



รายงานผลการดำเนินการโครงการ (Final Project)

การพัฒนาเกมเตตริสด้วย FPGA

คณะผู้จัดทำ

66070501004 กัญญวรรณ ทยานิพันธ์

66070501005 กัลยา สุทธปัญญา

66070501009 จิรพรรณ ศักดิ์พิสุทธิพงศ์

66070501034 เปมิกา จงขวัญยืน

66070501055 ศิริประภา ชินวงษ์ทัน

เสนอ

ผศ. สนั่น สระแก้ว

ภาคเรียนที่ 1 ปีการศึกษา 2567

รายวิชา CPE222 อิเล็กทรอนิกส์ดิจิทัลและการออกแบบวงจรเชิงตรรกะ

(Digital Electronics and Logic Design)

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

หน้าที่ความรับผิดชอบ

66070501004 กัญญวรรณ ทยานิพันธ์ - Game Mechanics

66070501005 กัลยา สุทธิปัญญา - Block Design and Rotation

66070501009 จิรพรรณ ศักดิ์พิสุทธิพงศ์ - Game Mechanics

66070501034 เปมิกา จงขวัญยืน - VGA Controlle

66070501055 ศิริประภา ชินวงษ์ทัน - Game Mechanics

วัตถุประสงค์

1. เพื่อบูรณาการความรู้ที่ได้รับจากรายวิชา CPE222 อิเล็กทรอนิกส์ดิจิทัลและการออกแบบวงจรเชิงตรรกะ มาประยุกต์ใช้ในการออกแบบและสร้างวงจรเชิงตรรกะ เพื่อใช้งานเป็นอุปกรณ์สำหรับสร้างเกม Tetris
2. เพื่อพัฒนาเกม Tetris โดยใช้ FPGA ในการประมวลผลและควบคุมการทำงานของเกม
3. เพื่อออกแบบและสร้าง Game Engine ที่ทำหน้าที่ควบคุมการเคลื่อนที่, ตรวจจับการชน, การลบแถว และคำนวณคะแนน
4. เพื่อพัฒนาระบบ VGA Signal Generator สำหรับแสดงผลเกม Tetris บนจอ VGA Monitor ด้วยความละเอียดมาตรฐาน 640x480 60Hz
5. เพื่อเรียนรู้และฝึกฝนการเขียนโค้ดลอจิกดิจิทัลด้วย Verilog สำหรับใช้งานบน FPGA

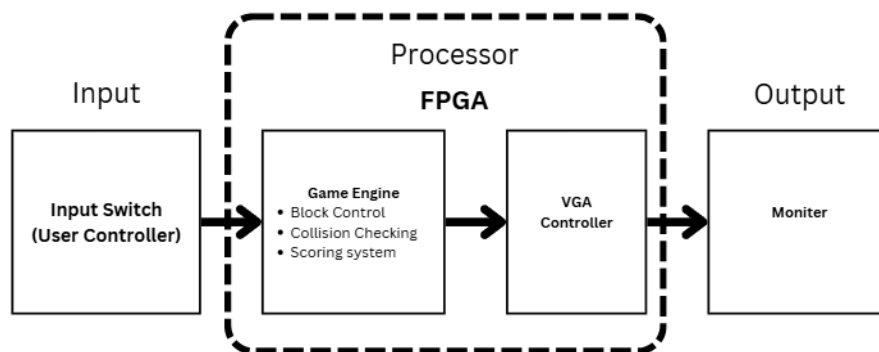
ขอบเขต

1. **การแสดงผล:** แสดงกริด 10x20 บนจอ VGA (ความละเอียด 640x480) แสดงคะแนนบน 7-Segment Display สูงสุด 9999 คะแนน
2. **Tetromino 7 รูปแบบ:** รูปแบบบล็อก: I, O, T, S, Z, J, และ L รองรับการหมุนตามเข็มนาฬิกา

3. การควบคุมเกม: ใช้ปุ่ม 4 ปุ่มบน Basys3 สำหรับการเลื่อนซ้าย, เลื่อนขวา, หมุนบล็อก, เร่งความเร็ว, และสวิตช์เพื่อรีเซ็ต
4. ฟังก์ชันเกมพื้นฐาน:
 - 4.1. ตรวจจบการชน
 - 4.2. คะแนนเพิ่มขึ้นเมื่อเล่นนานขึ้น โดยจับเวลาจาก Clock อีกทั้งลบแถวเมื่อเต็มและเพิ่มคะแนน
 - 4.3. Game Over เมื่อ ไม่มีพื้นที่สำหรับ Tetromino ใหม่
5. ข้อจำกัด: ใช้ทรัพยากร FPGA บน Basys3 ทั้งหมด และไม่รองรับพีเออร์ชั้นสูง

