บทที่ 3

การออกแบบการถอดรหัสแอลดีพีซี

ในบทนี้จะกล่าวถึงหลักวิธีการออกแบบตัวถอดรหัสแอลดีพีซีในการออกแบบนั้นจะมีพื้นฐานการออกแบบอยู่หลายรูปแบบแล้วแต่ปัจจัยหลักของผู้ออกแบบในส่วนการออกแบบพื้นฐานของเรานั้นจะมีปัจจัยหลักคือ ความเร็วในการถอดรหัสและพลังงานที่ใช้ โดยการใช้FPGAจำนวนเกตต่ำๆ หรือใช้เวลาในการทำงานโดยรวมน้อย จะส่งผลทำให้ระบบมีการบริโภคพลังงานต่ำลงไปด้วย โดยการออกแบบของเรานั้นพยายามให้ระบบมีการบริโภคพลังงานที่ต่ำที่สุดและขนาดของฮาร์ดแวร์ที่มีขนาดเล็กที่สุด แต่การออกแบบนั้นต้องคำนึงถึงอัตราข้อมูลและประสิทธิภาพที่ได้โดยผลลัพธ์ที่ได้ต้องเป็นไปตามความต้องการของระบบหรือมาตราฐานในส่วนนี้เราได้นำเสนอเทคนิคการออกแบบเพื่อเพิ่มความเร็วในการถอดรหัสให้เสร็จเร็วยิ่งขึ้น และยังคงจำกัดจำนวนหน่วยประมวลผลไม่ให้มีมากจนเกินไป โดยผลลัพธ์ของการออกแบบนี้จะทำให้หน่วยประมวลผลของ CNU และ VNU ทำงานซ้อนกันในช่วงขณะหนึ่ง

ส่วนสำคัญของการถอดรหัสแอลดีพีซีคือ เมตริกซ์พาริตี้เช็ค H โดยประสิทธิภาพการถอดรหัสขึ้นอยู่กับเมตริกซ์นี้ ดังนั้นการจะทำให้ฮาร์ดแวร์มีความซับซ้อนน้อยลงหรือเพิ่มความเร็วในการทำงานมากยิ่งขึ้นจะต้องนำเอาเมตริกซ์พาริตี้เช็คไปปรับปรุง แต่เนื่องจากการถอดรหัสแอลดีพีซีของเรานั้นจะนำไปใช้บนการสื่อสารไร้สายในพื้นฐานของ IEEE 802.11nทำให้ไม่สามารถแก้ไขเมตริกซ์ได้เพราะได้มีการกำหนดเมตริกซ์พื้นฐานสำหรับการเข้ารหัสไว้แล้ว ถึงอย่างไรก็ตาม ยังมีการนำเสนอวิธีการปรับปรุงเมตริกซ์โดยไม่ส่งผลต่อการเข้ารหัสแอลดีพีซีและประสิทธิในการถอดรหัส หนึ่งในวิธีนั้นก็คือวิธีการสลับลำเมตริกซ์พาริตี้เช็ค

3.1 การสลับลำดับเมตริกซ์พาริตี้เช็ค(Reordering Parity Check Matrix)

