MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION
"NOVOSIBIRSK NATIONAL RESEARCH UNIVERSITY
STATE UNIVERSITY"
(NOVOSIBIRSK STATE UNIVERSITY, NSU)

15.03.06 - Mechatronics and Robotics
Focus (profile): Artificial intelligence

TERM PAPER

Authors
Калашников И.А., Мироненко П.А., Ильиных Т.В.

Job topic:

# "DOOR GAME"

Novosibirsk, 2025

# 1. Введение

Представляем уникальную адаптацию классической игры “Виселица”, реализованную в виде цифровой микросхемы в Logisim. Игроку предстоит помочь человечку пройти путь через серию дверей, за которыми спрятано загаданное слово из восьми букв, выбранное из словаря. Если игрок угадывает букву, входящую в слово, дверь открывается, и персонаж продвигается вперёд. Если же буква неверная, дверь остаётся закрытой, а игрок теряет одну из шести доступных жизней. Цель — открыть все двери, угадав слово, прежде чем закончатся жизни.

Игра "Виселица" (Hangman) — это классическая словесная головоломка, корни которой уходят в викторианскую Англию. Игрок пытается угадать загаданное слово, вводя буквы по одной. За каждую ошибку игрок теряет "жизнь" или приближается к проигрышу, что изображается в виде нарисованной виселицы. Цель игры — угадать слово до того, как закончатся попытки. Первые упоминания о ней датируются концом XIX века, когда она появилась как бумажная игра. Официально игра была описана в книгах по развлечениям в начале XX века, а её популярность выросла в 1970-х годах.

## 2. Аналоги

Игра «DOOR GAME», реализованная в Logisim как аппаратная адаптация «Виселицы», не имеет прямых аналогов. Однако существуют игры и проекты с частично схожими механиками, тематикой или технической реализацией, включая вариации популярных словесных игр Wordle и Scrabble.

1. **Wordle и его вариации (например, Quordle, Scrab Wordle)**
   - **Описание**: Wordle — онлайн-игра, где игрок угадывает слово из 5 букв за 6 попыток с цветными подсказками. Вариации, такие как Quordle (угадывание 4 слов одновременно) и Scrab Wordle (с учётом очков Scrabble), добавляют новые элементы.
   - **Сходства**: Как и DOOR GAME, фокусируется на угадывании слова с ограничением попыток и визуальным прогрессом (матрицы в DOOR GAME, цветовые клетки в Wordle).
   - **Различия**: Wordle и его вариации — программные игры, а не аппаратные. В DOOR GAME слово из 8 букв, а механика «дверей» уникальна.
2. **Scrabble и его вариации (например, Bananagrams, Upwords)**
   - **Описание**: Scrabble — настольная игра, где игроки составляют слова из букв на поле для получения очков. Вариации, такие как Bananagrams (быстрое создание слов без поля) и Upwords (переписывание букв), предлагают альтернативные механики.
   - **Сходства**: Работа с буквами и словами, как в DOOR GAME. Цифровые версии (например, Words With Friends) используют ввод букв.
   - **Различия**: Scrabble и вариации фокусируются на составлении слов, а не угадывании. Аппаратная реализация отсутствует, а попытки не ограничены.


рис. 1 Wordle и Scrabble

# 3. Аппаратная часть

## 3.1 - Внешняя схема

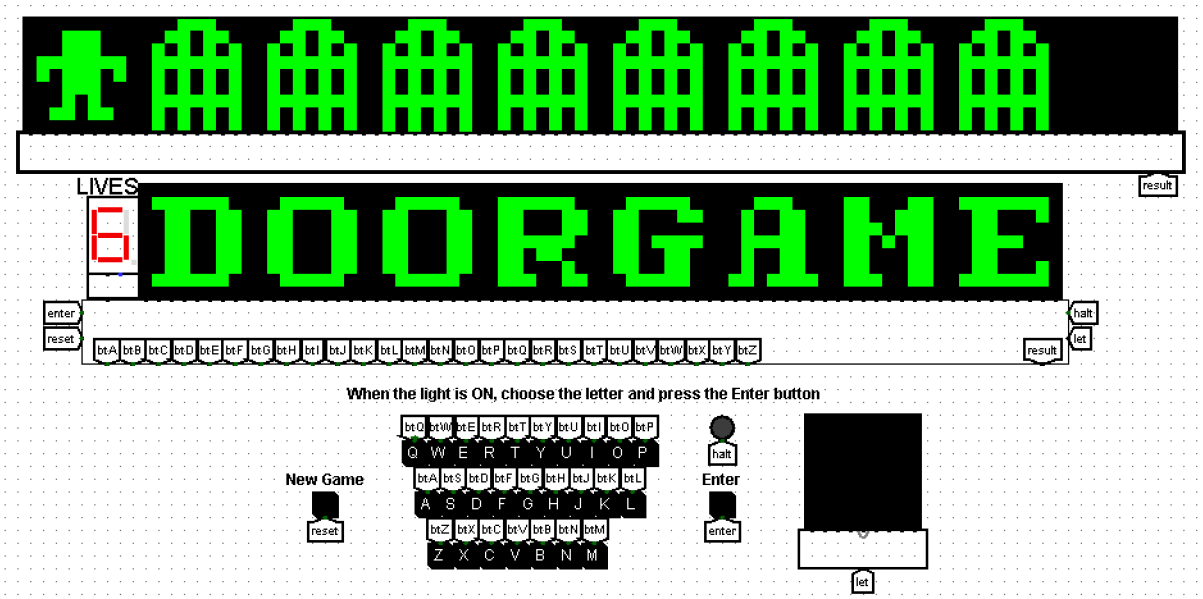



рис. 2 Внешний вид проекта

В проекте используется 18 матриц размером 9x9 - 10 в верхнем ряду и 8 в нижнем.

Подробнее об использовании проекта рассказано в разделе “Руководство Пользователя”, здесь же будет представлен список модулей на схеме main с кратким описанием. Список приведён так, чтобы можно было рассматривать схему сверху вниз, однако подробные описания будут приведены в порядке, не совпадающем со списком, по мере увеличения сложности и важности подсхем.

### 3.1.1 - Контроллер анимаций

Располагается непосредственно под верхним рядом матриц и отвечает за анимацию движения персонажа. Будет подробно рассмотрен далее в разделе “3.3 - Animation”.

### 3.1.2 - Схема Backend

Располагается под нижним рядом матриц и выполняет целый ряд функций - отвечает за работу процессора, взаимодействие с ним пользователя, хранение и передачу слов процессору, взаимодействие с нижним рядом матриц, просчёт и отображение жизней, определение состояния игры. Является основным блоком, контролирующим ход игры и будет рассмотрен наиболее подробно далее.

### 3.1.3 - Блок отображения жизней

Располагается слева от нижнего ряда матриц и состоит непосредственно из дисплея и инвертора значения жизней между дисплеем и модулем Backend. Дисплей выводит количество жизней, а инвертор инвертирует порядок их счёта - получает из модуля Backend количество ошибок и передаёт на дисплей количество оставшихся жизней. Подробнее о подсчёте жизней рассказывается в разделе “3.2.5 - Блок контроля игры”.

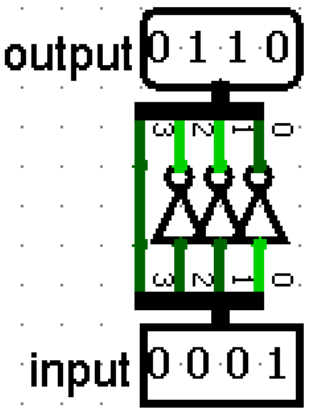



рис. 3 Схема инвертора количества жизней

### 3.1.4 - Клавиатура

Расположена под нижним рядом матриц и представляет собой английскую раскладку QWERTY. Позволяет игроку осуществлять ввод буквы. Подробно принцип действия разобран в разделе “3.2.3 - Клавиатура”.

### 3.1.5 - Дополнительные кнопки и светодиод

Кнопка “New Game” расположена слева от клавиатуры. При нажатии сбрасывает все модули к начальному положению и выбирает из словаря новое слово.

Кнопка “Enter” вместе со светодиодом расположена справа от клавиатуры. Кнопка подтверждает ввод буквы пользователем и запускает на процессоре цикл проверки этой буквы, а светодиод указывает состояние процессора. Если светодиод горит, значит процессор завершил выполнение программы и готов ко вводу новой буквы.

### 3.1.6 - Матрица выбора буквы

Расположена в правом нижнем углу и отображает текущую букву, выбранную пользователем. Состоит из матрицы и упрощённого контроллера матрицы (подробнее о нём в разделе “3.2.4.3 - Переключатель положения матрицы”).

### 3.2 - Backend

Основной аппаратный модуль, отвечающий за работу процессора, взаимодействие с ним пользователя, хранение и передачу слов процессору, взаимодействие с нижним рядом матриц, просчёт и отображение жизней, определение состояния игры.

#### 3.2.1 - Блок словаря

Этот блок отвечает за хранение и выбор слов и последовательную передачу букв в ОЗУ процессора и на модули взаимодействия с матрицами.

