



ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

ภาคการศึกษาที่.....ปีการศึกษา.....

รหัสวิชา 010113026 ชื่อวิชา Digital Laboratory

ตอนเรียน ..... หมายเลขโต๊ะ.....

รหัสนักศึกษา..... ชื่อ-นามสกุล.....

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

## การทดลองที่ 9

### State Machines and it's Applications

#### วัตถุประสงค์

1. เพื่อให้สามารถใช้โปรแกรมคอมพิวเตอร์เพื่อจำลองการทำงานของวงจรลอจิกเกตได้
2. เพื่อให้สามารถประยุกต์ใช้วงจรและอุปกรณ์ดิจิทัลเพื่อออกแบบระบบงานที่ซับซ้อนได้
3. เพื่อให้สามารถประยุกต์ใช้การออกแบบระบบงานดิจิทัลด้วยเทคนิคทาง state diagram ได้

#### อุปกรณ์

1. ระบบคอมพิวเตอร์ 1 เครื่อง พร้อมติดตั้งโปรแกรม Quartus II เวอร์ชัน 8.0 (Student Edition) ขึ้นไป
2. บอร์ดทดลอง Cyclone3-Lab01 1 บอร์ด
3. สาย J-TAG 1 เส้น ใช้รุ่น USB-Blaster (สำหรับเครื่อง Notebook) หรือรุ่น Byte-Blaster (สำหรับเครื่อง PC)
4. บอร์ดแสดงผล 7-segment (รุ่นแป้นพิมพ์ Keypad สีขาว)

#### การทดลอง

##### A1 บทนำ: แนวทางการออกแบบงาน

ในการทดลองนี้จะสร้างระบบงานชื่อ “**เครื่องจำหน่ายเครื่องดื่มแบบหยอดเหรียญ Vending Machine**” ซึ่งจะเป็นตัวอย่างการนำเทคนิคการออกแบบ state diagram มาช่วยในการออกแบบ ดังนั้นจึงมีความจำเป็นที่จะต้องทำความเข้าใจการทำงานของเครื่อง Vending Machine ให้เข้าใจพฤติกรรมของมันอย่างถูกต้องก่อน แล้วค่อยเริ่มต้นลงมือทำการออกแบบ หากทำตามขั้นตอนนี้จะช่วยให้ได้เห็นประโยชน์ที่ชัดเจนของแนวทางการออกแบบด้วยเทคนิค state diagram ได้เป็นอย่างดี

#### การทำงานของเครื่อง Vending Machine:

##### การทำงานของเครื่อง Vending Machine

- 1 **มีเฉพาะเครื่องดื่มแบบกระป๋อง** ราคากระป๋องละ 15 บาท เมื่อเหรียญครบตามราคาจะปล่อยสินค้าให้ทันที
- 2 **เหรียญที่รับ** เฉพาะเหรียญ 5 บาท (Nikel ใช้ตัวย่อ **N**) และเหรียญ 10 บาท (Dime ใช้ตัวย่อ **D**)
- 3 **ไม่ทอนเงิน** หากหยอดเหรียญเกินราคาที่ขาย เช่น 10 บาท 2 เหรียญ จะปล่อยสินค้าให้ 1 กระป๋อง และรอรับเงินเพิ่มอีก 10 บาทเพื่อให้ครบ 15 บาท สำหรับการขายเพิ่มอีก 1 กระป๋อง
- 4 **มีตัวเลขแสดงจำนวนเงิน** ที่ใส่เข้าไปหากเงินยังไม่ถึง 15 บาท แต่ถ้าเกิน 15 บาทจะแสดงเงินที่เหลืออยู่หลังจากหักราคาขายสินค้าไปแล้ว



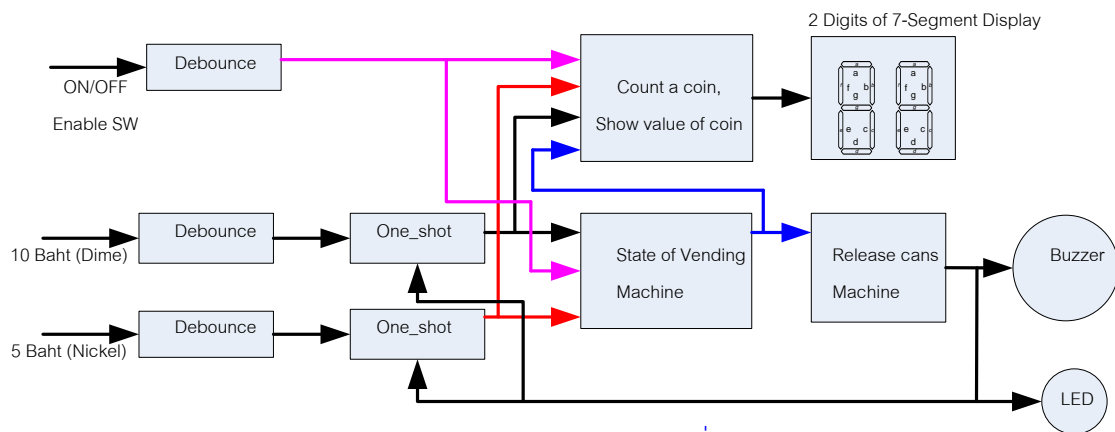
5 มีการป้องกันการกินเหรียญ ในช่วงที่ปล่อยกระป๋องน้ำเครื่องจะ**ไม่รับเหรียญเพิ่ม**จนกว่าจะสิ้นสุดขั้นตอนการปล่อยกระป๋อง (จะมีไฟสัญญาณกระป๋องเตือน พร้อมส่งเสียงบีบราวๆ 2 วินาที)

