บทที่ 4

การคำนวณ 2-D IDCT โดยใช้ Multiply Accumulator

ในบทนี้จะกล่าวถึงการออกแบบวงจรคำนวณ 2-D IDCT โดยใช้ MAC (Multiply Accumulator) คำนวณ 1-D IDCT 2 ขั้นตอนด้วยกัน โดยจะนำเสนอแผนภาพบล็อกของวงจรและอธิบายการทำงานของวงจรโดยการยกตัวอย่างข้อมูล

4.1 การออกแบบวงจร 2-D IDCT

การออกแบบ 2-D IDCT ด้วย MAC นั้นใช้โครงสร้างการคำนวณ โดยการแตกสมการ 2-D IDCT ออกเป็นการคำนวณ 1-D IDCT แต่ละหลักก่อน แล้วจึงนำผลลัพธ์ไปคำนวณในแต่ละแถวต่อไป ซึ่งจากสมการ (7) ในบทที่ 3

$$[s(x,y)] = [C(x,u)][S(u,v)][C(y,v)]^T \quad \text{..... (1)}$$

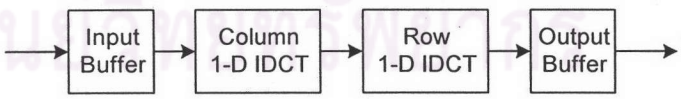
ทำการคำนวณในแต่ละหลักก่อน

$$[A(x,v)] = [C(x,u)][S(u,v)] \quad \text{..... (2)}$$

และผลลัพธ์ที่ได้ คือ $[A(x,v)]$ไปคำนวณอีกครั้งหนึ่ง ในแต่ละแถว

$$[s(x,y)] = [A(x,v)][C(y,v)]^T \quad \text{..... (3)}$$

ซึ่งขั้นตอนการทำงานเป็นไปตามผังงาน

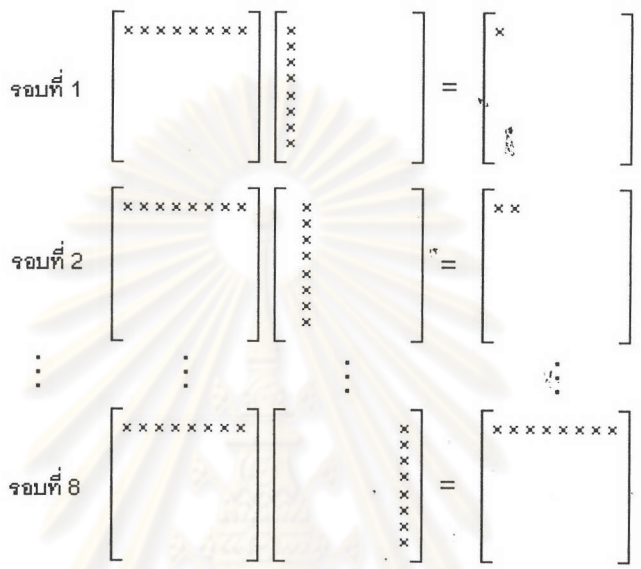


รูปที่ 4.1 ผังงานของ 2-D IDCT

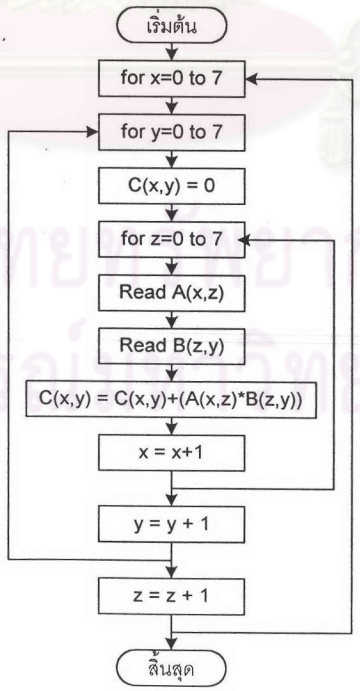
การคำนวณหาผลลัพธ์การคูณกันของเมตริกซ์ คือการเอาข้อมูลแถวของตัวตั้งนำไปคูณกับข้อมูลหลักของตัวคูณที่ตำแหน่งเดียวกันและรวมผลคูณเข้าด้วยกัน ดังตัวอย่างสมการ (4) ซึ่งเป็นการคูณกันของเมตริกซ์ขนาด 2 x 2 แต่การคำนวณ 1-D IDCT จะใช้การคำนวณกันของเมตริกซ์ขนาด 8 x 8 โดยใช้หลักการเดียวกัน

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix} \quad \text{..... (4)}$$

เราใช้ MAC ทำการหาค่าการคูณกันของเมตริกซ์ขนาด 8 x 8 ซึ่ง MAC แต่ละชุดจะต้องคำนวณทั้งหมด 8 รอบด้วยกันจึงจะได้ผลลัพธ์ 1 แถว (มี 8 ค่า) ดังแสดงในรูปที่ 4.2 ดังนั้นการคำนวณเพื่อให้ได้ข้อมูลครบทั้ง 8 แถวจะต้องคำนวณทั้งหมด 64 รอบด้วยกัน สำหรับขั้นตอนการทำงานของ MAC จนได้ผลลัพธ์ทั้งสิ้นแสดงไว้ในรูปที่ 4.3

