Novosibirsk State University

The Informational Technologies Faculty

Computing Platforms Course

The Game of TV-Tennis

Project Group 6

Panfilov Vyacheslav - group 20213

Malov Alexey - group 20214

Ivanov Oleg - group 20215

Novosibirsk, 2021

Contents

Brief Overview

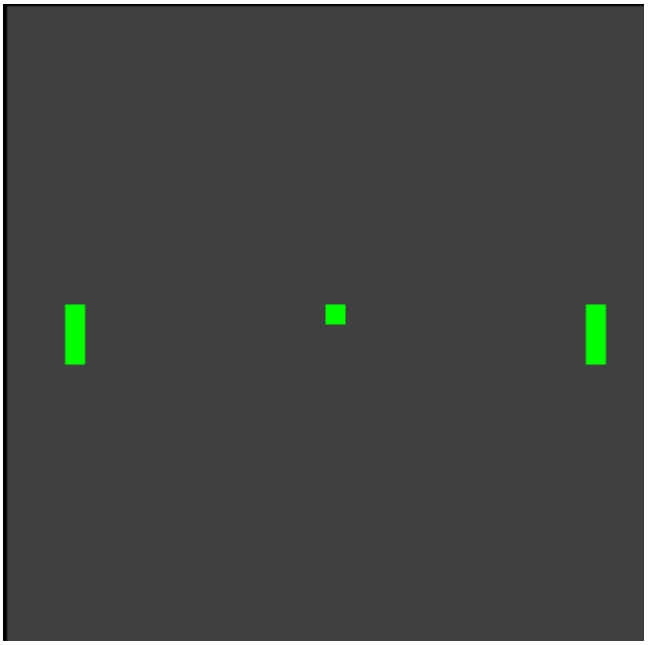
Our group consists of three people: Vyacheslav, Alexey, Oleg. We decided to choose project B, as it seemed to us the most interesting to implement. To complete our tasks, we had to use Logisim and CDM-8.

We decided to split the project into equal parts for everyone. But soon we realized that working on each part of the project together would be more productive, because we could solve this problem together. Thus, we spent much less time on the project than we had planned.

We faced a lot of difficulties and obstacles, which we will discuss next. The whole group worked very closely and each member helped each other.

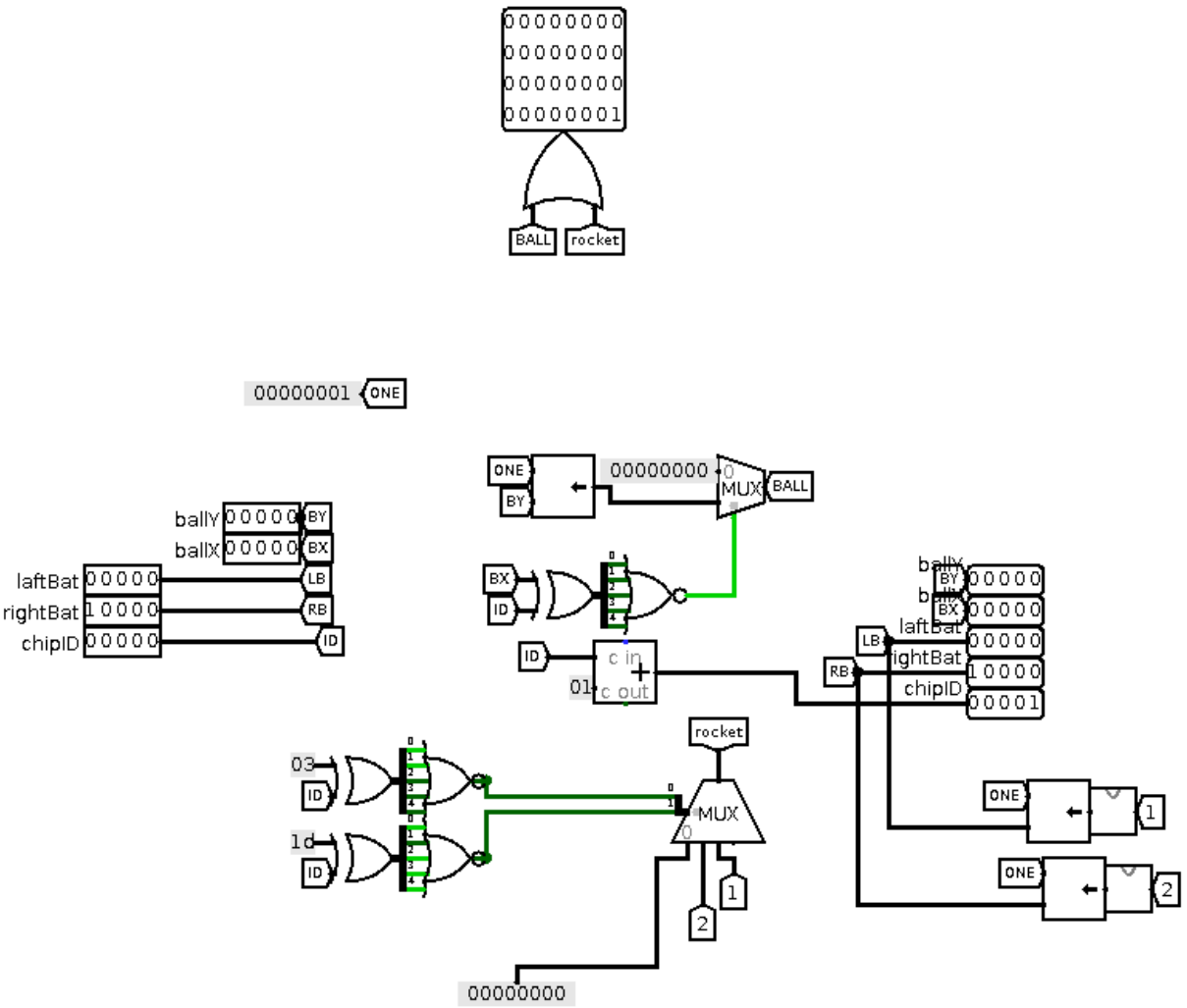
All documentation is divided into several chapters, in which we will talk about the main components of the project.