## A2 เตรียมการออกแบบ: โครงสร้างของเครื่อง Vending Machine

เราจำเป็นต้องจินตนาการกลไกของบอร์ดทดลองเพื่อให้ทำหน้าที่เป็นเครื่อง Vending Machine ดังนี้

- ใช้ SW6 (สวิตช์เลื่อน) ทำหน้าที่เป็นสวิตช์**ปิดเปิดเครื่อง** (ควบคุมที่ขาสัญญาณ Enable)
- ใช้ PB2 (สวิตช์กดติดปล่อยดับ) ทำหน้าที่แทนการ**หยอดเหรียญ 5 บาท** 1 เหรียญต่อการกดหนึ่งครั้ง
- ใช้ PB3 (สวิตช์กดติดปล่อยดับ) ทำหน้าที่แทนการ**หยอดเหรียญ 10 บาท** 1 เหรียญต่อการกดหนึ่งครั้ง
- ใช้ 7 Segment 2 หลัก แสดงจำนวนเงิน ซึ่ง**ควรจะมีตัวเลขเพียง 5, 10, 15, 20** บาทเท่านั้น
- ใช้บัซเซอร์ & LED0 ส่งเสียง & ส่ง**แสงไฟกระพริบ** แทนการปล่อยกระป๋องน้ำดื่ม

A3 แยกงานใหญ่ๆ ออกให้เป็นงานย่อยๆ: จากงานข้างต้นเมื่อนำมาแยกเป็นงานย่อยๆ จะได้โครงสร้างโดยรวมของระบบแสดงดังรูปที่ 1



รูปที่ 1

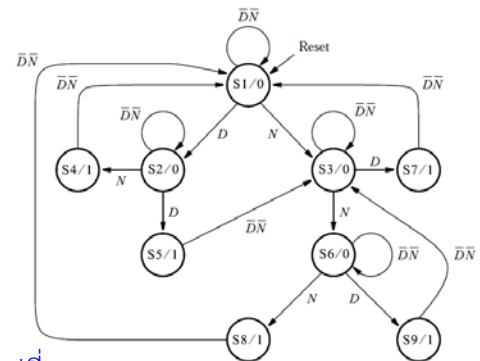
แต่ละระบบงานย่อยมีหน้าที่ทำอะไรบ้าง :

1. **Debounce** เนื่องจากเราใช้สวิตช์แทนการหยอดเหรียญ จำเป็นจะต้องป้องกันสวิตช์ทำงานผิดพลาด (เราได้ทำเตรียมไว้แล้วในการทดลองที่ 8)
2. **7-Segment** วงจรแสดงผลตัวเลข (เราได้ทำเตรียมไว้แล้วในการทดลองที่ 8 )
3. **Buzzer และ LED** เป็นอุปกรณ์ที่มีบนบอร์ดทดลอง ใช้ส่งเสียง และส่งแสง
4. **One\_Shot** เนื่องจากธรรมชาติของการหยอดเหรียญ แต่ละคนอาจจะหยอดซ้ำ เร็ว ต่างกัน และต้องหยอดทีละ 1 เหรียญ จำเป็นต้องสร้างวงจรตรวจจับพฤติกรรมนี้โดยปราศจากข้อจำกัดด้านเวลา  
การทำงานของ One\_Shot จะทำงานโดยคอยตรวจสอบว่าสวิตช์มีการถูกกดหรือไม่  
**ถ้าสวิตช์ถูกกด** มันจะปล่อยเอาต์พุตเปลี่ยนจาก '0' ไปเป็น '1' (เกิดขอบขาขึ้น**เพียง 1 ขอบ**)  
แล้วรอจนถึงหนึ่งคาบของสัญญาณนาฬิกา ก็จะเปลี่ยนกลับไปเป็น '0' เหมือนเดิม  
**โดยที่ไม่สนใจว่าสวิตช์จะถูกกดนานเท่าใดก่อนที่จะปล่อย (อย่างไรก็กด 1 ครั้งเท่านั้น)**



