Факультет инфокоммуникационных сетей и систем
Кафедра программной инженерии и вычислительной техники
Березин А. Ю., Резников Б. К.

Вычислительная и микропроцессорная техника

M27 ИМИТАТОР УСТРОЙСТВА В ВИДЕ КОНЕЧНОГО АВТОМАТА

ъыъ.рф/aeЬIA

Задание на проектирование

Реализовать устройство-имитатор (в дальнейшем "устройство") на языке Verilog в среде Quartus под SoC Intel Cyclone V CSEMA5F31C6.

- Определить свойства, состояния и события устройства и также сигналы управления устройством;
- Определить состояния и события устройства как конечного автомата;
- Изобразить граф конечного автомата;
- Разработать таблицы состояний и событий устройства;
- Реализовать схемы КЦУ;
- Описать конечный автомат на языке Verilog
  - Структурным способом
  - Поведенческим способом
- Описать драйвер устройства вывода информации в ASCII формате о состояниях имитатора;
- Описать top-level модуль, реализующий имитатор устройства;
- Произвести тестирование устройства
  - Предоставить функциональные (временные) диаграммы как top-level модуля, так и описанных модулей устройства;
  - Произвести анализ полученных временных диаграмм;
  - Описать логику работы реализованного устройства;
- *Определить предельную частоту работы устройства*;

Общая теория конечных автоматов

Проектируемое устройство можно представить в виде так называемого конечного автомата (Finite State Machine - автомат с конечным количеством состояний - в зарубежной литературе). Конечный автомат представляет собой цифровое устройство, обладающее некоторыми **свойствами**, комбинации значений которых образуют **состояния**, а также **событиями**, соответствующими этим состояниям тем или иным образом.

Конечный автомат как цифровое устройство можно представить в виде трех схем: **устройство хранения**, **устройство, реализующее смену состояний в зависимости от предыдущих состояний**, **устройство, реализующее соответствие между состояниями и событиями**.

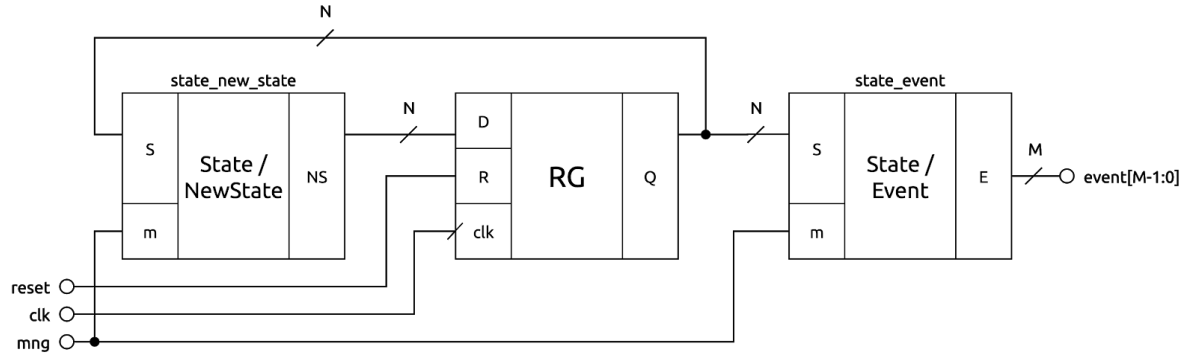


Рисунок 1 - Структурная схема конечного автомата (Mealy)

У конечного автомата, как и у других схем, есть входные и выходные сигналы. В качестве входных стоит выделить сигналы тактирования и управления, в качестве выходных - события. В зависимости о того, как **связаны сигналы управления и события** конечные автоматы подразделяются на два типа: автоматы **Mealy** и автоматы **Mooге**.

Данная зависимость реализуется в соответствующей схеме, которая является комбинационным цифровым устройством. В автомате типа Mealy **события зависят не только от состояний, но и от сигналов управления**. В автомате типа Mooге **события от сигналов управления не зависят, каждое событие соответствует только одному состоянию**.

Продолжая рассматривать схему конечного автомата, остановимся на схеме, реализующей **переходы между состояниями конечного автомата**. Данная схема является комбинационным цифровым устройством, входные сигналы которого -- сигналы управления общей схемы и сигнал, несущий информацию о текущем состоянии. Комбинация этих сигналов образует информацию следующем состоянии. Таким образом видно, что состояние в текущий момент времени определяется предыдущим состоянием и сигналом управления.

Третья схема представляет собой **устройство хранения информации** (параллельный регистр). Разрядность регистра выбирается необходимой и

достаточной, чтобы охватить все состояния. Входными сигналами регистра являются сигнал тактирования и сигнал, приходящий из схемы, реализующей переходы между состояниями. Выходной сигнал - сигнал, несущий информацию о текущем состоянии. По цепи обратной связи данный сигнал подается на устройство, реализующее смену состояний, где происходит образование информации о следующем состоянии. Эта информация записывается в регистр по сигналу синхронизации. Информация о текущем состоянии также подаётся на схему соответствия состояний и событий.

Таким образом, текущее **состояние** конечного автомата определяется содержимым регистра, а соответствующее **событие** - выходным сигналом конечного автомата.

