МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ"

(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

15.03.06 – Мехатроника и робототехника
Специализация (профиль): Искусственный интеллект

КУРСОВАЯ РАБОТА

Тема задания: **‘PROJECT B’**

Неверов Глеб (24940)
Основская Юлия (24940)
Лапин Егор (24940)

Новосибирск
2025

Введение

Наша команда выбрала как тему проекта «Project B», которая подразумевает реализацию игры Pong.

Pong - одна из ранних аркадных видеоигр, представляющая из себя спортивный симулятор настольного тенниса. Впервые её разработала фирма Atari, а точнее Аллан Алькорн, в качестве первой практики работы над видеоиграми. Первый автомат с игрой Pong был поставлен в 1972 году, и за несколько дней набрал большую популярность. Уже в том же году Atari начала продавать свои автоматы. Эта игра называется первой коммерчески успешной видеоигрой.

Игра Pong по-настоящему легендарная, поэтому мы решили реализовать её на платформе Logisim с использованием CocoIDE.

Правила игры в нашем варианте:

На экране слева и справа расположены две ракетки. Управляя ракетками нужно отбивать мяч, летающий по полю. Когда мяч попадает в стену, “охраняемую” одной из ракеток, это значит, что был забит “гол!”. Игра продолжается до 10 очков. Но если отставание между игроками меньше 2 баллов, то игра не остановится, и даст игрокам возможность определить победителя на 12 очках, и так далее.

В нашей реализации есть два режима игры:

- 1 игрок. Здесь можно посоревноваться с ботом.
- 2 игрока. Здесь можно посоревноваться с другом.

Pong был предоставлен в качестве базового проекта, но мы внесли в него **дополнения**!

- Меню игры,
- 2 режима игры, описанные выше,
- Улучшенные правила победы, также описанные выше,
- Пауза и выход в меню,
- Управление с клавиатуры,
- Бортики по краям поля,
- Поле 64x32, что больше, чем предложено в базовом проекте.

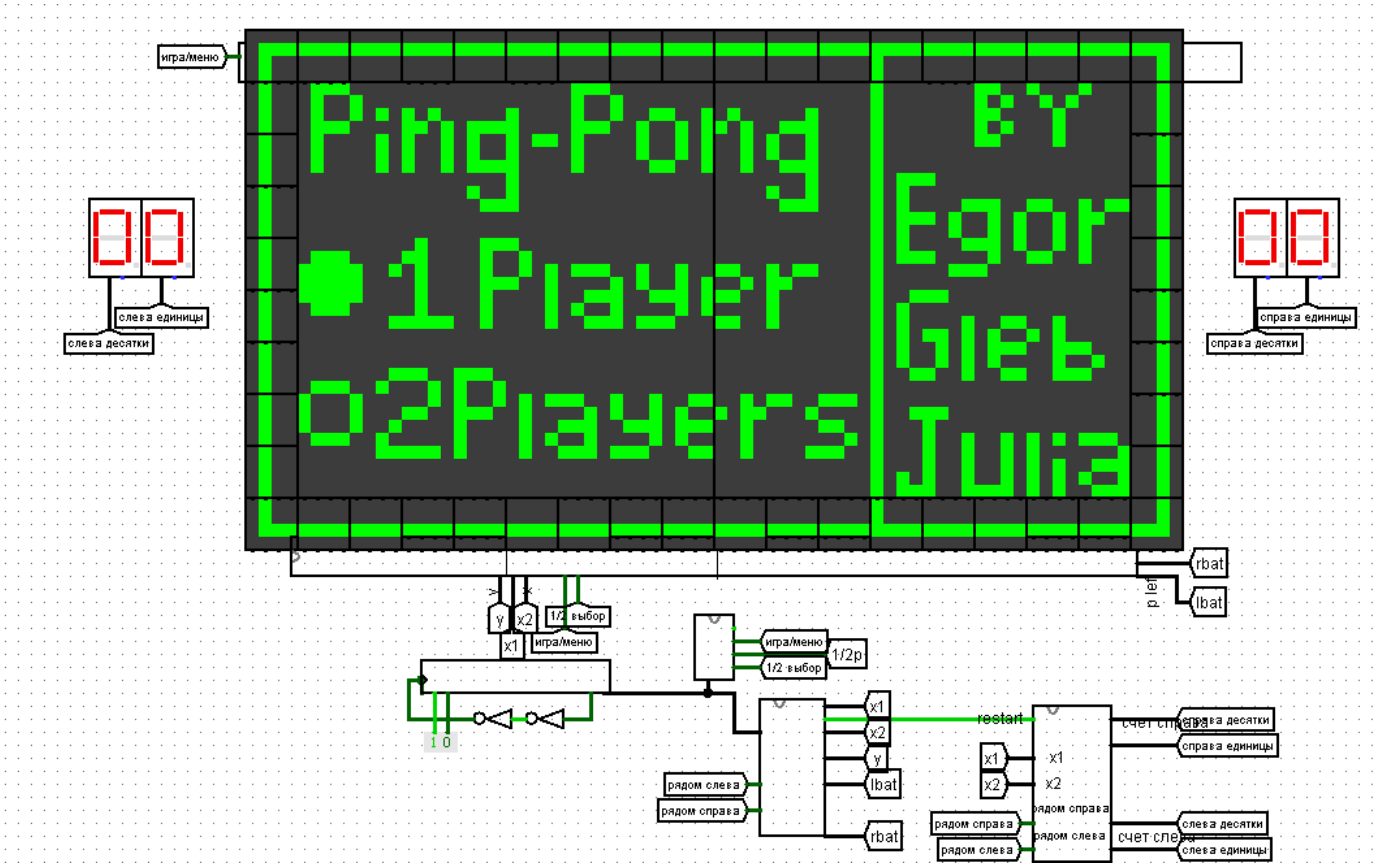
Назначение и область применения

Для реализации игры Pong нам понадобилось разработать:

- Экран, который отображает ракетки и мячик.
- Ракетки, от которых отбивается мяч.
- Схема для определения попадания.
- Счётчик очков у каждого игрока.
- Робот, который играет с человеком.
- Кинематику шарика, который скачет по экрану 32x32.
- Меню с выбором количества игроков и паузой.
- Клавиатура для управления меню и ракетками.

Технические характеристики

Схема (main):



Заглавная схема выполняет роль соединения всех рабочих модулей, а также экрана 64x32 с клавиатурой управления. Для запуска игры нужны выбрать режим игры (2 игрока или 1 игрок), включить моделирование и включить такты(скорость тактов влияет на скорость движения шарика).