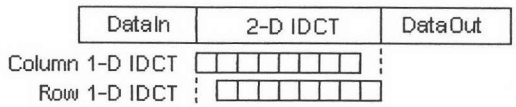


รูปที่ 4.2 การหาผลลัพธ์จากการคูณกับของเมตริกซ์



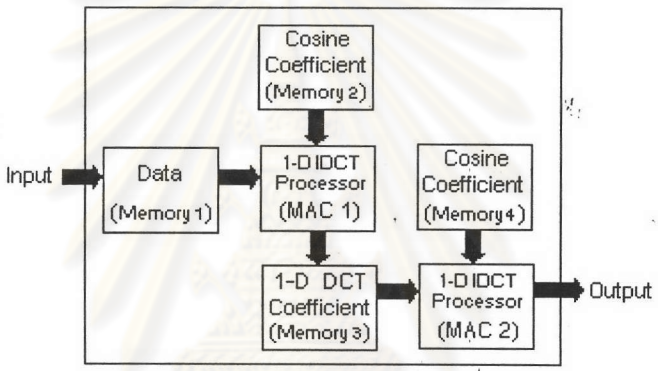
รูปที่ 4.3 ผังการคำนวณ 1-D IDCT

แต่ในทางปฏิบัติเมื่อได้ผลลัพธ์ 1 แถวแล้ว เราสามารถนำผลลัพธ์ที่ได้ไปคำนวณในขั้นตอนที่ 2 ได้โดยที่ไม่จำเป็นต้องรอให้ผลลัพธ์มาครบทั้ง 8 แถวโดยการใช้ MAC 2 ชุด ดังแสดงในผังเวลาในรูปที่ 4.4 ซึ่งเป็นการประหยัดเวลาทำให้สามารถทำงานได้พร้อมกันและเป็นอิสระต่อกัน

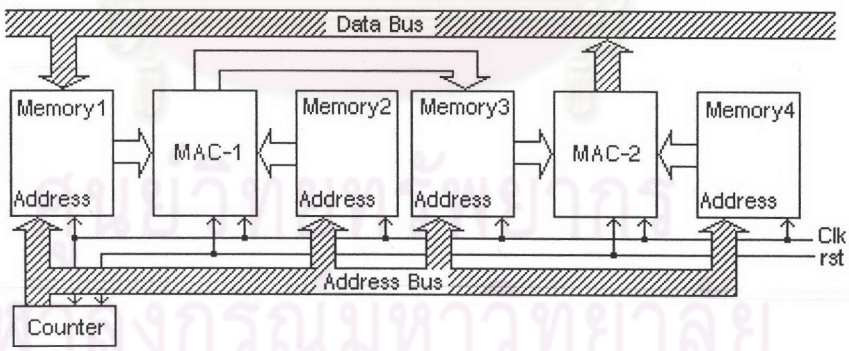


รูปที่ 4.4 ผังเวลาสำหรับการคำนวณ 2-D IDCT

4.2 แผนภาพบล็อกของวงจร



รูปที่ 4.5 แผนภาพบล็อก 2-D IDCT โดยใช้ MAC 2 ตัว



รูปที่ 4.6 การเชื่อมต่อทางตรรกของวงจร 2-D IDCT โดยใช้ MAC 2 ตัว

โครงสร้างวงจรประกอบด้วย MAC 2 ตัว ซึ่งข้อมูลเข้าจะใช้สองส่วนคือเป็นตัวตั้งและตัวคูณโดยจัดเก็บไว้ในหน่วยความจำ (memory) การทำงานเริ่มจากนำข้อมูลจากอินพุต (สัมประสิทธิ์ DCT) ที่เก็บใน Memory 1 เป็นตัวตั้งและตัวคูณคือสัมประสิทธิ์โคไซน์จาก Memory 2 ผลลัพธ์จาก MAC 1 จะถูกเก็บใน Memory 3 เพื่อเป็นตัวตั้งของ MAC 2 ส่วนตัวคูณของ MAC 2 คือสัมประสิทธิ์โคไซน์จาก Memory 4 และผลลัพธ์จาก MAC 2 จะถูกเก็บเป็นเอาต์พุตต่อไป

4.3 วงจร Multiply Accumulator

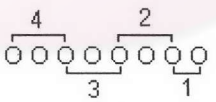
MAC คือวงจรที่ทำหน้าที่บวกผลลัพธ์ทั้งหมดที่ได้จากการคูณกันของข้อมูลเข้าแต่ละคู่ วงจรนี้ออกแบบสำหรับการคำนวณ 1-D IDCT จึงเป็นการรวมกันของการคูณทั้งหมด 8 คู่ด้วย ดังสมการ (2) ที่แสดงไว้ในบทที่ 3

Inverse Discrete Cosine Transform :

$$s(x) = \sum_{u=0}^{7} \frac{C(u)}{2} S(u) cos\left(\frac{(2x+1)u\pi}{16}\right) \quad \text{..... (5)}$$

กระบวนการทำงานของวงจรจะเริ่มจากการคูณกันของข้อมูลคู่ที่ 1 และนำผลลัพธ์ที่ได้ไปพักไว้ในที่พักข้อมูล จากนั้นจะทำการคูณข้อมูลคู่ที่ 2 และนำผลลัพธ์ข้อมูลคู่ที่ 2 ไปรวมกับข้อมูลในที่พักข้อมูล และนำผลรวมกลับไปเก็บในที่พักข้อมูลอีกครั้งหนึ่งเพื่อรอการรวมกับผลคูณของข้อมูลคู่ที่ 3 คู่ที่ 4 คู่ที่ 5 คู่ที่ 6 คู่ที่ 7 และ คู่ที่ 8 ซึ่งรายละเอียดของวงจรได้แสดงไว้ในรูป 4.12

ข้อมูลเข้าของวงจรนี้ซึ่งเป็นตัวตั้งขนาด 12 บิต และตัวคูณมีขนาด 8 บิต โดยข้อมูลออกซึ่งเป็นผลลัพธ์ที่ได้จากการคูณแต่ละคู่มีขนาด 20 บิต ส่วนที่พักข้อมูลสำหรับการรวมผลคูณมีขนาด 24 บิต ยูนิตของการคูณเราจะใช้ขั้นตอนวิธีของบูธที่ปรับปรุงใหม่ (modified Booth's algorithm) ซึ่งจะได้ผลคูณย่อย (partial product) ออกมาเพียง 4 ชุดเท่านั้นสำหรับตัวคูณที่มีขนาด 8 บิต โดยตัวคูณจะถูกแบ่งออกเป็น 4 กลุ่มด้วยกันดังแสดงไว้ในรูปที่ 4.7 แต่ละกลุ่มมีขนาด 3 บิตโดยกลุ่มแรกจะใช้ข้อมูลบิตที่ 1 และ 0 ส่วนบิตสุดท้ายของกลุ่มมีค่าเป็น 0 เสมอ กลุ่มที่ 2 ใช้ข้อมูลบิตที่ 3 2 และ 1 กลุ่มที่ 3 ใช้ข้อมูลบิตที่ 5 4 และ 3 และกลุ่มที่ 4 ใช้ข้อมูลบิตที่ 7 6 และ 5 โดยข้อมูลแต่ละกลุ่มดังที่กล่าวมาแล้วจะใช้ในการหาค่าของผลคูณย่อยทั้ง 4 ชุดโดยเทียบจากตารางที่ 4.1

