2110261 DIG COMP LOGIC LABS

เรื่อง Latch และ Flip-Flop

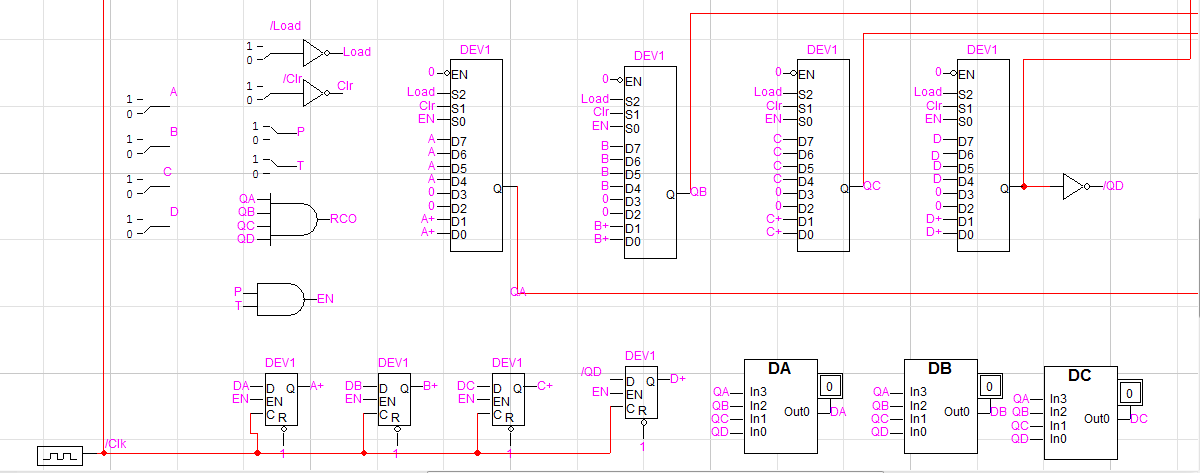
วันที่ 24 สิงหาคม 2553

การทดลองที่ 1

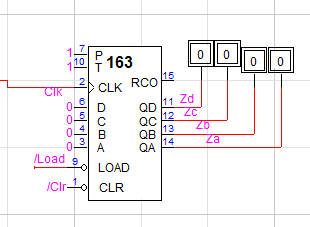
ออกแบบและสร้างวงจรที่ทำงานเหมือนกับ Counter 74163

ผลการทดลอง

วงจรที่สร้างได้คือ



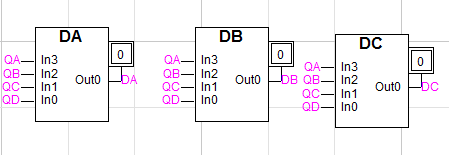
Part ต้นแบบ



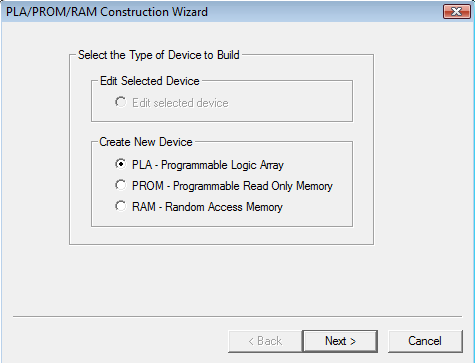
ตารางความจริง

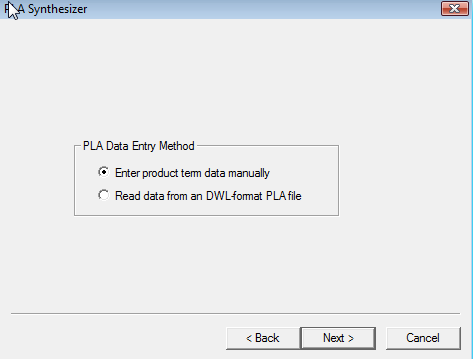
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Present State | | | | Next State & D Flip-flop | | | |
| QA | QB | QC | QD | QA+  (DA) | QB+  (DB) | QC+  (DC) | QD+  (DD) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