Варианты работы

1 Дорожный светофор
2 Легковой автомобиль
3 Лифт
4 Башенный кран
5 Электронный секундомер
6 Радиоприемник
7 Автоматически регулируемый железнодорожный переезд
8 Цифровой вольтметр
9 Цифровой фотоаппарат
10 Кондиционер
11 Электрический чайник
12 Автомат по продаже кофе
13 Сотовый телефон
14 Телевизор
15 Монитор
16 Калькулятор
17 Банкомат
18 Магнитофон
19 Плеер
20 Автоматическая стиральная машина
21 Телефон-факс
22 Накопитель DVD-RW
23 Поезд метро
24 Модем
25 Источник бесперебойного питания
26 Клавиатура
27 Лазерный принтер
28 Цветной струйный принтер

29 Холодильник
30 Тестер
31 Ноутбук
32 Электронная кофеварка
33 Электронный дверной замок
34 СВЧ-печь

Анализ устройства

Свойства и состояния

В этом разделе рассматривается, **какие свойства и состояния есть у устройства**. Выбрать 3-5 свойств, 7-16 состояний.

Например, для электрических устройств существует свойство “Питание”, значения которого 0 или 1. Питание = 0, состояние - отключено.

В соответствии с выбранными свойствами и состояниями составить таблицу соответствия свойств и состояний. Выявить необходимые состояния устройства и представить их в виде таблицы следующего вида (табл. 1).

Состояния следует называть адекватно - в соответствии с логикой работы, например, START, IDLE, WORKING, OFF и т. д.

Таблица 1. Пример таблицы соответствия свойств и состояний.

Состояние	**Свойства и их значения**			
	Свойство_1	**Свойство_2**	**...**	**Свойство_M**
1 Состояние_1	0	0	...	0
2 Состояние_2	1	0	...	1
...	...	...	...	...
M Состояние_M	1	1	...	0

Граф конечного автомата

Переходы между состояний можно описывать разным способом. Удобнее всего делать это при помощи графа. Пример графа представлен на рис. 2.

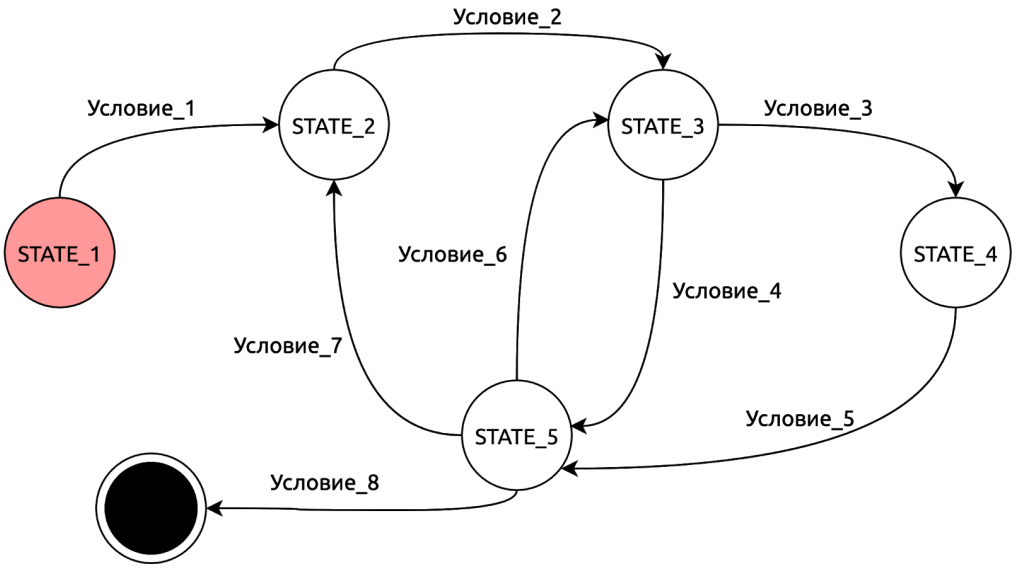


Рисунок 2 - Граф состояний

Разработка таблиц состояний и событий

В соответствии с разработанной таблицей соответствия состояний и свойств необходимо составить таблицу нумерации состояний (двоичный код) для записи состояний в регистр.

Таблица 2. Пример таблицы соответствия свойств и двоичных кодов

Состояние	**Код состояния**
1 Состояние_1	000_{bin}
2 Состояние_2	001_{bin}
...	...
M Состояние_M	111_{bin}

Каждому состоянию соответствует одно или более событий. Характер этой зависимости определяется во втором КЦУ. Прежде чем приступить к его синтезу, необходимо определить, какие события будут соответствовать каким состояниям (табл 3).

Таблица 3. Пример таблицы соответствия свойств и событий

Состояние	**Управляющий сигнал**			
	mng[0] = 0 mng[1] = 1 ...	**mng[0] = 1 mng[1] = 0 ...**	**...**	**mng[0] = 1 mng[1] = 1 ...**
1 Состояние_1	Событие_11	Событие_21	...	Событие_N1

2 Состояние_2	Событие_12	Событие_22	...	Событие_N2
...	...	...	...	...
M Состояние_M	Событие_1M	Событие_2M	...	Событие_NM

В соответствии с разработанной таблицей соответствия состояний и событий необходимо составить таблицу нумерации событий (двоичный код). Пример таблицы соответсвия событий и их двоичных кодов приведен ниже (табл. 4).