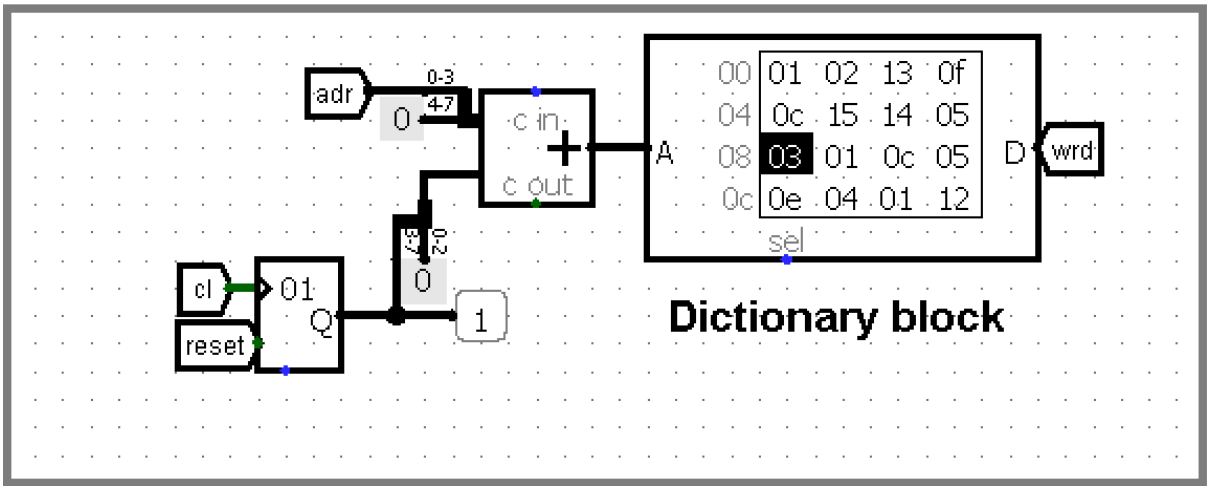
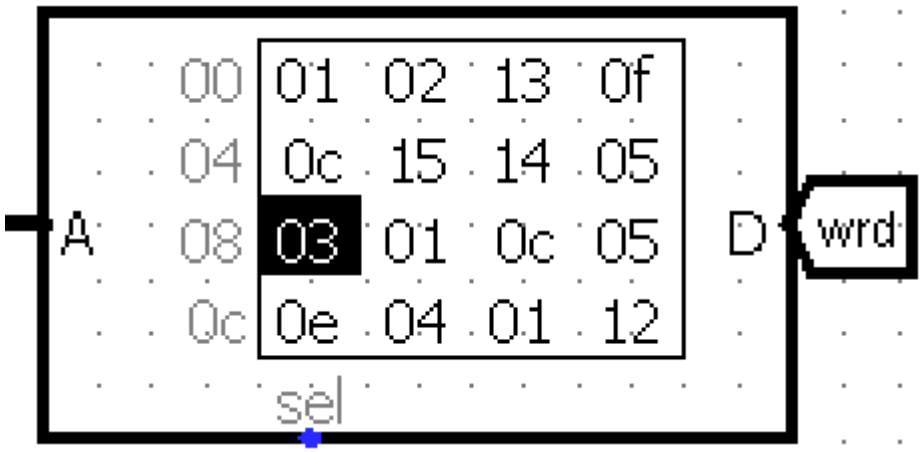



рис. 4 Блок словаря




Словарь - ПЗУ, хранящее 32 слова из 8 букв. Каждая буква представляет собой число от 1 до 26 в шестнадцатеричной кодировке, где 1 - буква A, 26 - буква Z.

рис. 5 Словарь

Рандомайзер  выдаёт указатель на начало случайного слова из словаря. Сбрасывается в начале новой игры.

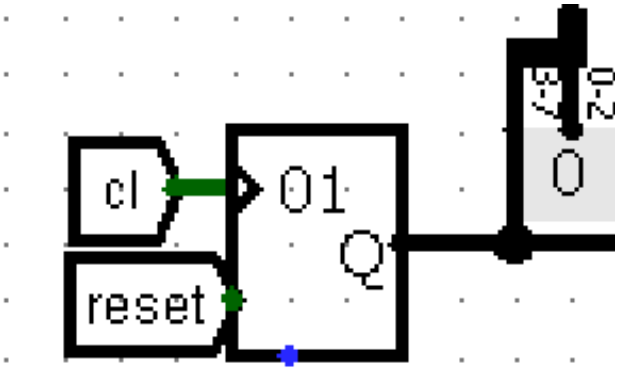



рис. 6 Рандомайзер

### 3.2.2 - Адресный и тактовый блоки

Тактовый генератор перекрывается при завершении игры одним из способов (победа или поражение). Остаток блока отвечает за контроль загрузки слова в ОЗУ процессора и блок взаимодействия с матрицами. Сигнал ldf означает что загрузка слова завершена и игра начата. Он же завершает взаимодействие блока с блоком контроля матриц.

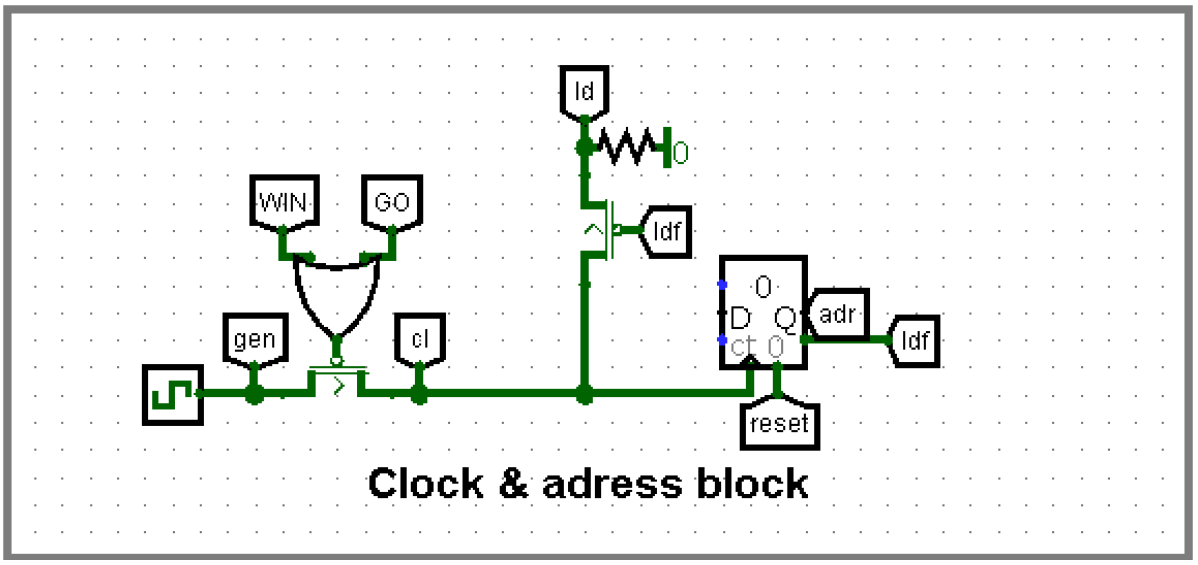



рис. 7 Адресный и тактовый блоки

### 3.2.3 - Клавиатура

В проекте реализована клавиатура. (Подробнее о взаимодействии с ней см. в Руководстве пользователя, раздел Клавиатура и ввод) Для каждой буквы существует отдельная кнопка, при нажатии на которую через туннель передаётся сигнал.

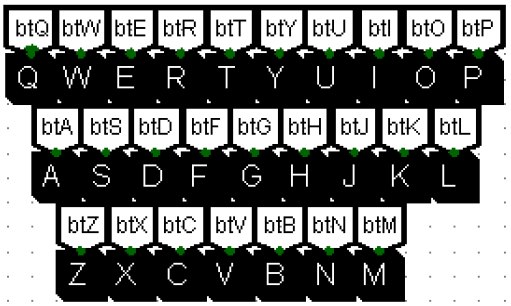



рис. 8 Клавиатура

Для каждой буквы реализована следующая схема:

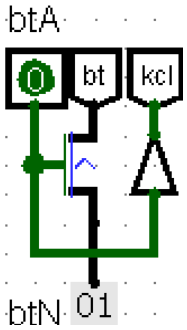



рис. 9 Кнопка клавиатуры

При подаче сигнала транзистор n-типа открывается и через туннель bt передаётся номер буквы. В это время через kcl передаётся сигнал для регистра в контроллере клавиатуры. Он запоминает букву и передаёт её через туннель let на RAM data controller (в ОЗУ процессора).

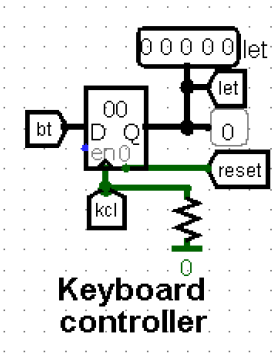



рис. 10 Контроллер клавиатуры

### 3.2.4 - Блок взаимодействия с матрицами

Отдельные модули для каждой из 8 матриц расположены попарно, выходящие проводы расположены “ёлочкой” для экономии пространства на схеме. Блок является связующим звеном между матрицами и прочими модулями (словарь, процессор)

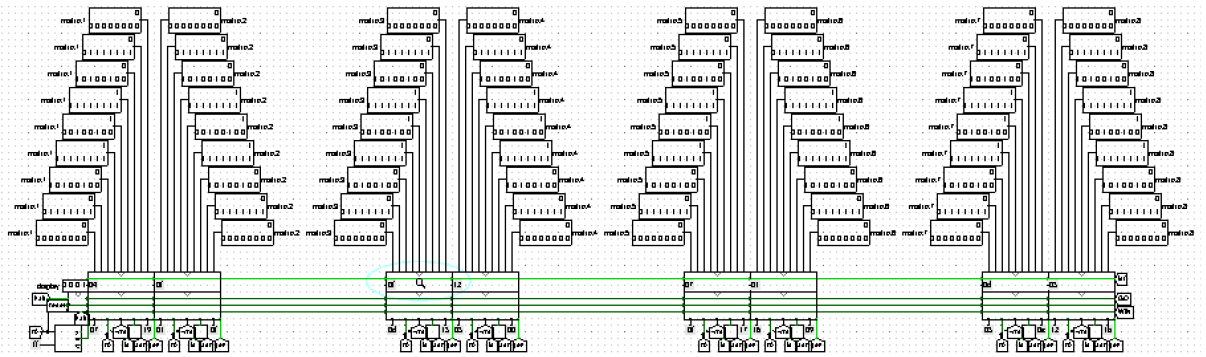


рис. 11 Блок взаимодействия с матрицами

Основные модули блока  - переключатели, дополнительные модули к переключателям, адресные контроллеры тактов.

