МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение
высшего образования «Санкт-Петербургский политехнический
университет Петра Великого»

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 «Математика и компьютерные науки»

Отчет о выполнении курсовой работы *Синтез функциональной схемы часов* по дисциплине «Теория алгоритмов»
2 вариант

Студент,
группы 5130201/20101 _____ Астафьев И. Е.

Преподаватель _____ Востров А. В.

«_____»_____ 2024г.

Санкт-Петербург, 2024

Содержание

Введение

Методология проектирования дискретных систем управления, базирующаяся на использовании конечных автоматов, играет ключевую роль в современной технике. Конечные автоматы предоставляют мощный инструмент для описания поведения сложных систем, позволяя моделировать их функционирование с учетом внешних событий и условий. В условиях широкого внедрения микропроцессорной техники этот подход становится особенно актуальным благодаря своей универсальности и высокой степени формализации.

Целью данной курсовой работы было создание функциональной схемы электронных часов с использованием конечных автоматов и моделирование работы построенной схемы. Такой подход позволил детально описать логику работы устройства, учитывая все возможные состояния и переходы между ними. Данная работа демонстрирует практическое применение теории конечных автоматов в области создания систем реального времени.

1 Постановка задачи

Для выполнения курсовой работы по заданному варианту (0 0 1 2 2 1 0), необходимо разработать функциональную схему электронных часов, которая будет включать следующие функции:

1. Отображение и корректировка: часы должны отображать и позволять корректировку стандартных значений – минут и часов.
2. Режим работы: часы должны работать в 12-часовом формате с индикацией a.m./p.m.
3. Отключение индикаторов: должно быть реализовано.
4. Останов часов: останов должен осуществляться после корректировки текущего времени.
5. Секундомер: должен присутствовать фиксацией накопленного времени и возможностью продолжения отсчета секундомера.
6. Звуковая сигнализация: каждый час должна подаваться звуковая сигнализация длительностью четыре секунды.

2 Описание объекта управления

Реализуемые электронные часы отображают часы, минуты и a.m./p.m., а также содержат индикатор звука и внешние кнопки управления a и b.

Для отображения работы часов используются:

1. 5 семисегментных дисплеев:
    - буква a или p в зависимости от соответствующей половины дня;
    - старший десятичный разряд часов;
    - младший десятичный разряд часов;
    - старший десятичный разряд минут;
    - младший десятичный разряд минут;
    - десятичный номер состояния;
2. Диод, отвечающий за индикатор звука.

Для управления часами используются кнопки внешнего управления - a и b. Входные воздействия на часы возможны нажатием одной из кнопок или их обеих одновременно.

3 Математическое описание

3.1 Конечный автомат

Конечный автомат [1], [2] — математическая модель дискретного устройства, которая описывается набором $A = (S, \Sigma, Y, s_0, \delta, \lambda)$, где:

- S — конечное множество состояний;
- Σ — конечное множество входных сигналов;
- Y — конечное множество выходных сигналов;
- s_0 — начальное состояние ($s_0 \in S$);
- $\delta : S \times \Sigma \to S$ — функция переходов;
- $\lambda : S \times \Sigma \to Y$ — функция выходов.

Конечный автомат работает в дискретные моменты времени, и в момент времени $t = 0$ автомат всегда находится в состоянии s_0.

3.2 Построение управляющего конечного автомата

Граф управляющего конечного автомата

На Рис. 1 представлен граф управляющего конечного автомата электронных часов.

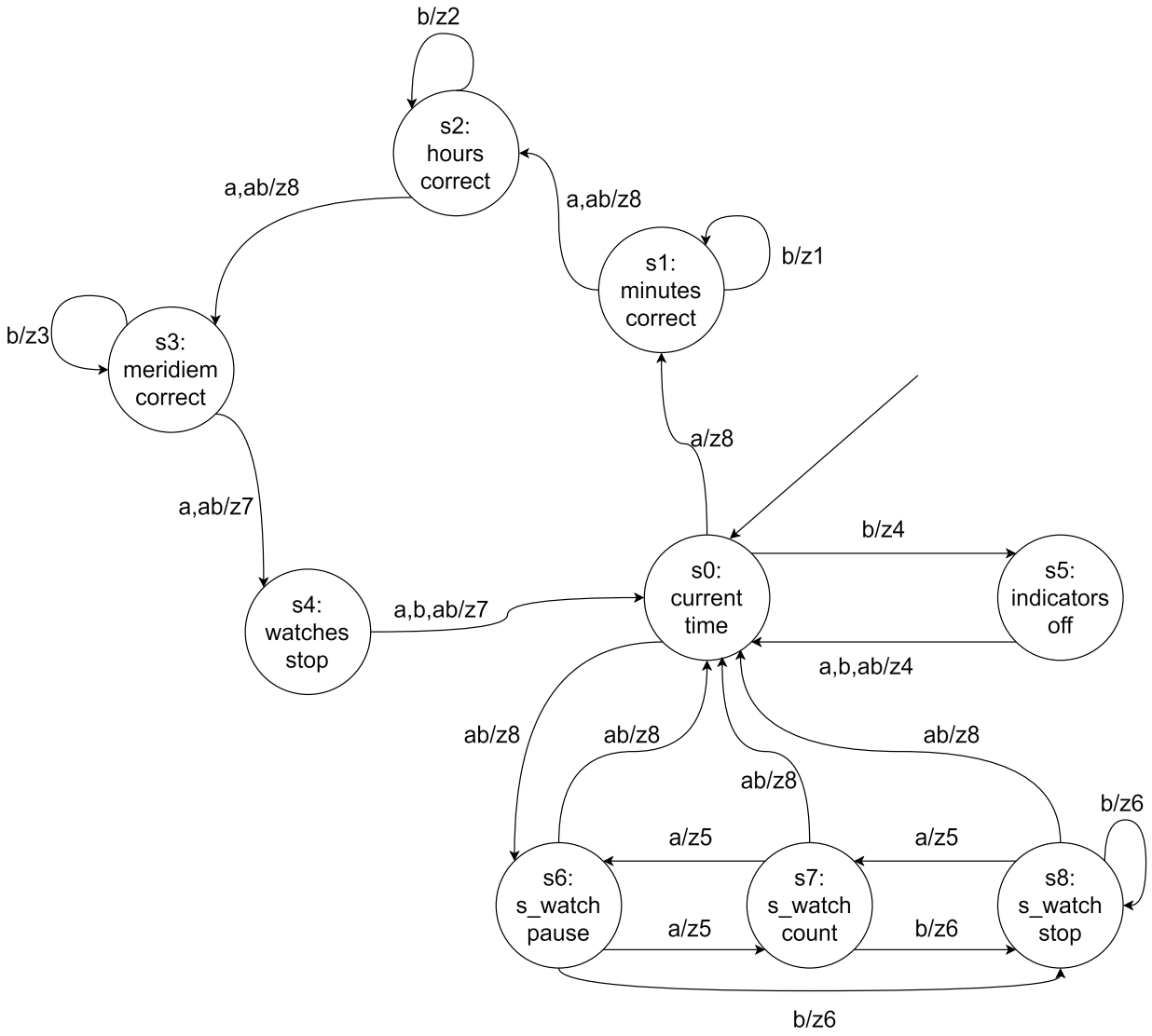

Рис. 1. Граф автомата, описывающего поведение электронных часов

Состояния

Было выделено 9 состояний $S = \{s0, s1, s2, s3, s4, s5, s6, s7, s8\}$.

