เนื้อหา

Serial Division

Design and Implementation of a Serial Division ASM

มีหลายวิธีในการสร้างความสามารถในการหารใน CPU ของ Computer คือ

- การสร้างลำดับเศษเหลือ โดยที่เศษเหลือจะถูกกำหนดโดย recursion formula ฟังก์ชั่นของการหารสามารถโปรแกรมเป็น subroutine ซึ่งเรียกคำสั่ง shift และ subtract ซ้ำ ๆ กันหลาย ๆ ครั้ง
- serial division ASM สามารถออกแบบขึ้นจาก adder ขนาด n+1 บิท และ Shift register เราจะมาวิเคราะห์ procedure ของการหารตัวเลข n บิท เพื่อหาผลหารและเศษ

Analysis of Binary Integer Division

ในการเขียนตำแหน่งของเลขฐานสอง P และ Q มีค่าดังนี้

$P = 2^3p_3 + 2^2p_2 + 2^1p_1 + 2^0p_0$

$Q = 2^3q_3 + 2^2q_2 + 2^1q_1 + 2^0q_0$

เมื่อ ตัวตั้ง = $P = p_3p_2p_1p_0$ ผลหาร = $Q = q_3q_2q_1q_0$ ตัวหาร = B

partial dividend {PD} แสดงให้เห็นภายใน { } เป็นส่วนของตัวตั้งที่จะถูกหาร ในแต่ละ step ใน step j ผลลัพธ์หรือบิท qj จะบอกจำนวนครั้งที่ตัวหารเข้าไปยัง partial dividend เศษเหลือที่แท้จริง $Rj = \{ PD \} - B\, q_I$ พิจารณาตัวอย่างต่อไปนี้

หาร 1101 ด้วย 11

```
        0 1 0 0.  =        q3q2q1q0
    ____________________
11 | { 1 } 1 0 1
   - 0
     { 11 }
   - 11
     { 00 }
   - 0
     { 01 }
   - 0
       1  = Remainder
```

ตัวอย่างต่อไปจะใช้พีชคณิตในการแสดงวิธีหารเลขจำนวนเต็ม ฐานสอง ในแต่ละค่าของ j เศษเหลือที่แท้จริง { R_j } หาได้จากการลบผลคูณของบิทนั้น { Bq_j } จาก partial dividend { PD } นั่นคือ

{ R_j } = { PD } – Bq_j เมื่อ j = 3, 2, 1, 0

Index j

q_3 q_2 q_1 q_0

B) { p_3 } p_2 p_1 p_0

\- Bq_3

If $B > p_3$, **then** $q_3 = 0$

else $q_3 = 1$

3 = R_3 $R_3 = \{ p_3 \} - Bq_3$

{ $2R_3 + p_2$ }

\- Bq_2

If $B > 2R_3+p_2$, **then** $q_2 = 0$

else $q_2 = 1$

2 = R_2 $R_2 = \{ 2R_3+p_2 \} - Bq_2$

{ $2R_2 + p_1$ }

\- Bq_1

If $B > 2R_2+p_1$, **then** $q_1 = 0$

else $q_1 = 1$

1 = R_1 $R_1 = \{ 2R_2+p_1 \} - Bq_1$

{ $2R_1 + p_0$ }

\- Bq_0

If $B > 2R_1+p_0$, **then** $q_0 = 0$

else $q_0 = 1$

0 = R_0 $R_0 = \{ 2R_1+p_0 \} - Bq_0$

An Algorithm for Binary Integer Division

วิธีที่ใช้ก่อนหน้านี้สามารถนำมาเขียน algorithm สำหรับการหารเลขจำนวนเต็มฐานสอง

Process ของการหาร P ด้วย B สามารถเขียนเป็นสมการได้ดังนี้

$P - BQ = R$

เมื่อ Q = ผลหาร R = เศษเหลือ และ $R<B$

ที่มาของสูตรของสัมประสิทธิ์ของผลหาร qi

การหาร P ด้วย B สามารถเขียนเป็นสมการในรูปต่อไปนี้

$2^3p_3 + 2^2p_2 + 2^1p_1 + 2^0p_0 - B(2^3q_3 + 2^2q_2 + 2^1q_1 + 2^0q_0) = R$ (1)

จุดมุ่งหมายของ process การหารคือหาสัมประสิทธิ์ของผลหาร q3, q2, q1, q0 และเศษเหลือ R สัมประสิทธิ์ qj และเศษเหลือ Rj จะถูกหาขึ้น โดย j = 3, 2, 1,0 ดังต่อไปนี้