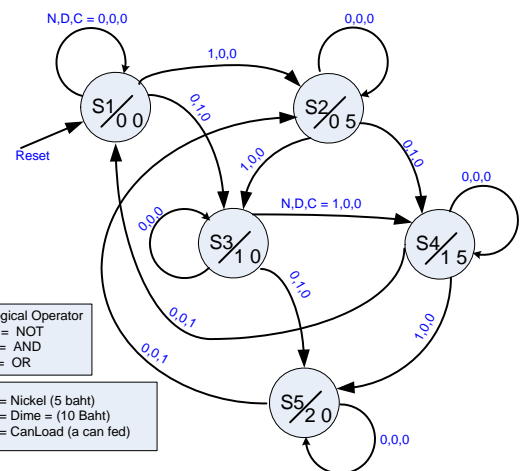
5. **State of Vending Machine** ระบบงาน (วงจร) ส่วนนี้จะทำงานตามขั้นตอนที่ได้อธิบายไว้ในหน้าแรก (หัวข้อ A1) เมื่อเขียนเป็นแผนผังลำดับขั้นตอน หรือ state diagram จะได้ดังรูปที่ 2

Example : ถ้าไม่มีการหยอดเหรียญ ระบบจะรออยู่ที่สเตต S1  
 ที่ S1 ถ้าใส่เหรียญ 10 ระบบจะไปที่ S2 แต่ถ้าใส่เหรียญ 5 ระบบจะไปที่ S3  
 ที่ S3 ถ้ามีการใส่เหรียญ 10 (จะได้ครบ 15) ระบบจะไปที่ S7 ปลอยกระป๋อง  
 และกระโดดต่อไปที่ S1 เมื่อปลอยกระป๋องเสร็จ  
 ที่ S3 ถ้ามีการใส่เหรียญ 5 (เงินยังไม่ครบ) ระบบจะไปที่ S6 รอเงินเพิ่มอีก ...



รูปที่ 2

6. **Count, Show value of Coins** ระบบงานนี้จะทำการตรวจนับเหรียญที่ถูกใส่เข้ามา ในทางปฏิบัติจะได้จำนวนเป็น 0, 5, 10, 15, 20 บาท (ไม่เกิน 20 บาทเพราะเครื่องจะจ่ายกระป๋องให้ทันที) ดังนั้นการเพิ่มหรือลดของจำนวนเงินจึงมีรูปแบบที่ตายตัว ทำให้เราสามารถใช่ state diagram มาทำงานแทนวงจร Adder ได้ ซึ่งจะง่ายและสะดวกกว่ามาก ดังรูปที่ 3



รูปที่ 3

7. **Release cans Machine** เป็นระบบงานปลอยกระป๋อง (ส่งเสียง & แสง) แต่ในชีวิตจริงเนื่องจากเป็นกลไกทางกล จะใช้เวลานานหลายวินาทีกว่าจะปลอยกระป๋องเสร็จ ทำให้ระบบต้องหยุดรับเหรียญเพิ่ม ระบบนี้จึงต้องส่งสัญญาณไปสั่งดับเหรียญ (ให้ One\_shot หยุดทำงาน) ไว้ชั่วคราวเพื่อป้องกันข้อผิดพลาดในการทำงานไม่สัมพันธ์กันของระบบรับเหรียญกับระบบปลอยกระป๋อง

## การทดลอง

ขั้นที่ 1 เตรียมนำอุปกรณ์ที่เคยสร้างไว้จากการทดลอง 7 – 8 มาใช้งาน

### คำสั่งการทดลอง

1. ให้สร้างโฟลเดอร์สำหรับเก็บงานชิ้นใหม่เพื่อเก็บงานในการทดลองนี้ชื่อ “Lab09\_State”
2. นำวงจรต่างๆที่เคยสร้างไว้ใน การทดลองที่ 5-8 (ทำการ copy ไฟล์ ดังรายชื่อด้านล่าง) มาไว้ในโฟลเดอร์ที่สร้างขึ้นใหม่นี้
  - ก) Debounce.qpf และ Debounce.bdf
  - ข) AsynchronousCounter.qpf และ AsynchronousCounter.bdf
  - ค) VHD\_7SEGM.qpf และ VHD\_7SEGM.vhd (ให้ใช้เวอร์ชันที่ปรับแก้ในการทดลองที่ 8)
  - ง) ClockDivider.qpf และ ClockDivider.bdf
3. ให้ทำการเปิดโปรเจกต์ไฟล์ตามข้อ 2 มาทำการคอมไพล์และสร้าง symbol file ใหม่ทั้งหมด
4. ให้ปิดโปรเจกต์ที่ดำเนินการในขั้นตอนที่ 2-3 ก่อนที่จะทำงานต่อไป



## ขั้นที่ 2 สร้างระบบ State of Vending Machine เพิ่มขึ้นมาใช้งาน

5. สร้างอุปกรณ์ **State\_Deve** (ชื่อย่อของระบบงาน State of Vending Machines) โดยดำเนินการดังนี้

5.1 สร้างโปรเจกต์ชื่อ “**State\_Deve**” ขึ้นมาและให้เก็บไว้ในโฟลเดอร์เดิม

ใช้ชิพ EP3C10E144C8

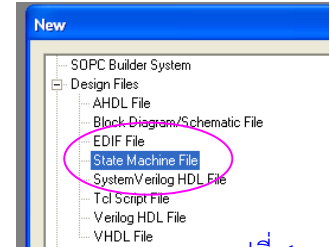
5.2 เปิดไฟล์สำหรับเก็บแผนภาพ **state diagram** โดยไปที่เมนู

