8×8 Tron

Summer 2018 ECE 150 (Data Logic Design)
Professor Risbud
Final Project Report

Nathaniel Kingsbury
Jonathan Lam

0. Abstract

Our aim will be to build the classic arcade game Tron (from 1982). In this multiplayer game, players move through an area and leave a trail behind themselves that acts as a wall. When a player hits any trail (be it their own or some other player's), they lose. Players are required to move constantly, and all move approximately once per second. If a player moves to one edge of the screen, they "wrap-around" to the other side of the screen. In our implementation, the screen will be an 8X8 grid of bi-color (red/blue) LEDs. Each of the two players will be assigned a color for their trail, and will be given a set of four buttons to change their direction of movement. Every second, both players will move forwards in their current direction of movement. Players will start out in predetermined locations for fairness. When one player hits a trail, the game will shut off (the clock will stop and motion will end), and a "victory LED" will turn on in the color of the winning player. If the two players enter the same spot on the same turn, or both hit a wall at the same time, the game will end but no victory LED will light, which will signal a draw. A "new game" button will be provided to reset the game and allow players to start over.

1: reference: http://www.cs.brandeis.edu/~pablo/tron/t1.html ("The opposite edges of the screen communicate with each other.")

1. Inventory

IC Chips

Count	IC #	Description
2	555	Timer
2	4013	Dual D-Type Flip-Flop
17	4015	Dual 4-Stage Static Shift Register With Serial Input/Parallel Output
5	4029	Presettable Up/Down Counter
1	4040	2-stage binary ripple counter
6	4051	Single 8-Channel Analog Multiplexer/Demultiplexer
1	4052	Dual 4:1 Analog Multiplexer/Demultiplexer
3	4053	Triple 2:1 Analog Multiplexer/Demultiplexer
2	4585	4-Bit Magnitude Comparator
3	4071	Quad 2-input OR gate
1	4081	Quad 2-in AND
1	4011	Quad 2-Input NAND Gate
1	4025	Triple 3-in NOR

Resistors and Capacitors

Count	Value	Description
1	1000µF	Decoupling capacitor
3	0.1µF	Signal-decoupling capacitor (prevents noise on a single signal, rather than on power and ground)
4	0.22µF	Edge-trigger capacitor
1	5.1kΩ	Edge-trigger resistor
1	6.8kΩ	Edge-trigger resistor
1	20kΩ	Edge-trigger resistor (in parallel with 33kΩ resistor)
1	33kΩ	Edge-trigger resistor (in parallel with 20kΩ resistor)
1	10kΩ	Edge-trigger resistor
2	3.3µF	555 Timing Capacitor
2	.015µF	555 Control Voltage Capacitor
1	470kΩ	555 timing resistor
1	910Ω	555 timing resistor
10	1kΩ	Pull-down resistors (8) and 555 timing resistors (2)
16	20kΩ	Pull-up resistor
65	470Ω	Current-limiting resistor
65	430Ω	Current-limiting resistor

Other Components

Count	Description
9	Pushbutton
9	Breadboard
65	Bicolor (blue/red) 5mm LED
4	Ribbon Cable Header (14-wide)

---

2. Design and Methodology

2a. Explanation of design

In brief, our design works as follows: an approximately 1 Hz clock with a duty cycle of nearly 100% synchronizes all events. The falling edge of this clock triggers the counters in the players' controllers to increment, and then the rising edge triggers the current coordinates to be written to memory, and to be checked against the memory to determine when the game ends (i.e. to determine if either player hit either trail). The brief delay between the falling and rising edges of the clock is large enough to make sure that the counters have fully updated before any writing or checking occurs, but is brief enough that it is not human-noticeable (so nobody will complain that they pressed a button to change directions and the game didn't respond).

The memory block consists of 8 8-bit shift registers and 3 muxes for each player (so a total of 16 shift registers and 6 muxes). Based on the y-coordinate of one player, the muxes select which shift register is active in both player's memory modules at once, by connecting that shift register's data in, Q8 (data out), and clock connected with the rest of the circuit (the inactive shift registers get no clock signals, so they simply store data for future use, and control the display). Hence, at any given time, corresponding shift registers in each player's memory modules are selected. Writing to memory and checking against memory is accomplished serially -- the rising edge of the master clock triggers another super-fast clock to give 16 pulses really rapidly. For the first 8 cycles, the first player's y-coordinate is used to select shift registers, and for the second 8 cycles, the second player's y-coordinate is used to select shift registers. Data is shifted cyclically on each falling edge of the super-fast clock (i.e. whatever is in Q8 of a shift register appears on that register's data in) except for the shift register for the player whose coordinates are being checked at the point corresponding with that player's current x-coordinate, at which point a 1 is fed into the data in regardless of the current contents of Q8. Because 8 clock pulses occur for each player, in the end nothing has changed within the shift registers for either player, except for the 1 bit that has been "written" into each memory bank. During this process, immediately before writing to one player's shift register, both players' shift register data outputs are checked for a 1.

At that moment, seeing a 1 on Q8 of a player's shift register signifies that that player has been in that location. In that case, a 1 is stored to a win/loss flags register, where it triggers the master clock to shut down. If the two players move to occupy the same spot simultaneously, a comparator similarly ends the game.

During an endgame, a win/loss LED lights up with the color of the winning player. In the event of a tie, there is no winner, so the game ends but the win/loss LED does not light. Pressing the "reset" or "new game" button causes the counters in the controllers to preset, and all registers to reset, causing the game to restart. See below for a verbal step-by-step in-depth explanation of the design, including a description of what's going on on the signal level with reference to how this plays into the larger game state. Additionally, on page 16, a timing diagram providing a visual representation of this information is given.

[0. Initial state]
When initially powered, the LEDs will appear lit in a random pattern, due to the indetermination of the original states of the bits in the shift registers (in the flip flops). Pressing the reset button will begin the game.

[1. Reset button pressed; rising edge of reset signal]
While the reset signal is high, all of the shift registers storing the LED states are reset, clearing the board. The preset enables on the counters in the controllers go high, which writes the starting positions to the controllers. Additionally, the win/loss shift register is cleared, which enables the slower (1 Hz) master clock.

[2. Reset button released; falling edge of reset signal]
On the falling edge of the reset signal, the timing counter is reset, which enables the fast clock to give 16 pulses to write the starting positions to the shift register banks.

[3. Wait for user input]
During this portion of the cycle, nothing dynamic occurs inside the game, save for the charging of the timing capacitor on the master 555 timer. The purpose of this period of time is to slow things down enough to allow for user input -- at any time, if the user presses a directional button, it updates a pair of D-type flip-flops in their controller, (potentially) changing which of their counters (x or y) is active, and whether it is in up mode or down mode. This can occur anywhere in the cycle, as it is independent of the rest of the circuit; however, in order to make things usable, this long period of time is needed so that things don't happen altogether too fast for informed user input.

[4. Falling edge of low-frequency clock pulse]

The falling edge of the low-frequency clock is used to indicate the beginning of each player's motion. This triggers the counters on the controllers to advance.

[5. Rising edge of low-frequency clock pulse]
An edge trigger on the rising edge of the low-frequency clock pulse sets the preset enable pin of the countdown 4029 counter high, allowing the x-position of player one to be inputted to the counter. The edge trigger is calibrated to last just long enough to overlap the first rising edge of the high-frequency clock pulse.

[6. Rising edge of first high-frequency clock pulse -- "Read"]
On the first rising edge of the high-frequency clock, the 4029 preset with the x-coordinate does not count down due to preset enable being held high. However, in any case, the checking logic does check to make certain that there is nothing already in the position currently being read. In the case where the x-coordinate that had been preset into the 4029 is a 0, this gets written into the win/loss shift register.