หารสมการ (1) ด้วย 2^3 เพื่อแยก q_3

$$\{ p_3 \} - B q_3 = R_3 = \text{เศษ}$$

หารสมการ (1) ด้วย 2^2 เพื่อแยก q_2

$$(2p_3 + p_2) - B(2q_3 + q_2) = R_2$$
$$2(p_3 - Bq_3) + p_2 - Bq_2 = R_2$$

แทน R_3 เข้าที่ $p_3 + Bq_3$

$$2R_3 + p_2 - Bq_2 = R_2$$

หารสมการ (1) ด้วย 2^2 เพื่อแยก q_1

$$4p_3 + 2p_2 + 1p_1 - B(4q_3 + 2q_2 + 1q_1) = R_1$$
$$2[2(p_3 - Bq_3) + (p_2 - Bq_2)] + p_1 - Bq_1 = R_1$$
$$2[2R_3 + (p_2 - Bq_2)] + p_1 - Bq_1 = R_1$$

แทน R_2 เข้าที่ $2R_3 + (p_2 - Bq_2)$

$$2R_2 + p_1 - Bq_1 = R_1$$

หารสมการ (1) ด้วย 2^1 เพื่อแยก q_0

$$8p_3 + 4p_2 + 2p_1 + p_0 - B(8q_3 + 4q_2 + 2q_1 + q_0) = R_0$$
$$8(p_3 - Bq_3) + 4(p_2 - Bq_2) + 2(p_1 - Bq_1) + p_0 - Bq_0 = R_0$$
$$2[2\{2(p_3 - Bq_3) + (p_2 - Bq_2)\} + (p_1 - Bq_1)] + p_0 - Bq_0 = R_0$$
$$2[2\{2R_3 + (p_2 - Bq_2)\} + (p_1 - Bq_1)] + p_0 - Bq_0 = R_0$$
$$2[2R_2 + p_1 - Bq_1] + p_0 - Bq_0 = R_0$$

แทน R_1 เข้าที่ $2R_2 + p_1 - Bq_1$

$$2R_1 + p_0 - Bq_1 = R_0$$

จะเห็นว่าแต่ละสมการของ Rj (j = 3,2,1,0) สามารถเขียนเป็นสูตรดังนี้

$$R_j = \{ PD \} - q_jB$$

การคำนวณค่า q_j และ R_j ใช้ส่วนของตัวตั้ง = { PD }, โดย j = 3, 2, 1, 0 สามารถทำให้สำเร็จได้ดังนี้

- เมื่อ j = 3 : $R_3 = \{ p_3 \} - q_3B$

  สมมติ $q_3 = 1$; คำนวณหาเศษ $Z_3 = \{ p_3 \} - 1B$

ถ้า $Z_3 < 0$: ตัวหารผิด ดังนั้น $q_3 = 0,\ R_3 = \{ p_3 \} - 0B = Z_3 + B$

ถ้า $Z_3 \geq 0$: ตัวหารถูก ดังนั้น $q_3 = 1, R_3 = \{ p_3 \} - 1B = Z_3$

- เมื่อ $j = 2 : R_2 = \{ 2R_3 + p_2 \} - q_2B$

สมมติ $q_2 = 1$; คำนวณหาเศษ $Z_2 = \{ 2R_3 + p_2 \} - 1B$

ถ้า $Z_2 < 0$: ตัวหารผิด ดังนั้น $q_2 = 0,\ R_2 = \{ 2R_3 + p_2 \} - 0B = Z_2 + B$

ถ้า $Z_2 \geq 0$: ตัวหารถูก ดังนั้น $q_2 = 1, R_2 = \{ 2R_3 + p_2 \} - 1B = Z_2$

- เมื่อ $j = 1 : R_1 = \{ 2R_2 + p_1 \} - q_1B$

สมมติ $q_1 = 1$; คำนวณหาเศษ $Z_1 = \{ 2R_2 + p_1 \} - 1B$

ถ้า $Z_1 < 0$: ตัวหารผิด ดังนั้น $q_1 = 0,\ R_1 = \{ 2R_2 + p_1 \} - 0B = Z_1 + B$

ถ้า $Z_1 \geq 0$: ตัวหารถูก ดังนั้น $q_1 = 1, R_1 = \{ 2R_2 + p_1 \} - 1B = Z_1$

- เมื่อ $j = 0 : R_0 = \{ 2R_1 + p_0 \} - q_0B$

สมมติ $q_0 = 1$; คำนวณหาเศษ $Z_0 = \{ 2R_1 + p_0 \} - 1B$

ถ้า $Z_0 < 0$: ตัวหารผิด ดังนั้น $q_0 = 0,\ R_0 = \{ 2R_1 + p_0 \} - 0B = Z_0 + B$

