เนื้อหา

Serial Multiplier

Design and Implementation of a Serial Multiplication ASM

มีทางเลือกหลายวิธีในการสร้างความสามารถในการคูณใน CPU ของ Computer ดังนี้

- โดยการสร้าง Sequence ของ Partial-Product Sums (PPS) โดยที่ partial-product sums สามารถหาได้โดย recursion formula ตัวอย่างเช่น ฟังก์ชั่นของการคูณสามารถโปรแกรมให้เป็น subroutine ที่ใช้คำสั่ง add และ shift ซ้ำๆ กัน
- สร้างวงจร Multiplier เป็นแบบ combination-logic ซึ่งอาจสร้างจาก logic gate หรือสร้างจาก ROM table lookup ก็ได้
- สร้าง ASM ของการคูณแบบอนุกรม ซึ่งสร้างได้จาก n-bit adder และ shift register ซึ่งวงจร Multiplier แบบ Combination-logic นั้นได้เคยกล่าวถึงไปแล้ว ต่อไปนี้เราจะได้มาดูวิธีในการสร้าง ASM ของการคูณแบบอนุกรมบ้าง

Analysis of Binary Integer Multiplication

ลองดูการคูณเลขขนาด 4 บิต ให้ เป็นตัวตั้ง และ P เป็นตัวคูณ

$$B = b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 2^0$$
$$P = p_3 2^3 + p_2 2^2 + p_1 2^1 + p_0 2^0$$

ในทางพีชคณิต ผลคูณ BP สามารถเขียนเป็นการบวกของผลคูณแต่ละบิตดังนี้

$$\sum bp_i = \sum B (p_i 2^i)$$

หรือ $$BP = B (p_3 2^3) + B (p_2 2^2) + B (p_1 2^1) + B (p_0 2^0)$$ **(1)**

ขั้นตอนของการคูณตัวเลขจำนวนเต็มสองจำนวนนั้นตัวตั้ง B จะถูกคูณโดยบิต pi ของตัวคูณ (i=0,1,2,3) ถ้า pi = 1 ผลคูณบิตนั้นได้จากการ copy ตัวตั้ง B ลงมา ถ้า pi=0 ผลคูณบิตนั้น = 0 ผลคูณแต่ละบิตจะถูกเลื่อนซ้าย 1 ตำแหน่งตามนัยสำคัญ เช่น

B = 1110
P = 1011

1110	=	$B (p_0 2^0)$	=	bp_0
1110	=	$B (p_1 2^1)$	=	bp_1
0000	=	$B (p_2 2^2)$	=	bp_2
1110	=	$B (p_3 2^3)$	=	bp_3
10011010	=	product BP	=	sum of bit products bp_i

เราสามารถทำตามวิธีนี้ได้โดยใช้ PPS Method เริ่มต้นด้วยการให้ผลบวกสะสมเป็น 0 แล้วบวกผลคูณแต่ละบิตเข้าไป

เริ่มด้วยการ initial ให้ผลบวกสะสม $S_0 = 0$, Sequence ของ PPS S_{i+1} (i=0,1,2,3) จะหาได้ดังต่อไปนี้

	0000.	S_0		
i=0	Add	+ 1110.	+ Bp_0	
		01110.	= $Bp_0 + S_0$	
	Shift Right	0111.0	S_1	$= 2^{-1}(Bp_0+S_0)$
i=1	Add	+ 1110.	+ Bp_1	
		10101.0	= $Bp_1 + S_1$	
	Shift Right	1010.10	S_2	$= 2^{-1}(Bp_1+S_1)$
i=2	Add	+ 0000.	+ Bp_2	
		01010.10	= $Bp_2 + S_2$	
	Shift Right	0101.010	S_3	$= 2^{-1}(Bp_2+S_2)$
i=3	Add	+ 1110.	+ Bp_3	
		10011.010	= $Bp_3 + S_3$	
	Shift Right	1001.1010	S_4	$= 2^{-1}(Bp_3+S_3)$
	24S4 =	10011010.	= BP	