[7. Falling edge of first high-frequency clock pulse -- "Write/Shift"]
This falling edge is directed to the correct shift register's clock, so that only the correct rows in both memory modules are shifted. The data input to the shift register comes from a 2:1 multiplexer that outputs a guaranteed logical 1 if the portion (group of 8 clock cycles) of the update cycle corresponds with the "write" time for the player corresponding with that memory bank (indicated by Q4 of a counter) and if the time within the cycle is correct to do a writing (from the carry out of the down counter fed the current player's x-coordinate). When these conditions are not both true, the multiplexer outputs the output of Q8 of the selected shift register otherwise. This writes a 1 into the memory module at the correct coordinate position, and maintains all of the other bit values.

[8. Rising edge of second fast clock signal until falling edge of sixteenth fast clock pulse]
The same rising-edge checking and falling-edge writing of data and shifting (from the previous two steps) occur during these cycles. The one change is that the 4029 is allowed to count down, such that it goes low on the nth rising edge of the high-frequency clock (counting from 0). This way, on the nth rising edge, the output of the checking logic is actually written to the endgame shift register.

[9. Rising edge of 4040 Q4 clock signal]
This happens concurrently with the falling edge of the eighth fast clock signal, as the 4040 counter is falling-edge triggered and Q4 goes from low to high on the eighth pulse. The Q4 output is connected to the address pins of the two 4053 2:1 multiplexers to switch

the shifting and reading from player one to player two (changing "phase" from zero to one; see PH input in 3b). While this is concurrent with Q4, the longer propagation delay of a change in address to output update for the 4053 multiplexers than the propagation delay of the shifting allows us to use the same edge to trigger both events and correctly shift before changing phase. The preset enable of the 4029 countdown counter is set high using an edge trigger on the rising edge of the 4040 Q4 clock signal, and calibrated to last just past the rising edge of the next high-frequency clock pulse.

[10. Rising edge of the 4040 Q5 clock signal]
This happens concurrently with the falling edge of the sixteenth fast clock signal. This is connected to the reset pin (pin 4) of the high-frequency clock timer, which stops the shifting and writing, and stops the counter as well keeping the system in this state until the master clock or reset button reset the 4040 and re-enable the fast clock. Assuming no endgame conditions, this returns things to state 3.

[11. Endgame Conditions]
If at the end of the counting of the controllers, the x and y coordinates of each are the same, or if at the end of the read/write cycle there is a 1 in the endgame shift register, the game ends. (This is accomplished by NORing the two least significant outputs of the shift register together with the output of a pair of comparators to create a unified "game over" signal). There are a few stages to this -- a multiplexer connected to the "game over" signal immediately holds the clock signal going to the counters low, to prevent any spurious edges going to the counters from causing bugs. Additionally, the "game over" signal goes to the reset pin of the master clock, shutting down the game in a more general way until the reset button is pressed. At this point, a few other signals come into play -- the individual red and blue pins of the endgame LED go to the stages in the shift register that signify one player's loss, such that the LED lights red when blue loses, and lights blue when red loses. Additionally, the common ground pin goes to an additional signal that goes high during a tie, such that during a tie the game stops, but the LED simply doesn't light, regardless of the signals going to the red and blue independent LEDs.

[11. Falling Edge of master clock]
Return to step 3. This indicates that another second has passed, and the next move is completed.

During the design process, we quickly found that it would be easiest to divide the problem into a modular solution, so that the physical building process could be divided, each team member would have a more specialized focus on a smaller project and have to worry about fewer technicalities of modules they are not building (allowing quicker build times), modules could be

tested in isolation, and so that future revisions to the internals of any section can be done so long as the outputs remain the same (i.e., modules are black-box abstractions).

2b. Modules

Name	Count	Description
Controller	2	includes interface for users and determines coordinates
Input management	1	selects correct controller inputs for writing and special tie condition
Memory	2	stores previous locations of players
Write/Endgame detection	1	processes new inputs and checks for endgame conditions
Timing	1	includes timer ICs and manipulation of clock signals
Reset	1	signal to restart game

2c. Notes about final design

The wires are color-coded as follows (CLK, RST, and LED categories take priority):

Green	CLK: oscillating timer signals, regular or inverted, from either 555 timer
White	RST: directly connected to the reset button
Yellow	Controller modules
Blue	LED: blue LED on display; or Input management module
Orange	Memory module
Brown	Write/Endgame detection module
Red	LED: red LED on display; or VCC
Black	GND; or Tie signal output (goes to ground pin of the bicolor win/loss LED)

The final design comprises nine breadboards and an LED display (solder protoboard), including two controller breadboards connected to the other seven by ribbon cables. Each controller has the four directional control buttons for its respective player, and the reset button is located on the display. The ribbon cable allows a player some mobility with his or her controller.