ถ้า $Z_0 \geq 0$: ตัวหารถูก ดังนั้น $q_0 = 1, R_0 = \{ 2R_1 + p_0 \} - 1B = Z_0$

Binary Division Algorithm

จากสูตรของการหาผลหาร qj และ เศษเหลือ Rj สรุปเป็น Algorithm ได้ดังนี้

- เมื่อ $j = 3 : R_3 = \{ p_3 \} - q_3B$

สมมติ $q_3 = 1$; คำนวณหาเศษ $Z_3 = \{ p_3 \} - 1B$

ถ้า $Z_3 < 0$: ตัวหารผิด ดังนั้น $q_3 = 0,\ R_3 = \{ p_3 \} - 0B = Z_3 + B$

ถ้า $Z_3 \geq 0$: ตัวหารถูก ดังนั้น $q_3 = 1, R_3 = \{ p_3 \} - 1B = Z_3$

- เมื่อ $j = 2, 1, 0 : R_j = \{ 2 (R_{j+1} + 2^{-1}p_j) \} - q_jB = \{ PD \} - q_jB$

สมมติ $q_j = 1$; คำนวณหาเศษ $Z_j = \{ 2 (R_{j+1} + 2^{-1}p_j) \} - 1B$

ถ้า $Z_j < 0$: ตัวหารผิด ดังนั้น $q_j = 0,\ R_j = Z_j + B$

ถ้า $Z_j \geq 0$: ตัวหารถูก ดังนั้น $q_j = 1, R_j = Z_j$

Restoring Method of Binary Integer Division

การสมมติว่า qj = 1 จะได้เศษสมมติ Zj = { PD } – 1B ถ้าค่าของ Zj < 0 ตัวหารผิด และ qj = 0

restoring Method ของการหารเลฐานสอง จะสร้างตัวตั้งโดยบวก B เข้าที่เศษสมมติ Zj เศษที่แท้จริง Rj หาได้จาก

ถ้า $Z_j < 0$: $q_j = 0$, $R_j = \{ PD \} - 0B = Z_j + B$

ถ้า $Z_j \geq 0$: $q_j = 1$, $R_j = \{ PD \} - 1B = Z_j$

restoring method ของการหารเลขฐานสองทำได้ตาม ขั้นตอนต่อไปนี้

Register A และ P ต่อกันเพื่อเป็น 8-bit register

Initial Condition : $R_4 = 0$; $A,P = 0000.p_3p_2p_1p_0$

ให้ j เป็น step ซึ่ง หา q_j และ R_j

ที่ $j = 3, 2, 1, 0$ $R_j = \{ 2 (R_{j+1} + 2^{-1}p_j) \} – q_jB$

ซึ่งหาได้โดยขั้นตอนต่อไปนี้

1. Shift Left AP : $A = \{ 2 (R_{j+1} + 2^{-1}p_j) \} – q_jB = 2 R_{j+1} + p_j$
2. Subtract : (สมมติ $q_j = 1$) $Z_j = A – 1B = \{ PD \} - 1B$
3. ถ้า $Z_j < 0$: $q_j=0, R_j =\{ PD \}-0B = Z_j + B = Rj \rightarrow A$

   ถ้า $Z_j \geq 0$: $q_j=1, R_j =\{ PD \}-1B = Z_j = Rj \rightarrow A$

จะเห็นว่า ถ้า $Z_j < 0$: $(\overline{Z_j \text{ sign}}) = \overline{(1)} = 0 \rightarrow q_j$

ถ้า $Z_j \geq 0$: $(\overline{Z_j \text{ sign}}) = \overline{(0)} = 1 \rightarrow q_j$

เมื่อ Z_j sign คือ sign bit ของ Z_j ดังนั้น $\overline{q_j} = (Z_j \text{ sign})$

Nonrestoring Method of Binary Division

Nonrestoring Method ของการหารเลขฐานสองจะใช้เศษสมมติ $Z_j = \{ PD \} - 1B$ โดยลดค่า Partial Dividend { PD } ไป B ที่ index j ถ้า $Z_j < 0$ ตัวหารผิด ดังนั้น q =0 และ 2B จะถูกบวกเข้าที่ {PD} ใน step ต่อไปเพื่อให้ส่วนของตัวตั้งถูกต้อง ขั้นตอนของ Nonrestoring Method แสดงให้เห็นดังตัวอย่างต่อไปนี้ จะเห็นว่า

ถ้า $Z_j < 0$, Z_j sign = 1 และ $q_j = 0$: ถ้า $Z_j \geq 0$, Z_j sign = 0, $q_j = 1$ ดังนั้น $\overline{q_j} = (Z_j \text{ sign})$

ตัวอย่างการหาร P = 1101 ด้วย B = 0011

0100 = ผลลัพธ์

1111 = ผลลัพธ์ทดสอบ

0011 | 0 0000 1101 = 0 0000 p3p2p1p0

Shift Left {p_3} 1

Subtract - B - 11

Z_3 = - 10 $Z_3 < 0$: $q_3 = 0$

Add Next

Shift Left $2Z_3 + p_2$ - 100+ 1

Add + B + 11

Z_2 = 0 $Z_2 = 0$:$q_2 = 1$

Subtract Next

Shift Left $2Z_2 + p_1$ 2(0) + 0

Subtract - B - 11

Z_1 = - 11 $Z_1 < 0$: $q_1 = 0$

Add Next

Shift Left $2Z_1 + p_0$ - 110+ 1

Add + B + 11

Z_0 = -10 $Z_0 < 0$: $q_0 = 0$

Add Next

R = Z_0 + B -10

+11

R = 1 = เศษ

สรุปจากตัวอย่างว่า ถ้าเศษสมมติ ถ้า $Z_{j+1} < 0$ ตัวหารผิด ดังนั้น q =0 เราต้องบวก 2B เข้าที่ {PD} ใน step ต่อไปเพื่อให้ส่วนของตัวตั้งถูกต้อง