### 3.2.4.1 - Адресные контроллеры тактов

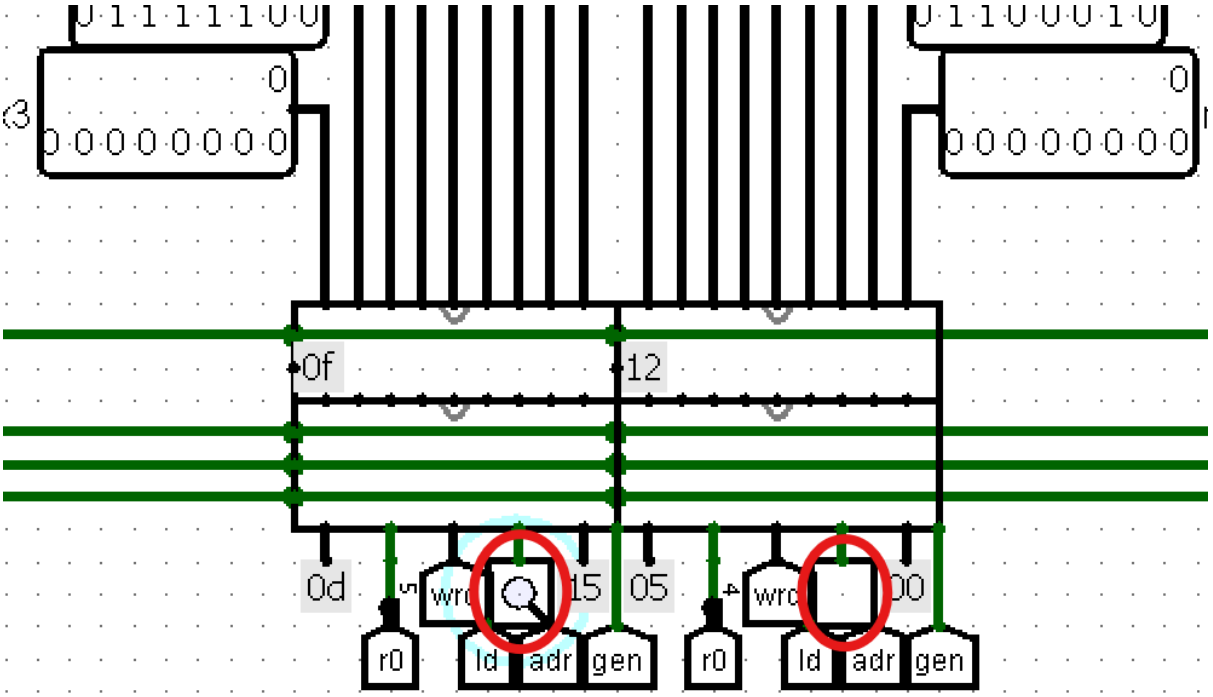
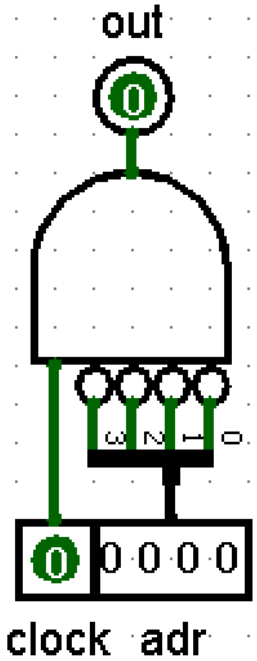



рис. 12 Адресные контроллеры тактов на схеме Backend




Располагаются перед одним из входов каждого переключателя матриц.

Необходимы чтобы вычленить из последовательности букв ту, которая должна быть привязана к определённой матрице. При соответствии адреса (от 0 до 7 по номеру буквы) такт пропускается в модуль переключателя, где используется для обновления значения регистра (подробнее в 3.2.4.3 переключатели матриц).

рис. 13 Адресный контроллер тактов для адреса 0

### 3.2.4.2 - Дополнительный модуль к переключателям

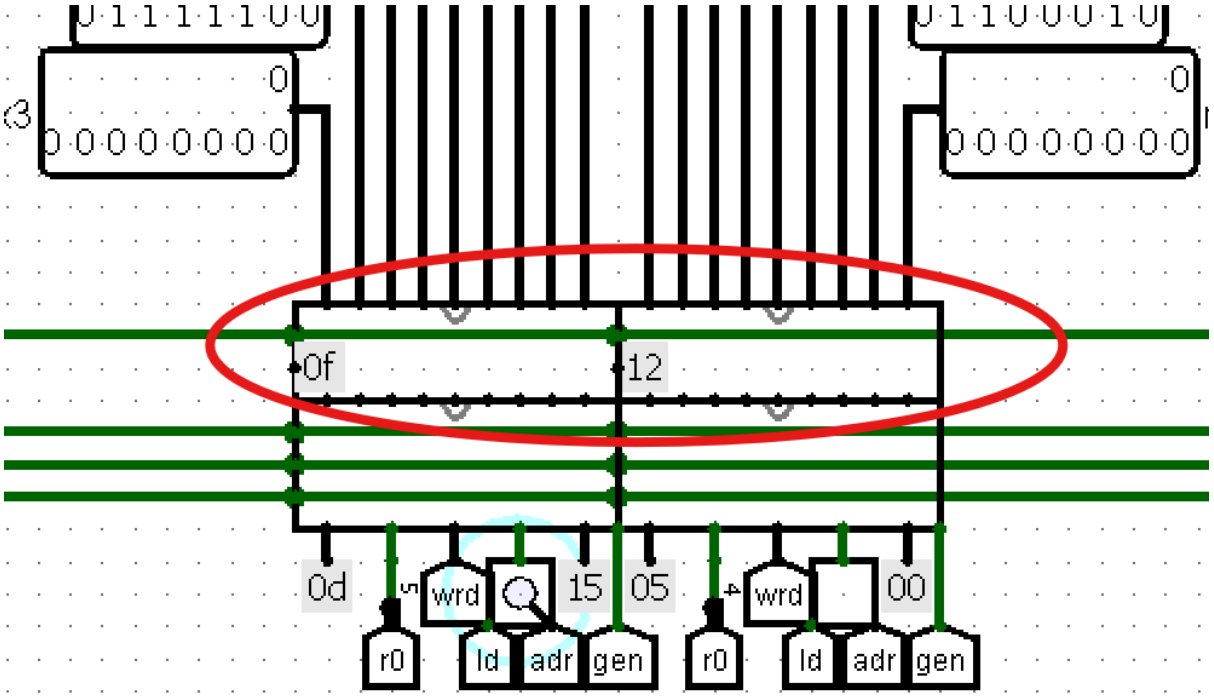



рис. 14 Дополнительные модули к переключателям на схеме Backend

Располагается на выходах каждого из 8 переключателей (сверху). Необходим для разгрузки и без того сложной схемы переключателя. Выполняет роль “заглушки” перед началом игры и отображает букву, подаваемую на вход до окончания загрузки слова в ПЗУ (см. раздел 3.2.6 - Процессор и память). В стандартной реализации в совокупности выводят на матрицы слова “DOORGAME” однако при необходимости надпись может быть заменена на любую другую посредством констант.

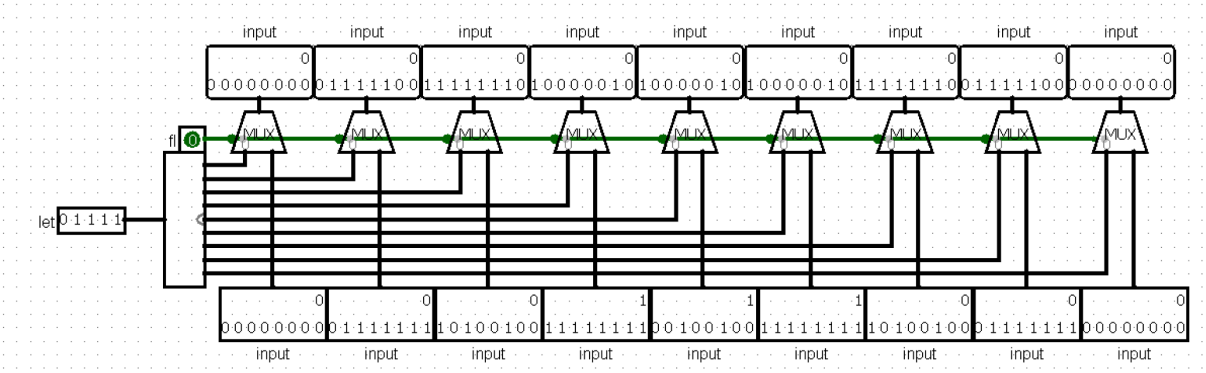



рис. 15 Схема дополнительного модуля

В схеме модуля используется упрощённая версия переключателя, которая получает на вход номер буквы и выводит 9 констант для рисунка буква на матрице.

### 3.2.4.3 - Переключатель положения матрицы

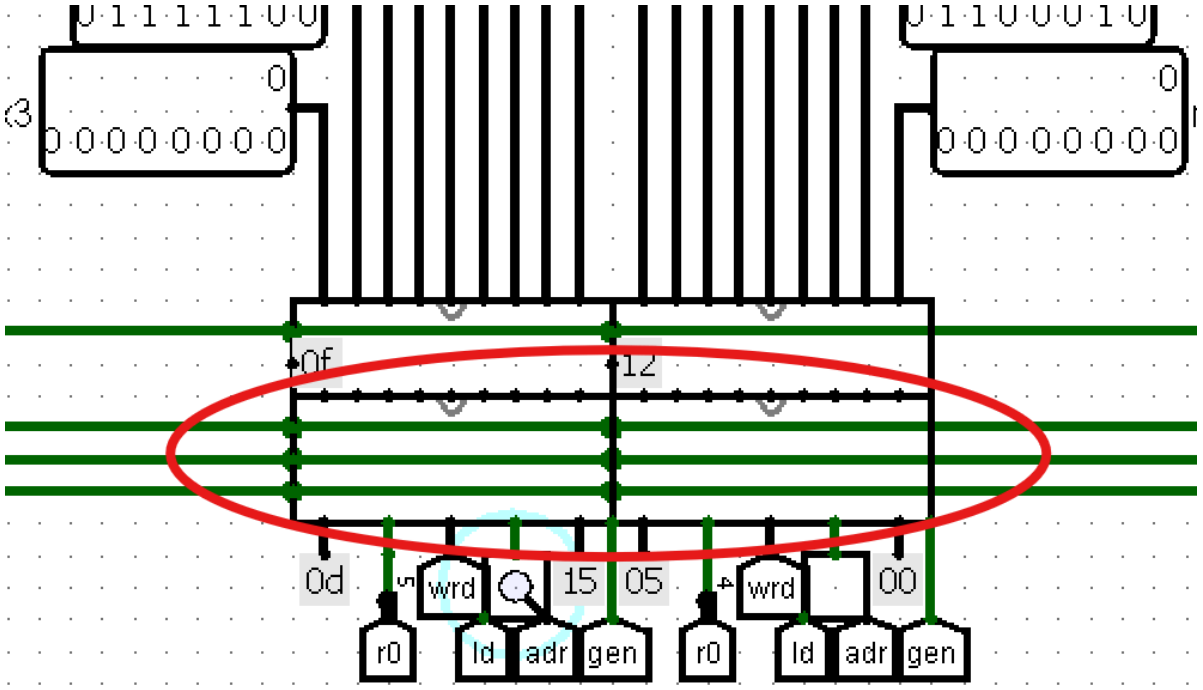



рис. 16 Переключатель положения матрицы на схеме Backend

Существует в проекте в 8 экземплярах, также присутствует 2 упрощённых экземпляра, которые просто выводят рисунок буквы по её порядковому номеру. Наиболее сложная схема данного блока, которая выполняет сразу несколько функций:

