МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

15.03.06 - Мехатроника и Робототехника
Направление: Искусственный интеллект

Пояснительная записка

Тема работы: **«Rectify»**

Булыкин Сергей Васильевич,
24942
Морозова Софья Дмитриевна,
24942
Ребриков Никита Сергеевич,
24942

Новосибирск
2025

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

В рамках образовательной программы CS/MR Digital Platforms на учебный год 2024/25 каждый студент должен разработать и внедрить программное обеспечение для игровой системы, работающей на микроконтроллере CdM-8.

Название программы:
Электронная реализация игры «Rectify» на базе микроконтроллера CdM-8.

Rectify - это оригинальная игра, придуманная участниками проекта.

2 ЦЕЛЬ ПРОЕКТА

Разработка электронной реализации игры «Rectify» с использованием программного обеспечения Logisim с применением ассемблера для написания кода, управляющего работой логических элементов и компонентов схемы.

3 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Определение проблемы программы:
Современные образовательные программы требуют практического освоения принципов работы цифровых схем и микроконтроллеров. Данный проект решает задачу создания интерактивной игровой системы, которая:

- Наглядно демонстрирует работу логических элементов;
- Позволяет отработать навыки программирования на ассемблере;
- Служит примером проектирования законченного устройства на базе CdM-8.

Область применения программы:

- Образовательный процесс – изучение архитектуры микроконтроллеров, цифровой схемотехники и ассемблерного программирования.
- Прототипирование электронных устройств – игра может быть использована как основа для более сложных проектов, связанных с управлением периферийными устройствами.

4 ЗАДАЧИ ПРОЕКТА

- Определить основные логические операции и взаимодействия, необходимые для реализации игры.
- Спроектировать основные блоки игры (игровое поле, счетчик очков, контрольные элементы).
- Разработать архитектуру схемы и определить необходимые компоненты Logisim.
- Разработать программное обеспечение для управления логическими элементами.
- Реализовать обработку игровых событий и взаимодействие с пользователем.
- Обеспечить корректную работу всех компонентов схемы.
- Подготовить документацию по разработке, описать логику работы, структуру схемы и функции кода.
- Распределить роли и обязанности внутри команды (схемотехники, программисты, тестировщики, документалисты).
- Обеспечить эффективную коммуникацию и координацию между членами команды.
- Вести общий репозиторий и систему контроля версий для совместной работы над проектом.

5 ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ

В ходе разработки электронной реализации игры «Rectify» были использованы следующие программы и инструменты:

- Logisim — для проектирования и моделирования логической схемы;
- Среда разработки (cocoIDE с аторскими поправками) — для написания и отладки кода на языке программирования: Асемблер;
- Sourcetree, GIT — для организации совместной работы над проектом;

6 ИГРА «RECTIFY»

Rectify — это интеллектуальная игра. В ней участвуют два игрока. Они захватывают поле 8 на 8 при помощи очков энергии.

6.1 ПРАВИЛА ИГРЫ

Игровое поле:

Состоит из клеток, включает в себя генератор задач, таблицу для фиксации текущих заданий, систему учета очков прогресса и дополнительной энергии. Взаимодействие с игровым полем осуществляется посредством клавиатуры.

Клетка:

Трехзначное число и дисплей. Каждая клетка изначально имеет в себе 7 щитов и 3 энергии. Дисплей предназначен для идентификации принадлежности клетки к определенному игроку и демонстрации задания. На дисплей для чисел выводиться только количество энергии.

Щиты - броня клетки. Захват клетки невозможен, если у нее есть щиты. Они восстанавливаются после завершения хода обоих игроков.

Энергия - ресурс клетки для захвата. Её можно перемещать.

Задание:

Прямоугольник, размещаемый на свободных клетках и граничащий с клеткой игрока. В генераторе задач отображаются два значения, представляющих длины сторон прямоугольника. Допустимые значения длины сторон варьируются от 1 до 4 единиц.

Задание создается на этапе планирования. Игрок может принять или отклонить его: в первом случае параметры прямоугольника сохраняются в таблице. Максимум игрок может принять пять заданий. Во втором случае игрок переходит в следующую фазу игры.

Для выполнения задания нужно захватить весь прямоугольник, размещенный на поле. Его площадь конвертируется в очки прогресса и в дополнительную энергию.

Очки прогресса: Площадь задания делится на четыре и округляется в большую сторону.

Дополнительная энергия: Площадь задания умножается на два.

Пример: Оранжевый игрок выполняет задание 3 2. Он получает 6 дополнительной энергии и 2 очка прогресса.

6.2 ИГРОВОЙ ПРОЦЕСС

Подготовка:

Игроки выбирают точку старта. Эти клетки красятся в цвет игрока и получают стартовые 10 энергии. (Можно выбирать любую стартовую позицию, но убедитесь, что между выбранными клетками есть минимум две свободные.)

Игра:

Игроки ходят по очереди. Ход каждого игрока состоит из четырех фаз. По окончании этих фаз ход передается другому игроку.

- *Фаза планирования*

Генератор задания создает два числа.
Игрок решает берет ли он задание и проверяет есть ли место в таблице.

Условия размещения задания:
Размещается на свободных клетках и граничит с клеткой игрока.
Прямоугольник должен быть размещен так, чтобы уже захваченные игроками клетки в нём не находились.
Задание можно поворачивать.
Принимая задание, игрок обязуется его выполнить (захватить все клетки выделенного прямоугольника), после выполнения игрок получит вознаграждение.
Игрок 1 может захватить клетку с заданием игрока 2. Для выполнения задания игроку 2 надо захватить клетку игрока 1 со своим заданием.

- *Фаза перемещения*

Игрок планирует перемещение энергии с одной клетки на другую. Можно передавать только ту энергию, которая была на начало хода игрока. Атаковать можно свободную клетку или занятую другим игроком.

Атака по свободной:

У свободной клетки 7 щитов и 3 энергии, чтобы её захватить, нужно потратить 10 энергии.

Атака по занятой:

Для захвата нужно перенести энергию, чтобы пробить 7 щитов, а после перебить значение противника. Когда игрок перенес энергию равную 7 щитам и количеству энергии противника, клетка становиться ничейной. Чтобы забрать её себе, нужно перенести хотя бы еще 1 энергию.

Если клетка после боя осталась ничейной, то она больше не вырабатывает энергию. То есть, для захвата надо будет пробить щиты клетки.

- *Фаза зарядки*

Каждая принадлежащая клетка игроку увеличивает свою энергию на 2 (свободные клетки не получают эту энергию). И каждая клетка восстанавливает щиты (7 единиц).

Передача энергии перемещенной игроком за фазу перемещения.

- *Фаза распределения*

Игрок может распределять дополнительную энергию, если она есть. Но можно и не использовать её.

Конец игры:

Есть два варианта определить победителя

- Побеждает игрок, занявший все поле.
- Игрок с наибольшим количеством очков прогресса. (Остановить игру можно только по обоюдному согласию)

6.3 УПРАВЛЕНИЕ

Управление в игре осуществляется с помощью кнопок клавиатуры.

a	Движение курсора влево
d	Движение курсора вправо
w	Движение курсора вверх
s	Движение курсора вниз
ENTER	Многофункциональная кнопка, используемая для подтверждения ввода.
Backspace	Многофункциональная кнопка, используемая для отмены действия.
A	Перемещение энергии влево
D	Перемещение энергии вправо
W	Перемещение энергии вверх
S	Перемещение энергии вниз
-	Позволяет исправлять количество переданной энергии

Последовательности:

- Выбор клетки старта: wasd - выбор клетки для начала, ENTER подтверждения выбора.

- Передача энергии: wasd - выбор клетки откуда игрок возьмет энергию. SHIFT и wasd - для выбора направления. Ввод в клавиатуру значения для передачи. ENTER подтверждения выбора. Можно повторить операцию с ‘-’ для того чтобы уменьшить количество передаваемой энергии.

- Размещение задания: wasd - выбор клетки рядом с которой игрок захочет разместить задания. Выбор брать задание или нет: брать - ENTER, не брать Backspace.
  wasd - выбор направления размещения задания.
  Ввод с клавиатуры - перемещает задание в плоскости, перпендикулярной выбранному направлению, относительно курсора.

В любой момент можно сделать шаг назад при помощи Backspace.

7 ФУНКЦИОНАЛЬНЫЕ ХАРАКТЕРИСТИКИ

Программная часть проекта представляется ассемблерным кодом в среде CocoIDE.

Главная задача ассемблерной программы: контролировать игру и реализовывать логически-сложные ситуации,например, при запуске игры, CDM-8 запускается первым и очищает поле, подготавливает его к игре, далее игроку предлагается выбрать стартовую клетку.

По ходу игры за обработку входящих команд будут отвечать два контроллера: CDM-8 - основной процессор игры и basic controller - вспомогательный контроллер. CDM-8 будет периодически “засыпать”, передавая управление вспомогательному контроллеру, который отвечает за действия, которые не требуют сложных вычислений. Так, напрмер, в начале игры CDM-8 передаёт управление вспомогательному контроллеру, игрок при помощи клавиатуры выбирает клетку. Как только игрок выбрал клетку, управление передаётся обратно CDM-8, который инициализирует выбранную клетку.

По ходу игры за генерацию заданий отвечает отдельный блок task_manager, CDM-8 к нему обращается и запрашивает задание, блок отправляет это задание, а затем CDM-8 даёт игроку выбор: брать задание или нет. В случае взятия задания, даёт игроку выбор как ставить данное задание. Затем снова CDM-8 подаёт команду Basic_controller’у о том, что теперь можно перемещать энергию между клетками, после этого игрок попадает в фазу перемещения. После завершения хода управление возвращается к CDM-8(т.е. к ассемблерному коду) и он уже инициирует процедуру завершения хода и проверки выполнения задания. В итоге ассемблерный код управляет всем, раздаёт команды и обрабатывает тяжёлые случаи (например, как поместить задание, выполнил ли его игрок или нет) В более простых ситуациях используется множество вспомогательных контроллеров для оптимизации и ускорения работы .

Во время написания кода, мы столкнулись с проблемой нехваткой памяти.
Решение:
Добавление файла нового типа .asmh:

- Этот файл описывает, какие тетрады (модули) присутствуют в проекте.
- Указывает, какие файлы нужно скомпилировать и слинковать (связать) для сборки проекта.

Доработка цикла симуляции:

- Эмулятор теперь переключается между тетрадами в среде CoCoIDE так же, как это происходит в среде LGSim.

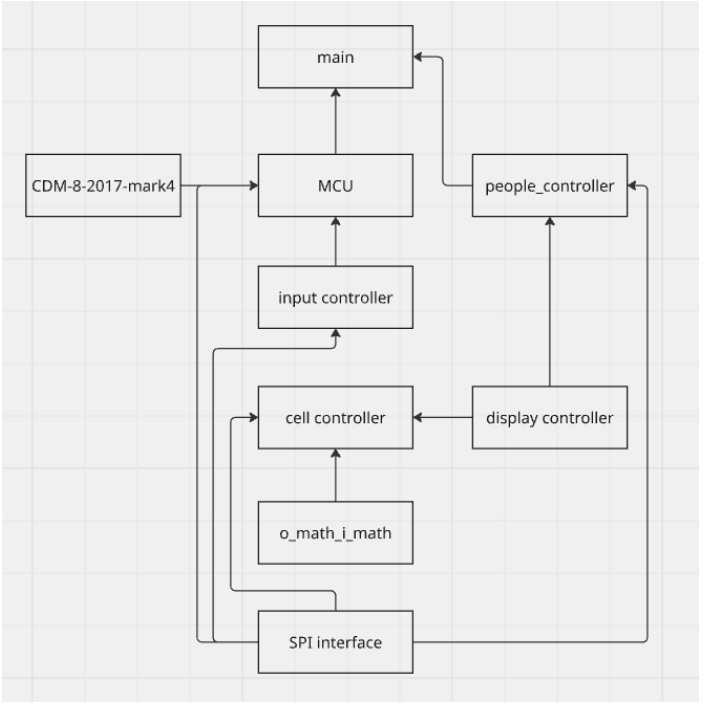
Добавление удобных кнопок:

- Кнопки для имитации работы внешних устройств.

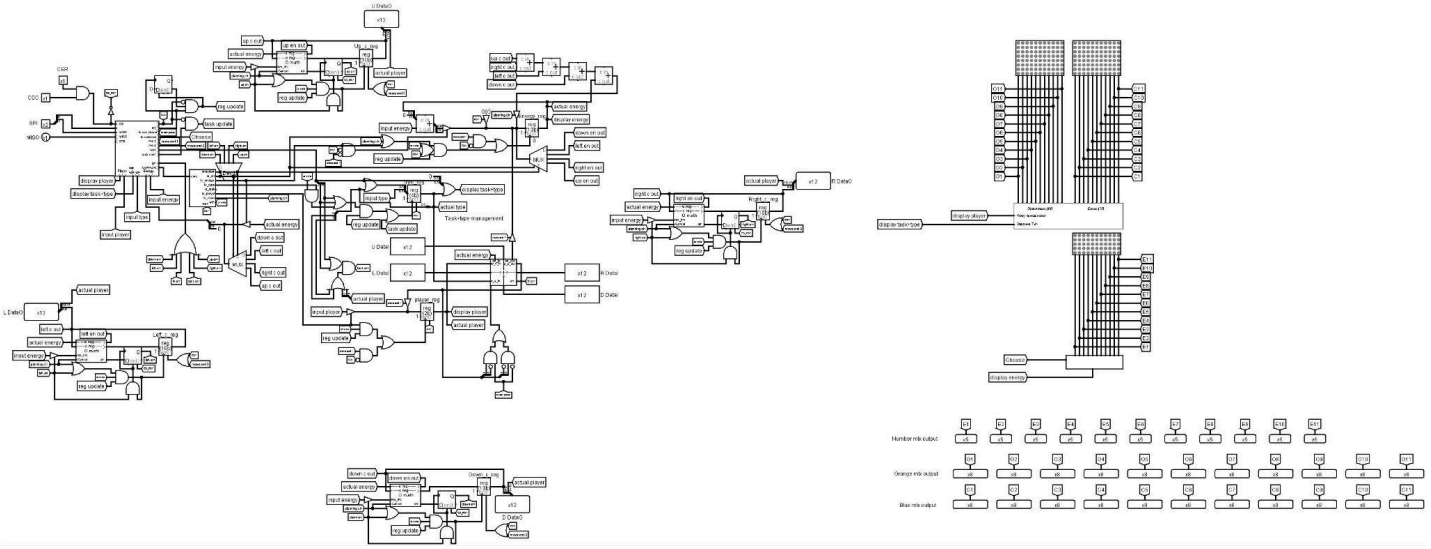
Эти изменения расширили возможности памяти и упростили отладку программ.

8 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Аппаратная часть проекта представлена набором взаимосвязанных цифровых схем, созданных в Logisim.



