NSU

“The Game of Noughts and Crosses”

Nurdolotova Sabina

Derunets Roman

Contents

[Contents 2](#_Toc72575695)

[**Brief Overview** 3](#_Toc72575696)

[**Hardware** 3](#_Toc72575697)

[TTTC 3](#_Toc72575698)

[SDR 4](#_Toc72575699)

[BPC 6](#_Toc72575700)

[GSDD 7](#_Toc72575701)

[**Software and main subroutines** 8](#_Toc72575702)

[playerturn 8](#_Toc72575703)

[drawcounter 8](#_Toc72575704)

[wincheck 9](#_Toc72575705)

[lose/win/draw 9](#_Toc72575706)

[computerturn 10](#_Toc72575707)

## **Brief Overview**

## From the three projects presented, we decided to choose project "A “-the game” Tic-Tac-Toe", as we found it quite entertaining. To create this game, we needed to use a mix of Logisim and Cdm-8.

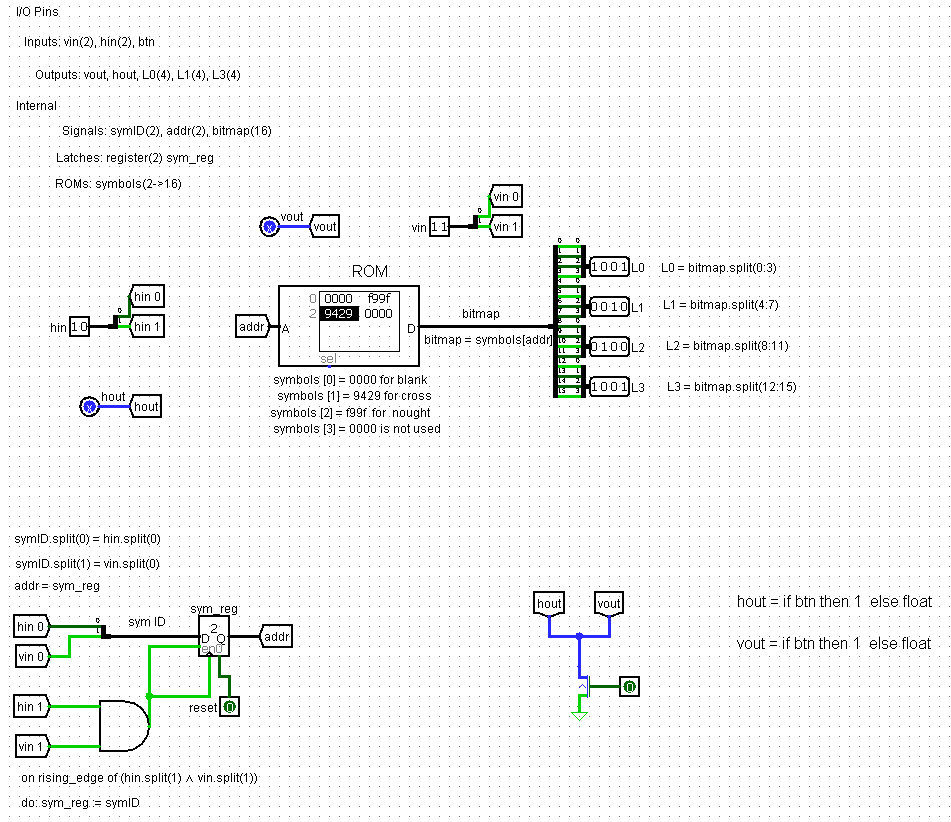
## Our group faced a lot of problems. Since we were not used to using Logisim and Cdm-8 in the complexity that we had to, this meant that we had to learn how to use both software and hardware at a more advanced level in order to use them to their full potential.

## The main task is to develop a 3x3 gamepad, connect it to the Cdm-8 and write a program in assembly language to control the game. At the same time, each of the 9 cells of the gamepad controls a 4x4-pixel LED that can display a cross, a zero, or an empty space. Also, each cell has an input button that the player uses for their turn. The gamepad has chips that can be used to connect to the I / O bus of the Cdm-8 processor to control the game.

## **Hardware**

TTTC

TTTC (Tic-Tac-Toe Chip) – this is a chip that controls the matrix display of the cell and the input button.

