Federal State Budgetary Educational Institution
of Higher Professional Education
"Novosibirsk National Research State University"
Faculty of Information Technology

Co-design Group project

“The Game of TV-tennis”

Completed:
A. A. Potapova
(1st year, group 21216),
P. I. Dzyuba
(1st year, group 21216)

Novosibirsk
2021

CONTENT

OVERVIEW

Our team consists of Peter and Anna. We decided to create one of the 3 projects offered to us, among which our choice fell on TV-tennis, because it seemed to us the most suitable for further changes. To create it, we had to use Logisim and CDM-8.

TV tennis (one of the names is pong) is one of the first and most famous games in the world. The essence of the game is to move the bat and hit the ball on the opponent's field. The round is considered won if the ball touches the opponent's wall.

Our project has acquired its own personality thanks to the improvements and additions we have made.

So, the movement of bats seemed very limited to us, in the end we decided that our bats would be able to move not only vertically, but also horizontally for some distance.

The graphics seemed rather boring to us, so we were looking for methods to modernize it. In the end, we decided to implement a flashing screen when any of the players lose. This idea came to us from retro games where the character blinked when taking damage.

To add a surprise effect for each of the players, we also decided to randomly change the angle of reflection of the ball by Y to make the game more exciting.

After selecting the main basic mechanics of the game, we started developing the program and its basic principles.

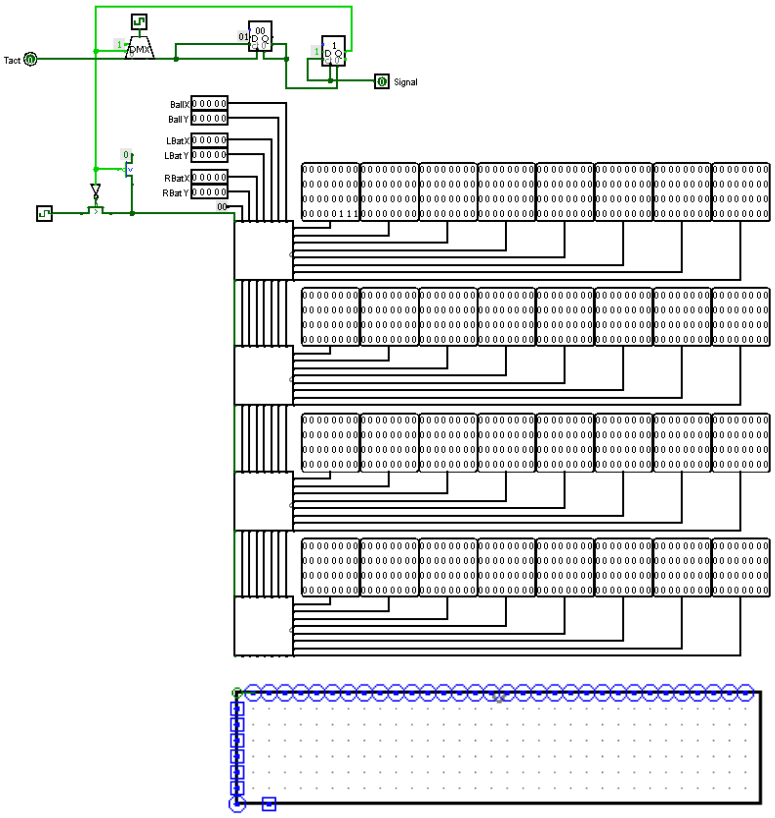
STAGES OF DEVELOPMENT

Stage 1. Program design and operating principles

Video subsystem

Input	Description
ballX	Ball coordinate (for video subsystem)
ballY	
LBatX	Left bat coordinate (for video subsystem)
LBatY	
RBatX	Right bat coordinate (for video subsystem)
RBatY	
Signal	Signal to turn on the animation of the loss