---

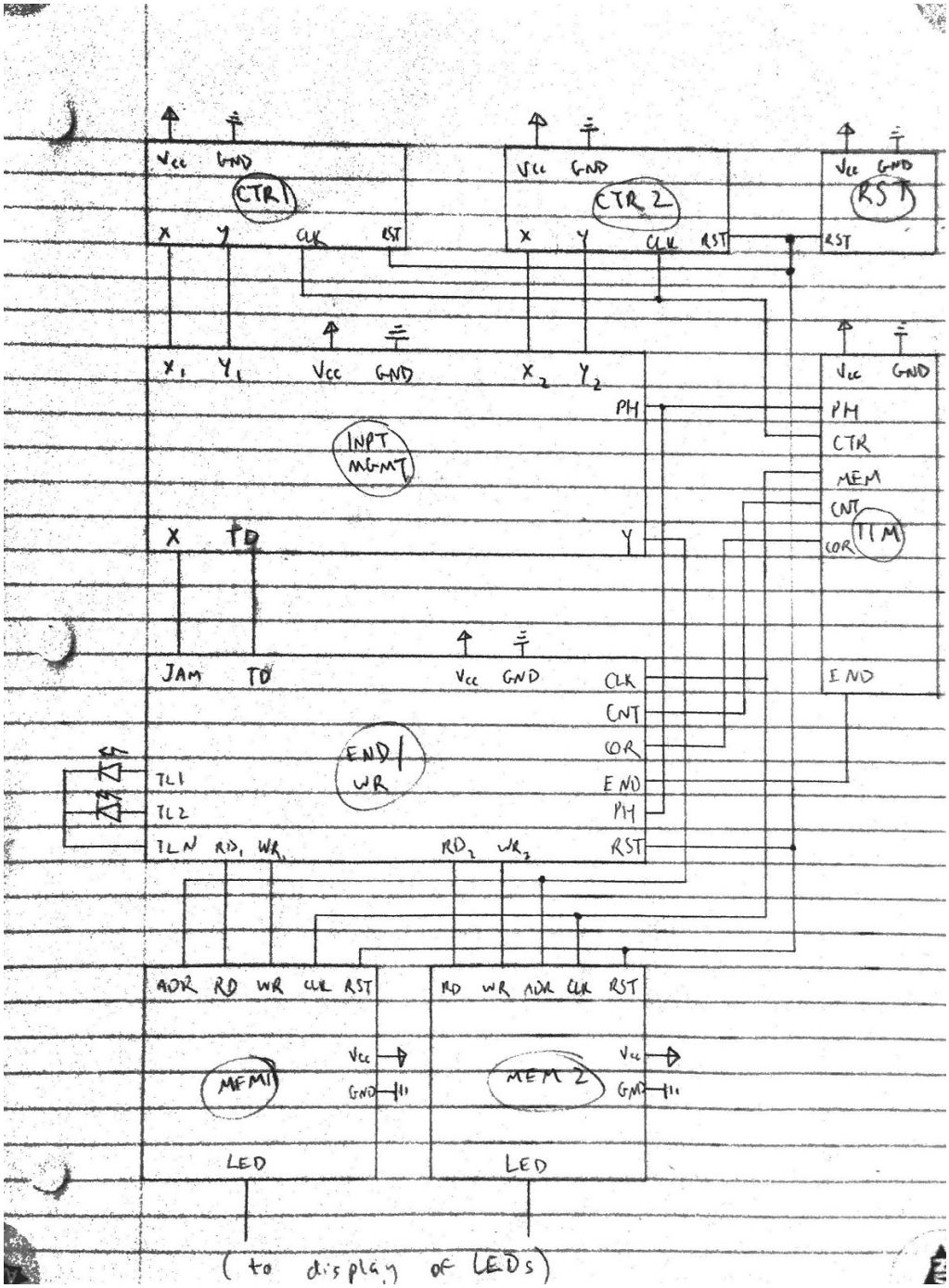
3. Schematic

Because of the size and the modular design of the Tron machine, the schematic is split into its modules. For each module described below, a table of module input and output signals, as well as the source or destination modules of those signals, is provided. Every module is completely standalone and can be unit-tested by supplying appropriate inputs and checking the module outputs.

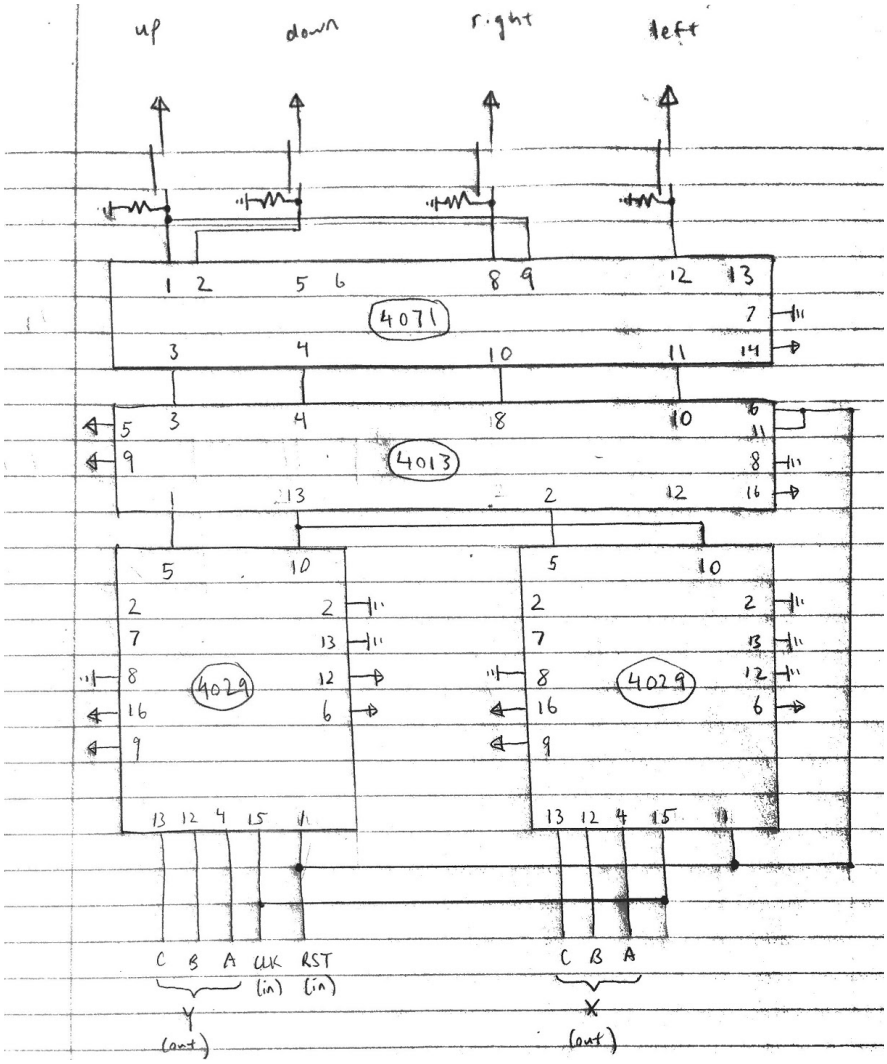
A single controller module and memory module schematic are provided below to reduce redundancy. Each of these modules are duplicated, with slight changes in the copy of the controller module (see the provided notes).

3a. System

The diagram below displays the connections between every module.



3b. Controller module



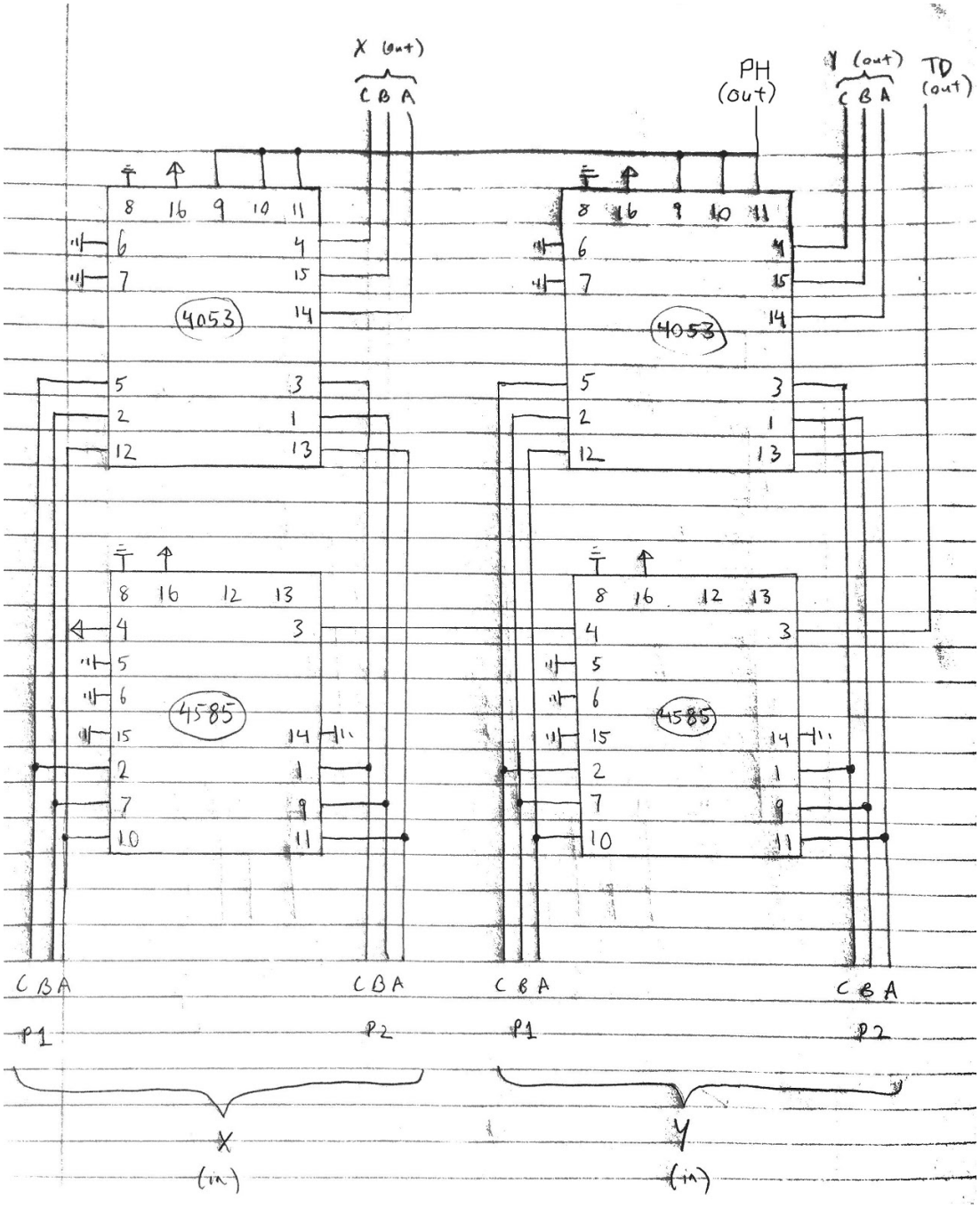
I/O Channel	I/O	Source or destination
RST	in	reset module RST output
CLK	in	timing module CTR output
X (CBA)	out	input management X inputs
Y (CBA)	out	input management Y inputs