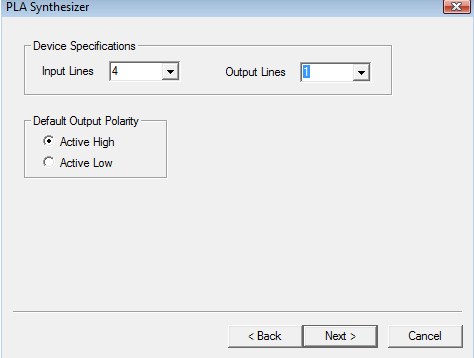
เราใช้ PLA เข้ามาช่วย



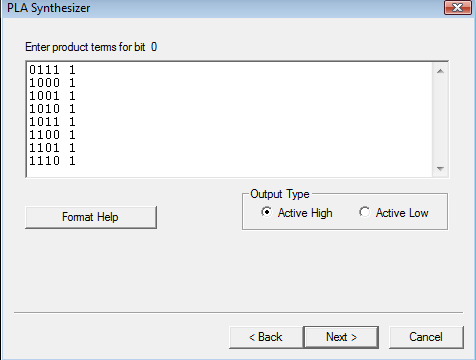
สร้าง PLA



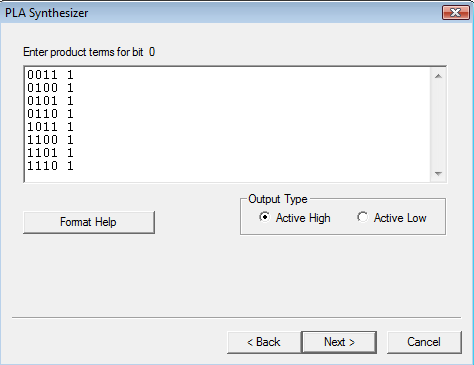




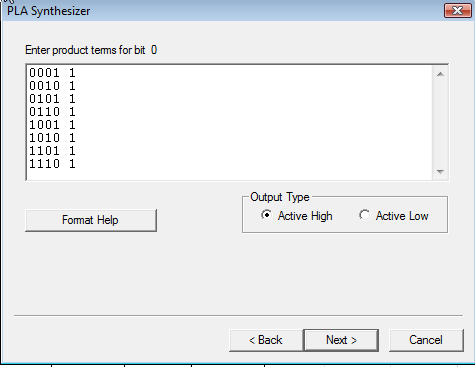
ใน PLA ของ DA



ใน PLA ของ DB

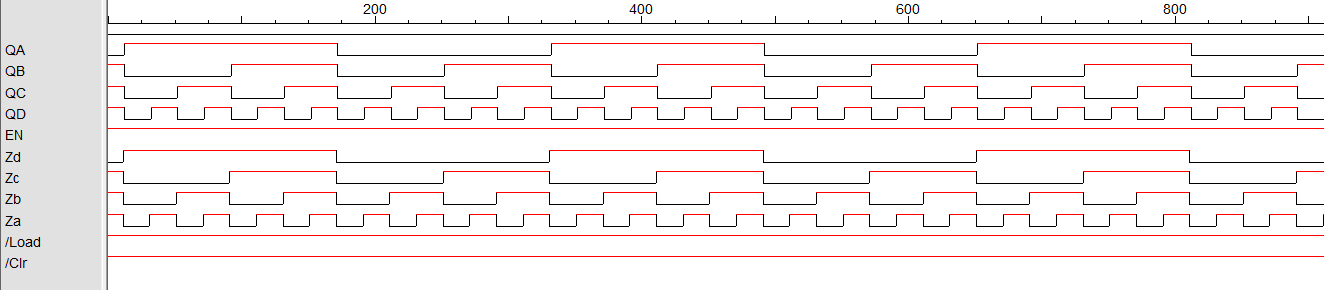


ใน PLA ของ DC



และจะสังเกตว่า DD =QD+ = D’

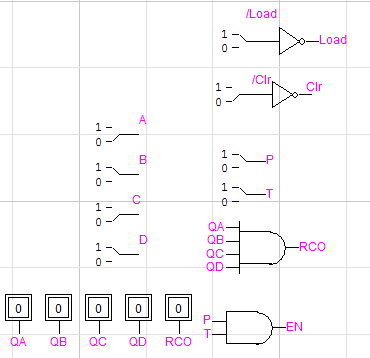
Timing Diagram



ค่า จะเป็นไปตาม Part ต้นแบบทุกประการ

วิเคราะห์ผลการทดลอง

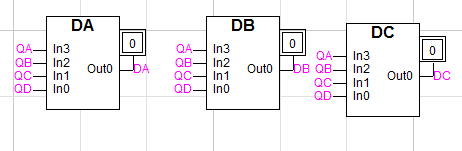
วงจรนี้จะมีการรับค่าอินพุทมา 4 ตัวคือ A B C D และรับอินพุทจาก Binary Switch 4 ตัวคือ P, T, /Load และ /Clr และจะมี Clock อีก 1 ตัว ส่วนเอาท์พุทจะเป็น QA QB QC QD และ RCO



เราจะแบ่งวิเคราะห์วงจรออกเป็น 3 ส่วน

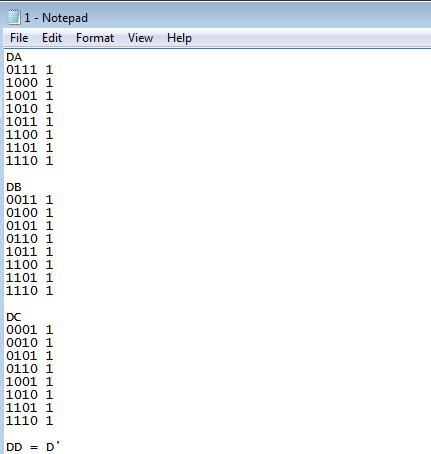
**ส่วนที่ 1**

**PLA ของ DA DB DC**



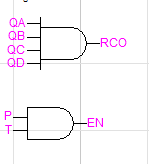
การหา Next state ผลจะออกมาเป็น DA, DB, DC และ DD ตามลำดับ