Let's define rules of the game.



On the screen we have two bats: the left one is being controlled by the player, another is being controlled by the bot. Also, there is a ball. Hitting the wall means scoring a goal: when the ball hits the right ball we increment the score for the left player, in the other way we add for the right player. First player who scored nine goals wins.

Chapter 1. Videosystem

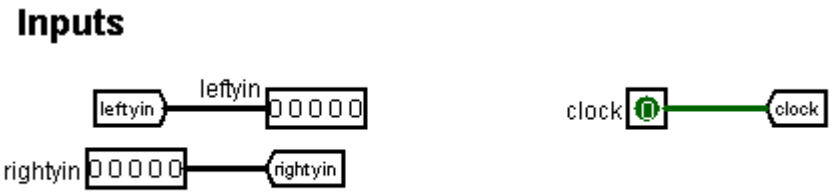


The diagram above shows the chip for the matrix column. In each such chip, we know the chip **ID**, which has a value of 1 more than the previous chip, starting from 0 (*chip **ID** equals the column number, counting from 0*). We also get the **X** and **Y** coordinates of the ball, and the **Y** coordinates of the two rackets.When the chip **ID** matches the **X** coordinate of the ball, we push the **Y** coordinate of the ball on that chip, converting it from **5-bit** to **32-bit** representation. We do the same with rackets, but when converting to a **32-bit** representation we draw a pixel at the top and bottom of the racket, and compare the **ID** with constant values, since the rackets only move up and down. We have 32 such chips, one for each column of the screen (*the screen width is 32 columns*).

Chapter 2. Kinematic controller

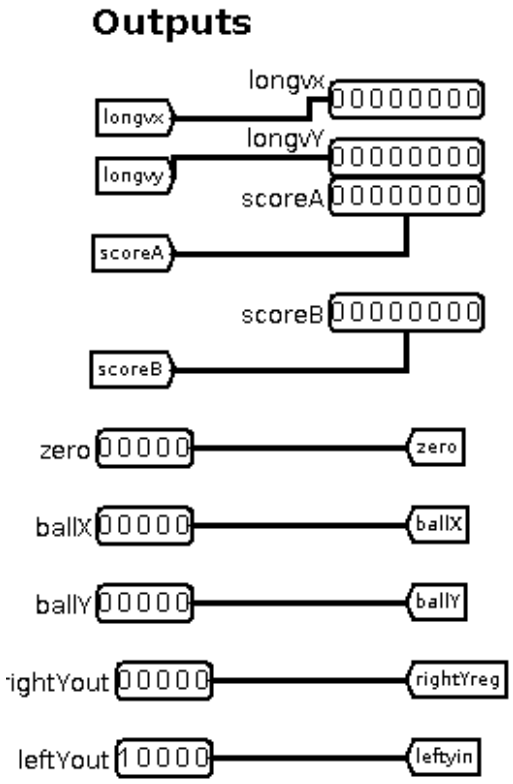
This chip is used to move the ball and bats, keeping scores and wall and bat collision detection. We decided to use tunnels as they make our scheme look good and neat. So, we don't have chaos of wires, schemes and so on. Moreover we could divide controller on 4 blocks:

