Mark Mchedlishvili
Eric Mendoza-Connor

# Tetris

## Abstract

The circuit is a partial simulation of the game Tetris. Pieces are generated pseudo-randomly and fall at a constant rate. The user may move one piece at a time horizontally or rotate it. When the piece lands on either the bottom row or another piece, it is locked in place and the user controls a newly generated piece at the top of the board. The game continues until the user is unable to place any more pieces. There are pause and reset switches available.

## Design Narrative

The four push buttons are attached to 555 timers in a debounce configuration such that pressing the button will output a 18ms high voltage pulse. Two of the buttons control the horizontal movement of the tetris piece, while the other two control the rotation of the piece. The piece type is generated pseudo-randomly by a linear feedback shift register. The row that the piece is on is controlled by a counter connected to an astable timer. A programmable chip uses the type of piece and its rotation to generate a 4x4 grid containing the correct piece, one row at a time. The clock of this chip is input from another programmable chip which takes that row that the piece is on as an input and outputs either the 4x4 grid or a row of 0's. This chip also generates the clock for the counters controlling rotation and horizontal movement. A third programmable chip uses this output and the horizontal position to shift the grid and output the final 8x8 grid which displays the falling chip. An SRAM chip is initialized to contain an 8x8 grid with all the edges other than the top being high and everything else low. The SRAM represents all the static components of the game. Each value in the SRAM chip is compared to

the value generated by the programmable chips. If collision is detected (two corresponding cells are high), each cell in the static grid is connected through an OR gate to the generated moving piece grid. These values are temporarily stored in an octal flip flop, and then written to the SRAM chip, thereby "adding" the falling piece to the static piece grid. Simultaneously, a new falling piece is generated and reset signals are sent to the counters in order to position the new piece at the top of the grid. Eight octal flip flops store all values of the grid which is displayed to the user. Each flip flop uses inputs of both SRAM chips through OR gates in order to write column values and a demultiplexer selects the chip containing the corresponding row being written to. An 8x8 array of LEDs is soldered onto a perfboard and connected to the corresponding values of the octal flip flops.

Programmable Chip

For our project we implemented the use of a PLD (Programmable Logic Device), specifically the GAL 22V10 chip. PLDs are chips that can be connected to a computer and programmed via various languages, the most popular being the language CUPL that, on Windows computers, is called WinCupl and is by the Atmel company. Each pin in the chip is assigned either an output or an input value. Various logical programming can be done using the syntax and tools described in the WinCupl user's manual.

We implemented the GAL 22V10 chips by using them with the most complicated part of our project: generating the values of the Tetris grid by rotating and shifting the pieces. The chips take inputs for the row and column location of a piece, the rotation state of that piece, and the type of Tetris piece that the chip was addressing, then returns in a sequence the address of each row and each of the 8 cells on that row's value as 8 individual bits. There are a significant

amount of conditions to consider when outputting the Tetris grid through the chip, and we ran into problems trying to address all those conditions while satisfying the limited capacity of the functions the chips can do.  We addressed the problem of being unable to implement all the functionality on a single chip by diving it amongst three chips.