ดูค่าอินพุทที่สามารถทำให้เอาท์พุท เป็น 1 ได้ของแต่ละตัว เราเขียนใส่ใน Notepad ก่อน แล้วจึงสร้าง PLA ทีละตัว โดยที่ DD มีค่าเท่ากับ D เลย



สร้าง PLA ตามวิธีสร้างในผลการทดลอง

และในส่วนนี้จะมีการหาส่วนของ RCO และการทำงานของ P, T

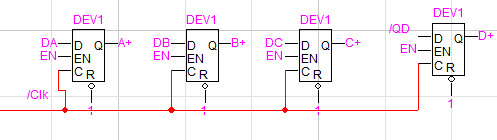


RCO จะเป็น 1 เมื่อค่าเป็น ABCD=1111 ดังนั้นหาก RCO จะเป็น 1 ก็ต้องนำ QA, QB, QC, QD AND กัน ทำให้เป็น 0 ในกรณีอื่นๆ

ส่วน P, T ที่อยู่ในวงจรต้นฉบับจะทำให้ค่าวิ่งไปเรื่อยๆได้ เมื่อ P, T เป็น 1 ทั้งคู่เท่านั้นส่วนในกรณีอื่นๆ จะเป็นการ Hold ค่าไว้ ค่าจะม่ขยับจะคงค่าเดิมไว้ ดังนั้น P, T ต้องเป็น 1 ทั้งคู่ ถึงจะวิ่งไปเรื่อยๆ จึงนำ P T มา AND กัน ให้ผลเป็น EN

|  |  |  |  |
| --- | --- | --- | --- |
| P | T | EN | วงจร |
| 0 | 0 | 0 | Hold ค่าไว้ |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 | วงจรทำงาน |

**ส่วนที่ 2**



เป็นส่วนของการนำค่าที่ได้ Next state DA DB DC DD ในส่วนที่ 1 มาต่อเข้ากับ D Flip-flop ดังนั้นจะได้ว่า

* ถ้านำ DA ต่อเข้ากับขา D Flip-flop ตัวที่หนึ่ง ค่า ที่ออกจาก Flip-flop จะเป็น Next State ของ ค่า DA ตั้งชื่อเป็น A+
* ถ้านำ DB ต่อเข้ากับขา D Flip-flop ตัวที่สอง ค่า ที่ออกจาก Flip-flop จะเป็น Next State ของ ค่า DB ตั้งชื่อเป็น B+
* ถ้านำ DC ต่อเข้ากับขา D Flip-flop ตัวที่สาม ค่า ที่ออกจาก Flip-flop จะเป็น Next State ของ ค่า DC ตั้งชื่อเป็น C+
* ถ้านำ DD ต่อเข้ากับขา D Flip-flop ตัวที่สี่ ค่า ที่ออกจาก Flip-flop จะเป็น Next State ของ ค่า DD ตั้งชื่อเป็น D+

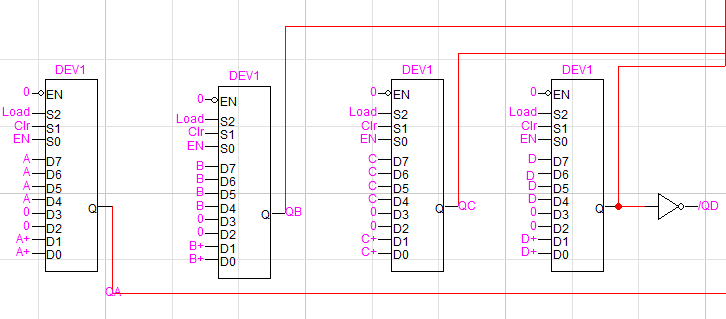
ขา EN จะต่อเข้ากับ EN จากส่วนที่ 1 (P and T)

* ถ้า EN เป็น 1 D Flip-flop ก็จะทำงานได้
* ถ้า EN เป็น 0 D Flip-flop ก็จะหยุดการทำงานลง นั่นคือ ค่า ไม่วิ่งไปเรื่อยๆจะคงที่ (แต่ไม่ได้ Reset ค่าให้เป็น 0)

Clock (Clk) ต่อเข้ากับขา C โดยทุกๆ D Flip-flop จะต้องต่อกับ Clock ตัวเดียวกัน ในส่วนของขา

