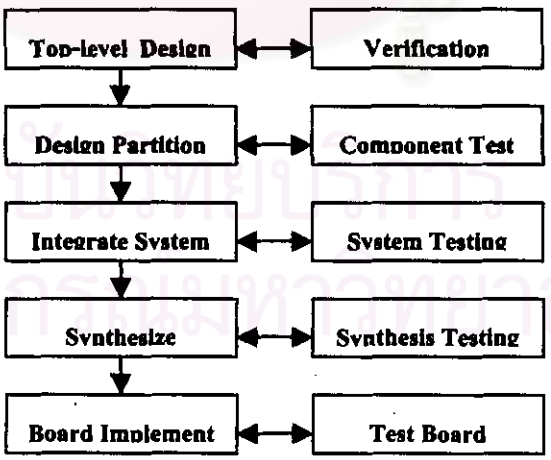
บทที่ 3

การออกแบบและการทำงานของอาร์ม 7 ไมโครโพรเซสเซอร์

อาร์ม7 เป็นไมโครโพรเซสเซอร์ในตระกูล 32 บิตของบริษัท Advanced RISC Machines (ARM) ที่ได้รับความนิยมเนื่องจากมีประสิทธิภาพในการทำงานสูง ประหยัดพลังงาน และราคาถูก โครงสร้างสถาปัตยกรรมจะเป็นแบบ Reduced Instruction Set Computer (RISC)[David A. Patterson and John L. Hennessy, 1997] พร้อมชุดคำสั่งที่มี ประสิทธิภาพสูง การทำงานของไปป์ไลน์ทำให้มีความต่อเนื่องของการทำงานและสถานะการทำงาน 3 สถานะ คือการนำเอาคำสั่งเข้ามาทำงาน (Fetch) การถอดรหัส (Decode) และเริ่มทำงานจริง (Execute)

จากการศึกษาคู่มือ โครงสร้างของอาร์ม7และกระบวนการออกแบบไมโครโพรเซสเซอร์ s สามารถสรุปและเสนอขั้นตอนของการออกแบบให้เหมาะสมกับอาร์ม 7 ได้ดังรูปที่ 3.1 ซึ่งแสดงขั้นต่างๆ ที่ในการออกแบบโดยจะมีการทำงานไปพร้อมกันของส่วนการออกแบบและการทดสอบ เมื่อพบข้อผิดพลาดจะต้องทำการแก้ไขจนได้ความถูกต้อง เพื่อให้ได้การทำงานของวงจรถูกต้องมากขึ้น เนื่องจากไมโครโพรเซสเซอร์อาร์มมีโครงสร้างการทำงานที่ซับซ้อน ดังนั้นการตรวจในทุกระดับของการ ออกแบบจะทำให้สามารถตรวจพบข้อผิดพลาดได้ง่ายกว่าการตรวจสอบระบบรวมทั้งหมดแต่เพียงอย่างเดียว



รูปที่ 3.1 ขั้นตอนการออกแบบ

3.1 ขั้นตอนของการออกแบบไมโครโพรเซสเซอร์ 32 บิต

ขั้นตอนการออกแบบระบบนั้นจะเหมือนกับการออกแบบโดยทั่วไป แต่ส่วนสำคัญที่ต่างกันคือ *"ไม่ได้สร้างระบบขึ้นมาใหม่ แต่สร้างระบบที่มีการทำงานเหมือนกับระบบที่เป็นต้นแบบให้มากที่สุด"* จึงต้องการศึกษาการทำงานของคำสั่ง (Instruction Set) และไทม์มิ่งของคำสั่ง (Instruction Cycle) ให้เข้าใจการทำงานแล้วจึงนำมาออกแบบโครงสร้างของไมโครโพรเซสเซอร์

3.1.1 การออกแบบส่วนบน (Top-Level Design)

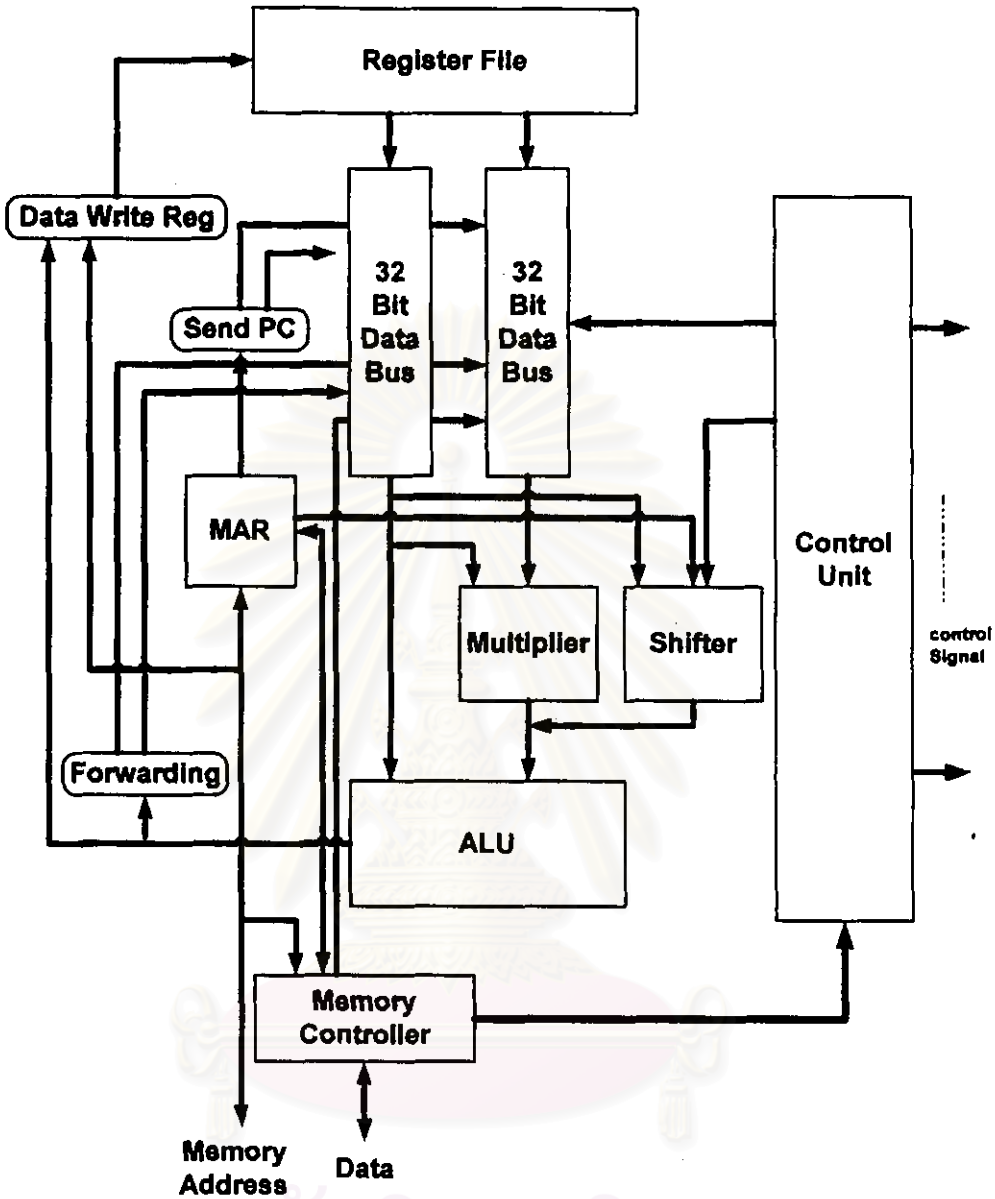
ขั้นนี้เป็นการออกแบบโครงสร้างสถาปัตยกรรมของไมโครโพรเซสเซอร์ 32 บิตที่แสดงถึงการเชื่อมต่อส่วนต่างๆในระบบทั้งหมดดังรูปที่ 3.2 รวมถึงการทำแผนการทดสอบในแต่ละส่วนของการออกแบบ ในที่นี้ใช้แผนภาพบล็อก (Block Diagram) แทนส่วนต่างๆของโครงสร้างของไมโครโพรเซสเซอร์ที่จะออกแบบ

3.1.2 การทวนสอบ (Verification)

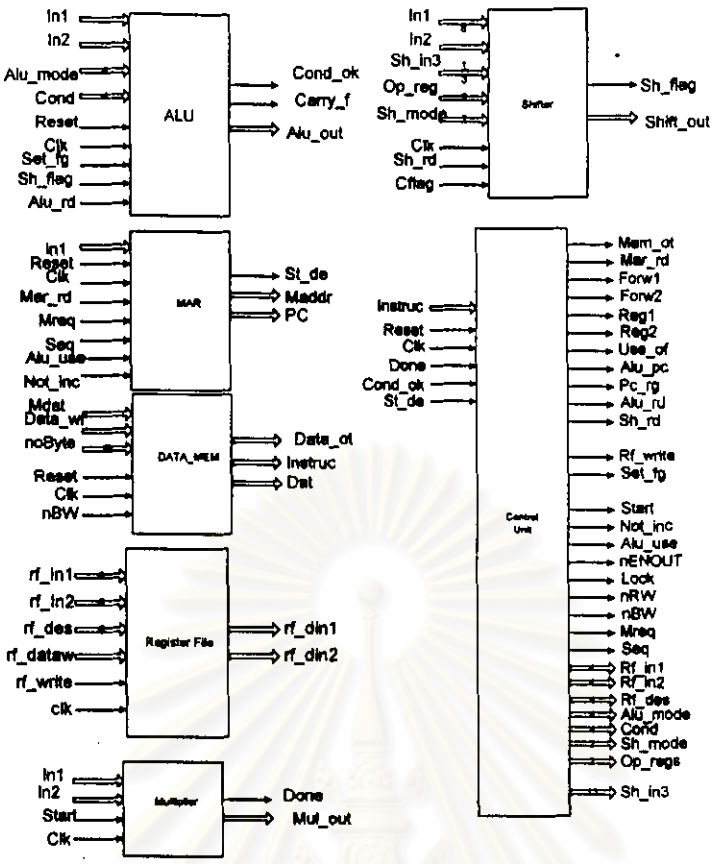
เป็นการวางแผนการตรวจสอบเพื่อให้ระบบที่ออกแบบให้มีการทำงานตรงกับที่คุณลักษณะที่กำหนด (Specification) ตลอดทั้งงานวิจัยมีการเทียบผลจากการออกแบบกับ Development Tool Kit ARM 2.11a เพื่อให้มั่นใจว่าที่ออกแบบมีการทำงานได้ถูกต้องเหมือน อาร์ม7 โดยใช้วิธีการจำลองการทำงาน (Simulate-based) เพื่อความสะดวกและง่ายต่อการตรวจสอบความถูกต้องของระบบใหญ่