Таблица 4. Пример таблицы соответствия свойств и событий

Событие	**Код события**
1 Событие_01	000_{bin}
2 Событие_02	001_{bin}
...	...
M Событие_NM	111_{bin}

Проектирование и реализация устройства

Электрическая схема КЦУ, реализующая переходы между состояниями

Приступая к синтезу КЦУ, необходимо составить таблицу его функционирования. Таблица выглядит следующим образом (табл. 5).

Таблица 5. Пример таблицы функционирования КЦУ переключения состояний

Текущее состояние				**Сигналы управления**			**Следующее состояние**			
Название	**Входные сигналы**						**Выходные сигналы**			**Название**
	S[2]	**S[1]**	**S[0]**	**mng [2]**	**mng [1]**	**mng [0]**	**NS[2]**	**NS[1]**	**NS[0]**	
Состояние_N	1 / 0	1 / 0	1 / 0	0	1 / 0	1 / 0	0	0	0	**Выключено**
Состояние_1	0	0	0	1	1 / 0	1 / 0	0	0	1	**Состояние_2**
Состояние_2	0	0	1	1	0	0	0	0	1	**Состояние_2** (текущее)
	0	0	1	1	1	0	0	1	0	**Состояние_3** (след.)
	0	0	1	1	0	1	1	1	0	**Состояние_M** (макс.)

	0	0	1	1	1	1	0	0	1	**Состояние_2** (текущее)

Один из удобных вариантов получения электрической схемы - использование карт Карно. Карта Карно разрабатывается для каждого выходного сигнала в отдельности. В соответствии с картой Карно выводится логическое выражение для данного выходного сигнала и строится электрическая принципиальная (логическая) схема КЦУ (см Приложение).

Электрическая схема КЦУ, реализующая соответствие состояний и событий

Данная схема показывает, как связаны состояния и события. События, кроме как от состояний, могут зависеть и от сигналов управления. В этом случае автомат является автоматом типа **Mealy**, иначе **Moore**.

Таблица выглядит следующим образом (табл. 4).

Таблица 4. Пример таблицы функционирования КЦУ соответствия состояний и событий

Состояние				**Сигналы управления**		**События**					
Состояние	**Входные сигналы**					**Выходные сигналы**					**Событие**
	S[2]	**S[1]**	**S[0]**	**mng [0]**	**mng [4]**	**E[4]**	**E[3]**	**E[2]**	**E[1]**	**E[0]**	
Состояние_1	0	0	0	1 / 0	1 / 0	0	0	0	0	0	**Событие_1**
Состояние_2	0	0	1	0	0	0	0	0	0	1	**Событие_2**
	0	0	1	0	1	0	0	0	1	0	...
	0	0	1	1	0	0	0	0	1	1	
	0	0	1	1	1	0	0	1	0	0	**Событие_3**

Электрическая схема строится аналогичным предыдущему КЦУ образом.

Поведенческая реализация на языке Verilog

При поведенческой реализации конечный автомат описывается в виде одного модуля, в котором содержатся сведения о смене состояний и соответствии состояний и событий. Пример поведенческой реализации приведен в приложении. Состояния S* следует называть адекватно - в соответствии с логикой работы, например, START, IDLE, WORKING, OFF и т. д.

Объявление модуля, входных и выходных портов (сигнала синхронизации, сигнала управления, сигнала сброса, сигнала, отображающего события:

```
module four_state_mealy_state_machine
(
    input clk, in, reset,
```

```
      output reg [1:0] out
);
```

Объявление регистра, хранящего информацию:

```
reg         [1:0]       state = S0;
```

Объявление состояний и их наименований:

```
parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
```

Описание переходов между состояниями:

```
      always @ (posedge clk or posedge reset) begin
            if (reset)
                  state <= S0;
            else
                  case (state)
                        S0:
                              if (in)
                              begin
                                    state <= S1;
                              end
                              else
                              begin
                                    state <= S1;
                              end
```

Описание соответствий состояний и событий:

```
      always @ (state or in) begin
                        case (state)
                              S0:
                                    if (in)
                                    begin
                                          out = 2'b00;
                                    end
                                    else
                                    begin
                                          out = 2'b10;
                                    end
```

Структурная реализация

При структурной реализации конечный автомат описывается в виде трех схем: регистра и двух КЦУ, первое из которых отвечает за смену состояний, а второе -- за соответствие состояний и событий (рис. 1).

Описание КЦУ смены состояний на языке Verilog

КЦУ, отвечающее за смену состояний описывается при помощи обычной логики КЦУ (см. M1).

Это можно сделать двумя способами.

1 Оператор assign, с помощью логических уравнений:

```
assign NS[0] =
assign NS[1] =
```

2 Оператор case, с помощью таблицы функционирования:

```
always @(S, mng) begin
            case ({S, mng})
                  5'b00000:   NS = ...
                  5'b00001:   NS = ...

                  default:    NS = ...
            endcase
end
```

После программирования КЦУ (с помощью логических уравнений через оператор assign) необходимо проверить их корректность по временным диаграммам - соответствие таблице функционирования.

Описание КЦУ соответствия состояний и событий на языке Verilog

Описание КЦУ соответствия состояний и событий производится аналогичным предыдущему КЦУ образом.

После программирования КЦУ (с помощью логических уравнений через оператор assign) необходимо проверить их корректность по временным диаграммам - соответствие таблице функционирования.

Описание регистра

Параллельный регистр описывается способом, указанным в M25. Разрядность регистра выбирается таковой, чтобы охватить все состояния. Например, если состояний 19, следует выбрать разрядность 5.