There is a block with **inputs**:



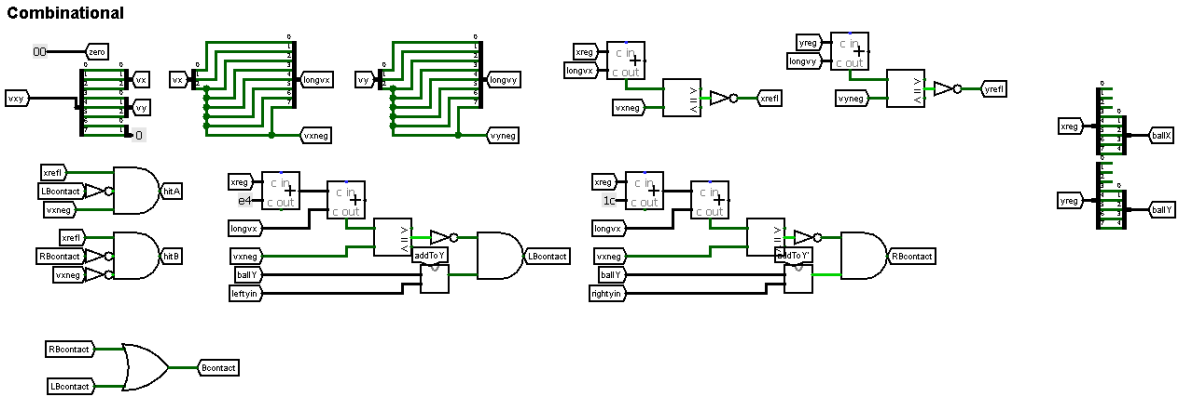
Here we have two 5-bit pins: **leftyin** is for a left bat, it connects to a joystick, **rightyin** is for right bat which connects to CDM-8. The 1-bit pin is for a **clock**.

This is for **output**:

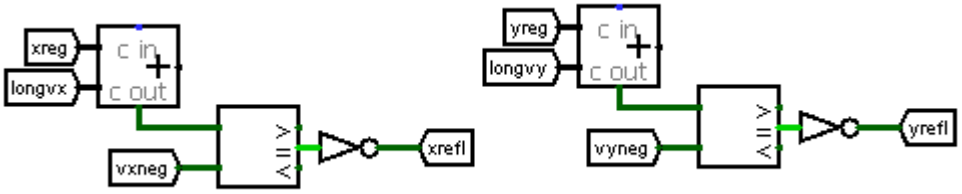


Two 8-bit pins **longVX** and **longVY** are used to send **X** and **Y** velocity to CDM-8. Also, we connected to the processor two 5-bit pins: **ballX** and **ballY** - **X** and **Y** coordinates of the ball.

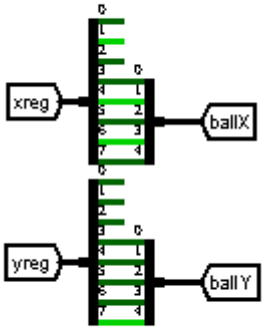
LeftYout and **RightYout** we use to send **Y** coordinates of two bats to the video sector. Also, we send **Zero** to the video sector. **ScoreA** and **ScoreB** are scorekeepers for player and bot. This block is called **Combinational.**



There we have all **hit detectors**.

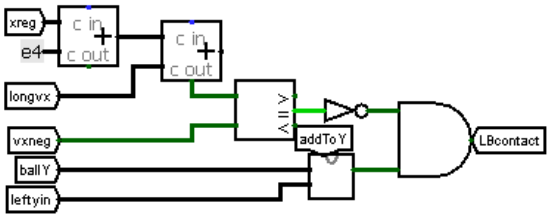


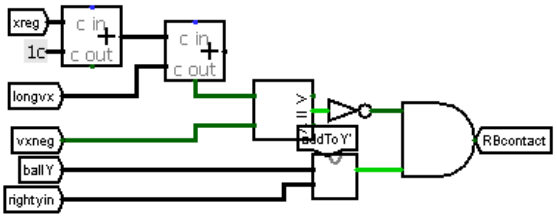
These two are for **edge hit detection**. We get 8-bit **xreg** and **yreg** values, which we get from this two schemas:



