МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЯ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

15.03.16 - Мехатроника и Робототехника

Специализация (профиль): Интеллектуальная робототехника

**Пояснительная записка**

Тема задания: **“Игровая консоль”**

Путилов Дмитрий, 24940

Саньжаева Полина, 24940

Крюк Михаил, 24940

Новосибирск

2025

**Содержание**

Термины и аббревиатура…………………………………………………………3

Введение…………………………………………………………………………...4

Цель и область применения ……………………………………………………...5

Функциональные характеристики……………………………………………….5

Технические характеристики……………..………………………………….......5

Заключение………………………………………………...……………………..13

Использованая литература…...................................…………………………….14

**Термины и аббревиатуры**

Logisim — программное обеспечение для моделирования цифровых схем.

CdM8 Mark5 — восьмиразрядный процессор, применяемый для обучения и проектирования цифровых систем.

Assembler — язык низкого уровня, близкий к машинному коду, используемый для программирования микропроцессоров.

**Введение**

Игровые консоли на сегодняшний день – распространнёное явление. Но раньше они были уникальностью, да и игры на них были значительно проще. Как визуально, так и технически.

Наша команда поставила перед собой задачу создать прототип игровой консоли, дополнительно создав несколько простейших игр для неё. За основу дизайна для пользовательского интерфейса бяла взять известная в пост-советских странах консоль “Brick Game”. Используя такой образец, мы не могли не реализовать игру “Tetris”. В качестве ещё одной тестовой игры мы решили выбрать “Flappy bird”. Оба этих проекта отличаются простым, но интересным игровым процессом.

Наш проект был реализован в среде Logisim с использованием виртуального процессора CdM8 Mark5. Код написан на языке Assembler в среде CocoIDE. Проект является частью учебной программы по цифровым платформам и направлен на изучение и разработку игровой логики на низком уровне с учетом ограничений встроенных систем и на практическое применение цифровой логики при создании програмной архитектуры.

**Цель и область применения**

Реализовать программу на низкоуровневых платформах, CocoIDE и Logisim. Данный проект предназначен для применения в игровой сфере.

**Функциональные характеристики**

Наша команда не смогла реализовать желаемые функции с помощью СdM8 и Assembler, поэтому в финальной версии проекта меню работает с помощью технической части. Другими словами, функциональная часть отсутсвует.

**Технические характеристики**

**Первый раздел: Симуляция консоли**

На рис.0 демонстрируется пользовательский интерфейс нашего проекта. Вывод для пользователя на матрице размерами 48\*32 пикселя. Ввод пользователя с 6 клавиш. “M” – переход в меню. “S” – запуск/перезапуск выбранной игры. “<” и “>” – клавиши движения влево и вправо. Отвечают за навигацию в меню, движение блока по горизонтали в игре “Tetris”. “L” и “R” – вспомогательные клавиши управления. Отвечают за поворот блоков в игре “Tetris”. Любая из 4 клавиш управления игрой так же отвечает за управление персонажем в игре “Flappy bird”.

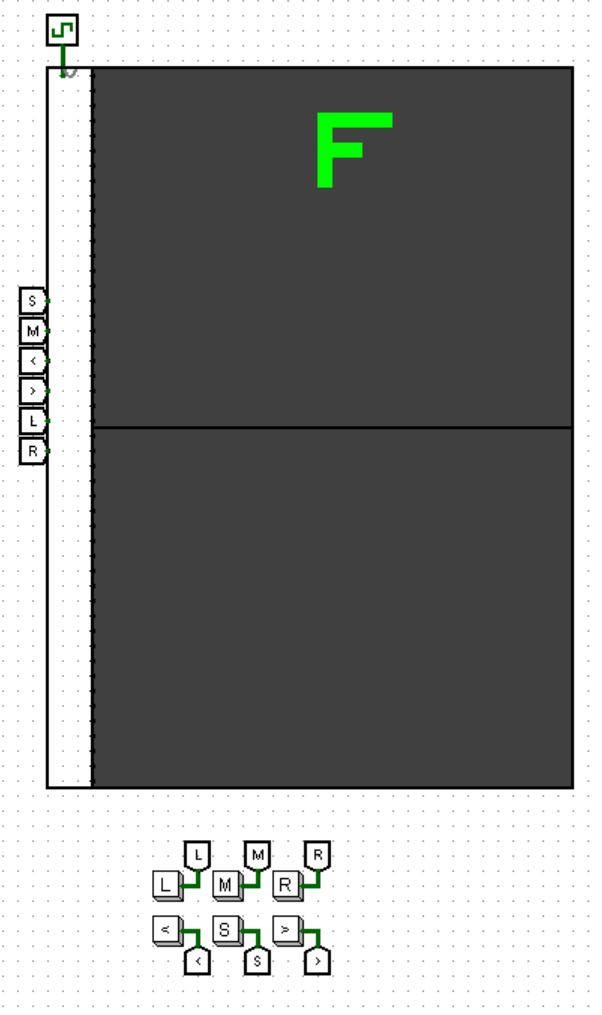


Рисунок 0

В схеме, представленной на рис.1 реализуется меню консоли. Левый масив пинов – вводные данные (клавиши управления). Центральная схема – смена состояний (0 – страничка “Tetris”, 1 – страничка “Flappy bird”, 2 – игра “Tetris”, 3 – игра “Flappy bird”). Правый массив – вывод на экран, в зависимости от состояния консоли.