**File >> New**

เลือก **State Machine file** ดังรูปที่ 4 จะปรากฏหน้าต่าง

Editor ว่างๆ ของไฟล์ชื่อ **SM1.smf** ขึ้นมาพร้อมแถบเครื่องมือ

มือด้านซ้ายของหน้าต่าง

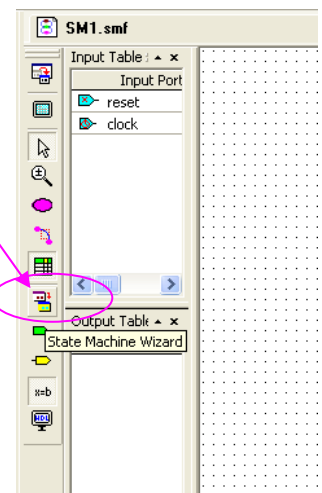


รูปที่ 4

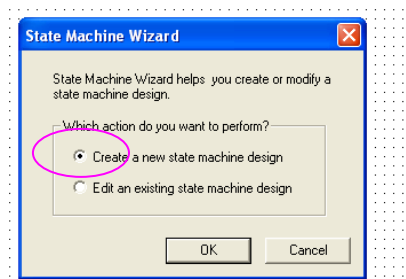
5.3 ให้เลือกใช้เครื่องมือเป็น **State Machine Wizard** ในรูปที่ 5

เราจะเริ่มด้วยการเขียน state table เป็นลำดับแรกแล้วจึงให้โปรแกรมทำการคอมไพล์เป็น state diagram ในภายหลัง

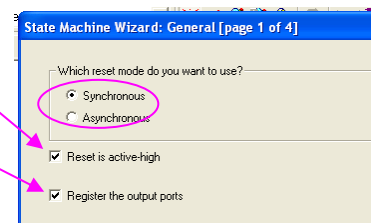
เมื่อเลือก State Machine Wizard แล้วจะปรากฏหน้าต่างขึ้นมาเพื่อให้ยืนยันการออกแบบ ดังรูปที่ 6a



รูปที่ 5



รูปที่ 6a



รูปที่ 6b

เมื่อเลือก **Create a ...** จะปรากฏหน้าต่าง [Page 1 of 4] สำหรับกำหนดคุณสมบัติทั่วไปดังรูปที่ 6b

- Synchronous กำหนดให้วงจรเปลี่ยน state โดยอาศัยสัญญาณนาฬิกา
- Reset in Active High กำหนดให้มีขาไว้สำหรับรีเซ็ต และทำการรีเซ็ตด้วยค่าลอจิก '1'
- Register the output port กำหนดให้สัญญาณเอาต์พุตออกจากตัวรีจิสเตอร์

กดปุ่ม **Next** เพื่อทำขั้นตอนต่อไป

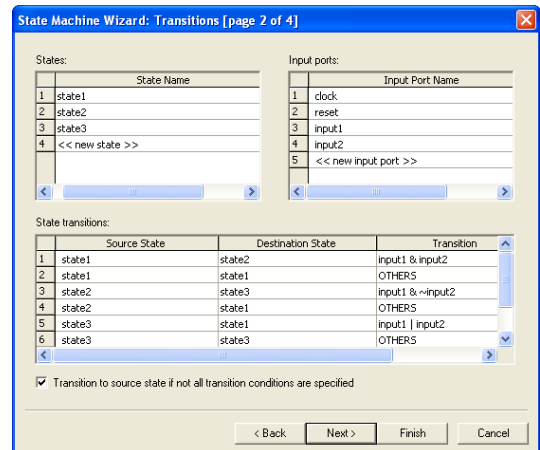
5.4 ที่หน้าต่าง [Page 2 of 4] จะเป็นการกำหนด ส่วนประกอบของ state ดังรูปที่ 7 ซึ่งประกอบด้วย

- ชื่อ State และจำนวนของ State
- สัญญาณอินพุตที่ใช้ ควบคุมการเปลี่ยน state
- ตารางแสดง state table และเงื่อนไขการเปลี่ยน state (State Transition)



หมายเหตุ เครื่องหมายของ Operator หรือตัวกระทำทางลอจิกพื้นฐานสำหรับใช้ในการเขียน state มีอยู่ 4 ตัวเท่านั้นคือ

~	NOT
&	AND
	OR
OTHERS	-----



5.5 ให้เขียน(ปรับแก้) ชื่อ ของ state ในตาราง [Page 2 of 4]