จากพื้นฐานเมตริกซ์พาริตี้เช็ค Hของ IEEE 802.11n ที่มีขนาดของคำรหัส 648 bits อัตรารหัส 0.5 ตามรูปที่ 3.1จะประกอบไปด้วยบล็อกช่องว่าง บล็อกเลขศูนย์และบล็อกเลขจำนวนเต็มที่ไม่ใช่ศูนย์ ซึ่งจะมีทั้งหมด 12 แถว 24 หลัก ในบล็อกช่องว่าง (-) จะสามารถขยายองค์ประกอบได้ว่ามีขนาดของเมตริกซ์ศูนย์เท่ากับ $s*s$ ในองค์ประกอบของบล็อกศูนย์จะเป็นรูปแบบของเมตริกซ์ที่ถูกเลื่อนวนขนาด $s*s$ ส่วนองค์ประกอบของบล็อกเลขจำนวนเต็มที่ไม่ใช่ศูนย์จะเป็นเมตริกซ์ที่ถูกเลื่อนวนโดยถูกระบุตำแหน่งเริ่มเลื่อนบิตด้วยเลขจำนวนเต็มที่แสดงอยู่มีขนาด $s*s$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	-	-	-	0	0	-	-	0	-	-	0	1	0	-	-	-	-	-	-	-	-	-	-
2	22	0	-	-	17	-	0	0	12	-	-	-	-	0	0	-	-	-	-	-	-	-	-	-
3	6	-	0	-	10	-	-	-	24	-	0	-	-	-	0	0	-	-	-	-	-	-	-	-
4	2	-	-	0	20	-	-	-	25	0	-	-	-	-	-	0	0	-	-	-	-	-	-	-
5	23	-	-	-	3	-	-	-	0	-	9	11	-	-	-	-	0	0	-	-	-	-	-	-
6	24	-	23	1	17	-	3	-	10	-	-	-	-	-	-	-	-	0	0	-	-	-	-	-
7	25	-	-	-	8	-	-	-	7	18	-	-	0	-	-	-	-	-	0	0	-	-	-	-
8	13	24	-	-	0	-	8	-	6	-	-	-	-	-	-	-	-	-	-	0	0	-	-	-
9	7	20	-	16	22	10	-	-	23	-	-	-	-	-	-	-	-	-	-	-	0	0	-	-
10	11	-	-	-	19	-	-	-	13	-	3	17	-	-	-	-	-	-	-	-	-	0	0	-
11	25	-	8	-	23	18	-	14	9	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
12	3	-	-	-	16	-	-	2	25	5	-	-	1	-	-	-	-	-	-	-	-	-	-	0

รูปที่3.1เมตริกซ์พาริตี้เช็คขนาด 648 bits อัตรารหัส 1/2 ตามมาตราฐาน IEEE 802.11n

ในการประมวลผลของการถอดรหัส LDPC การทำงานของบิตโหนดของหลักแรกจะต้องรอการทำงานของเช็คโหนดเสร็จเสียก่อน หรืออีกนัยหนึ่งก็คือ การทำงานของเช็คโหนดในแถวแรกจะต้องรอการทำงานของบิตโหนดเสร็จก่อน เพื่อเป็นการลดเวลาว่างในการทำงานระหว่างเช็คโหนดกับบิตโหนด จะสามารถปรับปรุงได้โดยสลับลำดับแถวและหลักของเมตริกซ์ H [4] ซึ่งการสลับลำดับแถวและหลักจะไม่ส่งผลกระทบให้ประสิทธิภาพในการถอดรหัสลดลง คำรหัสที่ถูกสลับลำดับแล้วจะไม่ทำให้ตำแหน่งของแถวเกิดการเปลี่ยนแปลงแต่จะเปลี่ยนเฉพาะตำแหน่งของหลักเท่านั้นที่ถูกสลับตำแหน่งซึ่งอยู่ในระดับบิต

	1	2	3	4	5	6
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	1	0	0	0	0	0

(ก)

	1	2	3	4	5	6
1	0	1	0	0	0	0
2	0	0	1	0	0	0
6	1	0	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
3	0	0	0	1	0	0

(ข)

	2	3	1	5	6	4
1	1	0	0	0	0	0
2	0	1	0	0	0	0
6	0	0	1	0	0	0
4	0	0	0	1	0	0
5	0	0	0	0	1	0
3	0	0	0	0	0	1

(ค)

รูปที่3.2การสลับลำดับเมตริกซ์พาริตี้เช็ค

(ก) เมตริกซ์พื้นฐาน

(ข) สลับลำดับของแถว

(ค) สลับลำดับทั้งแถวและหลัก

ตัวอย่างการสลับลำดับเมตริกซ์พาริตี้เช็คจะแสดงให้ดังรูปที่ 3.2ในรูป (ก) จะเป็นตัวอย่างของเมตริกซ์พื้นฐาน รูป (ข) แถวของเมตริกซ์ได้ถูกสลับลำดับที่แถว 3 กับ 6รูป(ค) หลังจากสลับลับดับแถวจึงนำมาสลับลำดับของหลักโดยที่นี้จะเปลี่ยนตำแหน่งของหลักทั้งหมด

