Novosibirsk State University

**4COM1042 [Computing Platforms]**

**Co-design Group**

# Project B “The Game of TV-Tennis”

Anton Pimonov 21931 & Mikhail Komarov 21931

2022

**Contents**

1. Introducton
2. Overview
3. Hardware
4. Software
5. Conclusion

## **Introduction**

For our group project we were able to choose one theme from three basic ones or come up with something original. We chose TV-Tennis, because we didn’t find other two basic themes interesting and because we think TV-Tennis great, yet simple example of using everything we have passed through the program of our course.

As you can see, we successfully implemented TV-Tennis on Logisim + CdM-8 platform (Logisim for circuits, CdM-8 for code).

We divided our showcase in 3 parts: overview, hardware and software.  
We’ll begin with overview.

## **Overview**

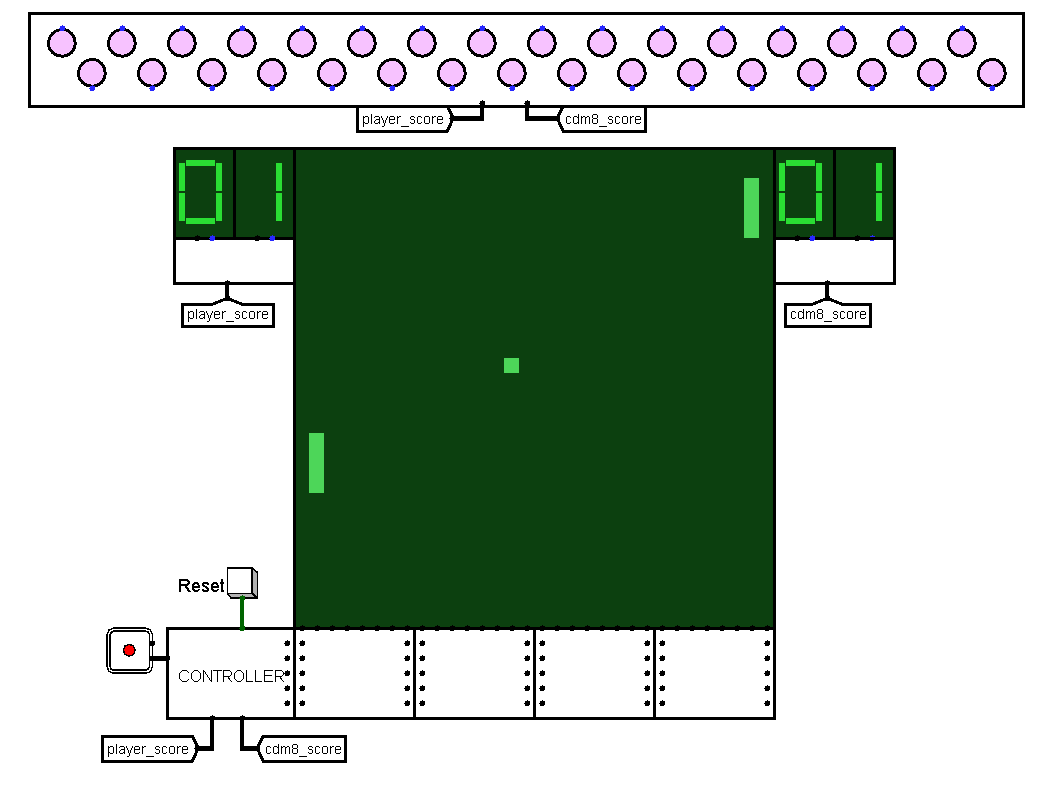
Let’s split components that player is able to see on “user” and “technical”.

“User” components: video display, joystick, restart button, score.  
These pieces have meaning to player, he is able to see what happens because of display and score, to control the bat because of joystick, and to restart the game because of restart button.

“Technical” components: video chip and kinematic controller.  
These pieces do all work, this is where everything being calculated (ball movement, bats movement etc). It has nothing to do with the player, he just does the inputs and get the results, he has no need to see or understand what happens between those things.

All of this you can see on the next page (main circuit, actually).

Now let’s move to the Hardware part.



## **Hardware**

### Display

What do we need to see on display? The ball and the bats.

How can we display it? By deciding what the ball and the bats are.

The whole display – pixel panel, 1024 pixels. In fact, this is 32 columns of pixels, counted from 0 to 31. Each column has a 32-bit input pin. If Nth bit is 1, Nth pixel turns on.