1. Хранит константы рисунков для матрицы и выводит нужную букву по её порядковому номеру.
2. Получает на вход и хранит соответствующую ему букву слова
3. “Открывает” и “закрывает” двери в зависимости от состояния регистра процессора r0.
4. Выводит на матрицы надписи, соответствующие победе и поражению

Модуль имеет 9 входов и 9 выходов. Все выходы являются значениями для матриц, которые обрабатываются дополнительным модулем и поступают в матрицу. На входы поступают следующие значения:

- 1-битные входы:
  - Game Over - вход, отвечающий за завершение игры поражением
  - Win - вход, отвечающий за завершение игры победой
  - reset - вход для сброса состояния переключателя (начало новой игры)
  - ctrl - вход, контролирующий “открытие” двери. Получает соответствующий бит регистра процессора r0
  - clock - вход тактового генератора

- adr - вход тактового генератора, контролируемый адресным контроллером тактов. Получает такт во время загрузки буквы слова, соответствующей данному переключателю в ОЗУ.

- 5-битовые входы
  - Let - вход буквы слова, соответствующей переключателю. Поступает напрямую из словаря
  - WINLet - вход буквы, выводимой при победе. Задан константой и может быть легко изменён
  - GOLet - вход буквы, выводимой при поражении. Задан константой и может быть легко изменён

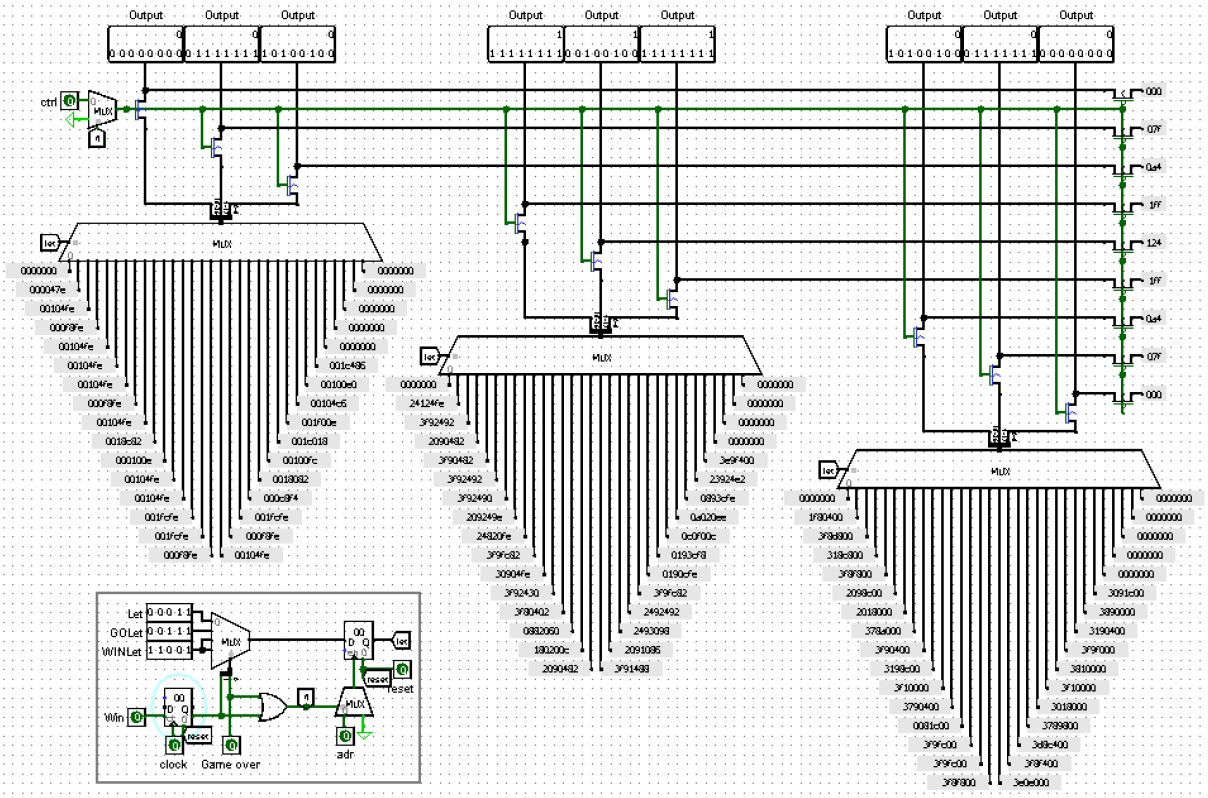



рис. 17 Полная схема одного из переключателей

Главными элементами схемы являются три мультиплексора с пятью выбирающими битами и 27-битными константами на входах. Использование 27-битных констант необходимо для упрощения схемы и минимизации используемых мультиплексоров - три 9-битных константы объединены в одну путём конкатенации и при выборе разделяются на 3 выхода, уходящих в матрицы.

Над мультиплексорами располагается блок, отвечающий за “открытие” и “закрытие” дверей. Справа за транзисторами p-типа располагаются константы изображения закрытой двери. До получения сигнала что данная буква открыта (поднят соответствующий бит регистра процессора r0) или что игра завершена

одним из способов на выход поступают константы закрытой двери, иначе буквы с мультиплексоров.

В переключателе матрицы присутствует блок контроля, задающий логику поведения (см. рис. 18). Основой блока является регистр, запоминающий порядковый номер буквы. Значение регистра обновляется при получении сигнала с входа adr или при получении сигнала что игра завершена одним из способов. На вход регистра подаётся одно из трёх 5-битовых значений (порядковый номер буквы, отображаемой на матрице в данный момент).
В начале игры параллельно с загрузкой слова в ОЗУ необходимо загрузить буквы в переключатели. Для этого напрямую из словаря на вход Let подаётся текущая буква и в нужный момент (когда из словаря поступает буква, адрес которой соответствует порядковому номеру переключателя (и, соответственно матрицы)) обновляется значение регистра. Это контролируется входом adr.
При поступлении сигнала что игра завершена поражением мультиплексор переключается на значение GOLet. Одновременно с этим второй мультиплексор переключается и подаёт сигнал на регистр для запоминания нового значения. В этот же момент все двери “открываются” и на матрицы выводится надпись “GAMEOVER”.
При поступлении сигнала что игра завершена победой игрока активируется счётчик, подключённый к тактовому генератору. Он обеспечивает задержку в половину секунды, которая позволяет игроку увидеть угаданное слово полностью. После переполнения значения счётчика он останавливается на финальном значении и передаёт сигнал на мультиплексоры. Так же как и при поражении они переключаются и на регистре обновляется значение. “Открываются” двери и матрицы выводят надпись “YOU WIN!”

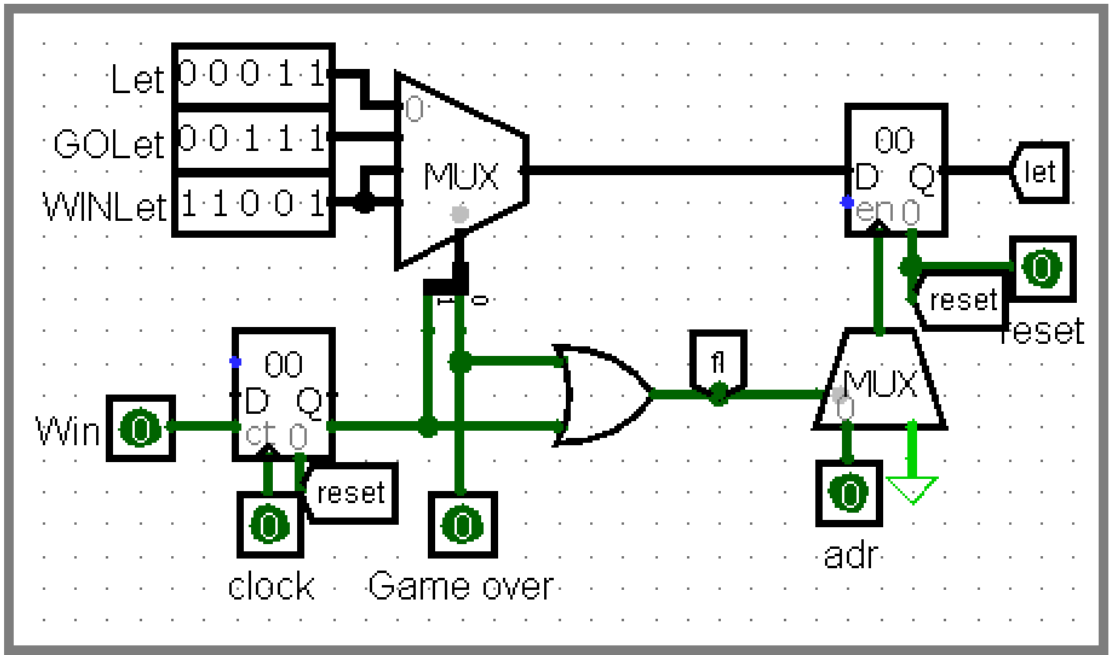



рис. 18 Блок контроля переключателя

### 3.2.5 - Блок контроля игры

Расположен слева от блока взаимодействия с матрицами и неразрывно с ним связан. Ведёт подсчёт жизней и отвечает за вывод их количества на дисплей и за контроль конечных состояний игры (победа, поражение). Все состояния передаются на блок контроля матриц, а также на процессорный и тактовый блоки.

