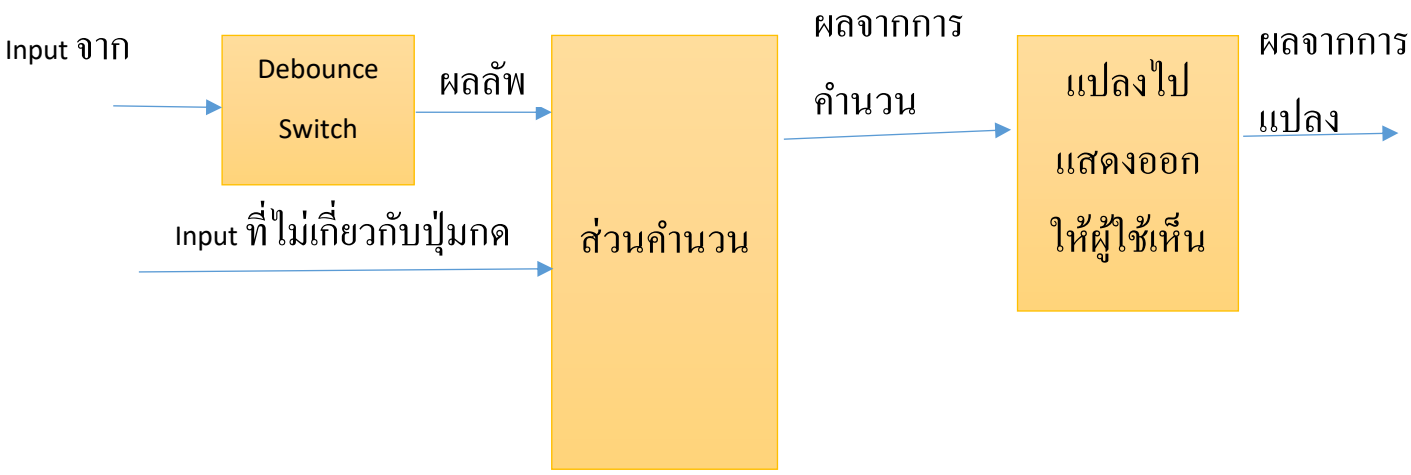
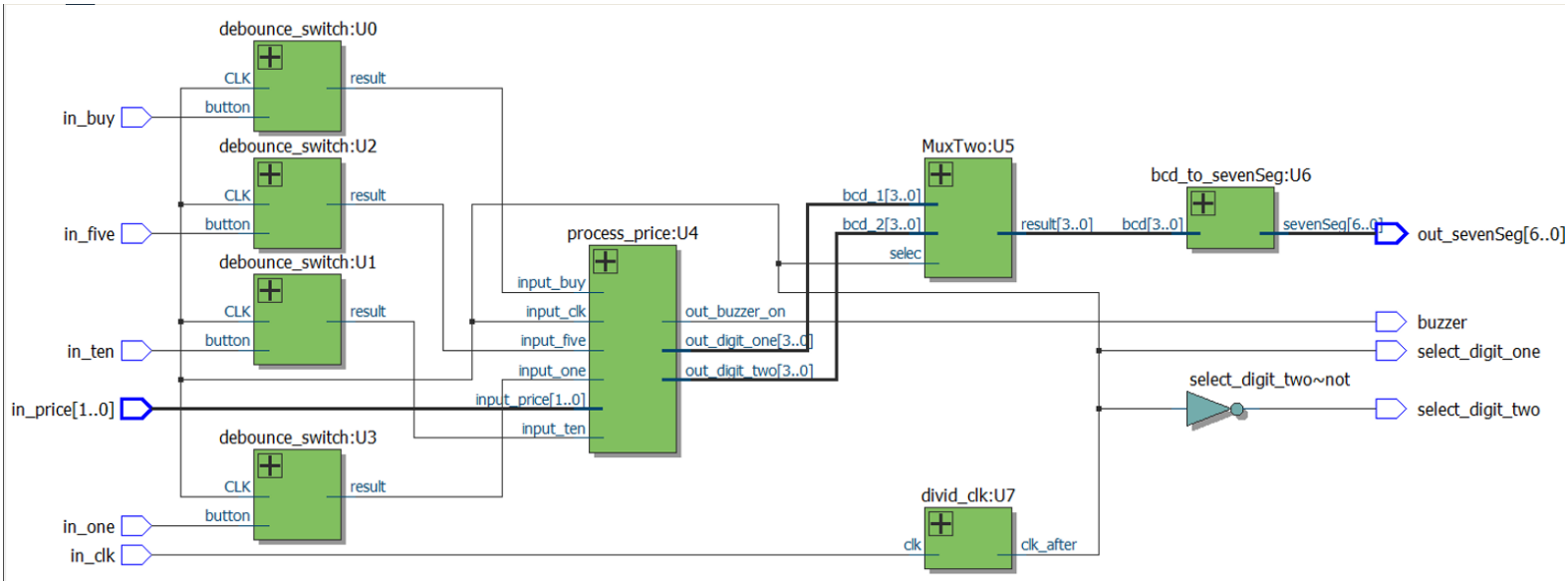
การออกแบบเบื้องต้น