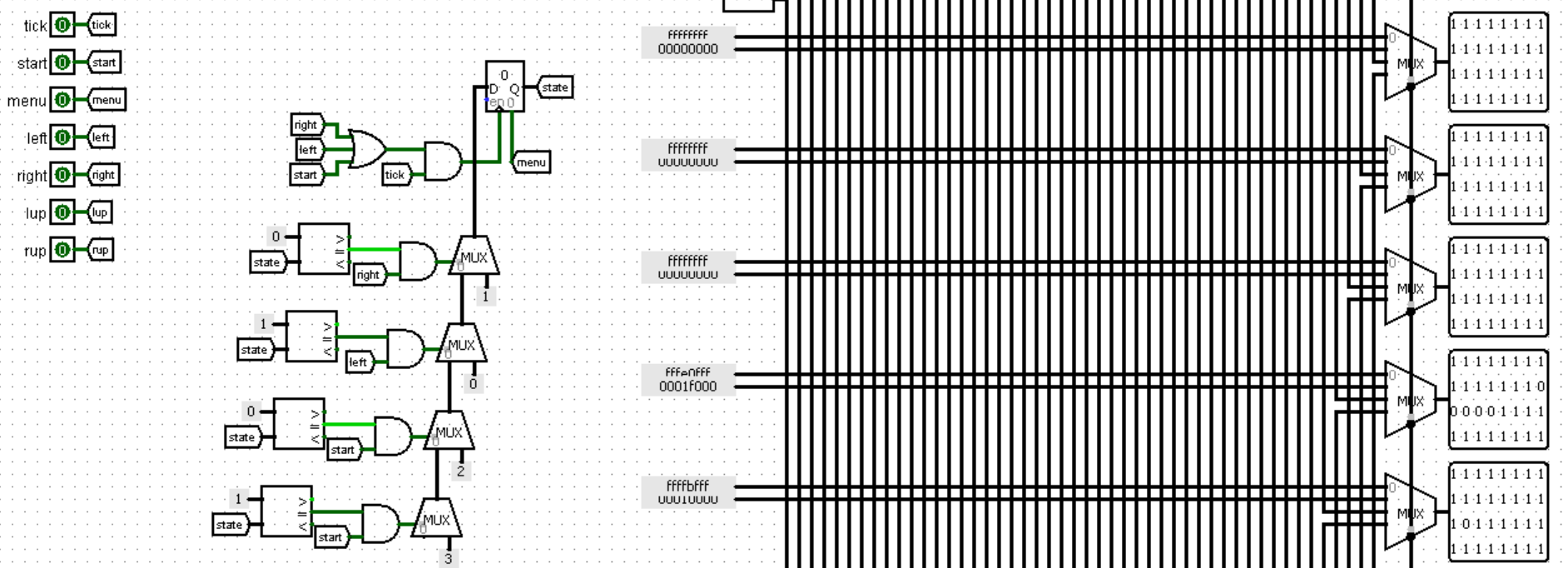


Рисунок 1

**Второй раздел: Tetris**

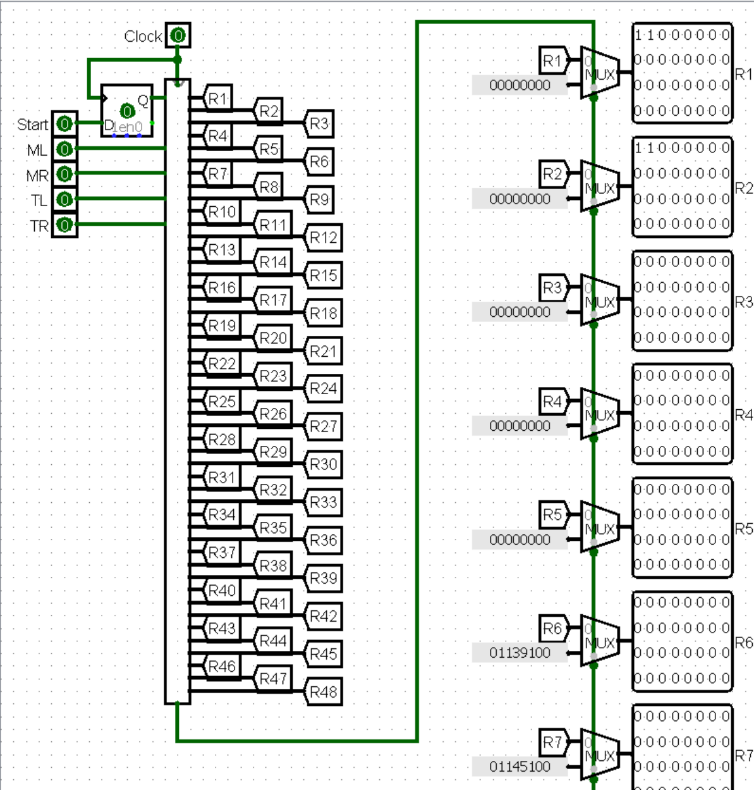
Главная схема игры представлена на рис.2.

Рисунок 2

Эта схема принимает на вход клавиши управления и такты, а выводит 48 32-битных сигналов, каждый из которых – строка поля. В зависимости от состояния игры (в процессе/проигрыш) на поле выводится либо экран поражения, либо сам игровой процесс.