An Algorithm for Binary Integer Multiplication

Algorithm สำหรับคูณเลขจำนวนเต็มฐานสองโดยใช้วิธี PPS สามารถหาได้โดยการดึงตัวร่วม 2^4 จากสมการที่ 1 และเขียนสมการในรูปแบบต่อไปนี้

$$BP = 2^4 (B p_3 2^{-1} + B p_2 2^{-2} + B p_1 2^{-3} + B p_0 2^{-4}) \quad \textbf{(2)}$$

สมการที่ 2 สามารถเขียนในรูปแบบ nested form (ซึ่งมี ผลบวกเริ่มต้น = 0) ดังนี้

$$BP = 2^4 [2^{-1} (B p_3 + 2^{-1} \{ B p_2 + 2^{-1} [B p_1 + 2^{-1} (B p_0 + 0)] \})] \quad \textbf{(3)}$$

Add
Shift right
Add
Shift Right
Add
Shift right
Add
Shift right
Shift Left 4 bit

เทอมในสุด ($Bp_0 + 0$) เมื่อ $0 = S_0$ คือผลบวกสะสมเริ่มต้น ผลบวกขนาด 5 บิต จะเก็บไว้ที่ Register และ register จะถูกเลื่อนขวาไป 1 บิต เพื่อคูณกับ 2^{-1} ผลคูณ BP สามารถหาได้โดยทำการ Add และ shift จำนวน 4 ครั้ง ตามด้วยการเลื่อนซ้ายมา 4 บิต

เริ่มด้วยค่าผลบวกเริ่มต้น $S_0 = 0$ sequence ของ PPS จะได้เทอมของ nest expression ดังนี้

$S_1 = 2^{-1} (B p_0 + S_0) = 2^{-1} (B p_0 + 0)$
$S_2 = 2^{-1} (B p_1 + S_1) = 2^{-1} [B p_1 + 2^{-1} (B p_0 + 0)]$
$S_3 = 2^{-1} (B p_2 + S_2) = 2^{-1} \{ B p_2 + 2^{-1} [B p_1 + 2^{-1} (B p_0 + 0)] \}$
$S_4 = 2^{-1} (B p_3 + S_3) = 2^{-1} (B p_3 + 2^{-1} \{ B p_2 + 2^{-1} [B p_1 + 2^{-1} (B p_0 + 0)] \})$

PPS Algorithm of Binary Integer Multiplication

แต่ละเทอมของ Sequence ของการคูณเลขจำนวนเต็มฐานสองสามารถทำได้โดย Recursion formula ต่อไปนี้

$S_{i+1} = 2^{-1} (B p_i + S_i)$ (i = 0, 1, 2, 3) โดยเริ่มที่ $S_0 = 0$

Add
Shift right

recursion formula จะสร้างเทอมของ S_{i+1} โดยคำนวณค่า $B p_i + S_i$ และ Shift ผลบวกไป 1 บิต ส่วนผลคูณที่ต้องการหาจาก $BP = 2^4 S_4$

Design of a Serial-Multiplication ASM

Step 1 : Problem Statement : แบ่ง Process เป็น 2 ส่วน และอธิบาย Spec ของระบบ

a. Process. เริ่มด้วย $S_0 = 0$ Sequence ของ PPSs สามารถสร้างจาก Recursion formula ดังนี้

$S_{i+1} = 2^{-1} (B p_i + S_i)$ (i = 0, 1, 2, 3) โดยเริ่มที่ $S_0 = 0$

Add
Shift right

b. Process Control Algorithm. เขียน Procedure ของ Process แบบ step-by-step ประกอบด้วย การ Initial, การ load data, การให้ค่าของผลลัพธ์

State a. Wait : Wait for start
State b. Reset, Load : $0 \rightarrow i, S_0$
Multiplicant $\rightarrow B$
Multiplier $\rightarrow P$
State c. Add : $Z = B p_i + S_i$
State d. Shift right : $Z 2^{-1} \rightarrow S_{i+1}$
$i = i + 1$