- $s0$: **Отображение текущего времени.** Это начальное состояние автомата, в котором отображается текущее время. При нажатии кнопки "b"происходит переход в состояние $s5$ (индикаторы отключены). Нажатие кнопок "a "ab"переводит в соответствующие состояния настройки или остановки часов.

- $s1$: **Корректировка минут.** В этом состоянии происходит изменение значения минут. Нажатие кнопки "a"увеличивает значение минут на единицу, при этом индикаторы продолжают показывать текущие часы и минуты с учётом корректировок. Переход на "b"остаётся в том же состоянии, а на "ab"переводит обратно в состояние $s0$.

- $s2$: **Корректировка часов.** В данном состоянии корректируются значения часов. Нажатие кнопки "a"увеличивает значение часов на единицу. При нажатии "b"происходит переход в состояние $s1$ (корректировка минут), а на "ab"— возврат в $s0$.

- $s3$: **Корректировка времени суток (AM/PM).** В этом состоянии можно изменить время суток. Нажатие кнопки "a"переключает состояние между AM и PM. Переход "b"переводит в состояние $s2$ (корректировка часов), а "ab"— в состояние $s0$.

- $s4$: **Остановка часов.** В данном состоянии часы прекращают ход. Нажатие кнопки "b"возвращает в состояние $s0$ (отображение текущего времени), "a"или "ab"не изменяют состояния.

- $s5$: **Отключение индикаторов.** В этом состоянии все индикаторы отключены для экономии энергии. Нажатие кнопки "a"включает индикаторы, возвращая в состояние $s0$. При нажатии "b"или "ab"состояние остаётся неизменным.

- $s6$: **Секундомер на паузе.** В этом состоянии секундомер приостановлен. Нажатие кнопки "a"возобновляет отсчёт секунд, переходя в состояние $s7$ (счёт секундомера), нажатие "b"переводит в состояние $s8$, а "ab"возвращает в $s0$.

- $s7$: **Счёт секундомера.** В этом состоянии секундомер отсчитывает время. Нажатие кнопки "b"переводит секундомер в паузу (состояние $s6$), а "ab"сбрасывает отсчёт и возвращает в $s0$.

- $s8$: **Секундомер остановлен.** Секундомер полностью остановлен. Нажатие кнопки "b"возвращает в состояние $s6$ (пауза секундомера), а "ab"сбрасывает отсчёт секундомера и возвращает в $s0$.

В Таблице 1 приведены коды, необходимые для реализации состояний:

Таблица 1. Коды состояний

Состояние	Код
$s0$	0000
$s1$	0001
$s2$	0010
$s3$	0011
$s4$	0100
$s5$	0101
$s6$	0110
$s7$	0111
$s8$	1000

Входы

Входной алфавит был определен как $\Sigma = \{a, b, ab\}$:

- Кнопка **a**.

- Кнопка **b**.
- Одновременное нажатие кнопок **ab**.

В Таблице 2 приведены коды, необходимые для реализации входов:

Таблица 2. Коды входов

Вход	Код
a	00
b	10
ab	11

Выходы

Было выделено множество выходных сигналов $Y = \{z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8\}$.

- $z1$: **Увеличение минуты на 1.** Этот выход используется для увеличения значения минут на единицу при корректировке времени. Он активируется в состоянии $s1$ (корректировка минут) при нажатии соответствующей кнопки.
- $z2$: **Увеличение часа на 1.** Этот выход увеличивает значение часов на единицу. Он активируется в состоянии $s2$ (корректировка часов) при корректировке времени.
- $z3$: **Переключение режима AM/PM.** Данный выход используется для изменения режима времени суток. Он активируется в состоянии $s3$ (корректировка времени суток) при выполнении соответствующего действия.
- $z4$: **Переключение режима индикаторов.** Этот выход отвечает за включение или выключение индикаторов часов. Он активируется в состояниях, связанных с переключением индикаторного отображения, таких как $s0$ (отображение текущего времени) или $s5$ (отключение индикаторов).
- $z5$: **Запуск/пауза счётчика секундомера.** Данный выход используется для управления работой секундомера. Он активируется в состоянии $s6$ (секундомер на паузе) для запуска или временной остановки отсчёта.
- $z6$: **Сброс счётчика секундомера.** Этот выход позволяет сбросить значение секундомера. Он активируется в состоянии $s8$ (секундомер остановлен) или $s7$ (счёт секундомера) при соответствующем действии.
- $z7$: **Старт/стоп часов.** Этот выход управляет запуском или остановкой хода часов. Он активируется в состояниях, связанных с режимами работы часов, таких как $s4$ (остановка часов).
- $z8$: **Пустой выход.** Этот выход не выполняет никаких действий и используется как заглушка в тех состояниях, где не требуется выполнение дополнительных операций.

Функции переходов состояний и выходов

В таблицах 3 - 4 отображены функция переходов автомата (δ) и функция выходов автомата (λ) соответственно.

Таблица 3. Таблица переходов состояний графа

δ	a	b	ab
$s0$	$s1$	$s5$	$s6$
$s1$	$s2$	$s1$	$s2$
$s2$	$s3$	$s2$	$s3$
$s3$	$s4$	$s3$	$s4$
$s4$	$s0$	$s0$	$s0$
$s5$	$s0$	$s0$	$s0$
$s6$	$s7$	$s8$	$s0$
$s7$	$s6$	$s8$	$s0$
$s8$	$s7$	$s8$	$s0$

Таблица 4. Таблица выходов графа

λ	a	b	ab
$s0$	$z8$	$z4$	$z8$
$s1$	$z8$	$z1$	$z8$
$s2$	$z8$	$z2$	$z8$
$s3$	$z7$	$z3$	$z7$
$s4$	$z7$	$z7$	$z7$
$s5$	$z4$	$z4$	$z4$
$s6$	$z5$	$z6$	$z8$
$s7$	$z5$	$z6$	$z8$
$s8$	$z5$	$z6$	$z8$

3.3 Управляющие воздействия

Импульсные микрокоманды

Импульсные микрокоманды - это кратковременные воздействия, которые подаются в момент нажатия внешних кнопок [3]. Значение импульсной микрокоманды (кратковременное воздействие) может быть отлично от нуля лишь во время перехода из одного состояния в другое.

Импульсные микрокоманды описаны в Таблице 5.

Таблица 5. Коды состояний

Импульсная микрокоманда	Описание
$i1$	увеличить минуты на 1
$i2$	увеличить часы на 1
$i3$	переключить a.m./p.m
$i4$	переключить режим индикаторов вкл./выкл.
$i5$	запустить/поставить на паузу секундомер
$i6$	сбросить секундомер
$i7$	переключить режим останова часов вкл./выкл.
$i8$	переключить режим сигнализации вкл./выкл.

Потенциальные микрокоманды

Потенциальные микрокоманды - это продолжительное воздействие, которое действует в период нахождения автомата в определенном состоянии и может измениться только при переключении автомата в другое состояние [3].