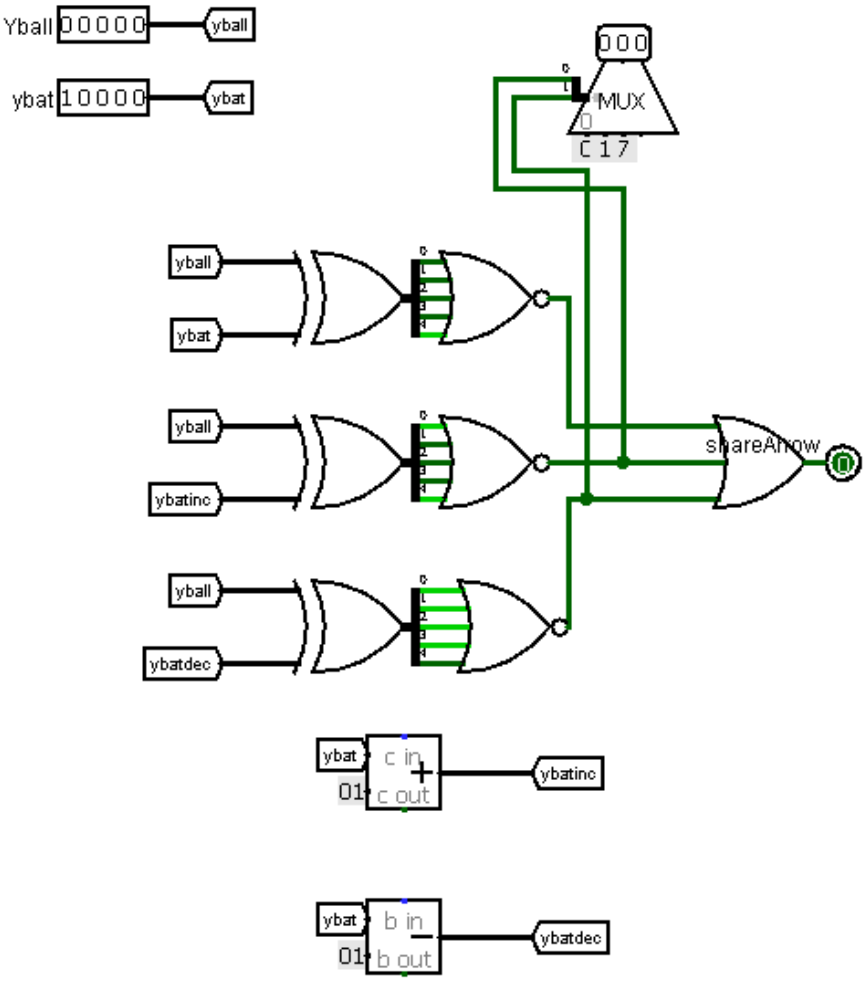
We add them to **VX** and **VY** and then carry out, after that compare that to **vxneg** and **vyneg** (**vxneg** and **vyneg** show the velocity is negative). If these two values are equal, it means that the ball didn't hit an edge, otherwise we will change the velocity of the ball.

These two schemes we made for **detecting if the ball hit a bat**:

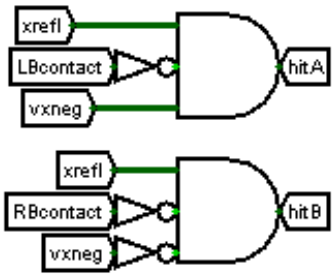




There we have **the first extension**: you can choose game mode between normal and crazy. About that extension you can read here.
There we check: has the ball hit the bat:

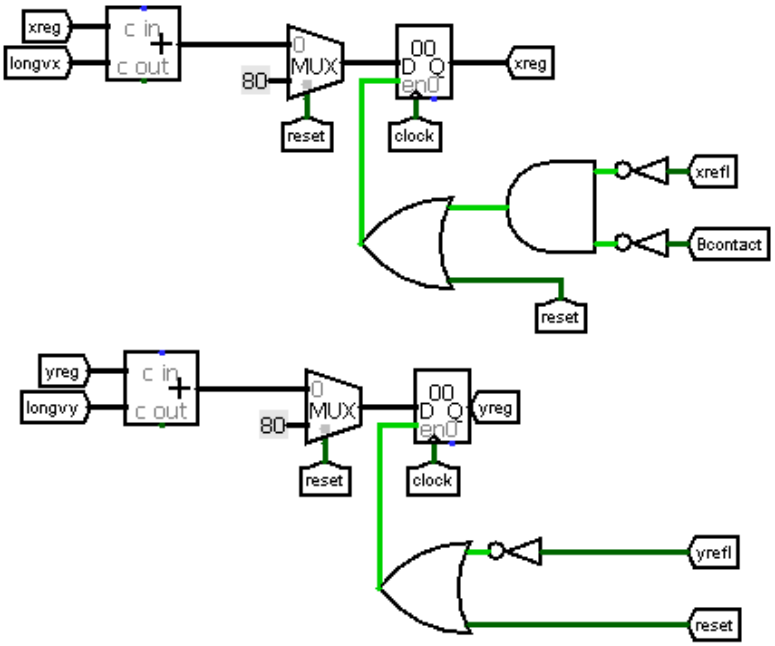


And finally, **goal detectors**:

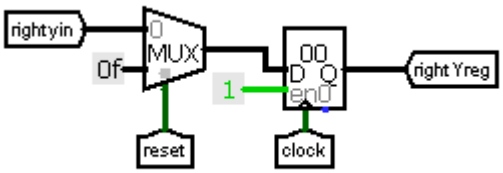


If the ball hit the wall && ball didn't hit the bat && VX.

The last block is called **behavior**. It is the main part of the controller, there we move the ball and the bats by updating their coordinates.

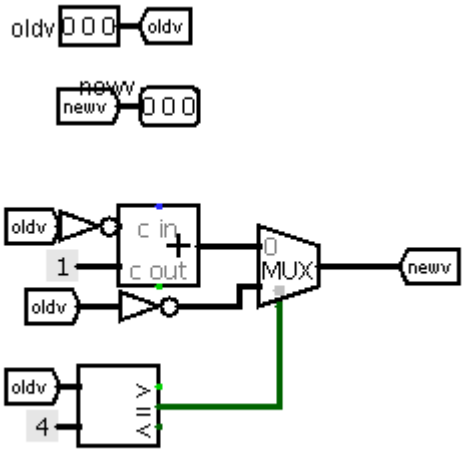


There we update velocity, X and Y coordinates when we want to reset the game. There we have another **extension** connected to restart. You can read about it here.

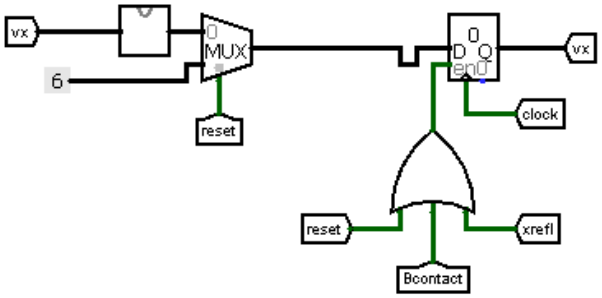


This scheme is used to update the **Y** coordinate of the right bat.

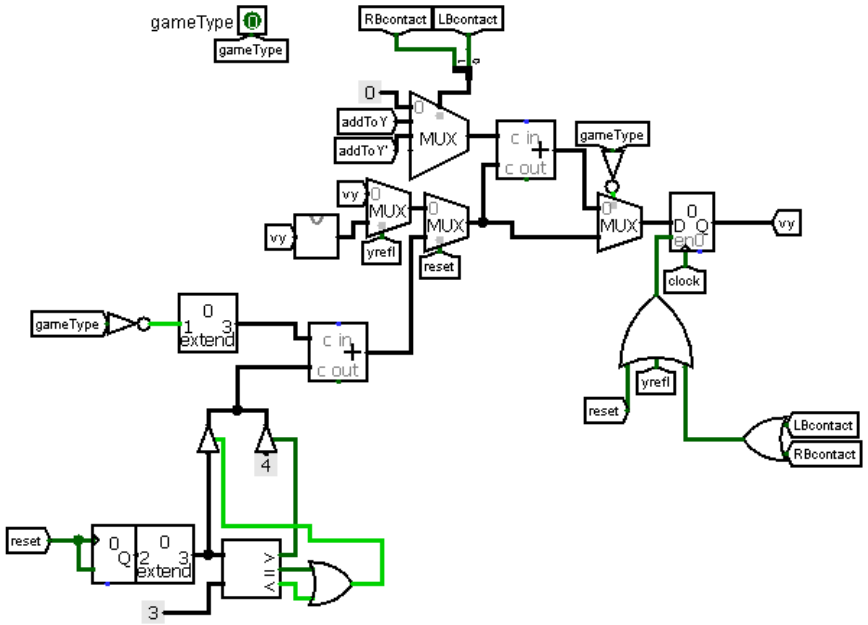
These two schemes are for **updating VX and VY**. There we used another chip: **reflect**.