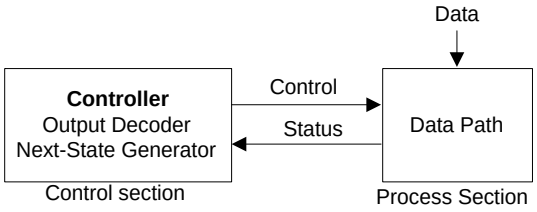
If i < 4, repaet state c and d
if i = 4 process of Multiplying BP is complete

System specification. ออกแบบ ASM ที่ทำการคูณตัวเลข 4 บิต 2 จำนวน X, Y

$X \leftarrow X\ Y$

Step 2 : Conceptualization :

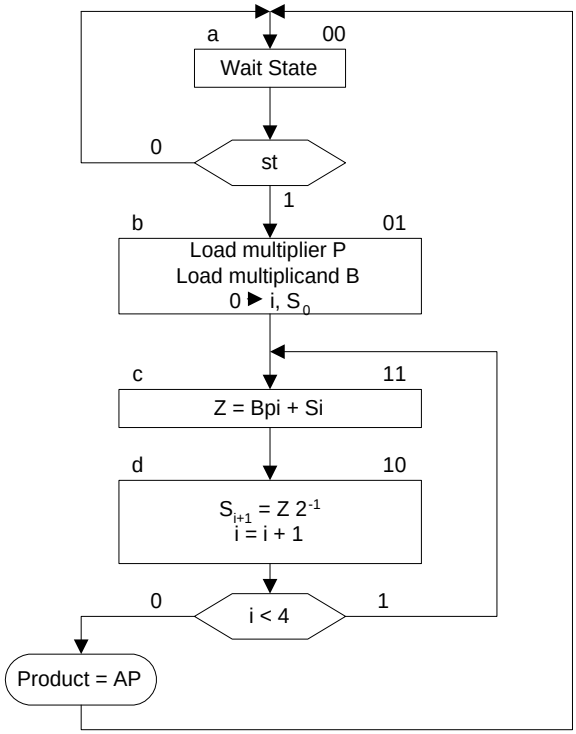
a. System structure Diagram ดังรูปที่ 1



รูปที่ 1 System Structure Diagram

b. Control Flow Diagram

PPS Method ของการคูณเลขจำนวนเต็มฐานสองสามารถเขียนเป็น ASM chart ได้ดังรูปที่ 2 (โดย st คือ Start the process ซึ่ง Process จะเริ่มทำงานเมื่อ st = 1)



รูปที่ 2 ASM Chart ของการคูณเลข 4 bit แบบอนุกรม

Step 3 : Solution/Simplification.

a. Data Path

ใช้วงจร 4-bit adder เป็นตัวบวกผลคูณ Bp_0 เข้ากับ accumulator A ซึ่งเริ่มต้นที่ 0 , BP_0 และ A จะนำมาบวกกันเพื่อให้ได้ 5-bit PPS Z ซึ่งจากนั้นจะนำกลับไปเก็บที่ Register E และ A หลังจากการบวก register E, A, P จะถูก shift right1 บิตเพื่อจัด binary point ของ I/P ของ adder

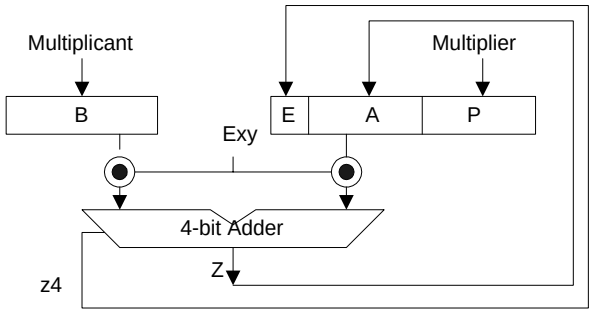
ผลคูณแต่ละบิต Bp_i จะเป็น I/P X ของ Adder และ Register A จะเตรียม I/P 4 บิตให้กับ I/P Y ของ Adder หลังจากการบวก register E, A, P ต้องถูก shift right 1 บิต ดังนั้น LSB ของตัวคูณจะถูกทิ้งไปหลังใช้งาน และ PPS ใน AP จะมากขึ้น 1 บิตในการบวกแต่ละครั้ง