Так выглядит сама игра:

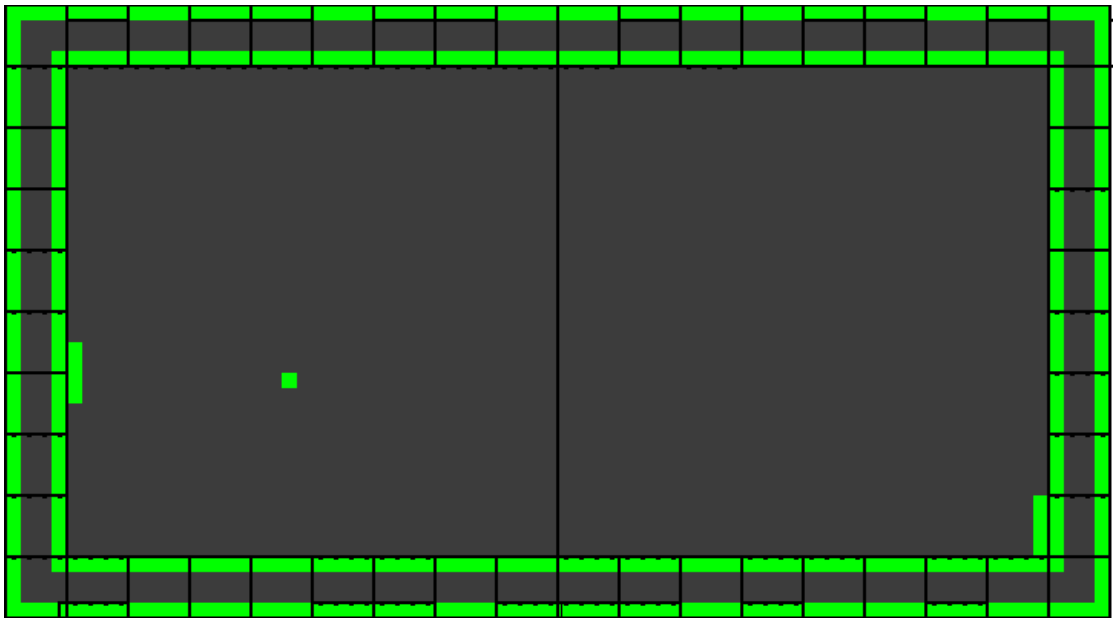
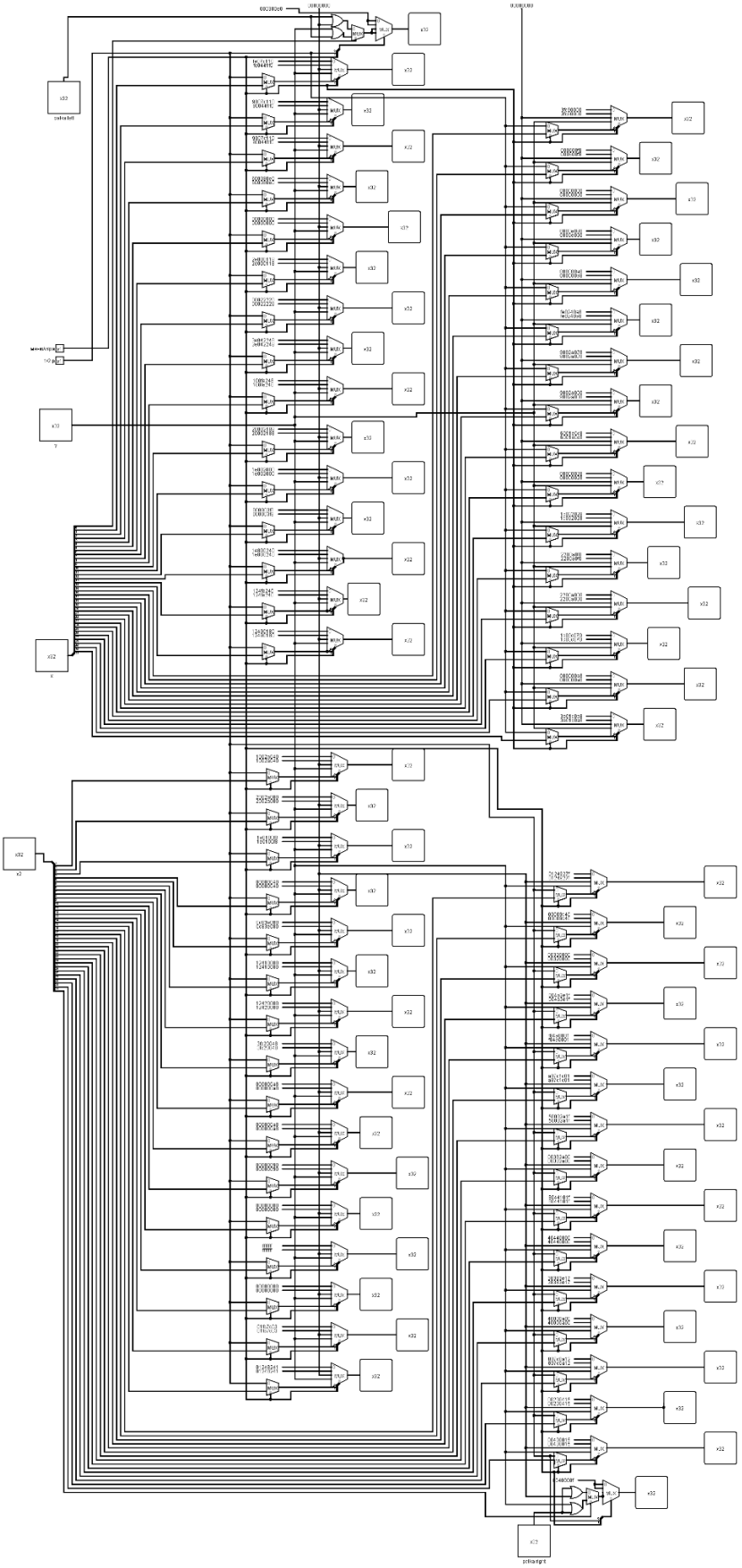


Схема (videocarta):



Из названия схемы следует, что она отвечает за отображение всех нужных элементов для игры. Также в ней реализована логика определения столкновения мяча с ракетками игроков.

