C				Output S			
	Inputs xy				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

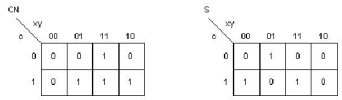
เขียน State Diagram เขียน State Table

ออกแบบวงจร

Serial Addition machine

Step 3 : Solution / Simplification :

หา NS Function และ O/P Fcuntion



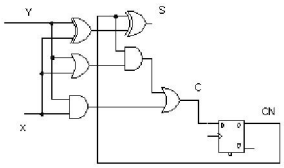
รูปที่ 14 NS และ O/P Map ของตัวอย่างที่ 7

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

ออกแบบวงจร

Serial Addition machine

Step 4 : Realization :



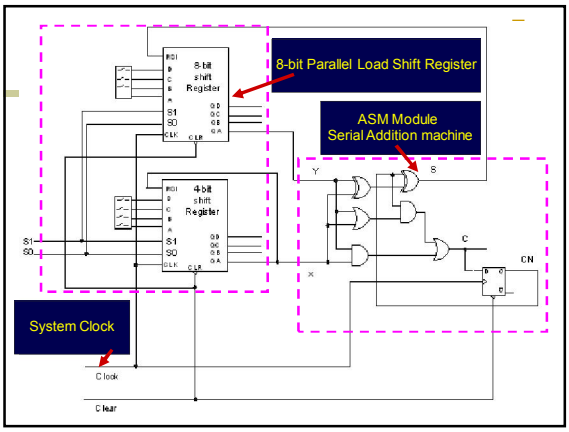
รูปที่ 15 ASM Module ของ serial Addition

ออกแบบวงจร

Serial Addition machine โดย Rudimentary Control

ประกอบด้วย

- System Clock
- 8-bit Parallel Load Shift Register
- ASM Module ของวงจร Serial Addition



Algorithmic State Machines and Systems-Level Design Procedure

Step 1 : Problem Statement :
วางโครงสร้าง Process เป็น 2 ส่วน (อีกแบบอยู่ในรูป Peseudocode) และนอก 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) การสร้างสัญญาณที่ควบคุมส่วนของ Data Path
Step 3 : Solution / Simplification : เขียนวงจรเชื่อมต่อไปนี้
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

Step 1 : Problem Statement :

a.Process

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

b.Process Control Algorithm

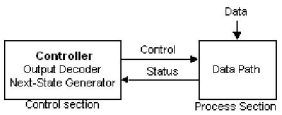
State a : Initialize index (I := 0)
Set Registers' Control to hold (S1, S0 := 0, 0)
State b : Set Registers' control to load (S1, S0 := 1, 1)
Parallel Load data into X and Y Registers
State c : Use Process Algorithm to produce Si and Ci+1
Set Registers' control to Shift right (S1, S0 := 0, 1)
Shift right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

System Conceptualization

Process Control สามารถเขียนเป็น ASM Model ได้
ASM แบ่งออกเป็น 2 ระบบ
1. Process Section (Data Path) ประกอบด้วยอุปกรณ์ที่ใช้เป็น Function เช่น Registers,Multiplexers, Demultiplexers, Adders, Encoders และ Decoders อุปกรณ์ในส่วนของ 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) ดังรูป



รูปที่ 17 Structure Diagram ของ ASM

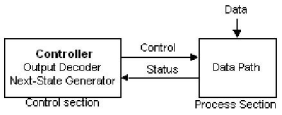
Structure Diagram and Control Flow Diagram

การทำงานของ Process Control Algorithm สามารถอธิบายโดยใช้ Control Flow Diagram ซึ่งจะแสดง State และเงื่อนไขของการเปลี่ยน 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 :

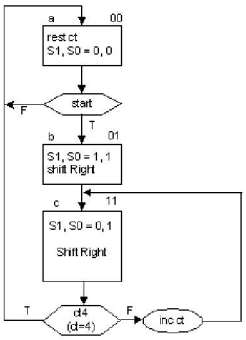
a.System Structure Diagram



รูปที่ 17 Structure Diagram ของ ASM

b.Control Flow Diagram

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)



รูปที่ 20 ASM Chart ของการบวกแบบอนุกรม

Step 3 : Solution / Simplification :

Step 3 : Solution / Simplification :

a. Process Data path

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

I = 0-3

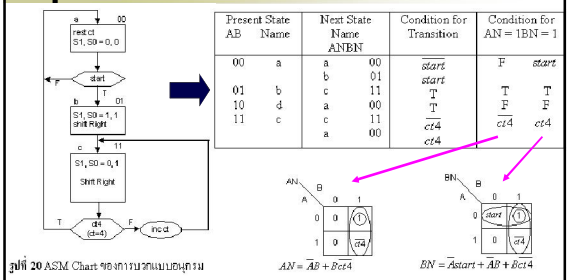
Step 3 : Solution / Simplification :

b. Control Unit

(1) Next State Generator ได้จาก NS Map หรือ Table

Present State AB	Name	Next State Name ANBN		Condition for Transition	Condition for AN = 1BN = 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	$\overline{ct4}$	*ct4*	*ct4*
		a	00	*ct4*		

Next State Generator

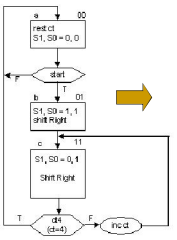


Present State AB	Name	Next State Name	ANBN	Condition for Transition	Condition for AN = 1BN = 1	
00	a	a	00	$\overline{start}$	F	$start$
		b	01	$start$		
01	b	c	11	T	T	T
10	d	a	00	T	F	F
11	c	c	11	$\overline{ct4}$	$\overline{ct4}$	$ct4$
		a	00	$ct4$		

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

รูปที่ 20 ASM Chart ของการบวกแบบอนุกรม

O/P Decoder

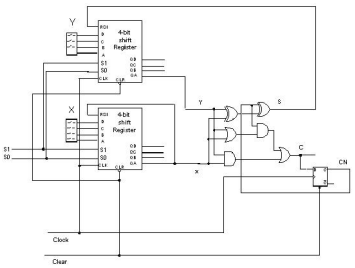


S1 = $\overline{A}B$
S0 = B
eClk = start . Clock
clr ctr = $\overline{AB}$
clk-X = $(\overline{A}B + AB) \cdot eclk$ = B . eclk
clk-Y = $(\overline{A}B + AB) \cdot eclk$ = B . eclk
clk-C = A B . eclk
inc ct = A B . eclk

รูปที่ 20 ASM Chart ของการบวกแบบอนุกรม

Step 4 : Realization :

a. Process Data Path



Step 4 : Realization :

b. Control Unit

(1) NS Generator (2) O/P Decoder