Описание драйвера вывода информации о состоянии устройства на языке Verilog

Показ состояния устройства должен быть реализован в виде ASCII символов на временной диаграмме. Информация о ASCII таблице приведена в Приложении.

В программном пакете Quartus поддерживается только символы ASCII-кода от 33 до 126 включительно. Следовательно, вся информация должна быть представлена с использованием только этих символов. Если необходимого символа нет в этой таблице, то необходимо подобрать похожий на него по смыслу или обозначению.

Для того чтобы отображать ASCII-символы необходимо реализовать 8-битные шины под каждый знак, например, чтобы отобразить значение температуры 70 °C необходимо 4 шины: первая для отображения цифры 7, вторая - для цифры 0, третья - для знака ° и четвертая - для знака C. В итоге на выходе получается шина разрядностью [8*4-1:0], то есть 28-разрядная.

```
output reg [8*4-1:0] ASCII_temp // выходной регистр
```

По ASCII таблице:

- 7 - это 8'd55
- 1 - это 8'd49
- ° - этот символ не включен в коды с 33 по 126, используем, например, символ % - 8’d37
- C - это 8'd67

Для того чтобы вывести строку 71°C необходимо записать данные символы по порядку в регистр:

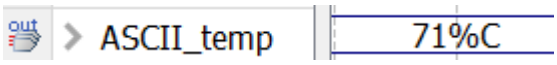
```
ASCII_temp = {8'd55, 8'd49, 8'd37, 8'd67};
```

Если для примера представить эту строчку в символы - будет выглядеть так

```
ASCII_temp = {7, 1, %, C};
```

Таким образом записываем необходимые символы в выходной регистр.

На временных диаграмма необходимо включить отображение информации (Radix) в виде ASCII для соответствующего сигнала.



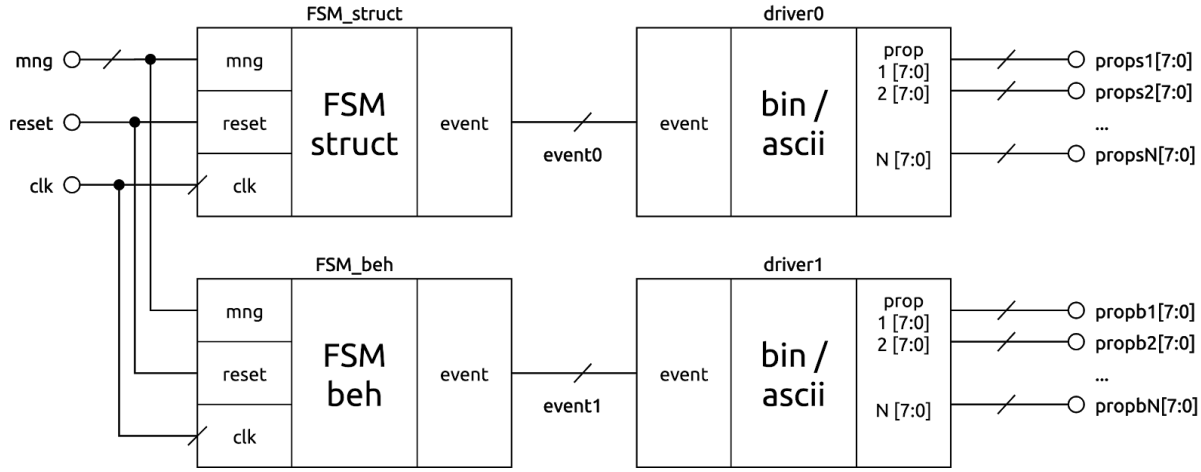
Сам драйвер ASCII кода может быть описан аналогично драйверу семисегментного индикатора. То есть преобразуем выходной сигнал с конечного автомата в отображаемую информацию, работая кодопреобразователем.

Каждая группа шин отображает состояние отдельного свойства. Например, четыре шины отображают температуру, одна шина - наличие питания и т. д.

Входом для драйвера служит шина event - событие. Необходимо описать соответствие кода события ASCII коду.

Описание объединения устройств на языке Verilog

Необходимо создать модуль, являющийся объединением FSM поведенческого и структурного описания.



Тестирование устройства

Необходимо произвести проверку устройства на выполнение заданной логики работы. Проанализировать полученные временные диаграммы с указанием примеров сигналов управление и сопоставить с анализом устройства.

Описать логику работы разработанного устройства

- Тестирование каждого отдельного модуля. Проанализировать полученные временные диаграммы с указанием примеров сигналов управления и сопоставить с данными полученными при анализе устройства и драйвера.
- Тестирование top-level модуля. Проанализировать полученные временные диаграммы с указанием примеров сигналов управления и сопоставить с логикой работы устройства.

Временные диаграммы должны быть читабельны, то есть значения отображаемые на диаграммах не должны сливаться, а если диаграмма получается достаточно протяженной, то разбивать ее на части.

Содержание отчета

- Задание на проектирование;
- Пояснительная записка:
    - описание всех пунктов анализа;
    - описание всех пунктов проектирования;

    - описание всех пунктов тестирования;
- Приложение - временные диаграммы функционирования (без пояснения);
- Приложение - схемы КЦУ по ГОСТ ЕСКД;
- Приложение - листинги кодов на языке Verilog.