ขา R คือการ Reset ค่า แต่เราไม่ต้องการให้ค่าถูก reset ด้วยD flipflopนี้ เราจึงใส่ค่า 1 ลงไป เนื่องจากขานี้มีการทำงานแบบ Active Low

**ส่วนที่ 3**



เป็นส่วนของการเลือกด้วย MUX 8:1 เพื่อนำค่าที่ได้จากการเลือกมาแสดงค่าออกเป็นเอาท์พุท เนื่องจาก มีหังก์ชัน การ /Clr การ Load และ EN เข้ามาเป็นอินพุทด้วย ใช้ สี่ตัวเนื่องจากต้องการแสดง เอาท์พุทคือ QA QB QC QD

|  |  |  |  |
| --- | --- | --- | --- |
| Load | CLR  (Clear) | EN  (P, T) | D  (Output) |
| 0 | 0 | 0 | ค่าจาก D flipflop |
| 0 | 0 | 1 | ค่าจาก D flipflop |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | ค่าอินพุทเข้า |
| 1 | 0 | 1 | ค่าอินพุทเข้า |
| 1 | 1 | 0 | ค่าอินพุทเข้า |
| 1 | 1 | 1 | ค่าอินพุทเข้า |

ในที่นี้ /Clr ได้ถูกผ่าน not gate ให้มาเป็น Clrแล้ว เช่นเดียวกับ Load

จาก part จริง ถ้าค่า /Load = 0 นั่นคือ Load =1 นั้น จะทำการโหลดค่าที่ อินพุทเข้าไป A B C D ในทางตรงข้ามถ้า /Load=1หรือ Load=0 ทำให้ไม่มีการโลหดค่าอินพุท A B C D มานับไปเรื่อยๆ ส่วน P T ต้องเป็น 1 ทั้งค่ถึงจะมีการนับไปเรื่อยๆได้ ถ้าเป็น 0 ตัวใด ตัวหนึ่งทำให้ EN เป็น 0 จะทำให้ Hold ค่า ส่วน /Clr เป็น 0 ถ้า Clr=1 จะทำให้ค่าเริ่มใหม่ เป็น0 และทำการเริ่มนับจาก 0 และจะวันกลับมานับ 0 เมื่อไปถึง 1111 แล้วถ้า/Clr=1 หรือ Clr=0 จะเริ่มวนมานับใหม่เมื่อวนครบ ถึง1111 ถึงอย่างไรก็ตาม การทำงานก็ต้องตามค่า Load กล่าวคือ

ในบรรดาตัวควบคุมทั้งหมด Load จะสำคัญที่สุด ถ้าหาก Load = 1 เมื่อไหร่ค่าจะเป็นค่าเริ่มต้นเสมอ (ค่าที่มีอยู่แล้วหรือที่กำลังนับวนอยู่) แต่ถ้า Load = 0 เราจะพิจารณาตัวที่สำคัญรองลงมาคือ CLR หาก CLR = 1 จะเริ่มนับจาก 0 ใหม่ทันที ถ้า CLR = 0 ต้องมาดูที่ตัวสุดท้ายคือ EN (P AND T) หาก EN = 1 เป็นการแสดงผลการนับที่วิ่งได้ ค่าที่เรานำมาใส่ใน MUX คือค่าที่ได้จากการผ่าน D Flip-flop ถ้า EN = 0 จะใส่ค่าที่ได้จาก D Flip-flop เช่นเดียวกัน เพราะสามารถ Hold ค่าไว้ได้เช่นเดียวกัน ใน D flipflop เนื่องจาก EN ที่ D flipflop = 0 (จากส่วนที่ 2)

วงจรที่เราได้นี้จะทำตัวเป็น เหมือน Part ต้นแบบ วงจรแบบ Counter ทุกประการ ซึ่งจะทำงานนับวนไปเรื่อยๆ คือ หา Next state ไปเรื่อย โดย outputมีการใช้ Muxเข้ามาช่วย