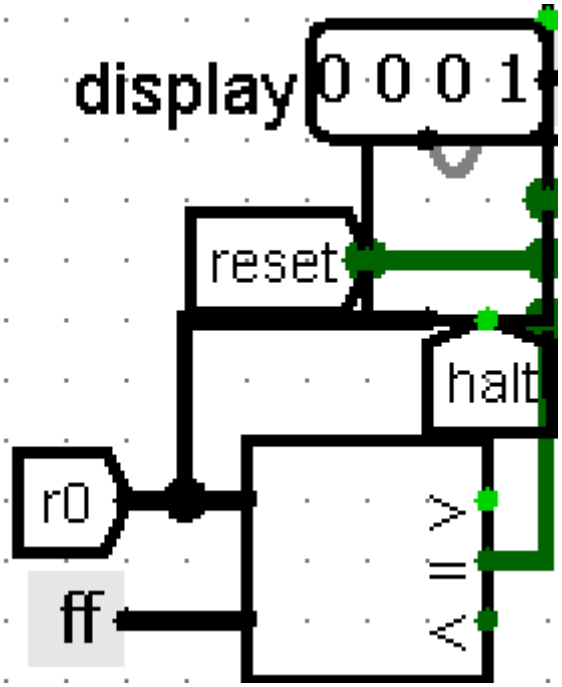



рис. 19 Блок контроля игры на схеме

Блок состоит из Счётчика жизней и компаратора.

Компаратор отвечает за состояние победы в игре. Он сравнивает текущее состояние регистра процессора r0 с маской 0b11111111 и при совпадении поднимает флаг победы, который сразу же передаётся остальным модулям.

Счётчик жизней имеет более сложную структуру.

Он состоит из регистра, компаратора, счётчика, D-триггера и мультиплексора.

Принцип действия счётчика прост - он сравнивает текущее значение регистра процессора r0 и значение на прошлом ходу пользователя. Совпадение означает что ни буква была введена неправильно и счётчик увеличивается на 1. Предыдущее значение регистра процессора r0 хранится в регистре внутри модуля. Значение в регистре обновляется после завершения одной итерации игры.

При переполнении счётчика на выход подаётся сигнал “Game Over”, означающий что жизни закончились и игра завершена поражением пользователя. Пользователь

имеет 6 жизней, поэтому трёхбитовое значение счётчика дополняется нулём и передаётся на мультиплексор, с которого и передаётся на инвертор (см. раздел Внешняя схема, инвертор значения жизней). Инвертор необходим поскольку подсчёт жизней ведётся с 0 до 7, а выводить необходимо оставшиеся жизни.

D-Триггер используется для контроля начального положения дисплея во время “нулевого круга” процессора (подробнее в пункте 3.2.6 “Процессор и память”)

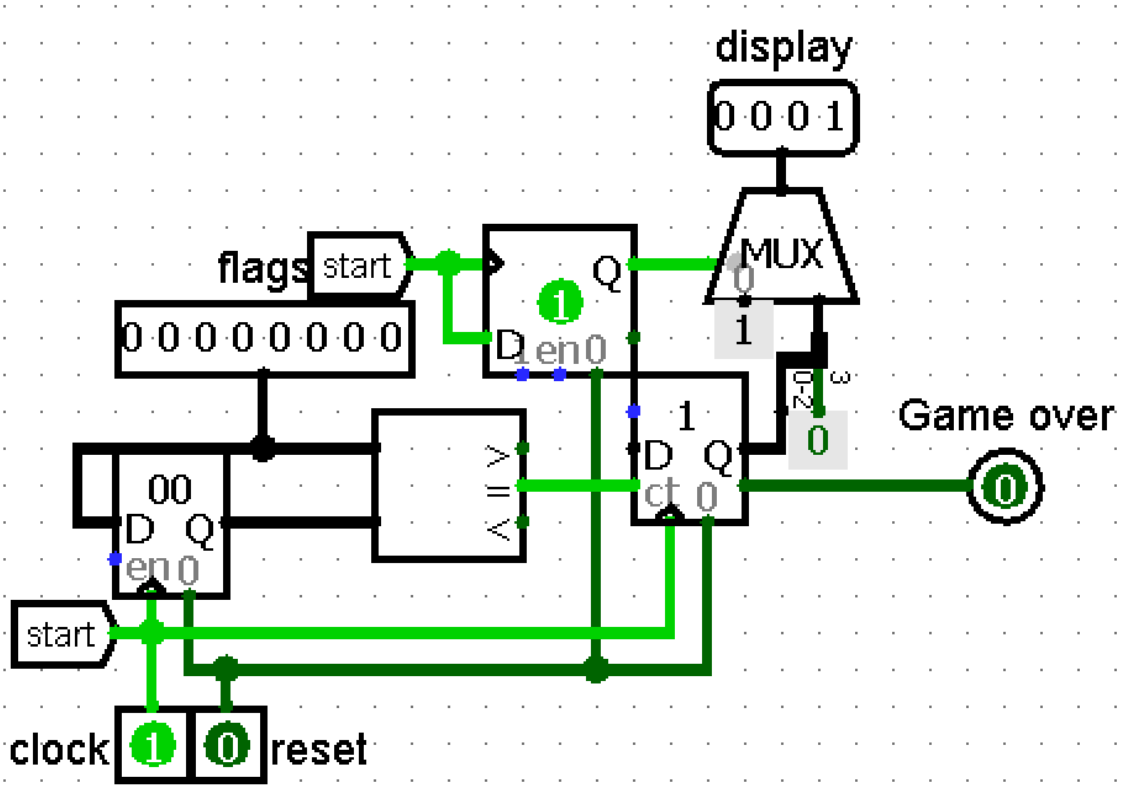



рис. 20 Схема счётчика жизней

### 3.2.6 - Процессор и память

Проект построен на взаимодействии с процессором Coco-de-Mer-8 версии 5, модифицированном в рамках текущего проекта. Процессор взаимодействует с памятью по Гарвардской системе. Схема подключения процессора к ОЗУ и ПЗУ приведена ниже.

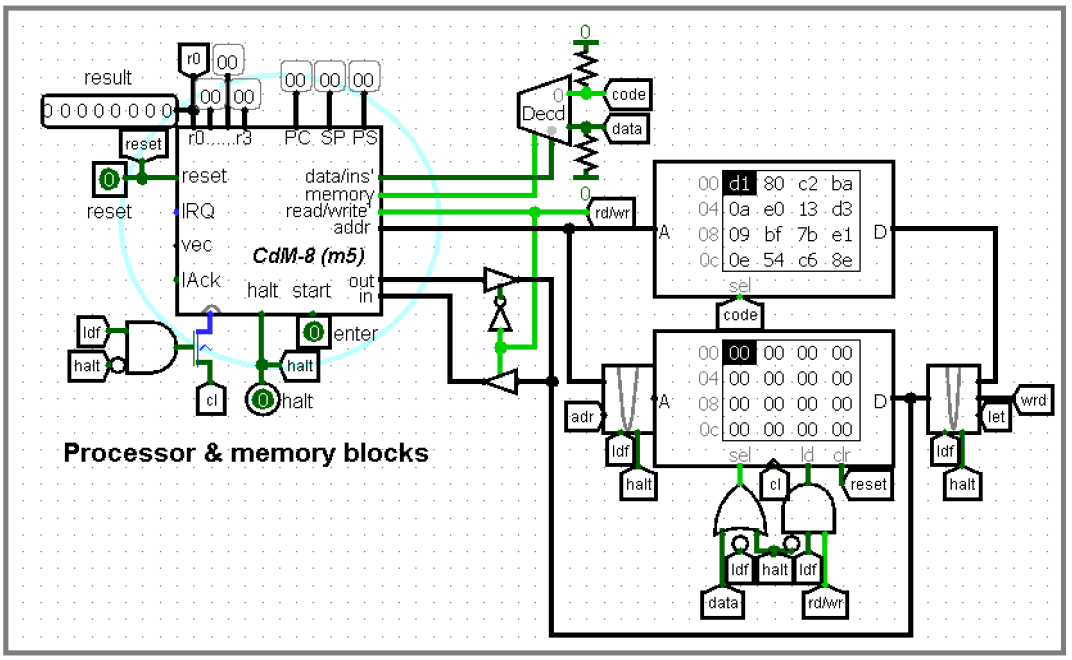



рис. 21 Блок процессора

### 3.2.6.1 - Модификации процессора

Для упрощения реализации проекта и значительного уменьшения дополнительных схем процессор был несколько модифицирован. Перед дальнейшим рассмотрением взаимодействия процессора с остальными блоками ознакомимся с изменениями, внесёнными в первоначальную схему. Красным цветом на рисунке обозначены изменения первоначальной схемы.

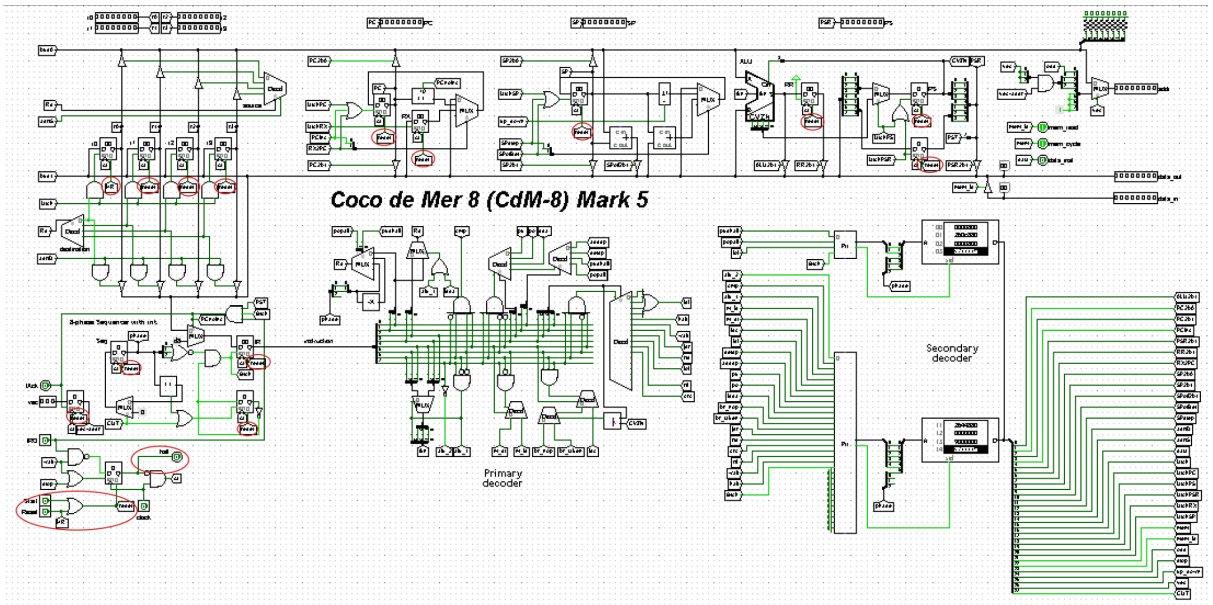



рис. 22 Изменения в схеме процессора

Модификация позволила вывести из процессора сигнал halt, обозначающий завершение выполнения кода процессором, который активно используется для контроля программно-аппаратного взаимодействия.