Now we can represent the ball by just turning on any pixel on the display, because the ball is just a single pixel. The player bat – three pixels in 1st column, player is able to move it up and down. The bot bat – three pixels in 30th column, same rules of movement, just like to player.

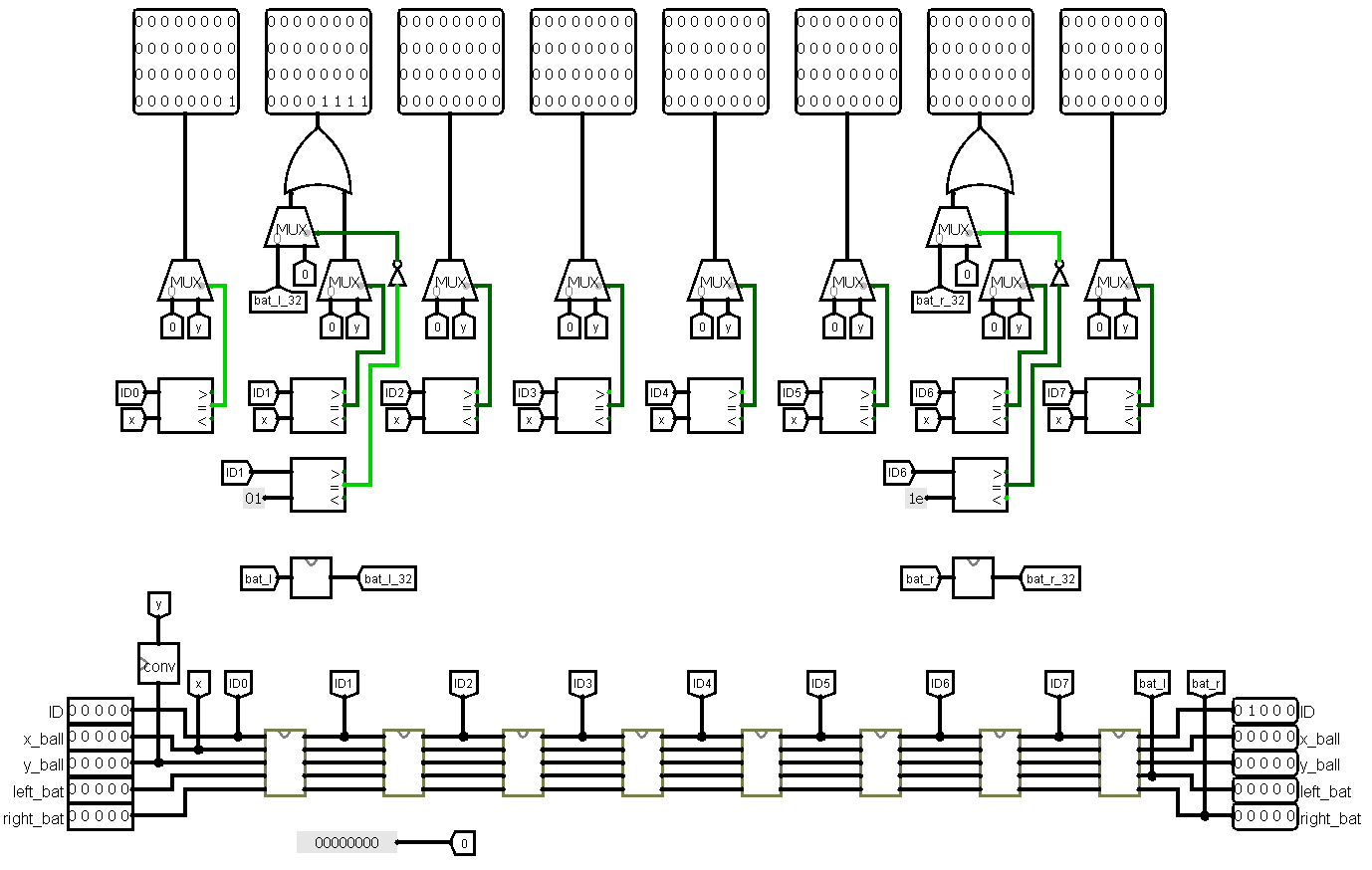
But it’s not enough to display the ball, we need to make it move. For this we need to know X and Y coordinates of the ball right now and the speed of the ball (speedX and speedY). This information represented by 5-bit numbers. More about it will be in “kinematic controller” part.