ส่วนประกอบวงจร

การพัฒนาเกมเตตริสด้วย FPGA ได้มีการแจกแจงส่วนประกอบวงจร



1. Input Switch (Controller)

อุปกรณ์รับคำสั่งจากผู้เล่นเพื่อควบคุมการเคลื่อนที่และการหมุนของบล็อกในเกม โดยปุ่มซ้าย (BTN_LEFT) ใช้สำหรับเลื่อนบล็อกไปทางซ้าย ปุ่มขวา (BTN_RIGHT) ใช้สำหรับเลื่อนบล็อกไปทางขวา เพื่อช่วยผู้เล่นในการจัดตำแหน่งบล็อกให้ตรงกับจุดที่ต้องการบนกระดาน นอกจากนี้ ปุ่มล่าง (BTN_DOWN) ถูกตั้งค่าให้ทำหน้าที่เร่งความเร็วในการตกของบล็อก (Soft Drop) ซึ่งช่วยลดระยะเวลาในการรอบล็อกตกลงถึงพื้น และสุดท้าย ปุ่มบน (BTN_UP) ใช้สำหรับหมุนบล็อกเพื่อปรับรูปร่างของบล็อกให้เหมาะสมกับพื้นที่ว่างบนกระดาน

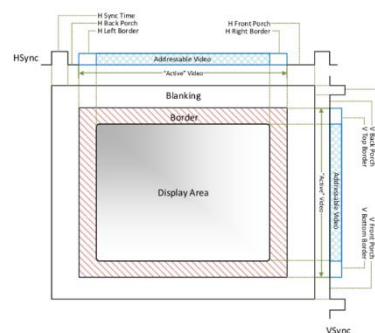
สำหรับการรีเซ็ตเกมหรือเริ่มต้นใหม่ Slide Switch ตัวแรก (S0) บน Basys 3 ถูกกำหนดให้ทำหน้าที่เป็น ตัวควบคุมการรีเซ็ต เมื่อผู้เล่นต้องการเริ่มเกมใหม่หรือรีเซ็ตสถานะของเกมให้กลับสู่จุดเริ่มต้น เพียงเลื่อนสวิตช์ S0 ระบบจะทำการรีเซ็ตสถานะทั้งหมด เช่น เคลียร์กระดานเกม รีเซ็ตคะแนน และรีโหลดบล็อกเริ่มต้นใหม่

2. Field Programmable Gate Array (FPGA)

FPGA เป็นอุปกรณ์แบบวงจรรวม (Integrated Circuit) แบบสามารถเขียนโปรแกรมเพื่อควบคุมได้ โดย ภายใน FPGA จะประกอบไปด้วยเกตขั้นพื้นฐาน เช่น AND gate, OR gate, NAND gate, Flipflops และ Block Memory ที่จะสามารถนำมาต่อกันเป็นวงจรดิจิทัลแบบต่าง ๆ ได้ ยกตัวอย่างเช่น วงจรนับ, วงจรบวก, วงจร หน่วยความจำ เป็นต้น FPGA นั้นนิยมนำไปใช้งานในด้านต่าง ๆ เช่น การทดลองเพื่อการศึกษา, การทำต้นแบบ ดิจิทัลไอซีก่อนการสร้างจริง, การทำตัวเร่งประสิทธิภาพการคำนวณเฉพาะทาง, วงจรสำหรับประมวลผล สัญญาณดิจิทัล เป็นต้น ในการเขียนโปรแกรมเพื่อควบคุม FPGA จะใช้ภาษาคำอธิบายฮาร์ดแวร์ (Hardware Description Language) เช่น VHDL หรือ Verilog หลังจากนั้นจะใช้ซอฟต์แวร์เพื่อสร้างไฟล์สำหรับควบคุม FPGA (FPGA configuration file) ในรูปแบบของ bitstream เพื่อใช้ในการโปรแกรม FPGA

3. Monitor display ที่มี VGA

“Video Graphics Array” หรือ VGA ซึ่งเป็นมาตรฐานการแสดงผลทางดิจิทัลที่ใช้ในคอมพิวเตอร์ และอุปกรณ์ที่เชื่อมต่อกับหน้าจอ รูปแบบนี้มีความสามารถในการส่งข้อมูลภาพและสัญญาณเสียงผ่าน สายสัญญาณแบบอนาล็อก (analog) หรือบางแบบดิจิทัล (digital) ซึ่งให้ความละเอียดภาพปานกลางถึงสูงสุด 640x480 พิกเซลในสี 16 บิต และรองรับการแสดงสีจำนวนมาก โดยทั่วไป VGA ถูกใช้กับการแสดงผลทางดิจิทัลในคอมพิวเตอร์ในหน้าจอ คอมพิวเตอร์ทั่วไป โดยในโปรเจกต์ใช้ทั้งหมด 6 ขา ประกอบไปด้วย สายสัญญาณสี 3 เส้น ได้แก่ RGB และสายสัญญาณ ควบคุม 2 เส้น ได้แก่ H-Sync (Horizontal-sync) และ V-Sync (Vertical-sync) และสาย ground 1 เส้น