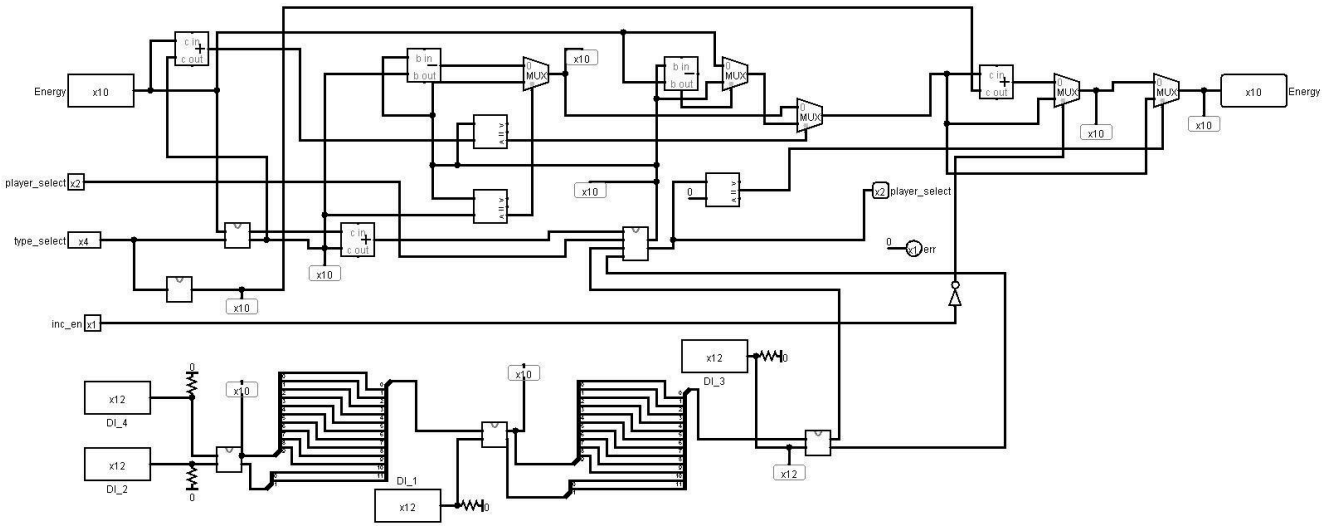
8.1 Cell_controller



Скриншот 1. “Схема cell_controller”

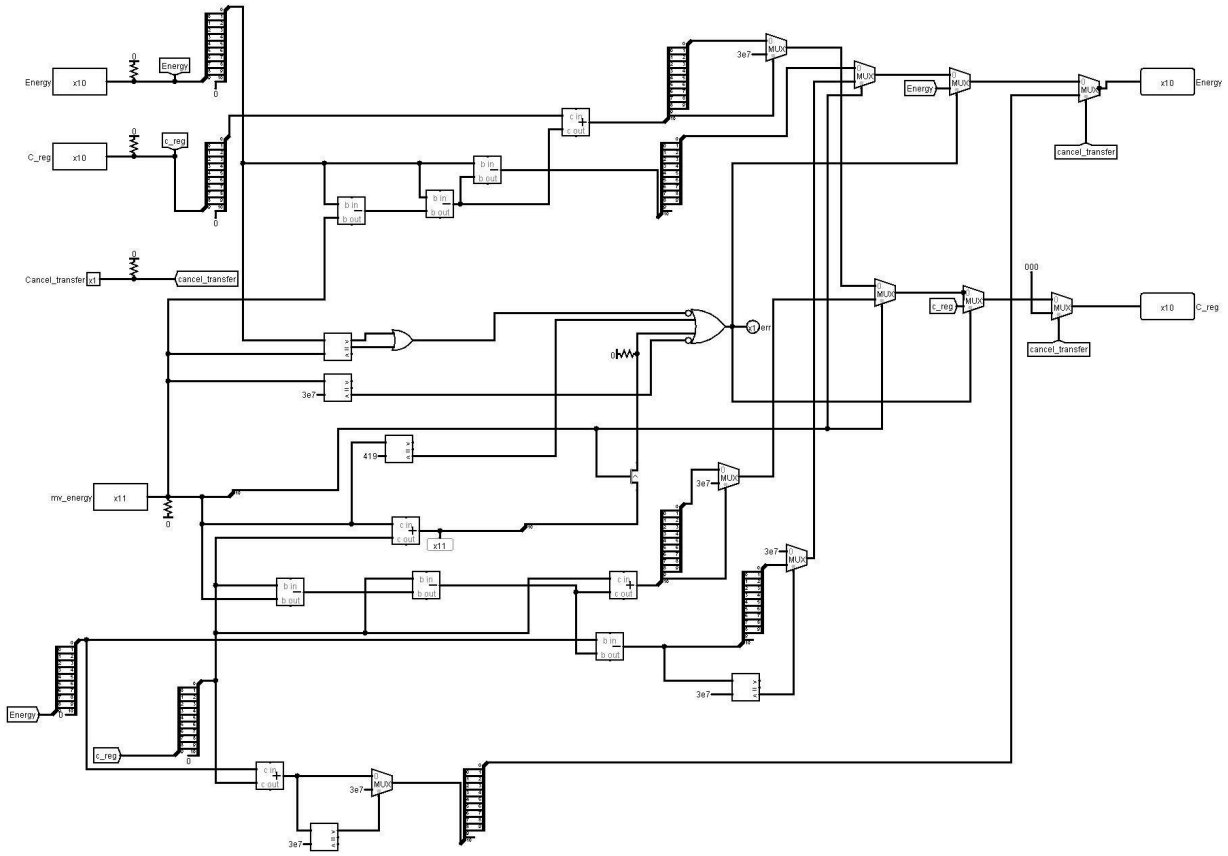
Данная схема позволяет хранить и выводить задание, принадлежность игроку и количество энергии в ней. Взаимодействует с соседними клетками реализовано в подсхеме “o_math” и ” i_math”. Реализовано взаимодействие клетки и процессора.

8.2 o_math_i_math



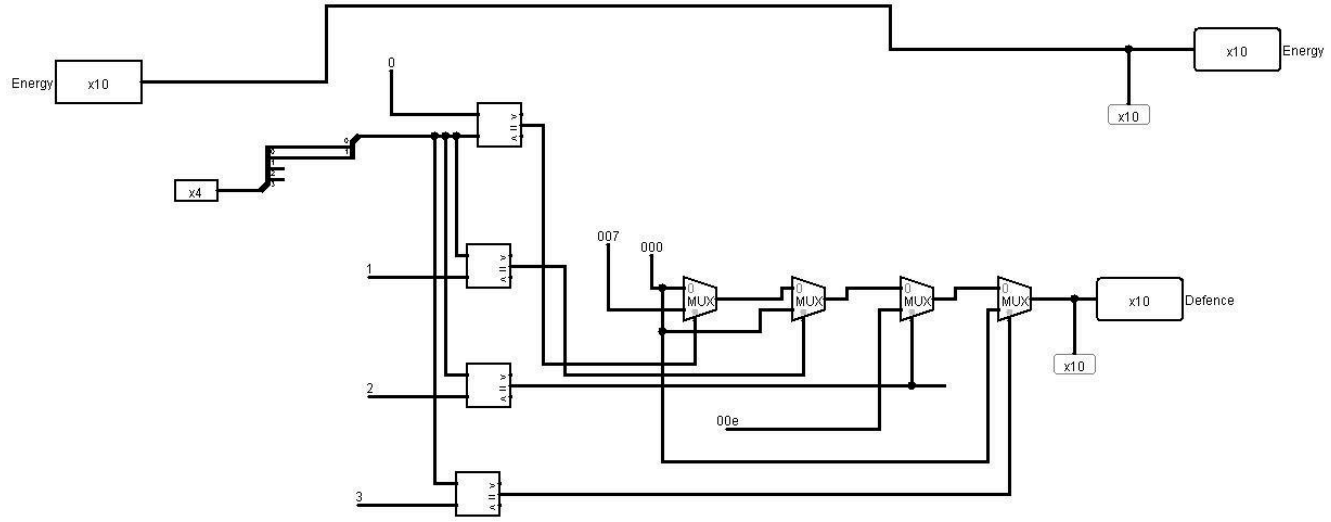
Скриншот 2: “Подсхема i_math”

Данная подсхема реализовывает логику того, как клетка принимает энергию из соседних клеток.



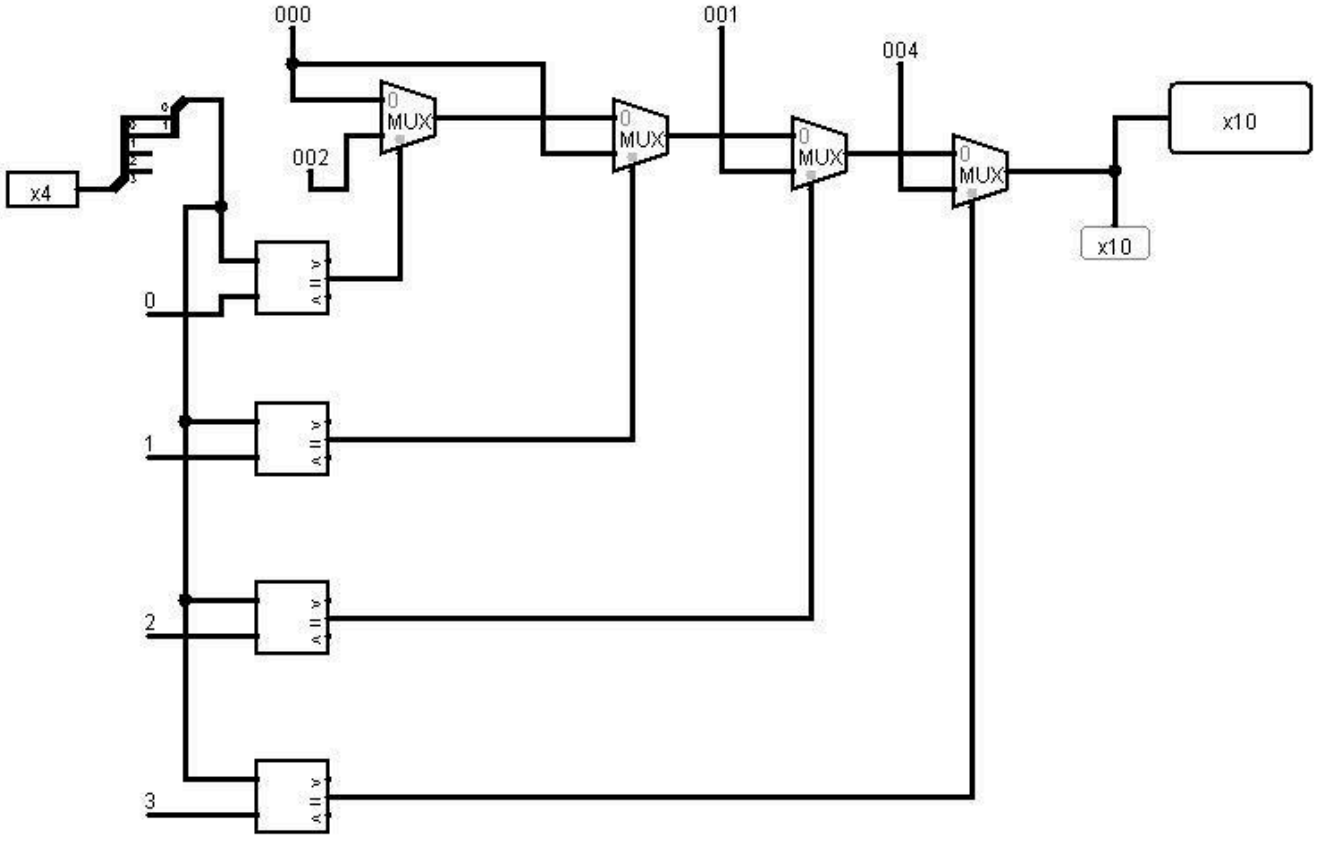
Скриншот 3: “Подсхема o_math”

Данная подсхема реализовывает логику того, как клетка перемещает энергию в соседнюю с ней.



Скриншот 4: “Подсхема type_math”

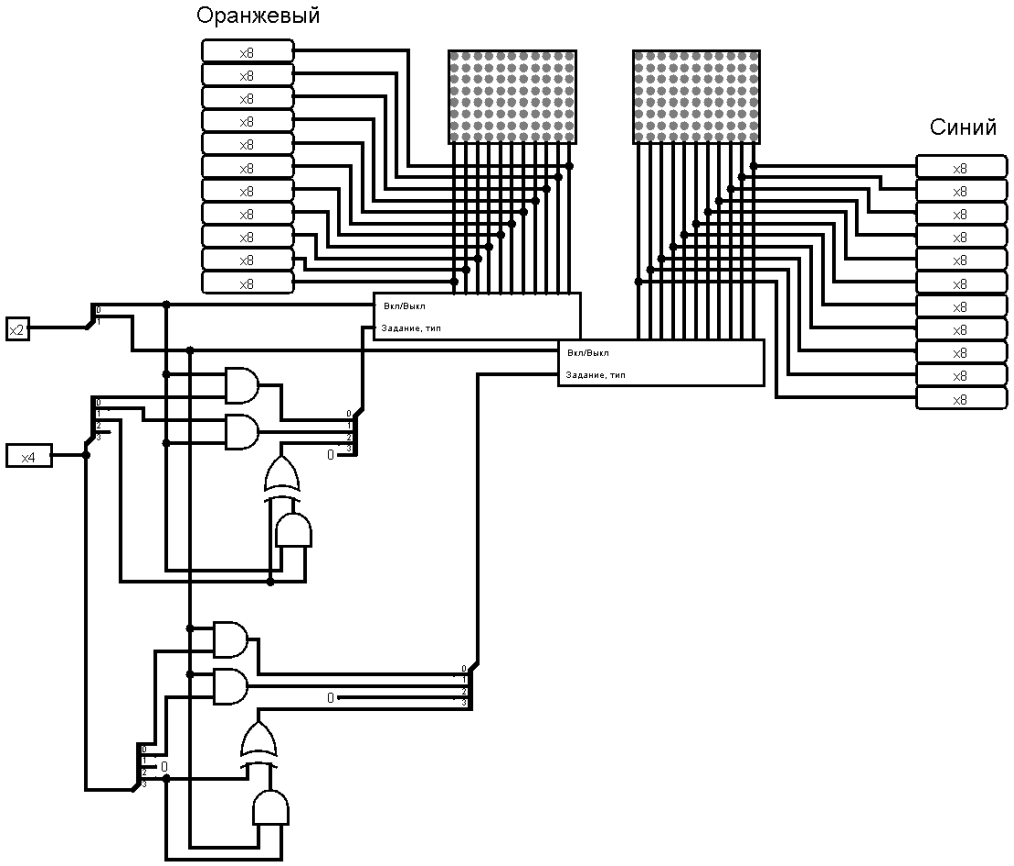
Подсхема определяет уровень защиты клетки и способы взаимодействия с ней.