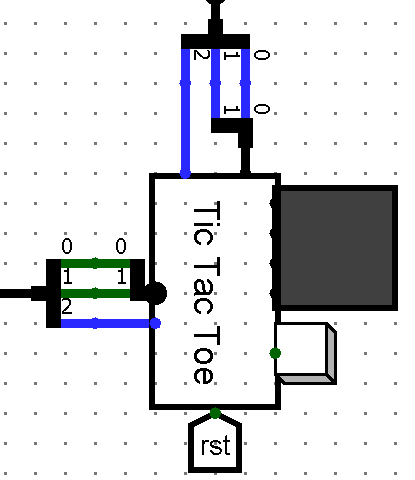


The chip consists of input pins: vin(2), hin(2) и btn, output pins : vout, hout, L0, L1, L2, L3 and rst (reset).

The ROM contains four values: symbols [0] = 0000 for blank space, symbols [1] = f99f for nought, symbols [2] = 9429 for cross, symbols [3] = 0000 (isn’t used).

The 0 bits of vin and hin represent the display symbol ID (sym ID). It is fixed in the sym\_reg register when the first bits of vin and hin are 1. The address in the ROM is the ID of the symbol to display.

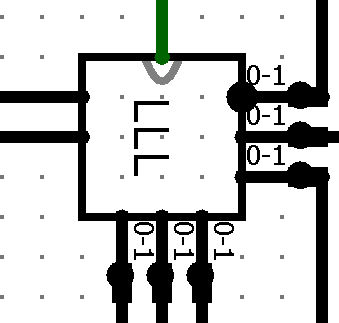
The image is read from the ROM using this address. Thus, when sym ID = 00 - an empty space is showed on the display, when sym ID = 01 - a zero, when sym ID = 10 - a cross

 On the North side of the TTTC chip (running East to West) we have a 1-bit pin hout, and a 2-bit pin hin. All three bits are connected to the cell’s horizontal bus (hout to wire 2 and hin to wires 0 and 1).). pin labelled hout carries output from the TTTC chip, and hin carries input to the TTTC chip.

On the West side of the TTTC chip there is a 2-bit vin and a 1-bit vout, for connection to the relevant vertical bus. A 4x4 LED matrix display is connected directly to the TTTC chip’s output pins L0, L1, L2, L3 (4 bits each) on its East side. These drive rows 0, 1, 2 and 3 of the matrix respectively, to display a nought, a cross, or a blank

A button is provided for a human to play a cross in the cell. It is connected to a 1-bit input pin on the East side of the TTTC chip, labelled btn. On the South side there is a rst tunnel to reset the cell

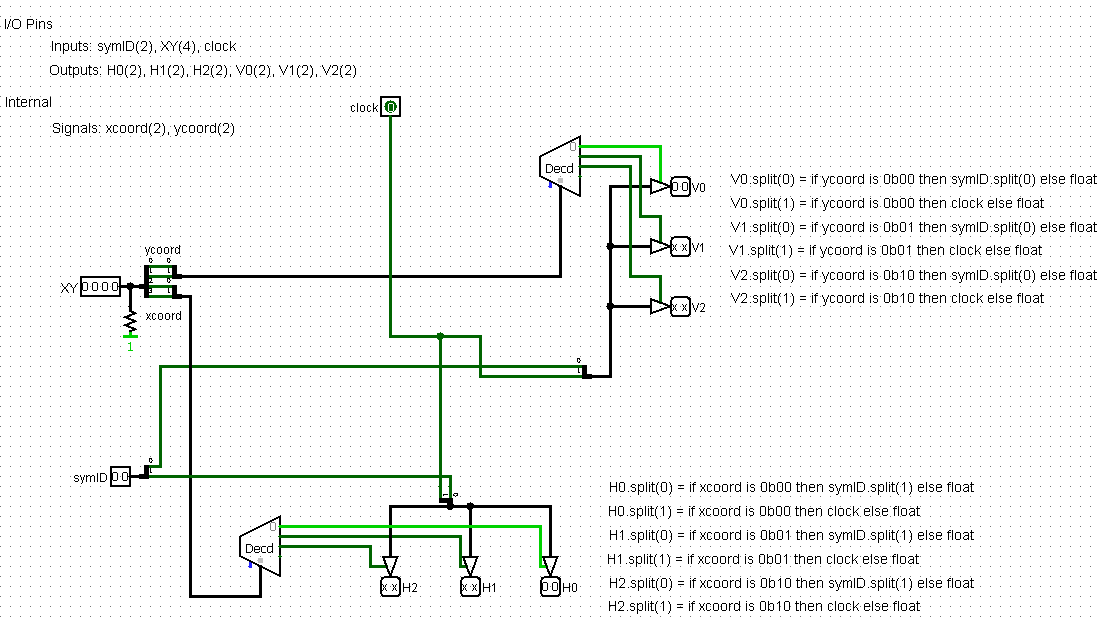
SDR

 It is used to convey the ID of a symbol to be displayed on the game pad to the correct TTTC chip controlling a given cell.