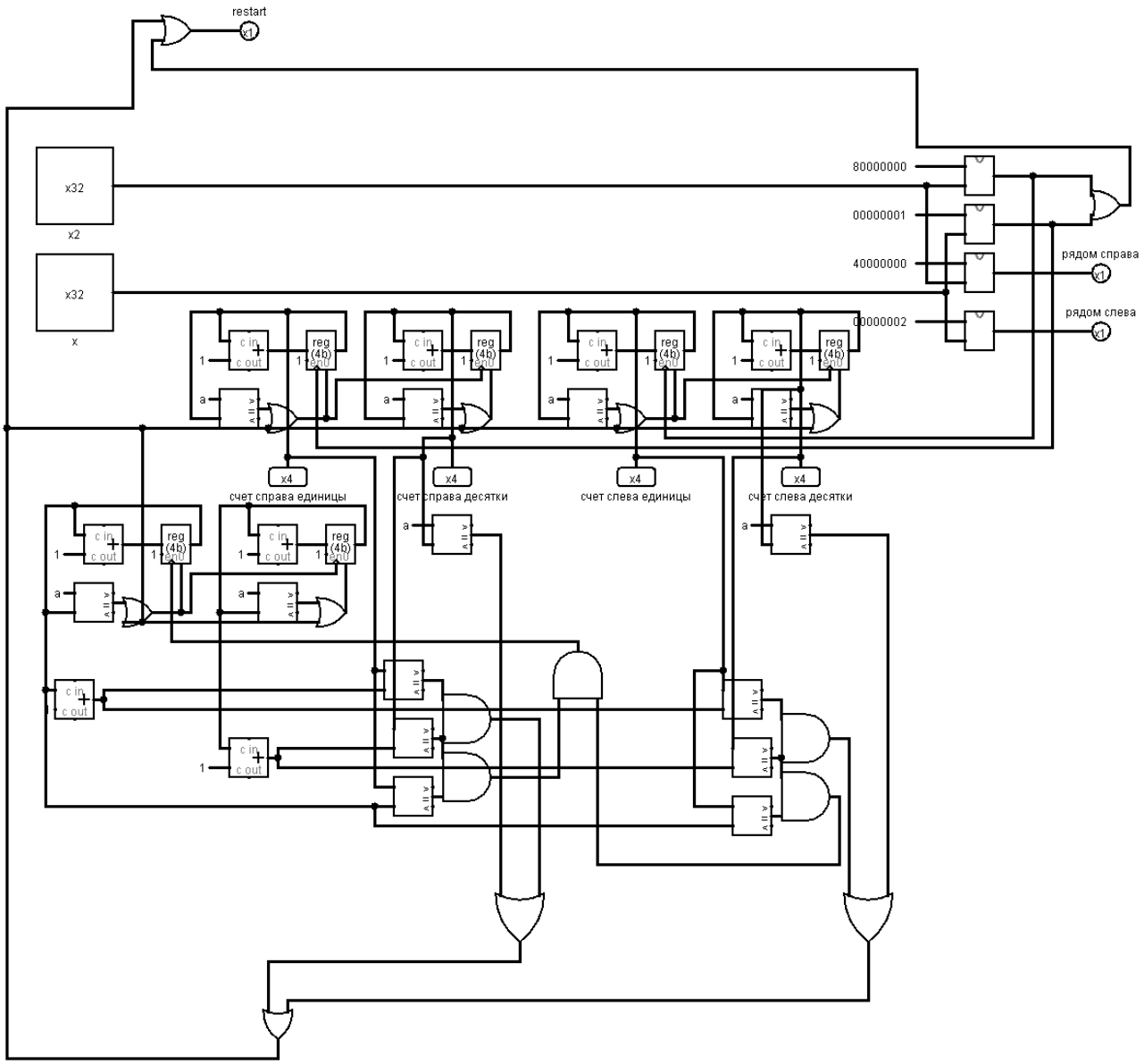
Входные сигналы:

- X, X2, Y — координаты мяча,
- palka left и palka right — положение ракеток игроков.

С помощью мультиплексоров и логических элементов (ИЛИ) схема сравнивает координаты мяча с положением ракеток.

Также в мультиплексоры входят константы, которые выводят картинку для меню.

Схема (score):

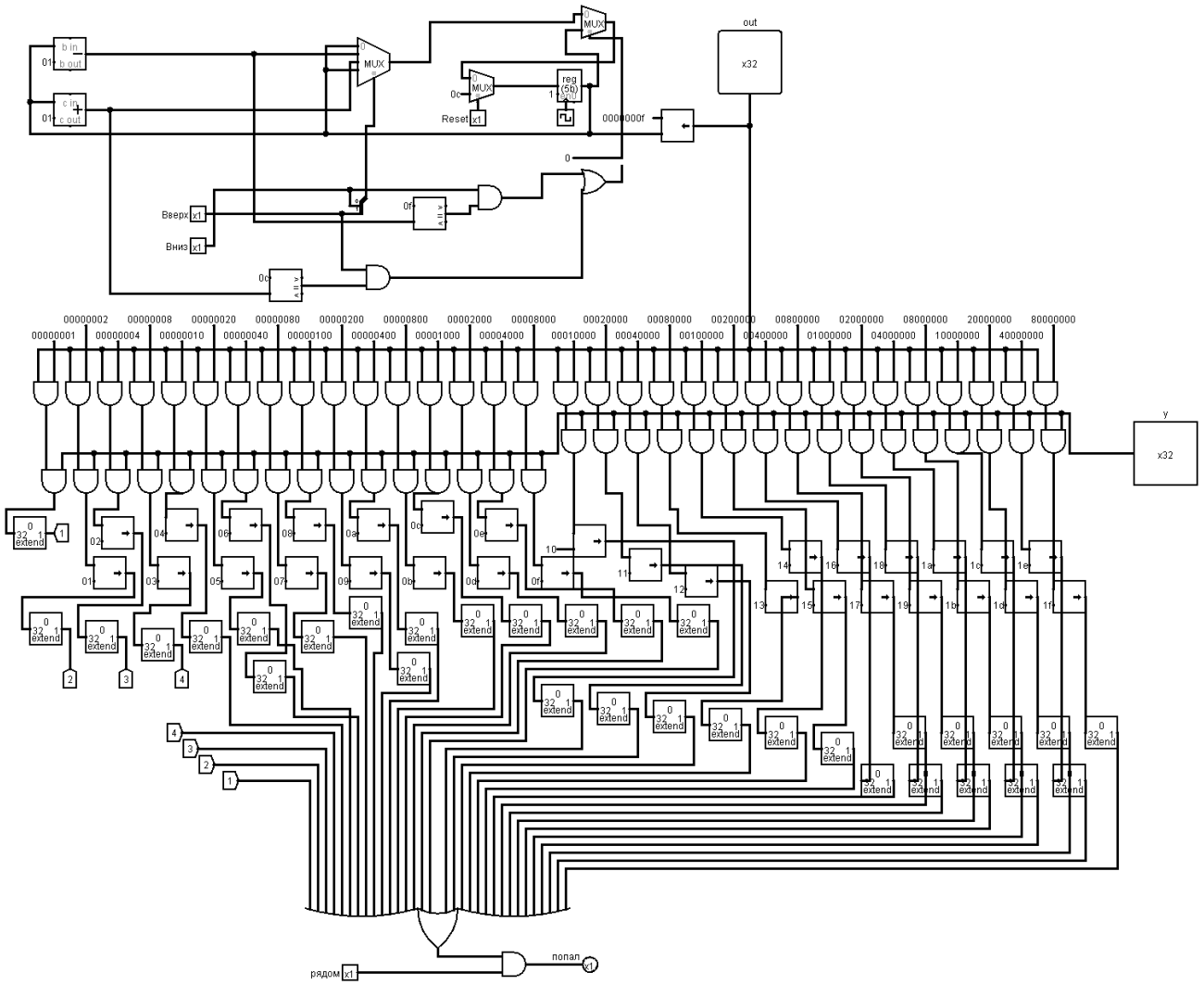


На этой схеме реализован подсчет очков игроков.
Входные сигналы:

- start — начало игры,
- restart — сброс игры,
- x, x2— координата мяча.