Скриншот 5: “Подсхема increment”

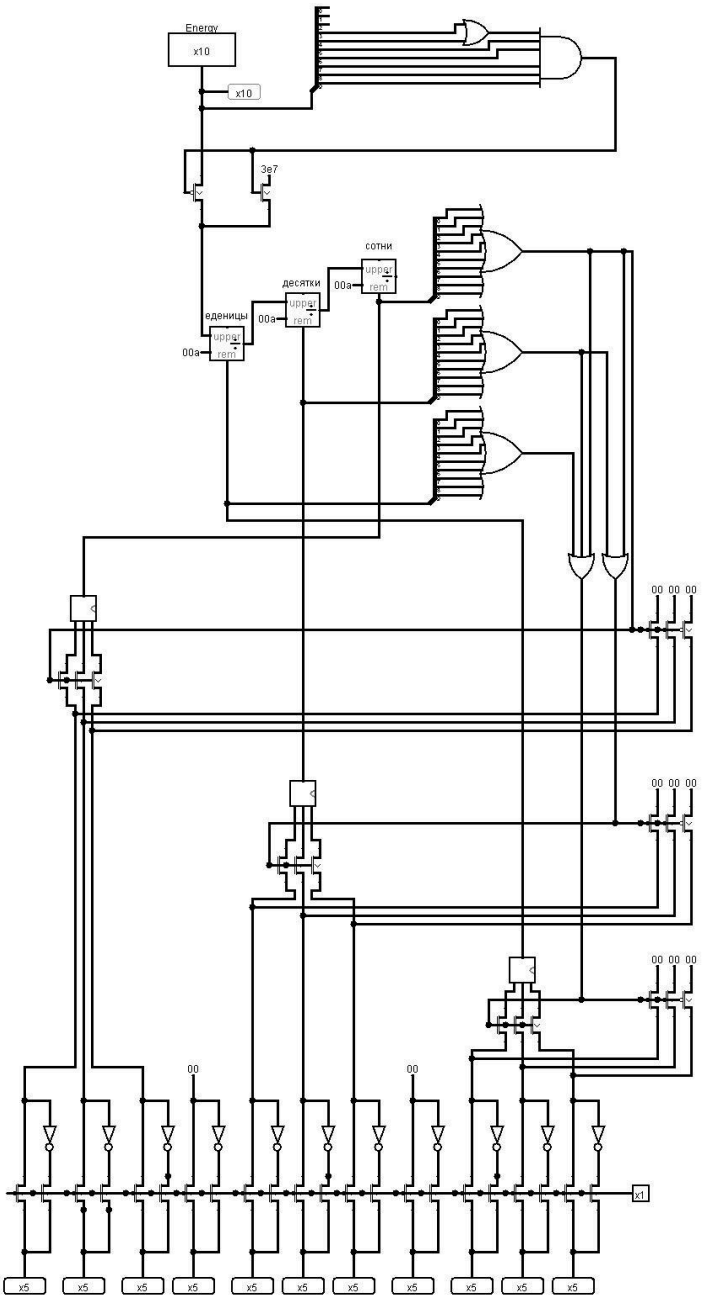
Схема для математических операций пополнения энергии в клетке.

8.3. display_controller



Скриншот 6: “Схема Player_type”

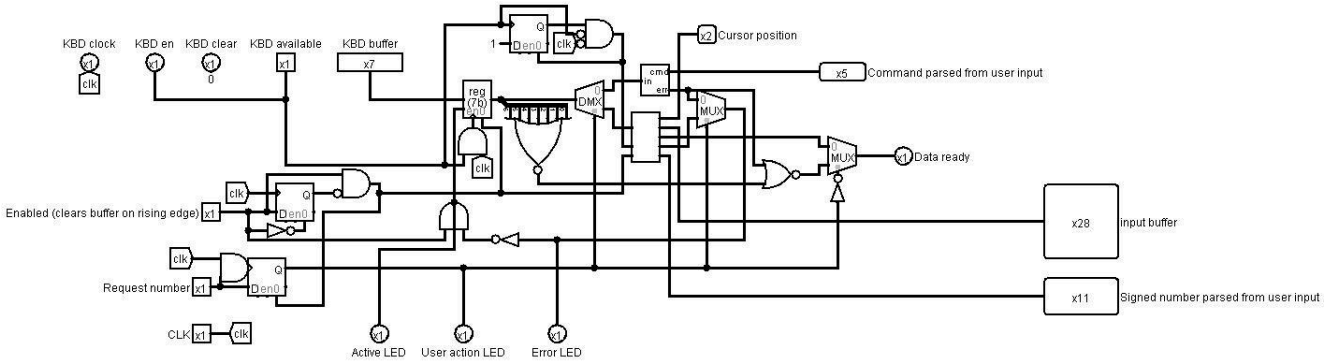
Эта схема показывает, кому принадлежит клетка — оранжевому или синему игроку, а также указывает, является ли клетка заданием.



Скриншот 7: “Схема Numbers”

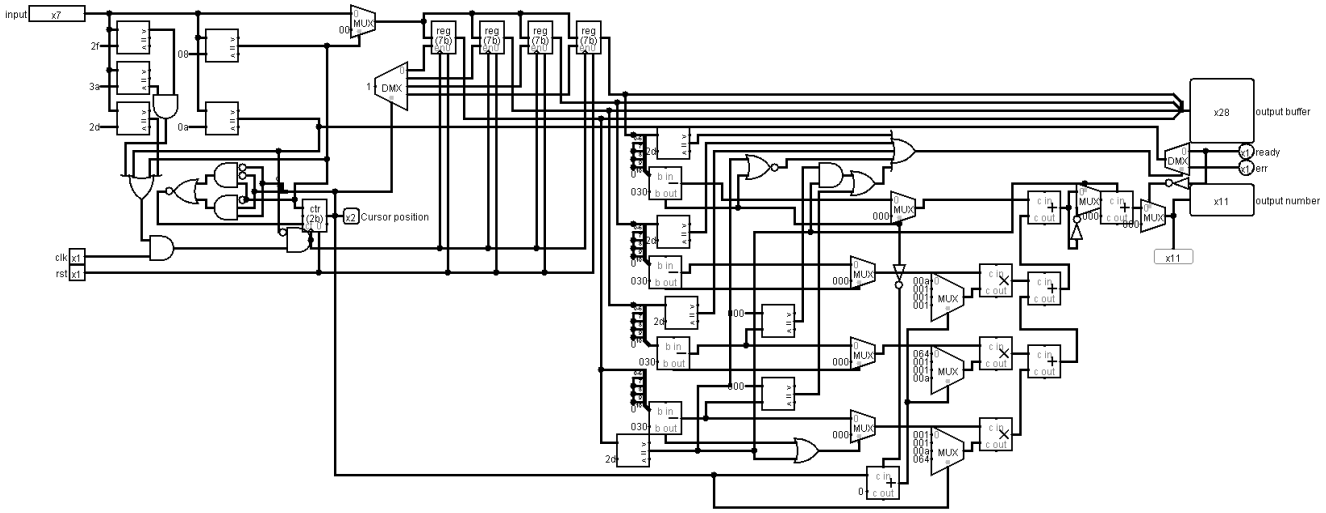
Выводит цифры числа на экран матрицы, показывая игроку количество энергии в каждой клетке. Максимальное число для вывода “999”.

8.4 input_controller



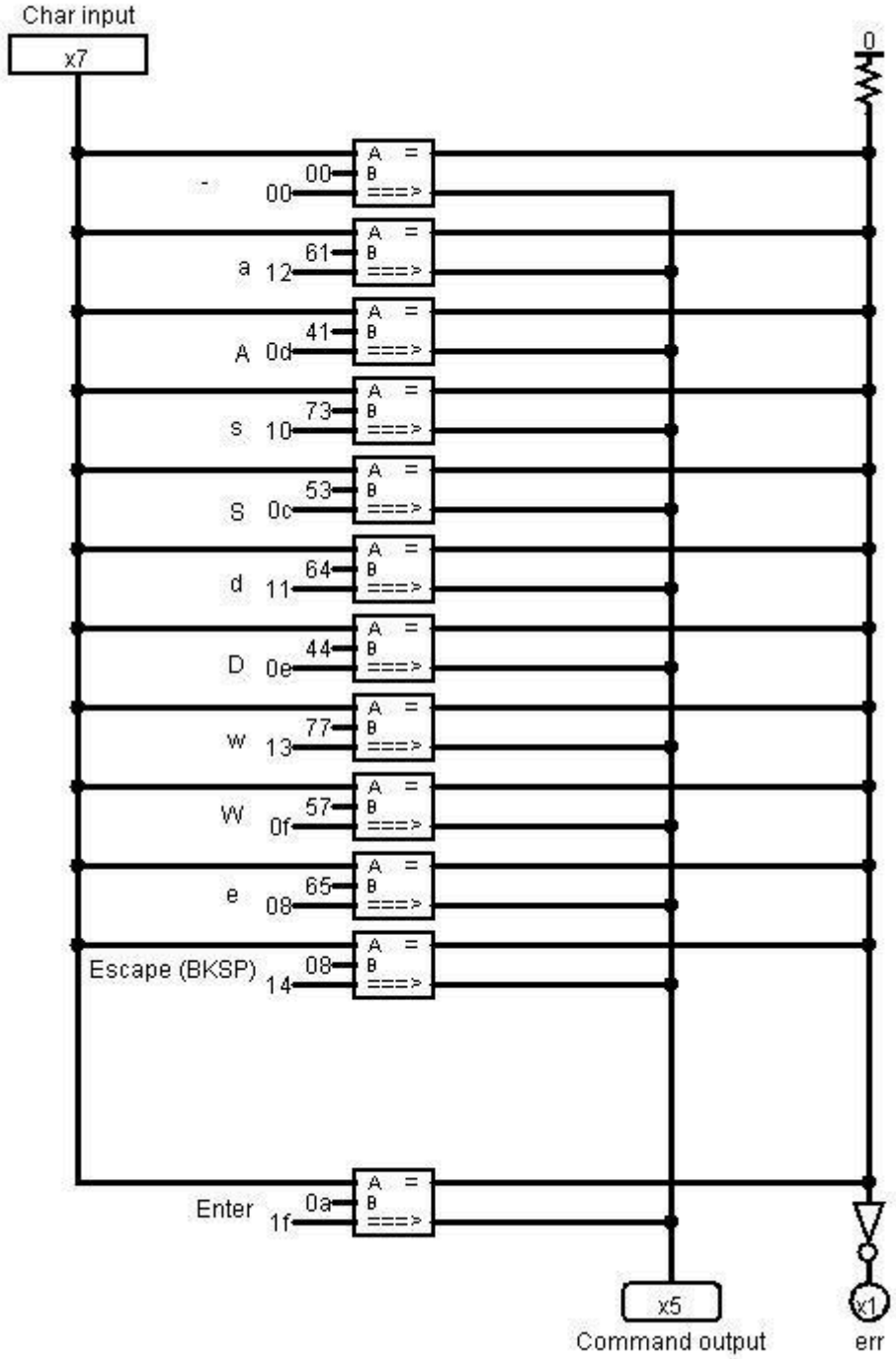
Скриншот 8: “Схема input controller”

Схема реализует считывание ввода команд и чисел игроком с клавиатуры.



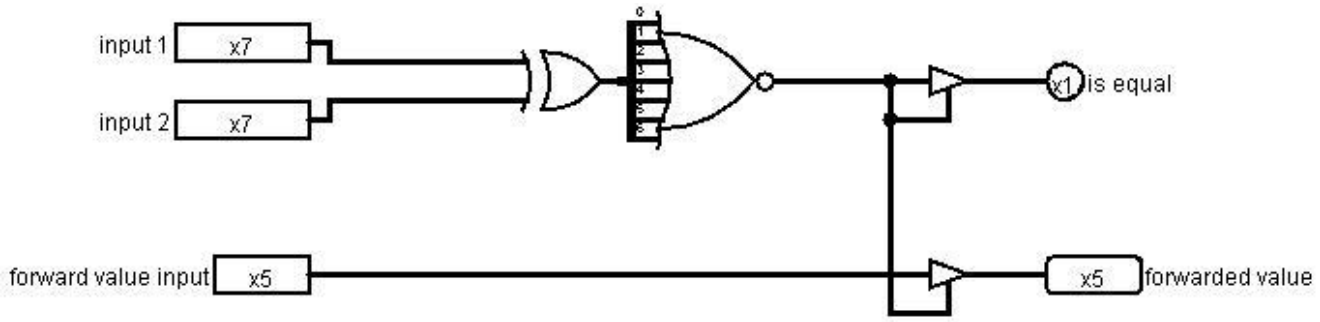
Скриншот 9: “number_parser”

Схема, отвечающая за преобразования пользовательского ввода в числа.