ПРИЛОЖЕНИЕ А

ПРИМЕР АНАЛИЗА КОНЕЧНОГО АВТОМАТА

Устройство - электроплита

Свойства:

- питание;
- температура;

Состояния:

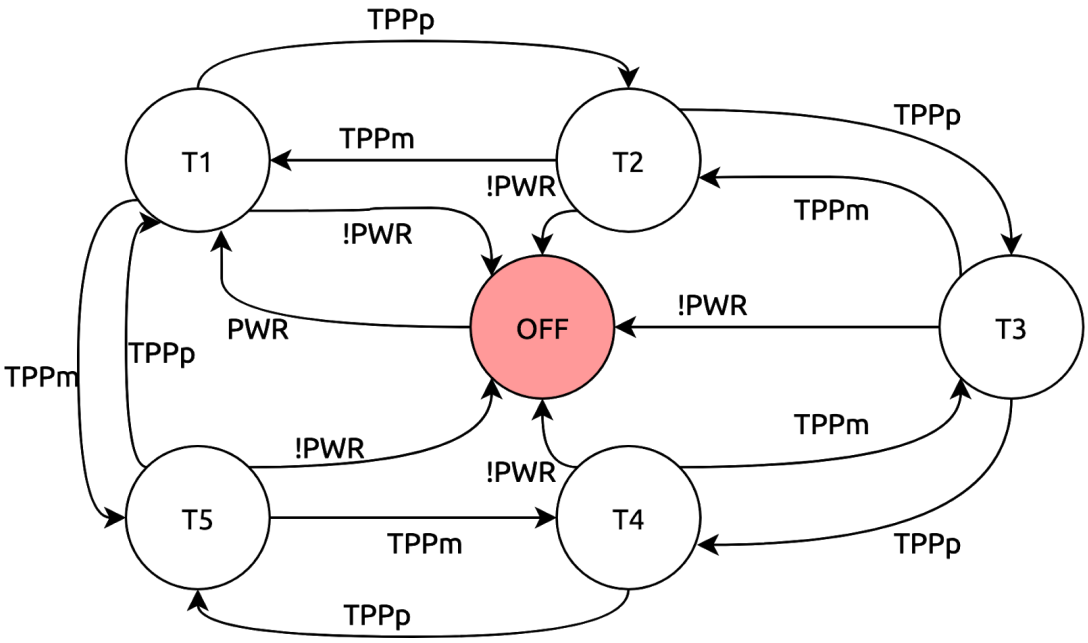
- питание: 0 или 1;
- температура плиты: 1, 2, 3, 4, 5, 6 (режимы);

События:

- Показание температуры;
- Показание гриля;
- Показание вентилятор;
- Показание питания

Таблица соответствия свойств и состояний

Состояние	**Свойства и их значения**	
	Питание	**Температура**
1 Выключено	0	0
2 T = 1	1	0
3 T = 2	1	1
4 T = 3	1	2
5 T = 4	1	3
6 T = 5	1	4
7 T = 6	1	5



Граф конечного автомата

Таблица соответствия свойств и двоичных кодов

Состояние	**Код состояния**
1 Выключено	000_{bin}
2 T = 1	001_{bin}
3 T = 2	010_{bin}
4 T = 3	011_{bin}
5 T = 4	100_{bin}
6 T = 5	101_{bin}
7 T = 6	110_{bin}

Таблица соответствия состояний и событий

Состояние	**Управляющий сигнал**			
	Гриль = 0 Вент. = 0	**Гриль = 1 Вент. = 0**	**Гриль = 0 Вент. = 1**	**Гриль = 1 Вент. = 1**
1 Выключено	Выключено	Выключено	Выключено	Выключено
2 T = 1	T = 1 Гриль выкл. Вент. выкл.	T = 1 Гриль вкл. Вент. выкл.	T = 1 Гриль выкл. Вент. вкл.	T = 1 Гриль вкл. Вент. вкл.
3 T = 2	T = 2 Гриль выкл. Вент. выкл.	T = 2 Гриль вкл. Вент. выкл.	T = 2 Гриль выкл. Вент. вкл.	T = 2 Гриль вкл. Вент. вкл.

4 T = 3	T = 3 Гриль выкл. Вент. выкл.	T = 3 Гриль вкл. Вент. выкл.	T = 3 Гриль выкл. Вент. вкл.	T = 3 Гриль вкл. Вент. вкл.
5 T = 4	T = 4 Гриль выкл. Вент. выкл.	T = 4 Гриль вкл. Вент. выкл.	T = 4 Гриль выкл. Вент. вкл.	T = 4 Гриль вкл. Вент. вкл.
6 T = 5	T = 5 Гриль выкл. Вент. выкл.	T = 5 Гриль вкл. Вент. выкл.	T = 5 Гриль выкл. Вент. вкл.	T = 5 Гриль вкл. Вент. вкл.
7 T = 6	T = 6 Гриль выкл. Вент. выкл.	T = 6 Гриль вкл. Вент. выкл.	T = 6 Гриль выкл. Вент. вкл.	T = 6 Гриль вкл. Вент. вкл.

Таблица соответствия событий и двоичных кодов.

