



ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

ภาคการศึกษาที่.....ปีการศึกษา.....

รหัสวิชา 010113026 ชื่อวิชา Digital Laboratory

ตอนเรียน หมายเลขโต๊ะ.....

รหัสนักศึกษา..... ชื่อ-นามสกุล.....

อาจารย์ผู้สอน.....เวลาที่ทำการทดลอง วันที่.....

Work Sheet II

Simple Calculator using “Datapath and Control Unit” Technique

วัตถุประสงค์

1. เพื่อให้สามารถประยุกต์ใช้แนวทางออกแบบวงจรลอจิกโดยใช้เทคนิค “Datapath and Control Unit”

อุปกรณ์

1. ระบบคอมพิวเตอร์ 1 เครื่อง พร้อมติดตั้งโปรแกรม Quartus II เวอร์ชัน 8.0 (Student Edition) ขึ้นไป
2. บอร์ดทดลอง Cyclone3-Lab01 1 บอร์ดพร้อมคู่มือการใช้งาน
3. สาย J-TAG 1 เส้น ใช้รุ่น USB-Blaster (สำหรับเครื่อง Notebook) หรือรุ่น Byte-Blaster (สำหรับเครื่อง PC)

1.แนวคิดของการออกแบบด้วยเทคนิค “Datapath and Control Unit”

การประยุกต์วงจรดิจิทัลเพื่อการประมวลผลทางคณิตศาสตร์ ก็เป็นสิ่งที่มักพบเห็นได้บ่อยในงานด้านวิศวกรรม ดังนั้นการศึกษานำทางการออกแบบวงจรหรือระบบงานเพื่อใช้ประมวลผลข้อมูลที่มีความซับซ้อนโดยใช้ความรู้ด้านวงจรดิจิทัลจึงเป็นสิ่งจำเป็นสำหรับนักออกแบบทางวิศวกรรม

เทคนิคที่ใช้ในการออกแบบวงจรประมวลผลข้อมูลมีหลากหลายแนวทางและไม่มีข้อกำหนดตายตัวใดๆ แต่จะขึ้นอยู่กับว่าเป้าหมายที่ต้องการของผู้ออกแบบนั้นเป็นเช่นใด อย่างไรก็ตามมีเทคนิคการออกแบบแนวทางหนึ่งที่ได้รับนิยมนิยม และเป็นแนวทางที่น่าศึกษาไว้เป็นพื้นฐานของการออกแบบคือเทคนิคที่มีชื่อว่า “Datapath and Control Unit” (เป็นแนวทางนิยมใช้ออกแบบสถาปัตยกรรมของหน่วยประมวลผลในซีพียู) โดยแบ่งโครงสร้างของระบบงานออกเป็นสองส่วนคือ

- 1) เส้นทางเคลื่อนที่ของข้อมูล (**Datapath**) ของเราจะต้องเคลื่อนที่ผ่านวงจรตัวกระทำทางคณิตศาสตร์ (Arithmetic Operator) ต่างๆ ที่มีอยู่ในระบบ
- 2) หน่วยควบคุมขั้นตอนการทำงาน (**Control Unit**) จะเป็นระบบหรือวงจรที่มีลักษณะคล้ายกับวงจรซีควเอนเชียล (Sequential) โดยจะทำงานอย่างเป็นจังหวะตามสัญญาณนาฬิกาเพื่อจัดลำดับและปรับเปลี่ยนทิศทางการเคลื่อนที่ของข้อมูลว่าจะให้ผ่านตัววงจรตัวกระทำทางคณิตศาสตร์ใดบ้างจึงจะได้ผลลัพธ์ตามที่ต้องการตัวอย่างวงจรซีควเอนเชียลที่เราได้เคยออกแบบมาแล้วคือวงจร Finite State Machine (ในการทดลองที่ 9 และ 10)

เพื่อให้เห็นภาพของ “Datapath and Control Unit” ที่ชัดเจนมากขึ้นเราจะลองออกแบบเครื่องคำนวณง่ายๆที่คิดเลขเฉพาะเลขจำนวนเต็ม (Integer) เท่านั้น และความสามารถของเครื่องคิดเลขตัวนี้ก็ไม่ต้องสูงมากเอาเพียงแค่ สามารถบวกลบเลขจำนวนเต็ม (เลขฐานสิบ) ขนาดเพียงไม่เกินสามหลักให้ได้อย่างถูกต้องเท่านั้น



2. Simple Calculator

การออกแบบสร้างเครื่องคำนวณอย่างง่ายนั้นดูเหมือนว่าไม่น่าจะยากหรือซับซ้อนอย่างใด น.ศ.บางคนอาจจะถึงกับบอกว่าเราใช้วงจร Adder ที่เคยเรียนมาก็น่าจะเพียงพอแล้ว แต่ในความเป็นจริงไม่เป็นเช่นนั้นเพราะเครื่องคิดเลขของเราควรจะต้องใช้งานได้ง่ายและเป็นมิตรกับผู้ใช้งานมันด้วย ดังนั้นเราจึงมีลำดับขั้นตอนการออกแบบเป็นหลายๆขั้นตอนดังตัวอย่างต่อไปนี้