The symbol ID and the xy coordinate of the cell come from the processor. This chip has six 2-bit output pins H0, H1, H2 and V0, V1, V2, each connected to bits 0 and 1 of one of the buses.

The symbol ID and cell address are received by the SDR chip on the west side via the 2-bit symID pin and the 4-bit XY pin.

When the processor needs to write a symbol (nought, cross or space) into the cell with coordinates xxyy it must send both a 2-bit symbol ID and the cell address xxyy to the SDR chip. The cell address is decoded by the SDR chip, and used to select the right horizontal and vertical bus pair for the cell, this routing the symbol ID to the correct TTTC chip.



XY содержит координаты x, y используемого чипа TTTC.

The SDR decodes the XY address as follows:

First, it divides the address into two 2-bit coordinates (xcoord and ycoord). Then the coordinate is sent to the decoder, which, depending on the value of the coordinate, determines which of the decoder outputs will be 1. Then the data from the decoder goes to three control buffers, which, in turn, take a 2-bit value (bit 0 depends on symID, and the first one depends on clock) and send this value to the output pins (V0, V1, V2 for ycoord and H0, H1, H2 for xcoord). Bit 0 of the vertical output depends on bit 0 of symID, and bit 1 depends on clock. Bit 0 of the horizontal output depends on bit 1 of symID, and bit 1 depends on clock.

symID collects the identity of the symbol to be displayed The XY coordinates are used to route the combination of symID and clock pulse to the correct TTTC chip Five TTTC chips will receive a clock pulse, but only one will receive simultaneous clock pulses on both hin and vin. This will cause it to latch the ID of the symbol to be displayed

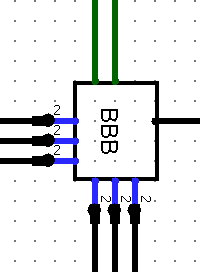
*How does the correct character ID get into the correct TTTC chip?*

Bit 1 of the symbol ID is directed to bit 0 of the selected horizontal bus, and bit 0 of the symbol ID is directed to bit 0 of the selected vertical bus. Thus, the zero bits of the two buses carry the ID of the displayed character.

The TTTC chip needs a trigger to lock the character ID in the internal 2-bit register. The trigger is provided on bit-1 of the corresponding pair of horizontal and vertical buses simultaneously, each of which is raised and then reset after the data is approved. This captures the data in a register. Only one cell on the crossbar will have the bit-1 of both the horizontal and vertical buses raised at the same time, so only one cell will capture data despite each bus being connected to 3 cells.

BPC

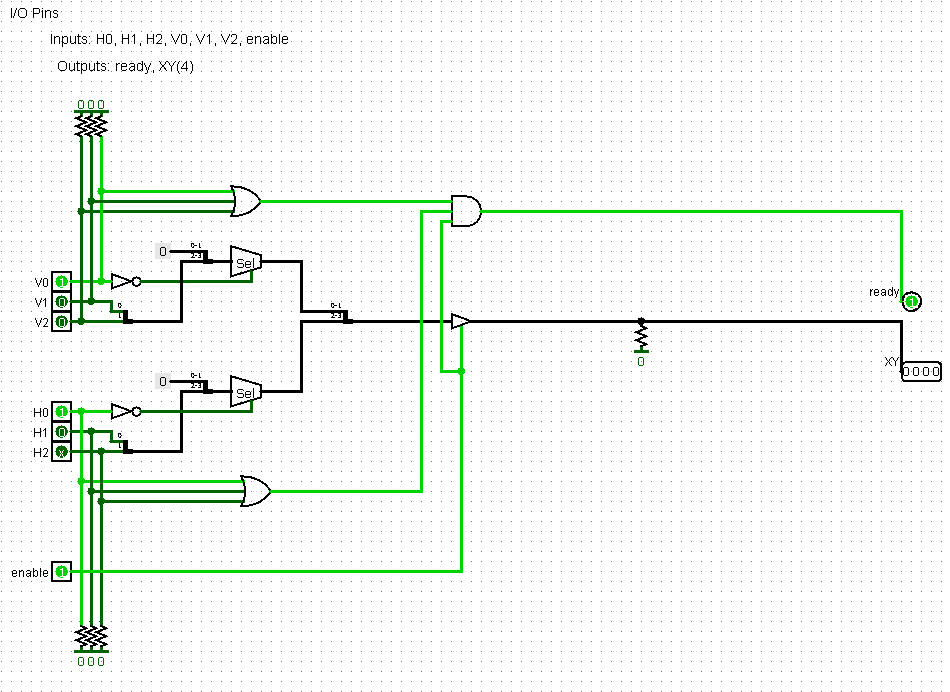
Its job is to capture and hold the address of the cell in which the most recent button press occurred. This data may then be read by the processor.