$Z_j = 2 Z_{j+1} + 2B + p_j \} - 1B \leftrightarrow (2 Z_{j+1} + p_j) + B$

Shift left Add

ถ้าเศษสมมติ ถ้า $Z_{j+1} \geq 0$ ตัวหารถูก ดังนั้น q =1 ไม่ต้องแก้ไข

$Z_j = 2 Z_{j+1} + 0B + p_j \} - 1B \leftrightarrow (2 Z_{j+1} + p_j) - B$

Shift left Subtract

$(\overline{Z_j \text{ sign}}) \rightarrow q_j$

ให้ P เป็น 4-bit shift register ที่เริ่มต้นด้วยการเก็บค่าตัวตั้ง P

ให้ B เป็น 4-bit shift register ที่เริ่มต้นด้วยการเก็บค่าตัวหาร B

Bs เป็น Flip-Flop เพิ่มเข้ามาที่ซ้ายสุดของ register B เพื่อเก็บ sign bit

ให้ A เป็น4-bit shift register ที่เริ่มต้นด้วยการ clear

As เป็น Flip-Flop เพิ่มเข้ามาที่ซ้ายสุดของ register A เพื่อเก็บ sign bit

Register A จะใช้สำหรับเก็บค่าส่วนของตัวตั้ง { PD } ที่แต่ละขั้นของ process

Register A, P จะต่อกันโดย A อยู่ด้ายซ้าย

ใช้ 2s Complement arithmetic เพื่อทำการลบตัวเลข 5 บิทใน BsB ออกจาก AsA

Nonrestoring Method สามารถทำได้โดยใช้ algorithm ต่อไปนี้

- เมื่อ j= 3, Shift left AP : { PD } = p_3

  สมมติ $q_3 = 1$; คำนวณหา $Z_3 = p_3 - \overline{1}B$; $Z_{3s} \rightarrow q_3$

- เมื่อ j = 2, 1, 0

ถ้า $Z_{j+1} < 0$ บวก 2B เพื่อให้ส่วนของตัวตั้งถูกต้อง

$Z_j = 2 Z_{j+1} + 2B + p_j \} - 1B \quad \leftrightarrow \quad (2 Z_{j+1} + p_j) + B$

Shift left Add

ถ้า $Z_{j+1} \geq 0$

$Z_j = 2 Z_{j+1} + 0B + p_j \} - 1B \quad \leftrightarrow \quad (2 Z_{j+1} + p_j) - B$

Shift left Subtract

$(\overline{Z_j \text{ sign}}) \rightarrow q_j$

ดูตัวอย่างของ nonrestoring binary division ดังต่อไปนี้ เมื่อ P = 1101, B = 0011

(1 1101 เป็นค่า 2s complement ของ BsB)

j	Time	Function	Bs	B	As	A	P
			0	0011	0	0000	1101
3	t1	Shift left				0	0001
	101_						
	t2	Subtract				1	1101
		$\overline{Z}s \rightarrow q_3 = 0$			1	1110	1010
		Add Next Since Zs = 1					
2	t1	Shift left				1	1101
	010_						
	t2	Add			0	0011	
		$\overline{Zs} \rightarrow q_2 = 0$			0	0000	0101
		Subtract Next Since Zs = 0					
1	t1	Shift left				0	0000
	101_						
	t2	Subtract				1	1101

$\overline{Zs} \rightarrow q_1 = 0$ 1 1101 1010

Add Next
Since Zs = 1

0 t1 Shift left 1 1011
010_
t2 Subtract 0 0011
_____

$\overline{Zs} \rightarrow q_0 = 0$ 1 1110 0100

Add Next

ผลหาร

Since Zs = 1

t1 _____
t2 R = A + B 0 0011
0 0001 = เศษ

Design of a Serial Division ASM

Step 1 : Problem Statement :

a. Process

- เมื่อ j= 3, Shift left AP : $\{ PD \} = p_3$

  สมมติ $q_3 = 1$; คำนวณหา $Z_3 = p_3 - 1B$; $Z_{3s} \rightarrow q_3$

- เมื่อ j = 2, 1, 0

  ถ้า $Z_{j+1} < 0$ บวก 2B เพื่อให้ส่วนของตัวตั้งถูกต้อง

  $Z_j = 2 Z_{j+1} + 2B + p_j \} - 1B \leftrightarrow (2 Z_{j+1} + p_j) + B$
  Shift left Add