Блок проверяет, достиг ли мяч правого или левого края поля. Если да, то происходит увеличение счёта игрока, который забил гол. Через компараторы координаты мяча сравниваются с контактами, благодаря инкременту и регистру увеличивается и сохраняется значение счётчика. Игра идёт до 10 очков, если оба игрока набрали 10, то игра идёт до 12, и так далее.

Схема (detect hit & rocketka):

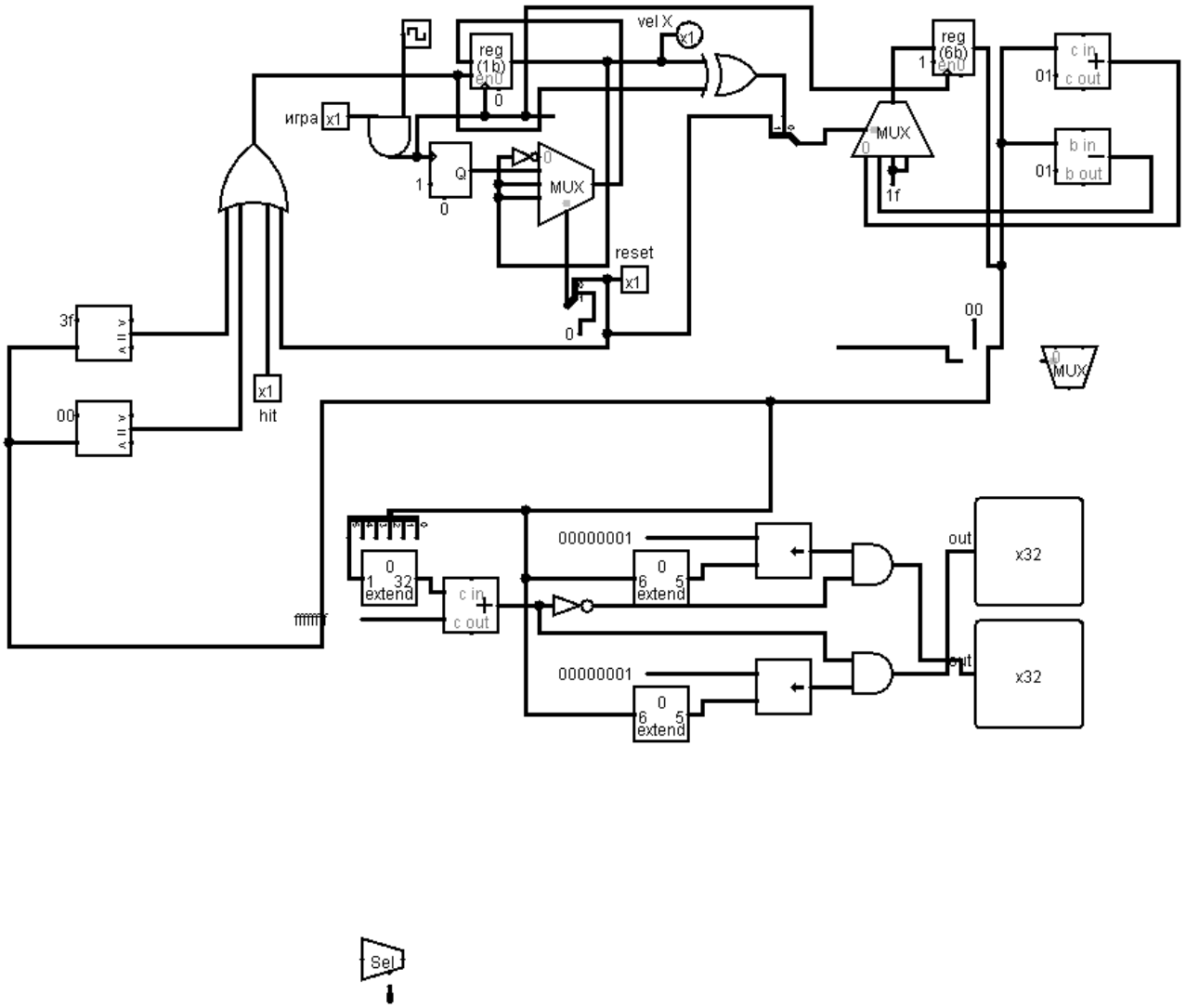


Логические элементы обрабатывают сигналы направления и сброса.

Схема сбрасывает положение ракетки при сигнале reset, возвращая ее в начальную позицию.

Нижняя часть схемы отвечает за проверку положения мяча на экране и пониманием: «произошло попадание или нет?» через сравнивание координат мяча с константами.

Схема (X):

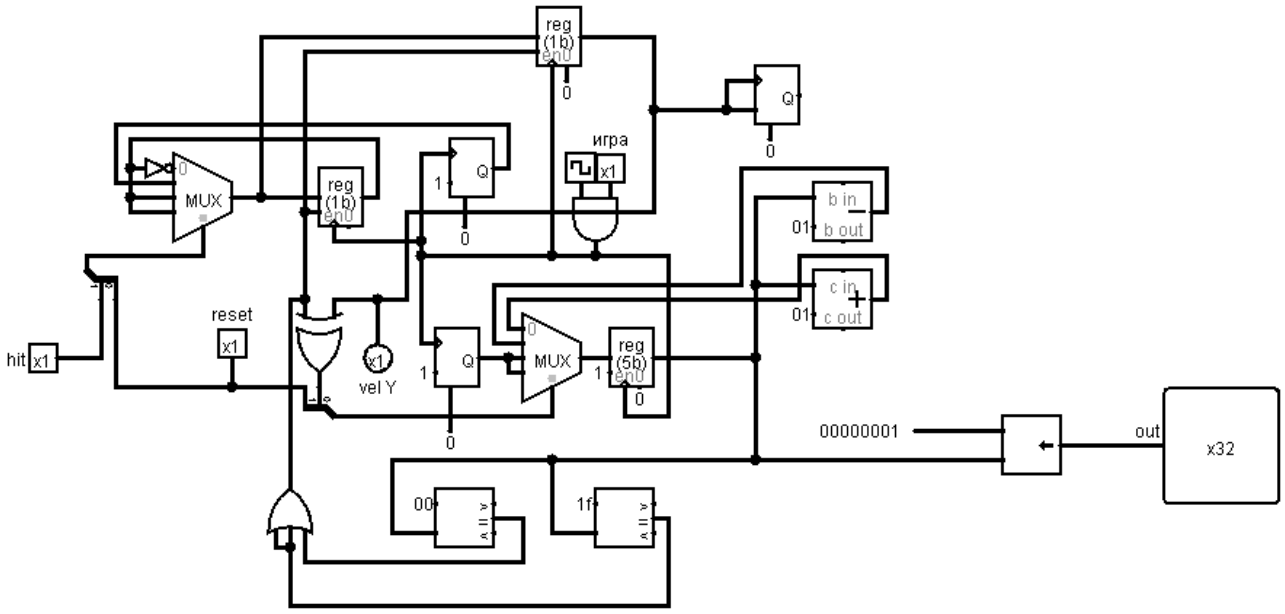


Эта схема описывает работу шарика по оси X.