Потенциальные микрокоманды описаны в Таблице 6.

Таблица 6. Коды состояний

Потенциальная микрокоманда	Описание
$L1$	корректировка минут
$L2$	корректировка часов
$L3$	корректировка a.m./p.m
$L4$	останов часов
$L5$	секундомер
$L6$	индикация отключена
$L7$	сигнализация

3.4 Минимизация функций

Функция SOPform

Функция `SOPform` из библиотеки `sympy.logic` [4] языка программирования Python используется для представления логической функции в СДНФ (Sum of Products, SOP).

Она преобразует список минтермов (входных комбинаций, приводящих к результату `1`) в упрощённое выражение, используя алгоритм минимизации логических функций.

```
sympy.logic.boolalg.SOPform(variables, minterms, dontcares=None)
```

Основной принцип работы

Функция `SOPform` применяет алгоритм Куайна - Мак-Класки [5]. Он работает путем анализа всех возможных минтермов (входных комбинаций, которые приводят к результату "1") и их объединения в группы, где можно устранить одинаковые части. Алгоритм поочередно объединяет минтермы, если они отличаются только в одном бите, и повторяет этот процесс до тех пор, пока не будет получено минимальное количество выражений, представляющих ту же функцию.

Результат представлен в виде логической функции `Or` (логическое *или*), которая соответствует минимальной форме суммы произведений.

Входные параметры

1. **Переменные:** список символьных переменных, задающих логическую функцию. Они передаются как первый аргумент.
2. **Минтермы:** список всех входных комбинаций, приводящих к результату `1`. Минтермы могут быть переданы в следующих форматах:
   - В виде списка списков (например, `[[0, 0, 0, 1], [0, 1, 1, 1]]`).
   - В виде списка целых чисел, представляющих двоичные комбинации (например, `[1, 3, 7, 15]`).
   - В виде словарей, задающих только известные переменные (например, `{w: 0, x: 1}`).
3. **Don't-care условия (опционально):** список входных комбинаций, которые могут быть проигнорированы. Это позволяет дополнительно минимизировать функцию. Также могут быть заданы в виде списков, чисел или словарей.

Результат

Функция возвращает одно из возможных минимальных логических выражений, которое удовлетворяет заданным условиям. Результат представлен в виде логических операций (`&` для *и*, `|` для *или*, `~` для отрицания).

Функции для переходов состояний

Так как состояния и входы автомата закодированы, была создана таблица истинности переходов состояний (Таблица 7).

Таблица 7. Переходы состояний автомата

	a	b	q1	q2	q3	q4	Q1	Q2	Q3	Q4
0	1	0	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	1	0	1
2	0	1	0	0	0	1	0	0	0	1
3	1	0	0	0	0	1	0	0	1	0
4	1	1	0	0	0	1	0	0	1	0
5	0	1	0	0	1	0	0	0	1	0
6	1	0	0	0	1	0	0	0	1	1
7	1	1	0	0	1	0	0	0	1	1
8	0	1	0	0	1	1	0	0	1	1
9	1	0	0	0	1	1	0	1	0	0
10	1	1	0	0	1	1	0	1	0	0
11	0	1	0	1	0	0	0	0	0	0
12	1	0	0	1	0	0	0	0	0	0
13	1	1	0	1	0	0	0	0	0	0
14	0	1	0	1	0	1	0	0	0	0
15	1	0	0	1	0	1	0	0	0	0
16	1	1	0	1	0	1	0	0	0	0
17	1	1	0	0	0	0	0	1	1	0
18	1	1	0	1	1	0	0	0	0	0
19	1	0	0	1	1	0	0	1	1	1
20	0	1	0	1	1	0	1	0	0	0
21	0	1	0	1	1	1	1	0	0	0
22	1	0	0	1	1	1	0	1	1	0
23	1	1	0	1	1	1	0	0	0	0
24	0	1	1	0	0	0	1	0	0	0
25	1	0	1	0	0	0	0	1	1	1
26	1	1	1	0	0	0	0	0	0	0

Затем для минимизации функций для $Q1$, $Q2$, $Q3$, $Q4$ была использована функция SOPform:

```
import pandas as pd
from sympy.logic.boolalg import SOPform
from sympy import symbols
```

```
4
5   table = pd.read_csv("table_state_to_state.csv")
6
7   columns = ['a', 'b', 'q1', 'q2', 'q3', 'q4', 'Q1', 'Q2', 'Q3', 'Q4']
8   result = pd.DataFrame(columns=columns)
9   for i in range(len(table)):
10      row = table.iloc[i]
11      a, b = 0, 0
12      match row['X']:
13          case 'a':
14              a = 1
15              b = 0
16          case 'b':
17              a = 0
18              b = 1
19          case 'ab':
20              a = 1
21              b = 1
22      q1, q2, q3, q4 = states_dict[row['current state']]
23      Q1, Q2, Q3, Q4 = states_dict[row['next state']]
24      result.loc[len(result)] = [a, b, q1, q2, q3, q4, Q1, Q2, Q3, Q4]
25
26  a, b, q1, q2, q3, q4, Q1, Q2, Q3, Q4 = symbols('a b q1 q2 q3 q4 Q1 Q2 Q3 Q4')
27
28  inputs = [[result.iloc[i][j] for j in range(6)] for i in range(len(result))]
29
30  outputs = {
31      'Q1': result['Q1'].to_list(),
32      'Q2': result['Q2'].to_list(),
33      'Q3': result['Q3'].to_list(),
34      'Q4': result['Q4'].to_list()
35  }
36
37  answers = dict()
38  for k in outputs.keys():
39      answers.update({k: SOPform([a, b, q1, q2, q3, q4], [inputs[i] for i in
        ↪  range(len(outputs[k])) if outputs[k][i] == 1])})
```

Таким образом для каждого Q были получены минимизированные функции (Рис. 2).

```
answers = dict()
for k in outputs.keys():
    answers.update({k: SOPform([a, b, q1, q2, q3, q4], [inputs[i] for i in range(len(outputs[k])) if outputs[k][i] == 1])})
```

```
answers['Q1']
```

$(b \wedge q_2 \wedge q_3 \wedge \neg a \wedge \neg q_1) \vee (b \wedge q_1 \wedge \neg a \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4)$

```
answers['Q2']
```

$(a \wedge q_2 \wedge q_3 \wedge \neg b \wedge \neg q_1) \vee (a \wedge q_3 \wedge q_4 \wedge \neg q_1 \wedge \neg q_2) \vee (b \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4) \vee (a \wedge q_1 \wedge \neg b \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4)$

```
answers['Q3']
```

$(a \wedge b \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_4) \vee (a \wedge q_2 \wedge q_3 \wedge \neg b \wedge \neg q_1) \vee (a \wedge q_3 \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_4) \vee (a \wedge q_4 \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3) \vee (b \wedge q_3 \wedge \neg a \wedge \neg q_1 \wedge \neg q_2) \vee (a \wedge q_1 \wedge \neg b \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4)$

```
answers['Q4']
```

$(a \wedge q_3 \wedge \neg b \wedge \neg q_1 \wedge \neg q_4) \vee (a \wedge q_3 \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_4) \vee (a \wedge \neg b \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4) \vee (b \wedge q_4 \wedge \neg a \wedge \neg q_1 \wedge \neg q_2) \vee (b \wedge \neg a \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3)$

Рис. 2. Минимизированные функции для $Q1$, $Q2$, $Q3$, $Q4$

Теперь можно построить логический блок F, в котором будут описаны переходы из состояния в состояние по известному входу.