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Old	7	2	15	16	20	21	17	18	9	8	5	12	1	14	3	11	13	10	4	6	19	22	23	24
1	5	-	-	-	-	-	-	0	0	0	-	3	11	23	-	-	9	-	-	-	-	-	-	-	-
2	2	0	0	0	-	-	-	-	-	12	0	17	-	22	0	-	-	-	-	-	-	-	-	-	-
3	3	-	-	0	0	-	-	-	-	24	-	10	-	6	-	0	0	-	-	-	-	-	-	-	-
4	8	8	24	-	-	0	0	-	-	6	-	0	-	13	-	-	-	-	-	-	-	-	-	-	-
5	4	-	-	-	0	-	-	0	-	25	-	20	-	2	-	-	-	-	0	0	-	-	-	-	-
6	6	3	-	-	-	-	-	-	0	10	-	17	-	24	-	23	-	-	-	1	-	0	-	-	-
7	7	-	-	-	-	0	-	-	-	7	-	8	-	25	-	-	-	0	18		-	0	-	-	-
8	9	-	20	-	-	-	0	-	-	23	-	22	-	7	-	-	-	-	-	16	10	-	0	-	-
9	1	-	-	-	-	-	-	-	-	0	-	0	0	0	0	-	-	1	-	-	0	-	-	-	-
10	10	-	-	-	-	-	-	-	-	13	-	19	17	11	-	-	3	-	-	-		-	0	0	-
11	11	-	-	-	-	-	-	-	-	9	14	23	-	25	-	8	-	-	-	-	18	-	-	0	0
12	12	-	-	-	-	-	-	-	-	25	2	16	-	3	-	-	-	1	5	-	-	-	-	-	0

รูปที่3.3เมตริกซ์พาริตี้เช็คH ที่ถูกสลับลำดับแล้ว

การสลับลำดับเมตริกซ์พาริตี้เช็ค H จะมีความเป็นได้ในการสลับออกมาได้หลายรูปแบบ แต่มีกฎสำคัญอยู่ว่าต้องย้ายทั้งแถวหรือทั้งหลักในเวลาเดียวกัน ในการนำเอาไปใช้งานบนฮาร์ดแวร์เราได้เพิ่มบัฟเฟอร์หลังจากรับคำรหัสมาจากช่องสัญญาณแล้ว จากนั้นจะทำการสลับตำแหน่งให้เข้ารูปแบบเมตริกซ์ที่ได้เราได้ออกแบบไว้ แล้วส่งไปยังหน่วยประมวลผลเพื่อทำการคำนวณตามอัลกอริทึม ทำนองเดียวกันเมื่อทำการถอดรหัสเสร็จแล้ว เราจำเป็นต้องสลับตำแหน่งของคำรหัสที่ได้ให้มีตำแหน่งเหมือนเดิม เพื่อให้ได้คำรหัสที่ถูกต้อง

จากการสลับลำดับเมตริกซ์ดังในรูปที่ 3.3จะเห็นว่าแถวที่ 9-12 ค่าผลลัพธ์ของการคำนวณจะไม่มีผลต่อหลักที่ 1-8 ซึ่งการทำงานที่ได้จากการสลับลำดับนี้จะส่งผลให้เกิดการทำงานที่ซ้อนกันหรือขนานกันได้ และการทำงานในหลักที่ 17-24 กับแถวที่ 1-4 ก็จะมีลักษณะเหมือนกัน โดยทั้งหมดนี้จะประมวลผลที่ 4 CNUs ต่อไซเคิลและ 8 VNUs ต่อไซเคิล และเมื่อมีความต้องการใช้จำนวนประมวลที่น้อยกว่านี้เราสามารถออกแบบให้ใช้หน่วยประมวลผลได้ที่ 3 CNUs และ 6 VNUs หรือสามารถใช้หน่วยประมวลที่น้อยที่สุดคือ 2 CNUs และ 4 VNUs โดยการสลับลำดับเมตริกซ์ทั้งหมดนี้ต้องเป็นไปกฎตามที่ระบุไว้ข้างต้น ดังแสดงให้เห็นดังรูปที่ 3.4 ซึ่งเป็นเมตริกซ์ที่เราออกแบบไว้โดยใช้หน่วยประมวลน้อยที่สุด