สรุปผลการทดลอง

Part ต้นแบบนั้นเป็นวงจรแบบ Counter ซึ่งจะทำงานดังนี้

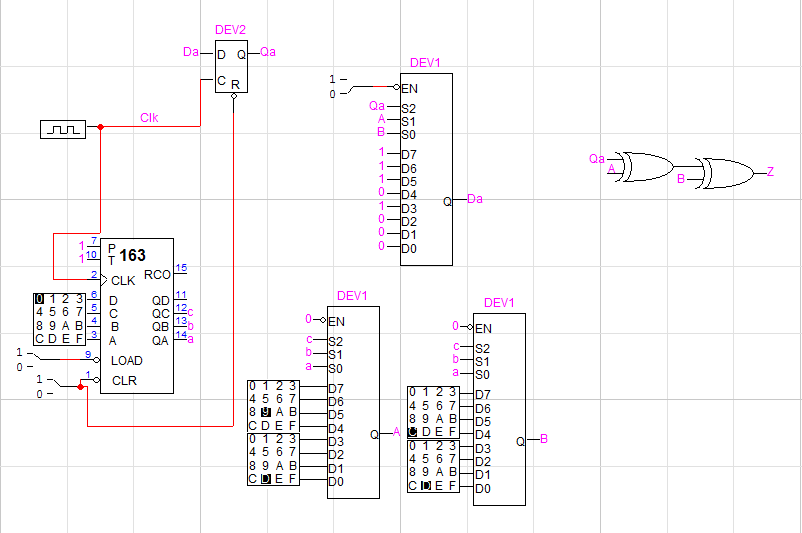
ใส่ Load เข้าไปเพื่อเป็นค่าเริ่มต้น A B C Dโดยเลือกให้ /Clr กับ /Load ให้อยู่ที่การ Load ค่า คือ Load = 0 จากนั้น จะสามารถ Load ค่าจาก Input A, B, C, D บังคับให้ Counter ทำงานโดยเลือก /Load=1, /Clr=1, P และ T ให้เป็น 1 ทั้งหมด ผลลัพธ์ที่ได้คือ QA, QB, QC และ QD และเมื่อ Output เป็น 1 หมด RCO ก็จะเป็น 1

1. ทำการ Clear ค่าเพื่อเริ่มนับใหม่ได้โดย เลือกให้ /Load เป็น 1 และ Clr เป็น 0 ก็จะได้ค่าเริ่มต้นเป็น 0000
2. ในขณะที่ทำการนับไปเรื่อยๆอยู่นั้น เราสามารถ Hold ค่าได้คือ /Load = 1 , /Clr = 1 , P and T = 0 คือ P หรือ T เท่ากับ 0 หรือเป็น 0 ทั้งคู่ก็ได้ แค่นี้ก็สามารถ Hold ค่าไว้ได้

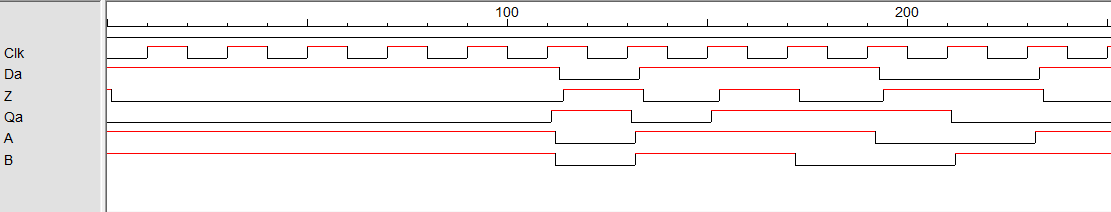
การทดลองที่ 2

ออกแบบและสร้างวงจร “Serial Adder” ซึ่งเป็นวงจรในการบวกเลข 2 จำนวน ที่เข้ามาทีละบิต วงจรมี Input 2 เส้นคือ A และ B (อย่างละ 1 บิต) และ output1 เส้นคือ Z ซึ่งเป็นผลบวกของ A และ B ใช้การออกแบบ แบบ Meary แสดงผลการบวกเลข 10011101 กับ 110001101

ผลการทดลอง



Timing Diagram



วิเคราะห์ผลการทดลอง

**Truth table และ State Diagram**

01/1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Present State Q | Input A | Input B | Next State  (Q+), Da | Output Z |
| S0 (0) | 0 | 0 | S0 | 0 |
| 0 | 1 | S0 | 1 |
| 1 | 0 | S0 | 1 |
| 1 | 1 | S1 | 0 |
| S1 (1) | 0 | 0 | S0 | 1 |
| 0 | 1 | S1 | 0 |
| 1 | 0 | S1 | 0 |
| 1 | 1 | S1 | 1 |

10/1

00/0

S0

00/1

00/1

11/1

11/1

00/1

11/1

S1

10/0

11/1

01/0

Q+ next state = Da ในที่นี้ใช้ Mux 8 เป็นตัวเลือกค่า Da ตาม input ที่ใส่เข้าไปใน Mux คือ Qa A B

Output จะต้องเป็นฟังก์ชันของ state และ input จะได้ว่า Output = Qa xor A xor B

จาก Meary State Diagram จะแบ่งได้ 2 State คือ

1. S0 เป็น State ที่มีความหมายว่าทดเข้า 0
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=0, B=0 จะได้ Next State เป็น 0 ส่วน output จะมีค่าเป็น 0 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=0, B=1 จะได้ Next State เป็น 0 ส่วน output จะมีค่าเป็น 1 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=1, B=0 จะได้ Next State เป็น 0 ส่วน output จะมีค่าเป็น 1 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=1, B=1 จะได้ Next State เป็น 1 ส่วน output จะมีค่าเป็น 0 โดยจะไม่รอ Clock
2. S0 เป็น State ที่มีความหมายว่าทดเข้า 1
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=0, B=0 จะได้ Next State เป็น 0 ส่วน output จะมีค่าเป็น 1 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=0, B=1 จะได้ Next State เป็น 1 ส่วน output จะมีค่าเป็น 0 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=1, B=0 จะได้ Next State เป็น 1 ส่วน output จะมีค่าเป็น 0 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ A=1, B=1 จะได้ Next State เป็น 1 ส่วน output จะมีค่าเป็น 1 โดยจะไม่รอ Clock