На рис.3 представлена схема, отвечающяя за игровой процесс. На неё так же передаются все клавиши управления, а возвращаются 48 32-битных сигналов. В самом начале игры (сигнал set) первая подсхема “get\_block” (рис.4) возвращает координаты блока. После этого происходит его движение по полю с помошью подсхемы ”movement\_block” (). Каждый такт с помощью подсхемы “block\_output” (рис.5) его координаты преобразуются в 48 32-битных сигналов, демонстрирующих его положение на поле. С помощью массива гейтов “or” происходит вывод изображения поля. В случае, если происходит совпадение блока с нижней границей или уже лежащими блоками, что отслеживается с помощью подсхемы “collision” (рис.6), происходит сохранения состояния поля в подсхеме “field” (рис.7) и управление передаётся на вновь созданый блок. В случае переполнения поля поднимается выходной сигнал “fail”

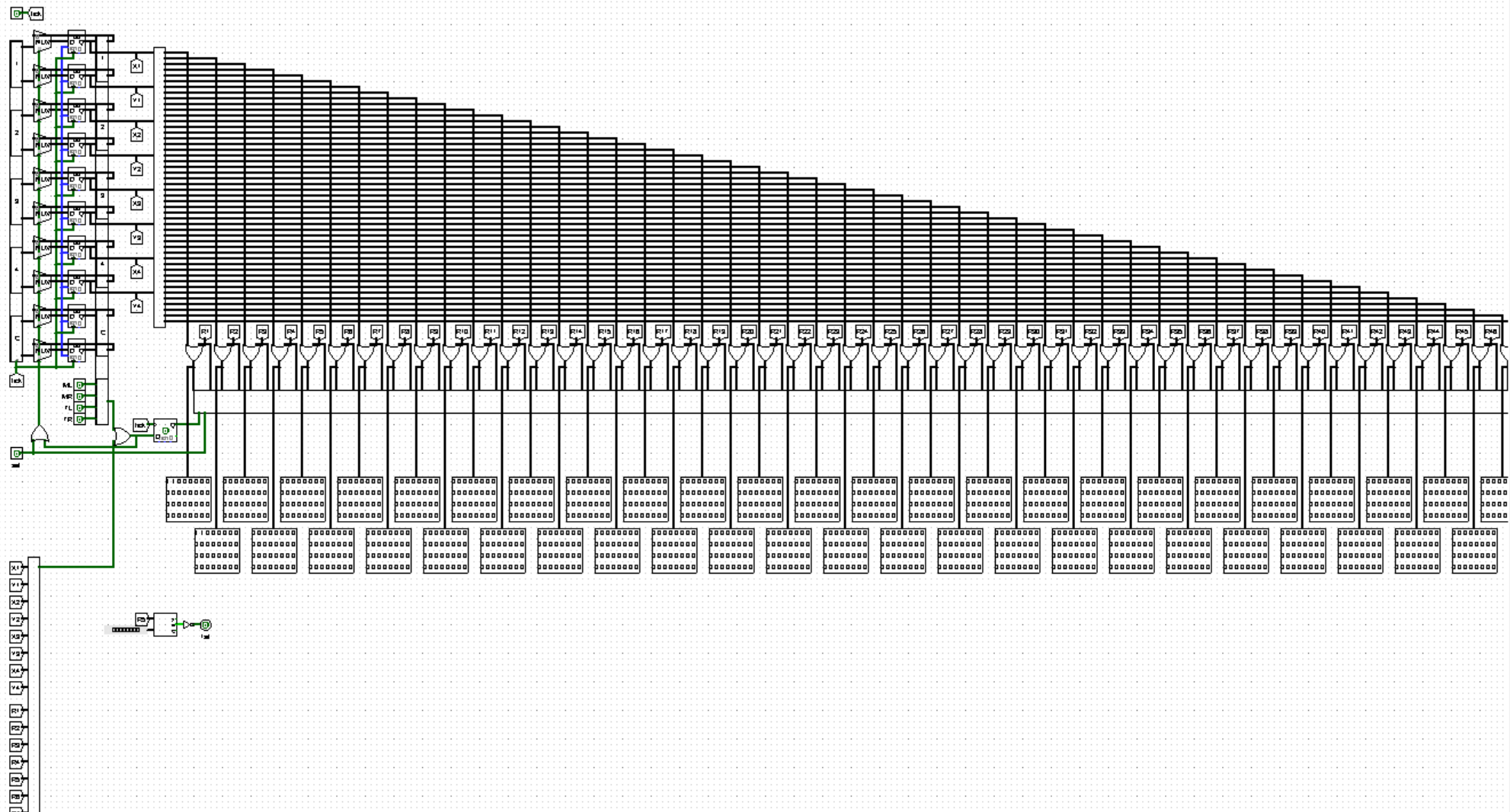


Рисунок 3

Подсхема “getblock” (рис.4) хранит в себе наборы данных всех возможных фигур тетрамино и в зависимости от сигнала, который выдаёт модуль рандома, возвращает конкретный набор.