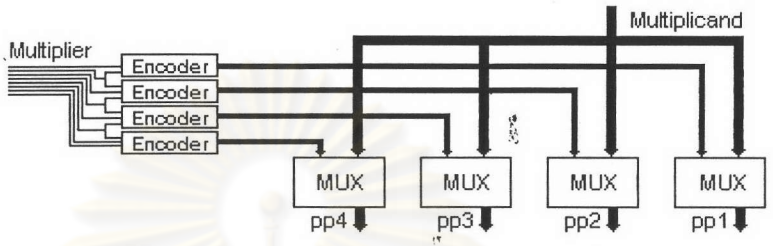


รูปที่ 4.7 การแบ่งกลุ่มของตัวคูณออกเป็น 4 กลุ่ม

Multiplier Bits	Selection	Encoder Output
000	+0	00100
001	+ Multiplicand	01000
010	+ Multiplicand	01000
011	+2 Multiplicand	10000
100	-2 Multiplicand	00001
101	- Multiplicand	00010
110	- Multiplicand	00010
111	-0	00100

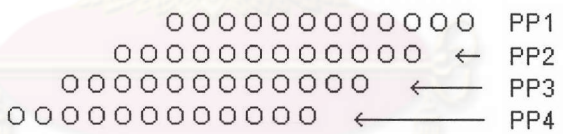
ตารางที่ 4.1 การเลือกผลคูณย่อยจากบิตข้อมูลตัวคูณ

ข้อมูลทั้ง 4 กลุ่มที่แบ่งได้จากตัวคูณนั้นจะผ่านเข้าวงจรเข้ารหัสบูธ (Booth Encoder) ผลลัพธ์จากตัวเข้ารหัสบูธแสดงไว้ในตาราง 4.1 ผลลัพธ์จากตัวเข้ารหัสบูธซึ่งเป็นเลข 5 บิตจะถูกใช้เป็นตัวเลือก (selector) ในวงจรตัวมัลติเพลกซ์ เพื่อกำหนดว่าค่าของผลคูณย่อยจะเป็นกี่เท่าของตัวตั้ง ดังแสดงรายละเอียดของวงจรในรูปที่ 4.8

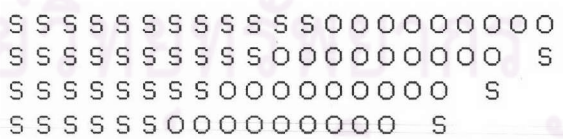


รูปที่ 4.8 วงจรการเข้ารหัสและตัวมัลติเพลกซ์

เมื่อได้ผลคูณย่อยทั้ง 4 แล้วจะนำผลที่ได้เอาไปวางตามไว้ในตำแหน่งที่แสดงในรูป 4.9 (a) แล้วนำมารวมกัน การหาค่าลบของเลขส่วนเติมเต็ม 2 (two's complement) คือการเปลี่ยนค่าของ 0 เป็น 1 1 เป็น 0 และบวกด้วย 1 เราสามารถประหยัดวงจรบวกด้วย 1 โดยการติดค่า 1 เอาไว้ก่อน ดังแสดงในรูปที่ 4.9 (b) 'O' คือบิตข้อมูล (data bit) 'S' คือ บิตเครื่องหมาย (sign bit) 'S' ที่ห้อยอยู่ด้านล่างจะเป็น "1" ในกรณีที่ผลคูณย่อยนั้นเป็น -1 หรือ -2 เท่า ของตัวตั้ง (multiplicand) เพื่อเอาไปรวมกับค่าผกผันทำให้เป็นเลขส่วนเติมเต็ม 2 และจะเป็น "0" ในกรณีอื่นที่เหลือ



(a)



(b)

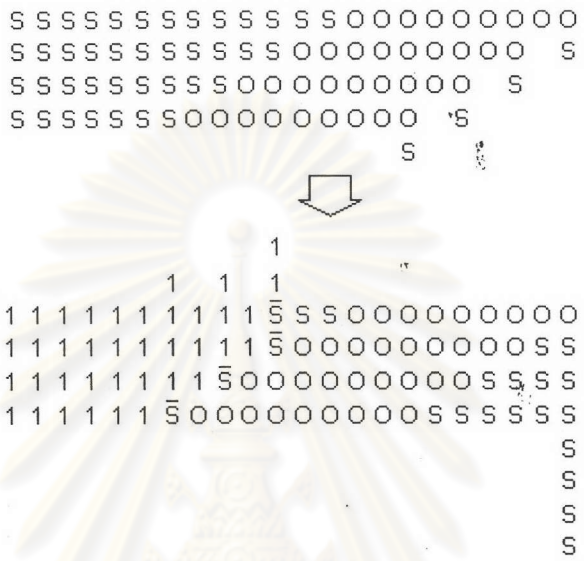
รูปที่ 4.9 ตำแหน่งการวางของผลคูณย่อยทั้ง 4

เราสามารถลดส่วนขยายของบิตเครื่องหมาย (sign extension) ('S' bit) จากสมการ

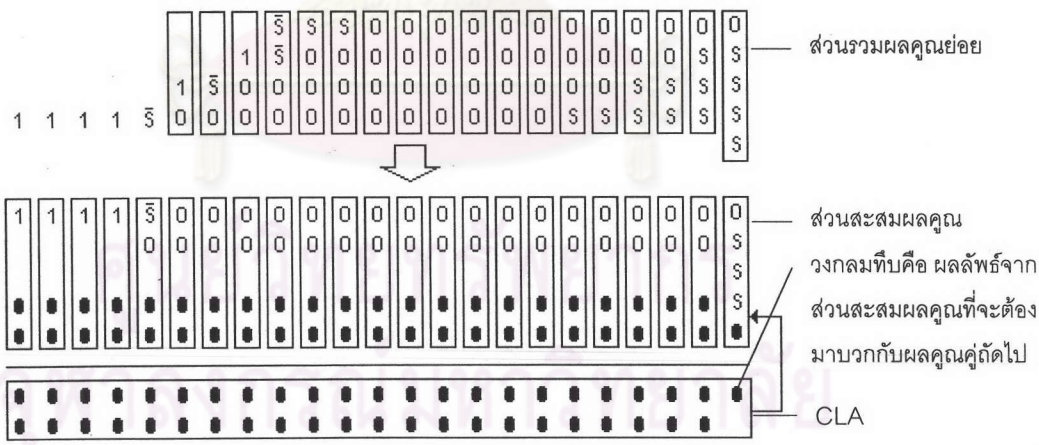
$$ss..ss = 11..11 + \bar{s} \quad \text{.....(6)}$$