PPS Algorithm สามารถสร้างขึ้นโดยใช้

4-bit adder
4-bit register B
4-bit parallel load shift register A และ P
1-bit extension flip-flop E

เริ่มต้น A = 0 และ P เก็บค่าตัวคูณ E และ A เป็น 5-bit Register ที่ใช้เก็บค่าผลบวกของตัวเลข 4 บิต Register E, A และ P จะรวมกันเป็น 9-bit shift right register

Block diagram ของ data path แสดงดังรูปที่ 3 โดย Z แทนค่า Output ของ Adder และ z4 คือ carry-out ของ adder



รูปที่ 3 Block Diagram ของ data path ของ serial multiplication ASM

b. Control Unit.

(1) Next-state Generator.

(a) ASM chart รูปที่ 2 สามารถนำมาเขียนตาราง control algorithm NS table ดังตารางที่ 1

วงจร counter ใช้สำหรับนับ index i ct4 หมายถึงนับได้ i = 4

ตารางที่ 1 control algorithm table ของ serial multiplication

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

(b) เขียน NS map ได้ดังรูปที่ 4

AN

A \ B	0	1
0	F	T
1	ct4	T

BN

A \ B	0	1
0	st	T
1	ct4	F

รูปที่ 4 NS map ของ Control Algorithm

(2) Output Decoder

จาก ASM Chart และ Data Path Diagram สามารถกำหนดลำดับของ microoperation ที่ต้องการในแต่ละ control state

state a (00) : Wait

state b (01) : Clear Accumulator และ Counter

Load ตัวคูณ ตัวตั้งเข้าที่ Register P และ B ตามลำดับ

state c (11) :โดยที่ เวลาระหว่าง t1 = state c = AB Set ขา control ของ A เป็น Load ที่เวลา t1 ค่าของ Bpi และ A ที่ I/P X, Y ของ Adder หลังจากการบวก ผลบวกปรากฏที่ O/P Z ของ adder โดยมี carry-out = z4 นั่นคือ

$$Z = Bp_i + S_i$$

หลังเวลา t1.eclk↑ (กึ่งกลางของ t1) z4 → E และ Z → A (เพื่อเก็บค่าผลบวกขนาด 5 บิต)

state d (10) : โดยที่ระยะเวลา t2 = state d = $A\overline{B}$ Set ค่า control ของ A, P เป็น shift right