2.1 ธรรมชาติของเครื่องคิดเลข และการปฏิสัมพันธ์กับผู้ใช้

ก่อนที่เราจะออกแบบสร้างอุปกรณ์ใดๆ สิ่งแรกที่เราผู้ออกแบบจะต้องรู้คือ ความเป็นธรรมชาติพื้นฐานของของสิ่งนั้น (ถ้าบังเอิญเป็นของใหม่ที่ไม่เคยมีในโลก ก็คงต้องใช้จินตนาการ**เพื่อให้รู้**) เพื่อให้เราสามารถนึกภาพที่จะเกิดขึ้นได้หากเมื่อของสิ่งนั้นถูกนำไปใช้งานจริงๆ โดยผู้อื่นที่ไม่ใช่ผู้ออกแบบเอง ด้วยวิธีการเช่นนี้จะทำให้ผู้ออกแบบสามารถวางข้อกำหนดรายละเอียดของการออกแบบได้ดีขึ้น รวมไปถึงป้องกันปัญหาที่อาจจะเกิดขึ้นอันเนื่องจากการใช้งานที่ผิดพลาดได้ด้วย ดังนั้นในขั้นนี้เราจะมาลองจินตนาการถึง**หน้าตา/การตอบสนอง**ต่อผู้ใช้ของเครื่องคิดเลขที่เราคิดสร้างขึ้น (ในที่นี้ขอตั้งชื่อว่า **My Calculator**) ว่าควรมีพฤติกรรมเป็นเช่นไร

2.1.1 เครื่องคิดเลขในจินตนาการของฉัน

ก) เมื่อเปิดเครื่อง จะแสดงผลลัพธ์ (มีค่าเป็น 0) ที่จอแสดงผล (ในที่นี้ใช้ 7-Segment) เป็น **8888**

ข) เมื่อกดปุ่ม **Clear** จะแสดงผลลัพธ์ (มีค่าเป็น 0) ที่จอแสดงผลเป็น **8888**

ค) เมื่อทำการบวกเลข เช่น $98+50 =$ จะมีขั้นตอนตามตัวอย่างดังนี้


เริ่มต้น		ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	9	ที่หน้าจอแสดงผลเป็น	8888
ตามด้วยปุ่ม	8	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	Add	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	5	ที่หน้าจอแสดงผลเป็น	8888
ตามด้วยปุ่ม	0	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	=	ที่หน้าจอแสดงผลเป็น	8888

ง) เมื่อทำการบวกเลขหลายๆจำนวนเช่น $98+50+24+ \dots$ จะมีขั้นตอนตามตัวอย่างดังนี้

เริ่มต้น		ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	9	ที่หน้าจอแสดงผลเป็น	8888
ตามด้วยปุ่ม	8	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	Add	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	5	ที่หน้าจอแสดงผลเป็น	8888
ตามด้วยปุ่ม	0	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	Add	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	2	ที่หน้าจอแสดงผลเป็น	8888
ตามด้วยปุ่ม	4	ที่หน้าจอแสดงผลเป็น	8888
กดปุ่ม	Add	ที่หน้าจอแสดงผลเป็น	8888



จ) ในกรณีที่เป็นการลบก็กระทำเช่นเดียวกันกับข้อ ค) และ ง) เพียงแต่เปลี่ยนเครื่องหมายเป็นปุ่ม **Sub**

ฉ) ในกรณีที่เลขมีจำนวนเป็นค่าลบจะใช้ปุ่ม **-Sign** และ LED  ติดสว่างแสดงค่าที่เป็นลบ

จากแนวทางของเครื่องคิดเลข (อย่างง่าย) ที่ได้กล่าวไว้ข้างต้นก็จะแสดงให้เห็นความสามารถของมันว่าครอบคลุมการบวกและลบ แบบพื้นฐานได้แล้ว ต่อจากนี้เราจะเข้าสู่ขั้นตอนการออกแบบในรายละเอียดต่อไป

2.2 กำหนดคุณสมบัติ (Specification) ของเครื่องคิดเลขที่เราต้องการ ในความเป็นจริงของงานออกแบบทางวิศวกรรม จำเป็นจะต้องตั้งเป้าหมายไว้ก่อนว่าอยากได้เครื่องอะไร ทำอะไรได้บ้าง (ดังที่ได้ยกตัวอย่างไว้ในข้อ 2.1.1) ดังนั้นเราจำเป็นต้องวางกรอบหรือขอบเขตการออกแบบเสียก่อน เพื่อจะได้กำหนดรายละเอียดอื่นๆได้อย่างถูกต้อง แต่ทั้งนี้รายละเอียดต่างๆ ขอให้ขึ้นอยู่กับ **มุมมอง แนวคิด และจุดมุ่งหมายของผู้ออกแบบ** แต่ละคนเป็นหลัก ส่วนในเอกสาร Work Sheet ชุดนี้จะเสนอเพียงแนวทางการออกแบบพอให้เป็นแนวหรือตัวอย่างเพื่อเป็นแรงบันดาลใจต่อผู้ที่สนใจบ้างเท่านั้น

2.2.1 เป้าหมายของการออกแบบ

“ออกแบบเครื่องคำนวณง่ายๆที่สามารถแสดงผลลัพธ์ของการบวก ลบ เลขจำนวนเต็ม (เลขฐานสิบ) ขนาดเพียงไม่กี่หลักได้อย่างถูกต้องเท่านั้น”

2.2.2 รายละเอียดของเครื่องคิดเลข

ก) เครื่องคิดเลขของเรามีส่วนประกอบอะไรบ้าง เพื่อใช้สำหรับติดต่อกับผู้ใช้งาน (User Interface)

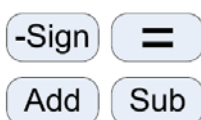
0000

ตัวแสดงผลแบบ 7-Segment 4 หลัก ใช้แสดงเลข 0 ถึง 9999



แป้นพิมพ์ ตัวเลข 0 ถึง 9

(ไม่มีความจำเป็นต้องใช้ปุ่ม # และ * แต่อย่างใด)