Входы:

- hit: сигнал, показывающий, что шарик столкнулся с ракеткой или стеной.
- reset: сигнал для сброса схемы или установки шарика в начальную позицию.
- Vel X: сигнал, определяющий направление движения. На каждом тактовом цикле сумматор прибавляет значение vel X к текущей координате X. Если vel X = +1, то X увеличивается (движение вправо).Если vel X = 0, то X уменьшается (движение влево).

Схема (Y):



Эта схема описывает движения шарика по оси Y.

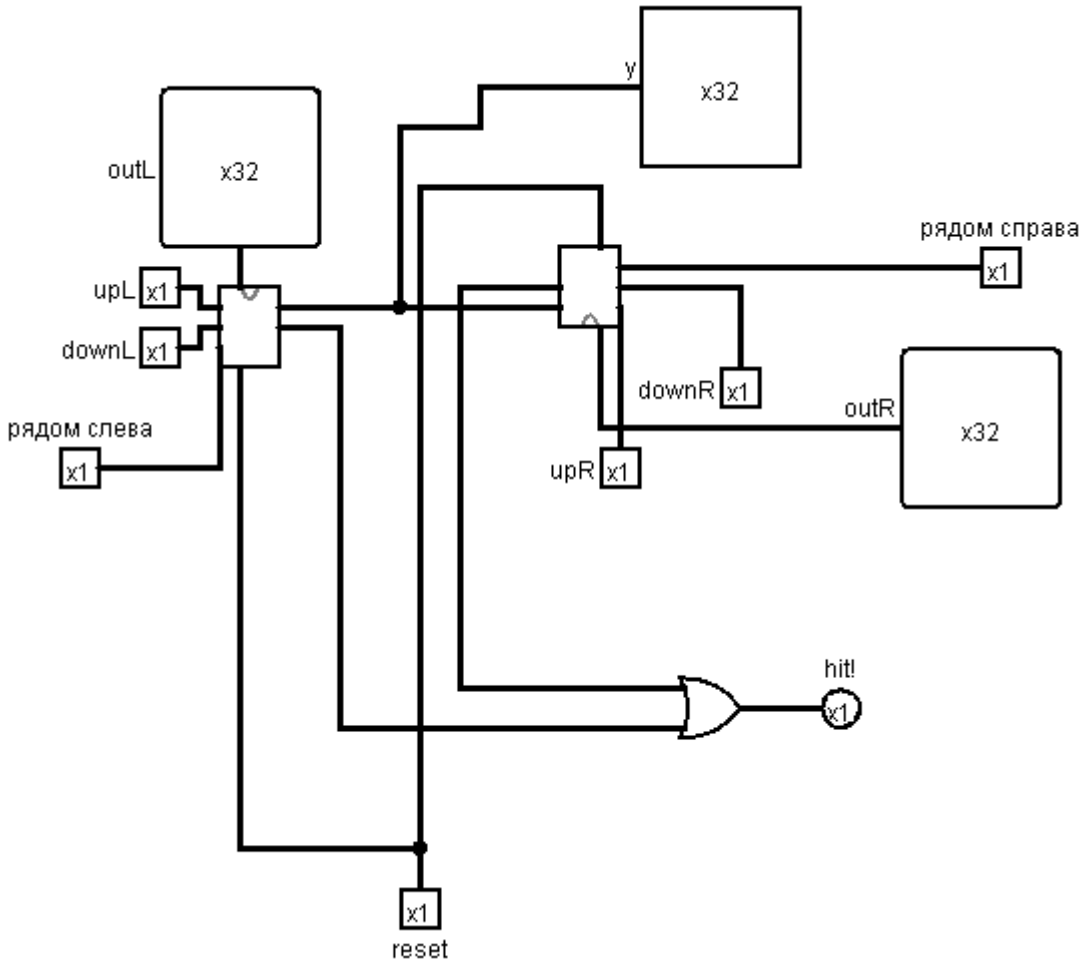
Входы:

- hit: сигнал, показывающий, что шарик столкнулся с ракеткой или стеной.
- reset: сигнал для сброса схемы или установки шарика в начальную позицию.
- Vel Y: сигнал, определяющий направление движения.

Сигнал vel Y через мультиплексор задает направление: вверх (уменьшение позиции) или вниз (увеличение позиции). Сумматор изменяет текущую позицию шарика на основе направления.

Регистр сохраняет новую позицию, которая используется для отображения шарика на экране.

Схема (temp):



Вспомогательная схема, которая служит для входа и выхода сигналов и значений в схему (detect hit & rocketka).

Схема (bot):

