**21930 HCI NSU**

**“Tic-tac-toe game”**

*Documentation*

Vyacheslav Berezhnoy and Fedotov Dmitriy, 2022

1

**Contents**

* Introduction………………………………………………….....3
* Hardware……………………………………………………….4
* Software………………………………………………………..9
* Resources……………………………………………………...10

2

**Introduction**

The latest achievements of Russian cyber-sportsmen have shown the importance of developing the gaming industry in Russia. The variety of gaming tournaments is provided by the number of different games. Today, games serve as a great tool for getting profit through the sale of new titles and peripheral components of the industry, streaming. Games are also a great way to spend your leisure time. To immerse ourselves in the topic of game development, we have implemented the software and hardware parts of the tic-tac-toe game.

3

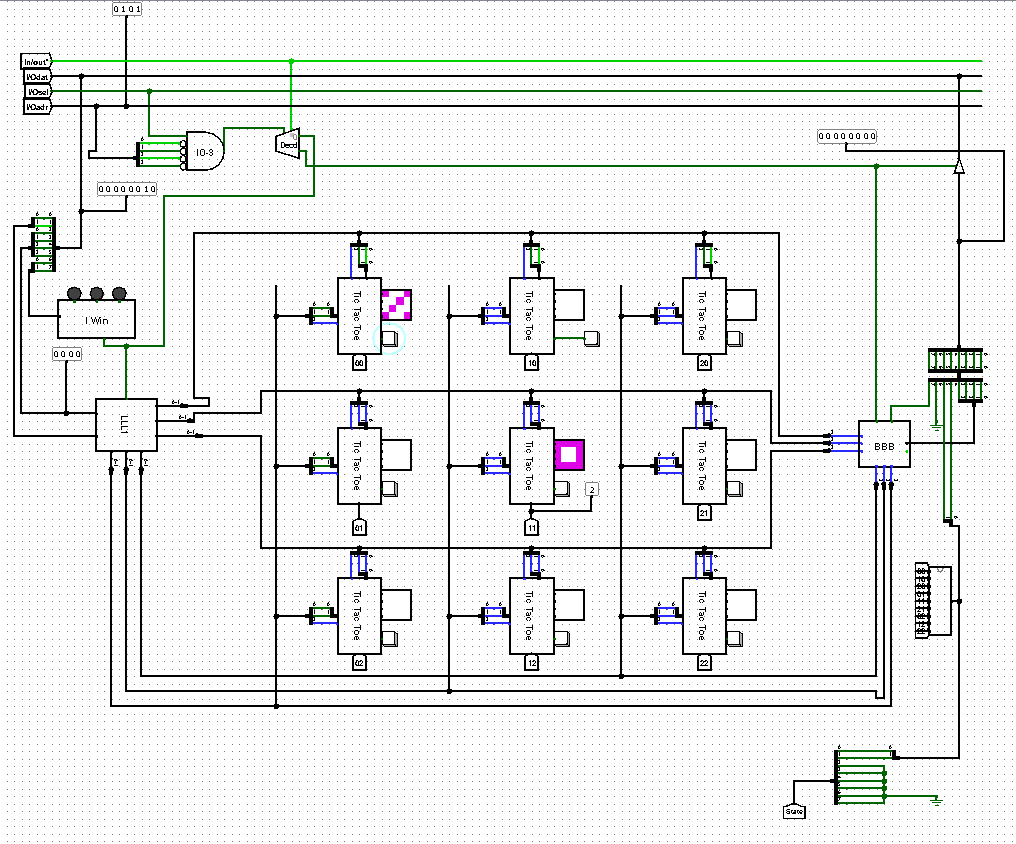
**Hardware**

The hardware of the game is pretty simple. It consists of a CdM-8 full-core processor with an I/O bus, connected to a 9 cell game field.

**Game field**

One of possible implementations of the game field is shown below. It employs three horizontal and three vertical buses, each of them connected to one of the I/O Tic Tac Toe cells. Cells have their own **x** and **y** coordinates presented with a 2-but string, **xx** and **yy**. They correspond to the buses like **xx** = 0b00 and **yy** = 0b10 and conected to **H0** and **V2** buses respectively. The **x** and **y** coordinate strings are concatenated into a 4-bit address string **xxyy**.

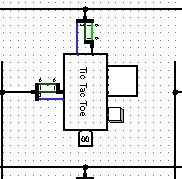
For example, in the diagram below the cell displaying a cross has the address 0b0000, and the displaying a nought has the address 0b0101.



4

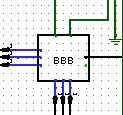
**Individual cells**

The I/O cells contain a button for input and 4x4-pixel LED display for output. I/O is controlled by a **Tic Tac Toe chip** (**TTTC**), connected to the buses.