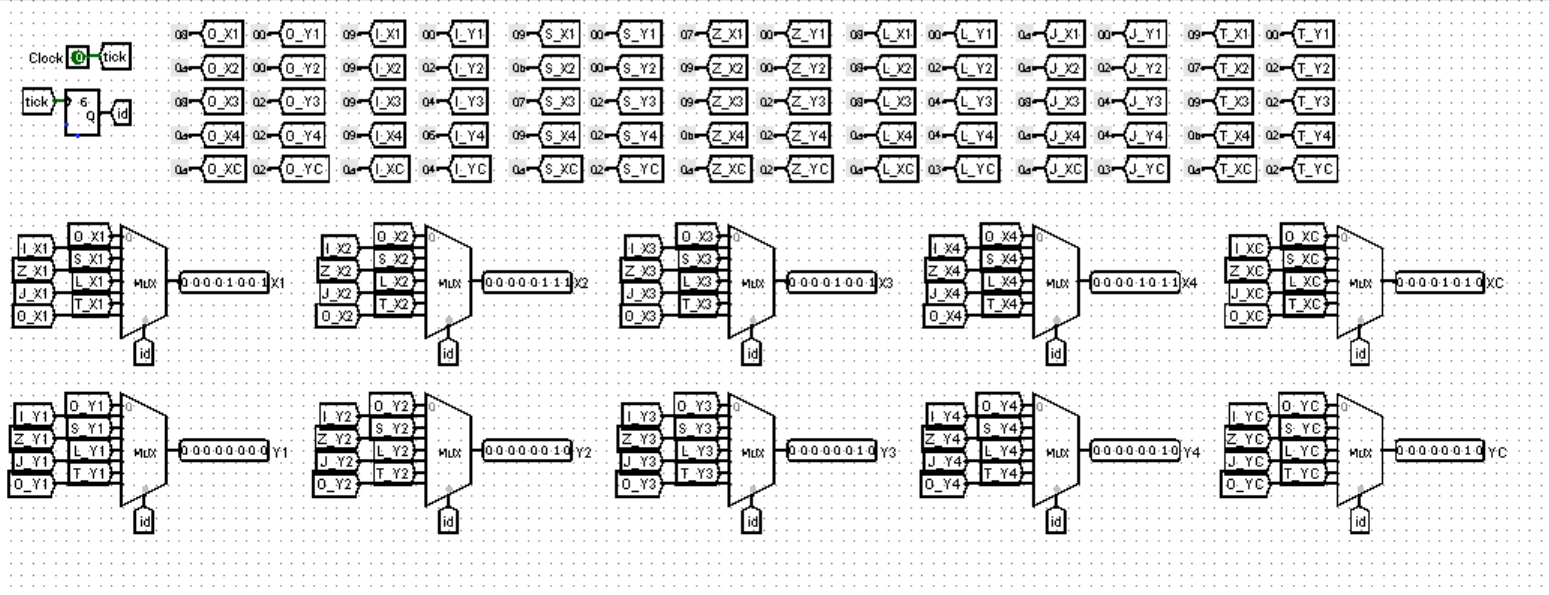


Рисунок 4

В схеме “block\_output” на каждую пару координат происходит два преобразования: преобразование 8-бит координаты X в 1 бит в 32-битном сигнале, представляющий собой строку поля, и вывод этого сигнала на определёную позицию (от 1 до 48 строки). С помощью двух массивов гейтов “or” так же происходит расширение изображения в два раза (вместо одного пикселя теперь отображается квадрат 2\*2)