Notes:

The schematic shown above is the controller for player one. The controller for player two is identical, except for the following changes (which create a different starting position and direction):

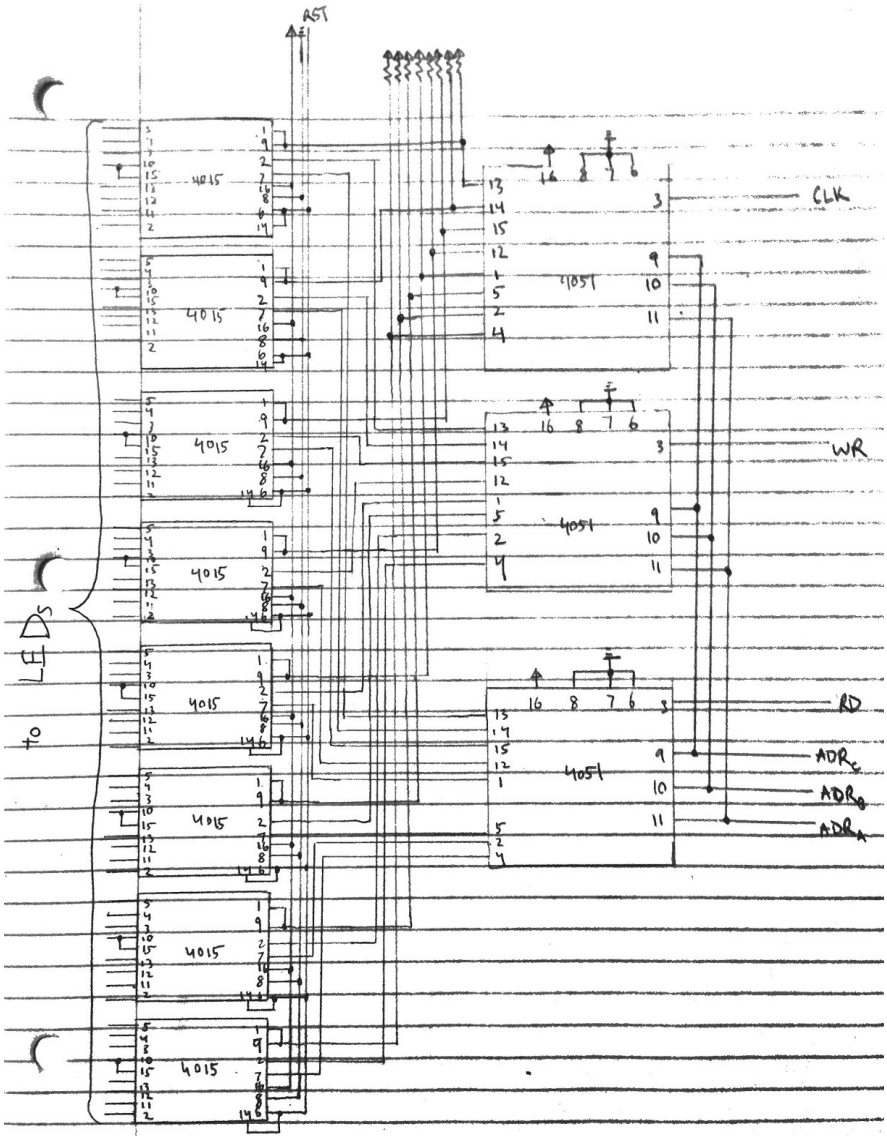
- Pin 9 (DATA 2) on the 4013 is connected to GND.
- Pins 13, 12, 6 (JAM inputs 3, 2, 1) on the right 4029 are connected to GND, VCC, GND, respectively.

3c. Input management module



I/O Channel	I/O	Source or destination
X (P1 CBA, P2 CBA)	in	controller module X outputs
Y (P1 CBA, P2 CBA)	in	controller module Y outputs
PH (phase)	in	timing module PH output
X (CBA)	out	write/endgame module JAM inputs
Y (CBA)	out	memory module ADR inputs
TD (tie detection)	out	write/endgame module TD input

3d. Memory module

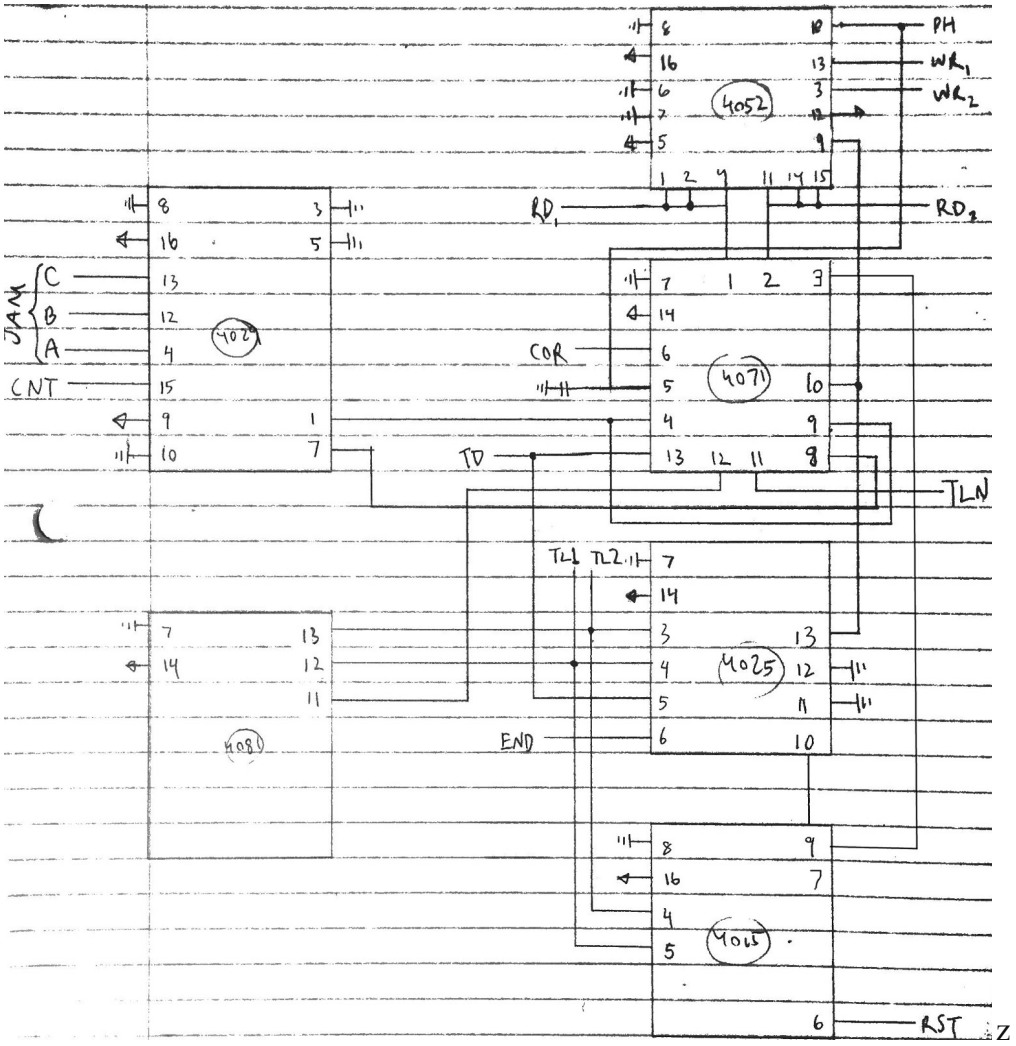


I/O Channel	I/O	Source or destination
ADR	in	input management module ADR outputs
RST	in	reset module RST output
WR (write)	in	write/endgame detection module WR output
CLK	in	timing module MEM output
RD (read)	out	write/endgame detection module RD input
LED	out	display LEDs