Событие	**Код события**
1 Выключено	00000_{bin}
2 T = 1 Гриль выкл. Вент. выкл.	00001_{bin}
3 T = 2 Гриль выкл. Вент. выкл.	00010_{bin}
4 T = 3 Гриль выкл. Вент. выкл.	00011_{bin}
5 T = 4 Гриль выкл. Вент. выкл.	00100_{bin}
6 T = 5 Гриль выкл. Вент. выкл.	00101_{bin}
7 T = 6 Гриль выкл. Вент. выкл.	00110_{bin}
8 T = 1 Гриль вкл. Вент. выкл.	00111_{bin}
9 T = 2 Гриль вкл. Вент. выкл.	01000_{bin}
10 T = 3 Гриль вкл. Вент. выкл.	01001_{bin}
11 T = 4 Гриль вкл. Вент. выкл.	01010_{bin}
12 T = 5 Гриль вкл. Вент. выкл.	01011_{bin}
13 T = 6 Гриль вкл. Вент. выкл.	01100_{bin}
14 T = 1 Гриль выкл. Вент. вкл.	01101_{bin}
15 T = 2 Гриль выкл. Вент. вкл.	01110_{bin}
16 T = 3 Гриль выкл. Вент. вкл.	01111_{bin}
17 T = 4 Гриль выкл. Вент. вкл.	10000_{bin}

18 T = 5 Гриль выкл. Вент. вкл.	10001_{bin}
19 T = 6 Гриль выкл. Вент. вкл.	10010_{bin}
20 T = 1 Гриль вкл. Вент. вкл.	10011_{bin}
21 T = 2 Гриль вкл. Вент. вкл.	10100_{bin}
22 T = 3 Гриль вкл. Вент. вкл.	10101_{bin}
23 T = 4 Гриль вкл. Вент. вкл.	10110_{bin}
24 T = 5 Гриль вкл. Вент. вкл.	10111_{bin}
25 T = 6 Гриль вкл. Вент. вкл.	11000_{bin}

ПРИЛОЖЕНИЕ Б

ПРИМЕР ТАБЛИЦ СОСТОЯНИЙ И СОБЫТИЙ

Таблица функционирования КЦУ смены состояний

Текущее состояние				**Сигналы управления**			**Следующее состояние**			
Название	**Входные сигналы**						**Выходные сигналы**			**Название**
	S[2]	**S[1]**	**S[0]**	**PWR**	**TMPp**	**TMPm**	**NS[2]**	**NS[1]**	**NS[0]**	
Выкл.	1 / 0	1 / 0	1 / 0	0	1 / 0	1 / 0	0	0	0	**Выключено**
	0	0	0	1	1 / 0	1 / 0	0	0	1	T = 1
T = 1	0	0	1	1	0	0	0	0	1	T = 1 (текущее)
	0	0	1	1	1	0	0	1	0	T = 2 (след.)
	0	0	1	1	0	1	1	1	0	T = 6 (макс.)
	0	0	1	1	1	1	0	0	1	T = 1 (текущее)
T = 2	0	1	0	1	0	0	0	1	0	T = 2 (текущее)
	0	1	0	1	1	0	0	1	1	T = 3 (след.)
	0	1	0	1	0	1	0	0	1	T = 1 (пред.)
	0	1	0	1	1	1	0	1	0	T = 2 (текущее)
T = 3	0	1	1	1	0	0	0	1	1	T = 3 (текущее)
	0	1	1	1	1	0	1	0	0	T = 4 (след.)
	0	1	1	1	0	1	0	1	0	T = 2 (пред.)
	0	1	1	1	1	1	0	1	1	T = 3 (текущее)
T = 4	1	0	0	1	0	0	1	0	0	T = 4 (текущее)
	1	0	0	1	1	0	1	0	1	T = 5 (след.)
	1	0	0	1	0	1	0	1	1	T = 3 (пред.)
	1	0	0	1	1	1	1	0	0	T = 4 (текущее)
T = 5	1	0	1	1	0	0	1	0	1	T = 5 (текущее)
	1	0	1	1	1	0	1	1	0	T = 6 (след.)
	1	0	1	1	0	1	1	0	0	T = 4 (пред.)
	1	0	1	1	1	1	1	0	1	T = 5 (текущее)
T = 6	1	1	0	1	0	0	1	0	1	T = 6 (текущее)
	1	1	0	1	1	0	0	0	1	T = 0 (мин.)
	1	1	0	1	0	1	1	0	1	T = 5 (пред.)
	1	1	0	1	1	1	1	0	1	T = 6 (текущее)

Таблица функционирования КЦУ соответствия состояний и событий

Состояние				**Сигналы управления**		**События**					
Состояние	**Входные сигналы**					**Выходные сигналы**					**Событие**
	S[2]	**S[1]**	**S[0]**	**GR**	**CL**	**E[4]**	**E[3]**	**E[2]**	**E[1]**	**E[0]**	
Выкл.	0	0	0	1 / 0	1 / 0	0	0	0	0	0	Выкл.
T = 1	0	0	1	0	0	0	0	0	0	1	T = 1 Гриль выкл. Вент выкл
	0	0	1	0	1	0	0	0	1	0	...
	0	0	1	1	0	0	0	0	1	1	...
	0	0	1	1	1	0	0	1	0	0	
T = 2	0	1	0	0	0	0	0	1	0	1	
	0	1	0	0	1	0	0	1	1	0	
	0	1	0	1	0	0	0	1	1	1	
	0	1	0	1	1	0	1	0	0	0	
T = 3	0	1	1	0	0	0	1	0	0	1	
	0	1	1	0	1	0	1	0	1	0	
	0	1	1	1	0	0	1	0	1	1	
	0	1	1	1	1	0	1	1	0	0	
T = 4	1	0	0	0	0	0	1	1	0	1	
	1	0	0	0	1	0	1	1	1	0	
	1	0	0	1	0	0	1	1	1	1	
	1	0	0	1	1	1	0	0	0	0	
T = 5	1	0	1	0	0	1	0	0	0	1	
	1	0	1	0	1	1	0	0	1	0	
	1	0	1	1	0	1	0	0	1	1	
	1	0	1	1	1	1	0	1	0	0	
T = 6	1	1	0	0	0	1	0	1	0	1	
	1	1	0	0	1	1	0	1	1	0	
	1	1	0	1	0	1	0	1	1	1	
	1	1	0	1	1	1	1	0	0	0	