 The BPC chip latches the grid address of the cell containing the last button that was pressed. Each of the pins H0, H1, H2 and V0, V1, V2 is connected to bit 2 on one of the six buses. The BPC chip is not connected to bits 0 and 1.

BPC has input pins H0, H1, H2, V0, V1, V2 and enable and output pins ready and XY. There are also matching registers, which in the case of floating values will change them to zero. Using the bit selectors, we convert the input pins. For example, if V0 = 1, then coordinate yy = 00, if V1 = 1, then coordinate yy = 01, if V2 = 1, then coordinate yy = 10. Same with xx.

Then, if the button was pressed, the xx and yy coordinates are directed to the output pin XY. So cell 1 in the gamepad has address 0 (0000), cell 2 – 1 (0001), 3 – 2 (0010), 4 – 4 (0100), 5 – 5 (0101), 6 – 6 (0110), 7 – 8 (1000), 8 – 9 (1001), 9 – a (1010)

When the button is pressed, the ready output pin is 1 (only if at least one of the pins H0, H1, H2 and at least one of V0, V1, V2 are equal to 1)



GSDD

GSDD (Game State Display Driver) - chip, which lights an LED indicting win/lose/draw when the game is over.

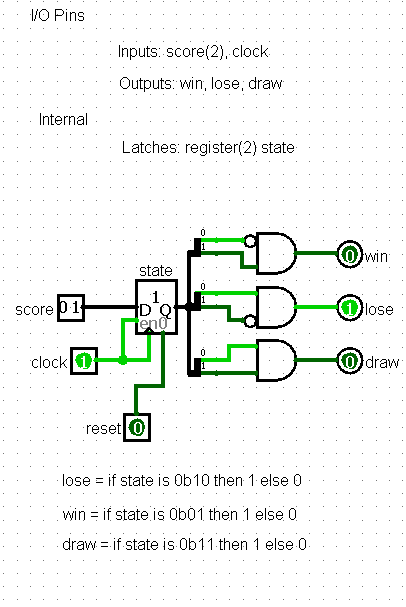
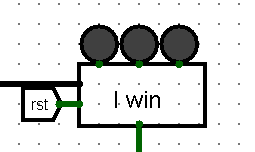
During play the processor sets both bits to 0. On the last turn (the one that results in a win, a loss, or a draw) the processor outputs a 2-bit signal on these two wires that identifies which of the LEDs should be lit.

GSDD has 2-bit input pin score and 1-bit clock, three output pins: win, lose, draw, 2-register state and reset.

When score = 01, game state is win (for computer player) and the red LED lights up (defeat for a person)

When score = 10, game state is lose and the green LED lights up

When score = 11, game state is draw and the yellow LED lights up

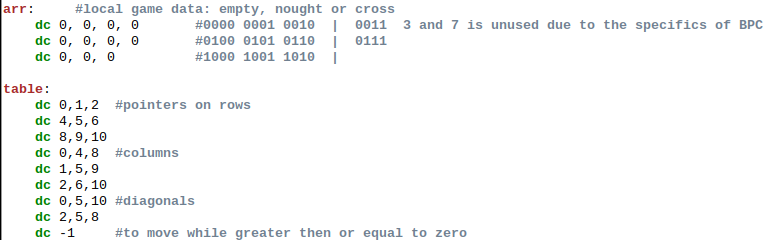
На западной стороне вход score (состояние игры)

На южной – clock

На северной - светодиоды

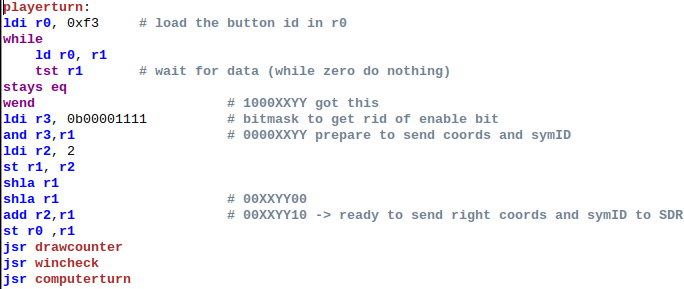
## 

## **Software and main subroutines**

 In memory, **arr** is allocated for local game data to store information where crosses, noughts, and blank space are located,