ปุ่มสำหรับบวก “Add” ใช้ปุ่ม PB2 ของบอร์ดทดลอง Cyclone3-Lab01

ปุ่มสำหรับลบ “Sub” ใช้ปุ่ม PB3 ของบอร์ดทดลอง Cyclone3-Lab01

ปุ่มประมวลผล “=” ใช้ปุ่ม PB1 ของบอร์ดทดลอง Cyclone3-Lab01

ปุ่มแสดงค่าลบ “-Sign” ใช้เมื่อต้องการกำหนดให้ค่าตัวเลขที่ป้อนนั้นเป็นลบ

ใช้ปุ่ม PB0 ของบอร์ดทดลอง Cyclone3-Lab01

SW0



Clear

ปุ่มสำหรับเคลียร์ค่าของผลลัพธ์ หรือเริ่มต้นก่อนการคำนวณ

ใช้สวิตช์เลื่อน SW0 ของบอร์ดทดลอง Cyclone3-Lab01



หลอด LED D0 ของบอร์ดทดลอง Cyclone3-Lab01

ใช้สำหรับแสดงเครื่องหมายของตัวเลขที่ปรากฏอยู่ที่ 7-Segment ในขณะนั้น

ข) เครื่องคิดเลขของเราทำอะไรได้บ้าง

- แสดงค่าผลลัพธ์ได้อย่างถูกต้องในช่วง -9999 ถึง +9999 เท่านั้น

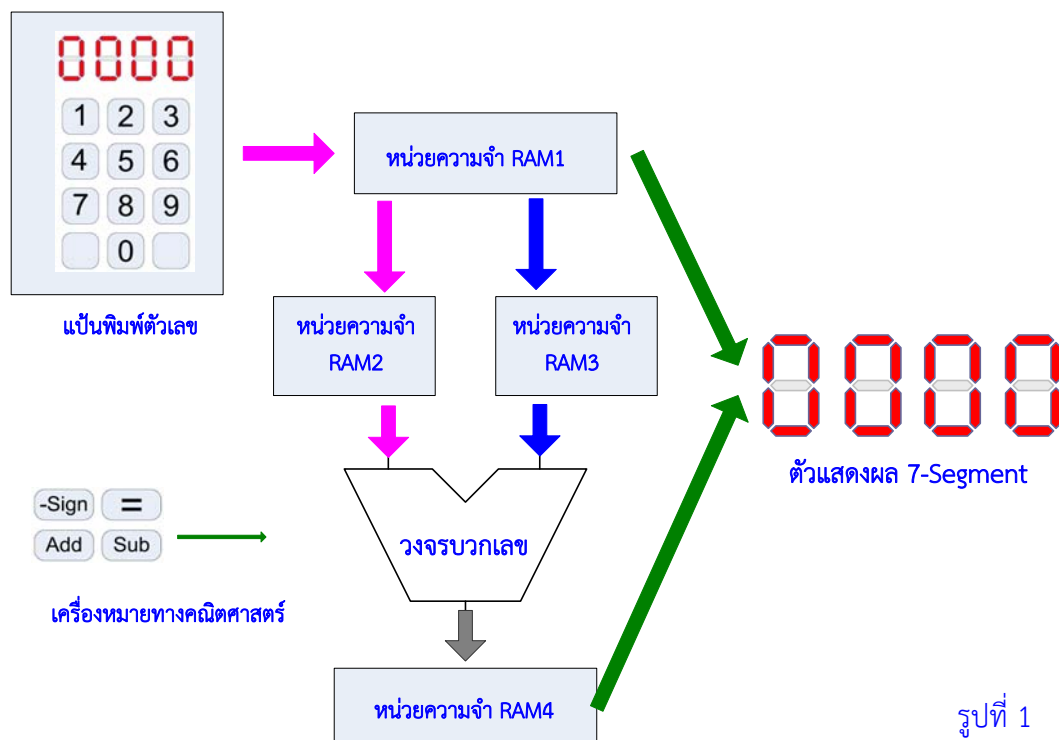
- แสดงค่าผลลัพธ์ในกรณีผลบวกเกินกำหนดด้วยสัญลักษณ์ **EEEE**



- สามารถกระทำทางคณิตศาสตร์ได้ในรูปแบบเครื่องหมาย + และ - เท่านั้น
- สามารถกระทำทางคณิตศาสตร์ได้ในรูปแบบเครื่องหมายเดี่ยวดังเช่น
$$\text{Ans} = A + B \quad \text{หรือ} \quad \text{Ans} = A - B$$
$$\text{Ans} = A + B + C + \dots \quad \text{หรือ} \quad \text{Ans} = A - B - C - \dots \quad \text{หรือ} \quad \text{Ans} = A - B + C - \dots$$
และไม่สามารถใช้กับเครื่องหมายในรูปแบบผสม (compound) เช่น $+=$, $-=$, $++$, $--$, etc.

ค) โครงสร้างหรือระบบย่อย(วงจร)ต่างๆ

จากรายละเอียดในตัวอย่างในข้อ 2.1.1 เราจะลองมาดูกลไกต่างๆ หรือขั้นตอนที่จะเกิดขึ้นภายในเมื่อเรากดแป้นของเครื่องคิดเลข โดยในที่นี้ขอให้พิจารณาในแง่ของพฤติกรรม(behavior) และวัตถุ(Entity หรือ Object) ของระบบงานที่เกี่ยวข้องทั้งหมด (อาจจะต้องใช้จินตนาการพอสมควรเพื่อให้นึกเห็นภาพเหล่านี้ให้ได้)



เริ่มแรก เราลองสมมุติว่าเครื่องคิดเลขมีส่วนประกอบอยู่ 8 ส่วน (8 ระบบ) มีความสัมพันธ์กันดังในรูปที่ 1 โดยแต่ละส่วนมีหน้าที่ดังนี้

1. **แป้นพิมพ์ตัวเลข** สร้างตัวเลขขึ้นมาครั้งละ 1 ตัว ต่อการกดปุ่มหนึ่งครั้ง

2. **หน่วยความจำ RAM1** เป็นที่สำหรับเก็บค่าตัวเลขชั่วคราว เพื่อรอผสมกันให้เป็นเลขได้หลายหลัก