Функции импульсных микрокоманд

Также была создана таблица истинности для импульсных микрокоманд, которые зависят от входа и текущего состояния автомата (Таблица 8).

Таблица 8. Зависимость импульсных микрокоманд от входов и состояний

	a	b	q1	q2	q3	q4	i1	i2	i3	i4	i5	i6	i7	i8
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	1	0	0	0	0
2	0	1	0	0	0	1	1	0	0	0	0	0	0	0
3	1	0	0	0	0	1	0	0	0	0	0	0	0	0
4	1	1	0	0	0	1	0	0	0	0	0	0	0	0
5	0	1	0	0	1	0	0	1	0	0	0	0	0	0
6	1	0	0	0	1	0	0	0	0	0	0	0	0	0
7	1	1	0	0	1	0	0	0	0	0	0	0	0	0
8	0	1	0	0	1	1	0	0	1	0	0	0	0	0
9	1	0	0	0	1	1	0	0	0	0	0	0	1	0
10	1	1	0	0	1	1	0	0	0	0	0	0	1	0
11	0	1	0	1	0	0	0	0	0	0	0	0	1	0
12	1	0	0	1	0	0	0	0	0	0	0	0	1	0
13	1	1	0	1	0	0	0	0	0	0	0	0	1	0
14	0	1	0	1	0	1	0	0	0	1	0	0	0	0
15	1	0	0	1	0	1	0	0	0	1	0	0	0	0
16	1	1	0	1	0	1	0	0	0	1	0	0	0	0
17	1	1	0	0	0	0	0	0	0	0	0	0	0	0
18	1	1	0	1	1	0	0	0	0	0	0	0	0	0
19	1	0	0	1	1	0	0	0	0	0	1	0	0	0
20	0	1	0	1	1	0	0	0	0	0	0	1	0	0
21	0	1	0	1	1	1	0	0	0	0	0	1	0	0
22	1	0	0	1	1	1	0	0	0	0	1	0	0	0
23	1	1	0	1	1	1	0	0	0	0	0	0	0	0
24	0	1	1	0	0	0	0	0	0	0	0	1	0	0
25	1	0	1	0	0	0	0	0	0	0	1	0	0	0
26	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Затем для минимизации функций для $i1$, $i2$, $i3$, $i4$, $i5$, $i6$, $i7$ была использована функция SOPform:

```
import pandas as pd
from sympy import symbols
from sympy.logic.boolalg import Or, And

data = pd.read_csv("table_is.csv")

a, b, q1, q2, q3, q4 = symbols('a b q1 q2 q3 q4')
```

```
inputs = [a, b, q1, q2, q3, q4]

output_columns = ['i1', 'i2', 'i3', 'i4', 'i5', 'i6', 'i7']

expressions = {}

for output in output_columns:
    terms = []
    for _, row in data.iterrows():
        if row[output] == 1:
            term = And(*[var if row[var.name] == 1 else ~var for var in inputs])
            terms.append(term)
    expressions[output] = Or(*terms)
```

Таким образом для каждого i были получены минимизированные функции (Рис. 3).

```
expressions['i1']
```

$b \wedge q_4 \wedge \neg a \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3$

```
expressions['i2']
```

$b \wedge q_3 \wedge \neg a \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_4$

```
expressions['i3']
```

$b \wedge q_3 \wedge q_4 \wedge \neg a \wedge \neg q_1 \wedge \neg q_2$

```
expressions['i4']
```

$(a \wedge b \wedge q_2 \wedge q_4 \wedge \neg q_1 \wedge \neg q_3) \vee (a \wedge q_2 \wedge q_4 \wedge \neg b \wedge \neg q_1 \wedge \neg q_3) \vee (b \wedge q_2 \wedge q_4 \wedge \neg a \wedge \neg q_1 \wedge \neg q_3) \vee (b \wedge \neg a \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4)$

```
expressions['i5']
```

$(a \wedge q_1 \wedge \neg b \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4) \vee (a \wedge q_2 \wedge q_3 \wedge q_4 \wedge \neg b \wedge \neg q_1) \vee (a \wedge q_2 \wedge q_3 \wedge \neg b \wedge \neg q_1 \wedge \neg q_4)$

```
expressions['i6']
```

$(b \wedge q_1 \wedge \neg a \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4) \vee (b \wedge q_2 \wedge q_3 \wedge q_4 \wedge \neg a \wedge \neg q_1) \vee (b \wedge q_2 \wedge q_3 \wedge \neg a \wedge \neg q_1 \wedge \neg q_4)$

```
expressions['i7']
```

$(a \wedge b \wedge q_2 \wedge \neg q_1 \wedge \neg q_3 \wedge \neg q_4) \vee (a \wedge b \wedge q_3 \wedge q_4 \wedge \neg q_1 \wedge \neg q_2) \vee (a \wedge q_2 \wedge \neg b \wedge \neg q_1 \wedge \neg q_3 \wedge \neg q_4) \vee (a \wedge q_3 \wedge q_4 \wedge \neg b \wedge \neg q_1 \wedge \neg q_2) \vee (b \wedge q_2 \wedge \neg a \wedge \neg q_1 \wedge \neg q_3 \wedge \neg q_4)$

Рис. 3. Минимизированные функции для импульсных микрокоманд

Теперь можно дополнить логический блок F, в котором также будут описаны импульсные микрокоманды по известному входу и состоянию.

Функции потенциальных микрокоманд

Также была создана таблица истинности для потенциальных микрокоманд, которые зависят от текущего состояния автомата (Таблица 9).

Таблица 9. Зависимость потенциальных микрокоманд от состояний

	a	b	q1	q2	q3	q4	l1	l2	l3	l4	l5	l6	l7
0	1	0	0	0	0	0	1	1	0	0	0	0	0
1	0	1	0	0	0	0	1	1	0	0	0	0	0
2	0	1	0	0	0	1	1	0	0	0	0	0	0
3	1	0	0	0	0	1	1	0	0	0	0	0	0
4	1	1	0	0	0	1	1	0	0	0	0	0	0
5	0	1	0	0	1	0	0	1	0	0	0	0	0
6	1	0	0	0	1	0	0	1	0	0	0	0	0
7	1	1	0	0	1	0	0	1	0	0	0	0	0
8	0	1	0	0	1	1	0	0	1	0	0	0	0
9	1	0	0	0	1	1	0	0	1	0	0	0	0
10	1	1	0	0	1	1	0	0	1	0	0	0	0
11	0	1	0	1	0	0	0	0	0	1	0	0	0
12	1	0	0	1	0	0	0	0	0	1	0	0	0
13	1	1	0	1	0	0	0	0	0	1	0	0	0
14	0	1	0	1	0	1	0	0	0	0	0	1	0
15	1	0	0	1	0	1	0	0	0	0	0	1	0
16	1	1	0	1	0	1	0	0	0	0	0	1	0
17	1	1	0	0	0	0	1	1	0	0	0	0	0
18	1	1	0	1	1	0	0	0	0	0	1	0	0
19	1	0	0	1	1	0	0	0	0	0	1	0	0
20	0	1	0	1	1	0	0	0	0	0	1	0	0
21	0	1	0	1	1	1	0	0	0	0	1	0	0
22	1	0	0	1	1	1	0	0	0	0	1	0	0
23	1	1	0	1	1	1	0	0	0	0	1	0	0
24	0	1	1	0	0	0	0	0	0	0	1	0	0
25	1	0	1	0	0	0	0	0	0	0	1	0	0
26	1	1	1	0	0	0	0	0	0	0	1	0	0