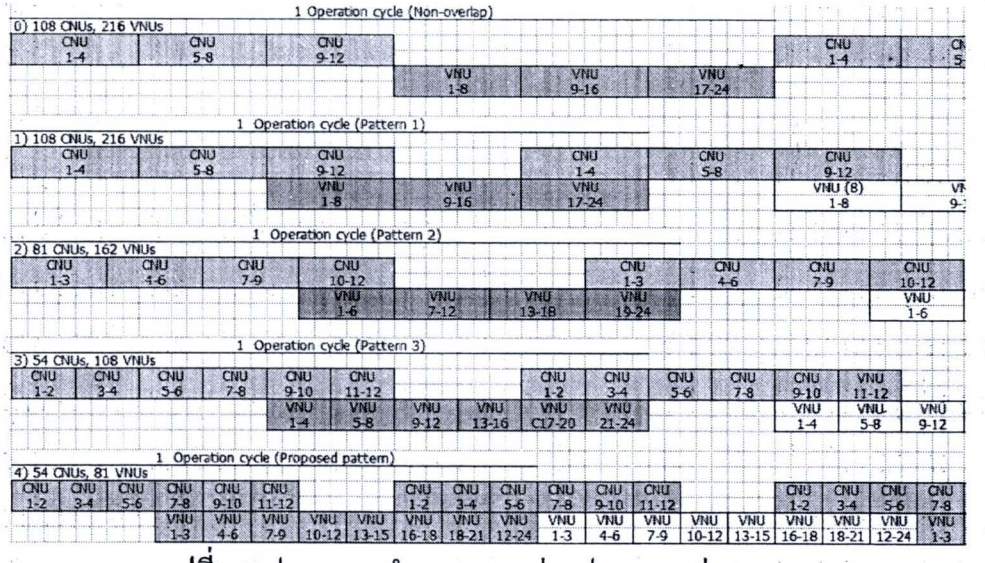
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Old	15	17	7	16	20	21	2	18	14	12	5	8	1	9	11	3	19	10	4	6	13	22	23	24
1	8	-	-	8	-	0	0	24	-	-	-	0	-	13	6	-	-	-	-		-	-	-	-	-
2	2	0	-	0	-	-	-	0	-	0	-	17	0	22	12	-	-	-	-	-	-	-	-	-	-
3	3	0	-	-	0	-	-	-	-	-	-	10	-	6	24	0	0	-	-	-	-	-	-	-	-
4	5	-	0	-	-	-	-	-	0	-	11	3	-	23	0	9	-	-	-		-	-	-	-	-
5	4	-	0	-	0	-	-	-	-	-	-	20	-	2	25	-	-	-	0	0	-	-	-	-	-
6	6	-	-	3	-	-	-	-	0	-	-	17	-	24	10	-	23	0	-	1	-	-	-	-	-
7	7	-	-	-	-	0	-	-	-	-	-	8	-	25	7	-	-	0	18		-	0	-	-	-
8	9	-	-	-	-	-	0	20	-	-	-	22	-	7	23	-	-	-	-	16	10	-	0	-	-
9	1	-	-	-	-	-	-	-	-	0	0	0	-	0	0	-	-	-	-	-	0	1	-	-	-
10	10	-	-	-	-	-	-	-	-	-	17	19	-	11	13	3	-	-	-	-		-	0	0	-
11	11	-	-	-	-	-	-	-	-	-	-	23	14	25	9	-	8	-	-	-	18	-	-	0	0
12	12	-	-	-	-	-	-	-	-	-	-	16	2	3	25	-	-	-	5	-	-	1	-	-	0

รูปที่3.4เมตริกซ์พาริตี้เช็คH ที่ถูกสลับลำดับแล้วในรูปแบบที่เราสำเสนอ

3.2 การทำงานซ้อนกันของ VNU และ CNU(Overlapped Opreations of VNUs and CNUs)

จากส่วนที่ผ่านมา เราสามารถเปลี่ยนรูปแบบของเมตริกซ์พาริตี้เช็คโดยการสลับลำดับเมตริกซ์ ซึ่งสามารถจัดรูปให้เห็นถึงการทำงานที่ไม่มีการซ้อนกันและการทำงานซ้อนกันของแต่ละรูปแบบดังรูปที่ 3.5 โดยรูปแบบที่ 1-3 จะเป็นรูปร่างตามการออกแบบของบทความ [4] และการออกแบบของเราจะอยู่ในรูปแบบการทำงานที่ 4