รูปที่ 7

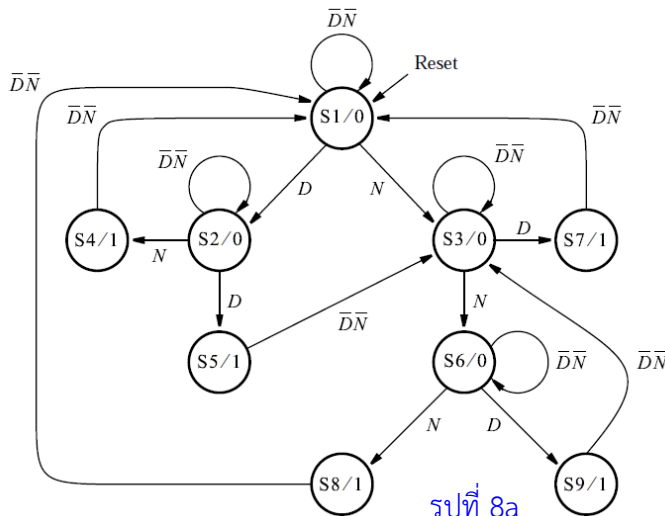
ให้มีจำนวน state = 9 state ชื่อ S1... S9 ตามลำดับดังรูปที่ 8b ด้วยการปรับแก้ตามรูปที่ 8c

โดยเปลี่ยนชื่อ(กดดับเบิลคลิก) จากเดิม **State 1** แก้เป็น **S1** (ตามตาราง state table ในรูปที่ 8b)

**State 2** แก้เป็น **S2**

...

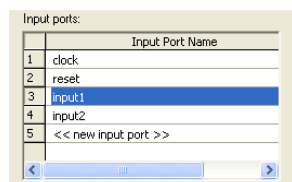
**State 9** แก้เป็น **S9**



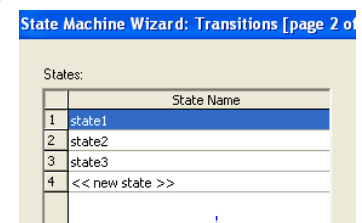
รูปที่ 8a

Present state	Next state				Output z
	DN = 00	01	10	11	
S1	S1	S3	S2	—	0
S2	S2	S4	S5	—	0
S3	S3	S6	S7	—	0
S4	S1	—	—	—	1
S5	S3	—	—	—	1
S6	S6	S8	S9	—	0
S7	S1	—	—	—	1
S8	S1	—	—	—	1
S9	S3	—	—	—	1

รูปที่ 8b



รูปที่ 8d



รูปที่ 8c

จากนั้นกำหนดอินพุตพอร์ทในรูปที่ 8d ที่จะทำหน้าที่ควบคุมการเปลี่ยนของ state โดยใช้

เปลี่ยนชื่อพอร์ทจาก **input1** (ของเดิม) เป็น **D** (ตามตาราง state table ของเรา)

เปลี่ยนชื่อพอร์ทจาก **input2** (ของเดิม) เป็น **N** (ตามตาราง state table ของเรา)



5.6 เปลี่ยนค่าในตาราง state transition ของรูปที่ 9 โดยเปลี่ยนไปใช้ข้อมูลจากรูปที่ 8b **ทั้งชื่อของ state** , **เงื่อนไขการเปลี่ยน state** (state transition) จนครบทุก state ก็กด Next เพื่อไปหน้าต่าง [Page 3 of 4]

State transitions:

	Source State	Destination State	Transition
1	S1	S2	input1 & input2
2	S1	S1	OTHERS
3	S2	S2	input1 & ~input2
4	S2	S3	OTHERS
5	S3	S5	input1   input2
6	S3	S6	OTHERS
7		S7	<< new transition >>

☒ Transition to source state if no other transition conditions are specified

รูปที่ 9

ข้อแนะนำในการเขียน State Transition ใน [Page 2 of 4]

ก) จากรูปที่ 8a จะมีบาง transition ที่มี exciting input เพียงตัวแปรเดียวเช่น N

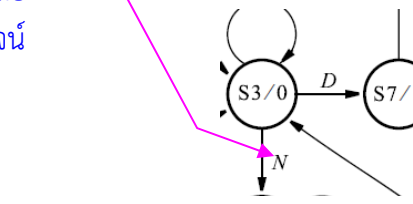
ในรูปนี้ความหมายคือ N = '1' และไม่มีการกล่าวถึง D เลย

แต่เราจะต้องเขียนให้ครบทั้งสองอินพุตโดยแสดงเป็นนิพจน์

ที่มีครบทั้งสองตัวแปรคือ D และ N โดยเขียนเป็น

$N \& \sim D$

Transition
N & ~D
OTHERS



การเขียนเช่นนี้จะหมายความว่า N = '1'

ตรงตามการกำหนดจาก state diagram ส่วน D ในที่นี้กำหนดเป็น ~D ซึ่งมีค่าเป็น D = '0' นั่นเอง

ข) ในบางกรณี ที่มี เงื่อนไขอื่นๆ ที่ไม่ใช่ตั้งในเงื่อนไขที่มีค่าเดียว เราสามารถใช้คำว่า OTHERS แทนการบอกว่าเป็นเงื่อนไขอื่นๆ ที่มีไม่น้อยกว่าหนึ่งเงื่อนไข มาเป็นเงื่อนไขทางเลือกก็ได้

6. กำหนดค่าของเอาต์พุต [Page 3 of 4] ดังรูปที่ 10

โดยชื่อของพอร์ตเอาต์พุตให้ใช้ชื่อเป็น Z

และกำหนดค่าของ Z ที่จะได้ที่ state ต่างๆ

(นำค่ามาจากรูปที่ 8b)