3.1.3 แบ่งส่วนการออกแบบ (Design Partitioning)

แบ่งการออกแบบเป็นส่วนเล็กๆเพื่อให้ง่ายต่อการสร้างด้วยภาษา VHDL และง่ายในการหาข้อผิดพลาดจากระบบที่ออกแบบมากกว่าการหาข้อผิดพลาดทั้งระบบที่ใหญ่กว่า จากรูปที่ 3.3 เป็นส่วนต่างๆที่ได้แบ่งการออกแบบไว้ และกำหนดสัญญาณและขาที่ใช้ในการติดต่อระหว่างชิ้นส่วนย่อย (component)



รูปที่ 3.2 คาต้าพาทของไมโครโพรเซสเซอร์ 32 บิต



รูปที่ 3.3 ชิ้นส่วนต่างๆในโครงสร้างของไมโครโพรเซสเซอร์

3.1.4 การทดสอบชิ้นส่วน (Component Testing)

ใช้ทดสอบการทำงานของแต่ละส่วนที่ได้จากการทดสอบชิ้นส่วนโดยการเขียนชุดทดสอบด้วยภาษา VHDL และทำการจำลองการทำงานด้วยโปรแกรม ModelSim ซึ่งชุดทดสอบจะทำการส่งอินพุทให้กับชิ้นส่วนที่ออกแบบ แล้วจึงจับสัญญาณผลลัพธ์ที่ได้พร้อมทั้งนำกลับเข้ามาทำการเปรียบเทียบกับผลลัพธ์ที่ควรจะเป็น ผลลัพธ์ที่ได้จะให้ความมั่นใจและความน่าเชื่อถือของการทำงานในแต่ละส่วนเพื่อลดความซับซ้อนในการตรวจสอบทั้งระบบในภายหลัง เมื่อในแต่ละชิ้นส่วนมีความถูกต้องแล้วจะทำให้สามารถตรวจสอบทั้งระบบได้ด้วยหลักนามธรรม (Abstract) เนื่องจากการตรวจสอบในระดับชิ้นส่วนจะทำให้ได้ชิ้นส่วนที่มีการทำงานถูกต้อง จึงไม่จำเป็นต้องมีการตรวจสอบซ้ำ ดังนั้นสามารถทำงานตรวจสอบเฉพาะการเชื่อมต่อและการทำงานร่วมกันระหว่างชิ้นส่วน

3.1.5 การรวมระบบ (Integrated System)

ทำการเชื่อมส่วนต่างๆเข้าด้วยกันเพื่อทำงานร่วมกัน โดยจะทำการเชื่อมสัญญาณในแต่ละชิ้นส่วนเข้าด้วยกันด้วยลักษณะของการอธิบายการทำงานแบบโครงสร้างของภาษา VHDL

3.1.6 ทดสอบระบบ (System Testing)

ทดสอบระบบที่ทั้งหมดหลังจากการทำการเชื่อมต่อส่วนต่างๆ ในที่นี้ได้ทำการสร้างหน่วยความจำเพื่อให้เก็บชุดคำสั่งในการทดสอบไมโครโพรเซสเซอร์ ที่จะทำการป้อนให้กับไมโครโพรเซสเซอร์และใช้วิธีการจำลองการทำงานเพื่อดูและเก็บผลลัพธ์ที่ได้จากการทำงานของวงจรที่ออกแบบ จากนั้นจึงนำไปเปรียบเทียบกับผลลัพธ์ที่ได้จากการใส่ชุดคำสั่งเดิมที่ให้โปรแกรมตรวจสอบจุดบกพร่อง (Debugger) ของอาร์ม 7 การตรวจสอบในระดับนี้เป็นระดับการทดสอบคำสั่ง

3.1.7 การสังเคราะห์วงจร (Synthesization)

สังเคราะห์วงจรที่ออกแบบในระดับสูงเป็นวงจรในระดับเกตด้วยโปรแกรม ในที่นี้ใช้ Leonardo Spectrum และทำกระบวนการย้อนกลับเพื่อนำโค้ด VHDL ที่ได้จากวงจรในระดับเกต

3.1.8 การทดสอบวงจรจากการสังเคราะห์ (Synthesis Testing)

นำวงจรที่เป็น VHDL ซึ่งได้จากการทำกระบวนการย้อนกลับด้วยโปรแกรมที่ใช้ในการสังเคราะห์วงจร โดยนำวงจรที่ได้มาทำการตรวจสอบด้วยวิธีการจำลองการทำงาน ซึ่งจะใช้คำสั่งทดสอบชุดเดิมกับในขั้นตอนของระบบทดสอบ (System Testing) ในขั้นตอนนี้วงจรที่ออกแบบจะมีการทดสอบส่วนของค่าคงที่ของเวลาในเกตที่มีการใช้งานจริงบนเทคโนโลยีของเอฟพีจีเอ ด้วยการจำลองการทำงานวงจรซึ่งได้จากขั้นตอนการวางอุปกรณ์ (Place) และเชื่อมต่อสาย (Route) ด้วยโปรแกรม Foundation Xilinx 1.5 ของบริษัท Xilinx

3.2 การทำงานของไมโครโพรเซสเซอร์ อาร์ม 7

อาร์ม มีโครงสร้างสถาปัตยกรรมแบบริกส์ซึ่งมีรูปแบบโดยทั่วไปดังนี้

- ทุกคำสั่งในหมวดคำสั่ง Data-Processing จะผ่านการทำงานทั้งในส่วนของหน่วยคำนวณและตรรกะและตัวเลื่อนค่าเสมอ
- มีการเพิ่มและลดค่าของการอ้างอิงหน่วยความจำเพื่อลดวงวน (loop) ที่เกิดขึ้นในโปรแกรม
- สามารถเก็บและบรรจุ (Load/Store) ข้อมูลได้ทีละจำนวนมากได้ภายในคำสั่งเดียว
- มีเงื่อนไขการประมวลผลของทุกๆคำสั่ง
- มีไปป์ไลน์ในการทำงาน 3 สถานะ คือการนำคำสั่งเข้า การถอดรหัส และการทำงาน

3.2.1 รีจิสเตอร์ใน อาร์ม

อาร์ม มีรีจิสเตอร์ขนาด 32 บิตจำนวน 31 ตัวแต่สามารถใช้และผู้ใช้สามารถมองเห็นและใช้งานได้ 16 รีจิสเตอร์ ซึ่งทำหน้าที่เป็นรีจิสเตอร์ทั่วไปในส่วนของโหมดการทำงานปกติ ในขณะเดียวกันรีจิสเตอร์ตัวที่ 15 สามารถใช้งานเป็น พีซี โดยค่า พีซี จะเพิ่มขึ้นครั้งละ +4 เนื่องจากชุดคำสั่งมีขนาด 32 บิต ดังนั้น พีซี จะมีค่า 30 บิตที่แปลงเปลี่ยน ส่วน 2 บิตล่างจะเป็น 0 เสมอ ส่วนรีจิสเตอร์ที่ 14 จะถูกใช้เก็บเป็นรีจิสเตอร์เชื่อมโยง (รีจิสเตอร์เชื่อมโยง) เมื่อมีการทำงานคำสั่งกระโดดข้ามแบบมีเงื่อนไข (BL) และรีจิสเตอร์ที่ 13 จะถูกใช้โดยซอฟแวร์ให้เป็น Stack Pointer (SP)

3.2.2 คำสั่งในไมโครโพรเซสเซอร์ อาร์ม

ชุดคำสั่งสามารถแบ่งออกได้เป็น

- Branch
- Data-Processing
- Load and Store
- Coprocessor *

<u>หมายเหตุ</u> ในส่วน coprocessor ไม่มีการออกแบบ

ในทุกๆคำสั่งใน อาร์ม จะมีการตรวจเงื่อนไขก่อนมีการทำงาน โดยเงื่อนไขในการทำงานจะเป็น 4 บิตบนของทุกคำสั่ง(บิตที่ 31-28)

<u>Branch</u> : คำสั่งกระโดดข้ามจะมีผลต่อการไหลของควบคุมโดยการเปลี่ยนแปลงค่าใน พีซี โดยคำสั่งกระโดดข้ามมีการใช้ 24 บิตเพื่อเป็นค่าออฟเซต ทำให้สามารถกระโดดข้ามทั้งไปหน้าและกลับหลังถึง 32 Mbytes