  ถ้า $Z_{j+1} \geq 0$

  $Z_j = 2 Z_{j+1} + 0B + p_j \} - 1B \leftrightarrow (2 Z_{j+1} + p_j) - B$
  Shift left Subtract

  $(\overline{Z_j \text{ sign}}) \rightarrow q_j$

b. Process control algorithm

state a : Wait: Wait for start

state b : Reset, Load : $0 \rightarrow I$, A; $3 \rightarrow j$
Divisor $\rightarrow$ B
Dividend $\rightarrow$ P
Define $Z_4 = 0$

state c : Shift Left AP : $Z = 2Z_{j+1} + p_j$

state d : Add/Subtract : If $Z_{j+1} < 0$, $Z_j = Z + B$
If $Z_{j+1} \geq 0$, $Z_j = Z - B$
$I + 1 \rightarrow I$, $j-1 \rightarrow j$
If I<4, repeat State c and d;

If I=4, process is completed; goto e.

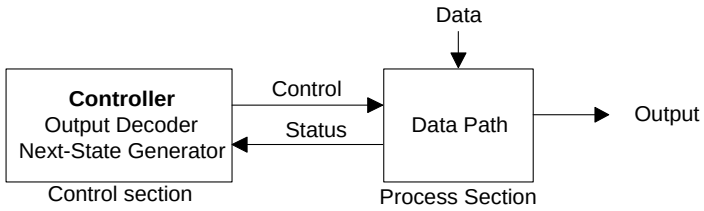
state e : Restore Remainder : If $Z_0 < 0$, Remainder $R = Z_0 + B$

System Specification: ออกแบบ ASM เพื่อทำการหารเลขจำนวนเต็ม X และ Y

$X \leftrightarrow X / Y$

Step 2 : Conceptualization :