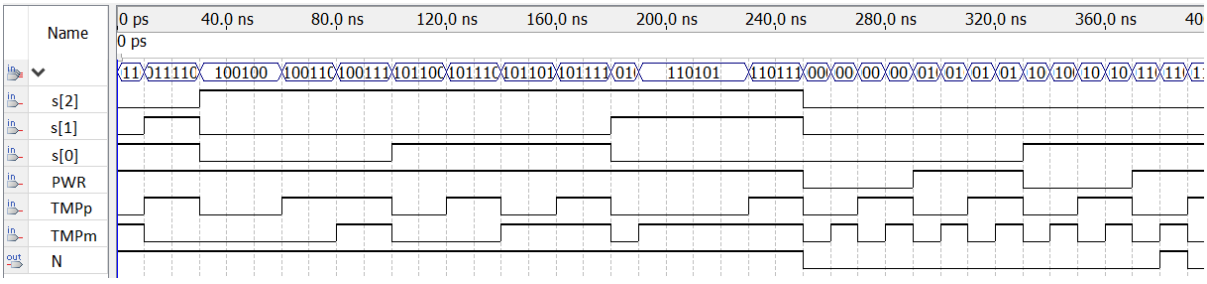
ПРИЛОЖЕНИЕ В

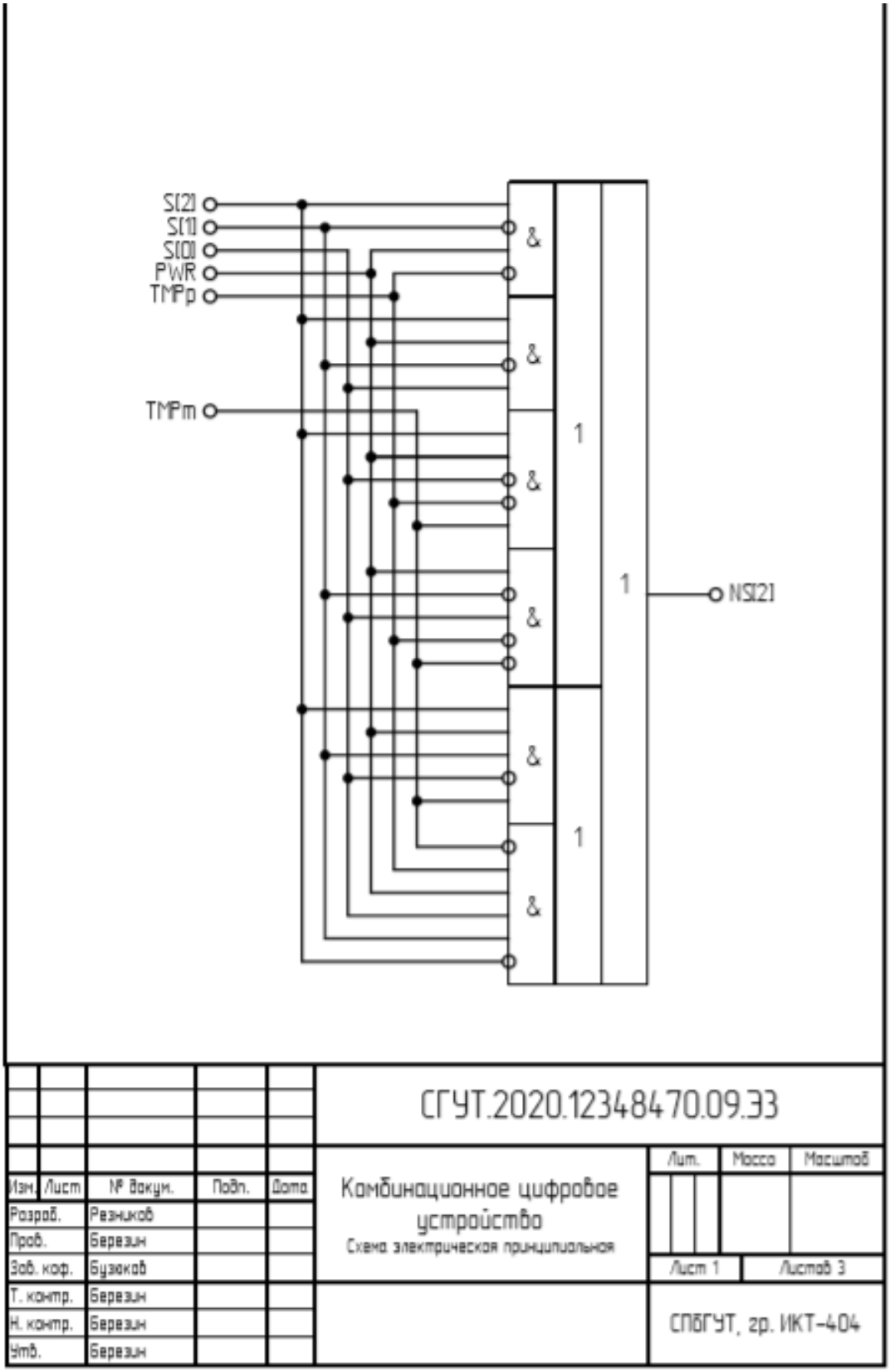
ПРИМЕР ЛОГИЧЕСКОЙ СХЕМЫ КЦУ

Для составления логической схемы КЦУ необходимо составить Карту Карно. Для значений 1 / 0 можем подставить любое удобное для свертки значение. Рассмотрим сигнал NS[2]. Вместо 1/0 подставим 0.

S[2]	S[1] S[0]	PWR							
		0				1			
		TMPp TMPm				TMPp TMPm			
		00	01	11	10	00	01	11	10
0	00	0	0	0	0	0	0	0	0
	01	0	0	0	0	0	**1**	0	0
	11	0	0	0	0	0	0	0	**1**
	10	0	0	0	0	0	0	0	0
1	00	0	0	0	0	**1**	0	**1**	**1**
	01	0	0	0	0	**1**	**1**	**1**	**1**
	11	0	0	0	0	0	0	0	0
	10	0	0	0	0	**1**	**1**	**1**	0

NS[2] = S[2] & PWR & !S[1] & TMPp
| S[2] & PWR & !S[1] & S[0]
| S[2] & PWR & !S[0] & !TMPp & !TMPm
| PWR & !S[1] & S[0] & !TMPp & TMPm
| S[2] & PWR & S[1] & !S[0] & TMPm
| !S[2] & PWR & S[1] & S[0] & TMPp & !TMPm.





					СГУТ.2020.12348470.09.33		
						Лит.	Масса	Масштаб
Изм.	Лист	№ докум.	Подп.	Дата	Комбинационное цифровое устройство Схема электрическая принципиальная			
Разраб.		Резников				Лист 1	Листов 3	
Пров.		Березин						
Зав. каф.		Бузаков						
Т. контр.		Березин				СПбГУТ, гр. ИКТ-404		
Н. контр.		Березин						
Утв.		Березин						

ПРИЛОЖЕНИЕ Г

ПРИМЕР ПОВЕДЕНЧЕСКОГО ОПИСАНИЕ FSM

```
// Quartus II Verilog Template
// 4-State Mealy state machine
// A Mealy machine has outputs that depend on both the state and
// the inputs.  When the inputs change, the outputs are updated
// immediately, without waiting for a clock edge.  The outputs
// can be written more than once per state or per clock cycle.

module four_state_mealy_state_machine
(
	input clk, in, reset,
	output reg [1:0] out
);

	// Declare state register
	reg		[1:0]  state  = S0;

	// Declare states
	parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;

	// Determine the next state synchronously, based on the
	// current state and the input
	always @ (posedge clk or posedge reset) begin
		if (reset)
			state <= S0;
		else
			case (state)
				S0:
					if (in)
					begin
						state <= S1;
					end
					else
					begin
						state <= S1;
					end
				S1:
					if (in)
```

```
                        begin
                              state <= S2;
                        end
                        else
                        begin
                              state <= S1;
                        end
                  S2:
                        if (in)
                        begin
                              state <= S3;
                        end
                        else
                        begin
                              state <= S1;