ช่วงที่ 1

จากที่เราต้องการใช้ ปุ่มกดในการ เป็น input บางส่วน เราจึงจำเป็น ตัวใช้ debounce switch ในการตัดสัญญาณ รบกวนออกไป แล้วจึงเอา ผลลัพธ์ที่ได้ไปเข้าสู่กระบวนการคำนวณ แล้วจึงเอาผลลัพธ์ จากการคำนวณนั้นไป แสดง ผล ซึ่งจะร่างออกมาคร่าวๆ ได้ประมาณ นี้

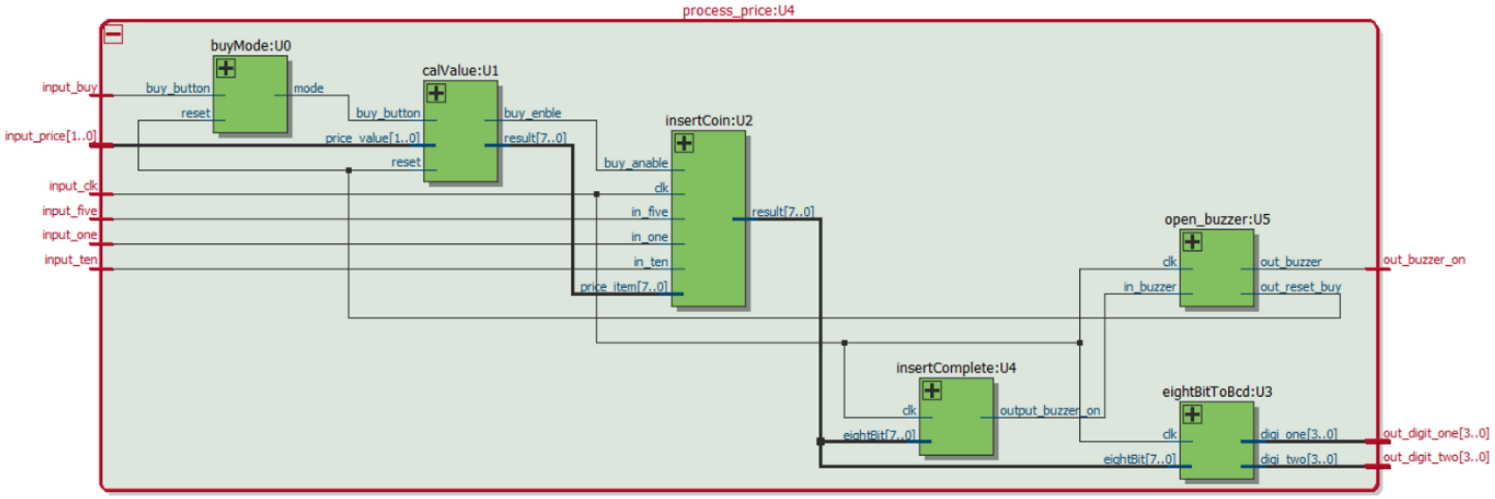


และ จากนั้นจึงได้นำ ไปทดลองเขียนจริงด้วย VHDL แล้วได้ block diagram ดังรูปด้านล่าง นี้ขึ้นมา



ซึ่ง การทำงานหลักๆ ก็คือ การให้ input ที่มาจากการกด ปุ่มนั้น ผ่าน การ ตัดสัญญาณ รบกวนก่อน แล้วนำผลลัพธ์นั้นไป เข้ากระบวนการคำนวณ ส่วน input ที่มาจาก slide switch นั้น ถูกนำไปใช้โดยตรง เลย ส่วนสัญญาณ clk ในช่วงนี้ ใช้ ผล จากการหาร สัญญาณ ให้เหลือประมาณ 50 hz ก่อน แล้วนำไปใช้ในการกำหนดจังหวะการทำงาน ทั้งระบบ ส่วน output นั้น buzzer จะ ถูกส่งตรงออกจากส่วนคำนวณเลย แต่ว่า ส่วนที่ แสดง ผลเงิน นั้น จะต้องนำไป แยกการแสดง ผล โดยใช้ mux 2_ 1 เพื่อที่จะเลือกการ แสดง ผล ออกทาง 7 segment เนื้องจาก 7 segment ที่ เลือกใช้นั้น มีการใช้ input ร่วมกัน ทั้ง 2 digit จากนั้นนำผลลัพธ์ ที่ได้นั้น ไป ผ่านการ แปลงเข้ารหัส เผื่อ แสดง ผล ออก ทาง 7 segement ถูกต้อง โดย input ที่จะนำเข้า mux2_1 นั้นจะเป็น output ที่มาจาก process_price ซึ่งจะแยก เป็น 4 bit ซึ่งมีความหมายเป็น หลักหน่วยและหลักสิบ จากนั้น output จาก mux 2_1 ที่เป็น BCD นั้นจะถูกนำไปแปลง ให้เป็น รหัส ของ 7 segment