Кроме того были добавлены входные сигналы start и reset. Первый перезапускает выполнение кода процессором, а второй полностью сбрасывает состояние процессора к стартовому. Особенность сигнала start состоит в том, что он не сбрасывает значение регистра r0, в котором хранятся результаты вычислений процессора.

#### 3.2.6.2 - Этапы работы схемы

Перед переходом к программно-аппаратному взаимодействию необходимо рассмотреть принцип и этапы работы схемы. Работу схемы можно разделить на 5 различных этапов:

##### 3.2.6.2.1 - Начальное состояние игры

Это состояние встречается всего один раз в начале каждой игры. Во время этого состояния работают в основном тактовый и адресный блоки и блок словаря. На этом этапе происходит загрузка слова в ОЗУ и переключатели состояния матриц. Загрузка происходит последовательно с первой по восьмую буквы слова и соответственно занимает 16 тактов Logisim. По завершении загрузки слова подаётся сигнал ldf и состояние сменяется.

##### 3.2.6.2.2 - “Нулевой круг”

Это особый этап работы процессора, когда он проходит одну итерацию программы без загруженной пользователем буквы. Он возник в связи со спецификой процессора и программно-аппаратного взаимодействия. На этом этапе слово проверяется на отсутствие пробелов и корректность загрузки.

##### 3.2.6.2.3 - Этап взаимодействия с пользователем

На этом этапе все процессы в схеме останавливаются и ожидается ввод буквы игроком. В этот момент выбранная буква автоматически загружается в ОЗУ процессора и ожидается подтверждение ввода нажатием кнопки “Enter”.

##### 3.2.6.2.4 - Этап работы процессора

При нажатии кнопки “Enter” управление ОЗУ передаётся от клавиатуры процессору и он начинает итерацию программы. Происходит проверка соответствия введённой буквы и результат считывается из регистра процессора r0. По завершении программы процессором загорается сигнал halt, рассчитывается количество жизней и наступает этап взаимодействия с пользователем или этап завершения игры.

#### 3.2.6.2.5 - Этап завершения игры

Это финальный этап, который наступает при завершении игры одним из способов - победе или поражении игрока. Во время этого этапа происходит переключения состояния матриц (подробнее в разделе 3.2.4) и схема “замирает”. С этого момента пользователю недоступно взаимодействие с игрой. Переход на этап начала игры осуществляется путём нажатия кнопки “New Game”.

### 3.2.6.3 - Программно-аппаратное взаимодействие

Взаимодействие процессора с остальными блоками схемы происходит посредством ОЗУ и регистра процессора r0. Состояние регистра r0 считывается постоянно, исходя из него, происходят соответствующие изменения в схеме, а взаимодействие с ОЗУ происходит попеременно. Поведение и очерёдность взаимодействия с ОЗУ определяется специальными контроллерами, расположенными слева, справа и снизу от ОЗУ. Все контроллеры работают в зависимости от положений сигналов ldf и halt.

Левый контроллер определяет к какому адресу ОЗУ происходит обращение, правый - значение, вносимое/выводимое с ОЗУ, а нижний - состояние сигналов sel и ld ОЗУ.

Контроллеры имеют по три положения, которые они принимают в зависимости от текущего этапа (см. Этапы работы схемы) - взаимодействие процессора с ОЗУ, загрузка слова в ОЗУ и взаимодействие пользователя с ОЗУ.

## 3.3 - Animation

### 3.3.1 - Animation Switch

Модуль Animation Switch используется в модуле Animation Controller. Он, подобно модулю переключателя положения матриц (см. раздел 3.2.4.3), в зависимости от значения, подаваемого на вход,  передаёт на выход числа, использующиеся для трансляции на матрицу нужного изображения.

Этот модуль имеет четыре входа, на три из которых подаются 3-битные числа, с помощью которых на мультиплексоре выбираются соответствующие доступные во время игры значения для конкретной матрицы. (подробнее см. в разделе Animation Controller) На четвёртый вход подаётся 2-битное число, которое передаётся к мультиплексорам через туннель pict и используется для выбора конкретного значения для матрицы из доступных. На рисунке представлена одна из 3 идентичных составных частей модуля. Так же, как и в переключателе положения матриц используются 27-битные константы вместо 9-битных (подробнее в разделе 3.2.4.3).

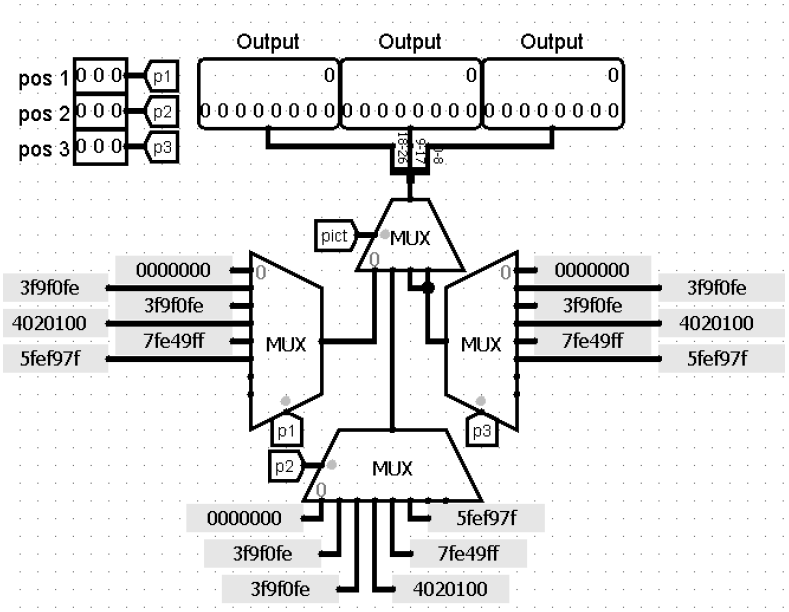



рис. 23 Схема выбора позиции двери или человечка на матрице

### 3.3.2 - Animation Controller

Блок Animation Controller отвечает непосредственно за трансляцию картинок на 10 матриц: первая и десятая матрица отвечают только за положение персонажа, остальные помимо этого отвечают за двери.

Как уже было сказано ранее (см. раздел Animation Switch), значение 3-х битного числа, подаваемого на Animation Switch, позволяет выбрать нужную позицию из доступных. Для этого используется одно из значений от 0 до 5. Каждое из них выбирает нужную картинку.

| Значение let | Положение на матрице |
|---|---|
| 0 | пустая матрица |
| 1 | стандартное положение персонажа |
| 2 | положение персонажа с поднятыми руками |
| 3 | пустая открытая дверь |
| 4 | закрытая дверь |
| 5 | открытая дверь с персонажем |

рис. 24 Таблица соответствий значения let и матриц

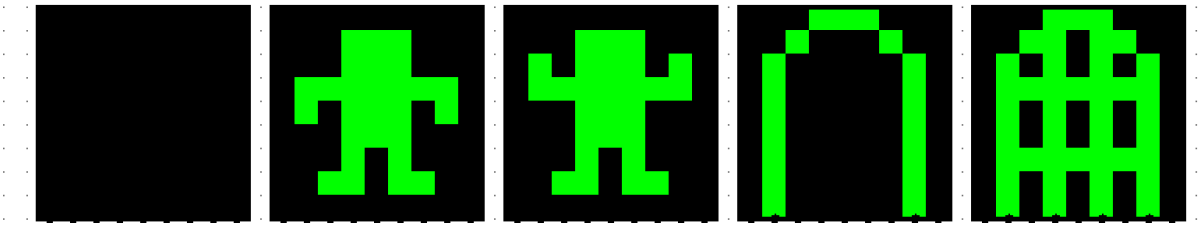


рис. 25 Доступные положения матриц

Поскольку вторая-девятая матрицы отвечают за двери, на соответствующие им Animation Switch подаются значения 4, 3, 5. На первый Animation Switch подаются только 2 состояния 1 и 0. А на последний подаются 3: 1, 2, 0.

На вход самого Animation Controller подаётся одно 8-битное число - значение регистра r0. От него зависит 3-битное значение let, подаваемое на Animation Switch. Рассмотрим логику выбора значения let в зависимости от значения r0 для каждой матрицы.

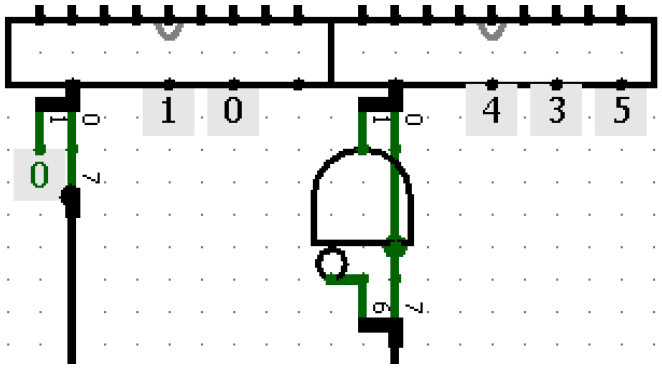



рис. 26 Animation Switches для первой и второй матриц

У первой матрицы всего 2 доступные позиции, поэтому бит 1 всегда нулевой. Бит 0 становится единичным только в том случае, когда седьмой бит r0 единичный, то есть когда угадана первая буква слова.