เช่น เมื่อกด **9** ค่าใน RAM1 จะเป็น 0009 ตามด้วยปุ่ม **8** ค่าใน RAM1 จะต้องเปลี่ยนโดย เลื่อนเลข 9 ไปด้านซ้าย 1 หลัก(หลักสิบ)ก่อนจากนั้นจึงเอาเลข 8 ไปวางต่อไว้ที่หลักหน่วย จะทำให้ค่าใน RAM1 เป็น 0098 เป็นต้น ค่านี้จะถูกย้ายไปไว้ใน RAM 2 หรือ RAM 3 แล้วแต่กรณีว่าเป็นตัวตั้งหรือตัวบวก เมื่อปุ่ม **Add** ถูกกด



จากรูปที่ 2 เราจะพบว่ามีส่วนประกอบเพิ่มขึ้นมาอีกหลายชิ้น เช่น

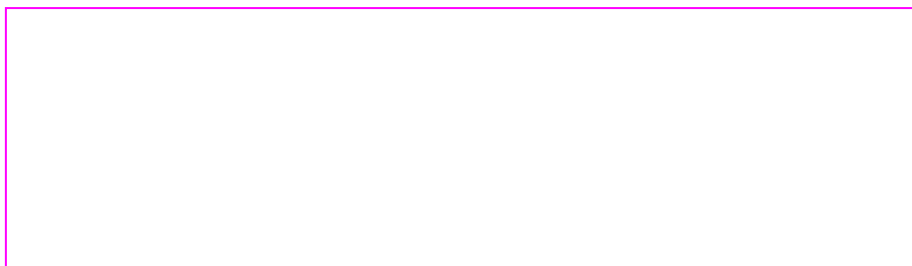
มัลติเพล็กซ์เซอร์ **ดีมัลติเพล็กซ์เซอร์** ใช้สำหรับการปรับเปลี่ยนเส้นทางวิ่งของข้อมูลเพื่อให้มีลำดับขั้นตอนตามที่เราต้องการ

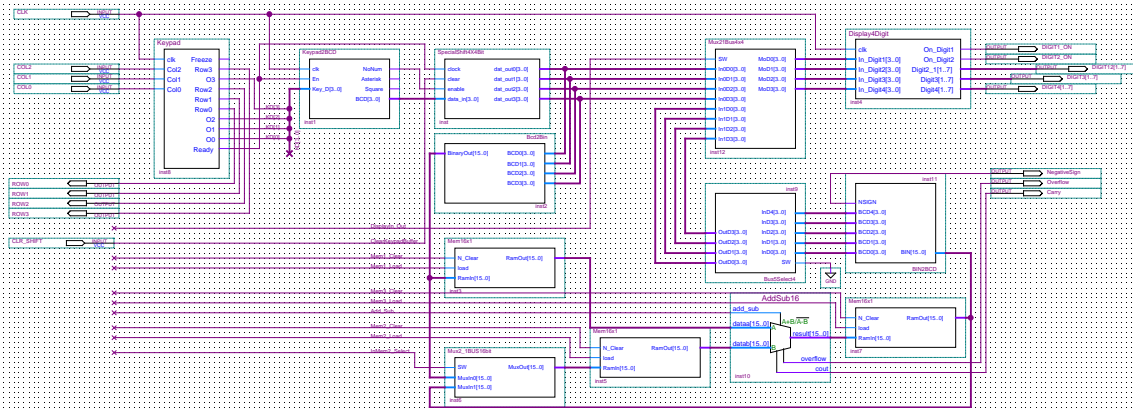
วงจรแปลงระบบเลขฐาน มีความจำเป็นในการเปลี่ยนค่าตัวเลข (ที่เคยมองเป็นแบบอักษรในการทดลองที่ 10) ให้เป็นค่าของเลขจำนวนเต็มในระบบที่เราออกแบบไว้ อาจจะเป็นฐานสองหรือ BCD ก็ได้

หน่วยควบคุม (Control Unit) วงจรนี้เป็นหัวใจสำคัญของระบบ ที่จะต้องสร้างสัญญาณต่างๆขึ้นมาควบคุมการทำงานของอุปกรณ์ที่มีในระบบทั้งหมดให้ทำงานก่อน/หลังเป็นลำดับตามที่ต้องการ สัญญาณควบคุมในที่นี้จะเขียนด้วยรูปลูกศรเล็กๆ สีดำ เขียนเพียงเส้นทางให้เห็นว่าอุปกรณ์ใดเป็นผู้ควบคุม ส่วนลักษณะการควบคุมนั้น ขอในจินตนาการว่าเป็นทุกอย่างที่จำเป็นต่อการทำงานก็แล้วกัน (ในความเป็นจริงแล้วอาจจะมีสัญญาณหลายเส้น เช่น clear load clk ... เป็นต้น)

สิ่งที่สำคัญที่สุดคือผู้ออกแบบจะต้องเข้าใจลำดับงานที่เกิดขึ้นภายในระบบเป็นอย่างดี จึงจะออกแบบสร้างหน่วยควบคุมได้อย่างถูกต้องมีประสิทธิภาพ (อันที่จริง ตัวมัลติเพล็กซ์เซอร์ และ ดีมัลติเพล็กซ์เซอร์ ในรูปที่ 2 สามารถที่จะลดจำนวนลงได้อีกแต่จะทำนักศึกษาที่เพิ่งเริ่มเรียนมองภาพไม่ออก ซึ่งผิดวัตถุประสงค์ ของการที่จะให้ work sheet นี้เป็นแนวทางสำหรับผู้ที่จะเริ่มต้นออกแบบทางดิจิทัล ให้มีพื้นฐานที่ดีพอที่จะนำไปต่อยอดในขั้นสูงขึ้นได้)