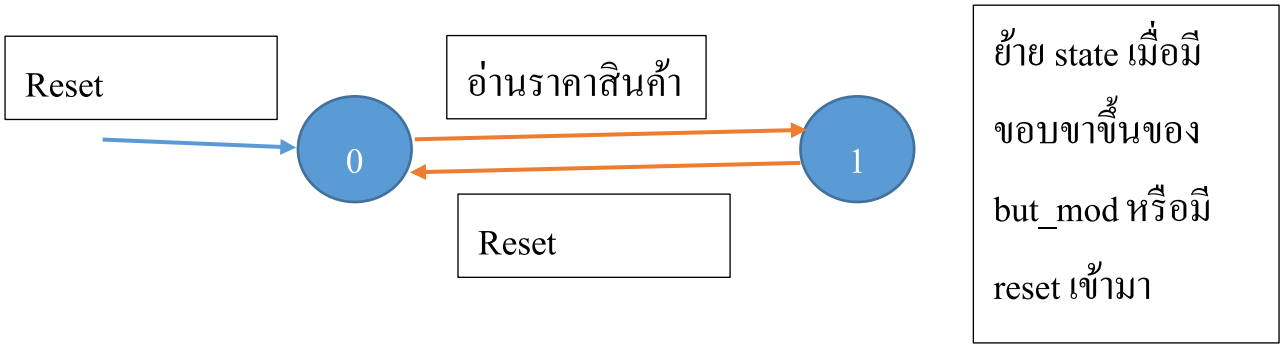
ซึ่ง ใน ส่วนของ process_price นั้น จะเป็นการประกอบกัน ของ component ย่อยๆ ภายใน ดังนี้



ซึ่ง ข้างในนี้จะมีหลักการการทำงานที่ ค่อนข้างซับซ้อน เกินความจำเป็น ซึ่งมีขั้นตอน คร่าวๆ ดังนี้

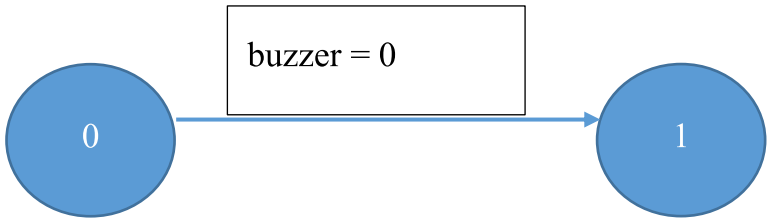
ในกล่อง buy_mode นั้นจะมีการเช็คว่า มีการกดปุ่ม buy มั้ย โดยมีการเช็ค ค่า reset ก่อน เสมอ จากนั้นเช็คว่ามีขาขึ้นของ buy เข้ามามั้ย ถ้ามี จะทำการสลับ output จาก low เป็น hight จาก hight เป็น low ซึ่ง ใช้การใน เปิด ปิด การทำงานของ cal_value ซึ่ง การ reset จะทำให้ เกิด output เป็น ‘0’

ซึ่งต่อมาในส่วนของ cal_value นั้น จะ เป็นส่วนที่กำหนดราคาสินค้างที่ผู้ซื้อต้องจ่ายซึ่งจะเช็คการ reset ก่อน หาก เป็น ‘1’ จะทำการ reset state เป็น state 0 และ เงินที่ต้องจ่าย นั้น ก็กลายเป็น 0 แต่หากว่า reset ไม่เท่ากับ ‘1’ แล้วนั้นจะเช็คขอบขาขึ้นของ mode ที่ได้มาจาก output ของ buy_mode ซึ่ง ถ้าอยู่ state 0 จะ เปลียนเป็น state 1 และ อ่านราคาส่งเป็น output ถ้าอยู่ state 1 จะทำเหมือนกับการ reset ค่า