<u>Data-Processing</u> : มีด้วยกัน 3 ชนิดคือ

- Data-processing ทั่วไป
- คำสั่งการคูณ
- คำสั่งบอกสถานะของรีจิสเตอร์

คำสั่งของหน่วยคำนวณและตรรกะ จะใช้ตัวถูกดำเนินการ (Operand) 2 ตัวและเก็บผลลัพธ์ใส่รีจิสเตอร์โดยมี บางคำสั่งของ data-processing ที่ไม่มีการเก็บค่าลงรีจิสเตอร์ เพราะเป็นการเปรียบเทียบค่าเท่านั้น จึงเปลี่ยนแปลงเฉพาะค่าบ่งชี้ ตัวถูกดำเนินการจะได้มาจาก 2 แหล่งคือ มาจากรีจิสเตอร์ทั้งคู่ และมากจากตัวอื่นได้แก่ ค่าจำนวนเต็ม, ค่าจากรีจิสเตอร์ที่สามารถมีการเลื่อนค่าได้ ถ้าหากเป็นการเลื่อนค่าในรีจิสเตอร์ จำนวนครั้งในการเลื่อนค่าจะมาจากคำจำนวนเต็ม หรือค่าจากรีจิสเตอร์ ดังนั้นทุกคำสั่งใน Data processing สามารถทำการเลื่อนค่าได้ อาร์ม จึงไม่มีคำสั่งการเลื่อนค่า

คำสั่งการคูณสามารถแบ่งออกได้เป็น 2 แบบโดยทั่วไปใช้รีจิสเตอร์ 32 บิต 2 ค่าในการเก็บ

(ปกติ) 32 บิต ผลลัพธ์เก็บในรีจิสเตอร์เดียว

(ยาว) 64 บิตผลลัพธ์เก็บในรีจิสเตอร์ 2 ตัวแยกกัน

* ซึ่งทั้ง 2 แบบสามารถทำการคำนวณการบวกพร้อมกันได้ด้วย

Load/Store : คำสั่งการเก็บและบรรจุ แบ่งออกได้เป็น 3 ชนิด

- เก็บและบรรจุ ด้วยการใช้รีจิสเตอร์ 1 ตัว
- เก็บและบรรจุ ด้วยการใช้รีจิสเตอร์หลายตัว
- แลกเปลี่ยนข้อมูลระหว่างรีจิสเตอร์และหน่วยความจำ

การทำงานของคำสั่งเก็บและบรรจุจะมีการอ้างอิงหน่วยความจำได้ 3 แบบคือ offset, pre-index และ post-index นอกจากนี้แล้วยังมีคำสั่งในการแลกเปลี่ยนข้อมูลระหว่างรีจิสเตอร์และหน่วยความจำ ซึ่งคำสั่งทั้งหมดสามารถส่งและรับข้อมูลได้ทีละ 8 บิต 16 บิต และ 32 บิต

ตารางที่ 3.1 สรุปคำสั่งที่สามารถทำงานได้ในไมโครโพรเซสเซอร์ที่ออกแบบ

คำสั่ง	Source 1	Source 2	Shift	Execute	Write-back
B	-	-	-	-	#Jump>Pc
BL	-	-	-	-	#Jump>Pc
AND	W	W	OP2	SHIFT,AND	W
EOR	W	W	OP2	SHIFT,XOR	W
SUB	W	W	OP2	SHIFT,SUB	W
RSB	W	W	OP2	SHIFT,RSB	W
ADD	W	W	OP2	SHIFT,ADD	W
ADC	W	W	OP2	SHIFT,ADC	W
SBC	W	W	OP2	SHIFT,SBC	W
RSC	W	W	OP2	SHIFT,RSC	W
TST	-	-	OP2	SHIFT,TST	-
TEQ	-	-	OP2	SHIFT,TEQ	-
CMP	-	-	OP2	SHIFT,CMP	-
CMN	-	-	OP2	SHIFT,CMN	-
ORR	W	W	OP2	SHIFT,ORR	W
MOV	-	W	OP2	SHIFT,MOV	W
BIC	W	W	OP2	SHIFT,OP2 AND NOT	W
MVN	W	W	OP2	OP1	W
MUL	W	W	-	SHIFT,NOT OP2	W
MLA	W	W	-	MUL	W
LDR	W,B	W,B	OP2	MUL,ADD	W,B
STR	W,B	W,B	OP2	SHIFT,LOAD	W,B
SWP	W,B	W,B	OP2	SHIFT,STORE	W,B
				LOAD,STORE	

3.3 การออกแบบและการสังเคราะห์วงจรในแต่ละส่วนของอาร์ม 7

3.3.1 หน่วยคำนวณและตรรกะ

ประกอบด้วย 3 ส่วนด้วยกันคือ

1. *ส่วนที่ทำการประมวลผลตามฟังก์ชันการทำงานที่ได้รับมาจากหน่วยควบคุม (Control Unit)* มีฟังก์ชัน (Function) การทำงาน 16 ฟังก์ชัน โดยแบ่งเป็น ตัวถูกดำเนินการลอจิก 8 ฟังก์ชัน และ ตัวถูกดำเนินการทางคณิตศาสตร์ 8 ฟังก์ชัน ดังนี้

0000 = AND	OUT_ALU = (OP1 AND OP2)
0001 = XOR (EOR)	OUT_ALU = (OP1 XOR OP2)
0010 = SUB	OUT_ALU = (OP1 - OP2)
0011 = RSB	OUT_ALU = (OP2 - OP1)
0100 = ADD	OUT_ALU = (OP1 + OP2)
0101 = ADC	OUT_ALU = (OP1 + OP2 + CARRY)
0110 = SBC	OUT_ALU = (OP1 - OP2 + CARRY - 1)
0111 = RSC	OUT_ALU = (OP2 - OP1 + CARRY - 1)
1000 = TST	SAME AND BUT NOT WRITE RESULT
1001 = TEQ	SAME EOR BUT NOT WRITE RESULT
1010 = CMP	SAME SUB BUT NOT WRITE RESULT
1011 = CMN	SAME ADD BUT NOT WRITE RESULT
1100 = ORR	OUT_ALU = (OP1 OR OP2)
111010 = BIC	OUT_ALU = OP1 AND (NOT OP2)

2. *ส่วนที่ทำการกำหนดค่า บ่งชี้ ต่างๆ* จะทำการกำหนดค่าบ่งชี้ ได้แก่ ตัวบ่งชี้การทด (Carry Flag, C) ตัวบ่งชี้การล้น (over Flag, V) ตัวบ่งชี้ค่าศูนย์ (Zero Flag, Z) และ ตัวบ่งชี้ค่าลบ (Negative Flag, N) ก็ต่อเมื่อค่า Set_fg มีค่าเป็น 1 คือ เป็นการประมวลผลที่กำหนดให้มีการเปลี่ยนแปลงค่า บ่งชี้ด้วย และจะทำการเปลี่ยนแปลงทุกๆ ขอบขาลงของสัญญาณนาฬิกา

3. *ส่วนที่ทำการกำหนดค่าเงื่อนไขการบ่งชี้ เพื่อส่งไปให้ตัวควบคุม* จะนำค่าตัวบ่งชี้ที่เปลี่ยนแปลงแล้วมาทำการตรวจสอบว่าเป็นไปตามเงื่อนไข (Condition) ของเงื่อนไขนั้นๆ หรือไม่ ซึ่งจะกำหนดเป็น 1 เมื่อเป็นไปตามเงื่อนไข โดยมีเงื่อนไข ดังนี้

0000 = EQ - Z set (equal)

0001 = NE - Z clear (not equal)

0010 = CS – C set (unsigned higher or same)

0011 = CC – C clear (unsigned lower)

0100 = MI – N set (negative)

0101 = PL – N clear (positive or zero)

0110 = VS – O set (overflow)

0111 = VC – O clear (no overflow)

1000 = HI – C set and Z clear (unsigned higher)

1001 = LS – C clear and Z set (unsigned lower or same)

1010 = GE – N set and O set , or N clear and O clear (greater or equal)

1011 = LT – N set and V clear, or N clear and V set (less than)

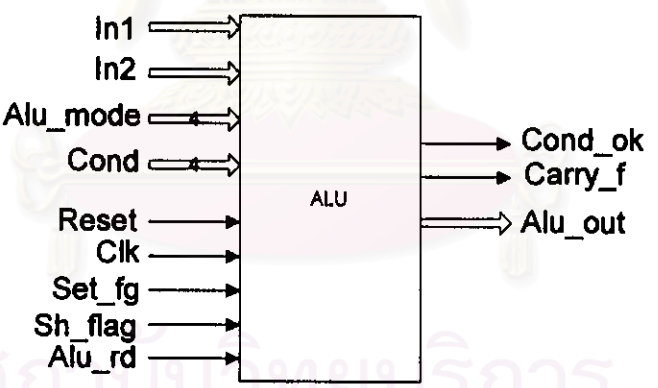
1100 = GT – Z clear, and either N set and V set, or N clear and V clear (greater than)

1101 = LE – Z set, or N set and V clear, or N clear and V set (less than equal)

1110 = AL – always

1111 = NV – never

โดยค่าตัวถูกดำเนินการที่ 2 ของหน่วยคำนวณและตรรกะ นั้นเป็นค่าที่มาจากตัวเลื่อนค่าหรือตัวคูณ