Скриншот 10: “Подсхема command_decoder”

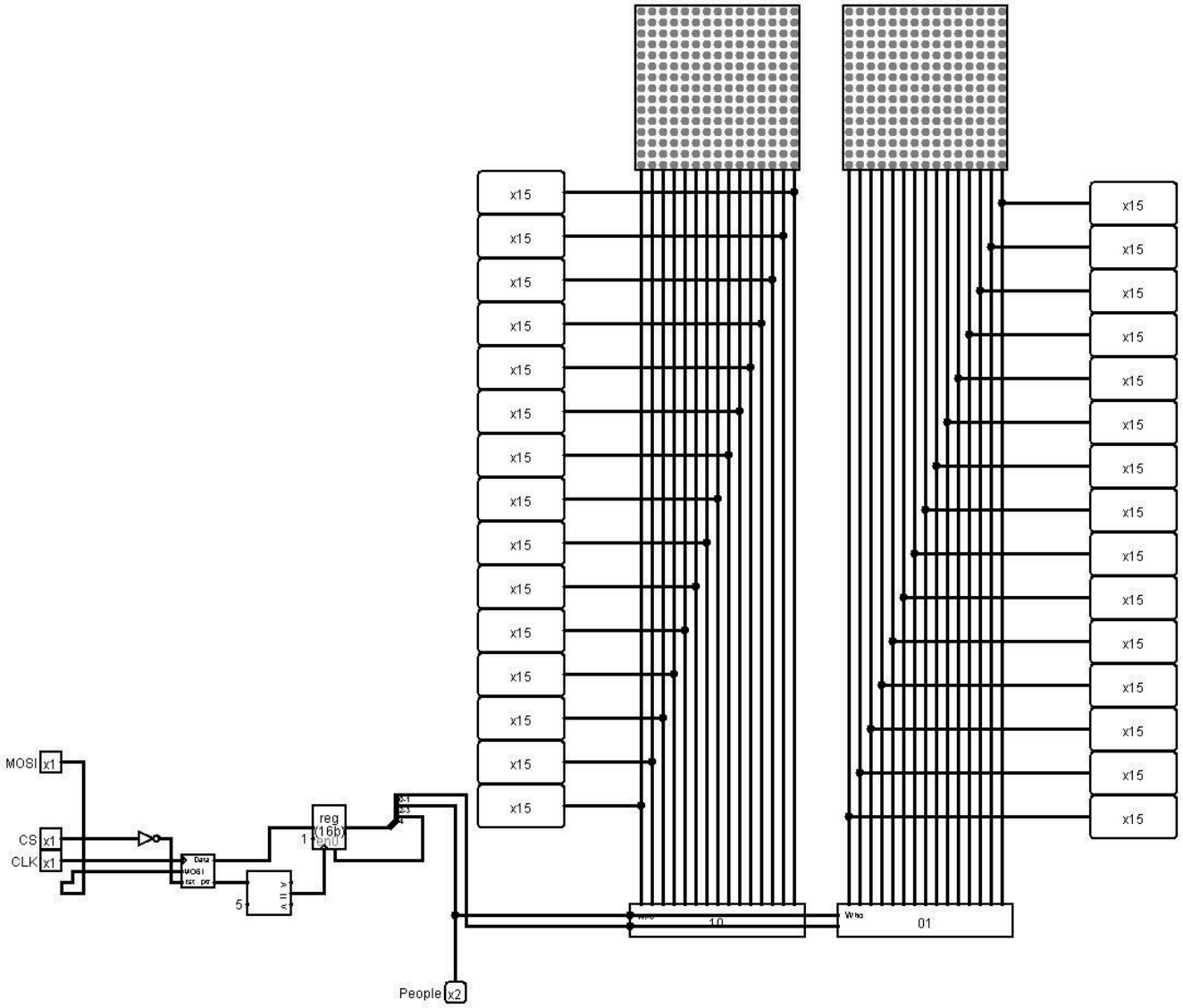
Декодер пользовательских команд. Он переводит символы ASCII в команды, понятные всем устройствам в схеме.



Скриншот 11: “Подсхема internal_comparator”

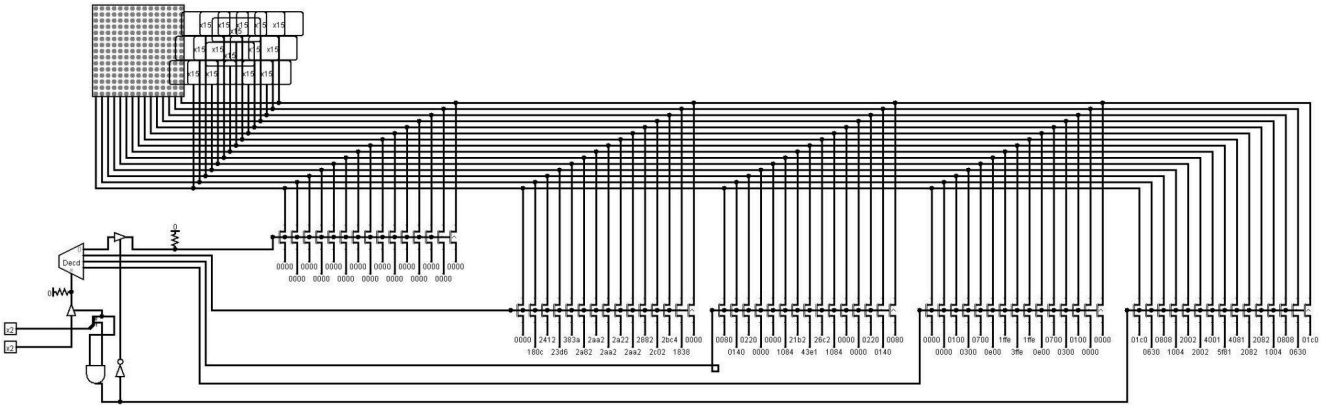
Вспомогательная схема для реализации декодера пользовательских команд.

8.5. People_controller

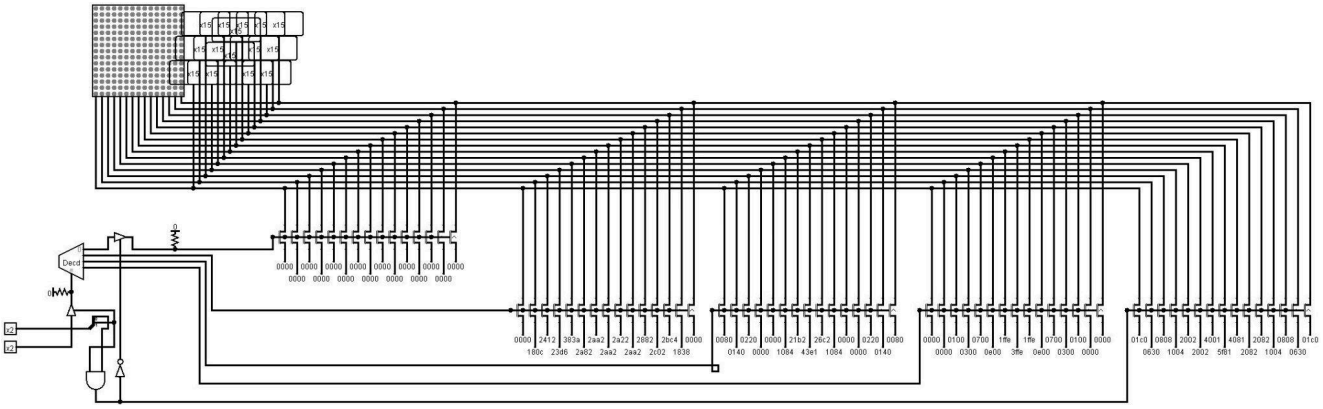


Скриншот 12: “Схема Fasa”

Отображает фазу игры.

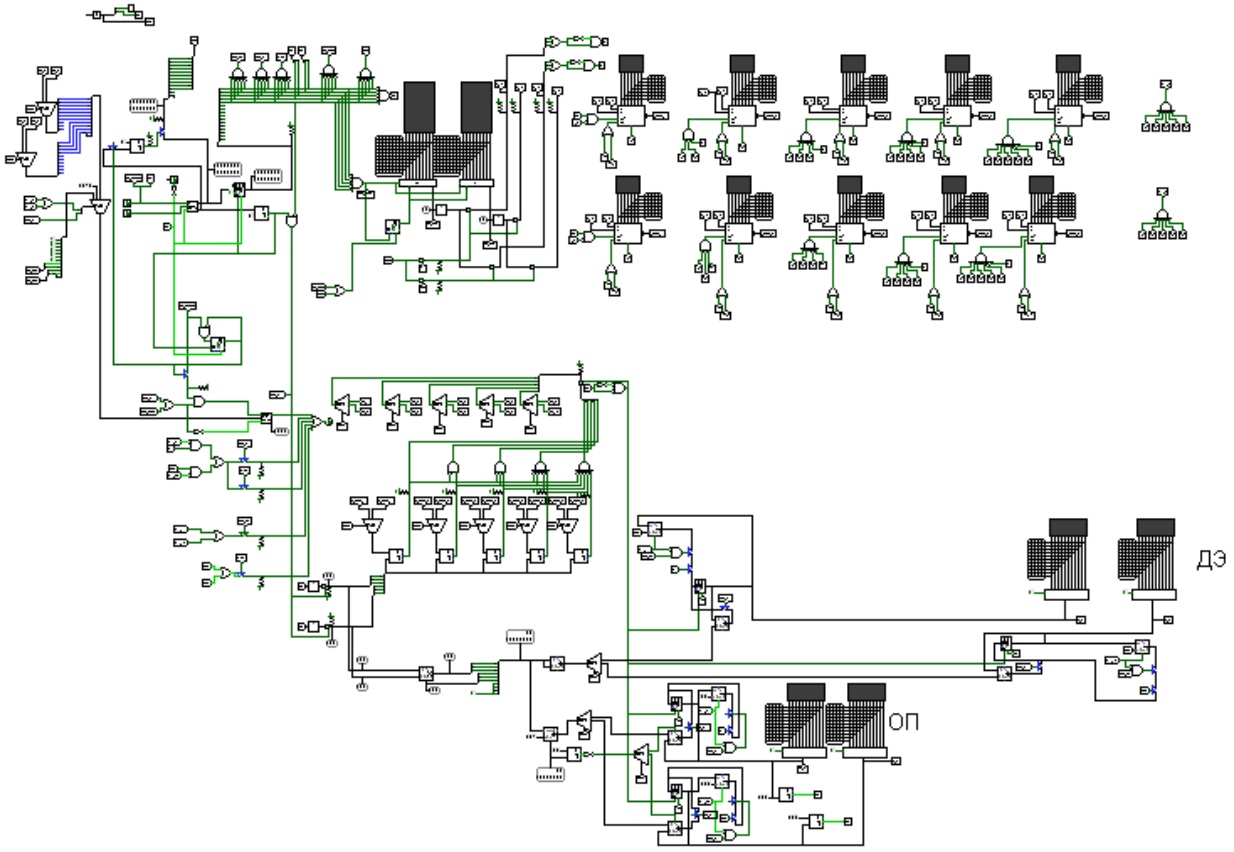


Скриншот 13: “Подсхема blue”



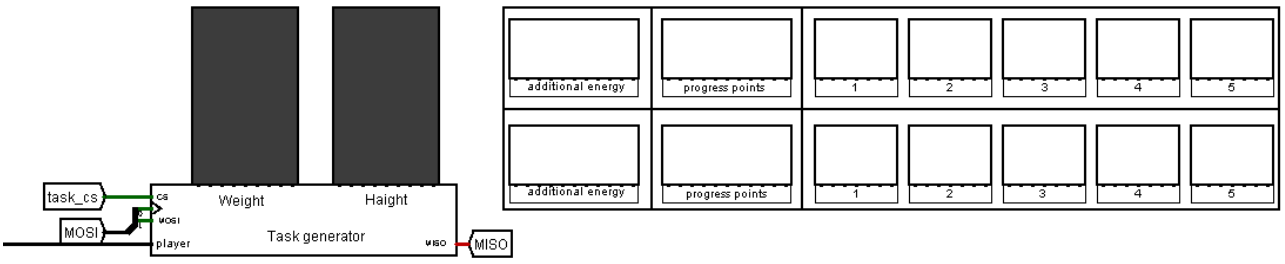
Скриншот 14: “Подсхема orange”

Выводят изображение фазы.

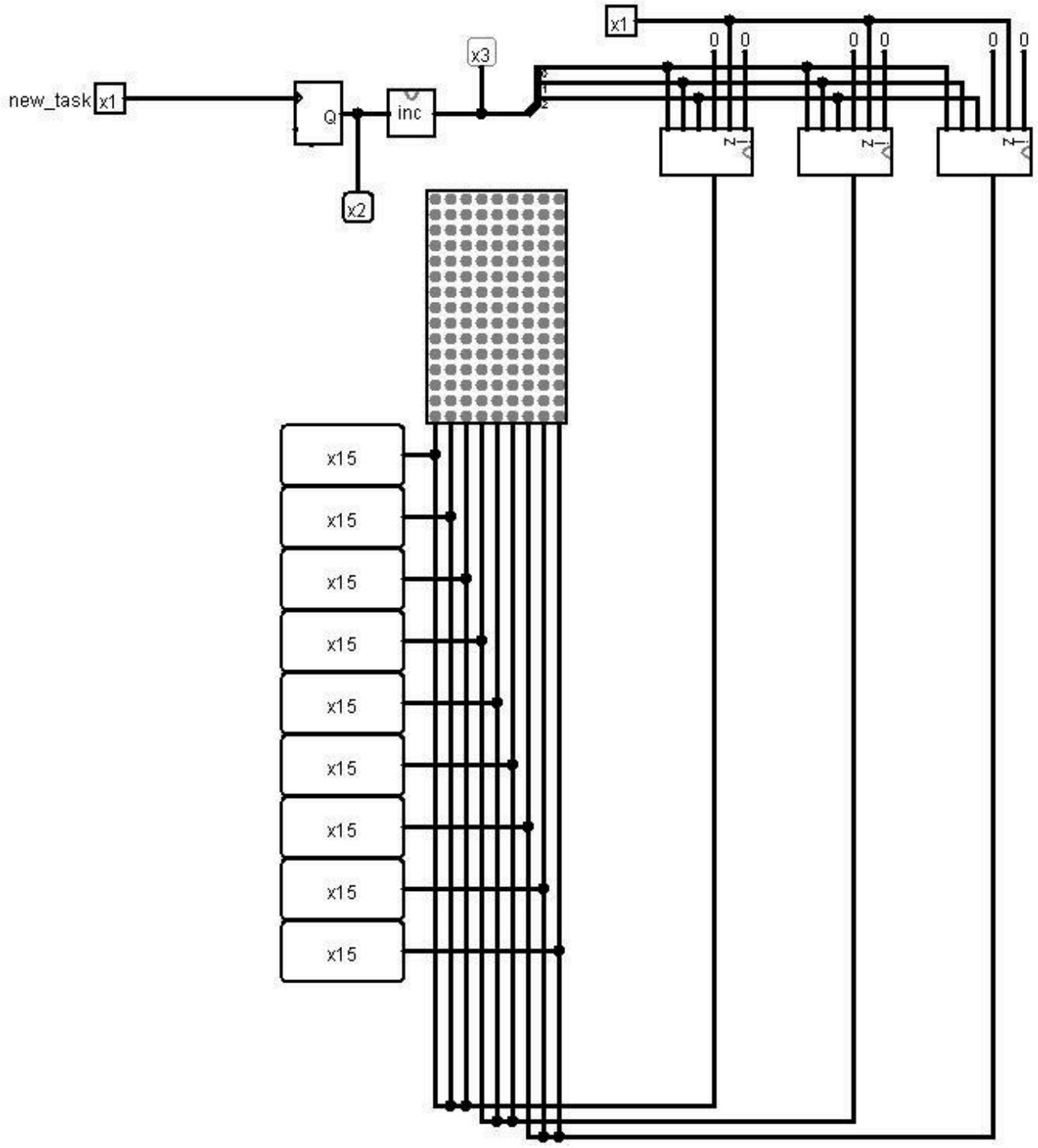


Скриншот 15: “Схема generator”

Схема для создания заданий и записи результатов в таблицу для каждого участника. Она рассчитывает очки прогресса и дополнительную энергию, а также сохраняет эти данные. В программе реализовано множество функций для отладки.