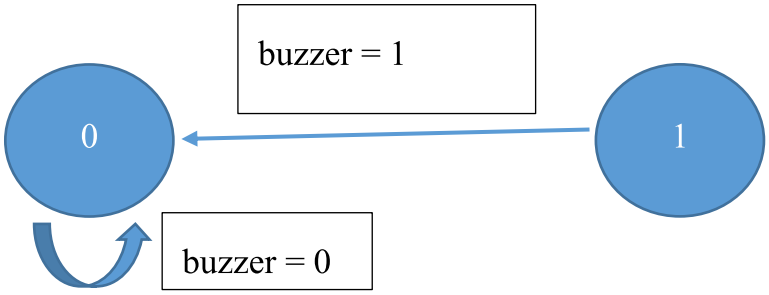


ซึ่งต่อในในส่วนของ insert_coin จะเป็นส่วนที่ ทำงานเมื่อมีขอบขาขึ้นของ clk และ buy_mode นั้นมีค่าเป็น ‘1’ การทำงานคร่าวๆคือ ลบราคาสินค้า หาก มีการหยอดเหรียญ และส่ง output เป็น ราคาสินค้าที่จะแสดงบน 7 segment โดยส่งเป็น 8 bit ซึ่งจะถูกนำไปเป็น input ของ component อื่น ซึ่งก็คือส่วนแรกเป็นส่วนที่แยก 8 bit ที่หมายถึง ราคาสินค้าที่รวมทั้งหลักหน่วยและหลักสิบ แยกออกเป็น หลักสิบ และ หลักหน่วย อย่างละ 4 bit ส่งออกเป็น output โดยใช้การ หาร10 และการ mod 10 ในการแยกหลักหน่วยและหลักสิบ และ อีกส่วน ก็นำไป ใช้ในการเช็คการจ่ายเงินว่าจ่ายเงินครบแล้ว ก็คือส่วนที่เรียกว่า insertcoin_complete การทำงานของส่วนนี้นั้นจะมีการแปลง ค่า 8 bit เป็นค่า integer ก่อนจากนั้นนำไปเช็คตาม state ดังนี้