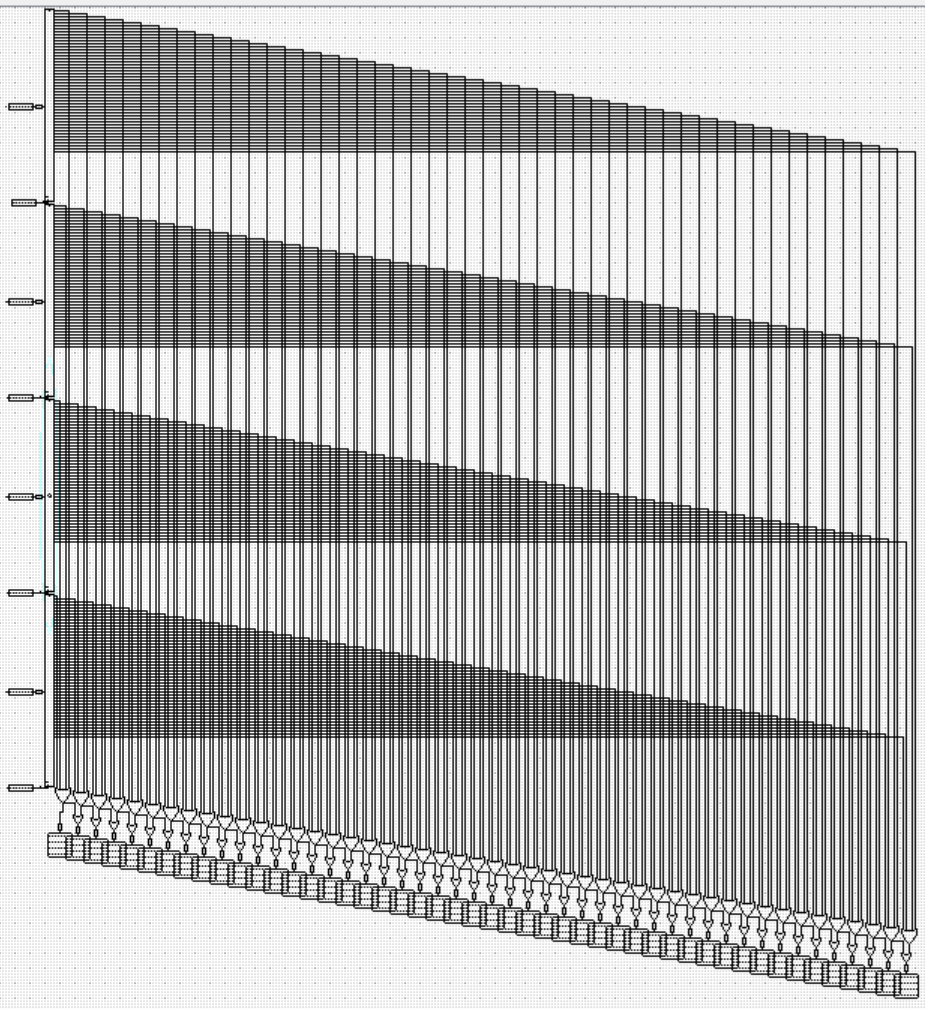


Рисунок 5

В схеме “collision” (рис. 6) на вход подаются координаты падающего блока и они преобразуются в 48 сигналов строк с помощью подсхемы “block\_output” (рис.5) и каждый из этих сигналов переадётся в тоннель с названием “RX”, где X – номер ряда. После этого проверяется, что ни на какой из строк поля (которые так же передаются в эту схему) не происходит соприкосновения падающего блока и блоков, лежащих на поле. Если это так, то выходной сигнал “collision” равен 0, иначе - 1.

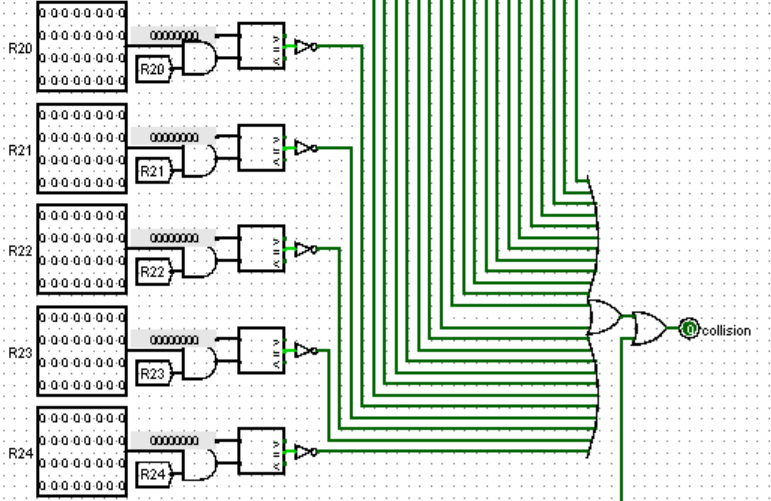


Рисунок 6

На рис.7 показана часть схемы “field”. На вход получаются 48 32-битных сигналов, каждый из которых – строка поля в нынешнем состоянии (в том числе с падающим блоком). С помощью несложной каскадной логики эти сигналы сдвигаются “вниз”, если обнаруживается “полная строка” (т.е. строка, в которой все клетки заняты блоком). В случае, если на схему подаётся сигнал “save” – сдвинутые строки сохраняются в регистры. На выходе схемы так же 48 32-битных сигналов, каждый из которых – состояниее строки поля, но только с уже упавшими блоками.