Затем для минимизации функций для $l1, l2, l3, l4, l5, l6$ была использована функция SOPform:

```
data2 = pd.read_csv("table_ls.csv")

q1, q2, q3, q4 = symbols('q1 q2 q3 q4')
inputs = [q1, q2, q3, q4]

output_columns = ['l1', 'l2', 'l3', 'l4', 'l5', 'l6']
```

```
7
8  expressions = {}
9
10 for output in output_columns:
11     terms = []
12     for _, row in data2.iterrows():
13         if row[output] == 1:
14             term = And(*[var if row[var.name] == 1 else ~var for var in inputs])
15             terms.append(term)
16     expressions[output] = Or(*terms)
```

Таким образом для каждого l были получены минимизированные функции (Рис. 4).

```
expressions['l1']
```

$(q_4 \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_3) \vee (\neg q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4)$

```
expressions['l2']
```

$(q_3 \wedge \neg q_1 \wedge \neg q_2 \wedge \neg q_4) \vee (\neg q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4)$

```
expressions['l3']
```

$q_3 \wedge q_4 \wedge \neg q_1 \wedge \neg q_2$

```
expressions['l4']
```

$q_2 \wedge \neg q_1 \wedge \neg q_3 \wedge \neg q_4$

```
expressions['l5']
```

$(q_1 \wedge \neg q_2 \wedge \neg q_3 \wedge \neg q_4) \vee (q_2 \wedge q_3 \wedge q_4 \wedge \neg q_1) \vee (q_2 \wedge q_3 \wedge \neg q_1 \wedge \neg q_4)$

```
expressions['l6']
```

$q_2 \wedge q_4 \wedge \neg q_1 \wedge \neg q_3$

Рис. 4. Минимизированные функции для потенциальных микрокоманд

Теперь можно создать логический блок FL, в котором будут описаны потенциальные микрокоманды по известному состоянию.

4 Особенности реализации

Для реализации логической схемы используется инструмент моделирования электрических схем Logisim [6].

Проект логической схемы электронных часов был разбит на 6 связанных между собой схем.

- **main** - главная схема, объединяющая в себе все остальные.
- **multiplexer** - схема с мультиплексером, преобразующим сигнал с счетчика в сигнал для семисегментного индикатора.
- **automata_main** - схема, описывающая блок F переходов состояний и импульсных микрокоманд.
- **automata_states** - схема, описывающая блок FL потенциальных микрокоманд.
- **stopwatch** - схема, описывающая работу секундомера.
- **signal** - схема, включающая сигнал.

4.1 Схема multiplexer

Данная схема преобразует 4 битовый сигнал поступающий с счетчика в сигналы для входов 7-сегментного индикатора (Рис. 5).

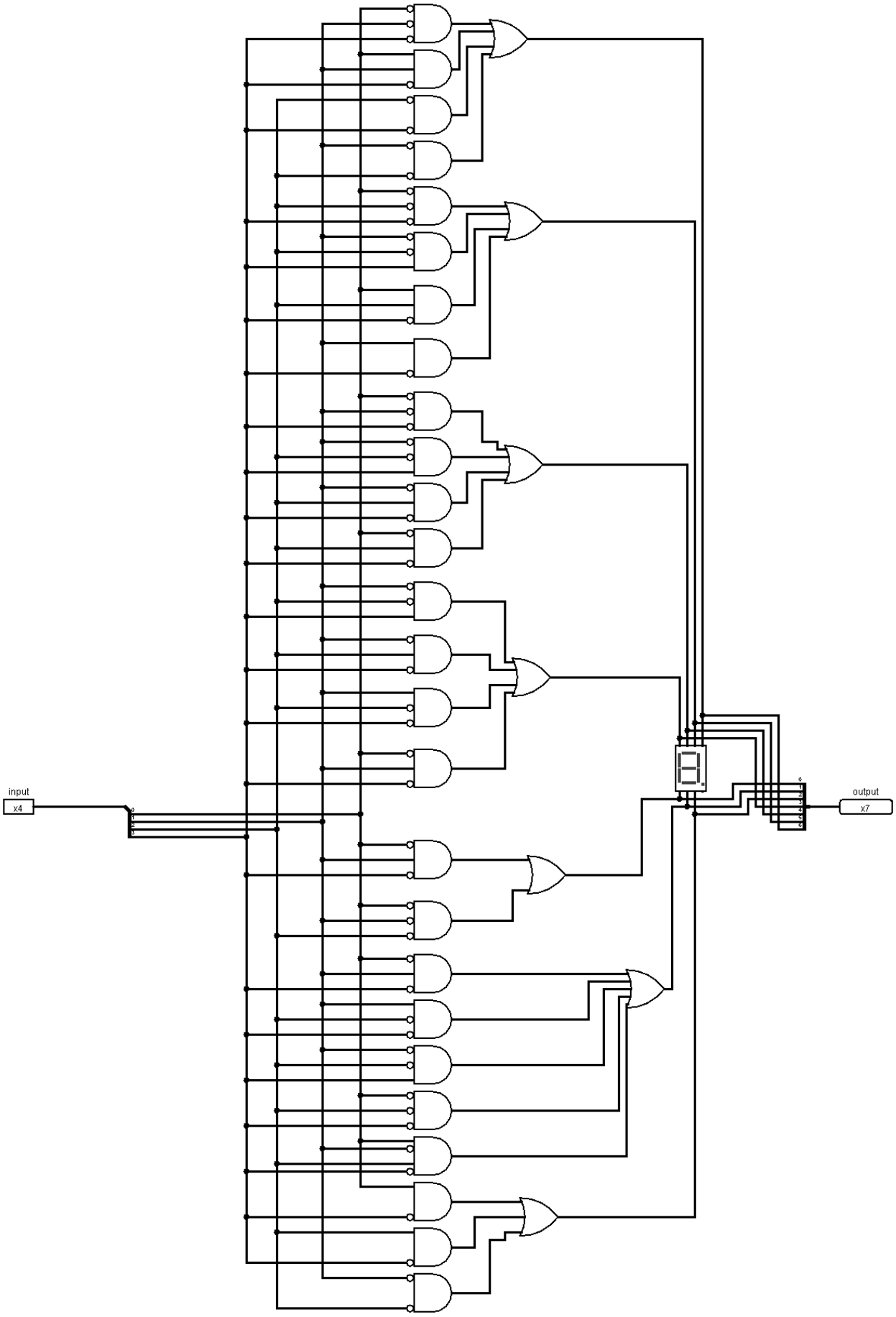

Рис. 5. Схема multiplexer

Схема состоит из одного 4 битового входа, элементов И, ИЛИ, 7-сегментного индикатора для отладки и 7 битового выхода.