รูปที่ 3.4 ส่วนของหน่วยคำนวณและตรรกะ

จากการศึกษารูปแบบการทำงานทั้งหมดของหน่วยคำนวณและตรรกะสามารถทำการออกแบบได้ดังรูปที่ 3.4 โดยการแบ่งเป็นส่วนของอินพุทที่รับข้อมูล 2 พอร์ทคือ In1 และ In2 โดยผลลัพธ์ที่ได้จากการทำงานในส่วนนี้จะถูกส่งออกทางพอร์ด Alu_out ทั้งนี้จะทำการขึ้นกับสัญญาณนาฬิกา โดยจะมีรายละเอียดของหน้าที่ในแต่ละขาสัญญาณดังนี้

Alu_mode	บอกฟังก์ชั่นการทำงานของหน่วยคำนวณและตรรกะ
Cond	บอกเงื่อนไขของการทำงาน
Set_fg	ควบคุมคำนวณค่าลงตัวบ่งชี้

Alu_rd	ทำหน้าที่บอกเมื่อมีข้อมูลรอที่พอร์ทอินพุทเรียบร้อย
Cond_ok	ส่งกลับไปยังหน่วยควบคุมเมื่อเงื่อนไขการทำงานถูกต้อง

จากการศึกษาจึงได้ทำการออกแบบส่วนของหน่วยคำนวณและตรรกะให้มีการทำงาน โดยมี 4 ส่วนย่อยที่ทำงานพร้อมกันดังนี้

- ส่วนของการรับค่า จะต้องทำหน้าที่ในการรับข้อมูลที่พอร์ทอินพุท ซึ่งจะทำการตรวจสอบค่าจาก Alu_rd และทำงานสอดคล้องกับสัญญาณนาฬิกาขาขึ้น (0-1)
- ส่วนของการตรวจสอบเงื่อนไขการทำงาน โดยจะนำค่าที่ได้จาก Cond ซึ่งมีขนาด 4 บิตมาทำงานตรวจสอบ หากถูกต้องจะทำการกำหนดค่าสัญญาณให้กับ Cond_ok เพื่อส่งให้ส่วนควบคุมทราบว่าเงื่อนไขในการเข้าทำงานถูกต้อง
- ส่วนของการประมวลผลทำหน้าในการตรวจฟังก์ชั่นการทำงานจาก Alu_mode แล้วประมวลผลเพื่อให้ได้ผลลัพธ์ส่งให้กับพอร์ทเอาท์พุท
- ส่วนของการตรวจค่าตัวบ่งชี้ ทำการตรวจสัญญาณ Set_fg เมื่อไมโครโพรเซสเซอร์ต้องการให้เปลี่ยนแปลงค่าในตัวบ่งชี้ ซึ่งถ้าต้องการให้ทำการเปลี่ยนแปลงค่า จะนำค่าตัวบ่งชี้เข้ามาประมวลผลด้วย และเมื่อมีผลต่อตัวบ่งชี้ carry ก็จะสร้างสัญญาณให้กับหน่วยควบคุมทางขา carry_f

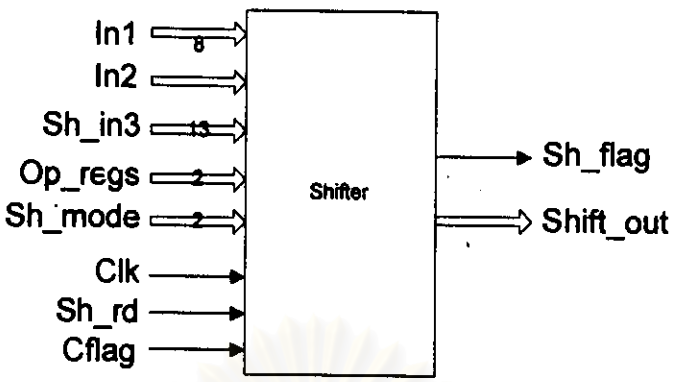
ผลการสังเคราะห์วงจรส่วนคำนวณและตรรกะ

พบว่าส่วนคำนวณและตรรกะที่ได้ทำการออกแบบใช้พื้นที่ในส่วนของเอฟพีจีเอ (XC4085XLA) ดังนี้

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	116 ขา
ตัวสร้างฟังก์ชั่น FG	257 ตัว
ตัวสร้างฟังก์ชั่น H	105 ตัว
ฟลิปฟล็อป	5 ตัว
สามารถทำงานได้ที่ความถี่	69.8 MHz

หมายเหตุ ฟังก์ชั่น เอฟจี และ เอช ต่างเป็นส่วนของวงจรเชิงผสม (Combination circuit) ต่างกันที่เอฟจีมีได้ 4 อินพุท ส่วน เอชมี 3 อินพุท

3.3.2 ตัวเลื่อนค่า (SHIFTER)



รูปที่ 3.5 ส่วนของตัวเลื่อนค่า

มี 4 ฟังก์ชันการทำงาน ได้แก่

1. Logical Shift Left (LSL) ทำการเลื่อนค่าไปทางซ้าย โดยถ้า N เป็นค่าจำนวนครั้งแล้ว

   N < 32 ได้ผลลัพธ์ตามค่าที่เลื่อน และตัวบ่งชี้ มีค่าตามบิตที่ 32-N

   N = 32 ผลลัพธ์เป็น 0 และตัวบ่งชี้มีค่าตามบิตที่ 0

   N > 32 ผลลัพธ์เป็น 0 และ ตัวบ่งชี้มีค่าเป็น 0

2. Logical Shift Right (LSR) ทำการเลื่อนค่าไปทางขวา โดยถ้า N มีค่าเป็น

   N < 32 ได้ผลลัพธ์ตามค่าที่เลื่อนค่า และตัวบ่งชี้มีค่าตามบิตที่ N-1

   N = 32 ผลลัพธ์เป็น 0 และ ตัวบ่งชี้ มีค่าตามบิตที่ 31

   N > 32 ผลลัพธ์เป็น 0 และ ตัวบ่งชี้ มีค่าเป็น 0

3. Arithmetic Shift Right (ASR) ทำการเลื่อนค่าไปทางขวา แล้วเติมบิตทางซ้ายด้วยค่าของบิตที่ 31 (Sign Bit) ของค่าตัวถูกดำเนินการที่ถูกเลื่อนค่าโดยถ้า N มีค่าเป็น

   N < 32 ได้ผลลัพธ์ตามค่าที่เลื่อน และตัวบ่งชี้ มีค่าตามบิตที่ N-1

   N >= 32 ผลลัพธ์เป็นค่า บิตเครื่องหมายจำนวน 32 บิต) และ ตัวบ่งชี้มีค่าตามบิตที่ 31

4. Rotate Right (ROR) ทำการหมุนค่าไปทางขวา โดยถ้า N มีค่าเป็น

   N < 32 ได้ผลลัพธ์ตามค่าที่ หมุนค่า และตัวบ่งชี้ มีค่าตามบิตที่ N-1

   N = 32 ผลลัพธ์เป็นค่าเดิม และ ตัวบ่งชี้ มีค่าตามบิตที่ 31

   N > 32 ผลลัพธ์ และ ตัวบ่งชี้ มีค่าเหมือนการ หมุนค่า ด้วยค่าจำนวนครั้งที่อยู่ระหว่าง 1-32 (ค่าเศษที่ได้จากการหารด้วย 32 นั่นเอง)

เมื่อทำการศึกษาและออกแบบส่วนเลื่อนค่าได้ตามรูปที่ 3.5 โดยสามารถแบ่งออกเป็นส่วนของพอร์ทรับข้อมูล พอร์ทผลลัพธ์ และสัญญาณต่างๆ โดยมีหน้าที่ต่างๆดังนี้

In1	มีขนาด 8 บิตทำหน้าที่เป็นอินพุทที่เป็นจำนวนครั้งในการเลื่อนค่าที่ได้จากค่าในรีจิสเตอร์ส่วนไบต์ล่าง (บิตที่ 0-7)
In2	มีขนาด 32 บิตทำหน้าที่เป็นอินพุทรับข้อมูลที่ต้องการเลื่อนค่า
Sh_In3	มีขนาด 13 บิตทำหน้าที่เป็นอินพุทรับจำนวนครั้งในการเลื่อนค่าหรือค่าที่ต้องการเลื่อนซึ่งเป็นจำนวนเต็มที่ได้จากหน่วยควบคุม
Op_reg	ขนาด 2 บิตทำหน้าที่กำหนดรูปแบบของการเลื่อนค่าดังนี้ 00 ได้จำนวนครั้งจาก In1 โดยทำการเลื่อนค่าใน In2 01 ได้จำนวนครั้งจาก Sh_In3 และเลื่อนค่าใน In2 10 ได้จำนวนครั้งและค่าที่ต้องการเลื่อนจาก Sh_In3
Sh_mode	มีขนาด 2 บิตใช้ในการกำหนดชนิดของการเลื่อนค่าดังนี้ 00 เลื่อนค่าไปทางซ้ายแบบลอจิก (Logical shift left) 01 เลื่อนค่าไปทางขวาแบบลอจิก (Logical shift right) 10 เลื่อนค่าไปทางขวาแบบคิดบิตทด (Arithmetic shift right) 11 การผ่านค่า
Sh_rd	ใช้ในการอ่านค่าจากพอร์ทอินพุท
Cflag	บิตทดเพื่อใช้ในการเลื่อนค่าแบบคิดบิตทด
Shift_out	ค่าผลลัพธ์ที่ได้จากการเลื่อนค่า
Sh_flag	ค่าตัวบ่งชี้ที่เปลี่ยนแปลงเมื่อมีการเลื่อนค่า

ในหน่วยของการเลื่อนค่าสามารถแบ่งออกเป็นส่วนย่อยในการออกแบบการทำงานภายในซึ่งทำงานพร้อมๆกันได้ดังต่อไปนี้