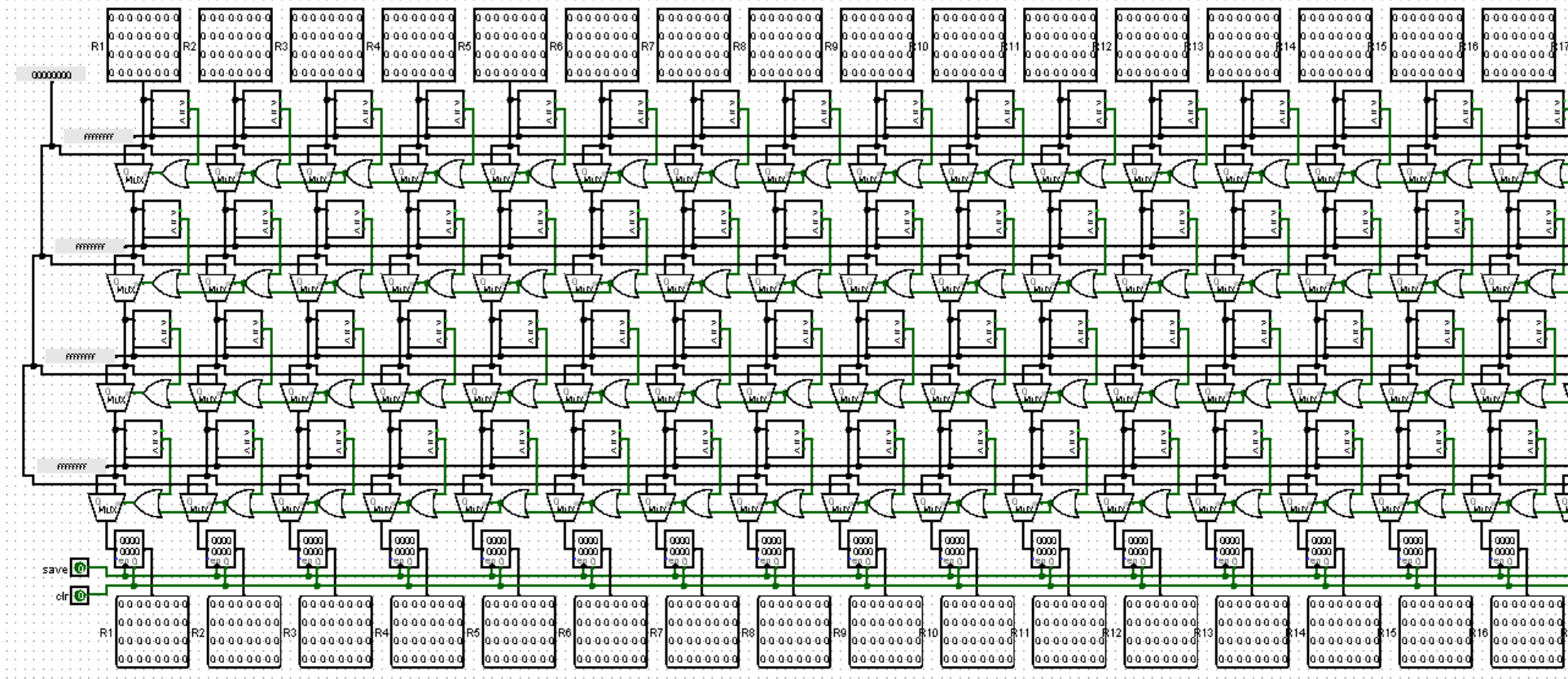


Рисунок 7

Схема “movement\_block” (рис.8) получает нынешние координаты падающего блока и клавиши управления. Следующим шагом происходят математические вычисления новых координат блока. В самое левой подсхеме – движение блока по горизонтали (увеличение/уменьшение координаты X). Во второй подсхеме – поворот блока вокруг своей оси (вычисление обоих координат с помощью математической формулы). Следующая подсхема является копией первой и досдвигает блок по горизонтали, если его координаты нечётны. Сделно это чтобы блок всегда находился в сетке 2\*2 и не возникало проблем при его фиксации. Последняя, самая правая, схема – движение блока вниз (уведичение координаты Y на 1). В случае, если блок достигает нижней границы поля – возвращается сигнал “Dropped”, который так же является и выходным сигналом схемы “movement\_block”. Новые координаты падающего блока являются выходными сигналами схемы.

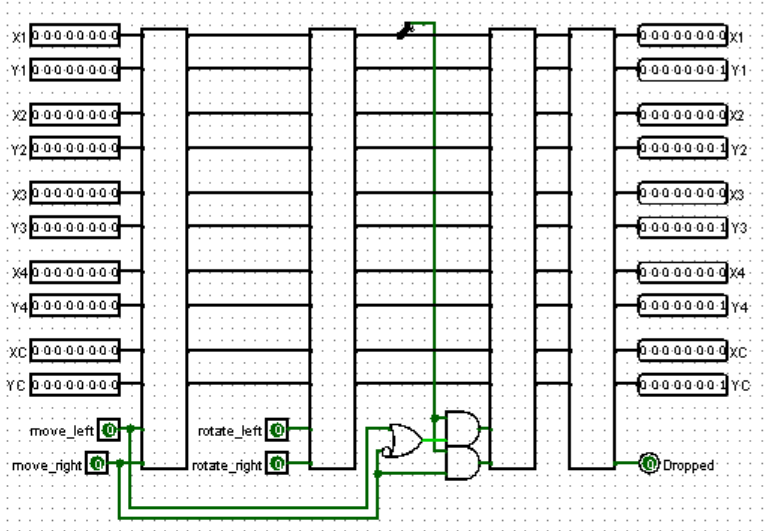


Рисунок 8

Все остальные схемы, использованные при создании игры, являются вспомогательными и не нуждаютсяя в пояснении логики своей работы.

**Третий раздел: Flappy bird**

Схема “screen\_united” (рис.9) является главной в этой части проекта. В ней происхоит объединение отображения моделей труб и модели персонажа. Так же эта схема выводит все 48 сигналов, необходимых для вывода игры на матрицу.