Notes:

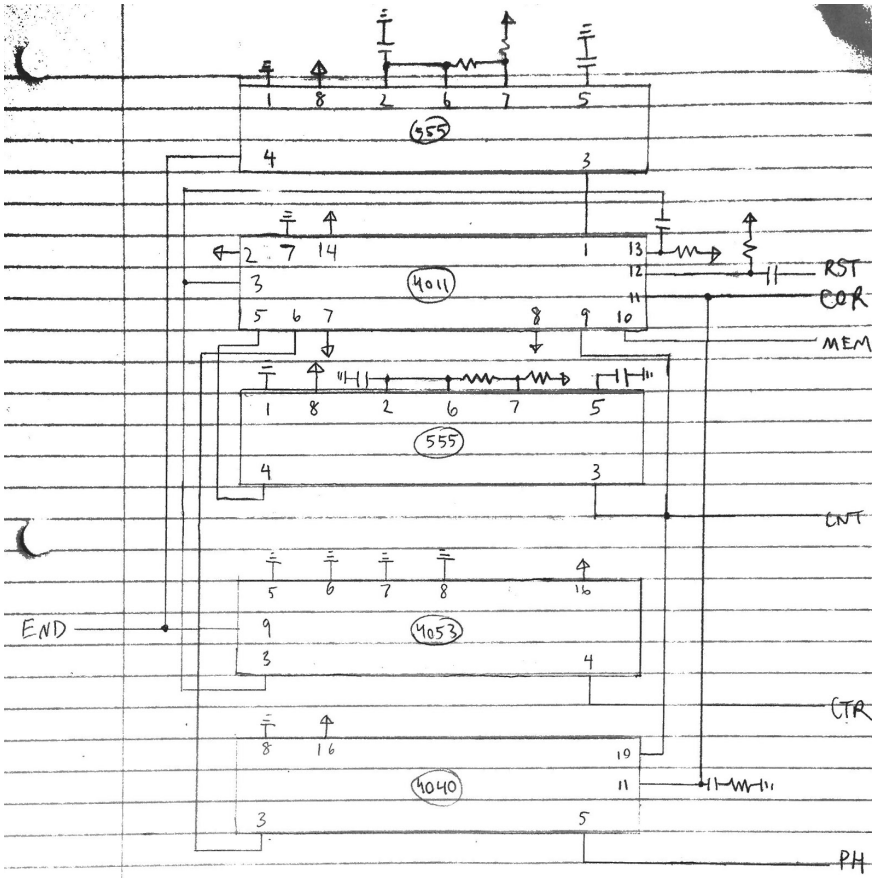
- This is duplicated for the second player. The CLK, ADR, and RST inputs are the same.
- The LED output goes to an 8x8 array of bicolor LEDs. Every output from one memory module is the same color. In our design, this was implemented on a solder protoboard.

3e. Write/Endgame detection module



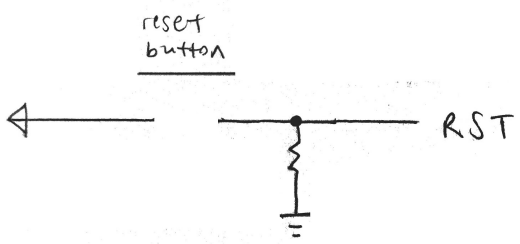
I/O Channel	I/O	Source or destination
CLK	in	timing module MEM output
CNT	in	timing module CNT output
COR	in	timing module COR (clock or reset) output
PH (phase)	in	timing module PH output
RST	in	reset module RST output
JAM (CBA)	in	input management JAM outputs
TD (tie detect)	in	input management TD output
RD (read)	in	memory module RD output
WR (read)	out	memory module WR input
END	out	timing module END input
TL1	out	winner LED player 1 indicator
TL2	out	winner LED player 2 indicator
TLN	out	winner LED cathode

3f. Timing module



I/O Channel	I/O	Source or destination
END	in	write/endgame detection module END output
MEM	out	memory module, write/endgame detection modules CLK input
CTR	out	controller module CLK input
COR	out	write/endgame detection COR (clock or reset) input
CNT	out	write/endgame detection CNT input
PH (phase)	out	write/endgame detection, input management PH input

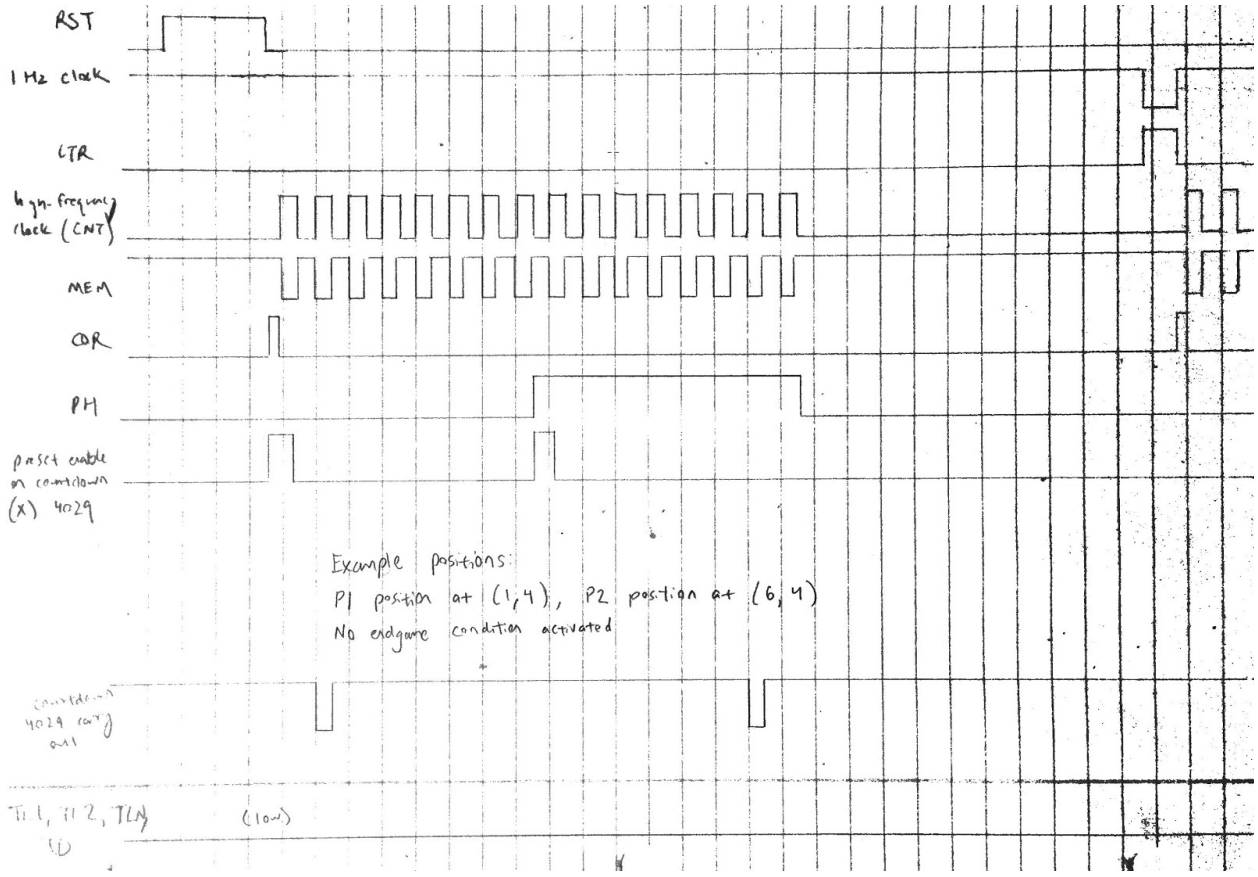
3g. Reset module



I/O Channel	I/O	Source or destination
RST	out	controller, memory, write/endgame detection modules RST inputs

4. Timing Diagram

Selected signals essential to the general timing of the circuit are shown in the timing diagram below.