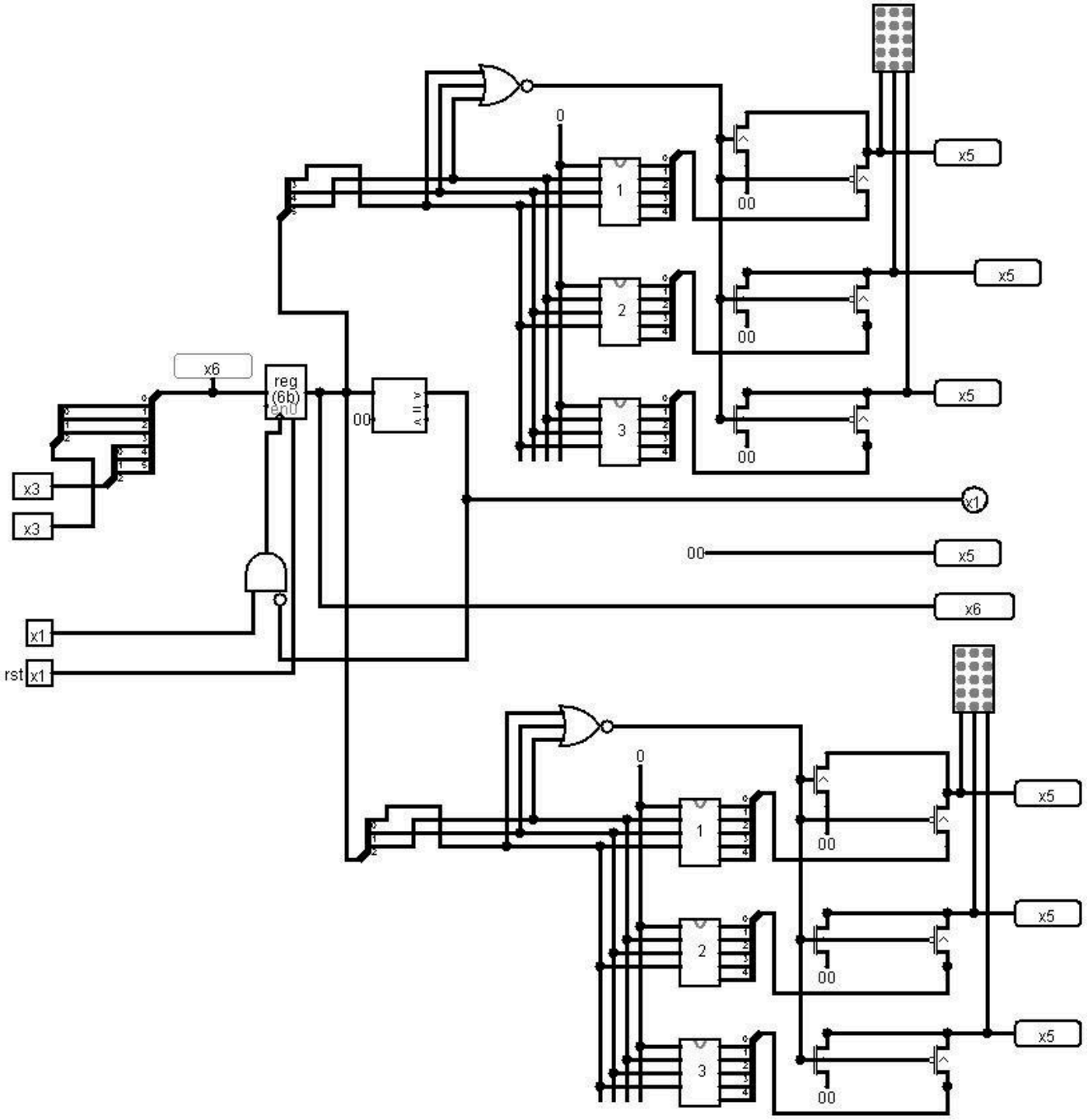


Скриншот 16: “Схема generator” для игрока



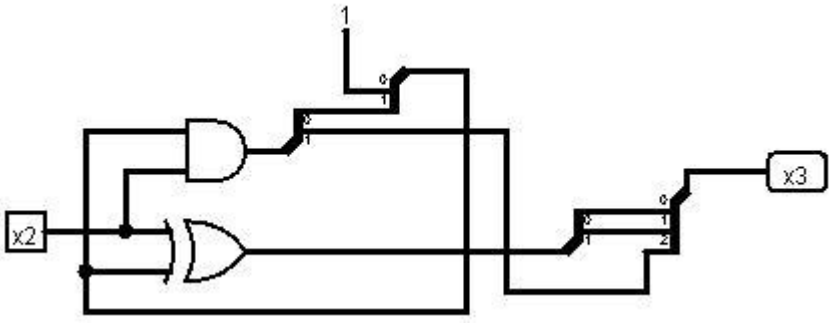
Скриншот 16: “Подсхема Weight и Haight

Схема для генерации заданий при помощи генератора случайных чисел.



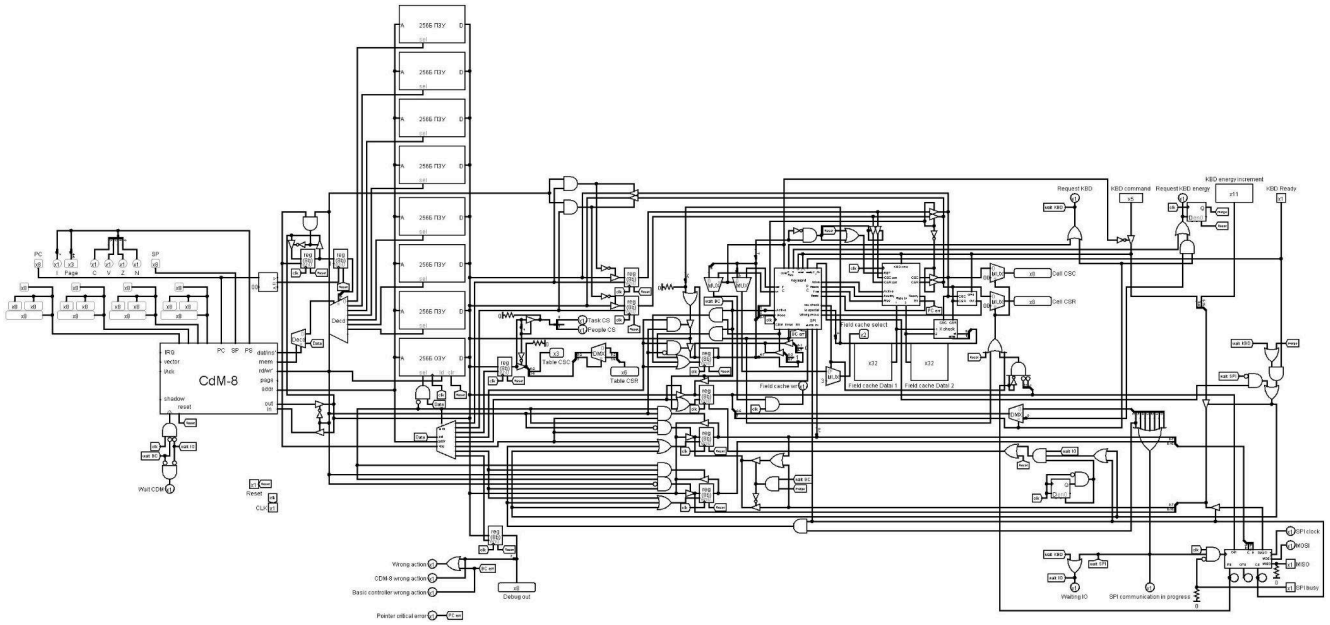
Скриншот 17: “Схема numder_2”

Вывод чисел для таблицы заданий. Клетка выводит 1 если занята, для логики записи.



Скриншот 18: “Подсхема inc”

8.6. MCU

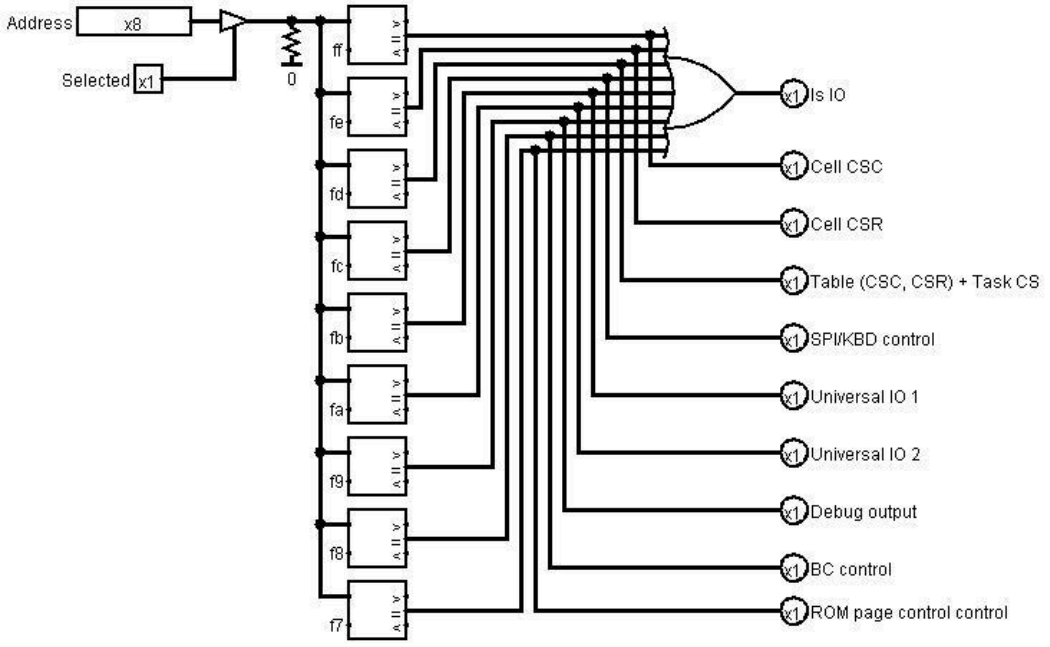


Скриншот 19: “Схема MCU v2”

Это самый главный блок проекта, он объединяет все вспомогательные контроллеры в игре и является, по сути, центральным узлом управления. Он состоит из:

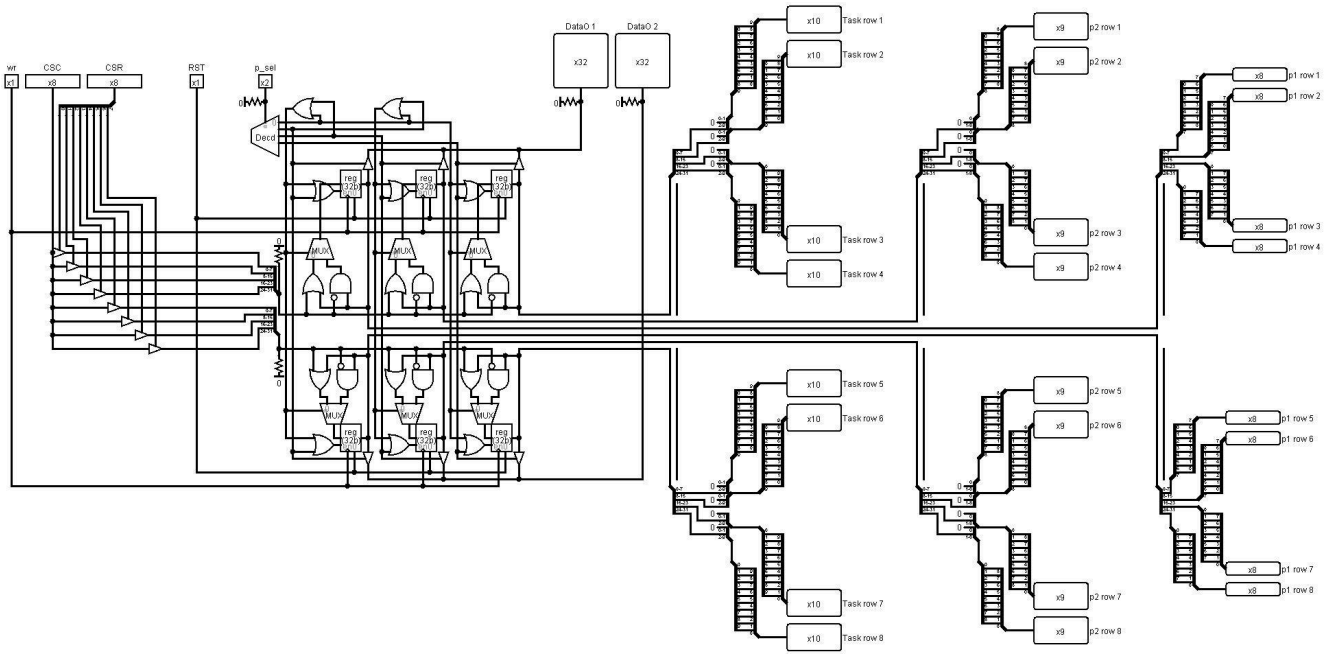
- Процессора CDM-8
- блоков ПЗУ и ОЗУ,
- I/O Регистров, с помощью которых процессор управляет внутренними и внешними устройствами
- Контроллера базовой логики - basic controller
- Контроеллера курсора
- Блока коммуникации SPI, связывающего контроллер с внешним миров

Так как программа для процессора получилась достаточно большой, мы использовали несколько блоков ПЗУ и специальный регистр, который будет осуществлять возможность программно переключать блоки памяти, из которых будет считываться программа.



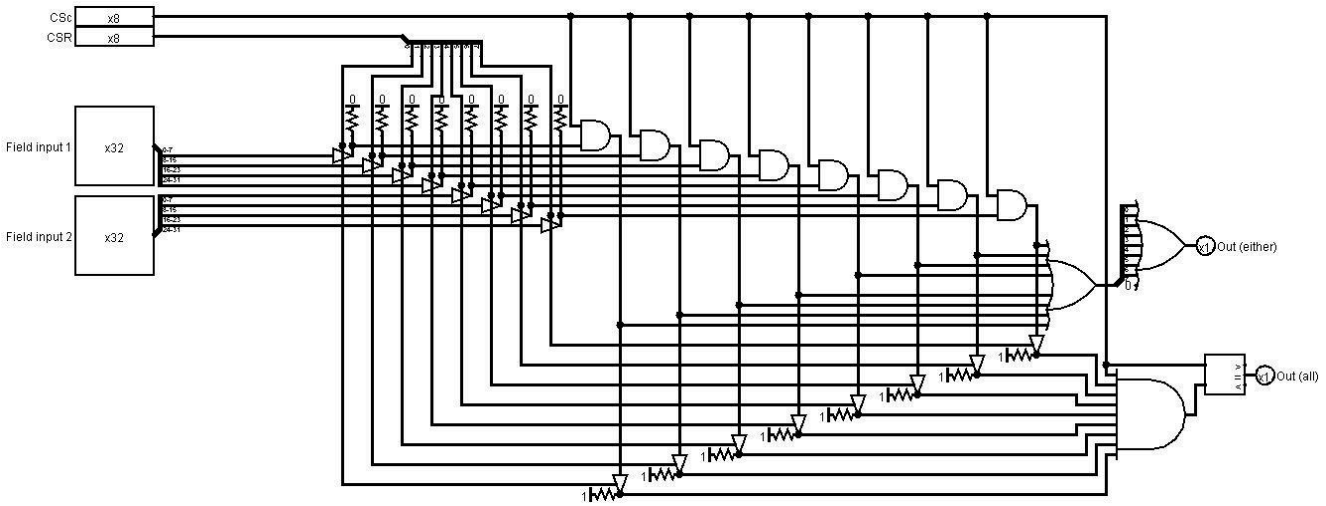
Скриншот 20: “Подсхема IO dmx”

Эта схема позволяет контроллеру выбирать I/O регистр, с которым он будет взаимодействовать.