ในรายงานนี้ขอยกตัวอย่างหน้าจอแสดงผลแบบ CRT เพื่อใช้ในการ อธิบายหลักการทำงานของสัญญาณ VGA โดยหน้าจอชนิดนี้จะทำการรับสัญญาณ Horizontal-sync และ Vertical-sync เพื่อทำหน้าที่ควบคุมการเบนลำอิเล็กตรอนให้กวาดไปยังตำแหน่งต่าง ๆ ของหน้าจอ เมื่อลำ อิเล็กตรอนไปกระทบกับหน้าจอทำให้ฟอสเฟอร์ที่เคลือบหน้าจอได้รับการกระตุ้นและคายพลังงานออกมาใน รูปของแสง ลำอิเล็กตรอนเหล่านี้จะถูกให้กำเนิดด้วยปืนอิเล็กตรอนสามกระบอกในหลอดภาพ CRT ที่จะรับ

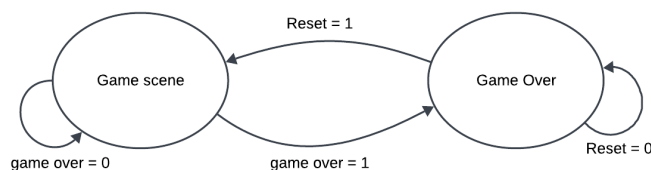
สัญญาณ RGB แบบ analog amplitude modulation โดยปริมาณอิเล็กทรอนิกส์ที่ป้อนแต่ละกระบอกยังออกมาจะแปรผันตรงกับ amplitude ของคลื่น RGB

หลักการทำงาน

1. ระบบรูปแบบการเล่นเกม (Game Mechanics)

Tetris เป็นเกมแนวพัชเชิลที่มีรูปแบบการเล่นที่เรียบง่ายแต่ทรงพลัง โดยสนามเกม (Playfield) จะเป็นพื้นที่ตารางสี่เหลี่ยมแนวตั้ง ซึ่งมีขนาด 10 ช่องในแนวนอน และ 20 ช่องในแนวตั้ง ผู้เล่นมีหน้าที่จัดเรียงบล็อกที่เรียกว่า Tetrominoes ซึ่งประกอบด้วย 7 รูปแบบ ได้แก่ I, O, T, S, Z, J, และ L แต่ละชิ้นส่วนประกอบด้วยบล็อกย่อย 4 บล็อกที่เชื่อมต่อกันในรูปแบบเฉพาะ และจะถูกปล่อยลงมาจากด้านบนของสนาม ผู้เล่นต้องวางชิ้นส่วนเหล่านี้ให้เต็มแถวแนวนอนเพื่อเคลียร์แถวนั้นออก หากชิ้นส่วนบล็อกซ้อนกันจนเกินขอบบนของสนาม เกมจะสิ้นสุดลง การควบคุมในเกม Tetris มีความคล่องตัวและเป็นหัวใจสำคัญของการเล่น ผู้เล่นสามารถเลื่อนชิ้นส่วนไปทางซ้ายหรือขวาเพื่อปรับตำแหน่ง และสามารถเร่งความเร็วการตกลงของบล็อกได้ด้วยการกดปุ่มให้ชิ้นส่วนตกเร็วขึ้น (Soft Drop) กฎการเล่นพื้นฐานมุ่งเน้นไปที่การเคลียร์แถว โดยเมื่อแถวแนวนอนใดในสนามถูกเติมเต็ม (ไม่มีช่องว่าง) แถวนั้นจะถูกลบออก และพื้นที่ว่างด้านบนจะเลื่อนลงมาแทนที่ การลบแถวไม่เพียงแต่ทำให้สนามโล่งขึ้น แต่ยังช่วยเพิ่มคะแนนให้ผู้เล่นอีกด้วย ในระบบคะแนนแบบพื้นฐาน ผู้เล่นจะได้รับคะแนน 10 คะแนนสำหรับการลบหนึ่งแถว และไม่มีคะแนนพิเศษเพิ่มเติม การจัดการตำแหน่งและการวางแผนเพื่อสร้างแถวให้ได้อย่างต่อเนื่องจึงเป็นหัวใจสำคัญของการเพิ่มคะแนนและการอยู่รอดในเกม