### video\_section

Inputs: ID, x\_ball, y\_ball, left\_bat, right\_bat (5 bit)

Outputs: ID, x\_ball, y\_ball, left\_bat, right\_bat (5 bit), 8 column outputs (32 bit)

There are 4 circuits like this under display, each one has 5 5-bit input pins, 5 5-bit and 8 32-bit output pins, connected to columns. 5-bit inputs and outputs carry information about column number, X and Y coordinates of the ball and coordinates of the bats. If X coordinate of the ball and column number are equal, bit equal to Y coordinate of the ball goes 1 and turns on the pixel. This is how we display the ball. There is also bat-check: if column number equal to 1 or 30, we are turning on 4 pixels, position of lower one we get from input, other ones are just 3 pixels above.

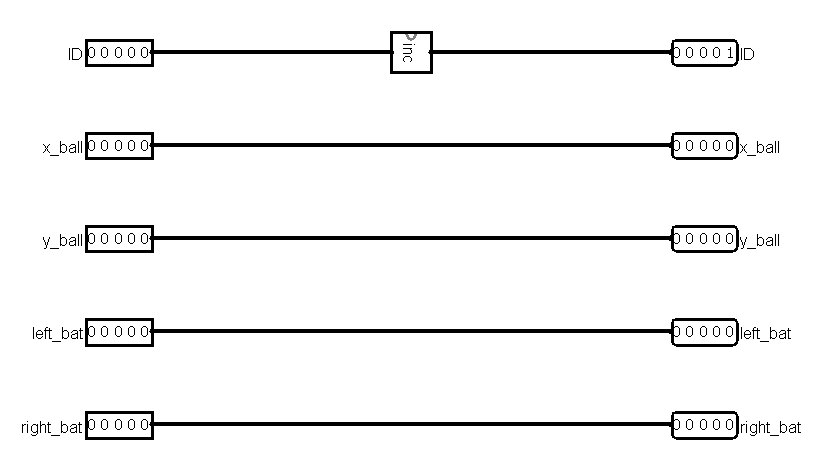


### video\_chip

Inputs: ID, x\_ball, y\_ball, left\_bat, right\_bat (5 bit)

Outputs: ID, x\_ball, y\_ball, left\_bat, right\_bat (5 bit)

This little circuit has 5-bit inputs and outputs identical to video\_section circuit. The only task of this circuit: it increases ID (column number) by 1. By doing this, we change column number and push new value to the next circuit. Other values remain untouchable.



### bat

Inputs: first\_y (5 bit)

Outputs: display\_col (32 bit)

This circuit displays both bats.

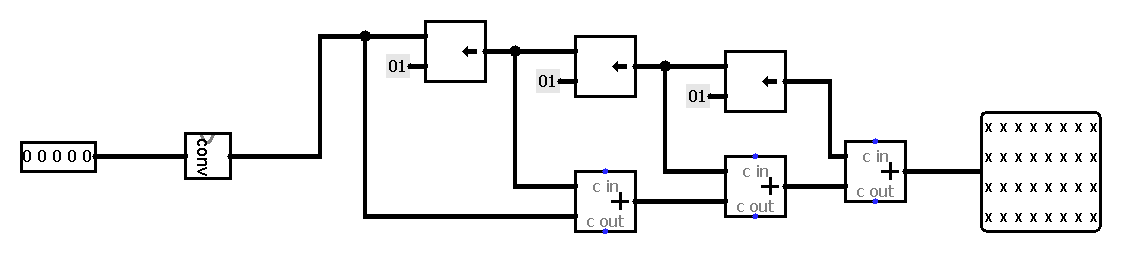
How it works:

Both bats have one characteristic value - the coordinate of the lower pixel.

First of all, we shift ‘1’ on n bits (n = first\_y) to calculate the lower pixel.

Then, we shift this value 3 times to calculate the other three bat’s pixels.

In the end, we just add all counted values to get a full bat.



### kinematic\_controller

Something I dunno

### ball\_move

Inputs: crash\_x, crash\_y (1 bit), vx, vy, ball\_x, ball\_y (8 bit)