                        end
                  S3:
                        if (in)
                        begin
                              state <= S2;
                        end
                        else
                        begin
                              state <= S3;
                        end
            endcase
end

// Determine the output based only on the current state
// and the input (do not wait for a clock edge).
always @ (state or in)
begin
            case (state)
                  S0:
                        if (in)
                        begin
                              out = 2'b00;
                        end
                        else
                        begin
                              out = 2'b10;
```

```
                        end
                S1:
                        if (in)
                        begin
                                out = 2'b01;
                        end
                        else
                        begin
                                out = 2'b00;
                        end
                S2:
                        if (in)
                        begin
                                out = 2'b10;
                        end
                        else
                        begin
                                out = 2'b01;
                        end
                S3:
                        if (in)
                        begin
                                out = 2'b11;
                        end
                        else
                        begin
                                out = 2'b00;
                        end
        endcase
    end

endmodule
```

ПРИЛОЖЕНИЕ Д

ТАБЛИЦА СИМВОЛОВ ASCII

ASCII (American Standard Code for Information Interchange – Стандартный американский код обмена информацией) – это код для представления символов в виде чисел, в котором каждому символу сопоставлено число от 0 до 127. В большинстве компьютеров код ASCII используется для представления текста, что позволяет передавать данные от одного компьютера на другой. Стандартный набор символов ASCII использует только 7 битов для каждого символа. Добавление 8-го разряда позволяет увеличить количество кодов таблицы ASCII до 255. Коды от 128 до 255 представляют собой расширение таблицы ASCII. Эти коды используются для кодирования символов национальных алфавитов, а также символов псевдографики, которые можно использовать, например, для оформления в тексте различных рамок и текстовых таблиц.

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]