และทำการเลื่อน 'S' ไปไว้ช่องว่างในส่วนหลัง นอกจากบิตนัยสำคัญต่ำสุด (Least Significant Bit) แล้วเราจะได้ความสูงของจำนวนบิตมากที่สุดเป็น 4 แทนที่จะเป็น 5 แต่ถึงแม้ว่า LSB จะมีความสูงถึง 8 เราใช้ 5 บิตในส่วนของการรวมผลคูณย่อยส่วนอีก 3 บิตที่เหลือใส่ในส่วนของการสะสมผลคูณดังแสดงไว้ในรูปที่ 4.10

สำหรับผลรวมของผลคูณย่อย (partial product reduction) ใช้ 4-2 Adder สำหรับ 15 บิตล่าง ส่วนที่เหลือ 3 บิตใช้ 3-2 Adder ในส่วนของตัวสะสมผลคูณ (accumulator) 19 บิต ล่างใช้ 4-2 Adder และ 4 บิตบนใช้ 3-2 Adder สุดท้ายใช้วงจรการทดมองล่วงหน้า (Carry Look Ahead) ในการรวมผลลัพธ์ขั้นสุดท้ายจาก 4-2 Adder และ 3-2 Adder ให้ความสูงเหลือเพียงบิตเดียวดังแสดงในรูปที่ 4.11



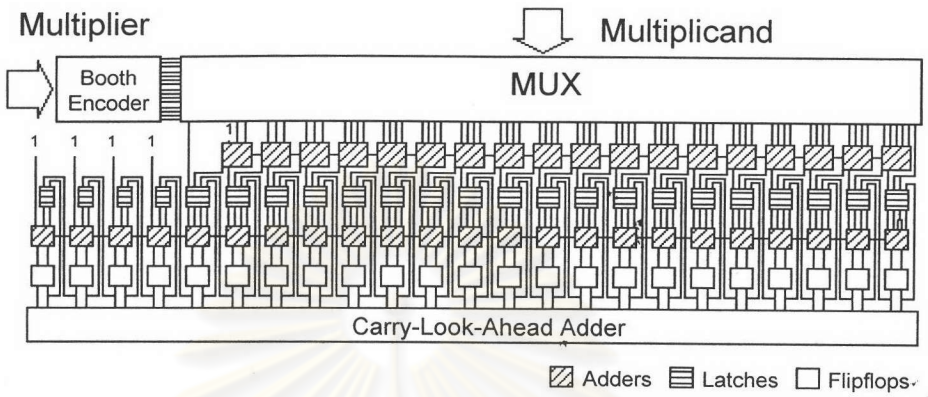
รูปที่ 4.10 ผลคูณย่อย และการลดรูปของส่วนขยายบิตเครื่องหมาย



รูปที่ 4.11 การรวมผลคูณย่อยและการสะสมผลคูณ โดยใช้ 4-2 และ 3-2 Adders

รูปที่ 4.12 วงจรสำเร็จของ MAC ซึ่งประกอบด้วยวงจรเข้ารหัสบูธและตัวมัลติเพลกซ์ สำหรับการเลือกผลคูณย่อยทั้ง 4 การหาผลคูณคือการรวมผลคูณย่อยทั้งสี่โดยใช้วงจร 4-2 Adder และ 3-2 Adder ซึ่งเราสามารถที่จะบวกเลขทุกหลักได้พร้อมๆกัน หลังจากนั้นผลคูณจะส่งไปเก็บไว้ในบัฟเฟอร์ (วงจร latch) ก่อนผ่าน

ค่าไปบวกกับค่าในฟลิบฟลอบโดยวงจร 4-2 Adder และ 3-2 Adder ที่นำผลคูณไปเก็บในบัฟเฟอร์ก่อนเพราะจะทำให้สามารถบวกทั้งสองขั้นตอนได้ในเวลาเดียวกัน



รูปที่ 4.12 วงจร Multiply Accumulator

4.4 กระบวนการทำงานของวงจร

ในหัวข้อนี้จะอธิบายกระบวนการทำงานของวงจรอย่างละเอียดโดยการยกตัวอย่าง ข้อมูลเข้าเป็นสัมประสิทธิ์ DCT 1 บล็อกขนาด 8 x 8 และสัมประสิทธิ์โคไซน์ขนาด 8 x 8 เท่ากัน ซึ่งสัมประสิทธิ์โคไซน์นั้นเราสามารถหาได้จากการแตกสมการ 1-D IDCT สมการ (5) ออกเป็นสมการย่อยๆ 8 สมการ

$$s(x) = \sum_{u=0}^{7} \frac{C(u)}{2} S(u) cos\left(\frac{(2x+1)u\pi}{16}\right) \quad \text{..... (5)}$$