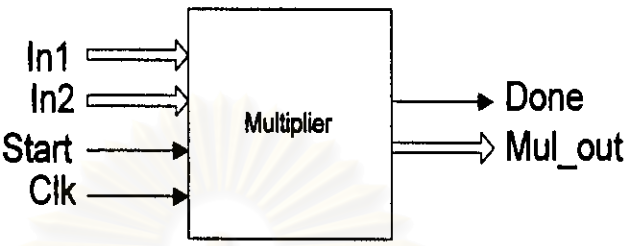
- ส่วนของการรับข้อมูลโดยจะทำงานตามสัญญาณนาฬิกา
- ส่วนของการเลื่อนค่าโดยจะทำการตรวจสอบจากฟังก์ชั่นที่ได้จากสัญญาณ Sh_mode และได้ผลลัพธ์ใส่ในพอร์ทของ Shift_out
- ส่วนของการคำนวณค่าตัวบ่งชี้ โดยค่าที่ได้จากการเปลี่ยนแปลงจะถูกส่งออกไปทางขา sh_flag ให้กับหน่วยควบคุม

ผลการสังเคราะห์วงจรส่วนตัวเลื่อนค่า

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	93 ขา
ตัวสร้างฟังก์ชั่น FG	636 ตัว

ตัวสร้างพังก์ชั่น H	249 ตัว
ฟลิปฟล็อป	0 ตัว
สามารถทำงานได้ที่ความถี่	69.8 MHz

3.3.3 ตัวคูณ (MULTIPLIER)



รูปที่ 3.6 หน่วยของการคูณ

ในงานวิจัยนี้ได้ออกแบบวงจรให้ทำการคูณแบบ Booth's Algorithm โดยในแต่ละรอบการทำงาน จะทำการพิจารณาบิตที่ 2 – 0 ของ ตัวถูกดำเนินการ ตัวคูณว่ามีค่าเป็นอะไร แล้วก็จะทำงานตามเงื่อนไขของค่านั้นๆ ดังนี้คือ

000 = Result เดิม +0

001 = Result เดิม + Operand1

010 = Result เดิม + Operand1

011 = Result เดิม + (2*Operand1)

100 = Result เดิม – (2*Operand1)

101 = Result เดิม – Operand1

110 = Result เดิม – Operand1

111 = Result เดิม - 0

จากรูปที่ 3.6 มีขาสัญญาณที่ใช้ดังนี้

In1 มีขนาด 32 บิต เป็นตัวตั้งของการคูณ

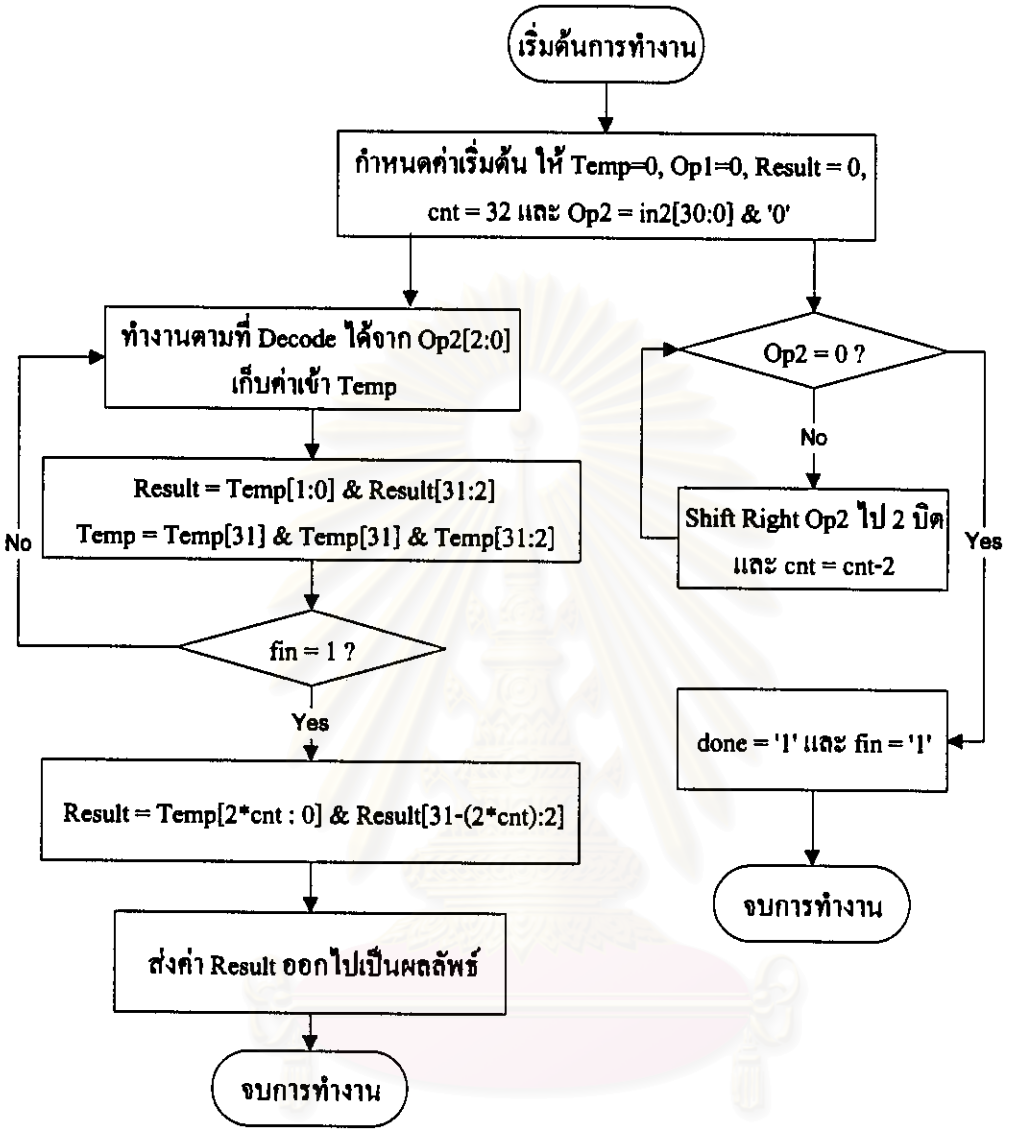
In2 มีขนาด 32 บิต เป็นตัวคูณ

Start เป็นสัญญาณเพื่อบอกให้หน่วยการคูณเริ่มทำงาน

Done เป็นสัญญาณที่สร้างขึ้นเมื่อทำการคูณเสร็จสิ้นส่งไปบอกยังหน่วยควบคุม

Mul_out มีขนาด 32 บิตเป็นผลคูณที่เกิดจาก In1 และ In2

Booth's Algorithm ที่ได้ทำการออกแบบมาใช้ในตัว ตัวคูณ นั้นมีการทำงานดังผังงานด้านล่างนี้



รูปที่ 3.7 แผนภาพการทำงานของการคูณแบบบูธ

ในที่นี่จะยกตัวอย่างของการคูณโดยใช้ค่าเพียง 8 บิตเท่านั้น โดยที่การทำงานกับ 32 บิตก็เหมือนกัน 01001001 * 00110110 (in1 * in2) โดยที่ In1 และ In2 เป็นพอร์ทที่ใช้รับข้อมูลขนาด 32 บิตและสัญญาณ Start จะทำหน้าที่ควบคุมการรับข้อมูล เมื่อทำงานเสร็จเรียบร้อยผลลัพธ์จะส่งออกทางพอร์ท Mul_out พร้อมกับให้สัญญาณ Done เพื่อบอกให้หน่วยควบคุมทราบเมื่อทำงานเสร็จ เนื่องจากการคูณจะเป็นต้องใช้เวลาในการทำงานมากกว่าปกติ ซึ่งอาจทำให้เกิดปัญหาของไปป์ไลน์ได้ จึงจำเป็นต้องมีการขยายช่วงเวลาของการทำงานจนกระทั่งการคูณจะเสร็จสิ้น

ตารางที่ 3.2 ตัวอย่างการทำงานของการคูณแบบ Booth's Algorithm

	Op1	Temp		Result	Op2	
Initial		0 00000000		0 00000000	00110110 0	in2 & '0'
Op1 = -2*in1	1 01101110	1 01101110 1 11011011	Temp + Op1 Shift right 2	1 00000000	0 00011011	Shift right 2
Op1 = +2*in1	0 10010010	0 01101101 0 00011011	Temp + Op1 Shift right 2	0 11000000	0 00000110	Shift right 2
Op1 = - in1	1 10110111	1 11010010 1 11110100	Temp + Op1 Shift right 2	1 00110000	0 00000001	Shift right 2
Op1 = + in1	0 01001001	0 00111101 0 00001111	Temp + Op1 Shift right 2	0 11001100	0 00000000	

ผลคูณที่ได้จะเป็น 8 บิตล่างเท่านั้น (การทำงานจริงก็คือ 32 บิตล่าง) ซึ่งก็คือค่า01100110 ตัวคูณประกอบด้วย 3 ส่วนด้วยกันคือ

3.3.3.1 ส่วนที่ทำการคำนวณ

จะทำงานที่ สัญญาณนาฬิกา ขาขึ้น ซึ่งจะทำการ ถอดรหัส ค่าของ ตัวถูกดำเนินการ ที่เป็นตัวคูณ 3 บิตหลัง แล้วทำการคำนวณตามการทำงานที่ ถอดรหัส ได้ และทำการ เลื่อนค่า ค่าผลลัพธ์ที่ได้ในแต่ละครั้งไปทางขวา 2 บิต โดยนำ 2 บิตที่ เลื่อนค่า ออกไปนั้นไปใส่ไว้ในตัวเก็บค่าผลลัพธ์ (ซึ่งก็จะถูก เลื่อนค่า ขวาไป 2 บิตเช่นกัน แล้วเติมด้วย 2 บิตท้ายของค่าผลลัพธ์ที่ได้จากการคำนวณในแต่ละครั้งนั่นเอง)