There we **update VX**:

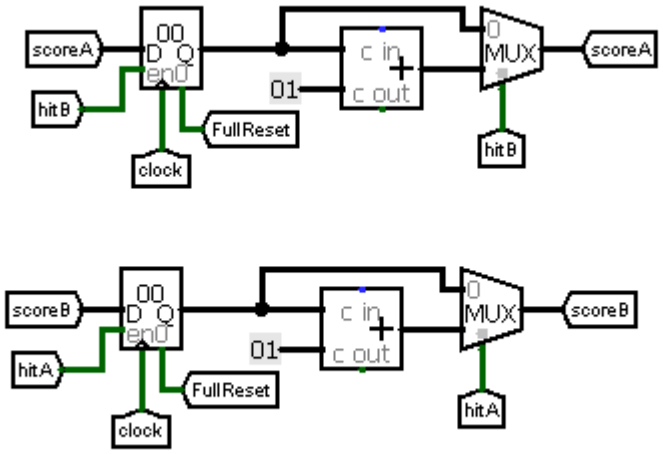


And there **VY**:



This scheme looks quite scary, as there we made an **extension**. You can read about it here. Also, there we realized half of another extension connected to game modes.

These two schemes are for **counting scores**:

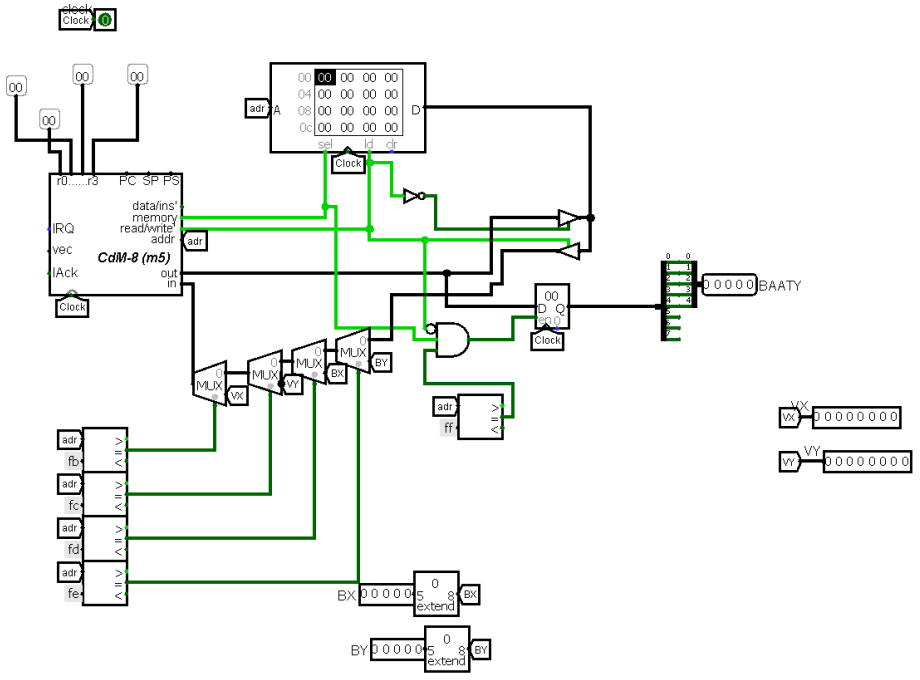


So, if the ball hits the right edge, we increment the score for the player, in the other way, we increment the score for the bot.

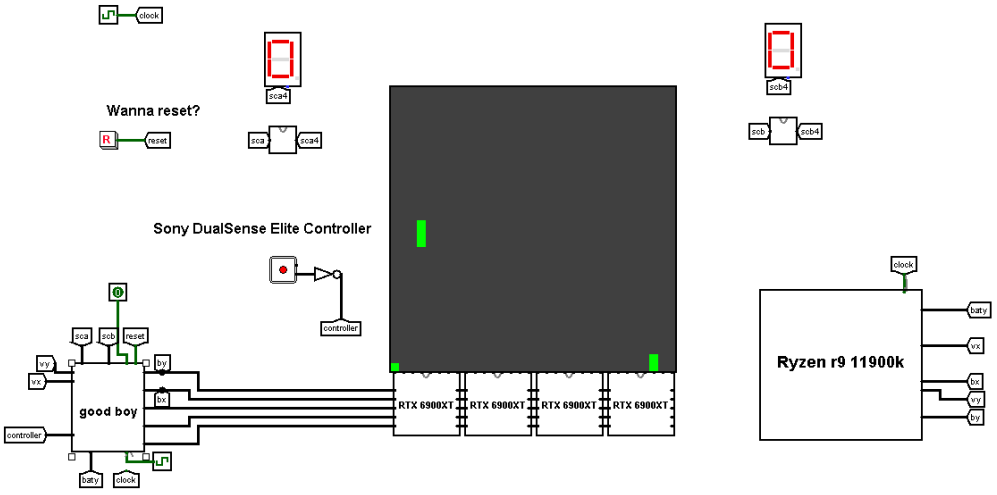
Finally, the kinematic controller was done, so we could place a flag here.