2. ระบบภาพรวมการทำงาน



2.1. Game scene state เป็น state เริ่มต้นและครอบคลุมการทำงานทั้งหมดของเกม Tetris ใน state นี้ เมื่อผู้เล่นเปิดสวิตช์ เกมจะเริ่มสุ่มบล็อกและนับคะแนนของผู้เล่น ผู้เล่นสามารถเคลื่อนย้ายบล็อกและเร่งความเร็วการร่วงของบล็อกได้ เมื่อผู้เล่นอยู่ในเงื่อนไข game over จะทำให้ game over มีค่าเป็น 1 เป็นสัญญาณเปลี่ยน State เข้าสู่ Game over state

2.2. Game over state เกมจะแสดงผลผ่านหน้าจอและบอร์ด Basys 3 ผู้เล่นจะไม่สามารถทำอะไรได้ ถือว่าเป็นการจบเกม แต่ผู้เล่นสามารถเริ่มเล่นเกมใหม่ได้โดยการเปิดสวิทช์อีกครั้ง

3. ระบบฮาร์ดแวร์ (Hardware)

3.1. VGA Controller

ทำหน้าที่ให้กำเนิดสัญญาณนาฬิกาเพื่อควบคุมหน้าจอ VGA และสัญญาณ gameclk จาก video active เพื่อเป็น clk สำหรับวงนับในโมดูลต่างๆ อีกทั้งส่งตำแหน่งปัจจุบันของบนหน้าจอไปยังโมดูลที่เกี่ยวข้องภายในวงจร เพื่อให้โมดูลดังกล่าวส่งข้อมูล RGB ที่ถูกต้องตรงกันกับตำแหน่งที่หน้าจอกำลังสแกน

ผู้จัดทำได้สร้างวงจรเพื่อให้กำเนิดสัญญาณ H-sync และ V-sync ด้วยการใช้วงจร counter 2 ชุด เพื่อควบคุมสัญญาณ H-sync และ V-sync โดย counter ที่ควบคุม H-sync หรือ horCnt จะได้รับ clk ที่เท่ากับ pixel freq. ที่ได้มาจากการนำเอาสัญญาณนาฬิกาที่ 100 MHz มาผ่าน vgaClkDivider ซึ่งเป็น ip-core DCM เพื่อให้ได้ความถี่ที่ 25.175 MHz เพื่อนำไปใช้งานเป็น pixel frequency โดย horCnt เป็น counter ที่นับจาก 0-799 และเมื่อมีค่าอยู่ในช่วง 656-751 จะทำการให้กำเนิดสัญญาณ horizontal sync pulse โดยทุก ๆ ครั้งที่ horCnt เกิดการ reset (ทำงานครบ 1 บรรทัด) วงจรจะทำการเพิ่มค่าของ counter ที่ควบคุม V-sync หรือ verCnt โดยเมื่อ verCnt นับถึงช่วง 490-491 จะทำการให้กำเนิดสัญญาณ vertical sync pulse เมื่อนับถึง 525 จะ reset กลับไปที่ 0 (เมื่อ vertical counter เกิดการ reset หมายความว่าจอภาพแสดงผลภาพครบ 1 เฟรม)

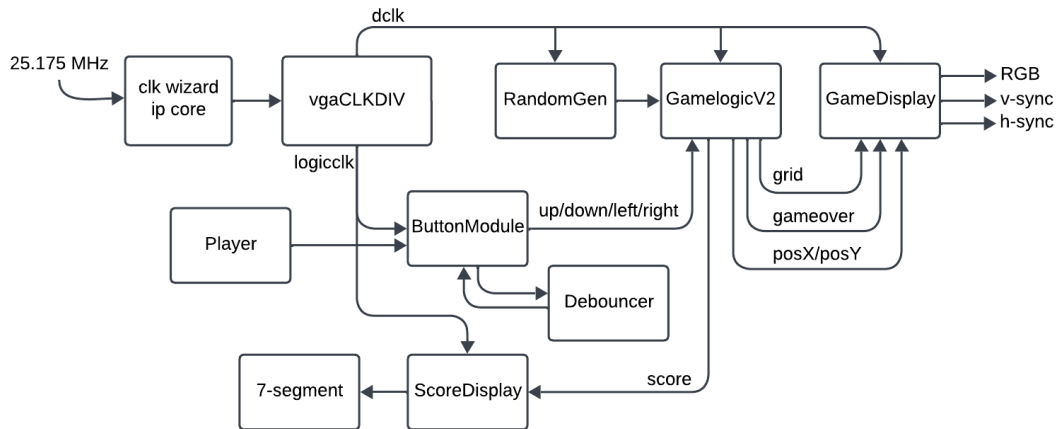
ส่วนสัญญาณ RGB ที่ทำหน้าที่ในการควบคุมสี ใช้ระบบการผสมแม่สีของแสงแบบสีบิตต่อช่อง (4 bits per channel) รวมกันทั้งหมด 12 บิต สำหรับควบคุมทั้ง 3 สี ทำให้สามารถแสดงผลสีได้ทั้งหมด 4,096 สี โดยสัญญาณบิตเหล่านี้ถูกส่งผ่าน Resistor Network ที่ทำหน้าที่เป็นวงจร Digital to Analog Converter เพื่อแปลงสัญญาณสีแบบดิจิทัลให้เป็นสัญญาณอนาล็อกที่อยู่ในช่วง 0 – 0.7VP ที่สามารถใช้งานเพื่อควบคุมสีบนจอแสดงผลแบบ VGA ได้