ที่เวลา t2. eclk↑ (กึ่งกลางของ t2) เลื่อนขวา EAP และเพิ่มค่า count i

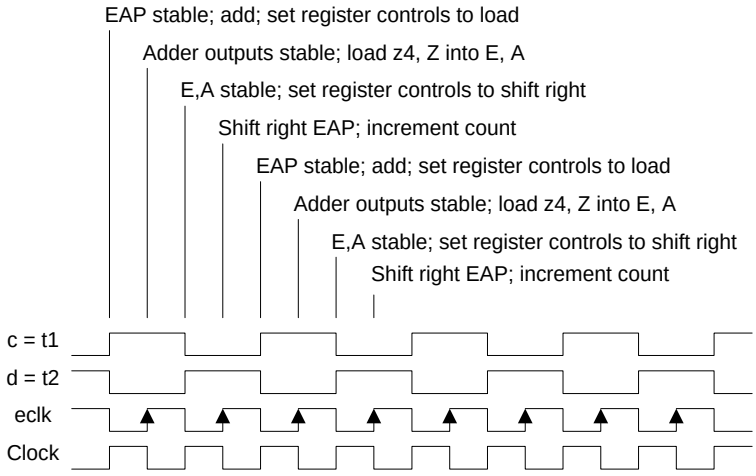
$$S_{i+1} = 2^{-1}Z$$
$$i = i + 1$$

ถ้า count i < 4 repeat state c และ d

ถ้า count i = 4 goto state a

ผลคูณ BP จะพบใน Register AP ที่binary point ที่ขวาสุดของ register P

เขียน timing diagram ของ state c และ d ได้ดังรูปที่ 5 ซึ่ง S1, S0 จะถูก Set เมื่อสัญญาณ Clock เป็น High ส่วน E, A, P จะถูก Load และ Shift ที่ขอบขาขึ้นของ Clock



รูปที่ 5 Timing Diagram

micro operation ของ state a, b และ timing diagram ของ state c และ d สามารถนำมาเขียนเป็น ตารางที่ 2

ตารางที่ 2 สัญญาณควบคุม Data Path ซึ่งสร้างโดย output Decoder

State	Function	Micro operation	Control Signals	S1S0 P	S1S0 A
a (00)					
b (01)	Clear Acc Clear Counter Load ตัวตั้ง, ตัวคูณ	Load B Load P Enable X, Y Carry-in = 0	clr-A = $\overline{A}$ B clr-ct = $\overline{A}$ B clk-B = $\overline{A}$ B . eclk clk-P = $\overline{A}$ B . eclk Exy = A B . p_0 Carry-in = 0	11	11
c (11)	Add Z = Bp_i + A Set S1S0 to load A Store $z_4 \rightarrow$ E $z \rightarrow$ A	Load E Load A	clk-E = AB . eclk clk-A = AB .eclk		11
d (10)	Set s1S0 to shift right Shift right EAP Increment Count	Shift right EAP Inc ct	clk-A = A $\overline{B}$. eclk clk-P = A B . eclk inc-$\overline{ct}$ = A $\overline{B}$.ct4.eclk	01	01

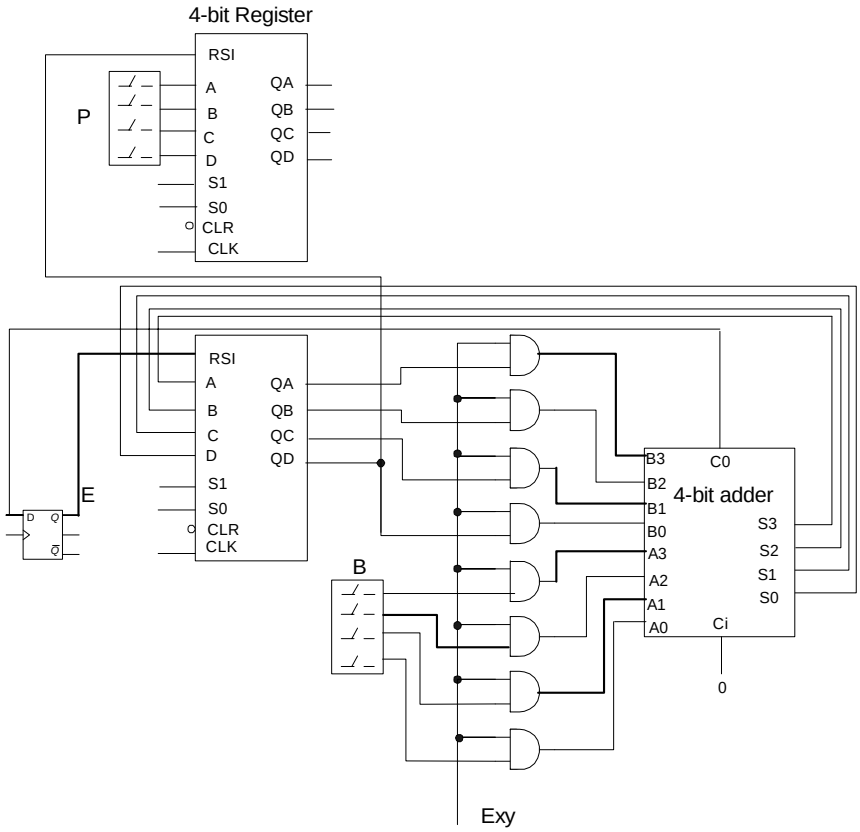
สัญญาณควบคุมต่างๆ หาได้จาก micro operation ดังนี้