У второй матрицы значение, передаваемое в Animation Switch также будет равно 00, пока седьмой бит r0 не станет равен 1. Однако в данном случае значение let также зависит от шестого бита. Если он равен 1, значит вторая буква угадана, и матрица перейдёт в позицию пустой открытой двери. Иначе на первой матрице должен остаться персонаж.

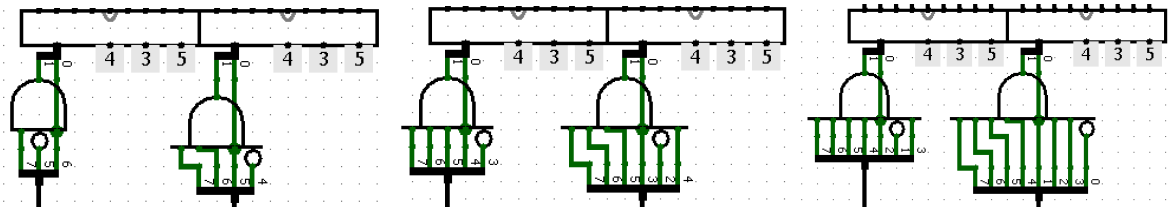


рис. 27 Animation Switches для третьей-восьмой матриц

Для оставшихся матриц дверей логика выбора позиции аналогичная. Для того, чтобы дверь была открыта буква в соответствующей позиции должна быть угадана. Для того, чтобы в двери не было персонажа, должна быть угадана следующая буква.

рис. Animation Switches для седьмой и восьмой матриц

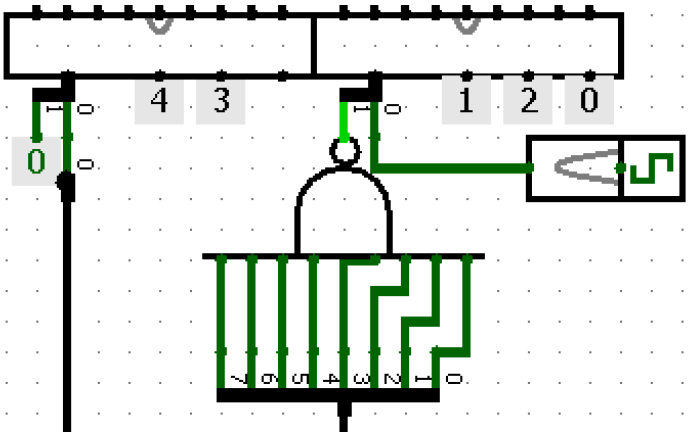



рис. 28 Animation Switches для девятой и десятой матриц

Десятая матрица отвечает за положение персонажа при угаданном слове и его анимацию.

При любом значении r0, кроме 0b11111111, первый бит let будет единичным, а потому матрица будет пустой. Если же слово полностью угадано, то с помощью тактового генератора и делителя тактов значение let будет меняться с 00 на 01.

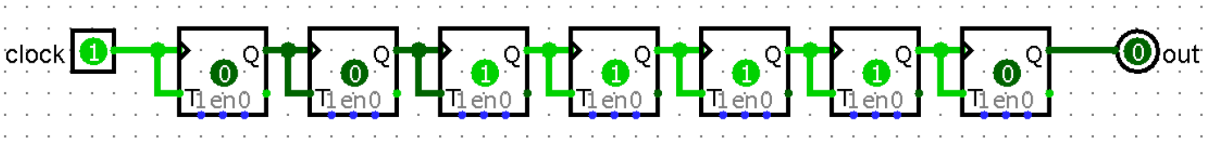



рис. 29 Схема делителя тактов

## 4. Программная часть

Программная часть реализована на процессоре CDM-8 (подробнее см. в разделе 3.2.6 - Процессор и память. Программно-аппаратное взаимодействие описывается в разделе 3.2.6)

Результат выполнения программы выводится непосредственно в регистр r0, откуда и считывается аппаратными модулями во время работы процессора. Результатом выполнения программы является строка из 8 битов, означающих состояние каждой отдельной буквы в слове. 1 на позиции, соответствующей букве означает что буква уже угадана и дверь открыта. Регистр r1 отведён под хранение битовой маски.

Перед выполнением кода в ОЗУ процессора аппаратно загружается слово на позиции 0x00 - 0x07 и буква, введённая пользователем на позицию 0x09. При завершении ввода слова и буквы на процессор подаётся сигнал и он начинает выполнение программы.

В начале программы процессор загружает маску в регистр r1, букву, введённую пользователем, в регистр r3 и начинает цикл проверки. В стеке сохраняется адрес текущей проверяемой буквы слова, буква загружается в регистр r2 и сравнивается с выбором пользователя. При совпадении маска применяется к результату, иначе игнорируется. В конце итерации цикла возвращается и инкрементируется адрес текущей буквы, маска сдвигается на бит вправо и начинается новая итерация цикла. Цикл завершается при достижении пустой ячейки ОЗУ (с адресом 0x08)

```
asect   0x00

ldi r1, 0b10000000 #маска
ldi r3, 0x09 #сохраняем букву
ld r3,r3

while
    push r2
    ld r2, r2 #отщепляем букву от фразы
    tst r2    #проверяем одну букву
stays nz     #пока оно не ноль
    if
        cmp r2,r3 #сравниваем с текущей
    is z #если буквы совпадают
        or r1,r0 #применяем маску
    fi
    pop r2
    inc r2 #переходим к следующему значению стека
    shr r1 #сдвигаем маску на 1 бит вправо
wend

halt
end
```

рис. 30 Программа для проверки наличия буквы в слове

# 5. Руководство Пользователя

## 5.1 - Подготовка

Открыв данный проект в Logisim, следует для начала открыть раздел «Моделировать» и убедиться, что возле пункта «Моделирование включено» установлена галочка. В данный момент вы должны увидеть на экране что-то похожее на рисунок, представленный ниже.

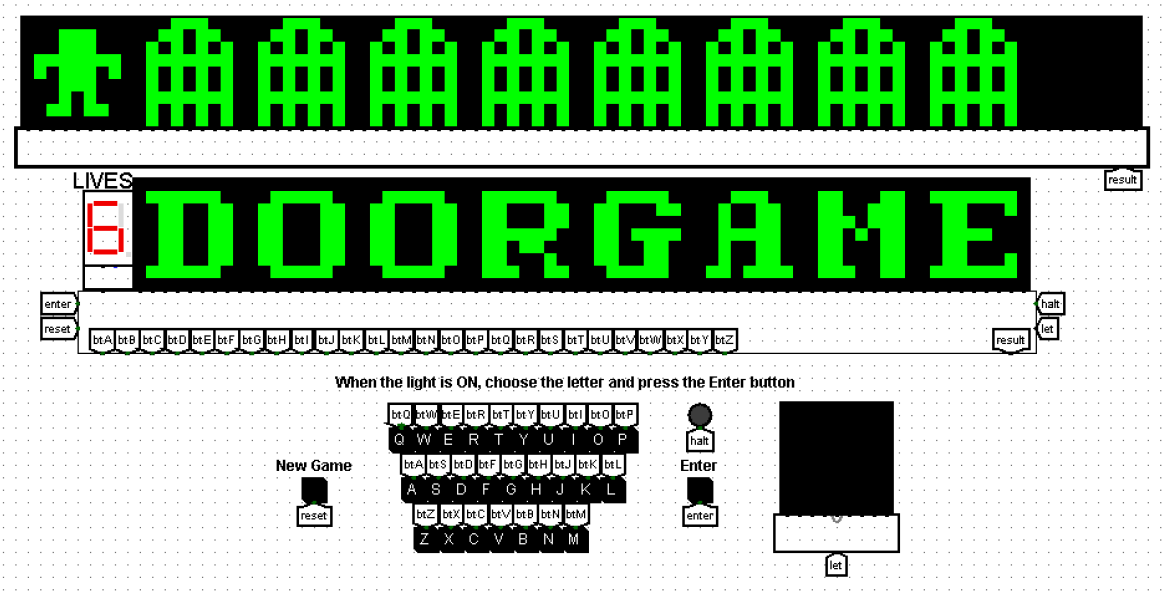



рис. 31 Внешний вид игры

Теперь откройте тот же раздел и установите «Тактовую частоту» на 512 Гц и убедитесь, что возле пункта «Такты включены» также установлена галочка. Теперь проект готов к использованию.

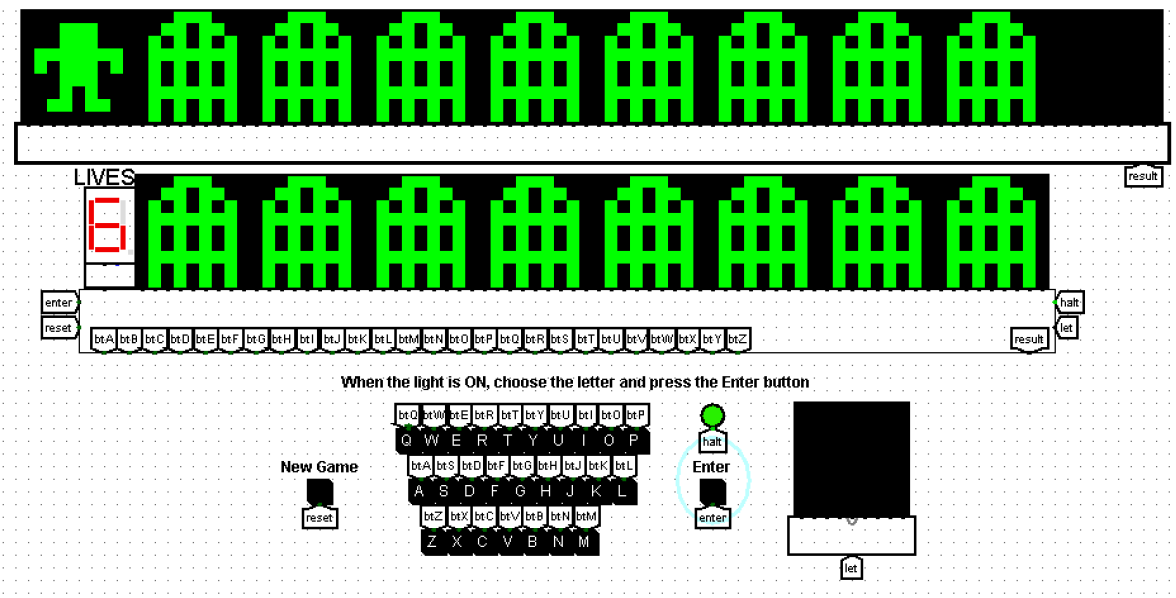



рис. 32 Вид игры перед первым вводом буквы

### 5.2 - Первый взгляд на игру

Посмотрим снова на интерфейс игры. Можно заметить несколько основных блоков:

- Матрицы, отражающие продвижение человечка
- Матрицы, которые будут отображать угаданное слово
- Количество жизней
- Клавиатура
- Кнопка ввода буквы
- Матрица, отражающая вводимую букву
- Кнопка перезагрузки для начала новой игры

Далее рассмотрим каждый из них.