กรณีที่ input หลังจากแปลง เป็น int แล้ว ไม่เท่ากับ 0



กรณีที่ input หลังจากแปลงเป็น int แล้ว เท่ากับ 0

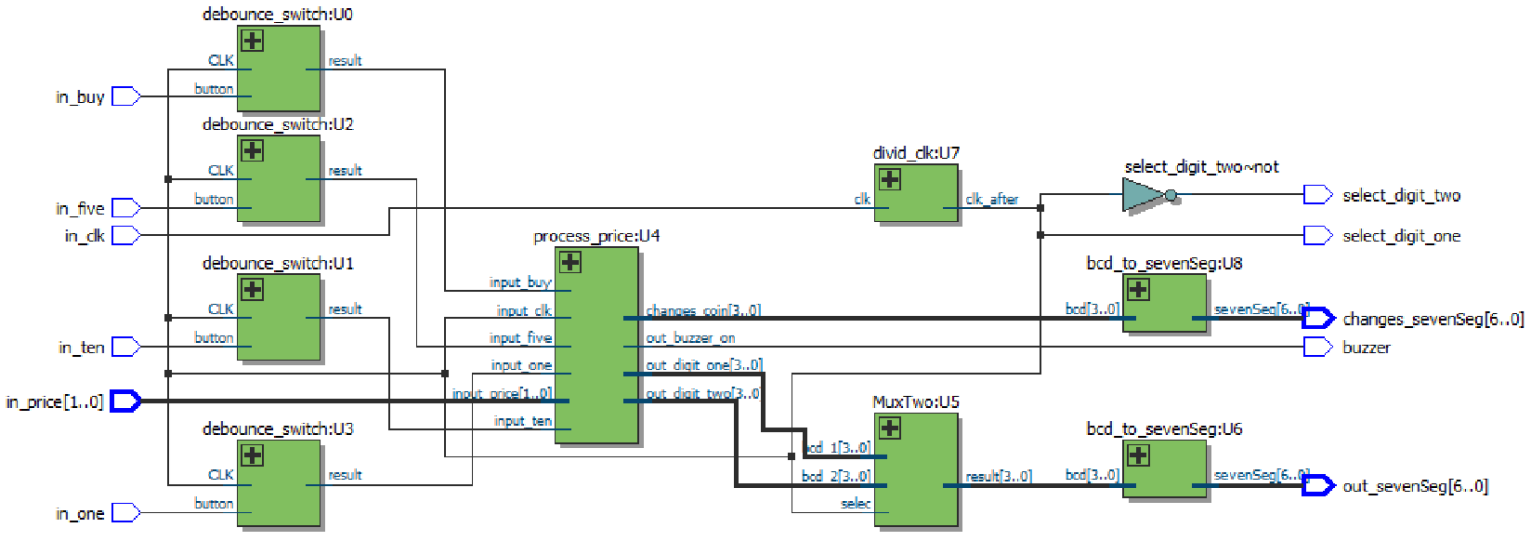


ซึ่งจะเช็ค ทุกๆ 1 clk ขาขึ้น ซึ่งหมายความว่า หากมีการจ่ายเงินครบ 1 ครั้ง หรือ เงินที่ต้องจ่ายนั้น มีการเปลียนแปลง จาก 0 ไปเป็นค่าใดๆ แล้ว เปลี่ยนจากค่าใดๆนั้น กลับเป็น 0 อีกครั้ง จะมี สัญญาณ output 1 clk ส่งออกไป ซึ่งจะนำไปใช้เป็น input ของส่วนที่ให้ buzzer ทำงานคือ ส่วนที่เรียกว่า open_buzzer ซึ่งมีการทำงานคร่าวๆคือ เมื่อเช็ค ได้ว่าการจ่ายเงินครบโดยใช้ input ที่มาจาก output ของ ส่วนเช็คการจ่ายเงิน แล้วจะเกิดเสียง buzzer ดัง ยาว 1 วินาที โดย ที่ จะมี output อีกส่วนนึง ที่ จะทำหน้าที่ ไปเป็นตัว reset ของ buy_mode และ cal_val ซึ่งจะทำให้ระหว่างที่ buzzer ดัง จะเป็นการ reset ระบบไปในตัวและ ไม่สามารถหยอดเหรียญ หรือ กระทำการใดๆ ได้

ซึ่ง การใช้ โค้ด ในช่วงนี้นั้น ผลการจำลอง ถือว่า ถูกต้อง แต่ว่า การใช้การจริงนั้น เกิดบัคที่ ยังไม่ทราบสาเหตุ คือ ระบบ ในตอนโหลดโค้ดลงไปครั้งแรกนั้น มีการข้ามสถานะ คือมีการอ่านค่า ว่ามีการกดปุ่มbuy 1 ครั้ง โดยที่ยังไม่ได้กดปุ่ม แต่หากว่าเรากด reset (กด buy 1ครั้ง) แล้ว ก็สามารถเล่นได้ตามปกติ