4.2 Схема automata_main

Данная схема представляет блок F, она принимает сигналы поступающий с кнопок и выходов этой же схемы и возвращает новые сигналы состояний и импульсные микрокоманды (Рис. 6) согласно минимизированным функциям перехода состояний и функциям импульсных микрокоманд.

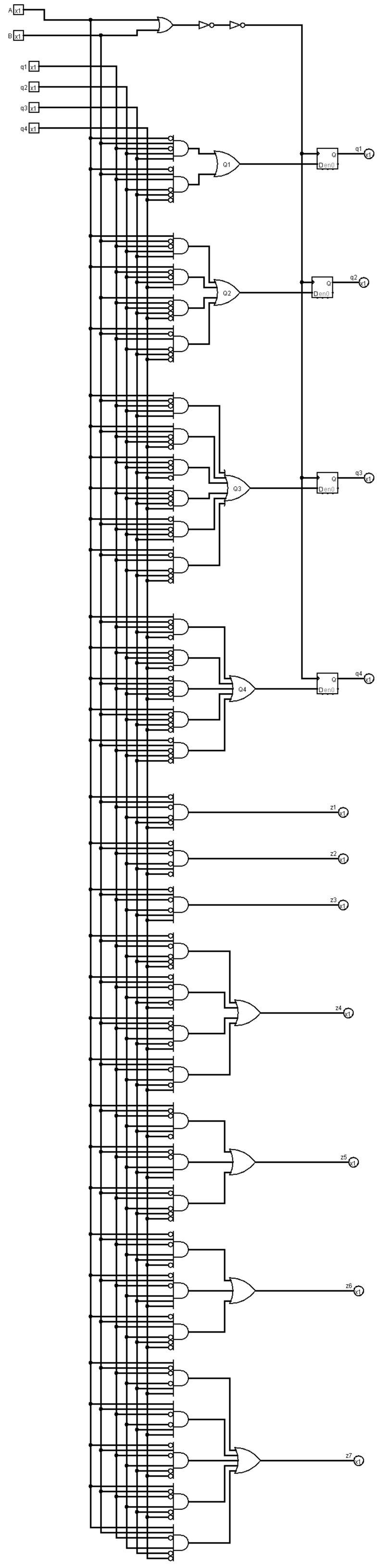

Рис. 6: Схема automata_main

Схема состоит из шести 1 битовых входов, элементов И, ИЛИ, D-триггеров и одиннадцати 1 битовых выходов.

4.3 Схема automata_states

Данная схема представляет блок FL, она принимает сигналы состояний и возвращает сигналы потенциальных микрокоманд (Рис. 8) согласно минимизированным функциям потенциальных микрокоманд.

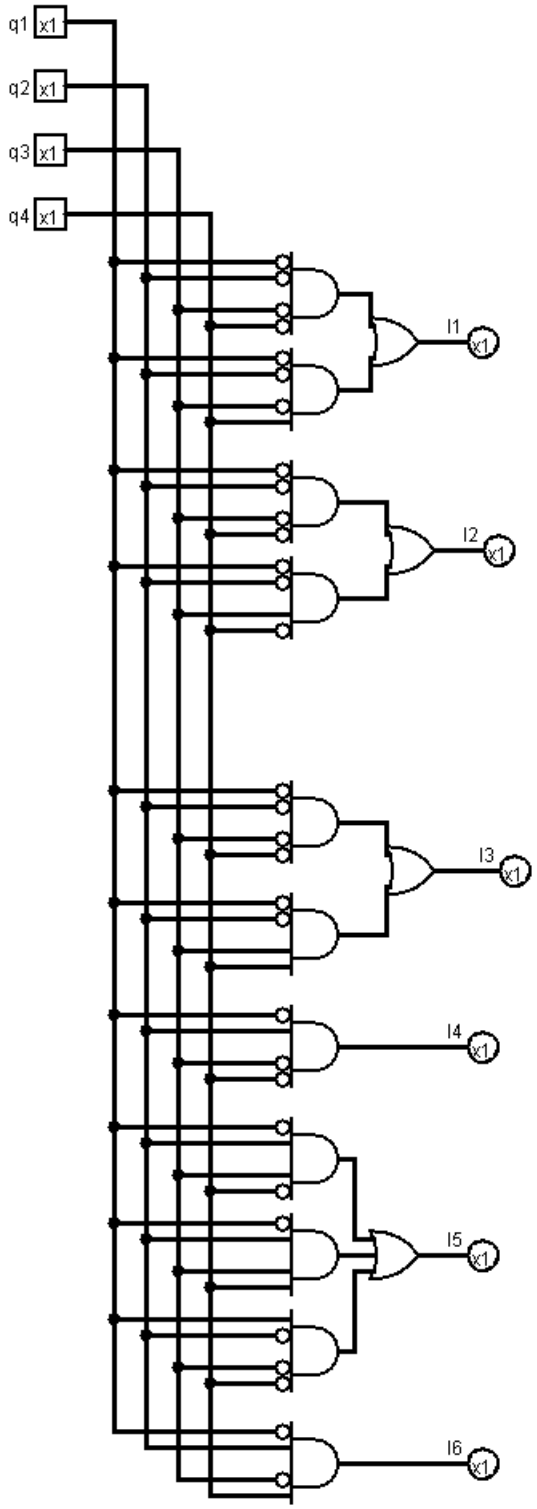

Рис. 8. Схема automata_states

Схема состоит из четырех 1 битовых входов, элементов И, ИЛИ и шести 1 битовых выходов.

4.4 Схема stopwatch

Данная схема представляет блок секундомера, она принимает сигналы разрешающего состояния, сброса, старта/паузы и генератора и возвращает четыре 4 битовых числа (Рис. 9).