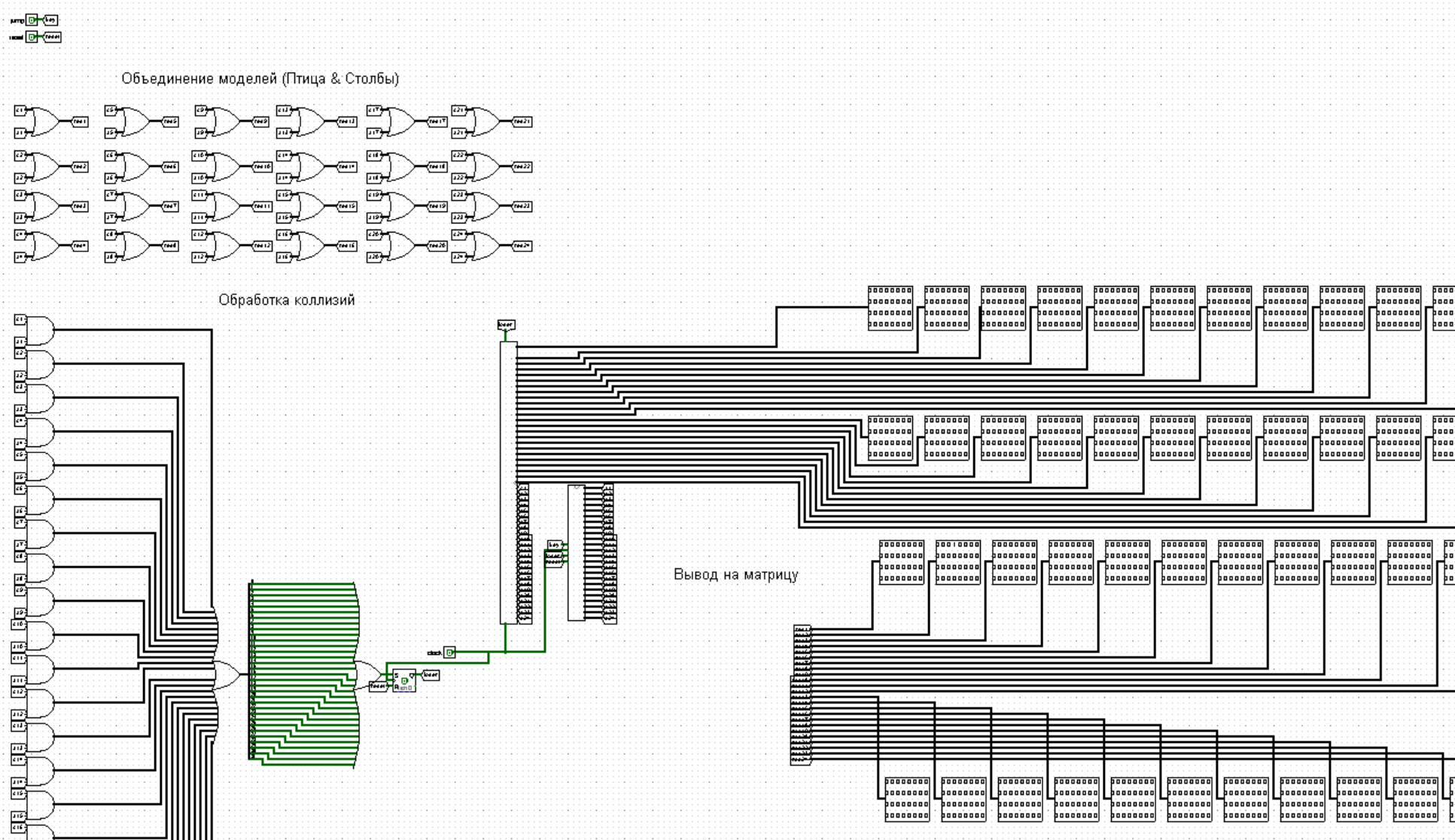


Рисунок 9

Схема, изображёная на рис.10, разработана для обрботки логики персонажа. Самая важная её часть – “Рассчёт высоты полёта”. Регистр в этой части схемы хранит координату модели персонажа, а остальные модули – меняют её в зависимости от того, падает персонаж или подпрыгивает.