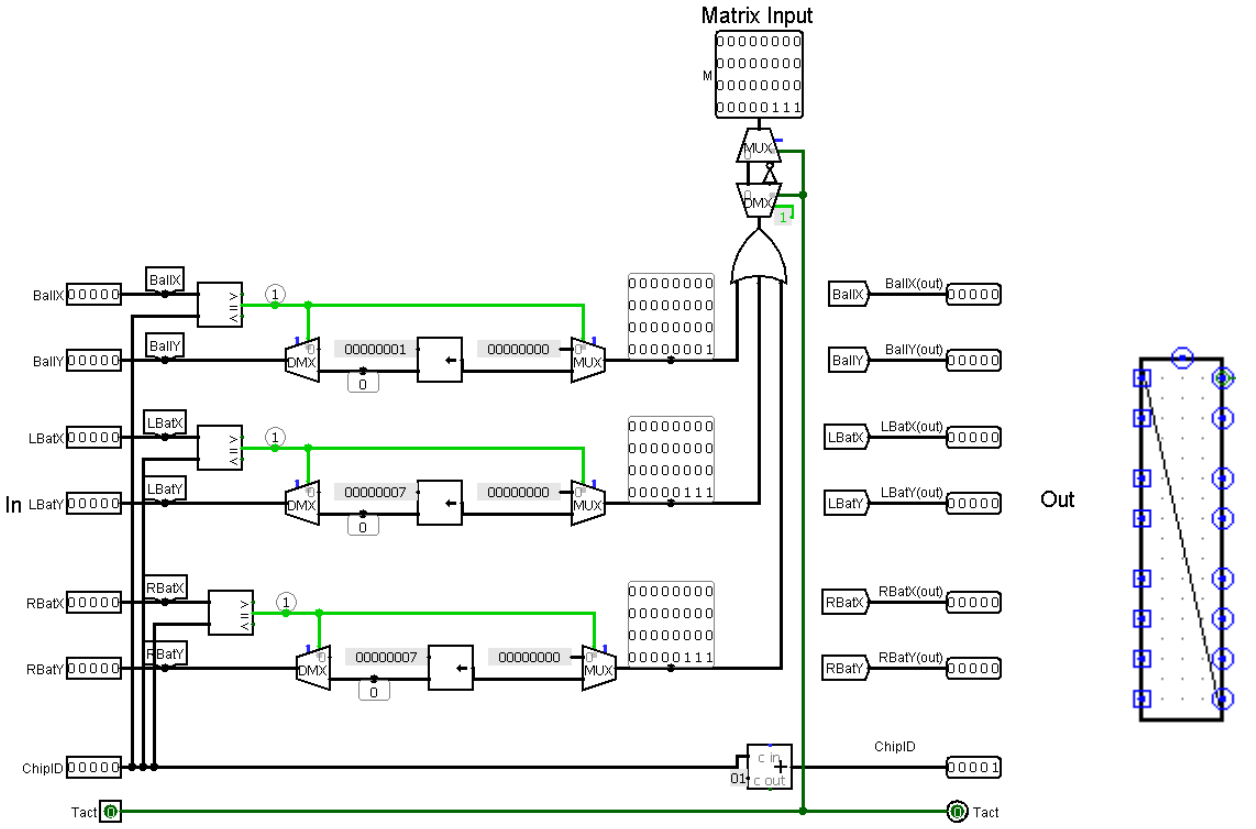
Output	Description
Tact	The value transmitted by the clock generator after animation



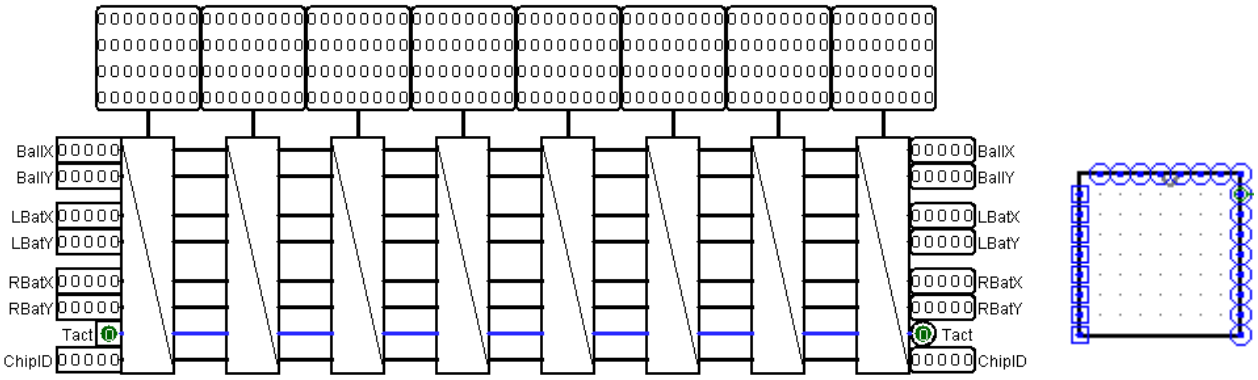
In order to display the image of the bats and the ball on the screen, a video subsystem was first created. Since our field is quite large (32x32 pixels), we connected the chips with a chain and used a cellular structure.