โดยกำหนดให้

$C(u) = \frac{1}{\sqrt{2}}$ สำหรับ $u = 0$

$C(u) = 1$ สำหรับ $u > 0$

สามารถแยกออกเป็นสมการทั้ง 8 ดังต่อไปนี้ (<u>หมายเหตุ</u> $C_k = Cos\left(\frac{k\pi}{16}\right)$)

$$s(0) = \frac{1}{2}\left(\frac{1}{\sqrt{2}} S(0) + C_1S(1) + C_2S(2) + C_3S(3) + C_4S(4) + C_5S(5) + C_6S(6) + C_7S(7)\right) \quad \text{..... (7)}$$

$$s(1) = \frac{1}{2}\left(\frac{1}{\sqrt{2}} S(0) + C_3S(1) + C_6S(2) - C_7S(3) - C_4S(4) - C_1S(5) - C_2S(6) - C_5S(7)\right) \quad \text{..... (8)}$$

$$s(2) = \frac{1}{2}\left(\frac{1}{\sqrt{2}} S(0) + C_5S(1) - C_6S(2) - C_1S(3) - C_4S(4) + C_7S(5) + C_2S(6) + C_3S(7)\right) \quad \text{..... (9)}$$

$$s(3) = \frac{1}{2}\left(\frac{1}{\sqrt{2}}S(0) + C_7S(1) - C_2S(2) - C_5S(3) + C_4S(4) + C_3S(5) - C_6S(6) - C_1S(7)\right) \quad \ldots\ldots\ldots (10)$$

$$s(4) = \frac{1}{2}\left(\frac{1}{\sqrt{2}}S(0) - C_7S(1) - C_2S(2) + C_5S(3) + C_4S(4) - C_3S(5) - C_6S(6) + C_1S(7)\right) \quad \ldots\ldots\ldots (11)$$

$$s(5) = \frac{1}{2}\left(\frac{1}{\sqrt{2}}S(0) - C_5S(1) - C_6S(2) + C_1S(3) - C_4S(4) - C_7S(5) + C_2S(6) - C_3S(7)\right) \quad \ldots\ldots\ldots (12)$$

$$s(6) = \frac{1}{2}\left(\frac{1}{\sqrt{2}}S(0) - C_3S(1) + C_6S(2) + C_7S(3) - C_4S(4) + C_1S(5) - C_2S(6) + C_5S(7)\right) \quad \ldots\ldots\ldots (13)$$

$$s(7) = \frac{1}{2}\left(\frac{1}{\sqrt{2}}S(0) - C_1S(1) + C_2S(2) - C_3S(3) + C_4S(4) - C_5S(5) + C_6S(6) - C_7S(7)\right) \quad \ldots\ldots\ldots (14)$$

ซึ่ง $\frac{1}{\sqrt{2}}$ นั้นเท่ากับ $Cos\left(\frac{4\pi}{16}\right)$ ส่วนค่า $\frac{1}{2}$ สามารถที่จะคำนวณทีหลังได้ ดังนั้นเมตริกซ์ของสัมประสิทธิ์โคไซน์จะได้จากการดึงตัวคูณของข้อมูลดังแสดงในรูปที่ 4.13 (a) ค่าของ C_1 - C_7 จะมีค่าอยู่ระหว่าง 0.0 - 1.0 โดย C_1 = 0.9878528 C_2 = 0.923879532 C_3 = 0.8314696123 C_4 = 0.707106781 C_5 = 0.555570233 C_6 = 0.382683432 และ C_7 = 0.195090322 เนื่องจากว่าวงจร MAC ที่ออกแบบขึ้นใช้กับเลขจำนวนเต็มถ้าปัดเศษเลขเหล่านี้ขึ้นเป็นจำนวนเต็ม ค่าจะน้อยเกินไปความผิดพลาดจะสูง ดังนั้นจึงจำเป็นจะต้องนำเลขเข้าไปคูณ เพื่อทำให้สัมประสิทธิ์เป็นเลขจำนวนเต็ม เลขที่นำเข้าไปคูณจะต้องเป็นเลข 2^N เช่น 2 4 8 16 เพราะเราไม่จำเป็นต้องใช้วงจรหารเลขนั้นออกภายหลังจากการคำนวณ อาศัยเพียงการเลื่อนบิตเท่านั้น เช่น การหารด้วย 4 คือตัดบิตที่ 1 และ 0 ทิ้ง เลขที่เราต้องการคือเลขขนาด 8 บิตซึ่งค่าระหว่าง -128 และ 127 ที่จะใช้ในวงจร MAC ดังนั้นตัวเลขที่เหมาะสมที่สุดที่จะนำไปคูณกับสัมประสิทธิ์โคไซน์คือ 128 เพราะผลลัพธ์จะไม่เกินค่า -128 และ 127 อีกทั้งเป็นเลข 2^N ที่มากที่สุด ทำให้ข้อมูลใกล้เคียงค่าเดิมมากที่สุด

$$\begin{bmatrix}
C_4 & C_1 & C_2 & C_4 & C_3 & C_5 & C_6 & C_7 \\
C_4 & C_3 & C_6 & -C_4 & -C_7 & -C_1 & -C_2 & -C_5 \\
C_4 & C_5 & -C_6 & -C_4 & -C_1 & C_7 & C_2 & C_3 \\
C_4 & C_7 & -C_2 & -C_4 & C_5 & C_3 & -C_6 & -C_1 \\
C_4 & -C_7 & -C_2 & C_4 & C_5 & -C_3 & -C_6 & C_1 \\
C_4 & -C_5 & -C_6 & C_4 & -C_1 & -C_7 & C_2 & -C_3 \\
C_4 & -C_3 & C_6 & C_4 & -C_7 & C_1 & -C_2 & C_5 \\
C_4 & -C_1 & C_2 & -C_4 & C_3 & -C_5 & C_6 & -C_7
\end{bmatrix}$$

(a)

รูปที่ 4.13 (a) เมตริกซ์ของสัมประสิทธิ์โคไซน์

01011011	01111110	01110110	01101010	01011011	01000111	00110001	00011001
01011011	01101010	00110001	11100111	10100101	10000010	10001010	10111001
01011011	01000111	11001111	10000010	10100101	00011001	01110110	01101010
01011011	00011001	10001010	10111001	01011011	01101010	11001111	10000010
01011011	11100111	10001010	01000111	01011011	10010110	11001111	01111110
01011011	10111001	11001111	01111110	10100101	11100111	01110110	10010110
01011011	10010110	01110001	00011001	10100101	01111110	10001010	01000111
01011011	10000010	01110110	10010110	01011011	10111001	00110001	11100111

(b)