Chapter 3. Connecting all schemes

So, all hardware components of the game were ready, we had to connect them to each other. To begin with, we connected the kinematic controller to the video sector. Kinematic sends X and Y coordinates of the ball and Y coordinates of the bats. Also, we had to connect the CDM-8 which controls the right bat. That's why we slowed down the kinematic controller twice, so this trick helps the processor to calculate the new position for the bat, as he will have more time before the ball will reach the X coordinate of the bat. After calculating the Y coordinate, we send it to the kinematic controller.



The final result:



Good boy is a kinematic controller (this name makes it work faster and better). **Ryzen r9 11900K** is a CDM-8 processor**. RTX 6900XT** is a video sector. **Sony DualSense Elite Controller** is a controller, which we use to move the left bat. **Wanna reset?** is a reset button. Finally, two Hex Digit displays are for showing scores.

Chapter 4. Bot

```
asect 0x00
start:
ldi r3,by
ld r3,r3
ldi r1,bat
st r1,r3

toPlayer: # when the ball goes to player
ldi r2,vy #bot chases it with V=+-1
ld r2,r2
tst r2
bmi minus
inc r3 #V=1
br continue
minus:
dec r3 #V=-1
br continue

continue:
ldi r0,vx
ld r0,r0
tst r0
st r1,r3
bmi toPlayer

ldi r0,vy
ld r0,r0

go: #ball  moves to bot
   # bot moves with V=VY_Ball


st r1,r3 # we always store Bat_y so his movement
        # will be smoother
add r0,r3

ldi r1,bx
ld r1,r1

ldi r1,bat

ldi r2,32 #Checking overflow of bot_y
cmp r1,r3
bmi go1

not r3
sub r2,r3
```

```
go1:
ldi r2, 25
cmp r2,r1

ble go

ready:
st r1,r3

br start


asect 0xfb
vx:

asect 0xfc
vy:

asect 0xfd
bx:


asect 0xfe
by:

asect 0xff
bat:ds 1
end
```

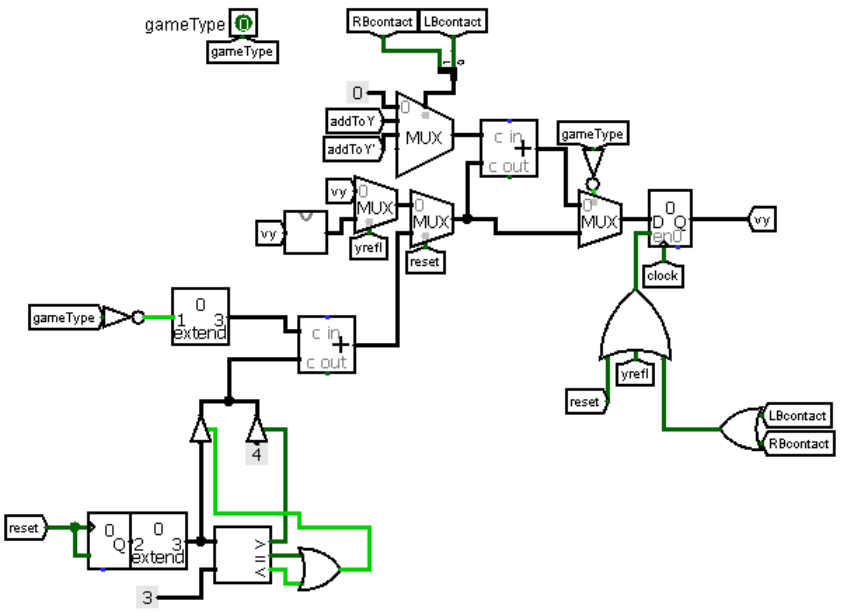
Above is the second version of the bot code. Initially, the racket, which was controlled by the bot, teleported to be perpendicular to the ball at the moment when it began to fly in the bot's direction, and only then, began to move with the vertical speed of the ball. It was impossible to beat such a bot. The second version works differently. The racket follows the ball at a vertical speed of 1 when the ball is flying away from it, and when flying in the opposite direction, the racket begins to move at the speed of the ball. We faced such a problem as an overflow, when the coordinates of the bot's racket took on a value greater than the allowable value. The racket started involuntary movements and teleported. We have solved this problem.