\* จาก Meary State Diagram outputจะเปลี่ยนค่าแสดงออกมา โดยไม่ต้องรอ Clock

\* D Flipflop ค่าที่ใส่ใน input D นั้นจะมีเท่ากับ Q+ ซึ่งไม่ต้อง remap วงจร

ในส่วนของ Mux 8 อีก 2 ตัว และ part 74\_163 (Counter) ในที่นี้เราใช้นำมาทดลองบวกเลขจากโจทย์กำหนดคือ

10011101 กับ 110001101 และเราต้องการอินพุทเข้าไปทีละบิต ซึ่งเป็นเรื่องยากมากที่จะเปลี่ยนเองและตรงตามจังหวะ clock เราจึงใช้ counter เข้ามาช่วยในการนับไปเรื่อยๆ โดยนำ เอาท์พุทจาก 74\_163 3ตัว (a, b,c) จากรูปด้านบน มาเป็นตัวควบคุมการเลือกใน Mux 8 ทั้ง 2 ตัวที่จะใช้เลือกค่า A และ B ทีละตัวเพื่อนำไปเข้า Serial adder ที่เราได้สร้างวงจรไว้แล้ว แล้วต่อ Hex keyboard w/o STB เข้ากับ Mux 8 อย่างละ 2 ตัว เริ่มจาก 000 Muxทั้งสองตัวจะได้ ค่า A = 1 B=1 เป็นต้น โดยที่เราจะกดค่าใน Hex keyboard w/o STB ให้ค่าตรงกับทุกตัวจากโจทย์ ในที่นี้ 10011101 กด 9 กับ D ตามลำดับจากบนลงล่างตามรูปด้านบน ให้เป็นค่า A และ 110001101กด C กับ D ตามลำดับจากบนลงล่างตามรูปด้านบน ให้เป็นค่า B เราก็จะได้อินพุท A B อย่างละตัวมาใช้ใน Serial adder

เพราะฉะนั้นถ้าใส่ อินพุทลงไปทั้งหมด คือ 10011101 กับ 110001101จะบวกจาก LSB ก่อน และจะได้ผลลัพธ์เป็นตาม Timeing diagram ข้างบน คือ 01010110(จากหลังมาหน้า) พอกลับแล้วจะได้ 01101010

สรุปผลการทดลอง

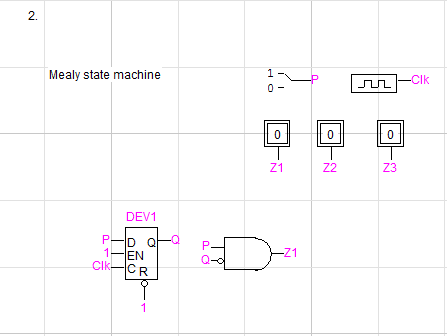
ในการออกแบบวงจรแบบ Meary นั้น output จะอยู่ในฟังก์ชันของ input และ state หากมี input เข้ามา output จะทำการเปลี่ยนในทันที ไม่จำเป็นต้องรอ clock ก่อน ในการทดลองนี้จะออกแบบให้ Present State เป็นค่า Cin และ Next State เป็นค่า Cout มี Output Z เป็น Sum และมี input A และ B

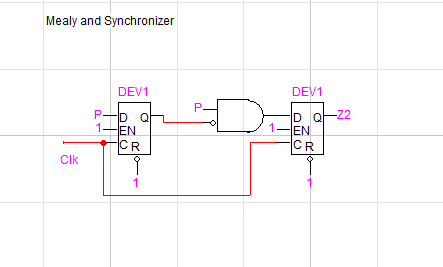
สมมติปัจจุบันอยู่ที่ S0 มี input A=0 B=1 เข้ามา คือจะได้ว่า input รวมกันได้ 1 + carry in คือ 0 ที่ state ปัจจุบัน ในที่นี้จะได้ผล sum = 1 และ carry out = 0 ซึ่งจะได้ว่าต่อไปจะเป็น state S1 นั่นเอง แสดงว่า Carry out เป็น nextstate นั่นเอง เราสามารถดูอย่างนี้ไปจนครบทุกกรณีแล้วก็นำมาเขียน state diagram ออกมา และ สร้างวงจรได้ตามรูป