รูปที่ 4.13 (b) เมตริกซ์ของสัมประสิทธิ์โคไซน์ ในระบบเลขฐานสอง

ค่าใหม่ที่ได้จะเป็นดังนี้

128 C_1 = 126 แปลงเป็นเลขฐานสอง = 01111110

128 C_2 = 118 แปลงเป็นเลขฐานสอง = 01110110

128 C_3 = 106 แปลงเป็นเลขฐานสอง = 01101010

128 C_4 = 91 แปลงเป็นเลขฐานสอง = 01011011

128 C_5 = 71 แปลงเป็นเลขฐานสอง = 01000111

128 C_6 = 49 แปลงเป็นเลขฐานสอง = 00110001

128 C_7 = 31 แปลงเป็นเลขฐานสอง = 00011001

นำค่าที่ได้เข้าไปแทนสัมประสิทธิ์โคไซน์ในรูปที่ 4.13 (a) ในระบบเลขฐานสอง ก็จะได้เมตริกซ์ดังแสดงในรูป 4.13 (b)

สำหรับสัมประสิทธิ์ DCT ที่จะนำมายกตัวอย่าง แสดงไว้ในรูปที่ 4.14 (a) ในระบบเลขฐานสิบ และรูปที่ 4.14 (b) ในระบบเลขฐานสองที่มีขนาด 12 บิต เมื่อได้ข้อมูลเมตริกซ์ทั้งเมตริกซ์ตัวตั้งและตัวคูณเป็นที่เรียบร้อยแล้ว เราสามารถเริ่มทำการคำนวณ 1-D IDCT ในขั้นตอนที่ 1 ได้ จากสมการ (2)

$$[A(x,v)] = [C(x,u)][S(u,v)]$$

เมตริกซ์ตัวตั้งคือสัมประสิทธิ์โคไซน์ ส่วนเมตริกซ์ตัวคูณคือสัมประสิทธิ์ DCT

$$\begin{bmatrix} 736 & -15 & -19 & -21 & 0 & 0 & 0 & 0 \\ 31 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 25 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 001011100000 & 111111110001 & 111111101101 & 111111101011 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \\ 000000011111 & 000000001111 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \\ 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \\ 000000010101 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \\ 000000010101 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \\ 111111100111 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \\ 000000001101 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \\ 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 & 000000000000 \end{bmatrix}$$

(b)

รูปที่ 4.14 ข้อมูลตัวอย่างในระบบ (a) เลขฐานสิบ (b) เลขฐานสอง

$$A_{(0,0)} = C_{(0,0)}S_{(0,0)}+C_{(0,1)}S_{(1,0)}+ C_{(0,2)}S_{(2,0)}+C_{(0,3)}S_{(3,0)}+ C_{(0,4)}S_{(4,0)}+C_{(0,5)}S_{(5,0)}+ C_{(0,6)}S_{(6,0)}+C_{(0,7)}S_{(7,0)} \text{ .. (15)}$$

เราสามารถหาผลคูณเมตริกซ์ตำแหน่ง (0,0) ได้จากสมการ (15) การคูณกันของเมตริกซ์นั้นไม่มีคุณสมบัติการสลับที่ แต่การคูณเลขเราสามารถสลับที่ตัวตั้งและตัวคูณได้เช่น $C_{(0,0)}S_{(0,0)}$ จะเท่ากับ $S_{(0,0)}C_{(0,0)}$ ดังนั้นเราสามารถเขียนสมการใหม่ได้เป็นดังนี้

$$A_{(0,0)} = S_{(0,0)}C_{(0,0)}+S_{(1,0)}C_{(0,1)}+ S_{(2,0)}C_{(0,2)}+S_{(3,0)}C_{(0,3)}+ S_{(4,0)}C_{(0,4)}+S_{(5,0)}C_{(0,5)}+ S_{(6,0)}C_{(0,6)}+S_{(7,0)}C_{(0,7)} \text{ .. (16)}$$

ที่เราเลือกให้สัมประสิทธิ์โคไซน์เป็นตัวคูณเนื่องจากว่าการออกแบบ MAC นั้นเราหาผลคูณย่อยโดยขั้นตอนของบูธที่ปรับปรุงใหม่ จำนวนผลคูณย่อยจะเท่ากับจำนวนบิตของตัวคูณหารด้วย 2 สัมประสิทธิ์โคไซน์มีขนาดเพียง 8 บิตจำนวนผลคูณย่อยจึงเท่ากับ 4 ค่าเท่านั้น ถ้าเราใช้สัมประสิทธิ์ DCT ซึ่งมีขนาด 12 บิตเป็นตัวคูณจำนวนผลคูณย่อยจะเท่ากับ 6 ค่าการออกแบบวงจรจะยุ่งยากซับซ้อนขึ้น

การทำงานของวงจรเริ่มจาก อ่านค่าตัวคูณตัวแรกคือ $C_{(0,0)}$ เข้าวงจรตัวเข้ารหัสบูธเพื่อแบ่งออกเป็น 4 กลุ่ม แต่ละกลุ่มนำไปเทียบค่าจากตาราง 4.1 เพื่อหาผลคูณย่อยทั้ง 4

$C_{(0,0)}$ = 0 1 0 1 1 0 1 1

กลุ่มที่ 1 คือ 110 pp1 = -1 เท่าของตัวตั้ง

กลุ่มที่ 2 คือ 101 pp2 = -1 เท่าของตัวตั้ง

กลุ่มที่ 3 คือ 011 pp3 = +2 เท่าของตัวตั้ง

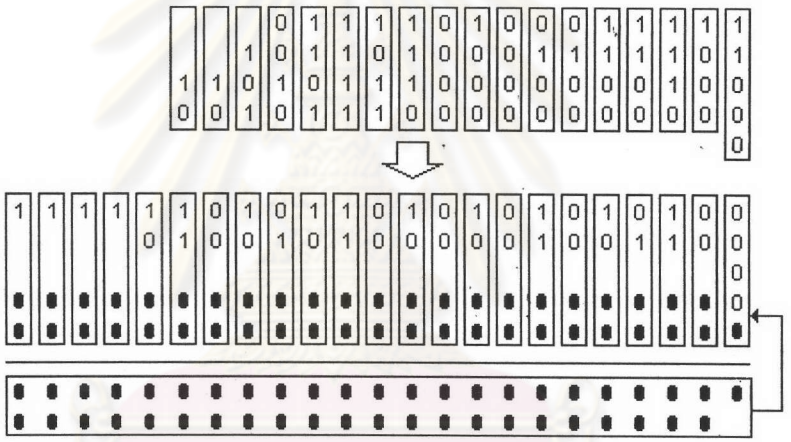
กลุ่มที่ 4 คือ 010 pp4 = +1 เท่าของตัวตั้ง