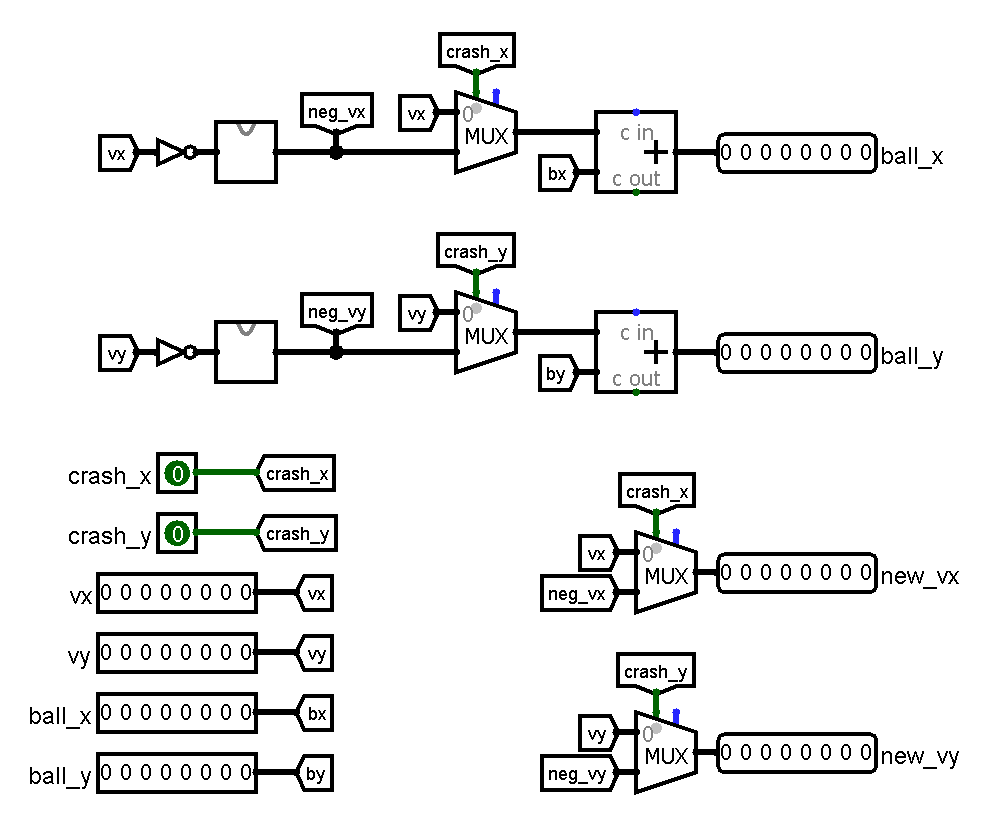
Outputs: new\_ball\_x, new\_ball\_y, new\_vx, new\_vy (8 bit)

Ball move is pretty simple. Every tick current velocity is added to the ball’s coordinate, i.g.

new\_ball\_y = ball\_x + vx

new\_ball\_y = ball\_y + vy.

When crash\_x or crash\_y is high, we take the corresponding velocity with the opposite sign.



### bat\_move

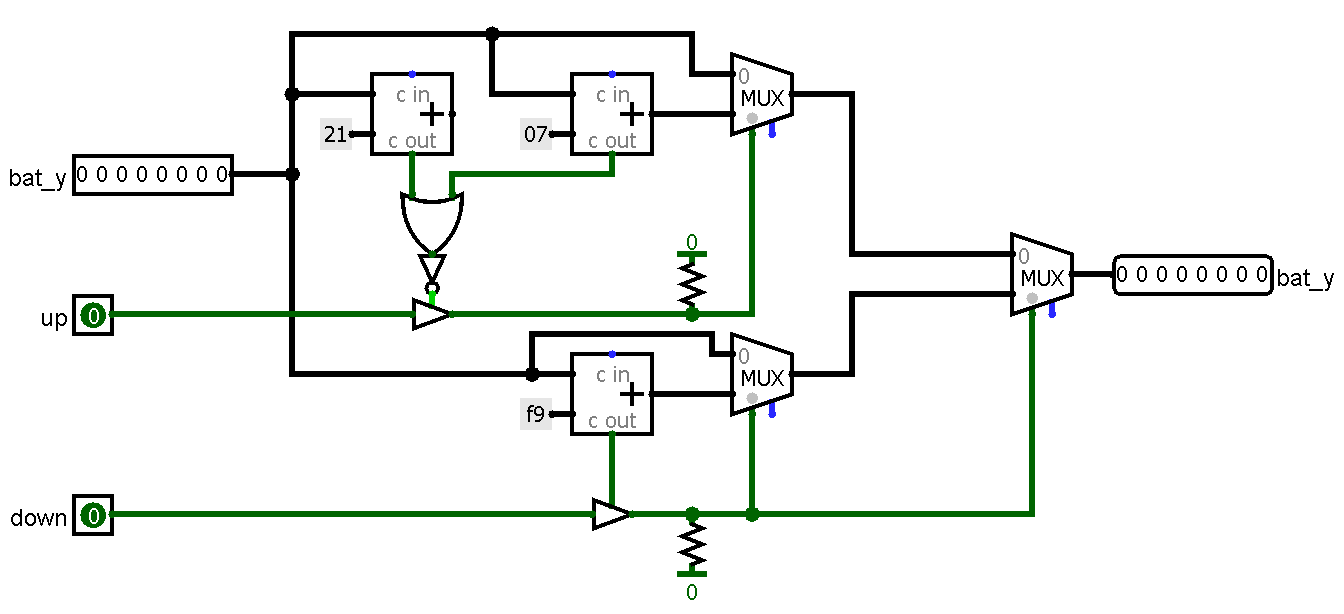
Inputs: bat\_y (8 bit), up, down (1 bit)