ช่วงที่ 2

ในช่วงนี้นั้น จะมี block diagram ดัง รูป นี้ คือ

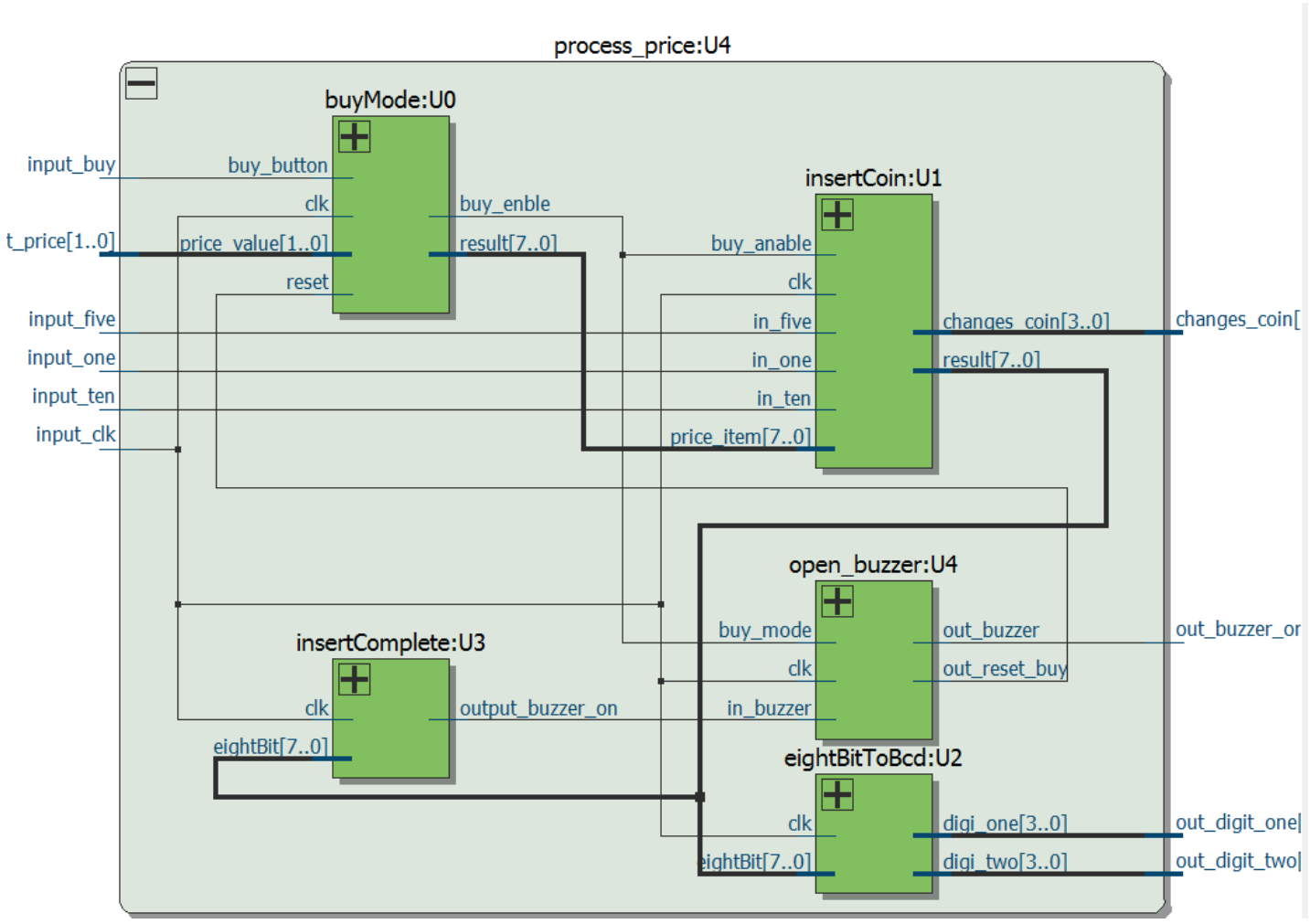


วงจรรอง : process_price การทำงาน ณ ช่วงที่ 2

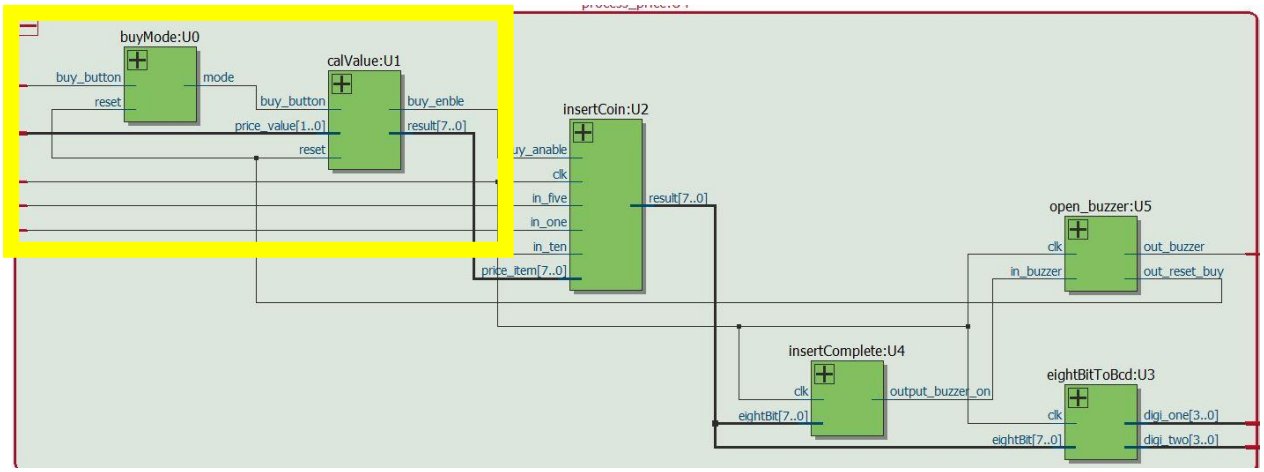
ในส่วนนี้เราจะใช้เป็นส่วนคำนวน โดยเมื่อเลือกสินค้าเสร็จ ราคาสินค้าจะถูกนำไปแสดง การคำนวณ เราให้การหยอดเหรียญในแต่ละครั้งจะถูกนำไปลบกับราคาสินค้า แล้วเมื่อลบเสร็จแต่ละครั้งก็จะถูกแสดงไปที่ 7 segment โดยที่แยกหลักหน่วยกับหลักสิบก่อนแล้วส่งไปที่ Mux เมื่อจ่ายตังครบ จะมีเสียง buzzer ดังขึ้นมา 1 วินาที หรือ ถ้าหากว่า ผู้ซื้อทำการยกเลิก การสั่งซื้อจะทำให้เกิดเสียงเป็นจังหวะ 1 วินาที