a. System structure diagram



รูปที่ 11 Structure Diagram ของ Serial Division ASM

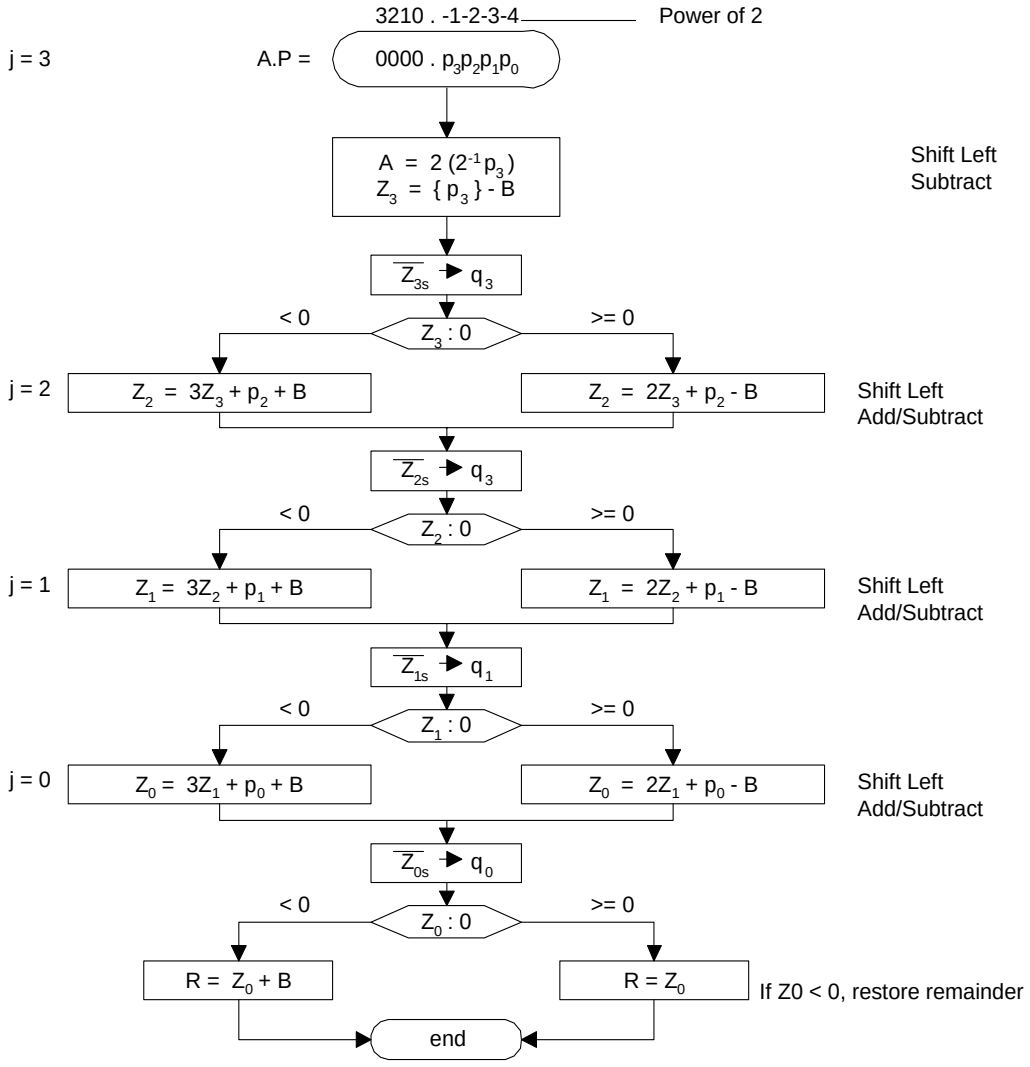
b. Control Flow Diagram

Control Flow Algorithm ของ nonrestoring method ของการหารแบบอนุกรม สามารถเขียนเป็น Diagramได้ดังรูปที่ 12

เมื่อ j = 2, 1, 0

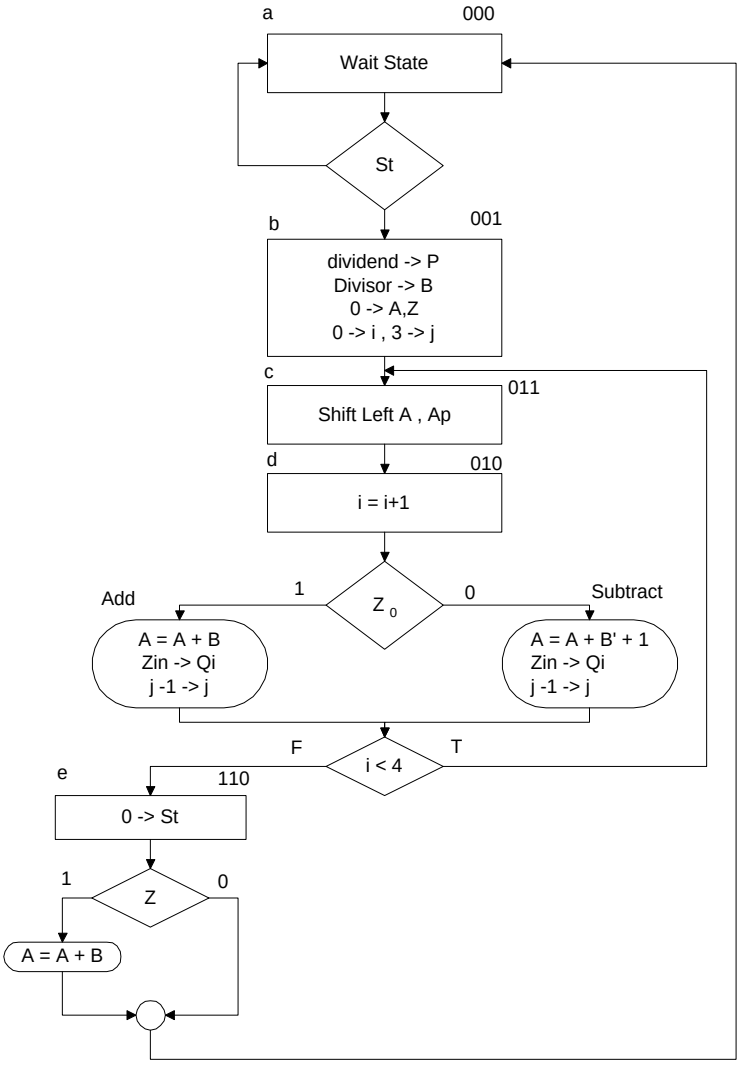
: If $Z_{j+1} < 0$, $Z_j = Z_{j+1} + \bar{B}$; $Z_{js} \rightarrow q_j$

If $Z_{j+1} \geq 0$, $Z_j = Z_{j+1} - \bar{B}$; $Z_{js} \rightarrow q_j$



รูปที่ 12 Diagram ของ nonrestoring Division

Algorithm ของการหารแบบ nonrestoring สามารถเขียนเป็น ASM chart ดังรูปที่ 13 ซึ่ง $Z_{sn} = (Z_{s0} \oplus A_s) \oplus C_0$ โดย Z_{s0} และ Z_{sn} แทนค่าเก่าและใหม่ของเครื่องหมาย Z_s ของผลลัพธ์ Z_j



รูปที่ 13 ASM Chart ของการหารแบบ nonrestoring

Step 3 : Solution / Simplification.

a. Process data path

ออกแบบ Data Path ที่สนับสนุน micro operation ต้องการในการสร้าง Algorithm ของการหารเลขจำนวนเต็มแบบ nonrestoring ซึ่งสามารถสร้างได้จาก

4-bit adder
4-bit register B
4-bit parallel Load shift registers A, P
1-bit Flip-Flops A_s และ Z_s