3.3.3.2 ส่วนควบคุมซึ่งทำการกำหนดค่าสัญญาณการเสร็จสิ้นการทำงาน

จะทำงานที่ สัญญาณนาฬิกา ขาลง ซึ่งจะทำการพิจารณาค่า Op2 (ซึ่งเป็นค่า ตัวถูกดำเนินการ 31 บิต ที่ได้ มาจากการเติมค่าบิตที่ 30 ด้วยค่า 0 และบิตที่ 29-0 นั้นได้มาจากค่าบิตที่ 30-1 ของค่า ตัวถูกดำเนินการที่เป็นตัวคูณ) ว่าเป็นศูนย์หรือยัง ถ้าใช่ก็จะทำการส่งค่าสัญญาณ done และ fin ไปให้หน่วยควบคุมและส่วนที่ทำการคำนวณตามลำดับ ถ้าไม่ก็จะทำการลดค่า Count ลงไป 2 (เริ่มต้นการทำงานจะมีค่าเป็น 32) และทำการ เลื่อนค่า ค่า Op2 ไปทางขวา 2 บิต

3.3.3.3 ส่วนที่ทำการส่งค่าผลลัพธ์

จะทำงานที่ สัญญาณนาฬิกา ขาขึ้น และเมื่อค่าสัญญาณ fin มีค่าเป็น 1 โดยจะทำการส่งค่า Result = Temp[cnt : 0] & Result[31-cnt:2] เมื่อ Temp คือ ตัวเก็บค่าผลลัพธ์ที่ได้จากการประมวลผลตามที่ ถอดรหัสได้ในแต่ละครั้ง โดยที่ Result คือ ตัวเก็บค่า 2 บิตหลังของ Temp ซึ่งจะถูก เลื่อนค่า ไปทางขวา 2 บิตเพื่อรับค่าใหม่ และ Cnt คือ ตัวเก็บจำนวนครั้งในการ เลื่อนค่า ก่อนที่จะส่งผลลัพธ์ออกไปนั่นเอง

ผลลัพธ์ที่ได้ออกมาจะมีค่าเพียง 32 บิตล่างโดยเมื่อทำการคูณเสร็จ จะทำการส่งสัญญาณ Done ไปบอก หน่วยควบคุม เพราะว่าระยะเวลา (Cycle Time) ในการคูณนั้นจะขึ้นตามขนาดของ ตัวถูกดำเนินการ ที่เป็นตัวคูณดังนี้

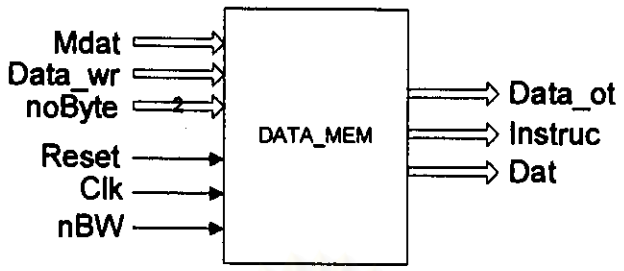
ตารางที่ 3.3 ค่าตัวถูกดำเนินการของการคูณ

Min	Max	Speed
0	1	1
2	7	2
8	31	3
32	127	4
128	511	5
512	2047	6
2048	8191	7
8192	32767	8
32768	131071	9
131072	524287	10
524288	2097151	11
2097152	8388607	12
8388608	33554431	13
33554432	134217727	14
134217728	536870911	15
536870912	2147483648	16

ผลการสังเคราะห์วงจรส่วนการคูณ

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	100 ขา
ตัวสร้างฟังก์ชั่น FG	558 ตัว
ตัวสร้างฟังก์ชั่น H	178 ตัว
ฟลิปฟล็อป	267 ตัว
สามารถทำงานได้ที่ความถี่	26.7 MHz

3.3.4 ส่วนติดต่อกับหน่วยความจำ (Data Memory)



รูปที่ 3.8 ส่วนติดต่อหน่วยความจำ

เป็นส่วนที่จัดการกับข้อมูลที่อ่านมาหรือ ที่จะส่งไปให้หน่วยความจำ ให้สามารถนำไปใช้งานได้เมื่อเป็นไบต์หรือเวิร์ดโดยมีการจัดการดังนี้

1. *การโหลดค่าข้อมูลมาจากหน่วยความจำ (Load)*

กรณีเป็นการโหลดแบบไบต์ (Byte) จะนำข้อมูลที่อ่านมาได้จากขาสัญญาณ Mdat ในไบต์ที่ต้องการไปไว้ในสัญญาณ Data_ot [7:0] และใส่บิตที่เหลือด้วยค่าศูนย์ ซึ่งตำแหน่งไบต์ที่ต้องการนั้นพิจารณาจากค่าAddress[1:0] แล้วส่งค่านั้นออกไปเพื่อเก็บค่าลง รีจิสเตอร์ กรณีเป็นการโหลดแบบ เวิร์ด (Word) จะนำค่าข้อมูลที่อ่านมาได้ส่งออกไปให้ รีจิสเตอร์ เลย

2. *การเก็บค่าข้อมูลเข้าหน่วยความจำ (Store)*

กรณีเป็นการเก็บค่าเข้าหน่วยความจำแบบไบต์ (Byte) จะนำค่า 8 บิตล่างของ Data_wr (ซึ่งก็คือค่าของรีจิสเตอร์) ใส่ลงไปทั้ง 4 ไบต์ของ บัสข้อมูลแล้วส่งค่าไปยังหน่วยความจำ กรณีเป็นการเก็บค่าเข้าหน่วยความจำแบบเวิร์ด (Word) จะนำค่าทั้ง 32 บิตของ รีจิสเตอร์ ใส่ลงไปในบัสข้อมูล

จากการศึกษาได้ทำการออกแบบให้ส่วนนี้มีขาสัญญาณเพื่อให้สามารถใช้งานดังรูปที่ 3.8 และมีรายละเอียดของสัญญาณดังนี้

Mdat	ขนาด 32 บิต เป็นบัสรับข้อมูล
Data_wr	ขนาด 32 บิต เป็นข้อมูลที่ต้องการเก็บในหน่วยความจำ
NoByte	ขนาด 2 บิต เพื่อเลือกว่าต้องการใช้ข้อมูลไบต์ไหน
NBW	ใช้เลือกว่าต้องการใช้ทีละไบต์หรือเวิร์ด
Data_ot	ขนาด 32 บิต เป็นข้อมูลที่ได้จัดเรียงแล้ว หลังจากเลือกไบต์หรือเวิร์ด
Instruc	ขนาด 32 บิต รับคำสั่งที่ถูกนำเอาเข้ามาทำงานในไมโครโพรเซสเซอร์
Dat	ขนาด 32 บิตใช้ส่งข้อมูลที่ต้องเก็บสู่หน่วยความจำ

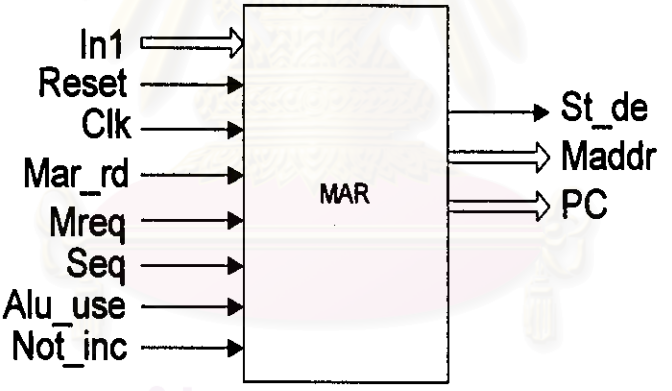
ส่วนของการติดต่อหน่วยความจำได้ทำการออกแบบโดยแบ่งเป็นส่วนย่อยภายในออกเป็นดังนี้

- ส่วนของการรับและส่งข้อมูล
- ส่วนของการเก็บข้อมูลเพื่อจัดเก็บลงสู่หน่วยความจำ
- ส่วนของการจัดเรียงข้อมูลเมื่อมีการเลือกไบต์หรือเวิร์ด

ผลการสังเคราะห์วงจรส่วนการติดต่อหน่วยความจำ

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	166 ขา
ตัวสร้างฟังก์ชั่น FG	88 ตัว
ตัวสร้างฟังก์ชั่น H	8 ตัว
ฟลิปฟล็อป	0 ตัว
สามารถทำงานได้ที่ความถี่	69.8 MHz

3.3.5 ส่วนจัดการและเก็บเลขที่อยู่



รูปที่ 3.9 ส่วนจัดการและเก็บเลขที่อยู่

เป็นส่วนที่จัดการส่งค่าตำแหน่งที่ต้องการจะอ้างอิงหน่วยความจำออกไป โดยมีโหมดการทำงานด้วยกันดังนี้

1. *อ้างอิงตำแหน่งโดยใช้ค่าจาก พีซี เพื่อเป็นการนำคำสั่งถัดไปเข้ามาเตรียมไว้*

จะทำการส่งค่า Program Counter (PC) ซึ่งได้ทำการนับเพิ่มไป 4 ตำแหน่งไว้แล้ว ส่งออกไปที่บัสเลขที่อยู่ซึ่งเป็นการอ้างอิงหน่วยความจำ เพื่อนำคำสั่งถัดไปเข้ามาเตรียมไว้ จากนั้นจะทำการเพิ่มค่า พีซี ไป 4 ตำแหน่ง