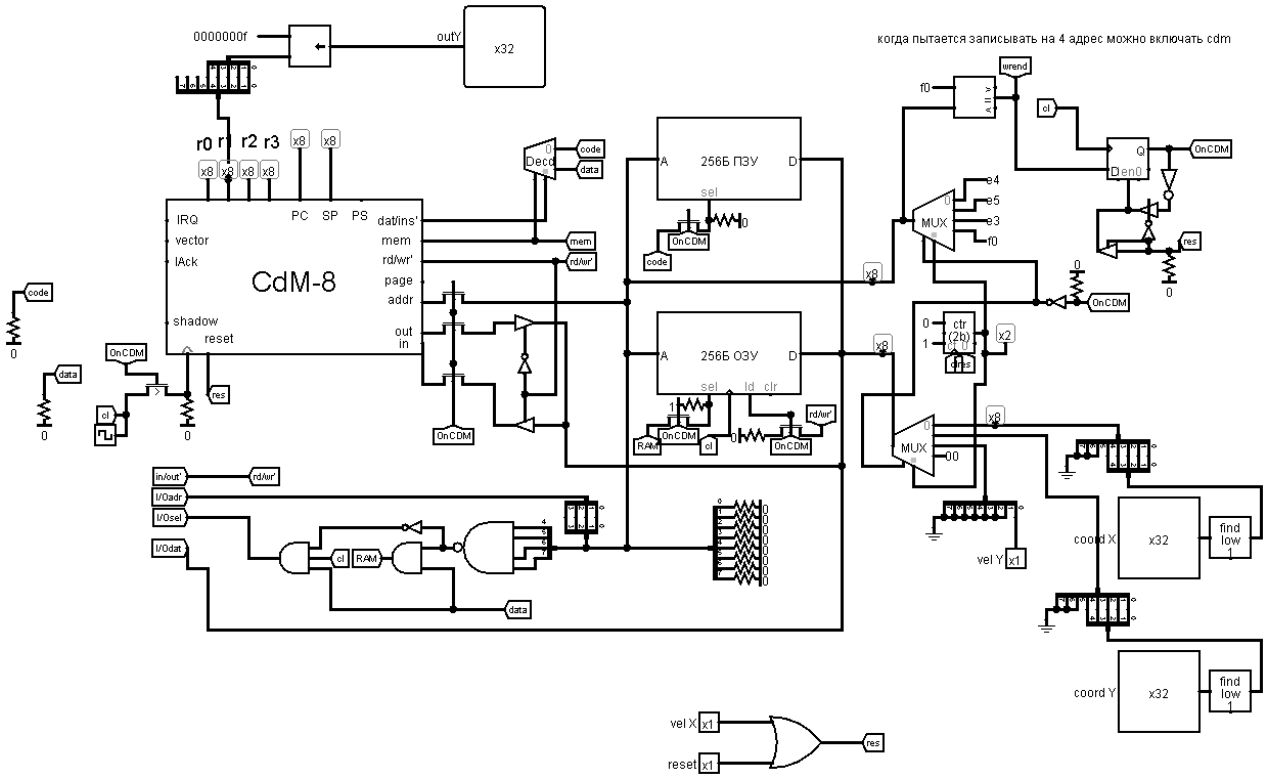
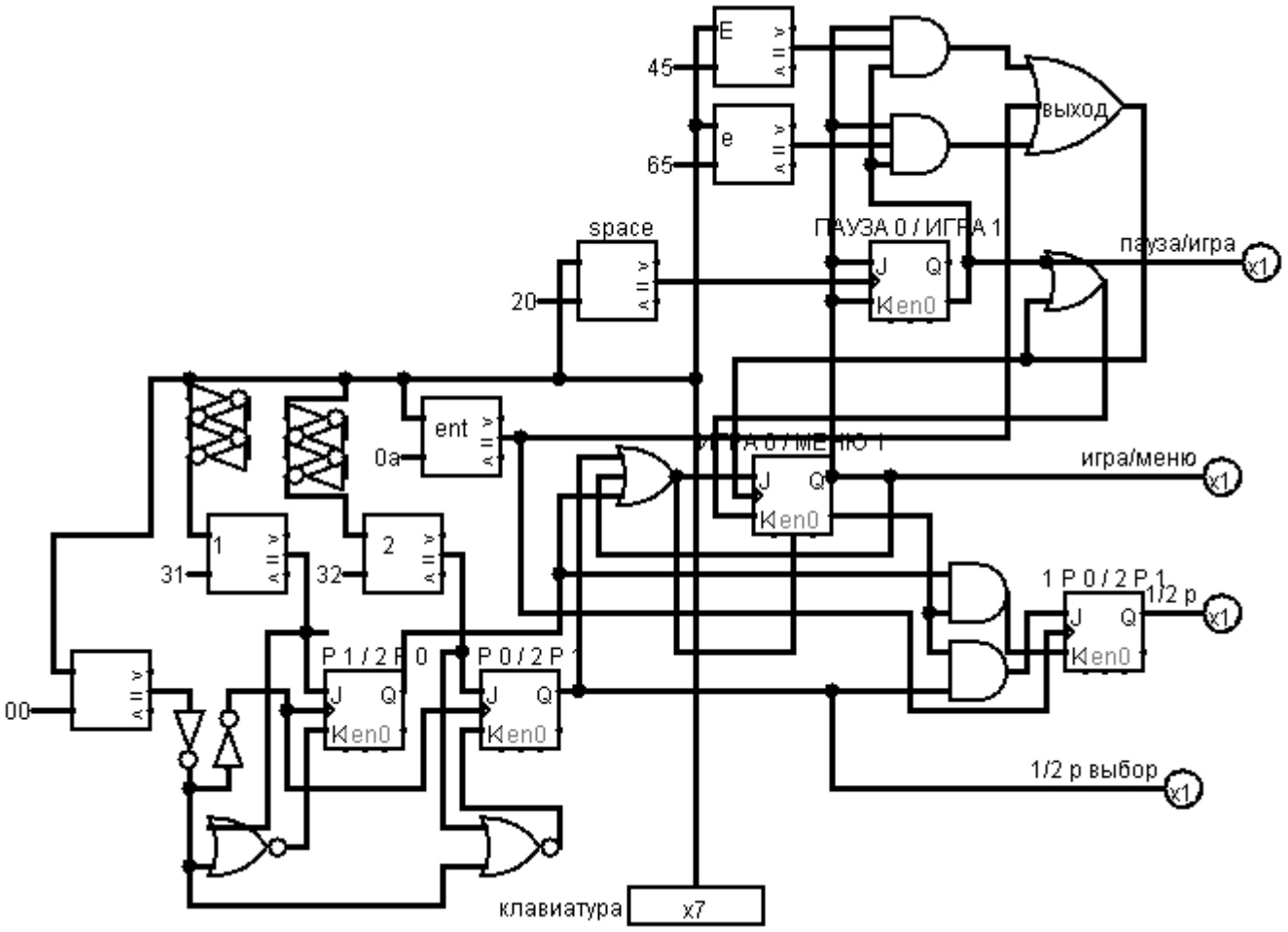


Схема для подключения процессора к аппаратной части игры, мы реализовали стандартную схему подключения процессора. На шину данных мы передаем координаты шарика по осям X и Y. Это нужно для программной части.