Chapter 5. Extensions

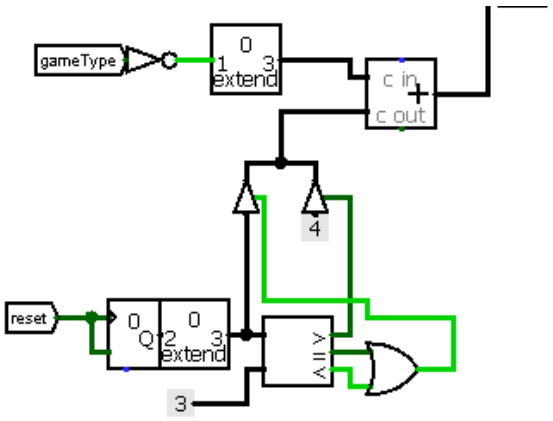
Original version of that game was boring, every game was similar to the previous one, so we made some extensions which affect VY and it will make the game more interesting.

1) We added two game modes with the default reflecting and crazy reflecting. Let's define what it means. Crazy game mode means that if the ball hits the top pixel, we make VY++, if the bottom pixel - VY--, so the player can affect the game more. In the reflect chip we check the place where the ball hit and after that rewrite VY by adding or subtracting 1 to the old VY value. Also, we added a multiplexer to use the reflect chip only when the ball hit the floor.

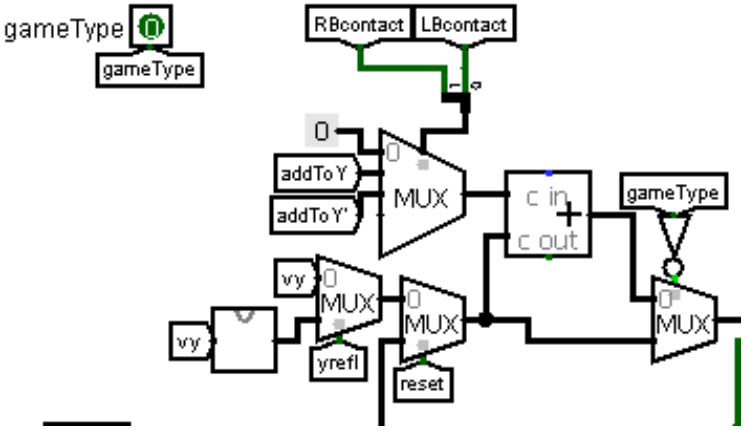
2) Random VY, when we reset the position of the ball.



There we have a 1-bit pin for changing the game mode: 0 is for normal mode, 1 is for crazy.

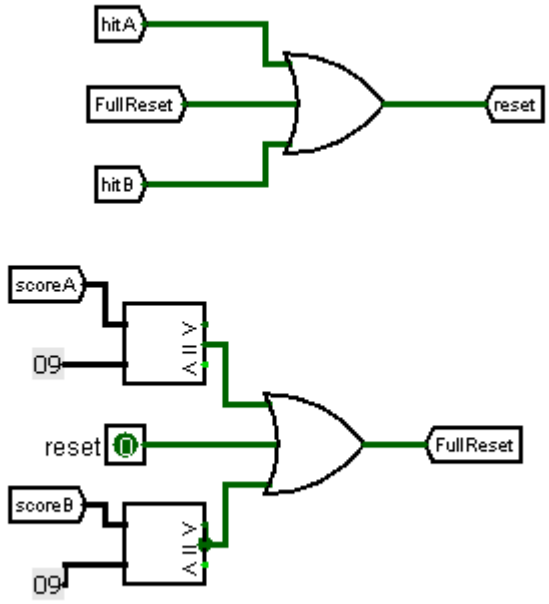


This part is for making random VY after placing the ball to the start position.



This part is for the game mode, where we change VY, when the ball hits the bat.

3) We reset the position of the ball when someone scores a goal. We have two resets: one just reset position of the ball, another also reset scores. The game stops, when the player or the bot get nine points.



The first scheme we use to detect when we need to reset the game. Second one is for stopping the game when we want to reset the game or when someone gets 9 points.

Chapter 6. Further development plan

1. In the future, we plan to make the reflection of angles more correct by calculating them through cosines and sines.
2. Make several levels of difficulty for the bot to make the game more interesting, make the movement of the bot smoother.
3. Add animations and sounds to give your game a brand identity