2. *อ้างอิงตำแหน่งโดยใช้ค่าจาก หน่วยคำนวณและตรรกะ โดยไม่มีการเพิ่มค่า พีซี*

จะทำการนำค่าที่ได้มาจาก หน่วยคำนวณและตรรกะ ส่งออกไปที่บัสเลขที่อยู่โดยที่จะไม่ทำการเพิ่มค่าของพีซีเพื่อทำการ Load, Swap, Store แบบ Pre-Index รวมทั้ง Multiple Load และ Multiple Store ค่าแรกจากหรือลงหน่วยความจำ

3. *อ้างอิงตำแหน่งโดยใช้ค่าจาก TO_MAR*

จะใช้ค่าจาก TO_MAR ซึ่งได้ทำการเก็บค่าไว้ใน วัฏจักร ที่แล้ว โดยเป็นค่าที่อ่านมาจาก รีจิสเตอร์ เพื่อใช้เป็นตำแหน่งอ้างอิงเพื่อการStore แบบ Post-Index ส่งออกไปที่บัสเลขที่อยู่โดยที่ไม่ทำการเพิ่มค่าของ พีซี

4. *อ้างอิงตำแหน่งโดยใช้ค่าจาก หน่วยคำนวณและตรรกะ และมีการเพิ่มค่า พีซี*

จะใช้ค่าจาก หน่วยคำนวณและตรรกะ ส่งออกไปที่บัสเลขที่อยู่ซึ่งเป็นการอ้างอิงหน่วยความจำ เพื่อกระโดด (Branch)ไปตำแหน่งที่ต้องการ และเพิ่มค่า พีซี โดยใช้ค่าของ ALU+4 เพื่อที่จะไว้อ้างอิงตำแหน่ง ในการนำเอาคำสั่งมาในรอบถัดไป

5. *อ้างอิงตำแหน่งจากตำแหน่งเดิมไปอีก 4*

จะใช้ค่าตำแหน่งเดิมเพิ่มไปอีก 4 ส่งออกไปที่บัสเลขที่อยู่เพื่อที่จะอ่านค่าจากหรือเก็บค่าเข้าหน่วยความจำลงไปในตำแหน่งถัดจากเดิม ใช้สำหรับในการทำคำสั่ง Multiple Load และ Multiple Store ซึ่งไม่ใช่การโหลดหรือเก็บค่าแรกลงหน่วยความจำ

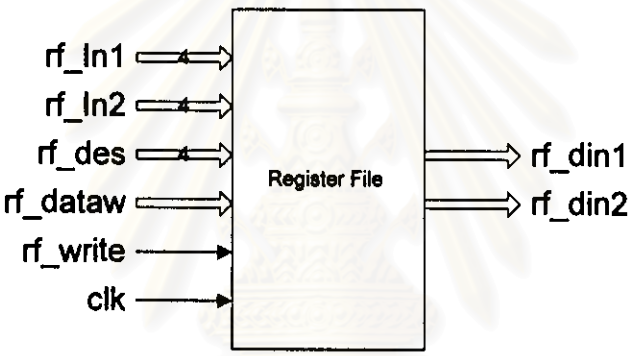
ส่วนของการจัดค่าตำแหน่งเลขที่อยู่ได้ทำการออกแบบโดยให้มีสัญญาณการใช้งานดังต่อไปนี้

In1	ขนาด 32 บิตรับค่าเลขที่อยู่
MAR_rd	สัญญาณเพื่อให้รับค่าในอินพุทเข้ามา
Mreq	ตรวจว่ามีการเรียกใช้หน่วยความจำหรือไม่
Seq	สัญญาณเพื่อบอกถึงลำดับการทำงานว่าต่อเนื่องหรือไม่
Alu_use	มีขนาด 2 บิตเพื่อเลือกรูปแบบของการเปลี่ยนค่าเลขที่อยู่ ดังนี้
	00 mar <= pc , pc <= pc + "100"
	01 mar <= to_mar (store in post index)
	10 mar <= alu_res , pc <= alu_res + "100"
	11 mar <= alu_res
Not_inc	สัญญาณบอกเมื่อไม่ต้องการเพิ่มค่าเลขที่อยู่
St_de	สัญญาณที่สร้างขึ้นให้สอดคล้องกับสัญญาณนาฬิกาเพื่อนำไปใช้ในการตรวจสอบ
Maddr	ขนาด 32 บิตค่าเลขที่อยู่ที่ได้ทำการเปลี่ยนแปลงแล้ว
PC	ขนาด 32 บิตค่าเลขที่อยู่ที่เก็บค่าตำแหน่งที่ทำงาน ณ. ปัจจุบัน

ผลการสังเคราะห์วงจรส่วนการติดต่อหน่วยความจำ

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	102 ขา
ตัวสร้างฟังก์ชั่น FG	155 ตัว
ตัวสร้างฟังก์ชั่น H	38 ตัว
ฟลิปฟล็อป	97 ตัว
สามารถทำงานได้ที่ความถี่	11.5 MHz

3.3.6 รีจิสเตอร์ (REGISTER FILE)



รูปที่ 3.10 ส่วนของรีจิสเตอร์

ประกอบด้วยรีจิสเตอร์ 32 บิตจำนวน 16 รีจิสเตอร์ คือ ตั้งแต่ R0-R15 ในบางคำสั่งจะไม่ควรใช้ค่าของ R15 เนื่องจากอาจทำให้เกิดความผิดพลาดได้ ซึ่ง R15 นั้นคือพีซี และมี R14 เป็นรีจิส-เตอร์ที่จะใช้เก็บค่าตำแหน่งเดิมที่ต้องการกระโดดกลับมา (ใช้ในคำสั่ง Branch with Link ซึ่งจะเป็นตำแหน่งของคำสั่งถัดจากคำสั่ง Branch) รีจิสเตอร์ นั้นจะมี 2 ช่องทางการอ่าน ทำให้สามารถทำการอ่านค่าของรีจิสเตอร์ ได้พร้อมกัน 2 ตัว จากรูปที่ 3.10 ได้ทำการออกแบบส่วนนี้และสามารถแจงรายละเอียดการทำงานของแต่ละขาสัญญาณได้ดังนี้

Rf_in1	ขนาด 4 บิตใช้ระบุรีจิสเตอร์ที่ใช้เป็นตัวตั้ง
Rf_in2	ขนาด 4 บิตใช้ระบุรีจิสเตอร์ที่ใช้เป็นตัวกระทำ
Rf_des	ขนาด 4 บิตใช้ระบุรีจิสเตอร์ที่เก็บผลลัพธ์
Rf_dataw	ขนาด 32 บิตเป็นพอร์ทการเขียนข้อมูลลงสู่รีจิสเตอร์
Rf_write	สัญญาณเลือกระหว่างการเขียนหรืออ่านค่าจากรีจิสเตอร์
Rf_din1	ขนาด 32 บิตเป็นข้อมูลที่ส่งออกสู่บัสข้อมูลที่ 1
Rf_din2	ขนาด 32 บิตเป็นข้อมูลที่ส่งออกสู่บัสข้อมูลที่ 2

การออกแบบในส่วนนี้ความง่ายโดยการประกาศจองเนื้อที่ว่างเป็นลักษณะของตารางของแถวลำดับ(Array) เพื่อเก็บค่าของรีจิสเตอร์ 16 ตัว โดยมีการเข้าถึงแต่ละแถวและข้อมูลได้จากสัญญาณ rf_in1, rf_in2 และ rf_des นอกจากนี้จะทำการตรวจสอบสถานะว่าเป็นการอ่านหรือเขียนข้อมูลจากสัญญาณ Rf_write เพื่อส่งข้อมูลออกทางพอร์ทที่ต้องการ

ผลการสังเคราะห์วงจรส่วนรีจิสเตอร์

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	110 ขา
ตัวสร้างฟังก์ชั่น FG	628 ตัว
ตัวสร้างพังก์ชั่น H	192 ตัว
ฟลิปฟล็อป	512 ตัว
สามารถทำงานได้ที่ความถี่	33.8 MHz

พบว่าผลจากการสังเคราะห์วงจรมีส่วนของฟลิปฟล็อป ที่ใช้งานจำนวนมากเนื่องจากใช้ในการค่าในแต่ละบิตและแถวของรีจิสเตอร์ นั้นคือ รีจิสเตอร์16 ตัวแต่ละตัวมี 32 บิต จึงต้องใช้งานถึง 512 ฟลิปฟล็อป

3.3.7 ส่วนควบคุม (Control Unit)