ตัวตั้งในที่นี้คือ $S_{(0,0)}$ = 001011100000 ซึ่งสามารถคำนวณหาผลคูณย่อยทั้ง 4 ได้ดังรูปที่ 4.15

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 1 1 1 1 1  1
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0  1
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 ,0
                                          0
```

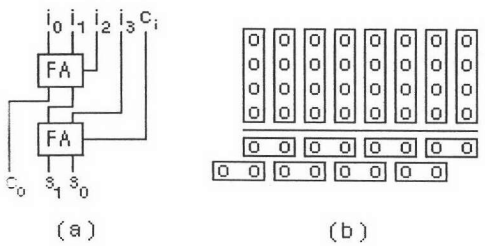
รูปที่ 4.15 ผลคูณย่อยทั้ง 4

และเมื่อผ่านเข้าวงจรบวก (4-2 Adder และ 3-2 Adder) ที่แสดงในรูปที่ 4.11 ได้ผลลัพธ์ดังรูปที่ 4.16



รูปที่ 4.16 ผลลัพธ์ที่ได้ภายหลังบวกผลคูณย่อยทั้ง 4

กรอบสี่เหลี่ยมแทนวงจรบวก ซึ่งเป็น 4-2 Adder 15 บิตล่าง ส่วน 3 บิตบนเป็น 3-2 Adder ที่เลือกใช้วงจรบวกนี้เพราะแต่ละหลักสามารถคำนวณโดยไม่จำเป็นต้องรอตัวทด (Carry Save Adder) เพราะวงจรบวกนี้จะทำงานในสองขั้นตอนด้วยกันคือบิตตัวทดออก (Co) จะถูกคำนวณออกมาในขั้นตอนที่ 1 แต่จะใช้บิตตัวทดเข้า (Ci) ในขั้นตอนที่สอง วงจรบวกทั้งหมดนี้จึงสามารถทำงานไปได้พร้อมๆกัน ผลลัพธ์ของวงจรนี้จะมีขนาดความสูง 2 ชั้น



รูปที่ 4.17 4-2 Adder

ค่าผลคูณที่ได้นี้ยังจะต้องนำไปรวมกับค่าในฟลิปฟลอป (ส่วนล่างสุด) แต่เนื่องจากคู่นี้เป็นการคูณคู่แรก ค่าในฟลิปฟลอบจะถูกตั้งใหม่เป็นศูนย์ การบวกยังคงใช้ 4-2 Adder ที่ 20 บิตล่าง ส่วน 4 บิตบนใช้ 3-2 Adder ผลบวกจะถูกเก็บกลับเข้าไปในฟลิปฟลอป จากนั้นจะทำการอ่านข้อมูลคู่ที่ 2 (ดูสมการ (15)) ตัวตั้งคือ $S_{(1,0)}$ = 000000011111 และ ตัวคูณคือ $C_{(0,1)}$= 01111110 แล้วทำกระบวนการซ้ำดังที่ผ่านมา เมื่อกระทำจนครบ 8 คู่แล้วเราก็จะได้ผลลัพธ์ $A_{(0,0)}$ = 000100100010 แต่เนื่องจากผลลัพธ์ออกมาจาก 4-2 Adder ซึ่งอยู่ในเลขฐานสองที่มีความสูง 2 ชั้น ดังแสดงในรูป 4.16 จึงใช้วงจรบวกให้คำนวณให้เหลือความสูงเพียงชั้นเดียว ผลลัพธ์ที่ได้จะมีขนาด 24 บิต จากในตอนต้นค่าของสัมประสิทธิ์โคไซน์ในรูปของเมตริกซ์ที่เป็นการคำนวณโดยการคูณด้วย 128 ดังนั้นผลลัพธ์ที่ได้จะต้องหารด้วย 128 นั้นคือการตัดบิตทางขวาทิ้งไป 7 บิต เพื่อให้ได้ค่าที่ถูกต้อง และค่าที่เก็บเราจะใช้เพียงแค่ 12 บิต ดังนั้นผลลัพธ์ที่ได้จึงใช้เฉพาะบิตที่ 18-7

การคำนวณจะดำเนินในลักษณะเดียวกันจนได้ผลลัพธ์ครบ 1 แถว ซึ่งเพียงพอที่จะคำนวณ 1-D IDCT ต่อไปได้ โดยจากสมการ (3)

$$\left[s(x,y)\right] = \left[A(x,v)\right]\left[C(y,v)\right]^T \quad \text{..... (3)}$$

$$s_{(0,0)} = A_{(0,0)}C_{(0,0)}+A_{(0,1)}C_{(0,1)}+ A_{(0,2)}C_{(0,2)}+A_{(0,3)}C_{(0,3)}+ A_{(0,4)}C_{(0,4)}+A_{(0,5)}C_{(0,5)}+ A_{(0,6)}C_{(0,6)}+A_{(0,7)}C_{(0,7)} \text{ .. (17)}$$

สังเกตว่าสัมประสิทธิ์โคไซน์เป็นเมตริกซ์สับเปลี่ยน (transposed matrix) เมตริกซ์สัมประสิทธิ์โคไซน์จะใช้เมตริกซ์เดิมในตอนแรกเมตริกซ์โคไซน์เป็นตัวตั้งต้องการใช้ข้อมูลทีละแถวในการคูณเมตริกซ์ ส่วนในขั้นตอนที่สองนั้นเมตริกซ์สัมประสิทธิ์โคไซน์เป็นตัวคูณต้องใช้ข้อมูลทีละหลักในการคูณ แต่เมตริกซ์นี้เป็นเมตริกซ์สับเปลี่ยน ถ้าข้อมูลในเมตริกซ์ยังไม่มีการสลับจากแถวเป็นหลักและจากหลักเป็นแถวแล้วนั้น ยังคงใช้ข้อมูลทีละแถวในการคูณเช่นเดิม ดังแสดงในสมการ (17) ส่วนผลลัพธ์จากการคำนวณ 1-D IDCT ในขั้นตอนแรกจะเป็น เมตริกซ์ตัวตั้งของการคำนวณ 1-D IDCT ของขั้นตอนที่สองนี้ ซึ่งผลลัพธ์ที่ได้แสดงไว้ในรูปที่ 4.18

$$\begin{bmatrix}
290 & 2 & -7 & -8 & 0 & 0 & 0 & 0 \\
265 & 0 & -7 & -8 & 0 & 0 & 0 & 0 \\
261 & -2 & -7 & -8 & 0 & 0 & 0 & 0 \\
251 & -4 & -7 & -8 & 0 & 0 & 0 & 0 \\
277 & -7 & -7 & -8 & 0 & 0 & 0 & 0 \\
269 & -10 & -7 & -8 & 0 & 0 & 0 & 0 \\
219 & -12 & -7 & -8 & 0 & 0 & 0 & 0 \\
256 & -13 & -7 & -8 & 0 & 0 & 0 & 0
\end{bmatrix}$$

(a)

รูปที่ 4.18 ผลลัพธ์จาก 1-D IDCTในระบบเลขฐานสิบ