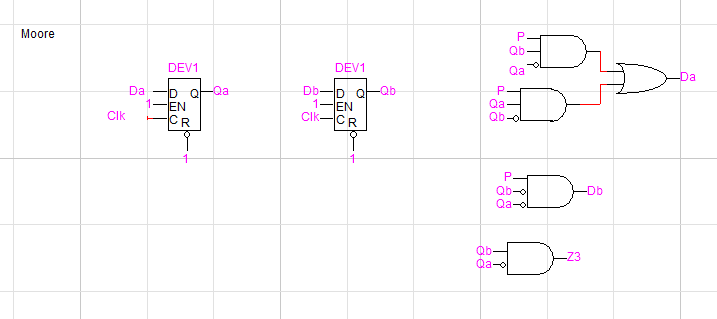
การทดลองที่ 3

ออกแบบและสร้างวงจร Single Pulser วงจรจะมีอินพุท 1 bit คือ P และเอาท์พุท 1 bit คือ Z การทำงานของวงจรคือ Z จะเป็น 1 อยู่ 1 clock ทุกครั้งที่ P เป็น 1 และไม่ขึ้นกับระยะเวลาที่ P เป็น 1 เช่น P อาจเป็น 1 ติดต่อกัน 10 clock แต่ Z จะเป็น 1 เพียง 1 clock ให้ออกแบบโดยใช้การออกแบบแบบ Mealy, แบบ Mealy ที่มี Synchronizer และแบบ Moore แสดงเอาท์พุทของวงจรทั้งสามแบบใน Timing Diagram เดียวกัน โดยใช้ clock และอินพุทเดียวกัน จากนั้นเปรียบเทียบการทำงานและขนาดของวงจรที่ได้ ให้ใช้ state diagram ในการออกแบบ

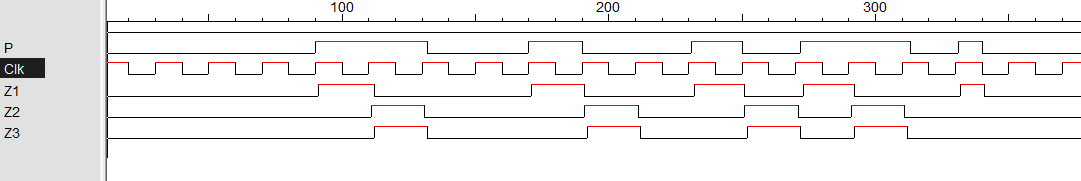
ผลการทดลอง







Timing Diagram



วิเคราะห์ผลการทดลอง

**Mealy**

**Truth table และ State Diagram ของการออกแบบแบบ Mealy**

1/1

0/0

0/0

1/0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | Present State (Q) | Input  (P) | Next State  (Q+) | Output Z1,Z2 |
| S0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| S1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |

จาก Truth table ด้านบน สังเกตว่าเราจะได้

Q+ next state จะมีค่า เท่ากับ P Output จะต้งอเป็นฟังก์ชันของ state และ input จะได้ว่า

Output = Q’P

จาก Meary State Diagram จะแบ่งได้ 2 State คือ

1. S0 เป็น State ที่มีค่าเท่ากับ 0
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 0 จะได้ Next State เป็น S0 ส่วน output จะมีค่าเป็น 0 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 1 จะได้ Next State เป็น S1 ส่วน output จะมีค่าเป็น 1 โดยจะไม่รอ Clock
2. S1 เป็น State ที่มีค่าเท่ากับ 1
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 0 จะได้ Next State เป็น S0 ส่วน output จะมีค่าเป็น 0 โดยจะไม่รอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 1 จะได้ Next State เป็น S1 ส่วน output จะมีค่าเป็น 0 โดยจะไม่รอ Clock

\* D Flipflop ค่าที่ใส่ใน input D นั้นจะมีเท่ากับ Q+ ซึ่งไม่ต้อง remap วงจร

การสร้างวงจรนำ Q+ ไปต่อกับ D Flip-flop จะได้ผลลัพธ์เป็นค่า Q ออกมา จากนั้นในการหา output จะได้ว่า output = Q’P ดังนั้นทำการกลับค่า Q ที่ได้จาก D Flip-flop แล้วนำมา AND กับ P ก็จะได้ output ออกมา

**Mealy** **Synchronize**

เป็นเหมือน Mealy ทุกประการ แต่เราเพิ่ม D Flip-flop อีกตัวลงไป โดยนำ output ที่ได้จากการออกแบบแบบ Mealy มาต่อเข้ากับ ขา D ของ D Flip-flop อีกตัว เราจะได้ output ที่ได้จากการต่อ D Flip-flop ตัวที่สอง ก็คือ output ของ Mealy Synchronize