ปรับแก้ให้ถูกต้องตามตารางในรูปที่ 8b

เมื่อปรับแก้จนเสร็จแล้วก็จะปรากฏหน้าต่างสรุปรายละเอียดทั้งหมดที่ได้ทำไป [Page 4 of 4] ให้เราตรวจสอบอีกรอบ จากนั้นก็ดำเนินการต่อไป

Output ports:

	Output Port Name
1	Z
2	<< new output port >>

Action conditions:

	Output Port	Output Value	In State	Additional Conditions
4	Z	1	S4	<< condition >>
5	Z	1	S5	<< condition >>
6	Z	0	S6	<< condition >>
7	Z	1	S7	<< condition >>
8	Z	1	S8	<< condition >>
9	Z	1	S9	<< condition >>
10	<< output value >>			<< condition >>

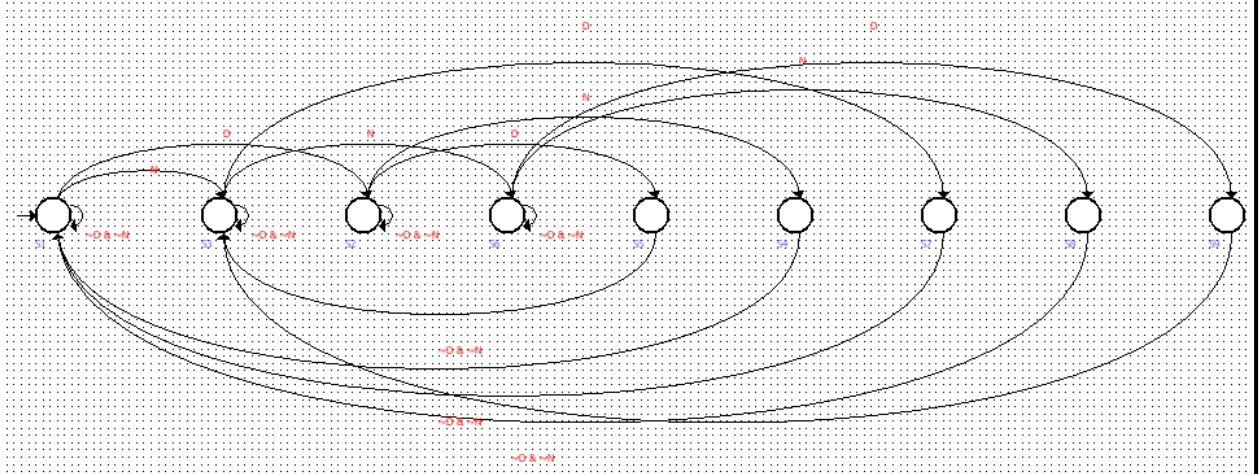
< Back Next > Finish Cancel

รูปที่ 10





7. เมื่อเสร็จครบทุกขั้นตอนแล้ว หากไม่มีขั้นตอนใดผิดพลาด โปรแกรมก็จะทำการสร้างรูปแผนภาพของ state Diagram ขึ้นมาให้ดังแสดงในรูปที่ 11 ทำการบันทึกไฟล์เพื่อเตรียมทำขั้นตอนต่อไป

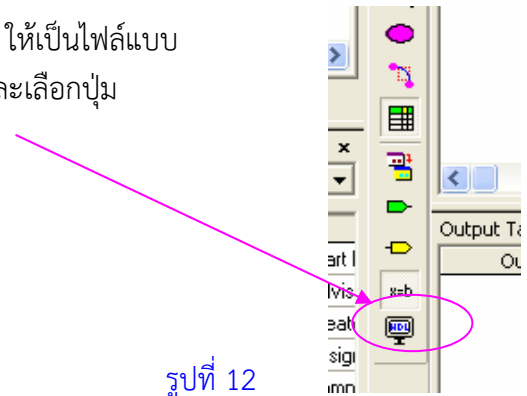


รูปที่ 11

หมายเหตุ: รูปที่ปรากฏอยู่นี้อาจจะไม่ตรงกับรูปที่ น.ศ. ทำในห้องปฏิบัติการ

8. ทำการแปลงไฟล์ของ state diagram ที่ได้จากในขั้นตอนที่ 7 ให้เป็นไฟล์แบบ ภาษา VHDL โดยไปที่แถบเครื่องมือด้านซ้ายของหน้าต่าง และเลือกปุ่ม Generate VHDL File ดังรูปที่ 12

เมื่อได้ไฟล์แบบ VHDL แล้วให้ทำการคอมไพล์ตามปกติ และทำการสร้าง symbol file ของวงจรขึ้นมา เพื่อเตรียมไว้ใช้ในขั้นตอนถัดไป