ตัวอย่างของ My Calculator ที่ได้ออกแบบในส่วนที่เป็น Datapath ได้นำมาแสดงไว้ในท้ายเอกสารนี้แล้ว ยังเหลือในส่วนที่จะเป็น Control Unit นักศึกษาจะต้องทดลองออกแบบเอง และหากติดขัดประการใดก็ขอให้ปรึกษาอาจารย์ผู้สอนจะช่วยให้ประสบความสำเร็จได้ดีขึ้น





หน้าที่และการทำงานของแต่ละ Block

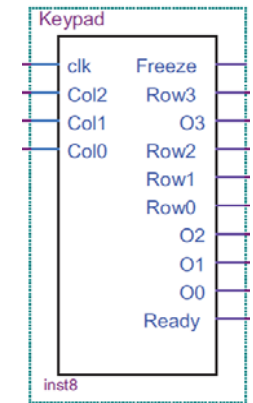
Keypad : เป็นวงจรชุดรับแป้นคีย์บอร์ด

ในที่นี้ใช้วงจรในการทดลองที่ 10 มาดัดแปลงให้มีขา Ready เพิ่มขึ้นมา (สัญญาณที่ขา Ready มีรูปร่างเหมือนกับสัญญาณที่ Freeze แต่จะเกิดช้ากว่าเล็กน้อย ทั้งนี้เพื่อให้มั่นใจได้ว่าสัญญาณ KD[3..0] นั้นมีค่าถูกต้องหรือนิ่งก่อนที่จะให้ Ready ออกไปบอกวงจรที่อยู่ถัดไปว่าข้อมูลพร้อมแล้ว) ที่ต้องทำเช่นนี้ก็เพื่อป้องกันการเกิดเวลาหน่วง (delay) ของข้อมูล (ทบทวนความสำคัญเกี่ยวกับเวลาหน่วงได้ในการทดลองที่ 1 และ 2)

Block Name : Keypad

Input port : clk , Col2, Col1, Col0

Output port : Row3, Row2, Row1, Row0, O3, O2, O1, O0, Freeze, Ready



Keypad2BCD : เป็นวงจรตรวจสอบชนิดของปุ่มที่ถูกกด

เนื่องจากข้อมูลที่มาจากบล็อก Keypad ยังไม่ได้ค่าของเลข (value) ตรงตามหมายเลขที่อยู่บนปุ่มกด วงจรนี้จึงมีหน้าที่ทำให้ค่าของปุ่มกดมีค่าตรงตามที่ต้องการ และยังมีการตรวจสอบด้วยว่า มีการกดปุ่มตัวเลข (Number) หรือมีการกดปุ่มอักขรพิเศษที่ไม่ใช่ตัวเลข (Asterisk * , Square #) แล้วแจ้งสัญญาณแสดงด้วย เมื่อเราอยากจะใช้สองปุ่มนี้ในอนาคต

ในที่นี้จะให้ค่าทางออกที่พอร์ท BCD[3..0] เป็นระบบ BCD ที่เขียนแทนด้วยเลขไบนารี 4 บิต เพื่อไม่ให้สับสนในที่นี้ขอ

เรียกชื่อเป็นอักษร (Character) 0, 1, 2, ... , 8, 9 ที่สอดคล้องกับแป้นปุ่มกดแทนที่จะเรียกว่าตัวเลข (Number)

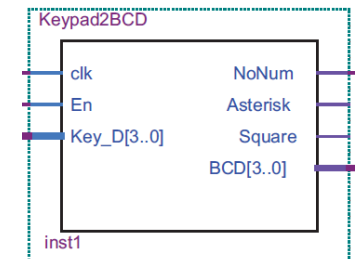
ความสัมพันธ์ของแป้นปุ่มกดกับค่าของสัญญาณ BCD[3..0] , NoNum , Asterisk , Square เป็นดังตาราง

เมื่อกดปุ่ม	NoNum	Asterisk	Square	BCD[3..0]
0	1	0	0	"0000"
1	1	0	0	"0001"
2	1	0	0	"0010"
3	1	0	0	"0011"
4	1	0	0	"0100"
5	1	0	0	"0101"
6	1	0	0	"0110"
7	1	0	0	"0111"
8	1	0	0	"1000"
9	1	0	0	"1001"
#	0	0	1	"1110"
*	0	1	0	"1111"

Block Name : Keypad2BCD (อ่านว่า Keypad to BCD Converter)

Input port : clk , En , Key_D[3..0]

Output port : BCD[3..0] , NoNum , Asterisk , Square



หมายเหตุ ผู้ออกแบบสามารถกำหนดค่าที่แตกต่างจากในตารางนี้ได้

SpecialShift4X4Bit : วงจรเลื่อนตัวอักษร BCD ไปทางซ้าย

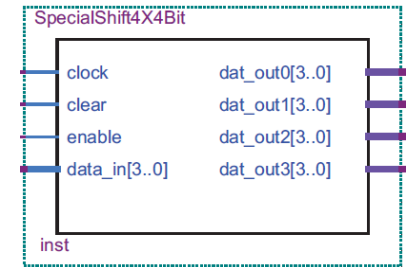
เป็นวงจรที่ทำหน้าที่นำเอาตัวอักษร ขนาด 4 บิต (BCD) ที่รับมาจากขา Data_in[3..0] เข้ามาครั้งละ 1 หลักเพื่อจัดเรียงให้เป็นอักษรที่มีหลายหลักตามลำดับ การเลื่อนค่าควรจะเลื่อนได้อย่างไม่จำกัดขึ้นอยู่กับการกดปุ่ม แต่จะเก็บค่าเพียงสี่หลักสุดท้ายล่าสุดเท่านั้น