ในรูปแบบการประมวลผลที่ 3 จะเห็นว่าหน่วยประมวลผล 8 VNUs จะมีช่วงเวลาที่ไม่สามารถทำงานซ้อนกันได้ หรืออีกนัยหนึ่งคือ จะมีช่วงเวลาว่างที่ไม่ทำงาน 8/24 หรือ 33% หน่วยประมวลผลของ CNU ในทำนองเดียวกัน CNU ก็จะมีลักษณะเหมือนกันคือจะมีช่วงว่างจำนวน 8 ช่วง ซึ่งอยู่ในช่วงเวลาการทำงานของ VNU ในการเปรียบเทียบเพื่อให้เห็นถึงความแตกต่าง สำหรับรูปแบบการประมวลแบบที่ 4 หน่วยประมวลผล VNU ทั้งหมดจะทำงานติดกันโดยไม่มีช่วงว่าง ขณะที่ CNU มีช่วงว่างอยู่ที่ 6 ช่วง ในช่วงเวลาการทำงานของ VNU ซึ่งจำนวนไซเคิลที่ซ้ำกันจะเป็นจำนวนของการซ้อนกัน โดยสามารถเลือกใช้ได้ตามความเหมาะสมดังที่เราออกแบบไว้ ดังเช่นกรณีของรูปแบบในกลุ่มการประมวลผลของรูปแบบที่ 3 โดยนำรูปแบบที่ 4 ซึ่งเราออกแบบไว้มาเทียบเคียงจะเห็นได้ว่าผลลัพธ์โดยรวมดีกว่า ดังนั้นผลลัพธ์ที่ได้ในการออกแบบการสลับลำดับเมตริกซ์นั้นจะมีหลายรูปแบบ ดังเช่นเมตริกซ์ของเราในรูปที่ 3.4



รูปที่3.5รูปแบบการทำงานของหน่วยประมวลแต่หละแบบ

จากการพิจารณาในรูปที่ 3.5 จะชี้ให้เห็นถึงการทำงานวนซ้ำ (iteration) ในหนึ่งรอบของการประมวลผลของยูนิตแถว (CNU) และยูนิตหลัก (VNU) โดยในการทำงานหนึ่งไซเคิลจะเป็นการคำนวณหนึ่งแถวหรือหนึ่งหลัก (สามารถทำในรูปแบบขนานได้) การวิเคราะห์ทางคณิตศาสตร์; ในรูปแบบทำงานที่ไม่ซ้อนกัน (pattern 0) จะถูกทำการประมวลผลทีหละ 4 แถวในหนึ่งไซเคิล (ต้องการ 108 CNUs) และจะทำการประมวลผลทีหละ 8 หลักในหนึ่งไซเคิล (ต้องการ 216 VNUs) โดยทั้งหมดจะทำงานที่ 20 รอบดังนั้นจะใช้ไซเคิลเท่ากับ 120 ไซเคิล ในรูปแบบที่ 1 (pattern 1) จะทำการประมวลผล 4 แถว 8 หลัก ต่อหนึ่งไซเคิล ใช้ 108 CNUs และ 216 VNUs เมื่อทำงานซ้อนกันไซเคิลจะใช้เท่ากับ [(4*Ite)+1] และทำงานไม่ซ้อนกัน [6*Ite] Iteจะบ่งบอกถึงจำนวนการวนซ้ำ ในทำนองเดียวกันของรูปแบบที่ 2, 3 และ 4 จะใช้วิธีคำนวณคล้ายกันข้างต้น ซึ่งผลลัพธ์จะแสดงอยู่ในตารางที่ 3.1 โดยเราจะเน้นการเปรียบเทียบกันระหว่างรูปแบบที่ 3 กับรูปแบบที่ 4 ที่เราได้ออกแบบไว้ ซึ่งเราจะใช้ ยูนิตของ VNU เพียง 75% จากรูปแบบที่ 3 และยังคงสามารถทำงานซ้อนกันได้เหมือนเดิม จึงทำให้เห็นว่าการใช้หน่วยประมวลผลของ CNU ยังคงเดิมแต่หน่วยประมวลผลของ VNU น้อยลง

Pattern	CNU (Units)	VNU (Units)	Overlapped (Cycles)	Cycles Utilization	
				CNU	VNU
Typ. Cons	324	648	-	(324/972)	(648/972)
0	108	216	-	(3/6)	(3/6)
1, [5]	108	216	81; [(4*Ite)+1]	(4/5)	(4/5)
2, [5]	81	162	121; [(6*Ite)+1]	(5/7)	(5/7)
3, [5]	54	108	162; [(8*Ite)+2]	(8/10)	(8/10)
4, [proposed]	54	81	163; [(8*Ite)+3]	(9/11)	(11/11)

ตารางที่ 3.1เปรียบเทียบการใช้เวลา (cycle)ของการทำงานแต่หละรูปแบบ