Each cell is responsible for one column of pixels and has its own identifier. Each chip at the input takes the coordinates of the bats, the ball, and the ID of the previous chip, increased by one, and transmits this data to the screenshot from the opposite side.

The chip looks like this:



Eight such chips are combined into one:



And 4 such cells are combined into a video subsystem. This design is easy to use and allows you to move the circuit freely.

Also, a clock switch was built into the video subsystem, which at the end of the round changes the flow direction from the output to the counter. After overflow, the counter sends a signal to the switch, and the flow switches to its original position.

This way we get a little time to play the flicker animation.

Creating a bat

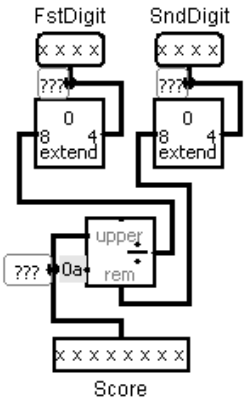
Initially, we decided to just connect the joystick directly to the bat, so that the joystick changes the coordinates of the ball. Despite the quick response, it was inconvenient to use - it was impossible to fix the coordinate of the ball.

For this reason, we decided that the joystick will only set the speed of rotation of the bat, and in the kinematic processor it will be processed and its coordinate changed.

Game score

Input	Description
Score	Points of this player

Output	Description
fstDigit	First or second digit of a number in decimal representation
sndDigit	

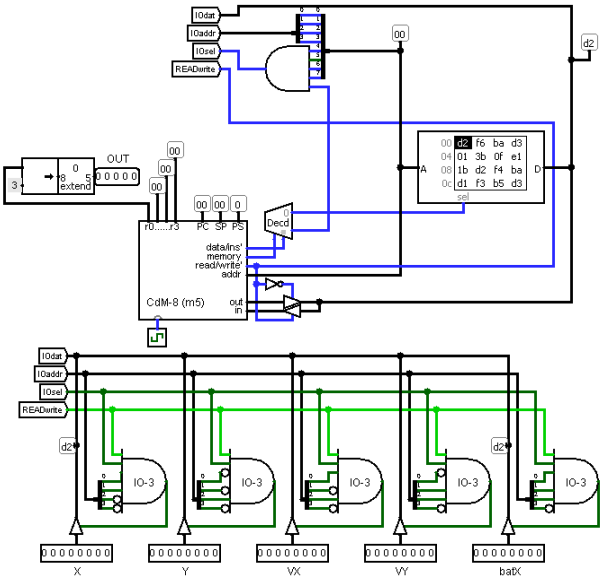


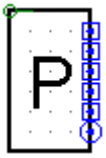
This is a small scheme that stores the number of wins a player has and transmits it to 2 hexadecimal indicators. Since this scheme is designed to display hexadecimal numbers, we had to adjust this format for decimal numbers.

Second player

Input	Description
X	Ball coordinates
Y	
VX	Ball speed
VY	
batX	X bat coordinate

Output	Description
OUT	Y bat coordinate





To create the second player, we needed to use CDM-8, as well as our knowledge of assembly language.

Due to the fact that we added the ability to move the bat horizontally, we had to significantly complicate the algorithm for calculating the position of the bats. After analyzing the situation, we decided to randomly select the x coordinate and calculate the y coordinate using the data obtained by CDM-8. Thus, we killed 2 birds with one stone: we solved the problem with calculating coordinates and added randomness to the game, which is good for the gameplay.

The y-axis coordinate of the bat is calculated in CDM-8 as follows:

The robot is able to determine where the ball will be on Y so that the rocket can hit it.

The ball coordinate in Y and X, the speed of the ball in Y and a random number that determines where the racket will be located in X are transmitted to the registers.