3.2. 7 - Segment Controller

คะแนนของผู้เล่นจะถูกแสดงผลแบบเรียลไทม์บนหน้าจอ 7-segment display ซึ่งประกอบด้วย 4 หลัก เพื่อรองรับคะแนนที่เพิ่มขึ้นเรื่อย ๆ ขณะเล่นเกม การแสดงผลดังกล่าวใช้เทคนิค Multiplexin สำหรับระบบการนับคะแนน คะแนนเริ่มต้นของผู้เล่นจะถูกตั้งค่าเป็นศูนย์ และเพิ่มขึ้นเรื่อย ๆ ตามสัญญาณนาฬิกาของระบบ (clk) เพื่อสะท้อนถึงเวลาและความคืบหน้าของเกม ในกรณีที่ผู้เล่นสามารถเคลียร์แถวในเกมได้ ระบบจะตรวจจับเหตุการณ์นี้ผ่านสัญญาณอินพุต และเพิ่มคะแนนเพิ่มเติมเข้าไปในตัวแปรคะแนน เช่น 10

คะแนนต่อแถว การเปลี่ยนแปลงของคะแนนดังกล่าวจะถูกส่งไปยัง 7-Segment Controller เพื่ออัปเดตค่าที่แสดงบนหน้าจอโดยอัตโนมัติ

4. ระบบเกมเอนจิน (Game Engine)



- 4.1. **vgaCLKDIV** ทำหน้าที่ในการแบ่งและส่งสัญญาณไปยังโมดูลเพื่อควบคุมการทำงานของเกม
- 4.2. **RandomGen** ใช้ Pseudorandom โดยวิธี Linear Feedback Shift Register (LFSR) เลือก Tetromino บล็อกต่อไป โดยเมื่อสุ่มแล้วจะส่งค่าไปยัง โมดูล GamelogicV2
- 4.3. **GamelogicV2** ควบคุมการทำงานของเกมตั้งแต่การสร้างบล็อก การเคลื่อนที่ การตรวจจับการชน ไปจนถึงการจัดการสถานะของเกม เช่น สถานะเกมโอเวอร์ อีกทั้งมีการสร้างกริด ตำแหน่งของกริด ตำแหน่งของบล็อก
- 4.4. **GameDisplay** ในการแสดงผลโมดูลจะควบคุมค่ากริด ตำแหน่งของกริด ตำแหน่งของบล็อกและสถานะเกมโอเวอร์ เพื่อนำไปแสดงผลบน VGA
- 4.5. **ButtonModule** รับสัญญาณจากปุ่มกด (up / down / left / right) โดยจะรับเป็นสัญญาณ btnU, btnD, btnL, btnR และส่งไปที่โมดูล Debouncer เพื่อให้สัญญาณเสถียร
- 4.6. **Debouncer** รับมือสัญญาณรบกวน (debounce) ที่เกิดขึ้นจากการกดปุ่ม ซึ่งในฮาร์ดแวร์จริงเมื่อกดหรือปล่อยปุ่มจะเกิดการสั้นของหน้าสัมผัส ส่งผลให้สัญญาณไม่เสถียรในช่วงเวลาสั้น ๆ
- 4.7. **ScoreDisplay** ควบคุม 4-digits 7-segment display เพื่อแสดงคะแนนที่ได้รับมาจากโมดูล GamelogicV2 ทำให้แสดงตัวเลขในแต่ละหลักได้ด้วยการเปิดทีละหลักในแต่ละรอบของ clock