Outputs: new\_bat\_y (8 bit)

Bat\_y pin refers to low y coordinate.

When “up” pin is high (i.e., player pulls the joystick up), we add 7 to the current bat’s y coordinate. But we should pay attention to the size of the bat. At the same time, we are adding 21 to the current bat’s coordinate to check a carry bit. If a carry bit is appearing, the bat meets the ceiling, and we should stop the bat moving.

When “down” pin is high, we do similar work, but we should check the disappearance of the carry bit, because adding positive y-coordinate to negative velocity always produces carry.



### cdm8\_bat\_move

Inputs: current\_ball\_x, new\_ball\_x (8 bit)

Outputs: calculated\_y (8 bit)

This circuit works like player\_bat, but instead of “up” and “down” pins, it has the y-coordinate of cross point calculated by the cdm8. The principle of work is simple: we just add 1/-1 to the current bat coordinate, until it becomes equal to the calculated value.

\*here will be picture\*

### check\_hit

Inputs: left\_bat, right\_bat, vx, vy, ball\_x, ball\_y (5 bit)

Outputs: x\_hit, y\_hit, right\_wall\_hit, left\_wall\_hit (1 bit)

How it works:

With some adders and comparators, we check the situation, is there any kind of hit right now. Actually, this circuit is some kind of predicate: is there hitting the up/down walls, left/right walls or maybe hitting the left/right bat? It may look like something difficult to understand, what is happening on this circuit, but it’s not that difficult at all, to be honest there is 4 very similar algorithms, they work almost identical.

Checking the hit on Y coordinate easier because on X coordinate we should consider existence of the bats.

“picture here”

### check\_carry

Inputs: a, b (1 bit)

Outputs: r (1 bit)

Very little circuit, it helps us to get the moment of hitting the wall, because hitting leads to carrying. We detect this carrying and sending 1 in output signal.

“picture here”

### cdm8

Inputs: ball\_x, ball\_y, vx, vy, player\_score, cmd8\_score (8 bit), reset (1 bit)

Output: cdm8\_bat (8 bit)

In our project, we use cdm8 mark 5 for bot-player.

In this part, we tell you about interacting cdm8 with memory. In software part, we will talk about software implementation.

In this implementation of tv-tennis we use Harvard architecture. Scheme has two memory banks: RAM and ROM.

All read/write processes take place in RAM.

Read:

The circuit has a counter. During the counting, data is written from the kinematics controller to the memory. The counter goes from e0 to e7. It is the addresses of variables in memory. There is a multiplexer in the west of the circuit, which, depending on the state of the counter, sends the necessary data to the RAM.

For example, “cmd8\_score” has 0×E1 address.

Obviously, 0×E1 0×E0 = 1, so, counter state is 1. Then, we send 1 to multiplexer. It sends data from 1-port to RAM by 0xE1 address.

Write:

When the counter stops, the circuit turns off the RAM and turns on the ROM with compiled program. During the running, program change some variables, which located in RAM. Cdm8 load calculated point to “cdm8\_bat” output pin. There is a comparator on the west part of the scheme, that compares the current value of pc-сounter with 0×2B. In this case, 0×2B is the address of a branch command, which transfers the control to the beginning of program. When pc-counter’s value is equal to 0×2B, we switch memory banks and the process starts again.

The last one, “reset” is used for RAM cleaning.

## **Software**

Software block scheme here, I guess.

## **Conclusion**

As was mentioned at the beginning, we used almost everything that we have been taught throughout whole course “Digital platform”. While working on this project we successfully implemented TV-Tennis, trained our skill in work with assembler code and creating circuits. We also found this project fun and interesting, but importantly, this project have shown us how important setting and following a plan and thoughtful separation of work between groupmates.