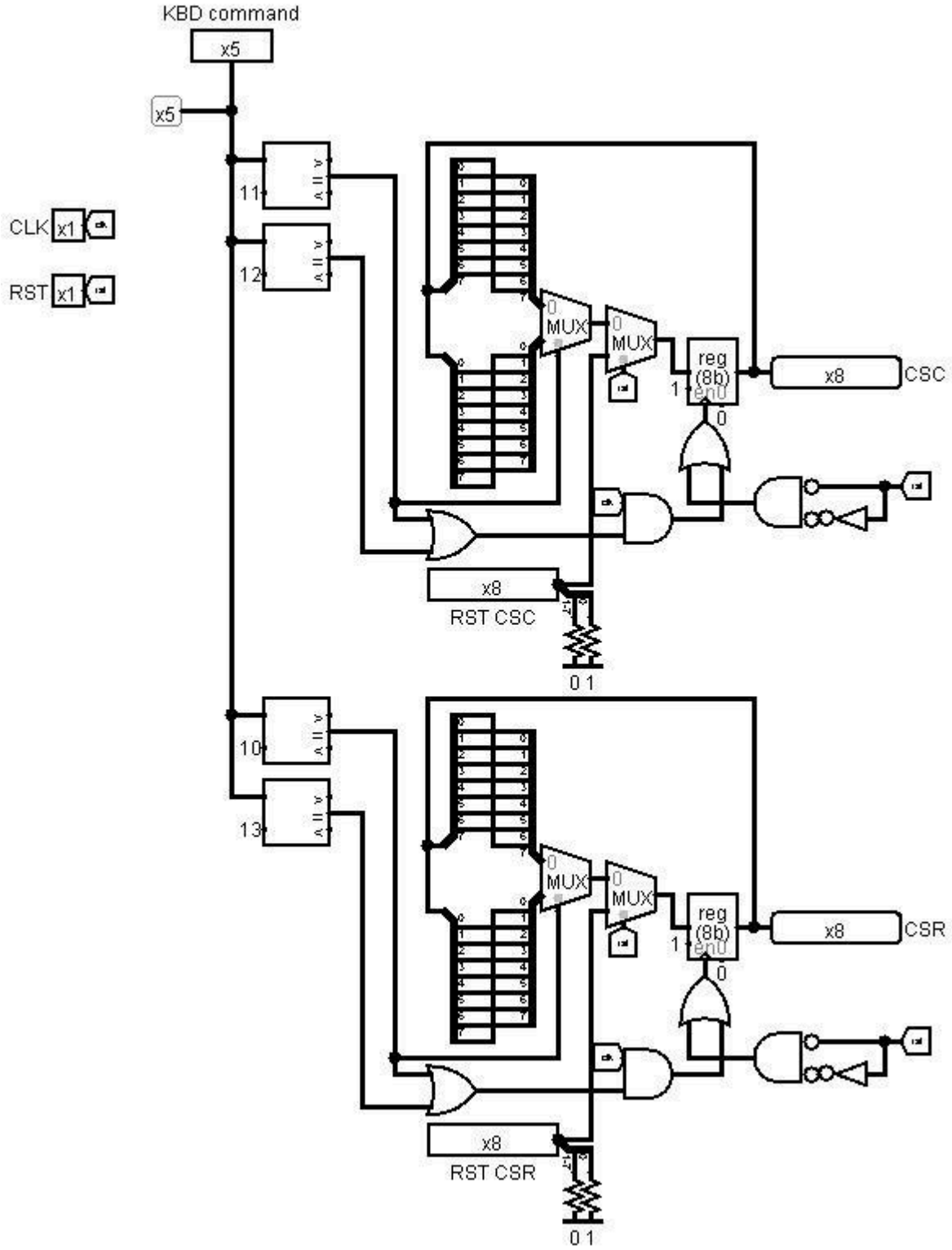
Скриншот 21: “Схема Field cahce”

Эта схема отвечает за хранения состояния поля. Этот блок нужен чтобы контроллеру не приходилось при проверки выполнения заданий или при перемещении курсора индивидуально обращаться к каждой клетке, считывая её принадлежность. Такой подход очень сильно ускоряет игру, т.к. уменьшает количество операций во время игры.



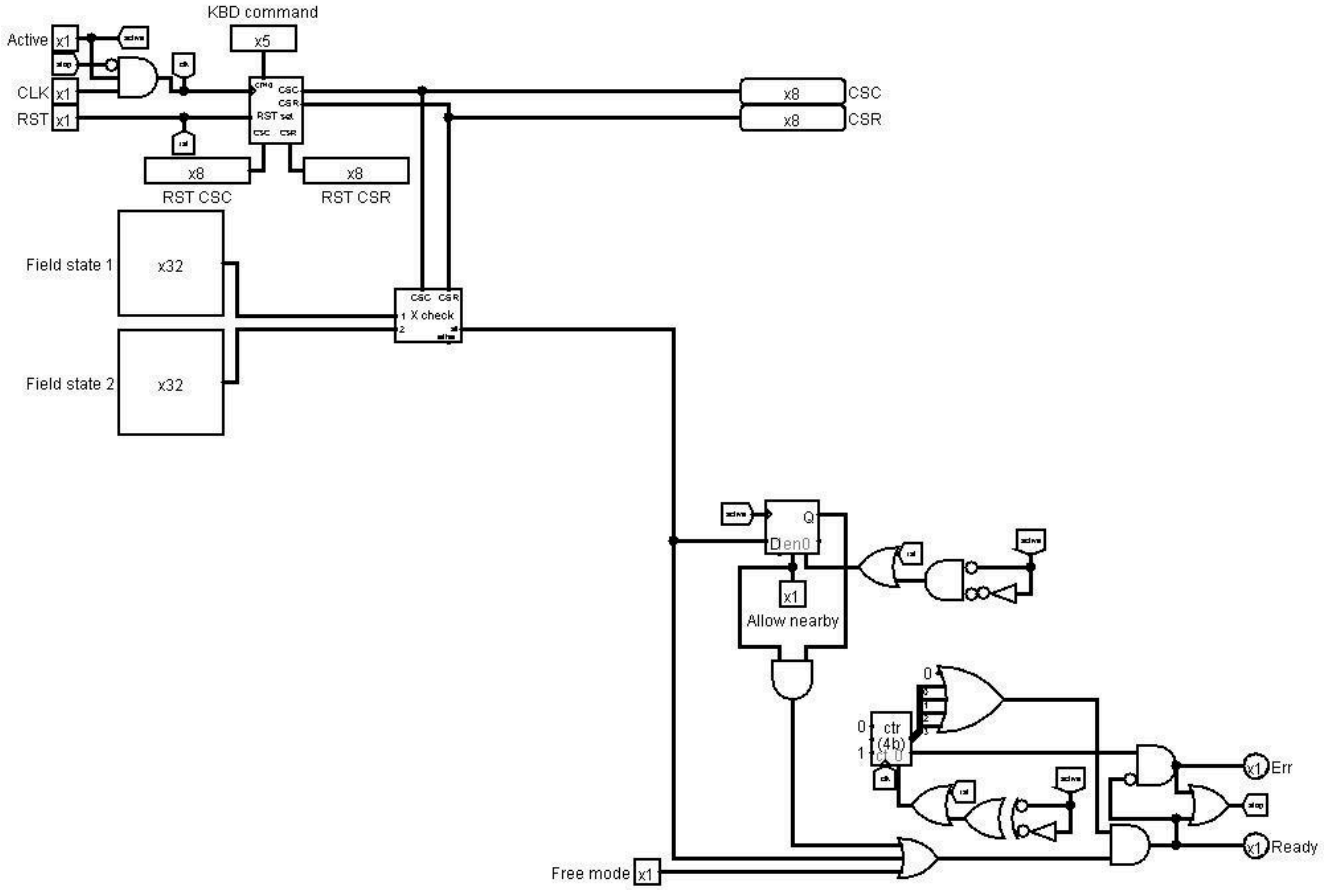
Скриншот 22: “Схема position checker”

Вспомогательная схема для проверки занятости поля, она получает на вход позицию (или позиции) проверяемой(ых) клетки(ок) и, опираясь на кэш поля проверяет принадлежит ли клетка(и) некоторому игроку



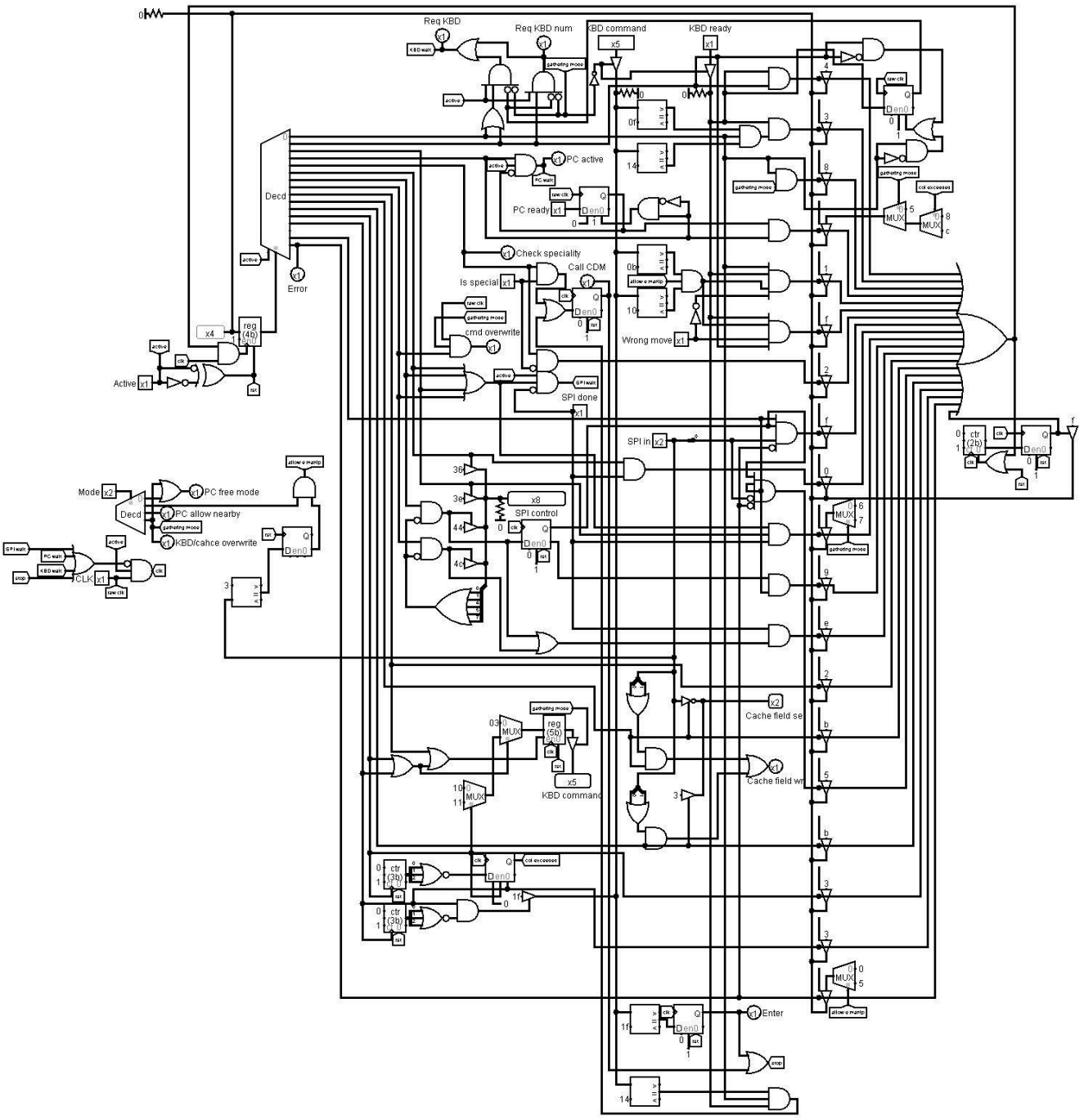
Скриншот 23: “Схема pointer mover”

Эта схема отвечает за перемещение курсора. Она оперирует побитовым сдвигом, направление которого зависит от клавиши, нажатой игроком.



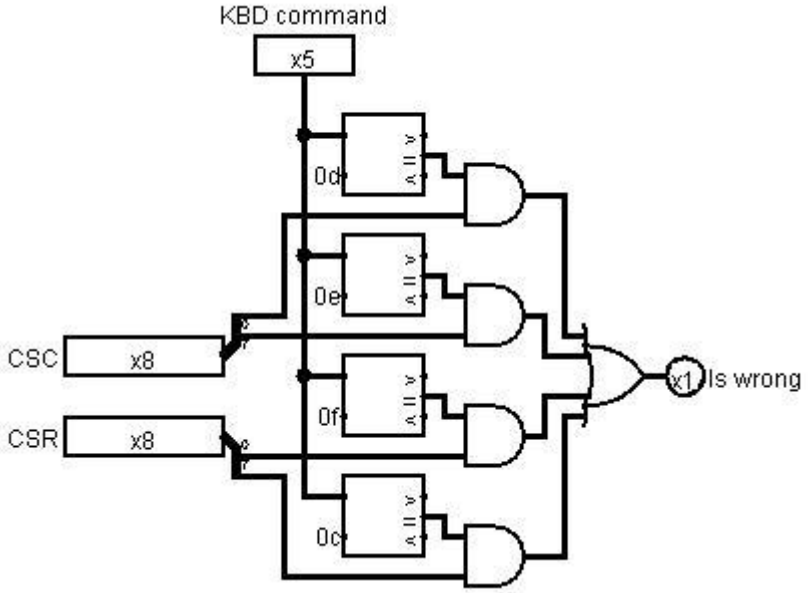
Скриншот 24: “Схема pointer controller”

Эта схема управляет положением курсора, не допуская ситуации когда курсор находится вне разрешённых положений. Разрешённые положения определяются контактами free mode и allow nearby. Если free mode активен, все положения являются разрешёнными. Если активен allow nearby, то разрешёнными считаются все положения соответствующие клеткам игрока, а также тем клеткам, которые прилегают к клеткам игрока, но не принадлежат ему. Если оба контакта неактивны, то курсор будет перемещаться строго в пределах поля игрока.