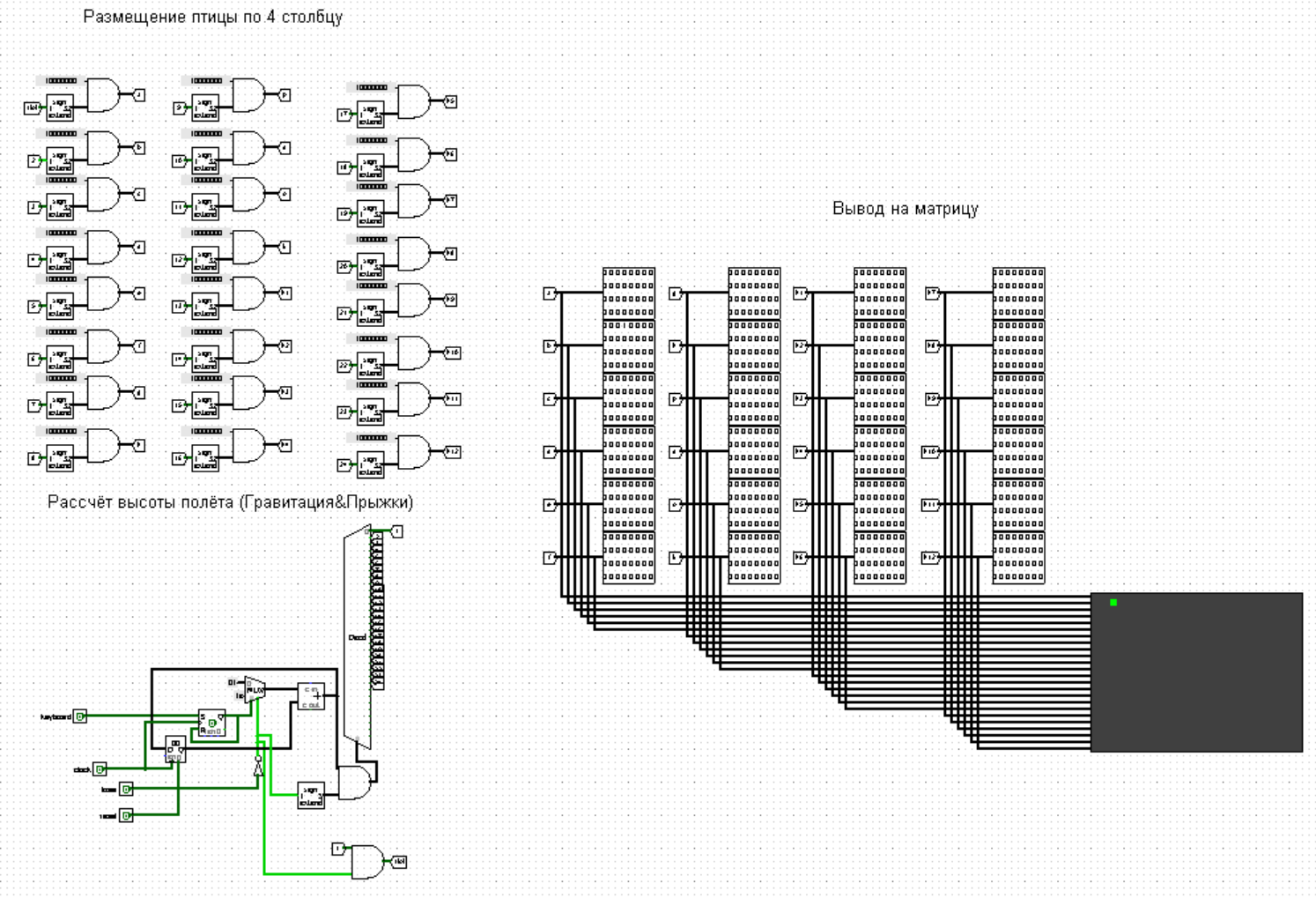


Рисунок 10

Схема с рис.11 выводит на экран колонны, создавая новую каждые n тиков.

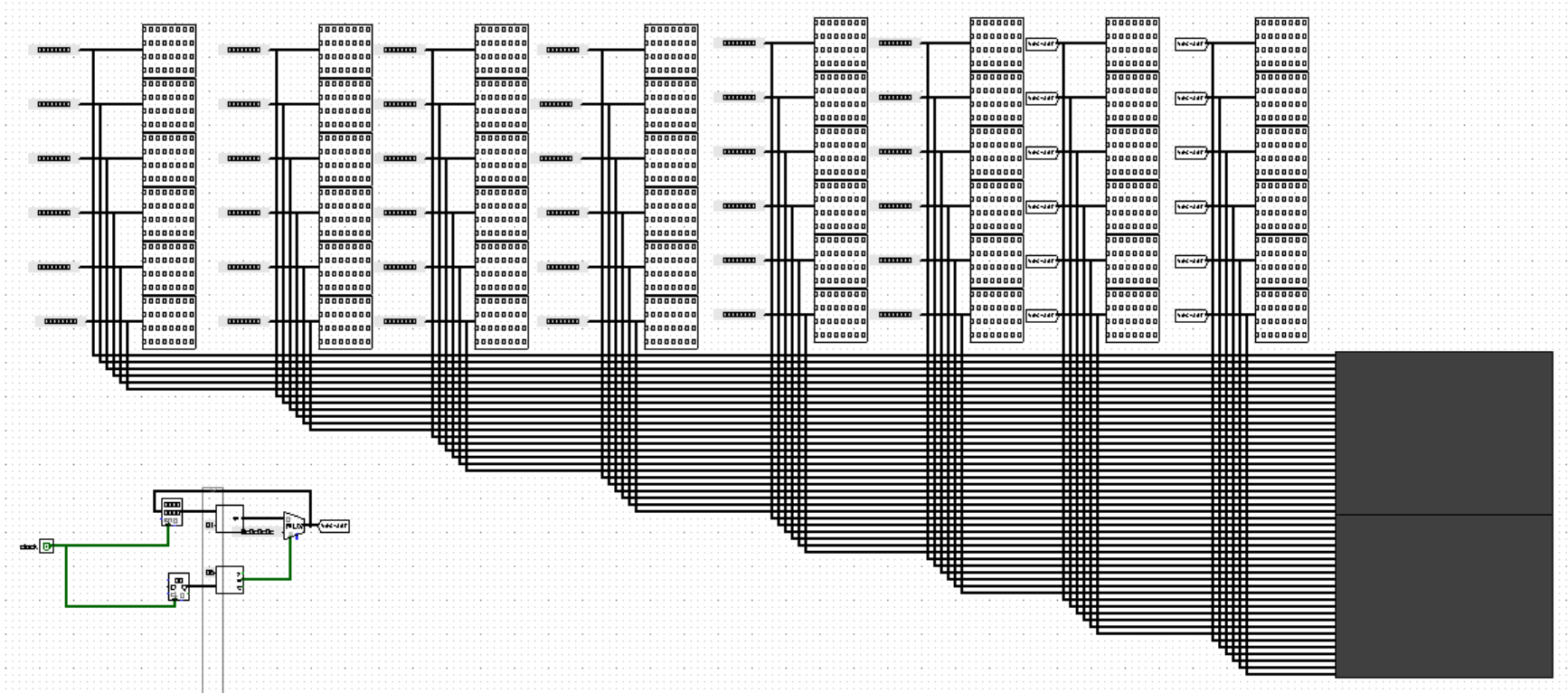


Рисунок 11

**Заключение**

Наша команда выполнила все поставленные цели, реализовав проект “Игровая консоль” и игры “Tetris” и “Flappy Bird”. Исключение составила задача по работе с функциональной частью.

**Список использованной литературы**

1. “Computing platforms”, A. Shafarenko and S.P. Hunt, School of Computer Science University of Hertfordshire 2015
2. [Тетрис — Википедия](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%82%D1%80%D0%B8%D1%81)
3. [Flappy Bird - Wikipedia](https://en.wikipedia.org/wiki/Flappy_Bird)