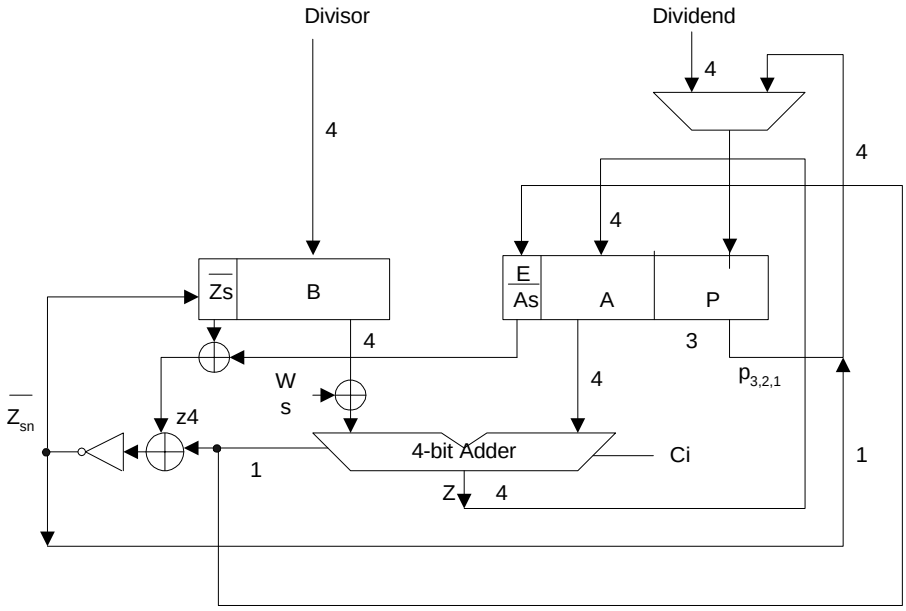
เริ่มต้น ให้ A_s, A, และ Z_s เป็น 0, P เก็บค่าตัวตั้ง ในแต่ละ cycle j, qj จะเก็บที่ LSB ของ P เมื่อการหารจบ register P จะเก็บค่าผลลัพธ์ q_3 q_2 q_1 q_0

ให้ Z เป็น O/P sum 4-bit ของ Adder และ Z_4 เป็น Carry-out

วงจร 2:1 MUX จะใช้สำหรับต่อ Register เข้ากับทั้งการ Load ตัวตั้ง และ p_3 p_2 p_1 Z_{sn}

5-bit addition/subtraction สามารถทำได้โดย

$$Z_{sn} = \overline{(Z_{s0} \oplus A_s)} \oplus C_0$$



รูปที่ 14 Block Diagram ของ Data Path

b. Control Unit

(1) Next State Generator

(a) จาก ASM Chart รูปที่ 13 สามารถนำมาสร้าง Next State Table ของ Control Algorithm ได้ดังตารางที่ 3

ตารางที่ 3 Control Algorithm Table ของ Serial Addition

Present State ABC Name		Next State Name AnBnCn		Condition for Transition	Condition for An=1	Bn=1	Cn=1
000	a	a	000	$\overline{St}$	F	F	st
		b	001	St			
001	b	c	011	T	F	T	T
010	d	c	011	ct4	ct4	T	ct4
		e	110	$\overline{ct4}$			—
011	c	d	010	T	F	T	F
110	e	a	000	T	F	F	F

(b) เขียน Next-State Map และ NS Function (ถ้าใช้ D flip-flop) ได้ดังรูปที่ 15

An BC

A \	00	01	11	10
0	0	0	0	ct4
1	0	0	0	0

$DA = \overline{A}B\overline{C}.ct4$

Bn BC

A \	00	01	11	10
0	0	1	1	1
1	0	0	0	0

$DB = \overline{A}C + \overline{A}B$

Cn BC

A \	00	01	11	10
0	st	1	0	$\overline{ct4}$
1	0	0	0	0

$DC = \overline{A}\,\overline{B}C + A\underline{B}.st + AB\underline{C}.ct4$

รูปที่ 15 NS Map

(2) Output Decoder

พิจารณา ASM Chart และ Data Path Diagram มากำหนด Sequence ของ Micro operation ที่ต้องการในแต่ละ Control State

State a (000) : Set Register Control For Load

State b (001) : Clear Counter ($0 \rightarrow i$) ; $3 \rightarrow j$; Load Dividend and Divisor to Register P and B

State c (011) : Shift Left

$Z = 2Z_{j+1} + p_j$

State d (010) : Add/Subtract (For j=3, subtract since Z(4) = 0 by definition)