- ในที่นี้ใช้ขาสัญญาณ enable เป็นตัวสั่งให้เกิดเลื่อนครั้งละหนึ่งหลักหลักตามการกดแป้นพิมพ์
- ใช้ขาสัญญาณ clear เมื่อต้องการเคลียร์ให้ทุกหลักมีค่าเป็นศูนย์
- สัญญาณทางออกเป็นอักษร 4 บิต (BCD) จำนวน 4 หลักยังไม่แปลงค่า (value) เป็นตัวเลขดังเช่น ถ้าเรากดแป้นเลข 1 ตามด้วย 2 ตามด้วย 3 และ 4 ค่าที่ไปปรากฏที่พอร์ต dat_out3[3..0] , ... , dat_out0[3..0] จะมีความหมายเป็น หนึ่งสองสามสี่ (อ่านเหมือนหมายเลขโทรศัพท์) ไม่ใช่หนึ่งพันสองร้อยสามสิบสี่

Block Name : SpecialShift4X4Bit

Input port : clock , enable , clear , data_in[3..0]

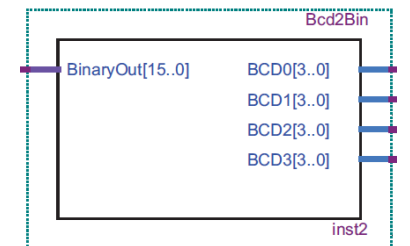
Output port : dat_out3[3..0] , dat_out2[3..0] , dat_out1[3..0] , dat_out0[3..0]



Bcd2bin : วงจรแปลงอักษร 4 บิต (BCD) จำนวน 4 หลักให้เป็นค่าตัวเลข (value)

เป็นวงจรที่ใช้แปลงอักษร 4 บิต (BCD) จำนวน 4 หลักให้เป็นค่าตัวเลขฐานสิบขนาด 4 หลักและเก็บผลที่แปลงได้อยู่ในระบบเลขไบนารี 16 บิต (ในความเป็นจริงไม่จำเป็นต้องมีขนาดถึง 16 บิตก็ได้ แต่ในขั้นนี้เราได้ออกแบบเผื่อไว้เท่านั้น) ส่วนหลักการแปลง ขอยกเป็นตัวอย่างพอมองให้เห็นภาพคร่าวๆ ดังนี้

ดังเช่น ตัวอักษร 1 2 3 4 (หนึ่งสองสามสี่) ที่ได้มาจากวงจร SpecialShift4X4Bit จะถูกเปลี่ยนเป็น 1234 (หนึ่งพันสองร้อยสามสิบสี่) และเก็บค่าไว้เป็นแบบไบนารีสิบหกบิตคือ 0000010011010010₂



ข้อสังเกต วงจรนี้ยังไม่ได้ออกแบบให้มีการแปลงในกรณีที่ต้องการค่าตัวเลขเป็นลบ ซึ่งจะต้องนำเครื่องหมายลบ (Negative Sign) มาพิจารณาด้วย และระบบเลขที่แปลงควรจะเป็นเลขแบบ signed number หรือ 2's complement แล้วแต่ว่าผู้ออกแบบจะชำนาญแบบใด

Block Name : Bcd2bin (อ่านว่า BCD to Binary Converter)

Input port : BCD3[3..0] , BCD2[3..0] , BCD1[3..0] BCD0[3..0]

Output port : BinaryOut[15..0]

คำแนะนำ ให้ น.ศ. ออกแบบเผื่อด้วยในกรณีที่ตัวเลขที่ป้อนนั้นมีค่าเป็นลบ

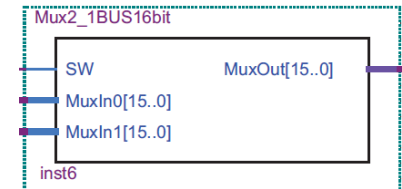
Mux2_1BUS16bit : วงจรมัลติเพล็กซ์เซอร์ แบบเข้า 2 อินพุต (16 บิต) ออก 1 เอาท์พุท (16 บิต)

เป็นวงจรสำหรับเลือกสัญญาณจากบัส 16 บิต ระหว่าง MuxIn0 และ MuxIn1 ให้ออกไปที่บัส MuxOut

Block Name : Mux2_1BUS16bit (อ่านว่า 16bit Bus Multiplexer 2 to 1)

Input port : MuxIn0[15..0] , MuxIn1[15..0] , SW

Output port : MuxOut[15..0]



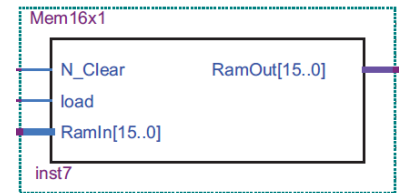
Mem16x1 : วงจรที่ทำหน้าที่เก็บข้อมูลขนาด 16 บิต

เป็นวงจรที่ทำหน้าที่เสมือนหน่วยความจำ ที่มีการควบคุมการนำเอาข้อมูลเข้าไปเก็บ (load) และสามารถเคลียร์ข้อมูลให้มีค่าเป็นศูนย์ได้ด้วยขาควบคุมชื่อ N_clear ซึ่งทำงานแบบ active low (เอาท์พุทให้ค่าเป็น 0 ทันทีเมื่อขานี้มีค่าลอจิกเป็น 0)

Block Name : Mem16x1 (อ่านว่า 16x1bit Memory unit)

Input port : RamIn[15..0] , N_clear , load

Output port : RamOut[15..0]



Mux21BUS4x4 : วงจรมัลติเพล็กซ์เซอร์ แบบเข้า 2 อินพุต (4 บิต) ออก 1 เอาท์พุท (4 บิต)