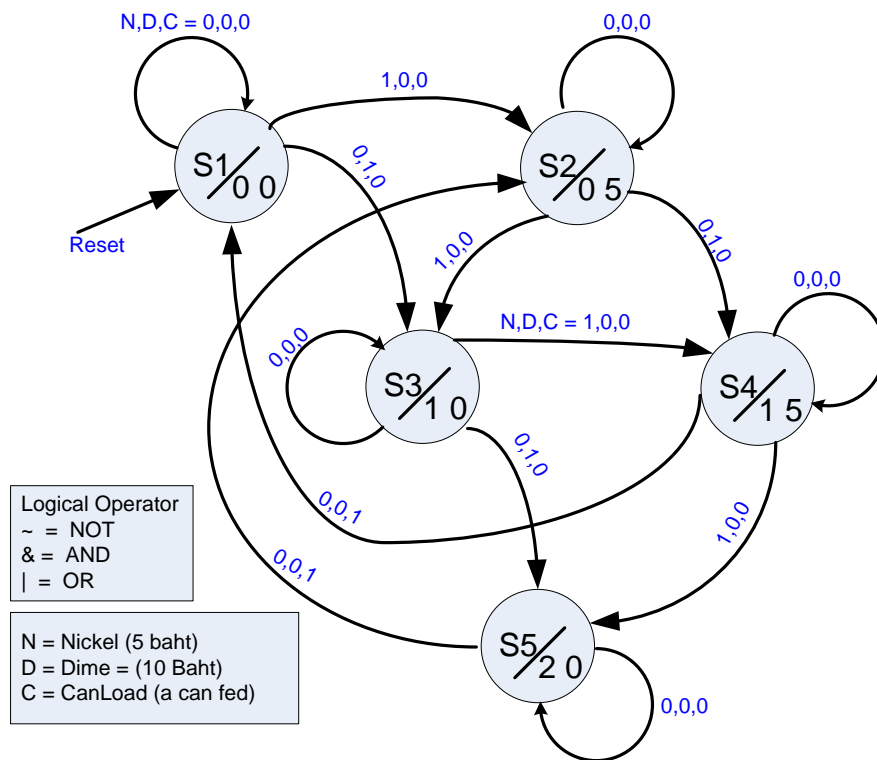


รูปที่ 12

9. ทำการปิดโปรเจกต์ที่สร้างมาในขั้นตอนที่ 5- 8 ก่อนที่จะดำเนินการต่อไป

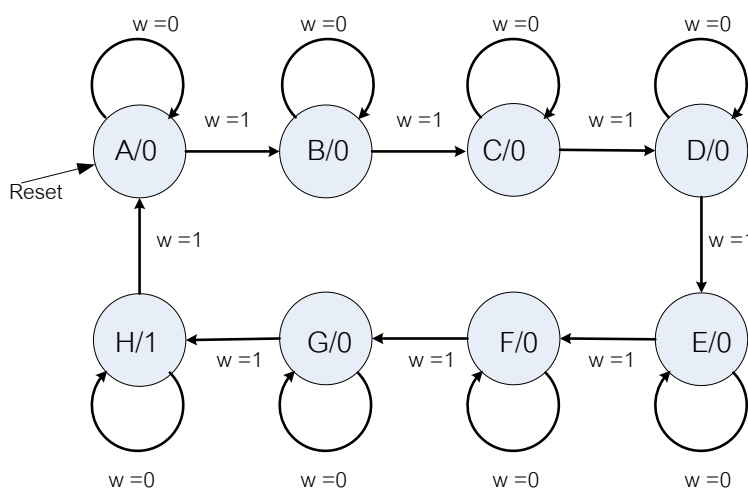
## ขั้นที่ 3 สร้างระบบ Count\_Unit และระบบ Count8to1 เพิ่มขึ้นมาใช้งาน

10. ให้ใช้วิธีในขั้นตอนที่ 5 – 9 มาทำการสร้างอุปกรณ์ดังต่อไปนี้
- ระบบงาน Count\_Unit ดัง state diagram ในรูปที่ 13
  - ระบบงาน Count8to1 (อยู่ภายในระบบงาน Hold5Sec อีกชั้นหนึ่ง) ดัง state diagram ในรูปที่ 14
- (ให้ศึกษาเพิ่มเติมได้จากหนังสือเรียน วิชาดิจิทัล : Fundamentals of Digital logic with VHDL Design 3<sup>rd</sup>, Stephen Brown, Z. Vranesic)



รูปที่ 13

ระบบงานชื่อ Count\_Unit



Present State	Next state		Output CY
	W =0	W =1	
A	A	B	0
B	B	C	0
C	C	D	0
D	D	E	0
E	E	F	0
F	F	G	0
G	G	H	0
H	H	A	1