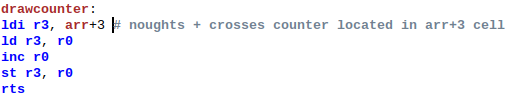
and **table** is allocated for pointers to rows, columns, and diagonals, so that we can conveniently cycle through them

playerturn



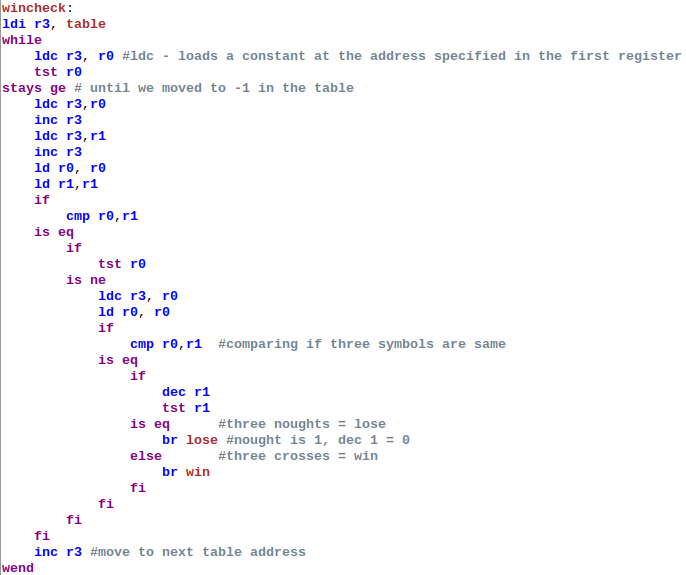
Reads data from the schema that the BPC transmits when the button is pressed, stores the value of the cross locally, and translates the data into a format that accepts SDR (the first 2 bits are symID, the next 4 are coordinates, the bits under GSDD are still nought)

drawcounter



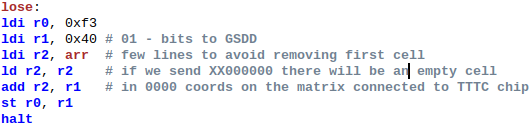
a counter of objects placed on the playing field to make it easier to check the draw condition. Stored at 0x03

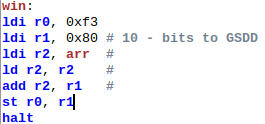
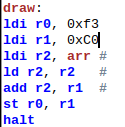
wincheck



It goes through the table and checks for the presence of three crosses or noughts in a row, column, or diagonal, moving to lose or win when the conditions are met. If the specified conditions are not met, then looks at the object counter, if it is equal to 9, then goes to draw

lose/win/draw

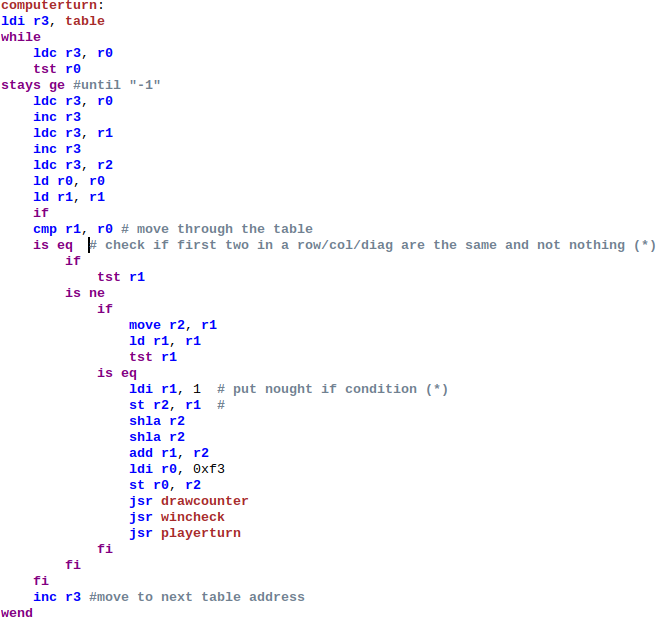




set the 6 and 7 bits that the GSDD chip processes, so that the desired LED lights up

calls the stop command halt

computerturn



The algorithm of the computer player checks whether there are two identical objects in the first two elements of the row/column/diagonal, if this occurs, then the third element is set to nought

If not, the nought is placed in the first free cell.