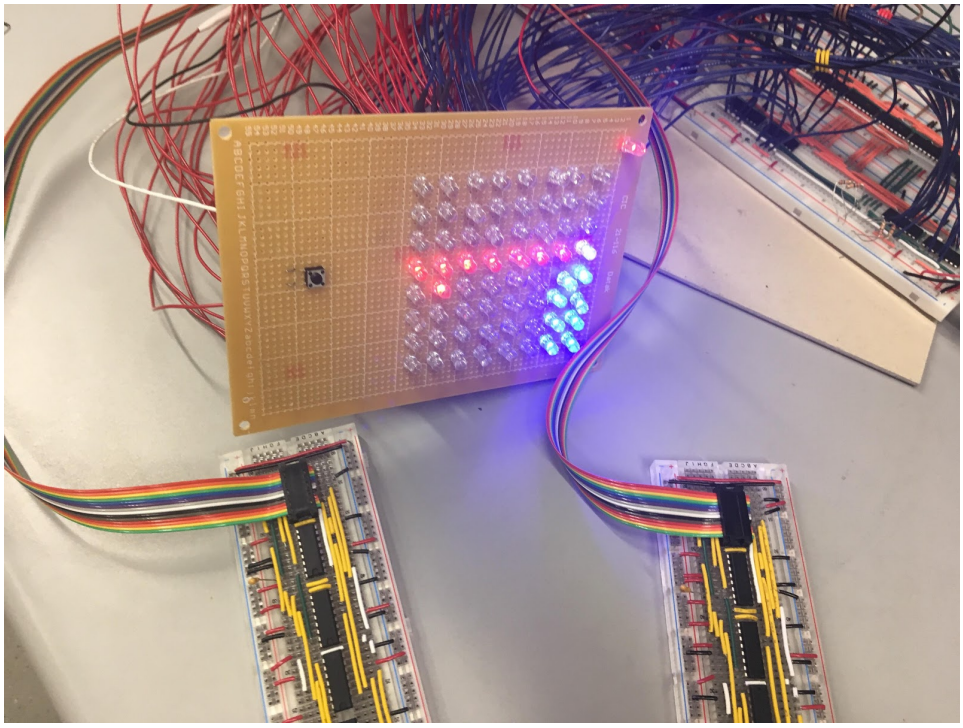


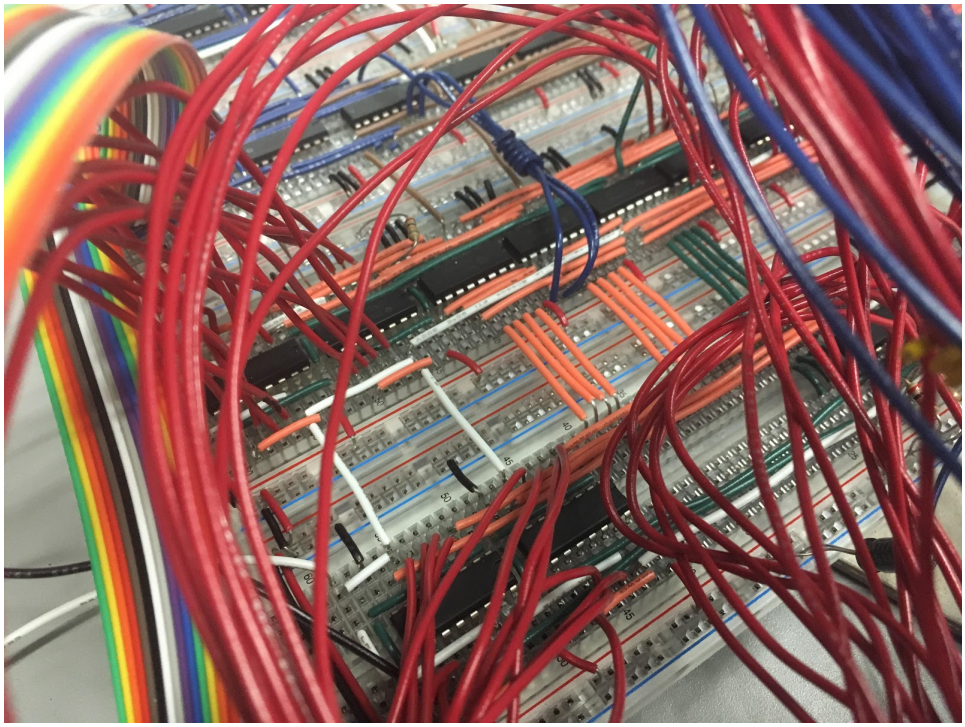
Notes:

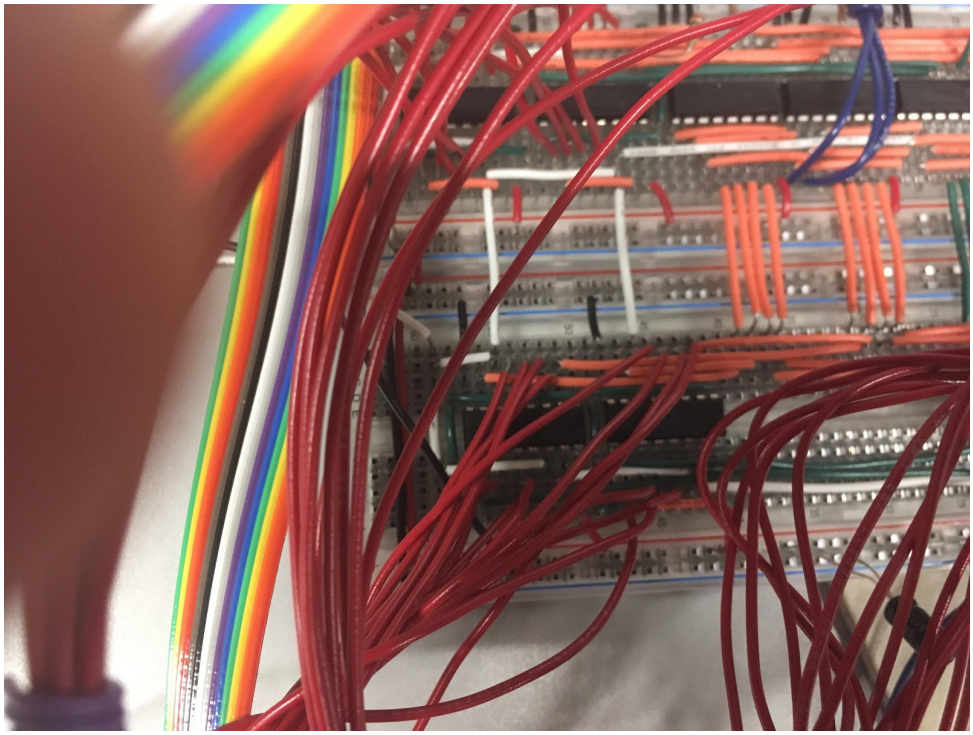
- Timing is not drawn to scale. Propagation delay between events may be exaggerated, and transition (LOW-HIGH or HIGH-LOW) delays are not emphasized.
- The RST signal at the beginning of the timing diagram only happens at the beginning of the game. After that, normal game operation is continued by the 1Hz clock signal, which gets disabled at endgame.
- The meaning of the abbreviations RST, CTR, CNT, MEM, COR, and PH can be found in the module schematics or the system schematic. The “preset enable on countdown 4029” and “countdown 4029 carry out” refer to the 4029 counter used to count down to the correct x position of a player (in the write/endgame detection module).
- On an endgame detection, TL1, TL2, or TD goes high, and the main clock is disabled. The reset button must be pressed (a high pulse on the RST signal) to restart the game.