000100100010	000000000010	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000
000100001001	000000000000	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000
000100000101	111111111110	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000
000011111011	111111111100	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000
000100010101	111111111001	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000
000100001101	111111110110	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000
000011011011	111111110100	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000
000100000000	111111110011	111111111001	111111111000	000000000000	000000000000	000000000000	000000000000

(b)

รูปที่ 4.18 ผลลัพธ์จาก 1-D IDCTในระบบเลขฐานสอง

ในการคำนวณ 1-D IDCT ในขั้นตอนที่สองจะเป็นการคำนวณหาผลลัพธ์ขั้นสุดท้าย คือค่าของสีของจุดบนภาพทั้ง 64 จุด การคำนวณจะไม่แตกต่างจาก 1-D IDCT ในขั้นที่หนึ่ง ผิดกันตรงที่อ่านข้อมูลจากคนละที่กัน ซึ่งการหาข้อมูลแต่ละจุดเป็นไปตามสมการ (3) การคำนวณเริ่มจากอ่านค่าสัมประสิทธิ์โคไซน์ตำแหน่ง (0,0) ซึ่งเท่ากับค่า 01011011 ผ่านเข้าวงจรตัวเข้ารหัสบูธเพื่อหาค่าผลคูณย่อยทั้ง 4 ค่า (เราจะใช้ สัมประสิทธิ์โคไซน์เป็นตัวคูณเสมอในวงจร MAC ไม่ว่าเมตริกซ์นี้จะเป็นตัวตั้งหรือตัวคูณในสมการ) หลังจากนั้นทำการอ่านเมตริกซ์ A ที่ตำแหน่ง (0,0) ซึ่งเท่ากับ 000100100010 (290) นำผลลัพธ์จากตัวเข้ารหัสบูธเข้าตัวมัลติเพลกซ์ก็จะได้ผลคูณย่อยทั้ง 4 ดำเนินกระบวนการเหมือน 1-D IDCT ในขั้นตอนแรกจนกระทั่งได้ผลลัพธ์ครบทั้ง 64 ค่า ดังแสดงในรูปที่ 4.19 เป็นอันเสร็จสิ้นกระบวนการ เช่นเดียวกับในขั้นตอนแรกผลลัพธ์จะเก็บเพียง 12 บิต (บิตที่ 18-7)

97	103	108	108	103	99	100	102
87	93	99	99	95	91	92	94
85	91	97	98	93	90	91	93
80	87	93	94	90	87	88	91
88	95	101	103	100	97	99	101
84	90	98	100	97	95	97	100
65	72	79	82	80	78	80	83
78	85	92	95	93	92	94	97

(a)

000001100001	000001100111	000001101100	000001101100	000001100111	000001100011	000001100100	000001100110
000001010111	000001011101	000001100011	000001100011	000001011111	000001011011	000001011100	000001011100
000001010101	000001011011	000001100001	000001100010	000001011101	000001011010	000001011011	000001011101
000001010000	000001010111	000001011101	000001011110	000001011010	000001010111	000001011000	000001011011
000001011000	000001011111	000001100101	000001100111	000001100100	000001100001	000001100011	000001100101
000001010100	000001011010	000001100010	000001100100	000001100001	000001011111	000001100001	000001100100
000001000001	000001001000	000001001111	000001010010	000001010000	000001001110	000001010000	000001010011
000001001110	000001010101	000001011100	000001011111	000001011101	000001011100	000001011110	000001100001

(b)

รูปที่ 4.19 ผลลัพธ์ขั้นสุดท้าย ในระบบ (a) เลขฐานสิบ (b) เลขฐานสอง

4.5 สรุป

ในบทนี้ได้กล่าวถึงการออกแบบวงจรคำนวณ 2-D IDCT โดยการแตกเป็นการคำนวณ 1-D IDCT 2 ขั้นตอน ซึ่งอยู่ในรูปการคูณกันของเมตริกซ์สัมประสิทธ์ DCT กับเมตริกซ์สัมประสิทธิ์โคไซน์ และใช้ MAC ในการคำนวณผลคูณเมตริกซ์ วงจร MAC ที่ออกแบบใช้ขั้นตอนวิธีของบูธซึ่งทำให้มีจำนวนผลคูณย่อยเพียงครึ่งหนึ่งของจำนวนบิตตัวคูณ และการใช้ไพป์ไลน์เพื่อให้สามารถคำนวณผลรวมผลคูณย่อยและตัวสะสมผลคูณได้พร้อมกัน สุดท้ายเป็นการยกตัวอย่างข้อมูลเพื่ออธิบายการทำงานของวงจรตั้งแต่เริ่มต้นจนได้เป็นเมตริกซ์ผลลัพธ์ ในบทต่อไปจะกล่าวถึงวงจรการคำนวณ 1-D IDCT อีกแบบหนึ่ง ที่ลดทอนการคำนวณส่วนที่ซ้ำซ้อนลง ทำให้สามารถหาผลลัพธ์ได้เร็วขึ้น