ประกอบด้วยกัน 2 ส่วนคือ

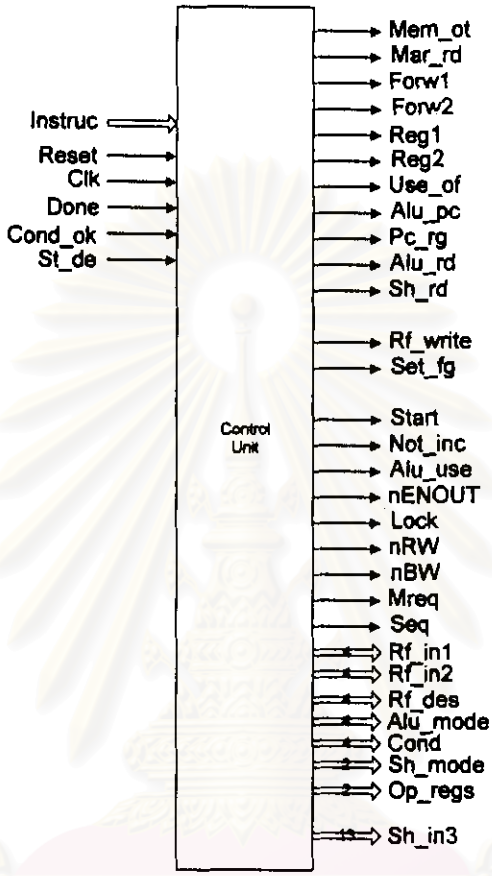
1. ส่วนที่ทำการเอาค่ามา และถอดรหัส คำสั่ง

ในการนำเอาคำสั่งมาจะรับคำสั่งมาจาก ส่วนติดต่อกับหน่วยความจำ โดยหากคำสั่งที่กำลังทำการ การประมวลผล อยู่นั้นเป็นคำสั่งที่มีการทำงานกับหน่วยความจำ จะมีการรำเอาคำสั่งมาเก็บไว้ในที่พักข้อมูลก่อน แทนที่จะทำการเก็บลงรีจิสเตอร์คำสั่ง

ในการ ถอดรหัส จะทำการถอดรหัสคำสั่งที่อยู่ในรีจิสเตอร์คำสั่งว่าคำสั่งใด เพื่อกำหนดสัญญาณควบคุมต่างๆ ไปยังแต่ละชิ้นส่วน โดยที่ในหนึ่งคำสั่งอาจจะมีการถอดรหัส มากกว่า 1 คาบ สัญญาณนาฬิกา ส่วน DataProcessing แบบที่มีการเลื่อนค่าที่มี ตัวถูกดำเนินการ เป็นค่าของรีจิสเตอร์ และจำนวนครั้งในการ เลื่อนค่ามาจากค่า 8 บิตล่างของรีจิสเตอร์ในวัฏจักรแรกจะทำการ ถอดรหัส เพื่อให้ได้ค่าของรีจิสเตอร์ที่เป็นจำนวนครั้งในการเลื่อนค่า ส่งไปให้ค่าที่จะทำการเลื่อน ส่วนในรอบที่สองของการทำงาน จะทำการถอดรหัสเพื่อให้ได้ค่ารีจิสเตอร์มาทำการประมวลผลในหน่วยคำนวณและตรรกะ

2. *ส่วนที่ควบคุมการ การประมวลผล ของคำสั่งในแต่ละรอบการทำงาน*

จะทำการกำหนดสัญญาณต่างๆ ไปยังแต่ละ ชิ้นส่วน ให้ทำงาน โดยที่แต่ละคำสั่งนั้นจะมีคาบในการประมวลผล ไม่เท่ากัน



รูปที่ 3.11 ส่วนควบคุม

จากการออกแบบได้ขาสัญญาณตามรูปที่ 3.11 โดยจะเป็นส่วนของการส่งสัญญาณไปควบคุมชิ้นส่วนต่างๆที่ทำงานให้เป็นระบบสอดคล้องกัน และส่วนของการรับคำสั่งจากหน่วยความจำ เพื่อทำการถอดรหัสและสร้างสัญญาณเพื่อควบคุมการทำงาน พบว่าการออกแบบในส่วนนี้จะมีขนาดใหญ่ และซับซ้อนกว่าส่วนอื่นๆ ดังนั้นการออกแบบที่ดีควรวางการไหลหรือการทำงานทั้งหมดของหน่วยนี้ก่อนเริ่มลงมือเขียนโค้ดการออกแบบ

ขาสัญญาณของส่วนควบคุมสามารถแบ่งออกได้เป็น 3 ส่วนคือ

1. ส่วนของสัญญาณที่ส่งไปควบคุมในแต่ละหน่วยที่ได้ออกแบบดังนี้แสดงรายละเอียดไว้ในข้อ 3.3.1 – 3.3.6 ดังต่อไปนี้

สัญญาณที่ส่งไปหน่วยคำนวณและตรรกะ	Alu_rd, Set_fg, Alu_mode, Cond
สัญญาณที่ส่งมาจากหน่วยคำนวณและตรรกะ	Cond_ok
สัญญาณที่ส่งไปหน่วยเลื่อนค่า	sh_rd, sh_mode, op_regs, sh_in3
สัญญาณที่ส่งไปหน่วยการคูณ	Start
สัญญาณที่ส่งมาจากหน่วยการคูณ	Done
สัญญาณที่ส่งไปหน่วยติดต่อหน่วยความจำ	n BW
ข้อมูลที่ส่งมาจากหน่วยติดต่อหน่วยความจำ	Instruc
สัญญาณที่ส่งไปส่วนคำนวณเลขที่อยู่	Mar_rd, Not_inc, Alu_use, Mreq, Seq
สัญญาณที่มาจากส่วนคำนวณเลขที่อยู่	St_de
สัญญาณที่ส่งไปหน่วยรีจิสเตอร์	Rf_write, Rf_in1, Rf_in2, Rf_des

2. สัญญาณส่วนของการทำไปป์ไลน์และกำหนดให้เป็นเอาท์พุทของหน่วยควบคุมเพื่อตรวจสอบความถูกต้องมีดังนี้

สัญญาณในการทำ Forwarding เพื่อแก้ปัญหาข้อมูลในไปป์ไลน์	Forw1, Forw2, Reg1, Reg2, Use_of
สัญญาณในการแก้ปัญหาจากการควบคุมในไปป์ไลน์	Alu_pc, Pc_rg,

3. สัญญาณที่ส่งออกจากตัวไมโครโพรเซสเซอร์อาร์ม และมีการทำงานดังนี้

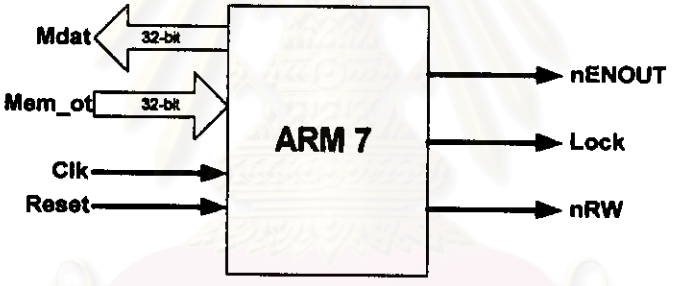
Mdat	บัสข้อมูลขนาด 32 บิต เป็นอินพุทที่เข้าสู่ไมโครโพรเซสเซอร์
Mem_ot	บัสข้อมูลขนาด 32 บิต เป็นเอาท์พุทที่ออกจากไมโครโพรเซสเซอร์
nENOUT	สัญญาณส่งไปยังหน่วยความจำเพื่อให้หน่วยความจำเริ่มทำงาน
Lock	สัญญาณที่ส่งออกเพื่อให้ทราบว่าไมโครโพรเซสเซอร์ทำการเข้าถึงข้อมูลของหน่วยความจำ เพื่อป้องกันไม่ให้ส่วนอื่นเข้ามารบกวนการทำงาน
nRW	สัญญาณติดต่อหน่วยความจำเพื่อบอกว่าเป็นการอ่านหรือเขียนข้อมูล
Clk	สัญญาณนาฬิกาหลักที่เข้าสู่ไมโครโพรเซสเซอร์
Reset	สัญญาณที่ใช้ในการเริ่มต้นการทำงาน

ผลการสังเคราะห์วงจรส่วนควบคุม

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	127 ขา
ตัวสร้างฟังก์ชั่น FG	631 ตัว
ตัวสร้างพังก์ชั่น H	240 ตัว
ฟลิปฟล็อป	228 ตัว
สามารถทำงานได้ที่ความถี่	9.6 MHz

พบว่าจากการสังเคราะห์วงจรได้ส่วนของหน่วยควบคุมดังผลข้างต้น และเป็นส่วนที่ใช้ความถี่ของสัญญาณนาฬิกาที่น้อยที่สุด เนื่องจากค่าความหน่วงของเวลาเกิดขึ้นในส่วนนี้มากที่สุด

3.3.8 ไมโครโพรเซสเซอร์อาร์ม



รูปที่ 3.12 ขาสัญญาณของไมโครโพรเซสเซอร์อาร์ม 7

เมื่อเชื่อมส่วนต่างๆเข้าด้วยกันทั้งหมด จึงนำไปสังเคราะห์วงจรรวมเพื่อให้ได้ไมโครโปรเซสเซอร์อาร์มตามที่ต้องการดังรูปที่ 3.12 โดยผลจากการสังเคราะห์วงจรมีดังนี้

ทรัพยากร	ใช้งาน
ขาอินพุทเอาท์พุท	109 ขา
ตัวสร้างฟังก์ชั่น FG	3109 ตัว
ตัวสร้างพังก์ชั่น H	1015 ตัว
ฟลิปฟล็อป	1375 ตัว
สามารถทำงานได้ที่ความถี่	9 MHz

โดยส่วนของรีจิสเตอร์จะเป็นส่วนที่ใช้ฟลิปฟล็อปในการทำงานสูงที่สุดถึง 512 ตัวและส่วนที่ใช้วงจรเชิงผสมมากที่สุดคือตัวเลื่อนค่า โดยใช้ทั้งหมด 858 ตัว ส่วนหน่วยที่ทำงานเร็วที่สุดคือหน่วยเลื่อนค่า หน่วยติดต่อหน่วยความจำ และหน่วยคำนวณและตรรกะ