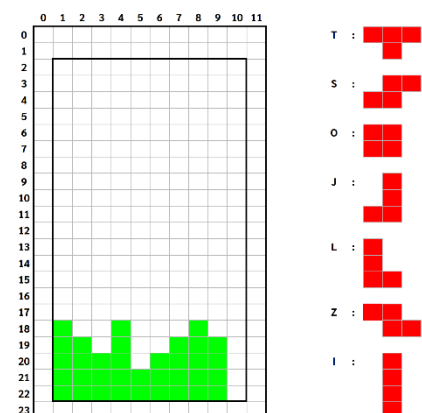
5. ระบบกราฟฟิค (Graphic Design)

5.1. กริด (Grid) เป็นพื้นที่เล่นเกมที่มีขนาด 10 ช่องแนวนอน (ความกว้าง) และ 20 ช่องแนวตั้ง (ความสูง) โดยในโค้ดมีการใช้ตัวแปร grid ขนาด 288 บิต เพื่อเก็บสถานะของแต่ละช่องในกริด ซึ่งสถานะนี้บ่งบอกว่าช่องนั้นเป็น ช่องว่าง หรือ ช่องที่มีบล็อก ในการแสดงผลกริด มีการตรวจสอบค่าพิกัดแนวนอน (hc) และแนวตั้ง (vc) เพื่อระบุว่าช่องปัจจุบันควรแสดงสีอะไร หากช่องดังกล่าวตรงกับตำแหน่งของส่วนใดส่วนหนึ่งในบล็อกปัจจุบัน (ตรวจสอบผ่านค่าพิกัด posX และ posY) จะถูกเปลี่ยนสีให้แสดงถึงบล็อกนั้น เช่น สีแดง หากช่องนั้นถูกรอบครองโดยบล็อกที่ค้างอยู่ในกริด (ตรวจสอบสถานะในตัวแปร grid เช่น `grid[2*12+1]`) จะเปลี่ยนเป็น สีเขียว ส่วนช่องว่างจะแสดงเป็นสีขาว

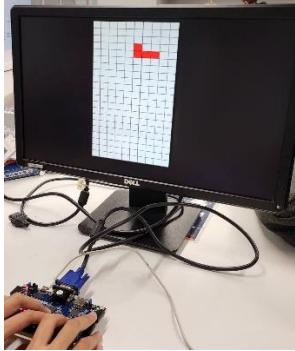
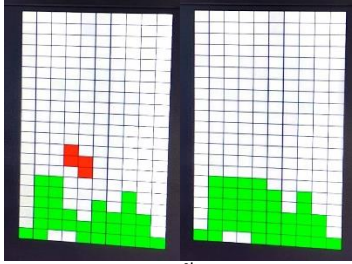
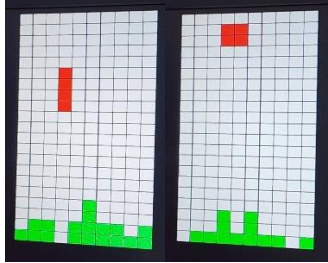
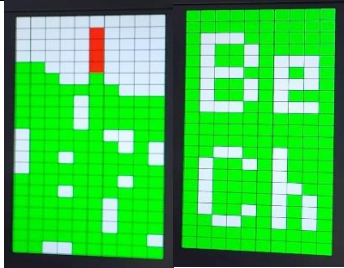
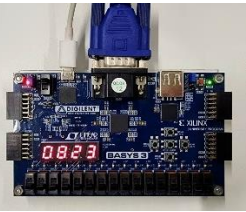
5.2. บล็อก (Block) บล็อกในเกม Tetris มีทั้งหมด 7 รูปแบบ ได้แก่ I, O, T, S, Z, J, และ L การกำหนดพิกัดเริ่มต้นของบล็อกถูกจัดการด้วยคำสั่ง `case (nextBlockRand)` เพื่อสุ่มเลือกบล็อกใหม่ และกำหนดค่าพิกัดเริ่มต้นในตัวแปร `xNreg` และ `yNreg` ($N = 1$ ถึง 4) ตัวอย่างบล็อก I จะเริ่มต้นในแนวอนพิกัด: (4,1), (5,1), (6,1), (7,1) และบล็อก O จะเริ่มต้นเป็นรูปสี่เหลี่ยมจัตุรัสพิกัด: (5,0), (6,0), (5,1), (6,1) เมื่อบล็อกถูกสร้างขึ้น จะมีการอัปเดตค่าพิกัดของแต่ละส่วนในบล็อกผ่านตัวแปรดังกล่าว เพื่อใช้ในการแสดงผลและคำนวณตำแหน่งในระหว่างการเล่น

5.3. การหมุนบล็อก (Block Rotation) การหมุนบล็อกเป็นฟังก์ชันสำคัญที่ช่วยให้ผู้เล่นสามารถจัดวางบล็อกให้เหมาะสมกับพื้นที่ว่างในกริด โดยบล็อกแต่ละแบบมีรูปแบบการหมุนเฉพาะตัว และมีการจัดการการหมุนบล็อกผ่านคำสั่ง `TRYROTATE` ซึ่งใช้การตรวจสอบชนิดของบล็อก (`currentBlock`) และสถานะการหมุน (`rotState`)