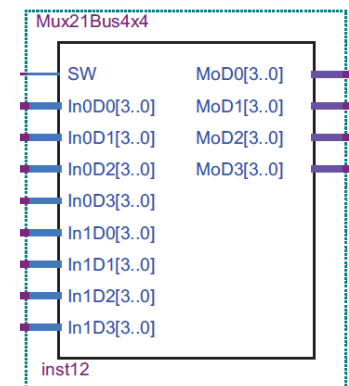
เป็นวงจรสำหรับเลือกสัญญาณจากบัส 4 บิต ระหว่าง In0Dx[3..0] และ In1Dx[3..0] ให้ออกไปที่บัส MoDx[3..0] จำนวน 4 ชุด รวมอยู่ในบล็อกเดียวกัน

Block Name : Mux21BUS4x4 อ่านว่า (4bit Bus Multiplexer 2 to 1) 4 ชุด

Input port : In0Dx[3..0] , In1Dx[3..0] , SW

Output port : MoDx[3..0]

เมื่อ $x = 0, 1, 2, 3$



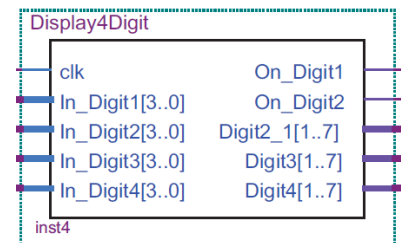
Display4Digit : วงจรแสดงผลตัวเลขแบบ 7-Segment จำนวน 4 หลัก (4 digit)

เป็นวงจรสำหรับนำค่าตัวเลขแบบ BCD (4บิต) ไปแสดงผลให้เป็นตัวเลข 0 ถึง 9 แบบ 7-Segment ซึ่งเป็นวงจรเดียวกันกับที่ใช้ในการทดลองที่ 7 และ 8

Block Name : Display4Digit

Input port : InDigit1[3..0] , InDigit2[3..0] , InDigit3[3..0] , InDigit4[3..0] , clk

Output port : Digit4[1..7] , Digit3[1..7] , Digit2_1[1..7] , On_digit2 , On_digit1



BIN2BCD : วงจรสำหรับแปลงค่าตัวเลขแบบ Binary ให้เป็นเลขแบบ BCD

เป็นวงจรสำหรับแปลงค่าตัวเลขแบบไบนารีขนาด 16 บิต ให้เป็นเลขแบบฐานสิบหรือ BCD จำนวน 5 หลักเพื่อนำไปแสดงผลที่ชุดแสดงผลแบบ 7-Segment ส่วนสัญญาณ NSIGN เป็นขาที่ใช้บ่งบอกว่าผลลัพธ์ BCD เป็นจำนวนบวก หรือ ลบ

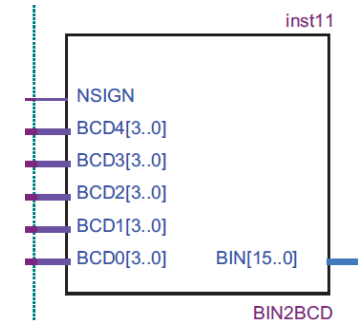
ข้อสังเกต ในวงจรนี้ได้ออกแบบให้เป็นการแปลงระบบเลขในแบบ Signed number ซึ่งจะทำให้ครอบคลุมค่าทั้งที่เป็นบวกและลบ

Block Name : BIN2BCD (อ่านว่า Binart to BCD Code Converter)

Input port : BIN[15..0]

Output port : BCDx[3..0] , NSIGN

เมื่อ $X = 0, 1, 2, 3, 4$



Bus5Select4 : วงจรมัลติเพล็กซ์สำหรับเลือกเอาเลขที่ต้องการเพียง 4 หลักจาก 5 หลัก

เนื่องจากในเครื่องคิดเลขที่ออกแบบนี้มีโอกาสที่จะได้ผลลัพธ์ของการคำนวณเกิดเป็นตัวเลขแบบ 5 หลัก ดังเช่น ถ้าบวกเลข ดังนี้ $7899 + 9899 = 17798$ ซึ่งเป็นการบวกเลขขนาด 4 หลักสองจำนวน จะทำให้ได้คำตอบเป็นเลข 5 หลัก แต่บนบอร์ดของเรา

สามารถแสดงได้เพียงสี่หลักเท่านั้น จึงได้มีการออกแบบวงจรนี้ให้สามารถเลือกได้ว่าจะแสดงเลขชุดใดก็ได้โดยเลือกที่ค่าของ SW

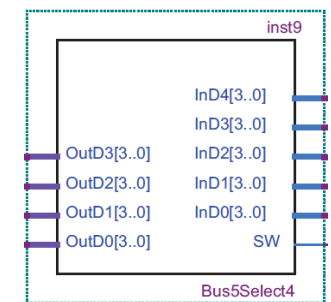
ดังเช่น 17798 ถ้าแสดงสี่หลักด้านขวาจะได้เป็น 7798 (เลข 1 จะถูกบังไว้ไม่สามารถมองเห็นได้)

ถ้าแสดงสี่หลักด้านซ้ายจะได้เป็น 1779 (เลข 8 จะถูกบังไว้ไม่สามารถมองเห็นได้)

Block Name : Bus5Select4

Input port : InD4[3..0] , InD3[3..0] , InD2[3..0] , InD1[3..0] , InD0[3..0] , SW

Output port : OutD3[3..0] , OutD2[3..0] , OutD1[3..0] , OutD0[3..0]



หมายเหตุ ในกรณีที่เรากำหนดไว้ว่าถ้าผลการคำนวณมีค่าเกิน 4 หลักแล้วให้แสดงเป็นสัญลักษณ์พิเศษ เช่น EEEE ดังที่ได้กล่าวไว้ในตอนแรกแล้ว วงจรนี้ไม่จำเป็นต้องมีก็ได้