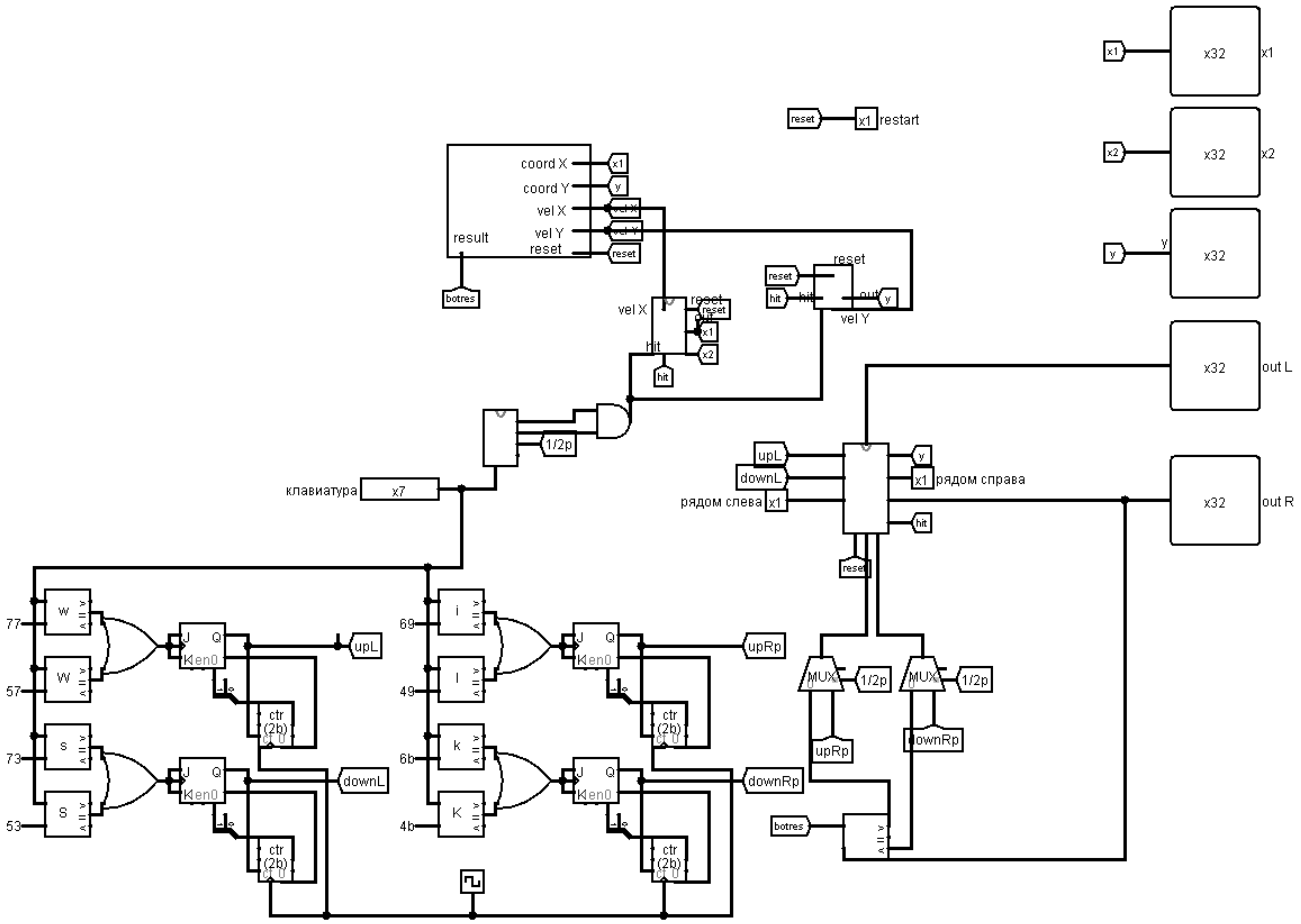
Процессор работает не всегда. Система ожидает, когда либо мяч будет отбит ракеткой игрока, либо пройдет команда reset. После чего начинается запись в память, программа вычисляет координату по Y, на которую нужно сдвинуть ракетку бота. Это значение остается на регистре r1, откуда оно выводится в остальную схему игры.

схема (menu):



Игрок набирает на клавиатуре “1” или “2”, соответствующий триггер запоминает выбранный режим. Нажатие “Enter” подтверждает выбор, сигнал “игрок/меню” гаснет, и запускается игра.
“Space” в любой момент ставит / снимает паузу, выдавая чистый сигнал “пауза/игра”.
“E”/”e” мгновенно формирует выход, по которому верхний уровень возвращает игрока в начальный экран.

схема (bigc):



Верхняя половина схемы bigc использует подсхемы для более удобного взаимодействия с ними в проекте, освобождая пространство схемы main. Тоннели способствуют читаемости схемы. Нижняя половина bigc отвечает за пользовательский ввод с клавиатуры для управления ракетками во время игры.

схема (borders):