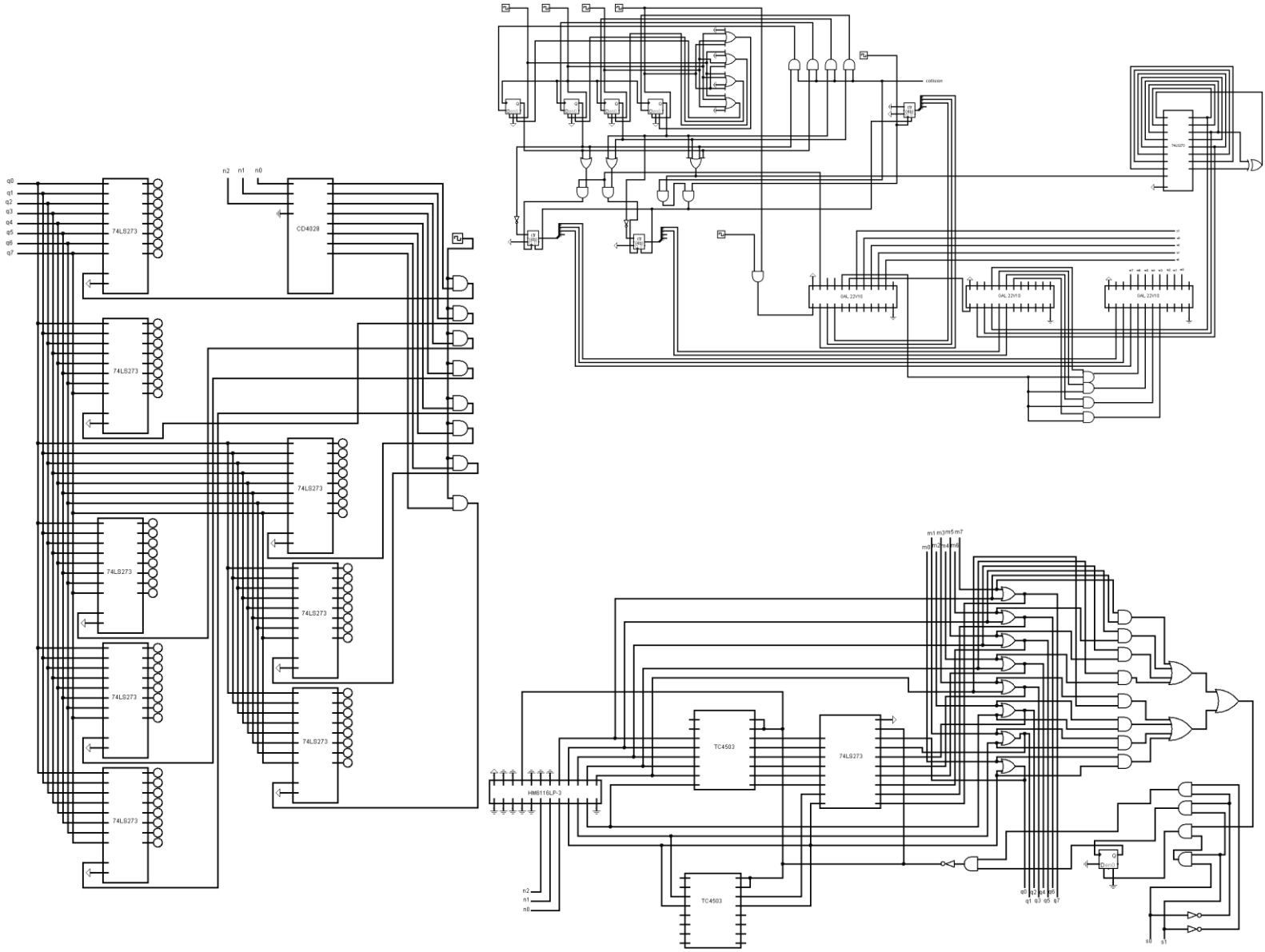
Inventory

| Component | Number | Quantity | Functionality |
|---|---|---|---|
| Push Button | N/A | 4 | Movement<br>Rotation |
| Timer | 555 | 6 | Button Debounce<br>Falling piece<br>Programmable Array |
| D Flip-flop | 4013 | 4 | Stores user input |
| Quad 2-input OR | 4071 | 5 | Combine falling piece with static piece grid<br>Input circuit<br>Collision |
| Quad 2-input XOR | 4070 | 1 | Random generator |
| Hex Buffer | 4503 | 2 | SRAM I/O |
| SRAM 2kx8 | 6116LP-3 | 1 | Storage of static piece grid |
| Quad 2-input AND | 4081 | 5 | Input circuit<br>Collision |
| Dual 4-input AND | 4082 | 1 | Collision |
| Dual 4-input OR | 4072 | 2 | Input circuit |
| Dual 4-input NOR | 4002 | 1 | Fall collision |
| Octal D Flip-flop | 74LS273 | 10 | Buffer to LED output<br>Temporary row storage<br>Random generator |
| Hex Inverter | 4069 | 1 | Write enable |
| 4-bit counter | 4516 | 3 | Rotation<br>Vertical position<br>Horizontal position |
| 8-bit binary decoder | 4514 | 1 | Output |
| Programmable Array | GAL22v10 | 3 | Memory management<br>Piece rotation |
| LED | N/A | 64 | Output Grid |
| Breadboard | N/A | 11 | N/A |