ส่วนประกอบของวงจรรอง

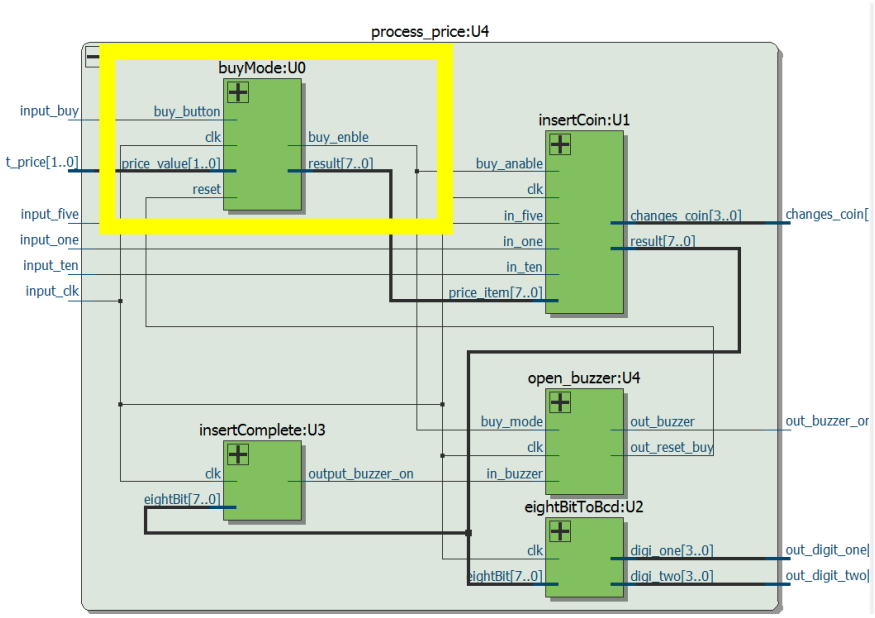
1) Buymode : เป็นส่วนที่ใช้เลือกราคาของสินค้า และการยกเลิกการซื้อสินค้า โดยจะมี 4 state ที่ใช้ในการเช็คสินค้าในแต่ละครั้ง
2) InsertCoin : เป็นส่วนที่ใช้ในนำค่าจากเหรียญไปลบกับราคาสินค้า โดยจะมี 2 state ที่จะเช็คการหยอดเหรียญในแต่ละครั้ง
3) insertComplete : เป็นส่วนที่ใช้เช็คว่าจ่ายเงินไปครบหรือไม่โดยนับทั้งจากการที่ผู้ใช้ กดยกเลิก หรือว่าทั้งจ่ายเงินครบก็ตาม
4) open_buzzer : จะดังเมื่อ insertComplete เช็คแล้วว่า จ่ายเงินครบ หรือ ว่าผู้ใช้กดยกเลิก โดยจะมี 3 stage ซึ่ง มีกรณีแตกต่างกันไป
5) eightBitToBcd : เป็นส่วนที่ใช้แยกระหว่างหลักหน่วยกับหลักสิบของเงินที่ผู้ซื้อต้องจ่าย



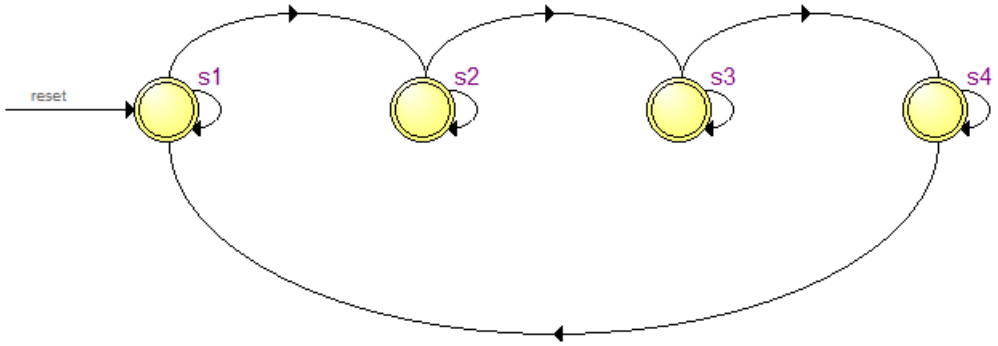
มีการเปลี่ยนแปลง component ภายใน Process Pirce คือ



ได้ทำการรวม component BuyMode และ CalValue ที่ทำหน้าที่คำนวนราคาสินค้า หลังจาก ที่ได้รับสัญญาณ ขาขึ้น จาก BuyMode มาเป็น Component BuyMode ชิ้นเดียวแทน ซึ่ง ใช้ 4 stage และให้ output เหมือนกันโดยใช้ input เหมือนเดิม



Stage ภายใน Buy Mode



โดย ที่ การทำงาน จะเช็ค ก่อนว่า reset mode ทำงานอยู่หรือไม่ (ทำงานเมื่อ buzzer ดัง) ถ้าทำงานจะให้มา stage 1 และ set ค่าเป็นค่าเริ่มต้นทั้งหมด และถ้าไม่ทำงาน จะเริ่มเช็คที่ขอบขาขึ้นของ clk ซึ่ง ถ้าอยู่

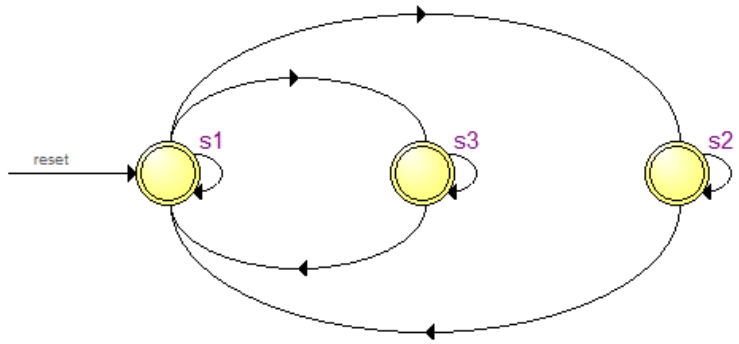
stage 1 และ มีการกดปุ่ม หรือก็คือ Input = ‘0’ จะเข้า stage 2 และ ทำการอ่านค่าราคาสินค้า และส่งไปที่ component insertCoin และทำการ ตั้งค่าโหมดการซื้อ Buy_enble = ‘1’ หรือก็คือ ให้สามารถ คิดราคาจากการหยอดเหรียญได้