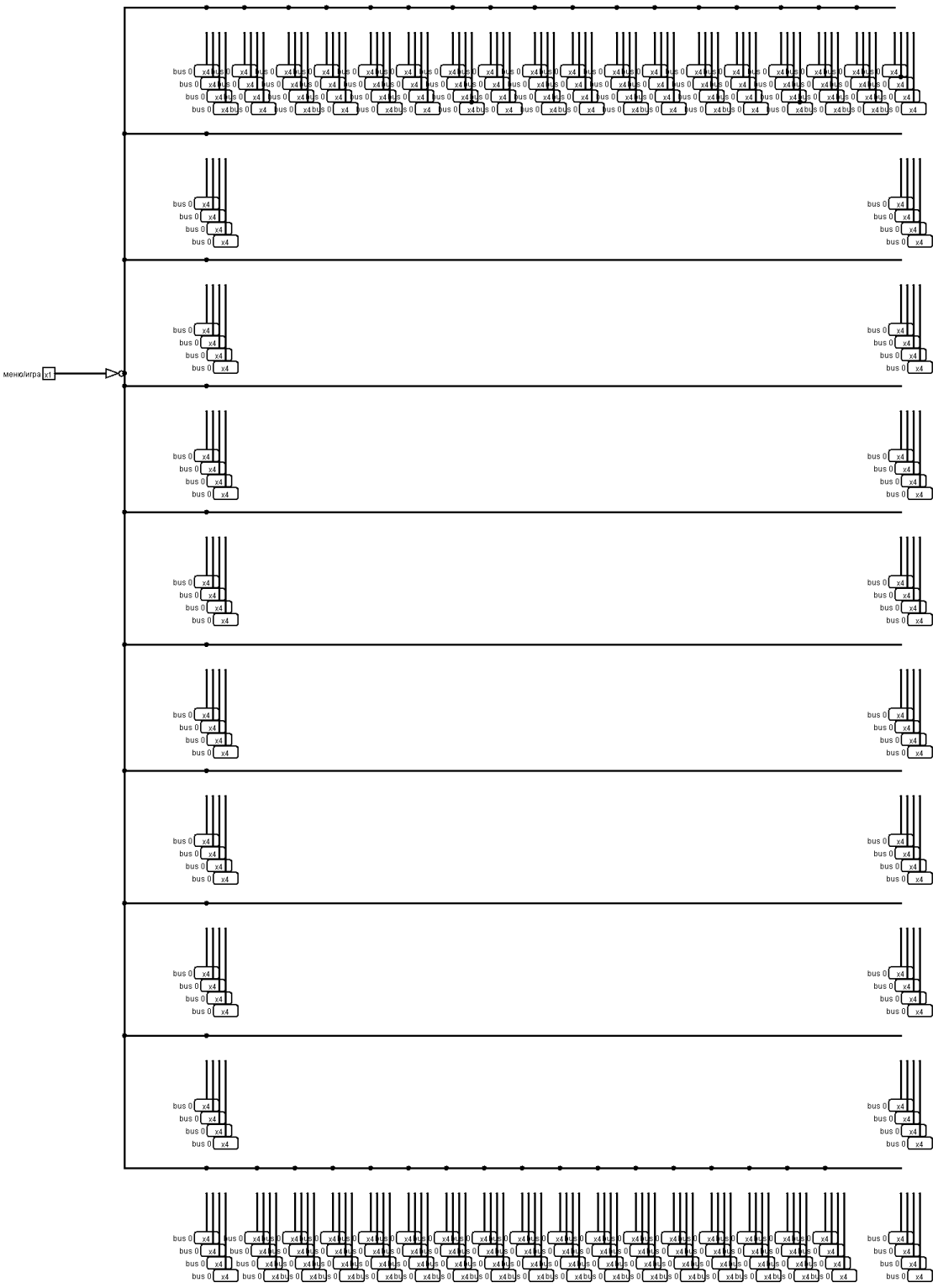
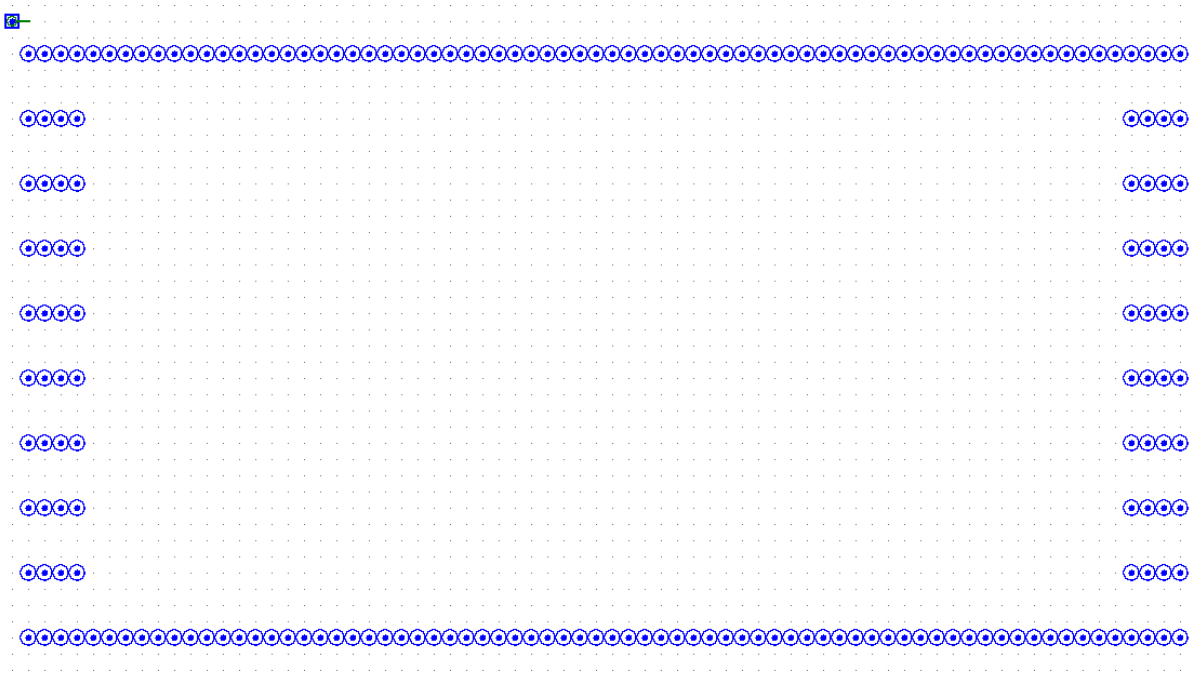
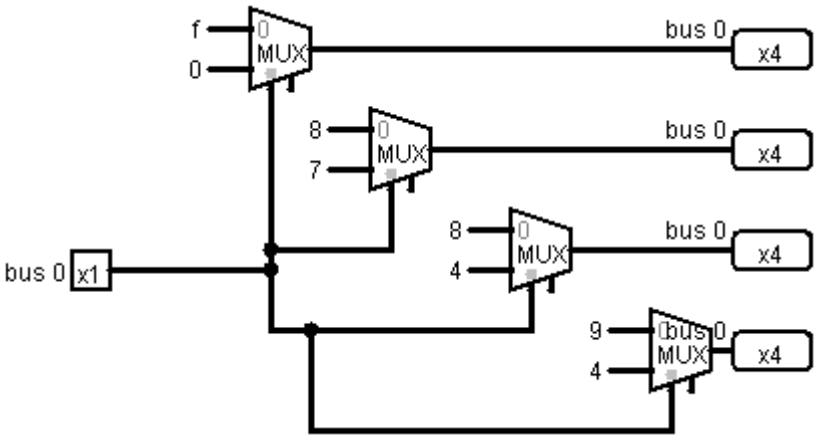


Схема для вывода пикселей бортиков, использующая вспомогательные схемы: b11, b12, b1_18, b1_13, b2_2, b2_18, b10_1, b10_2, b10_13, b10_18. Каждая из вспомогательных схем служит для вывода значений определённой матрицы 4x4.

Внешний вид схемы позволяет соединять матрицы схемы main с схемой borders без проводов:



вспомогательные схемы (b):



Функциональные характеристики

В программной части мы реализовали бота, который играет с пользователем. Пока шарик находится на половине человека, после отбития ракеткой, бот начинает следить за мячом и старается предугадать, куда боту нужно поставить ракетку, чтобы отбить мяч.

```
##########################################
# where is data?
# 0xE1 - coordinates of cdm8's bat on y
# 0xE2, 0xE3 - ball's velocity by x and y
# 0xE4, 0xE5 - ball's coordinates by x and y
##########################################
asect 0xe1
bat_coords_y:           ds 1
ball_velocity_x:        dc 1
ball_velocity_y:        ds 1
ball_coord_x:           ds 1
ball_coord_y:           ds 1
##########################################
```

Эта часть кода четко определяет ячейки памяти, из которых код будет считывать информацию, передаваемую ему из аппаратной части.

```
asect 0x00
        ldi r0, 0xe3
        ld r0, r3 #vY
        inc r0
        ld r0, r1 #X
        inc r0
        ld r0, r2 #Y

        ldi r0, 62
        neg r1
        add r0, r1 # 62 - X

        if
                tst r3
        is nz
                neg r1 # (62 - X) * vY
        fi

        add r2, r1 # Y + (62 - X) * vY
        if
                tst r1
        is mi
                neg r1
        fi
        ldi r3, 32
        if
                cmp r1, r3
        is gt
                sub r1, r3
                move r3, r1
        fi
halt
end
```

Далее идут вычисления. Основная используемая формула d = Y + (62 - X) * vy, где (X, Y) - начальная координата мяча, vy - начальная скорость мяча по Y.

После основных вычислений мы выполняем корректировку. Основная ее задача - сделать так, чтобы вычисленная координата ракетки не выходила за границы поля. Так как мы решили сделать поле прямоугольным, компонента (62 - X) может оказаться сильно больше остальных значений, поэтому необходимо скорректировать получившийся результат.

Заключение

Мы успешно реализовали игру Pong на платформе Logisim с использованием процессора CdM8.

В этом проекте мы столкнулись с некоторыми проблемами с коммуникацией и налаживанием работы. Благодаря совместной работе над этим проектом мы получили важный опыт взаимодействия в команде.

Также мы изучили низкоуровневую работу с программой на CocoIDE и связывание программного обеспечения с процессором и смогли попробовать себя в работе над более масштабными схемами, чем те, что выполняли в рамках паков. Работа над проектом позволила нам на практике применить материалы курса Цифровые Платформы, разработать собственные решения и проявить креативность.

Завершив работу над проектом, мы стали увереннее в своих навыках построения схем и программирования CdM8.

Источники

Tome.pdf