### 5.3 - Матрицы движения персонажа

При угадывании любой буквы, дверь, расположенная над ней, откроется для персонажа. Если перед этой дверь все двери открыты, персонаж пройдет до последней открытой двери.

Иначе он будет оставаться на месте. Этот блок позволит следить за продвижением игры.

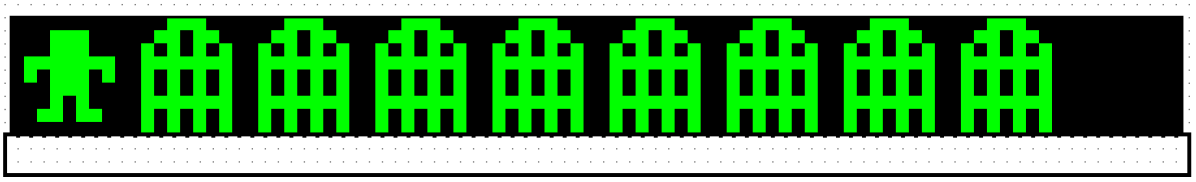


рис. 33 Матрицы движения человечка

### 5.4 - Матрицы слова

В начале игры, слово уже будет загадано. Задача, стоящая перед пользователем - угадать его, до того, как у него закончатся жизни. Матрицы, расположенные ниже позволят вам видеть, какие буквы были угаданы и на каких позициях они стоят.

В начале игры перед вами будет строка, подобная той, что расположена ниже.

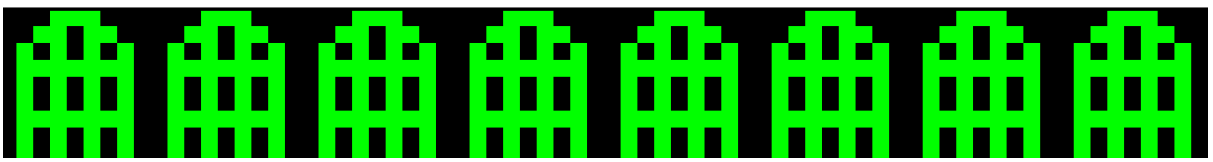


рис. 34 Закрытые матрицы слова

По мере отгадывания букв, они будут отображаться на соответствующих позициях, что упростит процесс отгадывания слова.

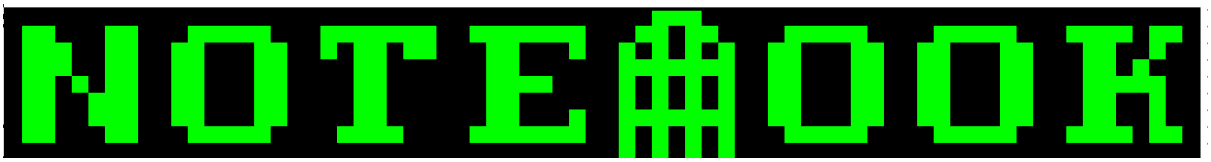



рис. 35 Матрицы слова

### 5.5 - Дисплей жизней

Как и в любой игре, количество допустимых ошибок ограничено, индикатор жизней позволит пользователю следить за этим. Игрок имеет 6 жизней в начале игры.

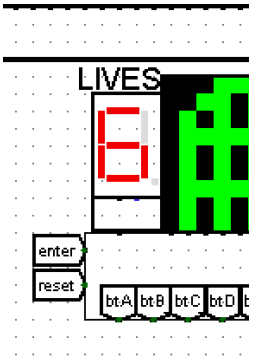



рис. 36 Индикатор количества жизней

### 5.6 - Клавиатура и ввод

Основной блок, с котором пользователю предстоит взаимодействовать во время всей игры - это клавиатура. С помощью неё будет осуществляться ввод всех букв. Ниже представлена сама клавиатура.

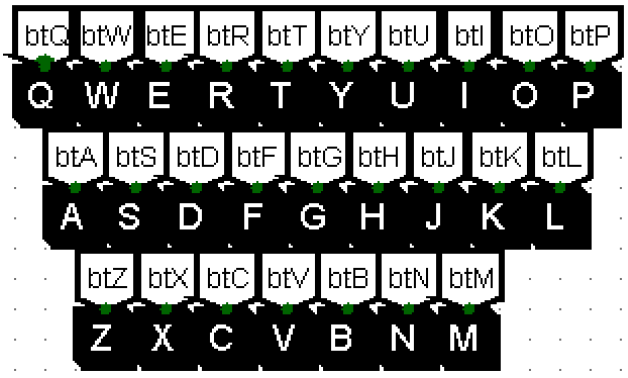



рис. 37 Клавиатура

Нажмите на любую букву, теперь её нужно каким-то образом ввести.

Кнопка Enter расположенная справа от клавиатуры, отвечает за подтверждение выбора буквы. Когда лампочка над кнопкой горит зелёным, нажмите кнопку и буква будет введена.

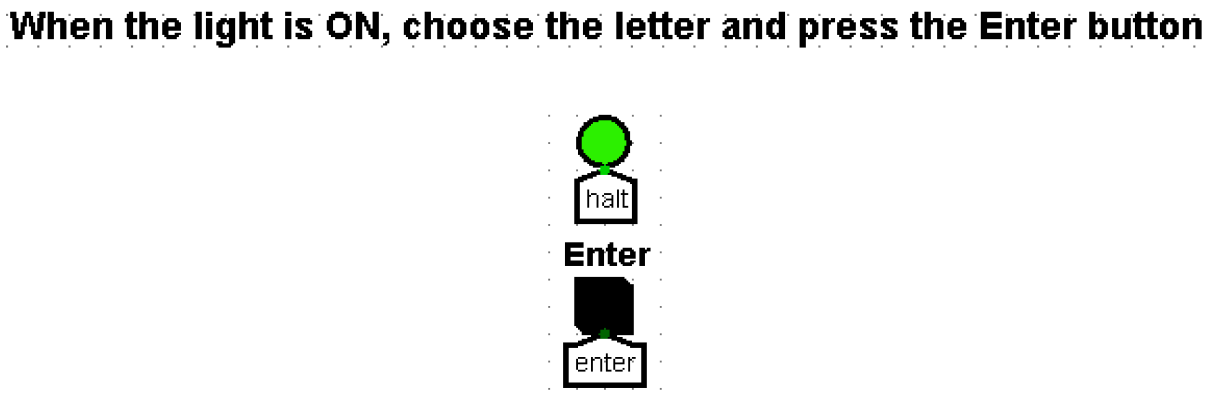



рис. 38 Кнопка ввода

**5.7 - Отражение выбранной буквы**

Как можно было заметить, при нажатии на любую букву клавиатуры, на матрице справа отобразиться соответствующая буква.

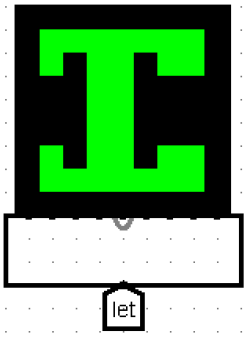



рис. 39 Матрица отображения вводимой буквы

**5.8 - Запуск новой игры**

Первая игра начнётся автоматически, однако в любой момент, когда игра наскучит или пользователь завершит игру и захочет сыграть ещё раз, следует нажать кнопку New Game для начала новой игры с совершенно новым загаданным словом.


рис. 40 Кнопка перезапуска

### 5.9 - Результат игры

Итак, пользователь завершил игру: у него закончились жизни или он отгадал слово. По завершении игры он увидит один из двух экранов, сигнализирующих о победе или поражении.


рис. 41 Дисплей по окончании игры

## 6. Заключение

В рамках курсового проекта была разработана и реализована цифровая версия игры под названием “DOOR GAME”, которая является аналогом “Виселицы”. В ходе проектной работы были использованы программные инструменты Logisim и CocoIDE (CDM-8). Игра представляет собой уникальную адаптацию классической словесной головоломки, где игроку необходимо угадать слово из восьми букв, открывая виртуальные двери за счёт правильного выбора букв. Проект объединяет аппаратную и программную части, обеспечивая полноценное взаимодействие пользователя с игрой через интуитивно понятный интерфейс.

Командой были выполнены следующие ключевые задачи:

1. **Разработка аппаратной части**: спроектирована цифровая микросхема, включающая блоки словаря, клавиатуры, взаимодействия с матрицами, контроля игры, а также процессор и память.
2. **Создание программной части**: реализована программа для процессора CDM-8, которая обрабатывает ввод пользователя, проверяет наличие букв в загаданном слове и обновляет состояние игры. Программа оптимизирована для работы с ограниченными ресурсами процессора.
3. **Интеграция hardware и software**: обеспечено согласованное взаимодействие между аппаратными модулями и программным кодом, что позволило реализовать динамическую игровую логику. Также реализован счетчик жизней, отображение слова и анимация движения персонажа.
4. **Разработка руководства пользователя**: создано подробное описание интерфейса и механики игры, позволяющее пользователю быстро освоить правила и начать игру.

Выводы: В результате был создан функциональный прототип игры, который успешно реализует задуманную концепцию.