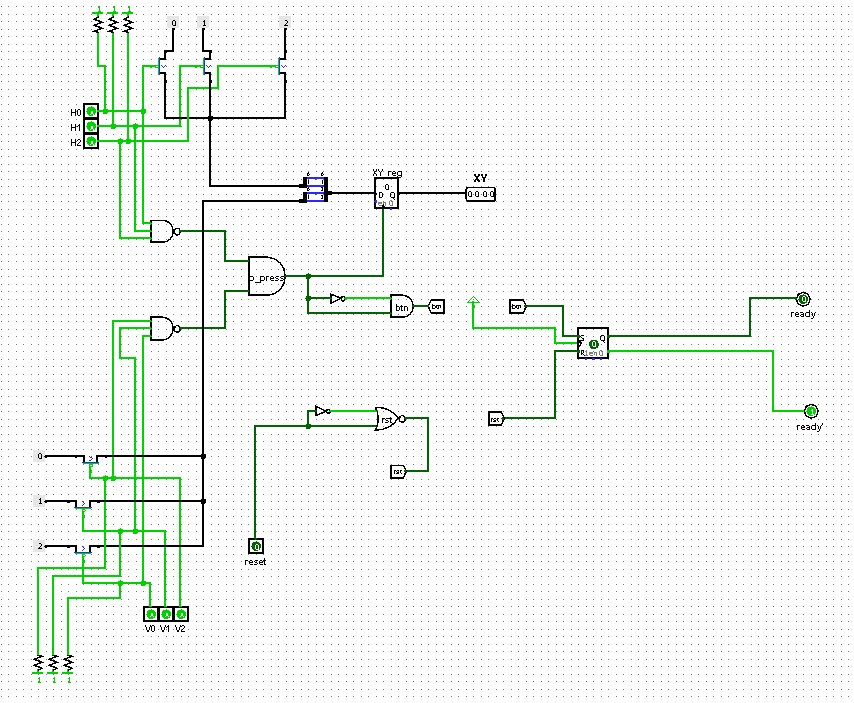
**TTTC** chip has 3 inputs 2-bit hin and vin for carrying in the coordinates, and 1-bit btn for button input. Also it has 2 1-bit output pins hout and vout pin for transferring coordinates to the **Button Press Capture** chip and 4 4-bit output pins for displaying a symbol on the display. Symbols are stored as 4-bit strings in ROM.

**Controller chips**

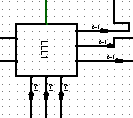


**BBB** chip is a **Button Press Capture** chip and has 6 pins connected to bit 2 of one of the buses. It holds addresses of the cells where the button was pressed. The 4-bit register inside latches the address of the pressed button. Then the chip routes this data to be processed by CdM-8.

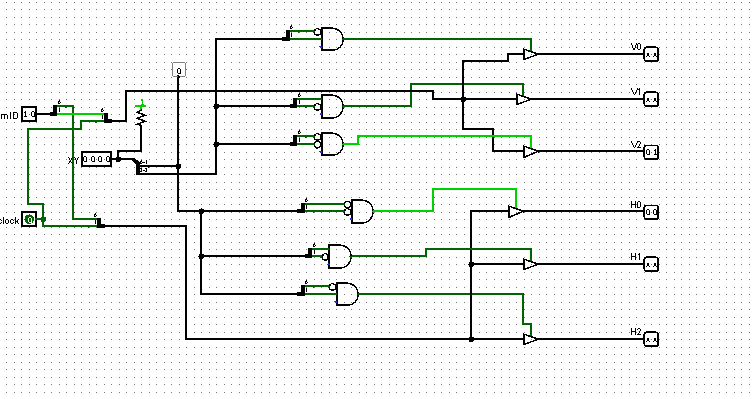
5



**LLL1** chip is a **Symbol Display Router**. It sends an ID of a symbol to be displayed on the screen to the correct **TTTC**. The ID of the symbol is an **xy** coordinate coming from the processor.