5.4. การแสดงผลสี ในแต่ละช่องของกริดกำหนดด้วย red, green, และ blue ซึ่งเป็นค่าสี RGB โดยช่องที่ว่างแสดงสีขาว ช่องที่มีบล็อกปัจจุบันแสดงสีแดง และช่องที่มีบล็อกที่ติดอยู่ในกริดแสดงสีเขียว



ผลลัพธ์การทำงาน

ภาพผลลัพธ์การทำงาน	คำอธิบาย
 <p data-bbox="342 695 657 730">สถิติการทำงานขณะเริ่มเกม</p>	<p data-bbox="824 436 1417 653">เมื่อเปิดสวิตช์ Reset เกมจะเริ่มรับ Input จากผู้เล่น และบล็อกจะเริ่มเคลื่อนไหว โดยทุก ๆ ครั้งที่บล็อกเคลื่อนลงมาจะบวกคะแนนเพิ่มที่ละหนึ่งคะแนนแสดงบนบอร์ด</p>
 <p data-bbox="240 1037 760 1073">ตัวอย่างบล็อกที่ชนกับพื้นจะเปลี่ยนเป็นสีเขียว</p>	<p data-bbox="824 814 1386 1031">กดปุ่มเลื่อนซ้าย - ขวา เพื่อปรับตำแหน่งของบล็อก กดปุ่มขึ้นเพื่อหมุนบล็อก และกดปุ่มลงเพื่อเร่งความเร็วการตกของบล็อก เพื่อบล็อกชนกับพื้นจะเปลี่ยนเป็นสีเขียว</p>
 <p data-bbox="282 1356 717 1392">บล็อก I เดิมแถวแรกจนเต็มและหายไป</p>	<p data-bbox="824 1163 1398 1325">เมื่อบล็อกแถวใด ๆ ถูกเติมจนเต็มแถวนั้นจะหายไป พร้อมบวกคะแนนอีก 10 คะแนน และบล็อกที่เหลือนในแถบนจะเลื่อนลงมาเติมแถวที่หายไป</p>
  <p data-bbox="204 1682 797 1780">บล็อกต่อไปซ้อนต่อบล็อก I (บล็อกสีแดง) ทำให้ชนกรอบด้านบนจึงเข้าสู่หน้าจอ Game Over</p>	<p data-bbox="824 1486 1414 1703">กรณีที่บล็อกถูกเติมจนเลยเพดานด้านบน เกมจะสิ้นสุดและแสดงหน้าจอ “Be Ch” ซึ่งเป็นตัวอักษรย่อชื่อกลุ่มคณะผู้จัดทำ ทั้งนี้ที่เกมจบลงคะแนนทั้งหมดจะแสดงบนบอร์ด</p>

ปัญหา อุปสรรคที่พบ และ แนวทางการแก้ไข

จากกระบวนการพัฒนาโครงการ คณะผู้จัดทำพบปัญหาสำคัญหลายประการ ซึ่งได้ดำเนินการแก้ไขอย่างเป็นระบบ ดังต่อไปนี้

1) ปัญหาการแสดงผลบนหน้าจอ VGA

สาเหตุ: การใช้ pixel clock ที่ 25 MHz ซึ่งไม่ตรงตามมาตรฐาน

แนวทางแก้ไข: ปรับเปลี่ยนความถี่เป็น 25.175 MHz เพื่อให้สามารถแสดงผลได้อย่างถูกต้อง

2) ข้อจำกัดการใช้งาน 2D array ด้วย Vivado

สาเหตุ: ข้อจำกัดทางซอฟต์แวร์ในเวอร์ชัน 2024.1

แนวทางแก้ไข: ใช้วิธีการระบุ index ด้วย offset

3) ความซับซ้อนคำสั่งแสดงผลบนหน้าจอ VGA

สาเหตุ: เนื่องจากการแสดงผลจำเป็นต้องตรวจสอบเงื่อนไขมากถึง 200 เงื่อนไข

ข้อเสนอแนะ: ควรแยกเงื่อนไขออกเป็นฟังก์ชันย่อย และใช้ lookup table หรือ state machine เพื่อจัดการเงื่อนไขอย่างมีประสิทธิภาพ

4) ข้อจำกัดด้านทรัพยากรของ Logic Cells

สาเหตุ: โมดูล GameLogicV2 ใช้ logic cells จำนวน 11,923 เซลล์ แต่เนื่องจาก Basys3 board มีความสามารถรองรับ 33,280 logic cells