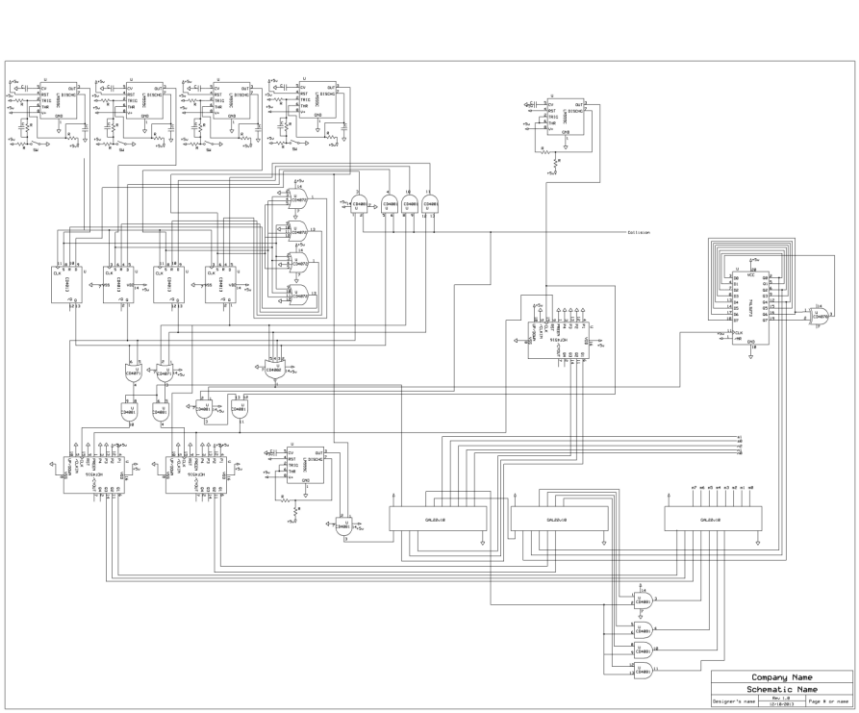
Issues Encountered

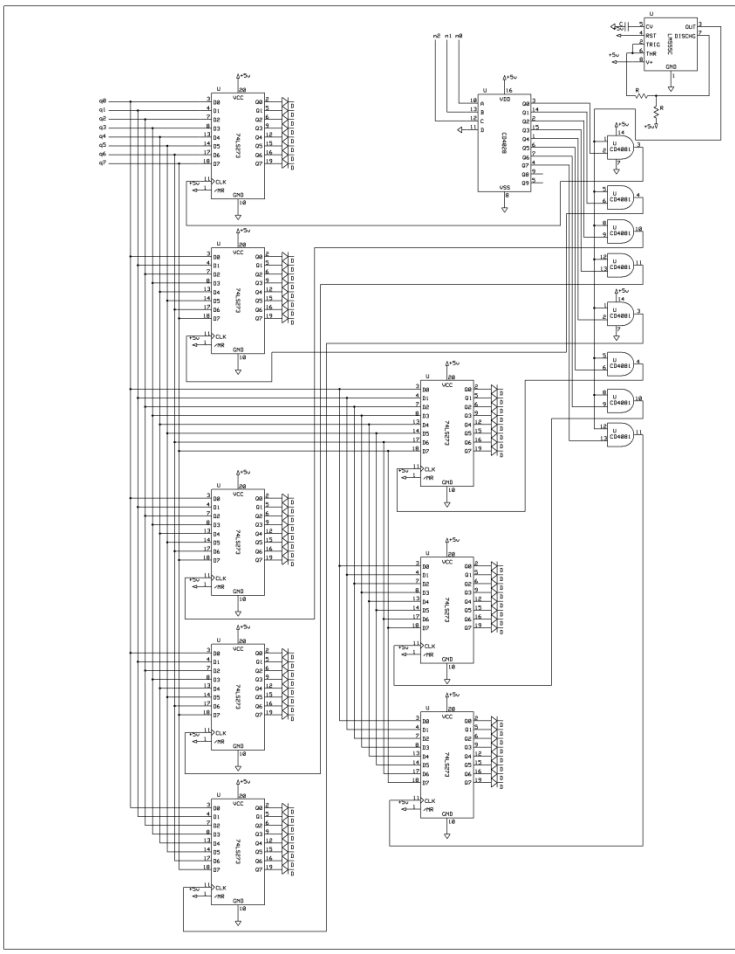
- Rotation of Tetris piece made it impossible to generate grid outputs in a feasible manner using only logic gates
- Soldering of LEDS onto perfboard initally caused several bad contacts due to small amount of space between LEDs
- Input/output and memory management of SRAM chip to store grid had to have very precise and specific timing
- Original use of diodes was made impossible due to alterations in voltage causing the SRAM chip to incorrectly read inputs
- Programmable chips were not properly working, so much functionality had to be implemented manually through the use of other chips
- Asynchronous counter caused problems with read/write of RAM and had to be replaced with a  synchronous counter