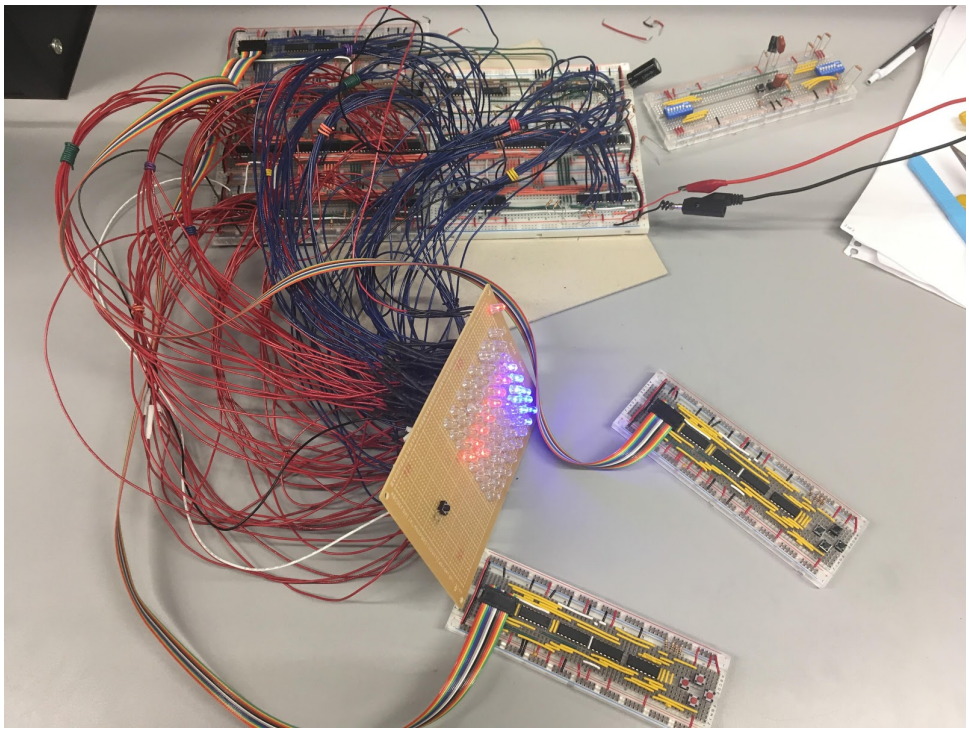
5. Media

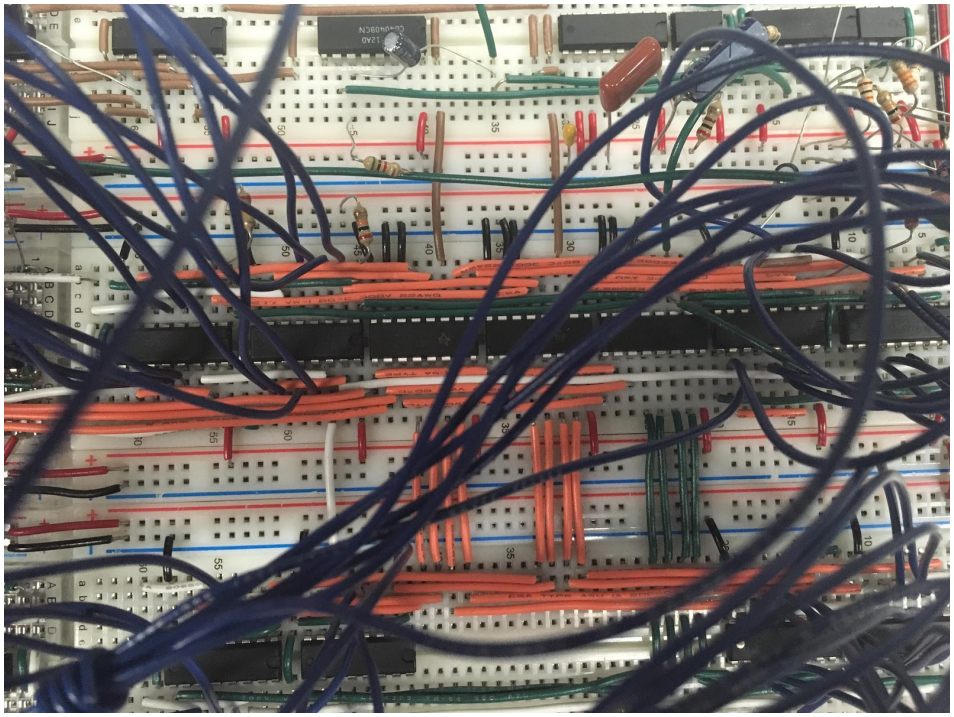
Images

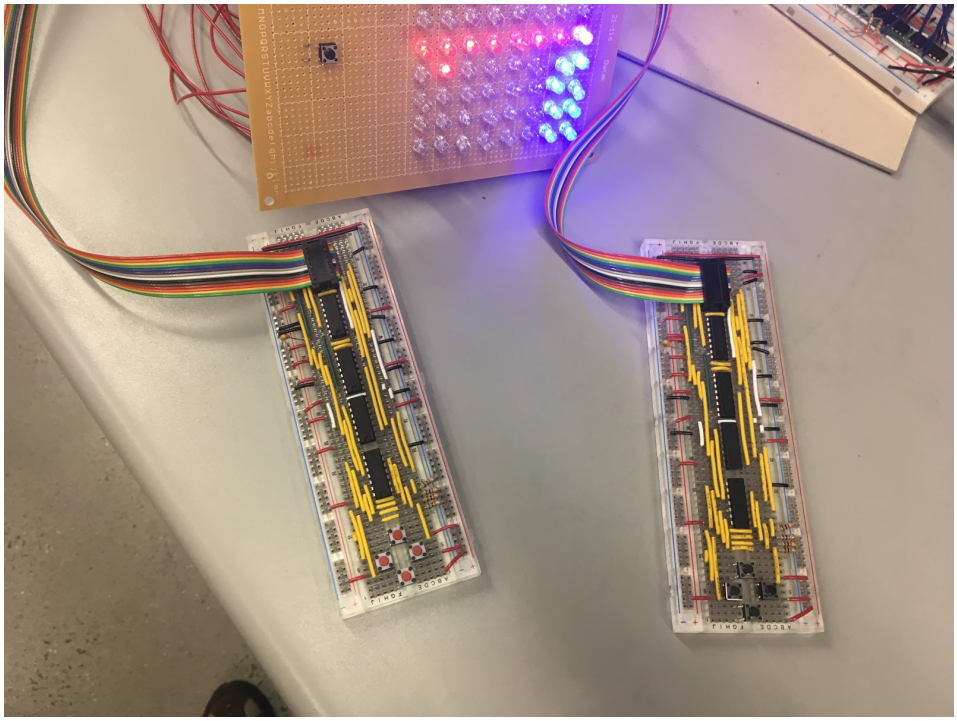


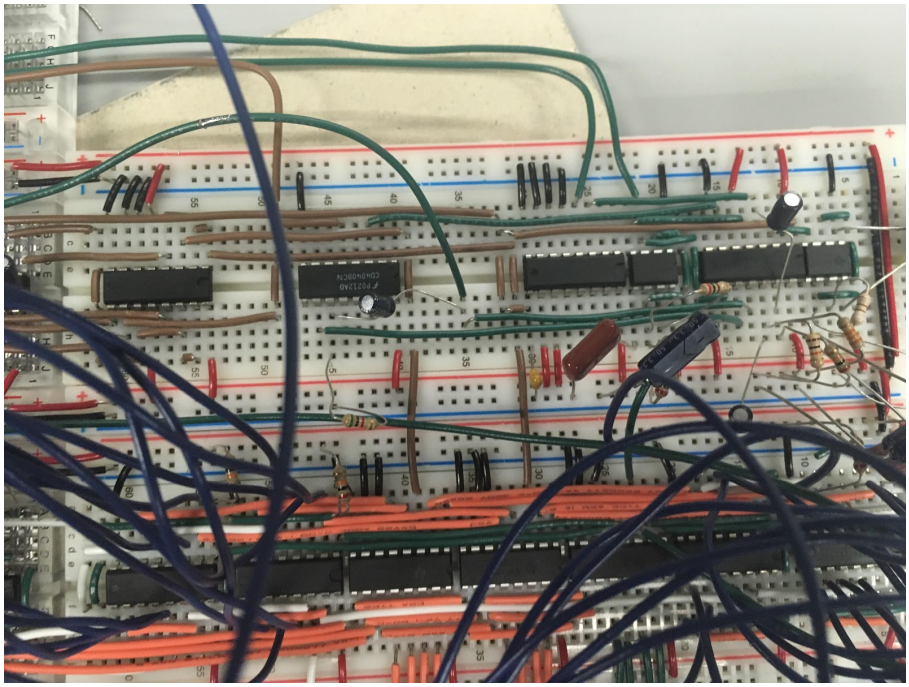


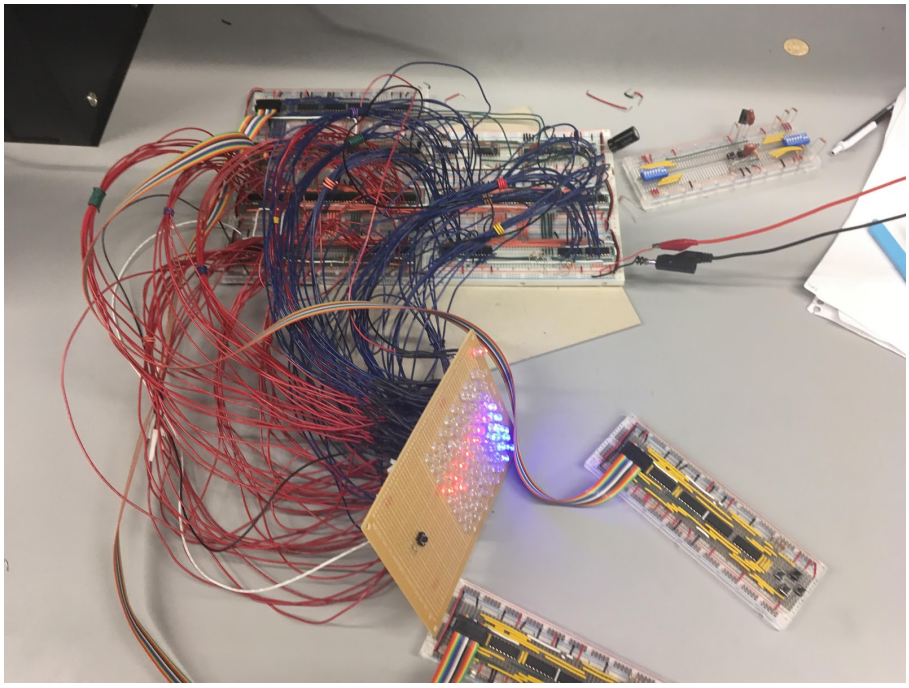












Link to video
https://drive.google.com/file/d/1CWRF6QlVLNOdmRWxrGb-mySU7Je7T2X5/view?usp=sharing

6. Debugging Process and Possible Future Extensions

Significant bugs

Listed below are a selection of bugs encountered during the construction of the project, and which took a considerable amount of debugging time or a modification to the original design.

- The pull-down resistors on the clock signals within the memory module were too small, leading to the clock signal going in between consistent HIGH and consistent LOW instead of going truly HIGH. This led to inconsistencies while shifting, which were eventually fixed by increasing the size of the pull-downs from 1 kΩ to 20 kΩ.
- The drop in voltage across power and ground caused by many LEDs being lit on the display simultaneously when powered by a battery pack caused several carefully-calibrated RC edge-detector circuits to fail. Switching to a power supply solved the problem.
- The initial position (position of players when the game is reset) of the players was not being written to the display, and therefore the LEDs at the starting position did not light (and players didn't lose when crashing into that spot), but pressing one of the direction keys on the controller still influenced the de-facto starting position (the first LED to light and first position to be written to). A NAND logic gate was added to also trigger writing the players' positions on the falling edge of the reset button.
- The 4029 counter determining when the offset in the selected shift register is correct was counting one too many times, creating an incorrect offset when inputting data into the shift registers. This required the observation that the edge triggers for the preset enable pin of that counter had to be increased. Careful calibration by increasing resistor size allowed for the preset enable to be HIGH for the correct amount of time.