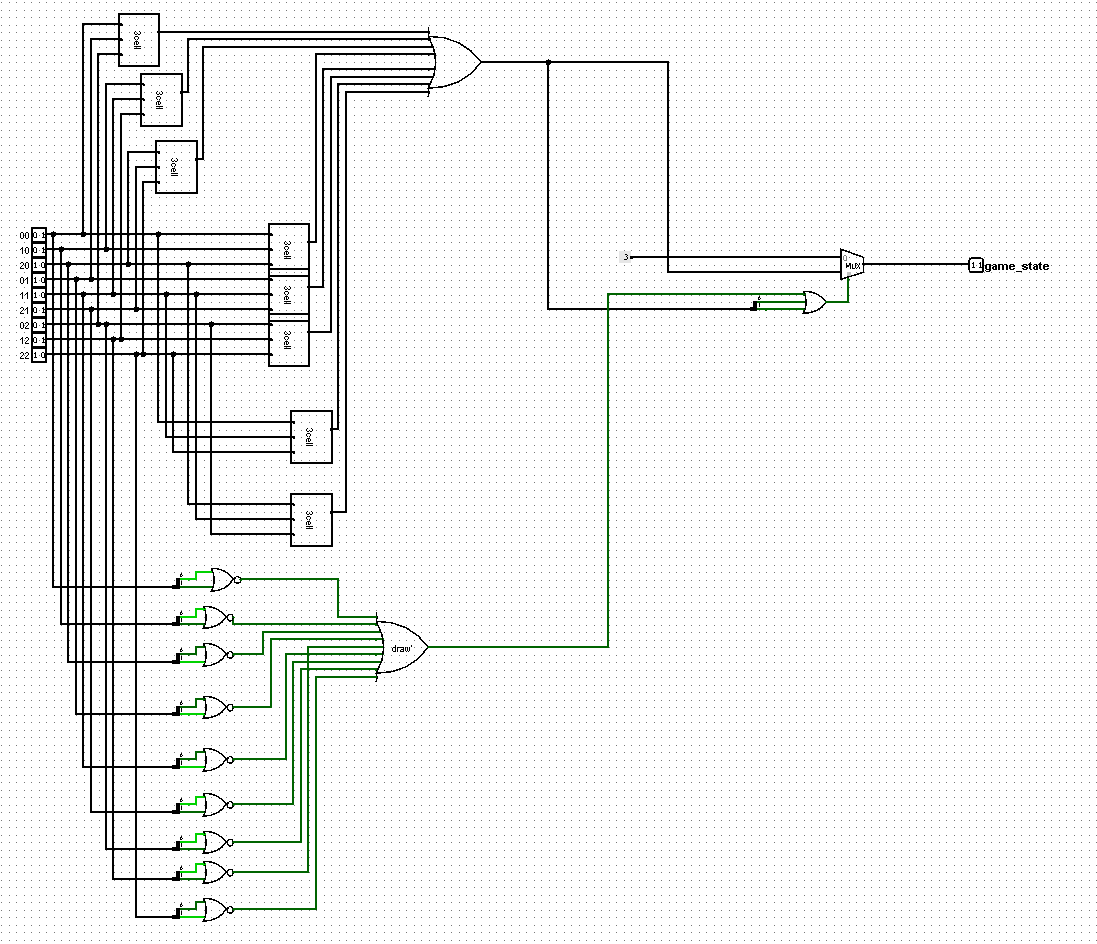
In the picture below, you can see that the **SDR** chip has 6 2-bit output pins that are connected to bits 0 and 1 of one of the buses. To write a symbol processor sends a 2-bit symbol ID and the cell address to the **SDR**. Then the chip decodes the cell address and sends the symbol ID on the correct bus.



6

**Game state**

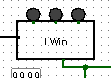
Game state is controlled by a **state** chip. It gets IDs of symbols in each cell and checks if there are any triplets in any column, row or diagonal. The ID of each cell is compared with the other two in **3cell** chips. Then it sends a 2-bit signal if it is a win(1), lose(2) or draw(3).



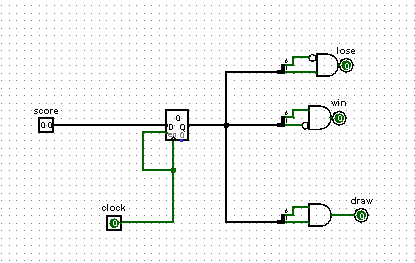
**Indication**

**Game State Display Driver** (**I Win**) is a chip which lights an LED indicating the game state: win/lose/draw. During the game the processor sets bits 6 and 7 of the **I/Odat** bunch to 0. After the last turn CdM-8 sends a 2-bit signal on these two wires which identifies the LED to be lit.