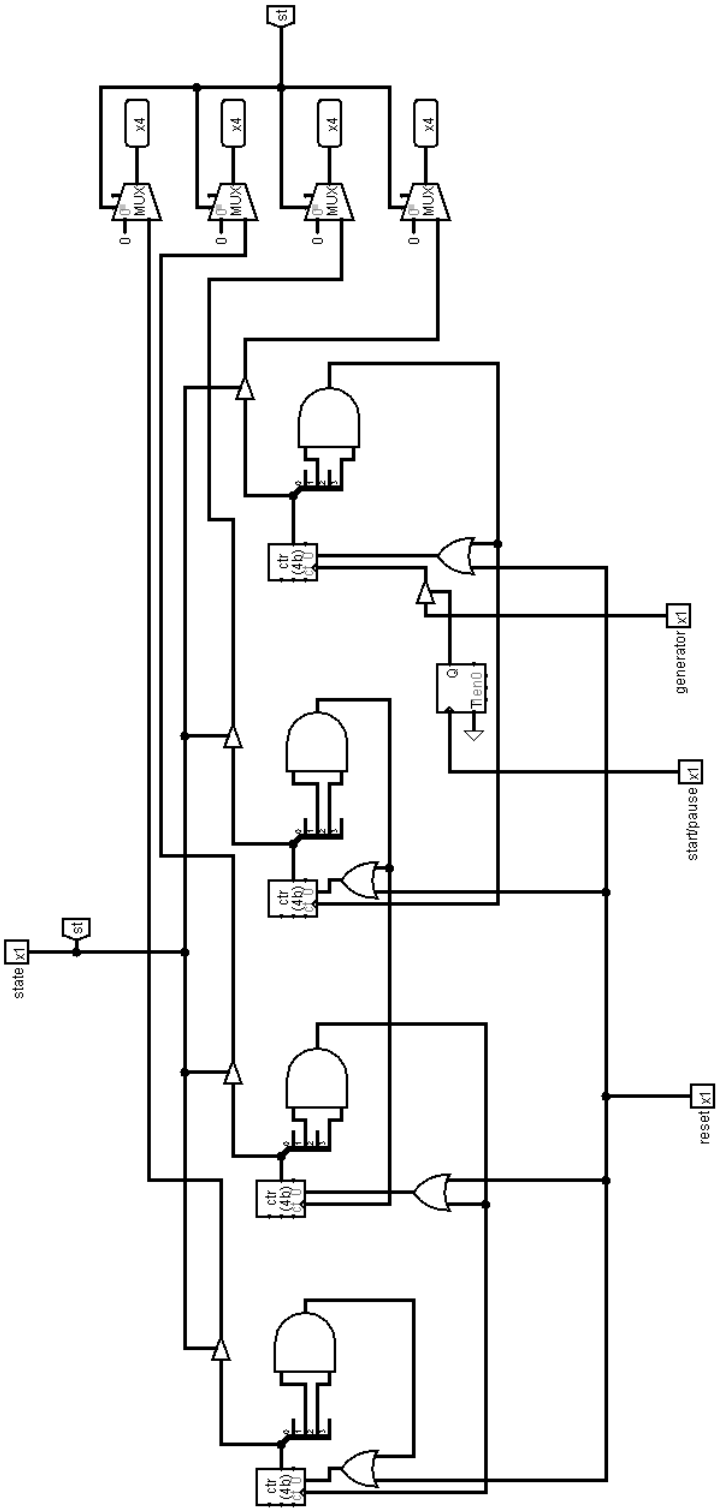

Рис. 9. Схема stopwatch

Схема состоит из четырех 1 битовых входов, элементов И, ИЛИ, счетчиков, мультиплексоров, D-триггера и четырех 4 битовых выходов.

4.5 Схема signal

Данная схема представляет блок сигнализации, она принимает сигналы счетчиков секунд и минут и возвращает сигнал, когда минуты равны (00), а секунды от (00) до (04) (Рис. 10).

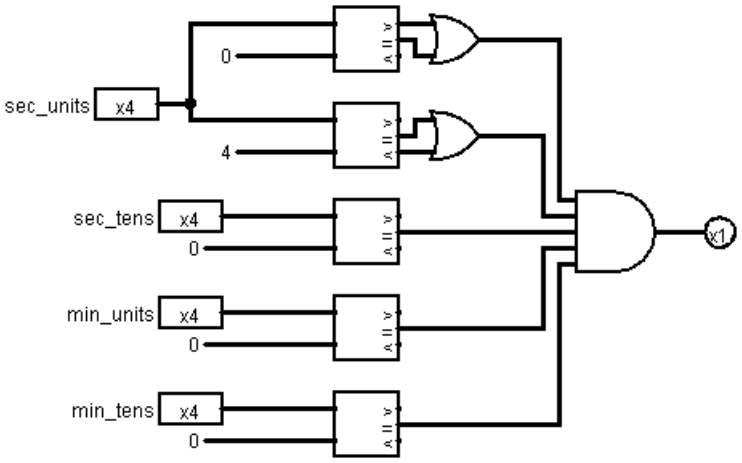

Рис. 10. Схема signal

Схема состоит из четырех 4 битовых входов, пяти констант, пяти компараторов, элементов И, ИЛИ и одного 1 битового выхода.

4.6 Схема main

Данная схема объединяет в себе все другие схемы (Рис. 10) и реализует полную функциональность электронных часов согласно заданию. Для отсчета времени используется один элемент тактового генератора. Тактовый генератор меняет значение на выходе по определённому расписанию. «Такт» - это единица времени в Logisim. По умолчанию частота тактового генератора равняется 1 Гц, что соответствует 1 секунде (1 такт в 1 секунду реального времени). Также Logisim позволяет делать частоту выше.