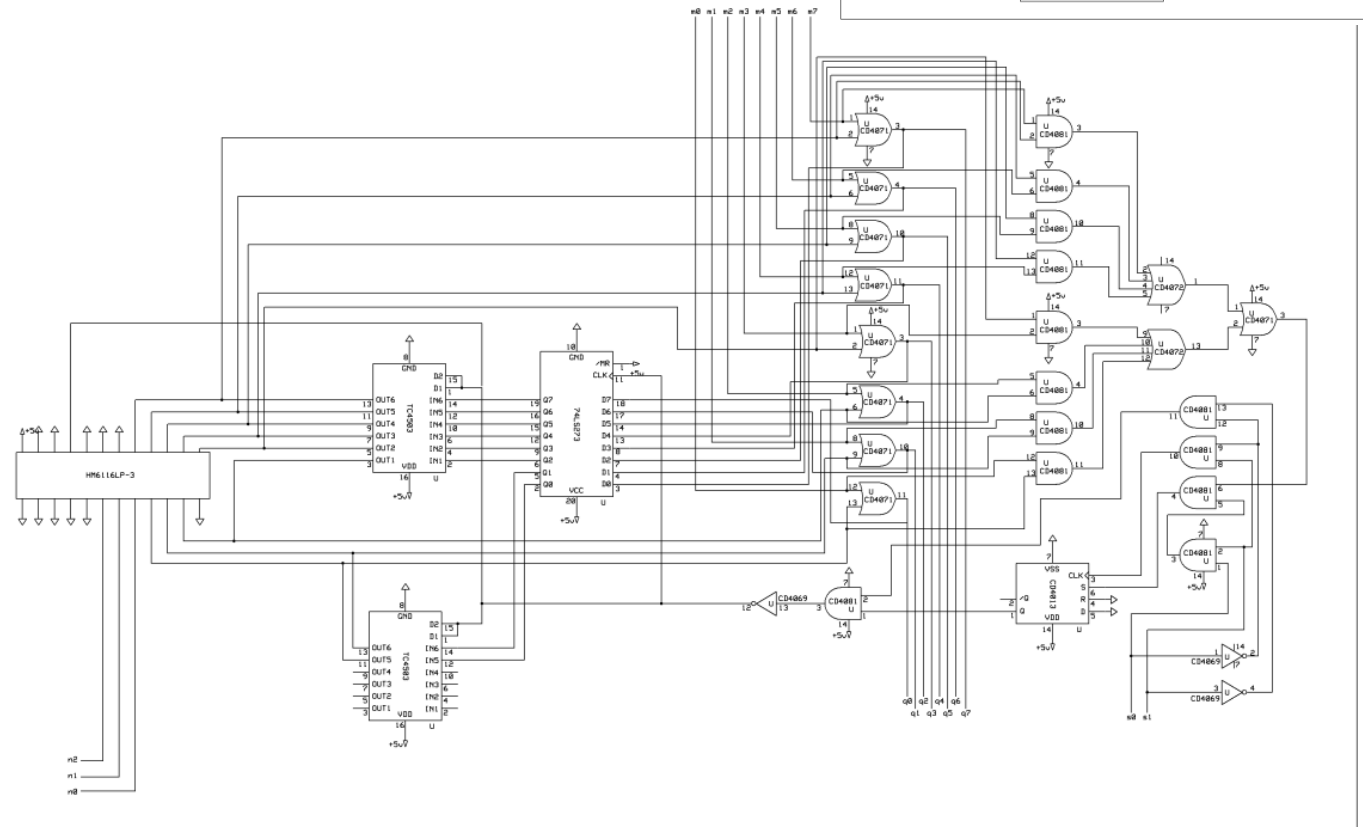
Logic Diagram



Schematic Diagram





Board Diagram