หมายเหตุ : D Flip-flop 2 ตัวต้องใช้ Clock เดียวกัน

**Moore**

**Truth table และ State Diagram ของการออกแบบแบบ Moore**

1

11/1

0

0

0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| State | Present State | | Input | Next State | |
| Qa | Qb | P | Qa+  (Da) | Qb+  (Db) |
| S0  Output [Z3=0] | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| S1  Output [Z3=1] | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| S2  Output [Z3=0] | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| xx | 1 | 1 | 0 | X | X |
| 1 | 1 | 1 | X | X |

จะได้ Da=Qa+ = Qa’QbP + P QaQb’, Db =Qb+ = P Qa’Qb’, เนื่องจากแบบ Moore Output เป็นฟังก์ชันของ state เพียงอย่างเดียว ดังนั้น Output = Qa’Qb

จาก State Diagram จะแบ่งได้ทั้งสิ้น 3 State คือ

1. S0 จะให้ output เป็น 0
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 0 จะได้ Next State เป็น S0 ส่วน output จะมีค่าเป็น 0 โดยจะรอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 1 จะได้ Next State เป็น S1 ส่วน output จะมีค่าเป็น 1 โดยจะรอ Clock
2. S1 จะให้ output เป็น 1
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 0 จะได้ Next State เป็น S0 ส่วน output จะมีค่าเป็น 0 โดยจะรอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 1 จะได้ Next State เป็น S2 ส่วน output จะมีค่าเป็น 0 โดยจะรอ Clock
3. S2 จะให้ output เป็น 0
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 0 จะได้ Next State เป็น S0 ส่วน output จะมีค่าเป็น 0 โดยจะรอ Clock
   * ถ้าหากมีค่า input เข้ามาเท่ากับ 1 จะได้ Next State เป็น S2 ส่วน output จะมีค่าเป็น 0 โดยจะรอ Clock

\* D Flipflop ค่าที่ใส่ใน input D นั้นจะมีเท่ากับ Q+ ซึ่งไม่ต้อง remap วงจร

การสร้างวงจรนั้นนำ Da และ Db ไปต่อกับ D Flip-flop แต่ละตัว ผลลัพธ์ที่ได้จาก D Flip-flop คือ Qa และ Qb ตามลำดับ เมื่อได้ Qa และ Qb นำมาหา Output โดยที่ มีแค่ State S1 (Qa=0, Qb=1) จะมี Output เป็น 1 เพราะฉะนั้นการที่ Output จะเป็น 1 ก็ต่อเมื่อ Q1’Q0

สรุปผลการทดลอง

จากการเปรียบทียบวงจรทั้งสามวงจร จะได้ว่า

การทำงานของ Mealy Synchronize และ Moore มีการทำงานเหมือนกัน สังเกตได้จาก timing diagram ด้านบน ในขณะที่ input มีการเปลี่ยนแปลง จาก 0 ไป 1 หาก input เข้าช้ากว่า positive edge(rising edge) ของ clock Output จะยังไม่ถูกแสดงออกมาเป็น 1 และจนกว่าค่า input ที่เป็น 1 มาเจอ positive edge (rising edge ) ของ clockครั้งที่ 2 จึงจะเปลี่ยนเป็น 1 จึงสรุปได้ว่าในการที่ output จะเปลี่ยนค่านั้นจะต้องรอ clock ก่อน ข้อสังเกตคือ ทั้งสองแบบนี้จะได้ Output ที่มีค่าเป็น 1 อยู่ 1 clock ทั้งสิ้น ดูจาก Timing diagram ข้างบน ไม่ว่า input จะอยู่เป็น 1 ค้างหรือเปลี่ยนเป็น 0 ลงก่อนที่ Output จะเป็น 0

ในขณะที่การออกแบบแบบ Mealy ทันทีที่ input เป็น 1 ค่าของ output จะเปลี่ยนไปตาม input โดยทันที โดยที่ไม่ต้องรอclock ดูจาก timing diagram ด้านบน หาก input จาก 0 ไป 1 เกิดขึ้นเร็วมาก แบบ Mealy จะเกิด glitch ขึ้น ดูจาก timing diagram ด้านบน เพราะค่าที่เปลี่ยนออกมาที่ output จะไม่รอ clock และเป็นไปตาม input ดังนั้นหากสับ input ยังไม่ครบ 1 clock output แบบ Mealy จะมีค่าเหมือน input และจะเกิดไม่ครบ 1 clock

ดังนั้นเราสามารถบอกได้ว่าการที่จะออกแบบวงจรแบบ Mealy ที่ให้ได้ผลเหมือนวงจรแบบ Moore นั้นก็คือการออกแบบโดยใช้แบบ Mealy Synchronize นั่นเอง ขนาดของวงจรที่ได้โดยใช้ Mealy state diagram ในการออกแบบ จะมีขนาดวงจรเล็กกว่าแบบ Moore แบบ Mealy จะได้ output ออกมาเร็วกว่า Moore 1 clock โดย state diagram ของ Mealy มีแค่ 2 state แต่แบบ Moore มี 3 state นอกจากนี้