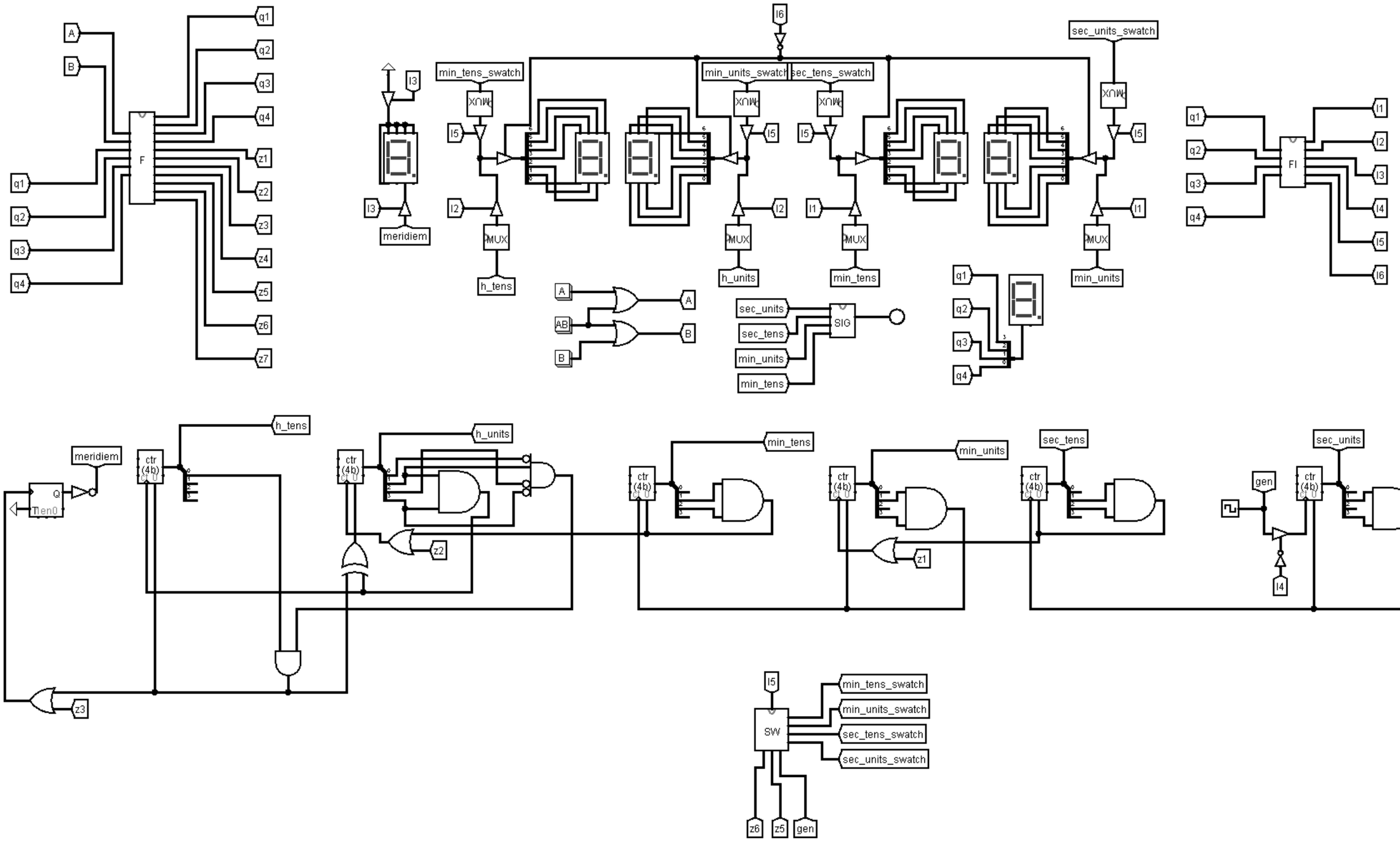

Рис. 10: Схема main

Основные компоненты схемы

- Верхняя часть схемы содержит систему отображения с несколькими 7-сегментными индикаторами
- В центре расположен блок управления с мультиплексорами (MUX)
- Нижняя часть содержит логику управления состояниями и сигналами

Функциональные блоки

Блок отображения времени

- Четыре группы 7-сегментных индикаторов для отображения часов и минут
- Мультиплексоры для переключения между режимами отображения

Блок управления (automata_main)

- Реализует логику переключения состояний
- Обрабатывает входные сигналы от кнопок
- Генерирует управляющие сигналы для других блоков

Секундомер (stopwatch)

- Отдельный блок для функции секундомера
- Имеет собственные счетчики и управление
- Поддерживает паузу и продолжение отсчета

Блок сигнализации (signal)

- Отслеживает время для подачи часового сигнала
- Генерирует 4-секундный звуковой сигнал каждый час

Управляющие сигналы

- min_lens_switch, sec_lens_switch - управление индикацией
- min_units_switch, sec_units_switch - переключение режимов
- h_lens, h_units - сигналы часов
- meridiem - индикация AM/PM для 12-часового формата

5 Расчет площади схемы

Для расчета площади необходимо оценить число транзисторов, которое потребуется для создания реальной схемы часов по построенной схеме. Используем следующую таблицу (Таблица ??):

Таблица 10. Количество транзисторов в элементах схемы

Элемент	Число транзисторов	Число элементов	Всего транзисторов
Элемент НЕ	1	5	5
Элемент И	4	84	336
Элемент ИЛИ	6	31	186
Элемент Исключающее ИЛИ	8	1	8
Управляемый буфер	6	20	120
Мультиплексор	8	4	32
Компаратор	4	5	20
D триггер	20	4	80
T триггер	20	2	40
Счётчик	64	10	640
Итог		166	1467

Для реализации схемы потребуется примерно 1467 транзистора. Производя оценку из расчета 1000 транзисторов на одном квадратном миллиметре площади кристалла, то площадь схемы примерно равна 1,47 mm^2.

Заключение

В ходе выполнения курсовой работы была разработана функциональная схема электронных часов в системе Logisim, полностью соответствующая заданным требованиям варианта (0 0 1 2 2 1 0). Реализованная схема обеспечивает следующий функционал:

- Корректное отображение времени в 12-часовом формате с индикацией AM/PM, включая возможность настройки текущего времени путем корректировки часов и минут
- Реализована система управления индикацией, позволяющая отключать отображение времени при необходимости
- Внедрен механизм автоматического останова часов после выполнения корректировки времени, что обеспечивает точность настройки
- Встроен полнофункциональный секундомер с возможностью фиксации промежуточного времени и продолжения отсчета
- Добавлена система звуковой сигнализации, активирующаяся каждый час и работающая в течение четырех секунд

Разработанная схема содержит 166 логических элементов, для реализации которых требуется 1467 транзисторов. Все компоненты схемы оптимально связаны между собой, обеспечивая надежное функционирование устройства и выполнение всех поставленных задач.

Согласно Таблице 7 количество использования кнопок для всего автомата одинаково. Нагрузка распределена между одиночными нажатиями и комбинированными нажатиями. В различных состояниях часов, например, в состоянии установки времени или состоянии секундомера, используются разные кнопки, что предотвращает концентрацию на одной кнопке. Из этого можно сделать вывод о том, что удалось добиться балансировки нажатия на кнопки, это позволяет сделать износ кнопок более равномерным.

Преимущества

- Благодаря проведенной минимизации функций для блоков управления и формирования сигналов удалось оптимизированно использовать логические элементы, что снижает общую сложность схемы.
- Все блоки вынесены в отдельные схемы, что позволяет разгрузить схему main и упростить восприятие.
- Компактная реализация звуковой сигнализации с фиксированной длительностью сигнала.

Недостатки

- При корректировке времени возможно только увеличение значений, что требует полного цикла прокрутки для установки меньших значений.
- При корректировке нельзя вернуться к предыдущему состоянию корректировки.
- Использование общего тактового генератора для часов и секундомера может приводить к незначительной погрешности (до 1 секунды) при работе секундомера.
- Ограниченное количество органов управления усложняет навигацию по меню и может создавать неудобства при частом использовании.

Возможности масштабирования Проект может быть расширен следующими функциональными возможностями:

1. Добавление отображения даты и календаря.
2. Реализация переключения между 12-часовым и 24-часовым форматами отображения времени.
3. Расширение функционала звуковой сигнализации:
    - Настройка громкости сигнала;
    - Выбор различных мелодий;
    - Возможность полного отключения;
4. Добавление будильника с настраиваемым временем срабатывания.

В результате проведенной работы создано полноценное устройство, которое может быть использовано как основа для реальной реализации электронных часов с расширенным функционалом.

Список литературы

[1] Айзерман М. А., Гусев Л. А., Розоноэр Л. И., Смирнова И. М., Таль А. А. Логика. Автоматы. Алгоритмы. — М.: Государственное издательство физико-математической литературы, 1963. — 556 с.

[2] Хопкрофт Дж. Э., Мотвани Р., Ульман Д. Дж. Введение в теорию автоматов, языков и вычислений. 3-е изд. — Рединг: Аддисон-Уэсли, 2006 [1979]. — ISBN 0-321-45536-3.

[3] Карпов Ю. Г. Синтез дискретных устройств управления. — СПб., 1997. — 19 с.

[4] SymPy Documentation. Logic — SymPy 1.12 documentation. URL: https://docs.sympy.org/latest/modules/logic.html (дата обращения: 13.12.2024).

[5] Edward, J., McCluskey. Minimization of Boolean functions // Bell System Technical Journal. 1956. Т. 35, № 6. С. 1417–1444. DOI: 10.1002/J.1538-7305.1956.TB03835.X.

[6] Logisim. URL: http://www.cburch.com/logisim/ru/index.html (дата обращения: 13.12.2024).