The formula by which the robot determines where the ball will be looks like this:

y + (xrightbat - x) * (vy / vx)

If the ball hits the wall before colliding with the racket (that is, when calculating "y + (xrightbat - x) * (vy / vx)" overflow occurs), the formula looks like this:

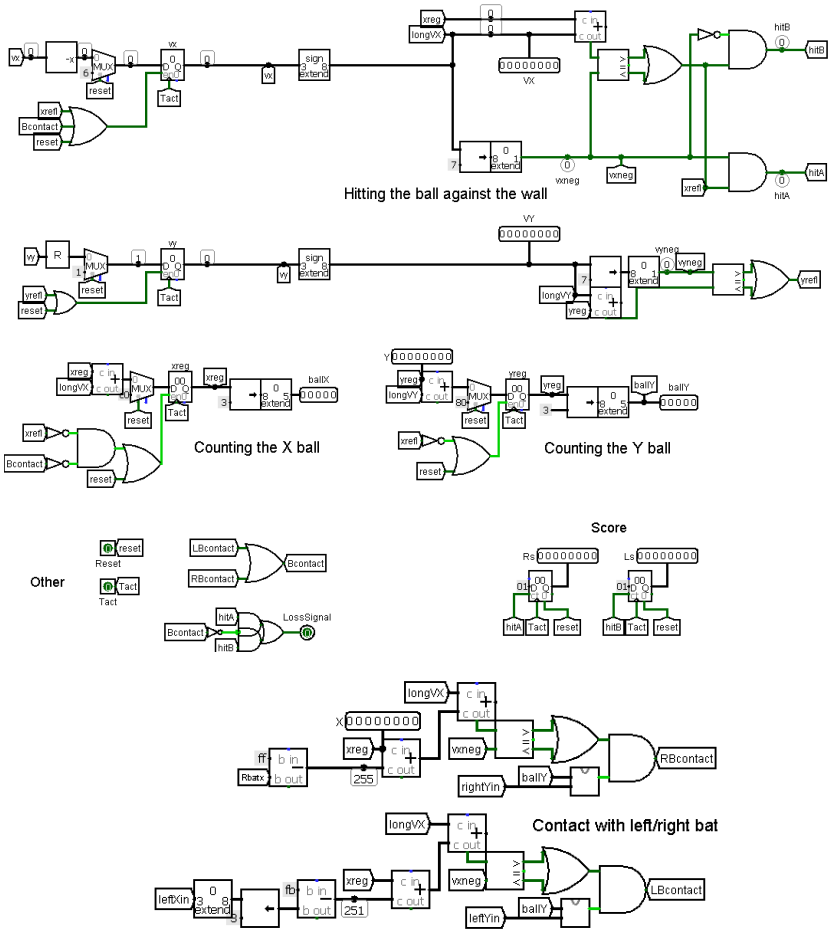
256 - (y + (xrightbat - x) * (vy / vx))

In our project, the velocity of the ball in X is constant 2. The robot itself consists of a system of if conditions - each value of vy has its own if block.

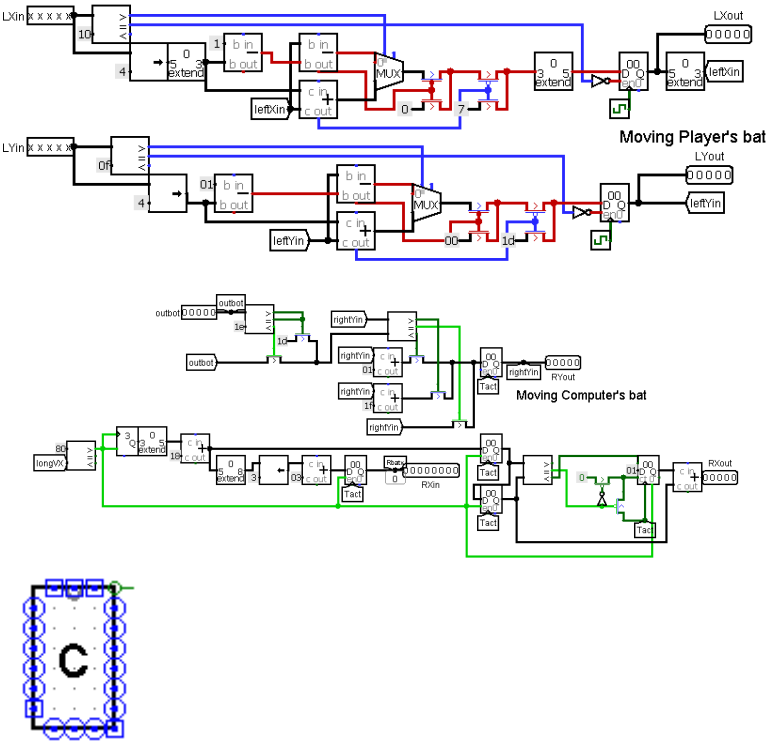
Kinematic controller

Input	Description
LXin	Output values from the joystick (the speed of movement of the left bat along the corresponding axes)
LYin	
Reset	Reset
Outbot	Coordinate of the right bat in y
Tact	The value transmitted by the clock generator