If $Z_{j+1} < 0$, $Z_j = Z + B$

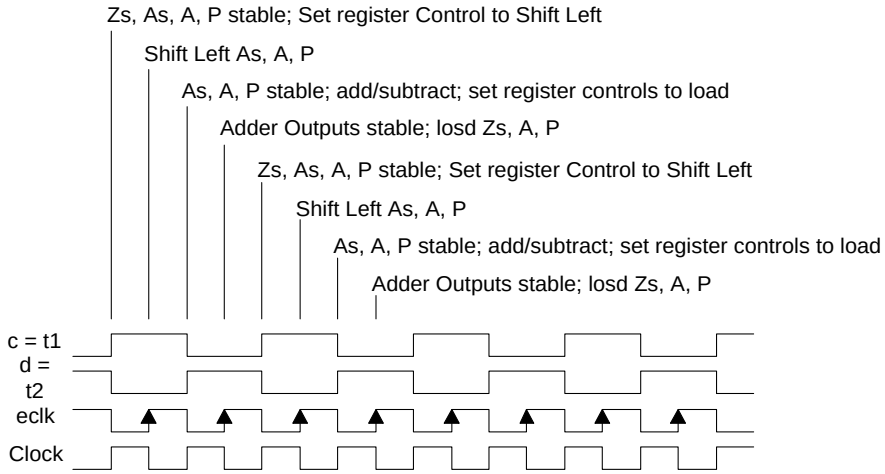
If $Z_{j+1} \geq 0$, $Z_j = Z - B$

Increment Counter i = i+1, j = j+1

If I<4, repeat State c and d;
If I=4, process is completed; goto e.

State e (110) : Restore Remainder
If Z0 < 0 , remainder R = Z0 + B
$0 \rightarrow$ st

สมารถเขียน Timing Diagram ของ State c, d ได้ดังรูปที่ 16



รูปที่ 16 Timing Diagram ของ State c, d ของ Serial division ASM

จากการทำงานในแต่ละ State สามารถนำมาเขียนตารางแสดง micro operation ได้ดังตารางที่ 4

Output Decoder จะสร้างสัญญาณควบคุมต่อไปนี้ไปยัง Data Path

clr-A = $\bar{A}\bar{B}C$
clr-ct = $\bar{A}\bar{B}C$
clk_B = $\bar{A}\bar{B}C$. eclk
clk-A = $(\bar{A}BC \mp A\bar{B}C + \bar{A}BCZ_{sn})$. eclk
clk-A_s = $\bar{A}BC$. eclk
clk-P = $(\bar{A}BC \mp A\bar{B}\bar{C} \mp ABC)$. eclk
clk-Z_s = $\bar{A}B\bar{C}$ .eclk
Wx = $\bar{A}B\bar{C}Z_{s0}$
carry-in = $\bar{A}\bar{B}\bar{C}Z_{s0}$
S1 = 1
S0 = $(\bar{A}BC)$
mux-set = $\bar{A}\bar{B}\bar{C}$
inc-ct = $\bar{A}B\bar{C}$. eclk

โดยที่มีการเริ่มต้นที่ $A_s = 0$, $Z_s = 0$ ($Z_s = 1$) และ $P = p_3p_2p_1p_0$

ตารางที่ 4 สัญญาณควบคุม Data Path ที่สร้างโดย Output Decoder

State/ Time	Function	Micro operations	Control Signal	S1S0	MUX Select
a (000)	Set S1S0 to Load			11	
b (001)	Clear Counter		clr-ct = $\overline{A}\,\overline{B}\,c$		
	Clear Accumulator		clr-A = $A\,B\,C$		
	Load Divisor and	Load B	clk-B = $\overline{A}\,B\,C$.		
	Dividend	Load P	eclk	11	1
c (011)			clk-P = $\overline{A}\,\overline{B}\,C$.	10	
	Set S1S0 to Shift Left	$S.LA_sAP \rightarrow A_sAP$	eclk		
	Shift Left A_sAP		clk-A_s = $A\,B\,C$.		
d (010)			eclk	11	
		$A \pm B \rightarrow A$	clk-A = $A\,\overline{B}\,C$.		
	Set S1S0 to Load Add/Subtract (Z = A ± B)	$i + 1 \rightarrow i$	eclk		
		$Z_{sn} \rightarrow Z_s$	clk-P = $A\,\overline{B}\,C$.		
		$Z \rightarrow A$	eclk		
	Increment Counter	$Z_{sn} \rightarrow P_0$	Wx = $A\,\overline{B}\,\overline{C}\,Zs0$		
e (110)	Store Z_s	$A+B \rightarrow A$	carry-in = $\overline{A}\,\overline{B}\,C$.	11	
	Store A		Zs0		
	Store P		inc-ct = $A\,\overline{B}\,C$.		
	Restore Remainder		eclk		
			clk-Z_s = $\overline{A}\,B\,C$.		
			eclk		
			clk-A = $\overline{A}\,\overline{B}\,C$.		
			eclk		
			clk-P = $A\,\overline{B}\,C$.		
			eclk		
			clk-A = $A\,B\,C\,Z_{sn}$.		
			eclk		