ข้อเสนอแนะ: ควรพิจารณาการเพิ่มประสิทธิภาพการ optimization โดยใช้เทคนิค resource sharing

เพื่อลดการใช้ logic elements ที่ไม่จำเป็น และ

ออกแบบ hardware description code ที่กระชับเพื่อ

ลดการใช้ทรัพยากร Basys3 board

5) ข้อจำกัดด้านการใช้ random function

สาเหตุ: Basys3 board ไม่มีฟังก์ชันสุ่มในตัว

แนวทางแก้ไข: พัฒนาระบบสุ่มแบบเทียม

(Pseudorandom) โดยใช้เทคนิค Linear Feedback Shift Register (LFSR)

6) ปัญหาจากการใช้คีย์บอร์ด

สาเหตุ: ความล่าช้าจากโปรโตคอลการสื่อสารและมีความจำเป็นในการพัฒนาไดรเวอร์เพิ่มเติม

แนวทางแก้ไข: เปลี่ยนมาใช้ปุ่มกดบน Basys3 Board เนื่องจากมีความแม่นยำ รวดเร็ว และลดความซับซ้อนในการออกแบบ

7) ระยะเวลาในกระบวนการพัฒนา

สาเหตุ: การ Synthesis, Implementation และ

Generate Bitstream ใช้เวลาประมาณ 5 - 10 นาที

แนวทางแก้ไข: ดำเนินการ debug หลายจุดพร้อมกันและจัดการ process ที่เดียว

แนวทางการพัฒนาโครงการในอนาคต

คณะผู้จัดทำได้กำหนดแนวทางการพัฒนาเกม Tetris ด้วย FPGA เพื่อเพิ่มประสิทธิภาพและความน่าสนใจของเกม โดยมีขอบเขตการพัฒนาดังนี้

1) การเพิ่มประสิทธิภาพการแสดงผล

- a) พัฒนาระบบแสดง Tetrimino ตัวถัดไปบนหน้าจอ
- b) ออกแบบการแสดงผลคะแนนที่มีความชัดเจนและสวยงาม

2) การยกระดับกลไกเกม

- a) พัฒนาระบบ score combo เพื่อเพิ่มความท้าทายให้กับผู้เล่น
- b) ออกแบบกลไกการเพิ่มความเร็วของ Tetrimino ตามระยะเวลาการเล่น

- c) สร้างระบบปรับระดับความยากของเกม

3) การเพิ่มคุณภาพทางกราฟิก

- a) พัฒนา Tetrimino ให้มีสีสันทันที่แตกต่างกัน
- b) เพิ่ม texture เพื่อสร้างมิติและความลึกให้กับชิ้นเกม

4) การพัฒนาระบบควบคุม

- a) ออกแบบระบบควบคุมผ่าน joystick เพื่อเพิ่มความสะดวกในการเล่น
- b) พัฒนาระบบเสียงประกอบเกมเพื่อเพิ่มประสบการณ์ที่น่าตื่นเต้น

สรุป

จากการดำเนินโครงการพัฒนาเกมเตตริสบนแพลตฟอร์ม FPGA พบว่าโครงการประสบความสำเร็จในระดับที่น่าพึงพอใจ โดยสามารถออกแบบและพัฒนาเกมเตตริสที่แสดงผลผ่านจอภาพ VGA monitor และแสดงคะแนนบน 7-segment display บนบอร์ด Digilent Basys3 FPGA Development Board แม้ว่ามีข้อจำกัดบางประการ อาทิ ยังไม่สามารถแสดงคะแนนบนหน้าจอ monitor และการปรับเปลี่ยนการรับ input จาก keyboard เป็นปุ่มกดบน Basys3 Board เพื่อเพิ่มความแม่นยำ ลดความล่าช้า และลดความซับซ้อนในการออกแบบ คณะผู้จัดทำได้นำความรู้จากรายวิชา CPE222 Digital Electronics and Logic Design มาประยุกต์ใช้ในการพัฒนาเกมเตตริสสำหรับ Xilinx Artix-7 FPGA อย่างมีประสิทธิภาพ

ตลอดกระบวนการพัฒนา คณะผู้จัดทำได้รับประสบการณ์และความรู้ในด้านต่าง ๆ โดยเฉพาะอย่างยิ่งการออกแบบเกม การสร้าง IP-core เพื่อควบคุมสัญญาณนาฬิกา เทคนิคการใช้ VRAM เป็น Buffer และการสร้างสัญญาณเพื่อขับหน้าจอแบบ VGA ซึ่งล้วนเป็นทักษะที่มีความสำคัญในการพัฒนาระบบดิจิทัลที่มีความซับซ้อน