- Inconsistencies with counters skipping controllers was fixed by adding decoupling controllers between VCC and GND. Later on, future inconsistencies caused by noise in the CLK signal to the controllers prompted the addition of small (0.1 μF) capacitors between the CLK signal and GND to reduce noise.
- The initial endgame detection allowed the CLK signal to the controllers to return high, creating another rising edge (and thus causing the players' position to jump once on endgame). A multiplexer, with one input tied to GND and the other to the CLK signal, with its addresses connected to the endgame condition, was added to eliminate the last rising edge.
- A glitch state caused noise in the address pin to the multiplexer described in the previous bug, prompting a capacitor to be put between that address and GND to reduce the noise.

- The zeroth-row LEDs were connected with shorter wires and with electrical tape. This caused problems and required much time un- and re-soldering LED connections that broke. (This is not a bug, but a time-consuming fix to the machine.)

Possible Extensions:
In future iterations of this project, better care should be taken to eliminate the flicker in the LEDs. Because all eight LEDs are shifted (because the shift registers used had serial input), the quick flashing may be distracting or unpleasant to view. This can be improved by using parallel-load shift registers or using resistor-capacitor circuits.
Another possible improvement is that the machine can prevent users from turning back on themselves, killing themselves immediately. This is usually disallowed by the machine in Tron games, but we felt that it did not affect gameplay significantly (as the player can simply learn to avoid trying to double-back) and didn't include in our implementation. Additional "add-on" extensions for a future implementation could also include a switch to enable or disable wraparound, making the game "best two out of three," and causing the master clock to speed up as the game progresses (either every n ticks of the clock or with successive matches).

---

7. Acknowledgements

1. Professor Risbud, for approving and evaluating the project.
2. Dr. Brian Kingsbury, for (informally, as part of normal conversation) bouncing ideas back and forth with Nathaniel as to the likely sources of bugs.