7

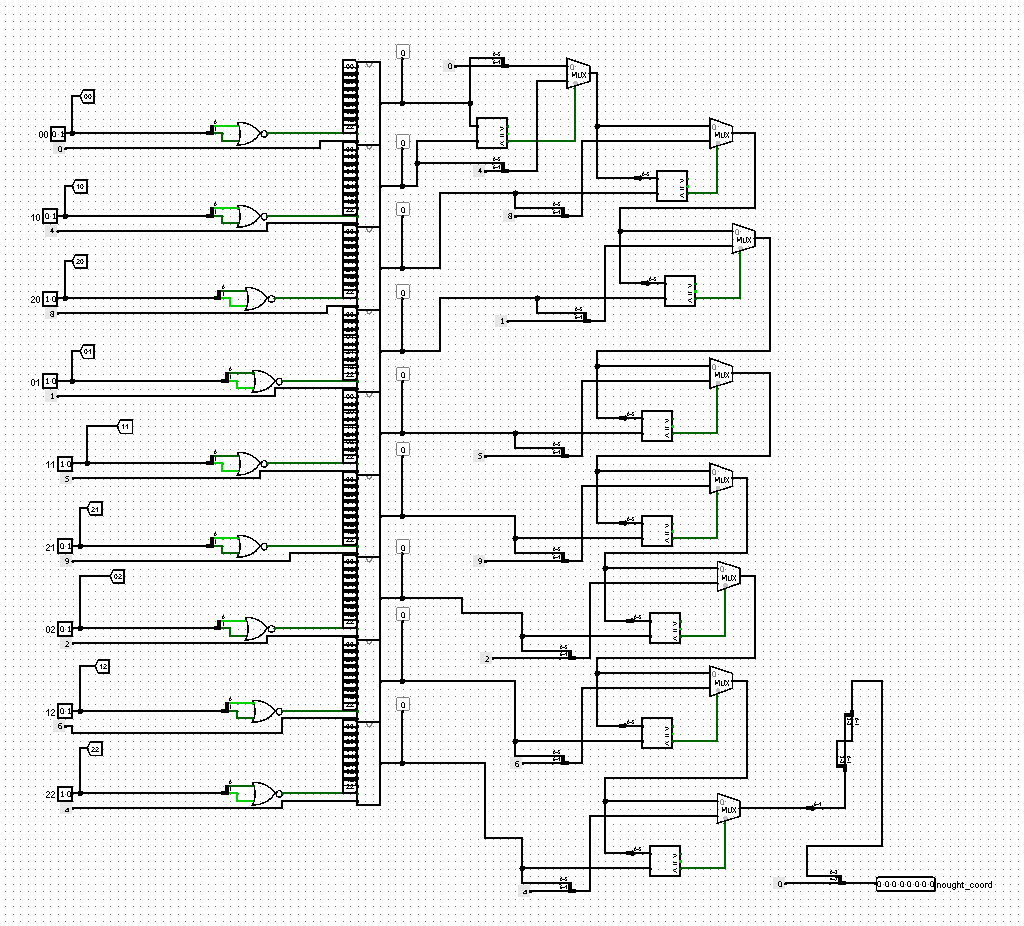


The chip has 2 inputs: 2-bit score input and 1 bit clock input. Outputs are presented as LEDs that light up in different colors.



**Computer logic**

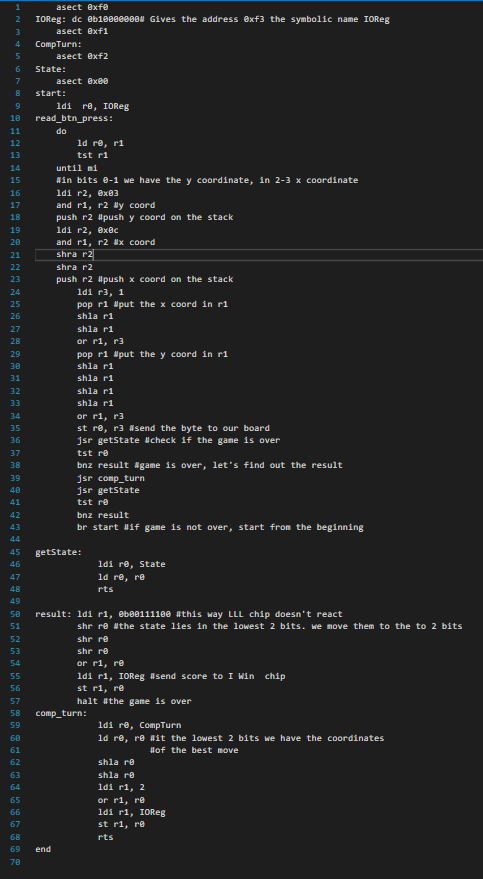
Turns of the computer are controlled by a **CT** chip. It gets all symbol IDs as input and goes according to algorithm\* transferred into a logical circuit.



\*See resources

8

**Software**



Assembly code is not overcomplicated. Main loop does this:

* Waits for button press
* Sends signal to I/Odat to place a cross
* Checks if the game is over
  + If not:
    - Loads the robot turn
    - Sends signal to place a nought
    - Checks if the game is over (if not the algorithm repeats)
* if yes:
  + Sends a signal to the I/Odat to light the LED

9

**Resources**

Here is the list of materials and resources we used:

* Lectures and circuitry materials: <http://ccfit.nsu.ru/~fat/Platforms>
* Computer Logic: <http://www.micsymposium.org/mics2016/Papers/MICS_2016_paper_28.pdf>
* CdM-8 processor: <https://github.com/leadpogrommer/Cdm8-asm>
* Group project documentation: <https://docs.google.com/document/d/1jfoV9fDeG4o3HlbWwSF_qu0PiGA-_jcPokDnFy5pYG0/edit>
* Irtegov D.V. and Nazarov A.D.

10