Скриншот 25: “Схема basic controller”

Эта схема обеспечивает обработку базовых команд. Например она позволяет выбрать какую-либо клетку управляя контроллером курсора и контроллером SPI (через эту шину данная микросхема сообщит клетке, что он выбрана). Также эта микросхема может обеспечивать цикл перемещения энергии между клетками и сбор информации о занятости клеток с последующим дублированием этой информации в кэше поля.

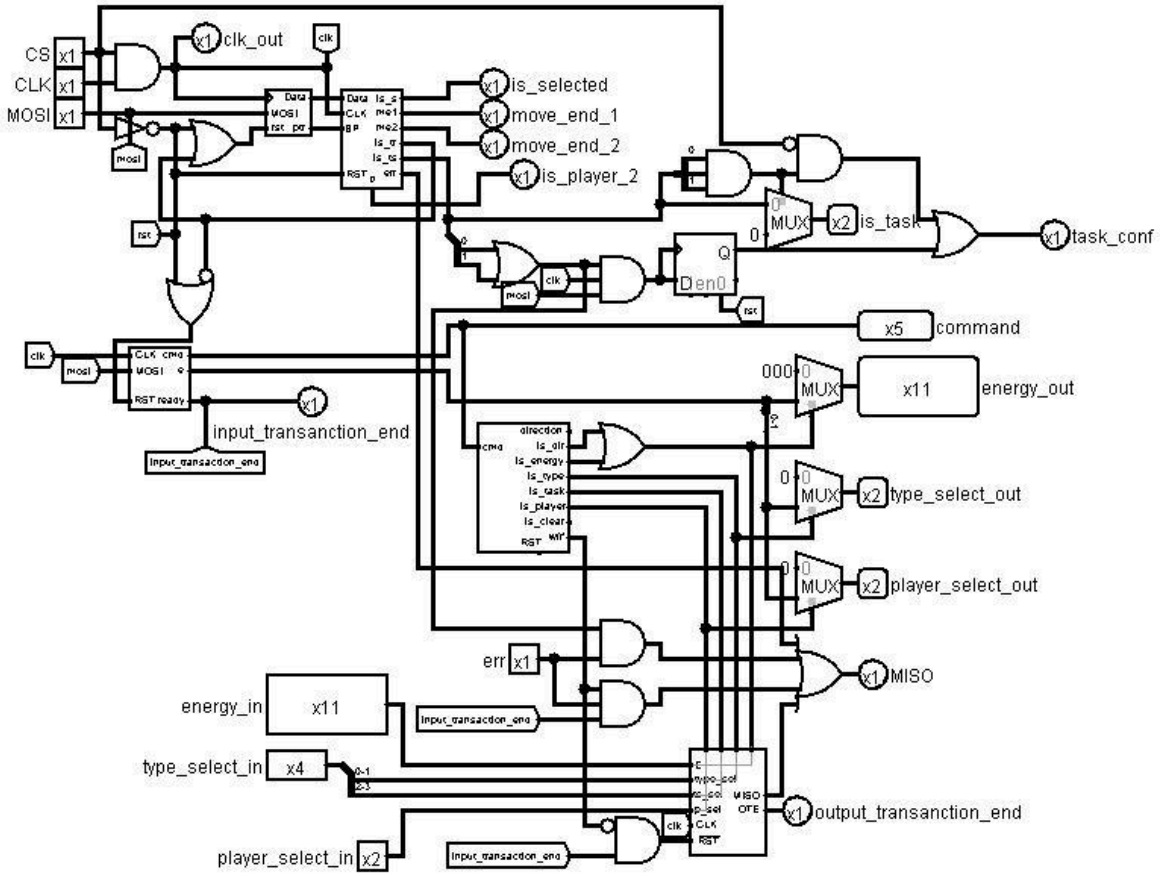


Скриншот 26: “Подсхема move checker”

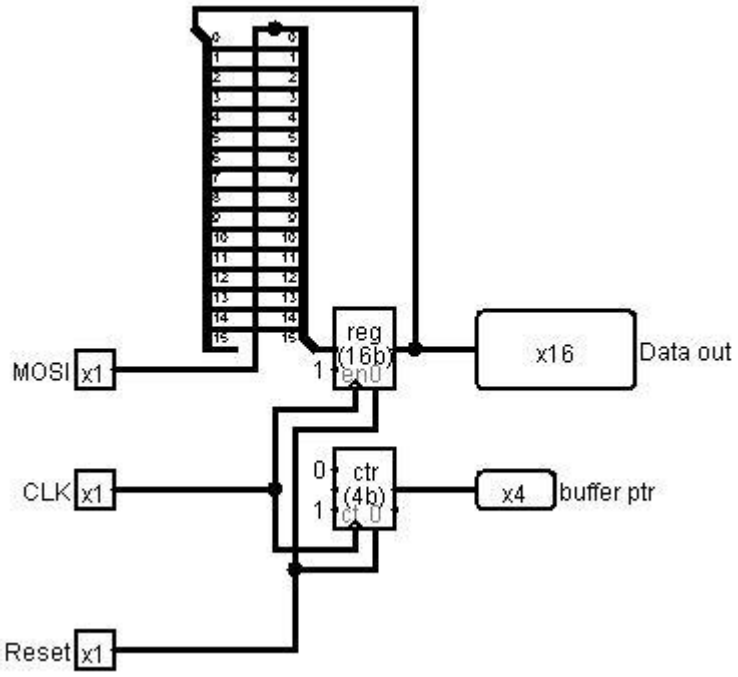
Схема, предотвращающая отправку энергии за границы поля.

8.7. SPI interface

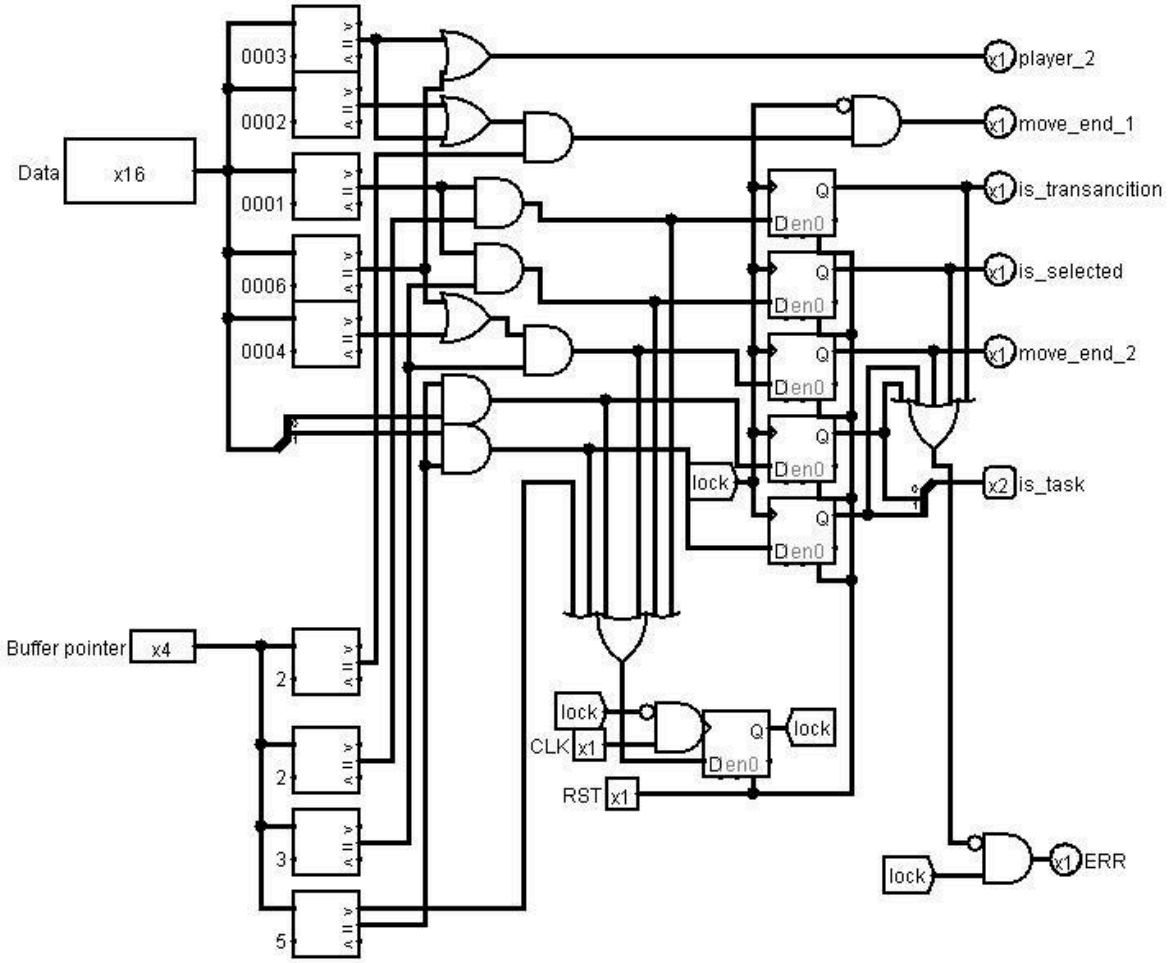
В данной схеме реализован последовательный последовательный периферийный интерфейс(SPI) для обеспечения связи между микроконтроллером и клетками



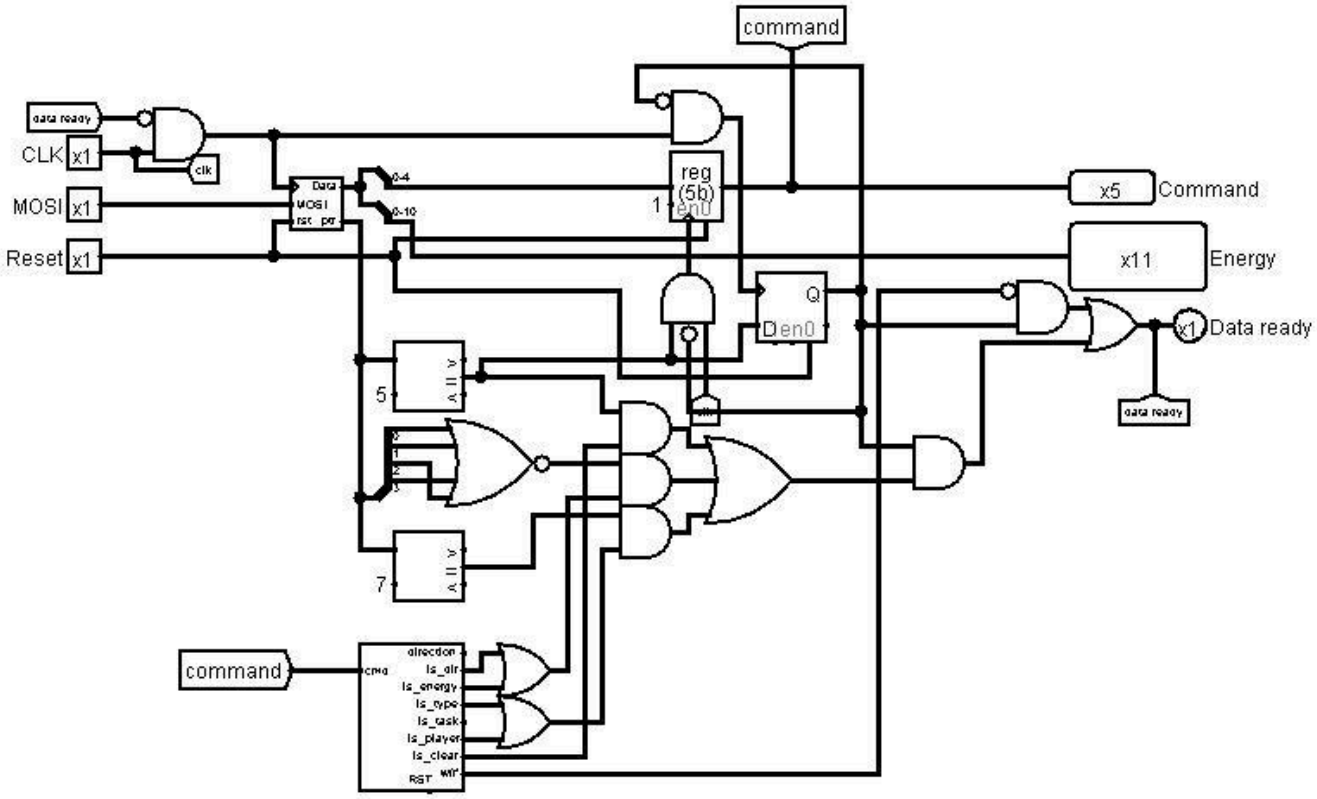
Скриншот 27: “Схема SPI module (cell)”



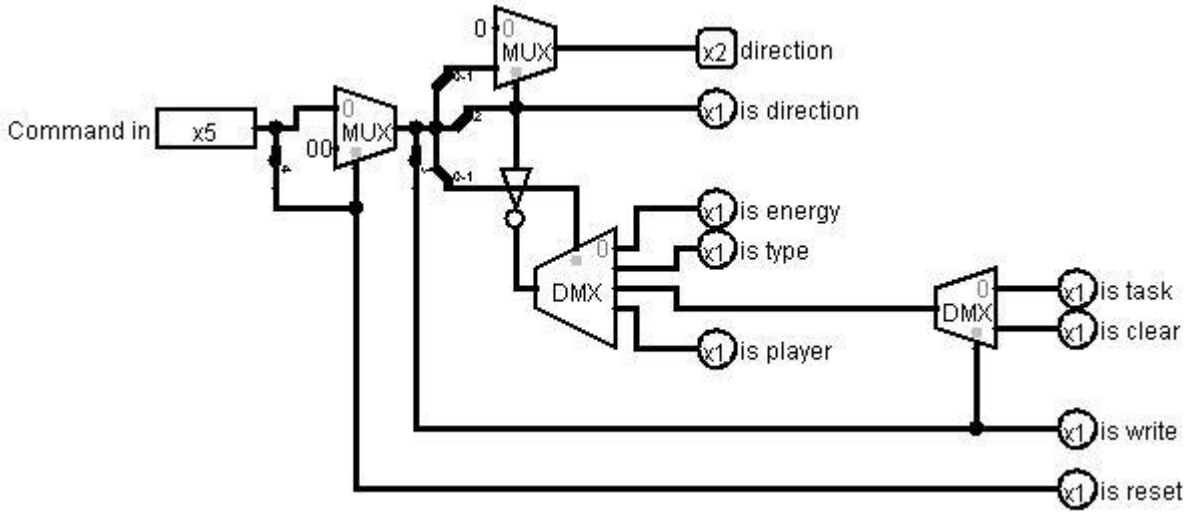
Скриншот 28: “Подсхема input shifter”