ControlUnit : เป็นระบบงานหรือวงจรที่คอยรับข้อมูลจากปุ่มกดต่างๆ รวมทั้งแป้นตัวเลข เพื่อนำมาสั่งการทำงานของอุปกรณ์หรือวงจรที่กล่าวมาข้างต้นให้มีลำดับการทำงานตามที่เราต้องการในแต่ละเครื่องหมายของ Operator

หมายเหตุ

1. กลไกการทำงานของ **ControlUnit** ไม่นำมาแสดงไว้ในเอกสารชุดนี้
2. ตัวอย่างของวงจร/ระบบงาน ของการแปลงรหัส BCD ให้เป็น Binary (แบบ Unsigned) ชื่อ Bcd2bin ได้นำมาแสดงไว้ด้านท้ายของเอกสารนี้ด้วย

```

1  -- File Mame/ชื่อไฟล์      BCD2BIN.VHD
2  -- Project/โครงการ        Simple Calculator
3  -- Version/รุ่น            Application08
4  -- Date/วันที่             October, 26 ,2014
5  -- Function/หน้าที่        ใช้สำหรับทำการแปลง ข้อมูลทางเข้าที่มีลักษณะเป็น ระบบ BCD (ขนาด 4 หลัก) ให้เป็นเลขไบนารี16บิต
6  -- Input/ทางเข้า           สัญญาณลอจิกชุดละ 4 บิต จำนวน 4 ชุด
7  -- Input Format/รูปแบบทางเข้า
8  --                        { "0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111",
9  --                        "1000", "1001" }
10 -- Output/ทางออก           สัญญาณลอจิก 16 บิต
11 -- Output Format/รูปแบบทางออกเข้า
12 --                        { "bbbbbbbbbbbbbbbb" }
13 --                        เป็นเลขไบนารีแบบไม่ติดเครื่องหมาย (Unsigned Binary Number)
14 --
15
16 LIBRARY ieee ;
17 USE ieee.std_logic_1164.all ;
18 USE ieee.std_logic_unsigned.all ;
19 USE ieee.std_logic_arith.all ;
20
21
22 ENTITY Bcd2Bin IS
23     PORT ( BCD0,BCD1,BCD2,BCD3      : IN STD_LOGIC_VECTOR(3 DOWNT0 0) ;
24           BinaryOut                  : OUT INTEGER Range 0 to 65535 ) ;
25 END Bcd2Bin ;
26
27
28
29 ARCHITECTURE Behavior OF Bcd2Bin IS
30     SIGNAL T0,T1,T2,T3      : INTEGER Range 0 to 65535 ;
31 BEGIN
32     PROCESS(BCD0)            -- Digit0 Conversion
33 BEGIN
34         CASE BCD0 IS
35             WHEN "0000" => T0 <= 0;
36             WHEN "0001" => T0 <= 1;
37             WHEN "0010" => T0 <= 2;
38             WHEN "0011" => T0 <= 3;
39             WHEN "0100" => T0 <= 4;
40             WHEN "0101" => T0 <= 5;
41             WHEN "0110" => T0 <= 6;
42             WHEN "0111" => T0 <= 7;
43             WHEN "1000" => T0 <= 8;
44             WHEN "1001" => T0 <= 9;
45             WHEN OTHERS => T0 <= 0;
46         END CASE ;
47     END Process;
48
49
50     PROCESS(BCD1)            -- Digit1 Conversion
51 BEGIN
52         CASE BCD1 IS
53             WHEN "0000" => T1 <= 0;
54             WHEN "0001" => T1 <= 10;
55             WHEN "0010" => T1 <= 20;
56             WHEN "0011" => T1 <= 30;
57             WHEN "0100" => T1 <= 40;
58             WHEN "0101" => T1 <= 50;
59             WHEN "0110" => T1 <= 60;
60             WHEN "0111" => T1 <= 70;
61             WHEN "1000" => T1 <= 80;
62             WHEN "1001" => T1 <= 90;
63             WHEN OTHERS => T1 <= 0;
64         END CASE ;
65     END Process;
66

```

```

67
68     PROCESS(BCD2)                                -- Digit2 Conversion
69     BEGIN
70         CASE BCD2 IS
71             WHEN "0000" => T2 <= 0;
72             WHEN "0001" => T2 <= 100;
73             WHEN "0010" => T2 <= 200;
74             WHEN "0011" => T2 <= 300;
75             WHEN "0100" => T2 <= 400;
76             WHEN "0101" => T2 <= 500;
77             WHEN "0110" => T2 <= 600;
78             WHEN "0111" => T2 <= 700;
79             WHEN "1000" => T2 <= 800;
80             WHEN "1001" => T2 <= 900;
81             WHEN OTHERS => T2 <= 0;
82         END CASE ;
83     END Process;
84
85
86     PROCESS(BCD3)                                -- Digit3 Conversion
87     BEGIN
88         CASE BCD3 IS
89             WHEN "0000" => T3 <= 0;
90             WHEN "0001" => T3 <= 1000;
91             WHEN "0010" => T3 <= 2000;
92             WHEN "0011" => T3 <= 3000;
93             WHEN "0100" => T3 <= 4000;
94             WHEN "0101" => T3 <= 5000;
95             WHEN "0110" => T3 <= 6000;
96             WHEN "0111" => T3 <= 7000;
97             WHEN "1000" => T3 <= 8000;
98             WHEN "1001" => T3 <= 9000;
99             WHEN OTHERS => T3 <= 0;
100        END CASE ;
101    END Process;
102        BinaryOut <= T3+T2+T1+T0;
103    END Behavior ;
104
105
106
107
108 -----
109 --      คำเตือน การออกแบบโค้ดชุดนี้ยังไม่ได้ครอบคลุมการแปลงตัวเลขที่มีค่าเป็นลบ      --
110 -----

```