รูปที่ 14

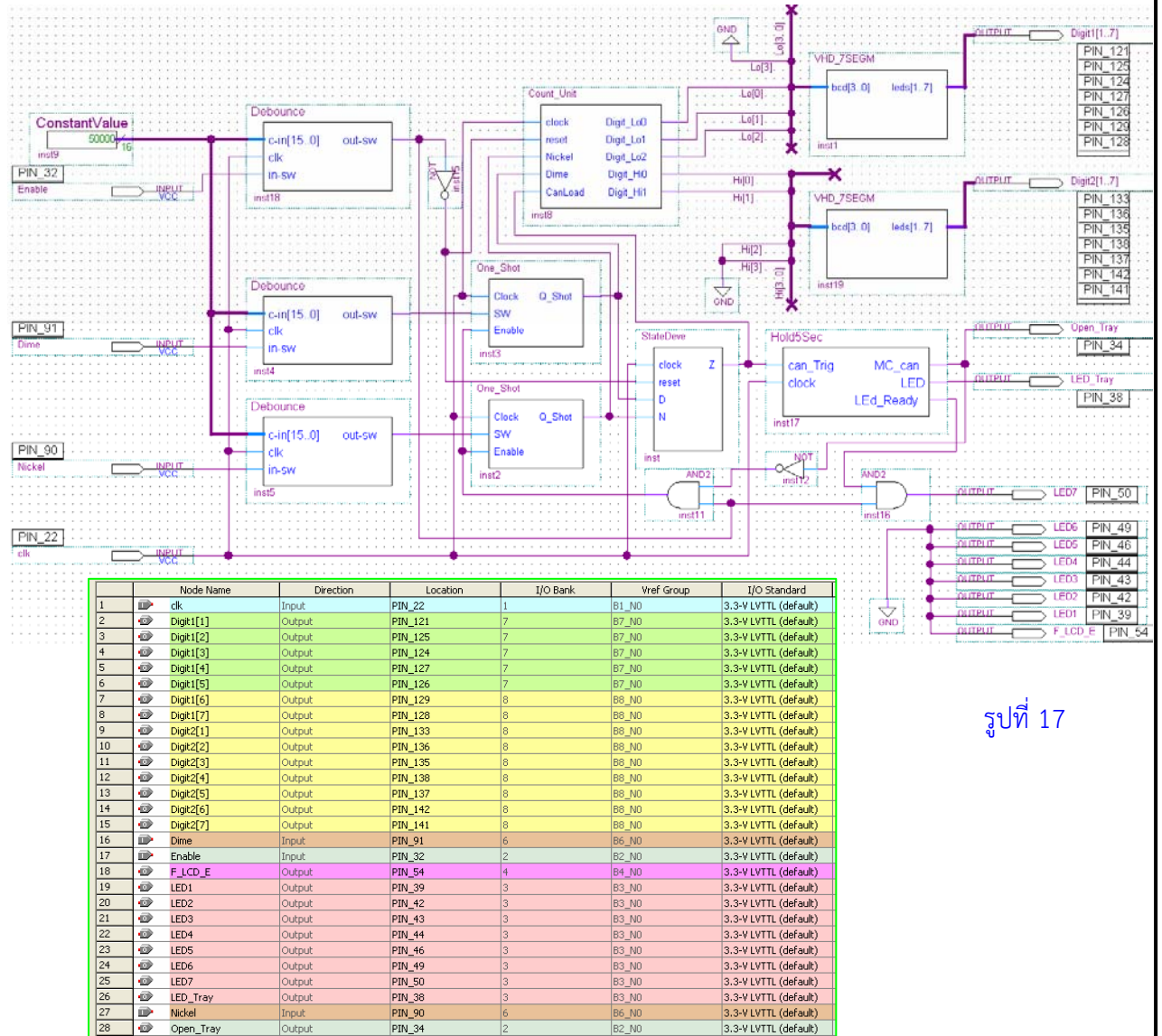
ระบบงานชื่อ Count8to1







- b) กำหนดขา PIN ตามรูป คอมไพล์ซ้ำอีกรอบ และทำการโปรแกรม configuration ลงบอร์ดทดลอง
17. ทำการทดสอบการทำงานของระบบ



รูปที่ 17

## งานมอบหมายท้ายการทดลอง

(ให้เขียนลงบนกระดาษ A4 ส่งในคราวถัดไป)

- ทำการทดสอบการทำงานของระบบให้เข้าใจอย่างลึกซึ้ง (โดยอาศัยแนวคิดจากการออกแบบ, หัวข้อ A1 –A3) เมื่อมีความเข้าใจระบบอย่างดีแล้วให้ น.ศ. อธิบายวิธีการทดสอบอย่างเป็นขั้นตอน จนทำให้อาจารย์ผู้ตรวจสอบสามารถสัมผัสได้ว่า น.ศ.มีความรู้จริงในงานที่ทำมาทั้งหมดนี้ โดยเขียนเป็นรายงานส่งหลังทำการทดลองเสร็จสิ้น
- ระบบในรูปที่ 8a , รูปที่ 13 , และรูปที่ 14 สามารถที่จะออกแบบโดยวิธีอื่นๆ โดยที่ไม่ใช้วิธีการทำ Finite State Machine ได้หรือไม่ ? ถ้าได้ จะต้องทำอย่างไร ทำรายงานส่งด้วย? ถ้าไม่ได้ ให้อธิบายสาเหตุ ทั้งนี้ น.ศ. จะต้องอธิบายจน**ทำให้อาจารย์ผู้ตรวจสอบสามารถสัมผัสได้ว่า** น.ศ.มีความรู้จริงในงานที่ทำมาทั้งหมดนี้

ประโยชน์ ... **ทำให้อาจารย์ผู้ตรวจสอบสามารถสัมผัสได้ว่า ...** Cr. NCH , : established in 2014