Output	Description
ballX	Ball coordinate (for video subsystem)
ballY	



LXout LYout	Left bat coordinate (for video subsystem)
RXout RYout	Right bat coordinate (for video subsystem)
RXBot Out	X bat coordinate (for second player processor)
X Y	Ball coordinate (for second player processor)
VX VY	Left bat coordinate (for second player processor)
Ls Rs	Score of the first or second player
LossSignal	Signal to turn on the animation of the loss



The task of the kinematic controller is to move the ball, the left and right bits, updating their coordinates.

The movement of the ball is carried out by adding a certain number to its X and Y coordinates, which is called speed.

The speed consists of two components - X and Y.

The velocity modulus in Y changes when colliding with something, the velocity modulus in X is constant.

When colliding with an upper or lower wall, the velocity in Y is inverted, when colliding with a bat, right wall or left wall, the velocity in X is inverted.

All changes in the coordinates of the ball are taken into account in a field of 256x256 cells . After calculating the coordinate change, it is converted to the corresponding coordinate in the 32x32 cell field (that is, from an 8-bit number to a 5-bit number with the preservation of the higher bits) and transmitted to the video subsystem.

The movement of the left bat is also implemented, except for the processing of coordinates in an 8-bit representation. Its coordinates are processed in a 5-bit representation.

The speed of the left bat changes depending on the joystick - in which direction the slider will be directed, in which direction the bat will move.

The movement of the right bat is carried out using coordinates. A random number is selected by X, and by Y - calculated by the robot. The bat gradually moves to this point, adding 1 to the current coordinate on each iteration.

Stage 2. Game development and challenges

During development, we encountered many problems: from a complete lack of understanding of the design of circuits and how to implement a certain thing or method.

For example, we faced a big problem with calculating the bounce of the ball from the bat. In game Development, any object subject to the laws of physics has a Rigidbody collider (hereinafter referred to as the “collider”), which, as it turned out, we did not move. Making a visual movement of the bat on the x-axis was not such a difficult task, unlike the collider. The problem was that this case was calculated using a variable overflow only for one position of the ball and was written as a constant. Therefore, we had to delve deeply into the counting process and change the value passed by this constant for ourselves.

Difficulties arose with the creation of the robot. We didn't understand how to make the processor count both coordinates for the bat, so we changed the approach and generated the X coordinate randomly.

Initially, we wanted to change the speed of the ball along the x coordinate, but later we abandoned this idea in favor of the speed of the processor.

Stage 3: Testing and debugging

Testing the game turned out to be the least difficult stage of our work, since during Testing the game turned out to be the least difficult stage of our work, since during the development of the game we consistently tested each new scheme.

During testing, we identified problems such as the bat flying out of the field, incorrect ball movement and mismatch of the position of the bat and its collider.

We have successfully solved all these problems.

In addition, testing played a very important role in fitting the values to create a flashing field when losing. This is how we selected the optimal value for a comfortable game.

OUR INNOVATIONS

After analyzing the innovations of other teams, we decided that this path is not for us: someone made different levels of difficulty, someone made a huge variety of angles at which the ball can move. We decided to do something innovative: this is the ability to move the bits horizontally.

This gave the player the opportunity to choose a style of play. A person can play fast and “aggressively”, moving the bat closer to the center and performing unexpected movements, or carefully calculate his move and have a side chance to stop the ball.

This innovation also added more randomness to the game - the second player randomly selects the X coordinate, which makes his actions more unpredictable for the player. Random deflection of the ball by Y during reflection also enhances this effect for both sides.

We also added a flashing animation to the game during a loss. At this point, the round ends. This gives the player time to rest or analyze his victory or loss.

CONCLUSION

So we have an exciting game of TV-tennis. We went through all the stages of software development and got an exciting and unique game in the Logisim environment using the CDM-8 chip and assembler.