stage 2 จะทำงานเมื่อปล่อยปุ่ม หรือ input = ‘1’ จะทำการย้ายไป stage 3

stage 3 จะทำงานเมื่อกดปุ่มอีกครั้ง หรือ input = ‘0’ จะทำการย้ายไป stage 4 และทำการ reset ราคา = 0 และ buy_enble = ‘0’ หรือก็คือไม่สามารถ จ่ายเงินได้อีกแล้ว

stage 4 จะทำงานเมื่อ ปล่อยปุ่มหรือ input = ‘1’ จะทำการย้ายไป stage 1 หรือ stage เริ่มต้นอีกครั้ง

Stage ภายใน OpenBuzzer



ซึ่ง การทำงานคือ จะเช็คก่อนว่า ในจังหวะที่เป็น clk ขาขึ้นนั้น in_buzzer หรือ input ที่ได้มาจาก component insertCoinComplete นั้น เป็น ‘1’ หรือไม่ ถ้าใช้ จะเช็ค ว่าถ้า อยู่ที่

Stage 1 นั้นจะเช็คอีกอย่างว่า buy_mode นั้นทำงานอยู่หรือไม่ (เพื่อเช็คว่า เป็นการที่ราคาเปลี่ยนจาก xxx > 0 นั้น เกิดจาก การที่ ผู้ซื้อยกเลิก หรือว่า เกิดจากการที่ ผู้ซื้อจ่ายเงินครบกันแน่) ถ้าทำงาน ให้ไป stage 2 ถ้าไป ให้ไป stage 3 และทำการกำหนด output buzzer =1 เพื่อไปเปิดเสียง buzzer และ กำหนด output reset_buy เพื่อไปปิดการทำงานของการซื้อ และ reset การทำงานของ Buymode

Stage 2 และ 3 จะทำการ set count_delay = 0

แต่ถ้าหากว่า in_buzzer นั้นไม่เท่ากับ 1 นั้นจะเริ่มทำการเช็คว่าอยู่ stage ไหน เช่นเดียวกัน

Stage 1 จะทำการ set counter_delay , counter_delay_duty = 0

Stage 2 จะเข้า stage นี้ได้ก็ต่อเมื่อ เกิดจากการที่ผู้ใช้จ่ายเงินครบเท่านั้นซึ่ง การทำงานจะเป็นการนับ count_delay ไปเรื่อยๆ จนครบจำนวนเวลาหนึ่ง ตามที่กำหนดไว้ (นับตาม clk) และพอมันนับครบ ตามที่กำหนดไว้ ก็จะทำการปิด buzzer และ ปิดการ reset buymode และกลับไป stage 1

Stage 3 จะเข้า stage นี้ ก็ต่อเมื่อ เกิดจากการที่ ลูกค้านั้น ทำการ ยกเลิกคำสั่งซื้อ ซึ่งข้างในวงจรนี้ นั้น นอนจากจะทำการนับ จำนวนเหมือน stage 2 แล้ว จะทำการนับ count พิเศษ มาอีก 1 ตัว ซึ่ง การนับตัวนี้จะเป็นการนับ เพื่อทำให้สัญญาณ นั้นเกิดการเปลี่ยนแปลงค่า duty cycle ซึ่งที่กำหนดไว้ตอนนี้คือ เมื่อนับจนถึง

count_delay_duty > max_delay /10 จะทำให้ out_buzzer = 0 และ นับต่อเรื่อยๆ จนกระทั่ง count_delay_duty > max_delay /5 จะ ทำการเปิด buzzer อีกครั้ง และทำการ reset การนับ count_delay_duty = 0 และทำอย่างนี้ไปเรื่อยๆ จนกระทั่ง counter_delay > max_delay

ซึ่ง ในช่วงนี้ ก็ยัง พบปัญหาอยู่ คือ เมื่ออัพโหลด โค้ด ลงบอร์ดครั้งแรกนั้น จะมีบัค ที่ว่า มีการกดปุ่ม buy 1 ครั้ง โดยที่ยังไม่ได้กดปุ่มใดๆเลย ซึ่งก็สามารถเล่นได้ตามปกติ ซึ่ง ในช่วงต่อไป หรือ ช่วงสุดท้าย นั้น จึงได้ ทำการแก้ไข ลำดับโค้ดใหม่เกือบทั้งหมด ให้กลายเป็น state รวมกัน แทนที่จะแยกเป็น component มาต่อกันใน process_price ซึ่ง ผลลัพธ์ ที่ได้ นั้น ไม่มีบัคเกิดขึ้นแล้ว

ช่วงที่ 3 เริ่มคิด ใหม่และ เขียนใหม่ แก้ไข ส่วนคำนวณผลจากมีวงจรย่อยให้หายไป