ctr-A = $\overline{A}$ B
clr-ct = $\overline{A}$ B
clk-B = $\overline{A}$ B . eclk
clk-E = A B . eclk
clk-A = (A B . p_0 + $\overline{A}$ B) eclk
clk-P = ($\overline{A}$ B + A$\overline{B}$) eclk
Exy = A B . p_0

carry-in = 0
S1A = A B
S1P = $\overline{A}$
S0 = 1
inc ct = A$\overline{B}$. $\overline{ct}$4 . eclk

step 4 : Realization

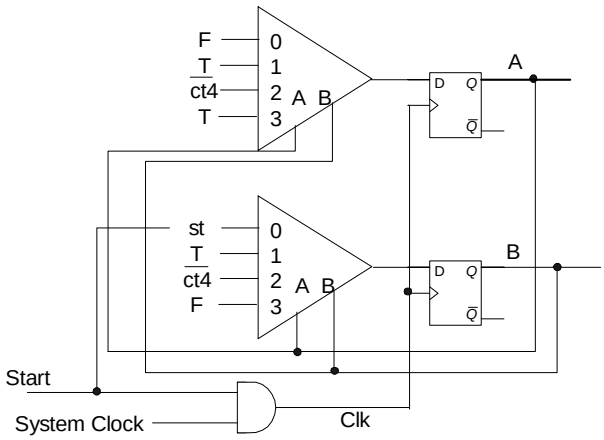
a. Process Data Path ใช้ Block Diagram ในรูปที่ 3 เขียน Logic Diagram ได้ดังรูปที่ 6



รูปที่ 6 Logic Diagram ของ Data Path

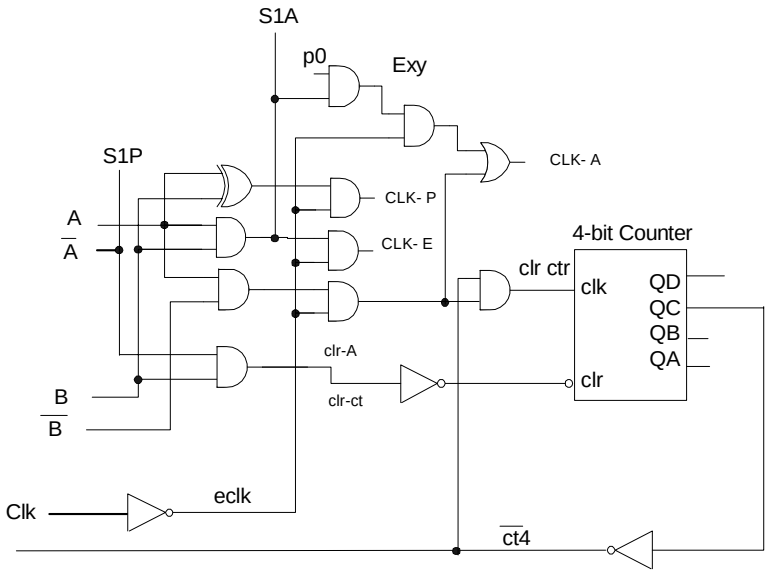
b. Control Unit

(1) Next-State Generator จาก NS Map สามารถสร้าง Next-State generator ได้ดังรูปที่ 7



รูปที่ 7 Next-State Generator

(2) Output Decoder. จาก logic equation ของสัญญาณควบคุม สามารถเขียน Logic Diagram ได้ดังรูป ที่ 9



รูปที่ 9 Output Decoder