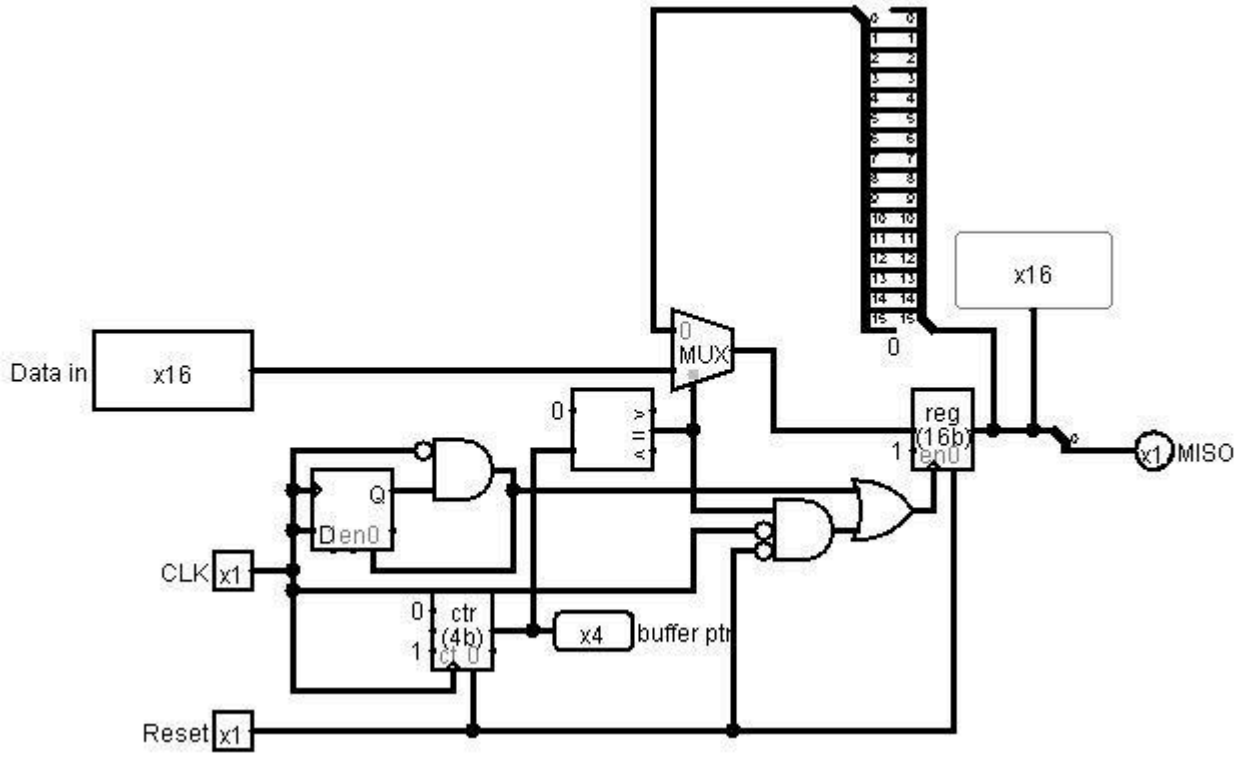
Скриншот 29: “Схема special sequence decoder”



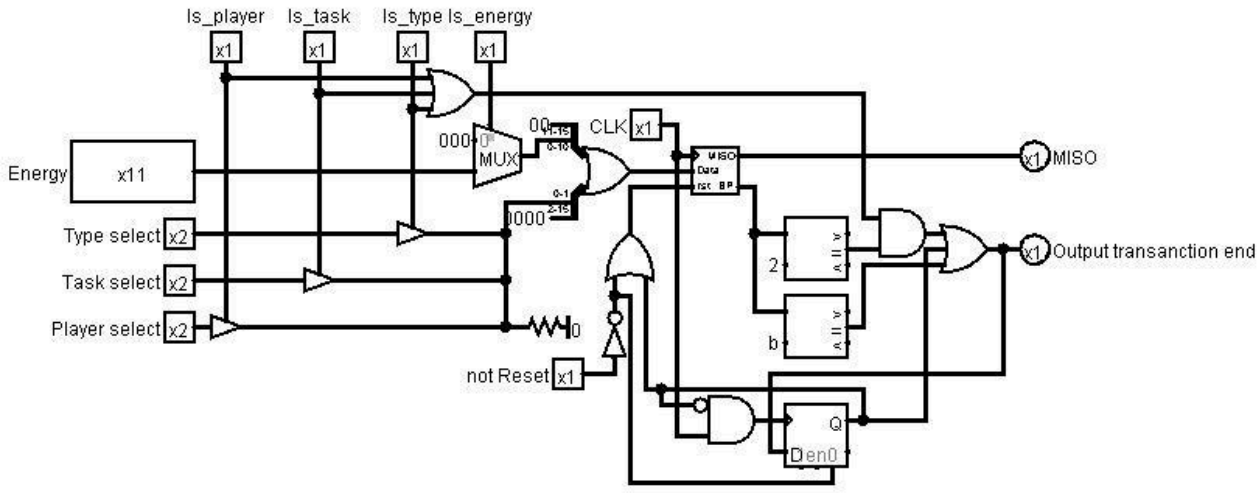
Скриншот 30: “Схема transaction parser”



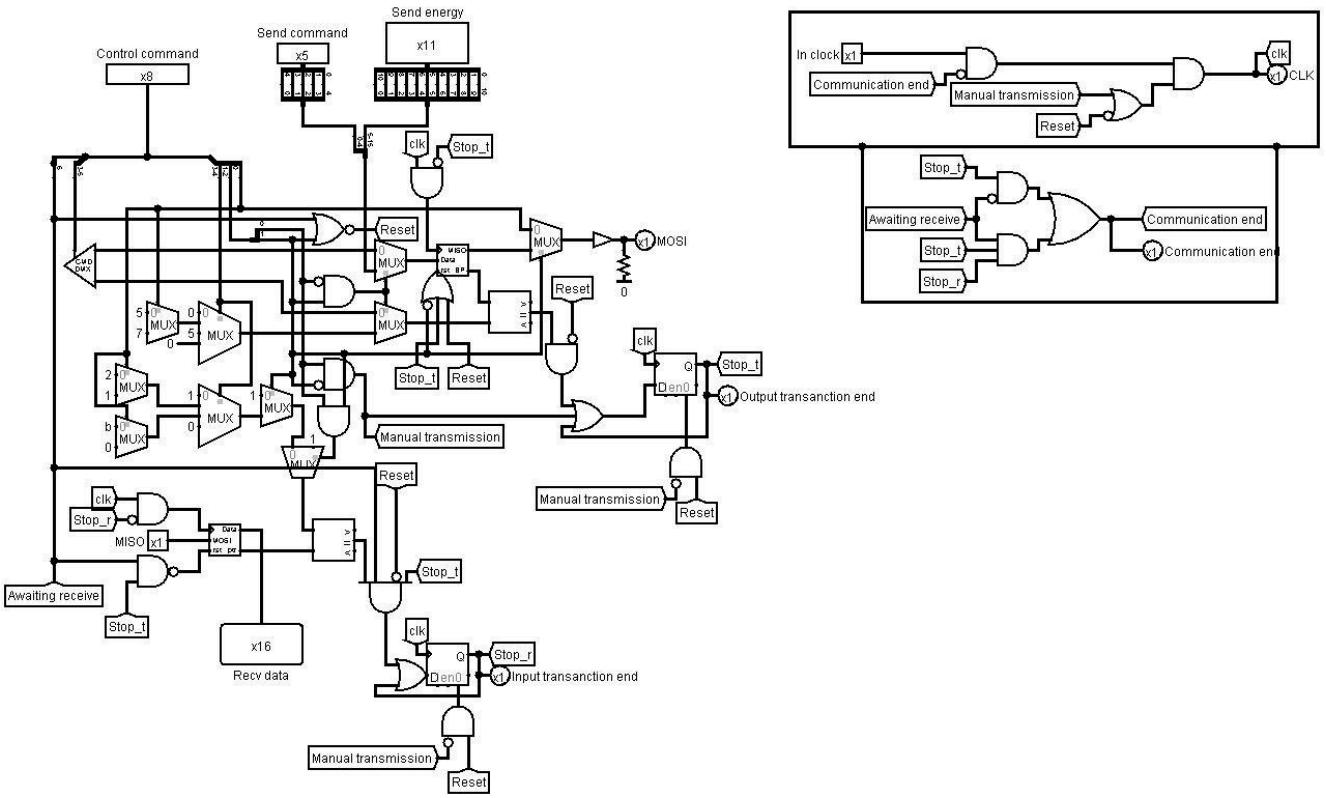
Скриншот 31: “Подсхема command decoder”



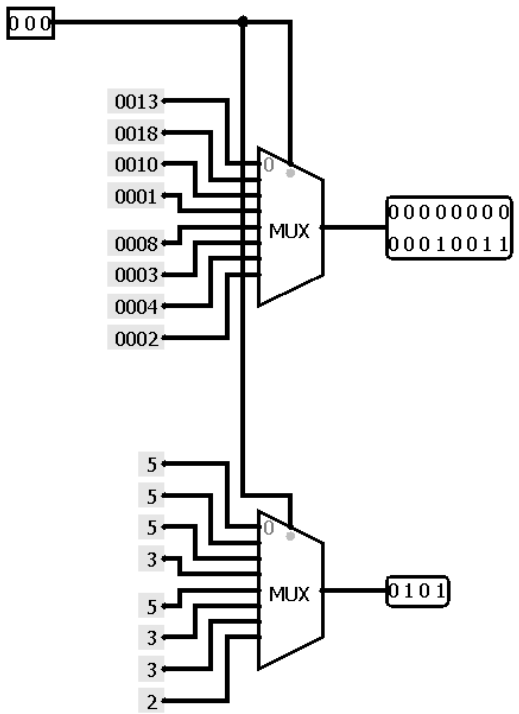
Скриншот 32: “Подсхема output shifter”



Скриншот 33: “Подсхема transaction sender”

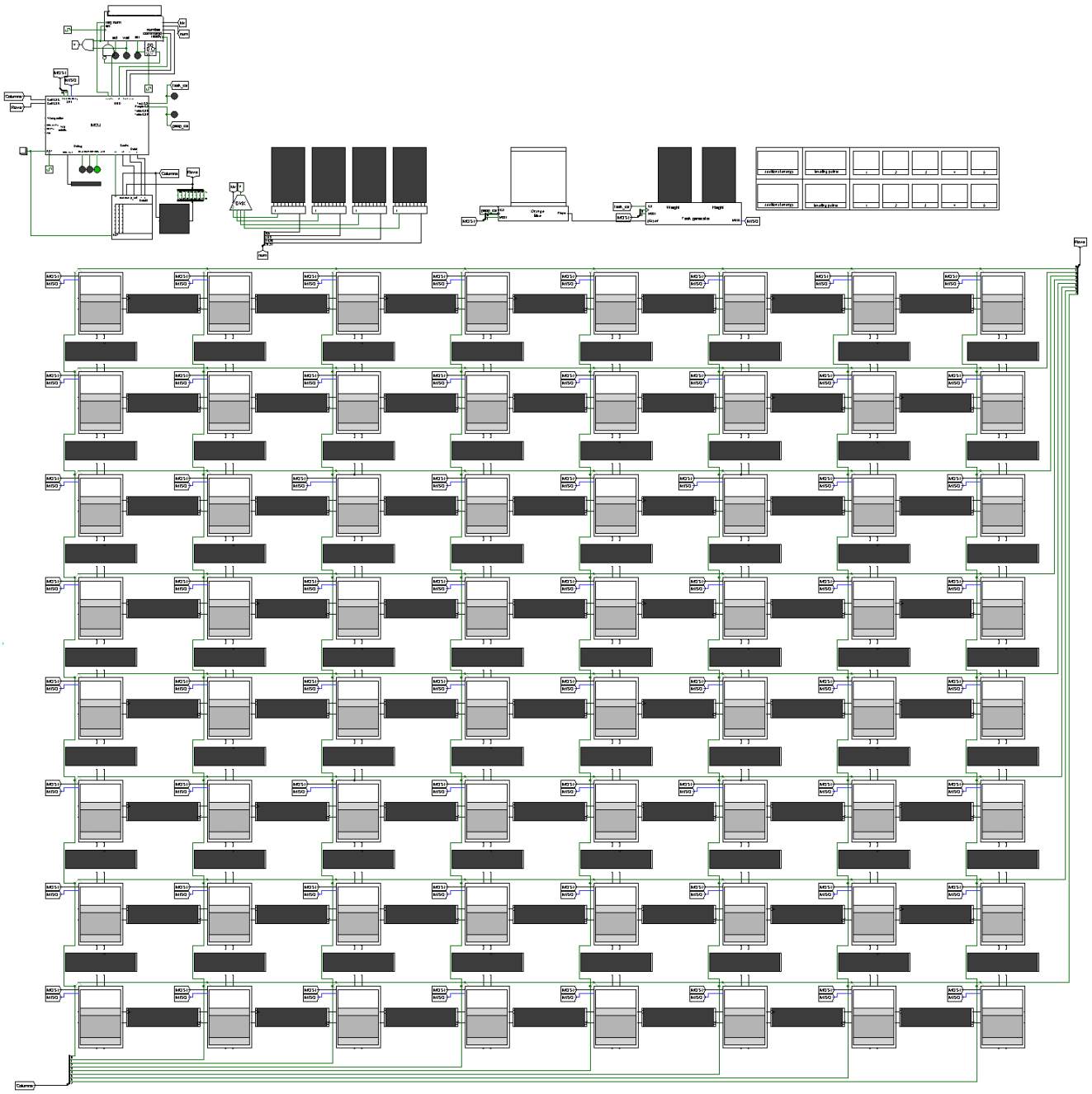


Скриншот 34: “Схема SPI module (master)”



Скриншот 35: “Подсхема predefined command selector”

8.8. Main



Скриншот 36: “Схема Main”

В данной схеме находится сама игра “Rectify”, здесь реализовано поле 8 x 8 клеток, а также взаимодействие игроков с полем

9 ЗАКЛЮЧЕНИЕ

В ходе проекта была создана электронная версия игры "Rectify" на микроконтроллере CdM-8 с использованием Logisim и ассемблера. Успешно реализованы: логическая схема игры, программное обеспечение, тестирование и документация. Проект демонстрирует практическое применение цифровой логики и низкоуровневого программирования, представляя ценность для образовательных целей. Все задачи выполнены, полученный опыт может быть использован в будущих разработках.

10 ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

В работе над проектом использовались материалы курса CS/MR Digital platforms 2024/25, размещенные в Google Classroom.

- tome.pdf