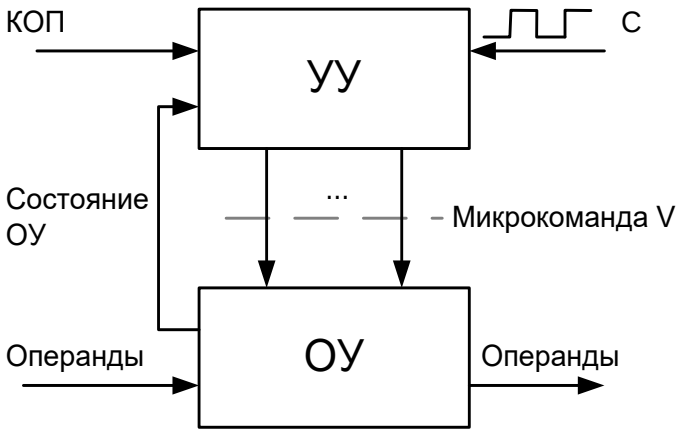
II.Устройства управления ЭВМ

- Принципы микропрограммного управления.
- Классификация устройств управления.
- Управляющие устройства с жесткой и программируемой логикой.
- Примеры реализации устройств управления на основе автоматов Мили и Мура.

Принципы микропрограммного управления



Любое цифровое устройство можно рассматривать, как совокупность операционного и управляющего блока.

Любая команда или последовательность команд реализуется в операционном блоке за несколько тактов

Последовательность сигналов управления должна выдаваться устройством управления в соответствии с поступающей на вход командой и текущим состоянием операционного блока

Состояние линий управления в каждом такте задает микрокоманду. Совокупность микрокоманд, необходимых для реализации команды, называется микропрограммой.

Устройство управления реализуется в виде автомата.

Классификация устройств управления:

По типу автомата:

- Автомат Мили.
- Автомат Мура.

По способу реализации:

- Устройство управления с жесткой логикой.

Функции выдачи сигналов управления и разделения во времени сигналов управления реализуются с помощью комбинационных схем и триггерной памяти.

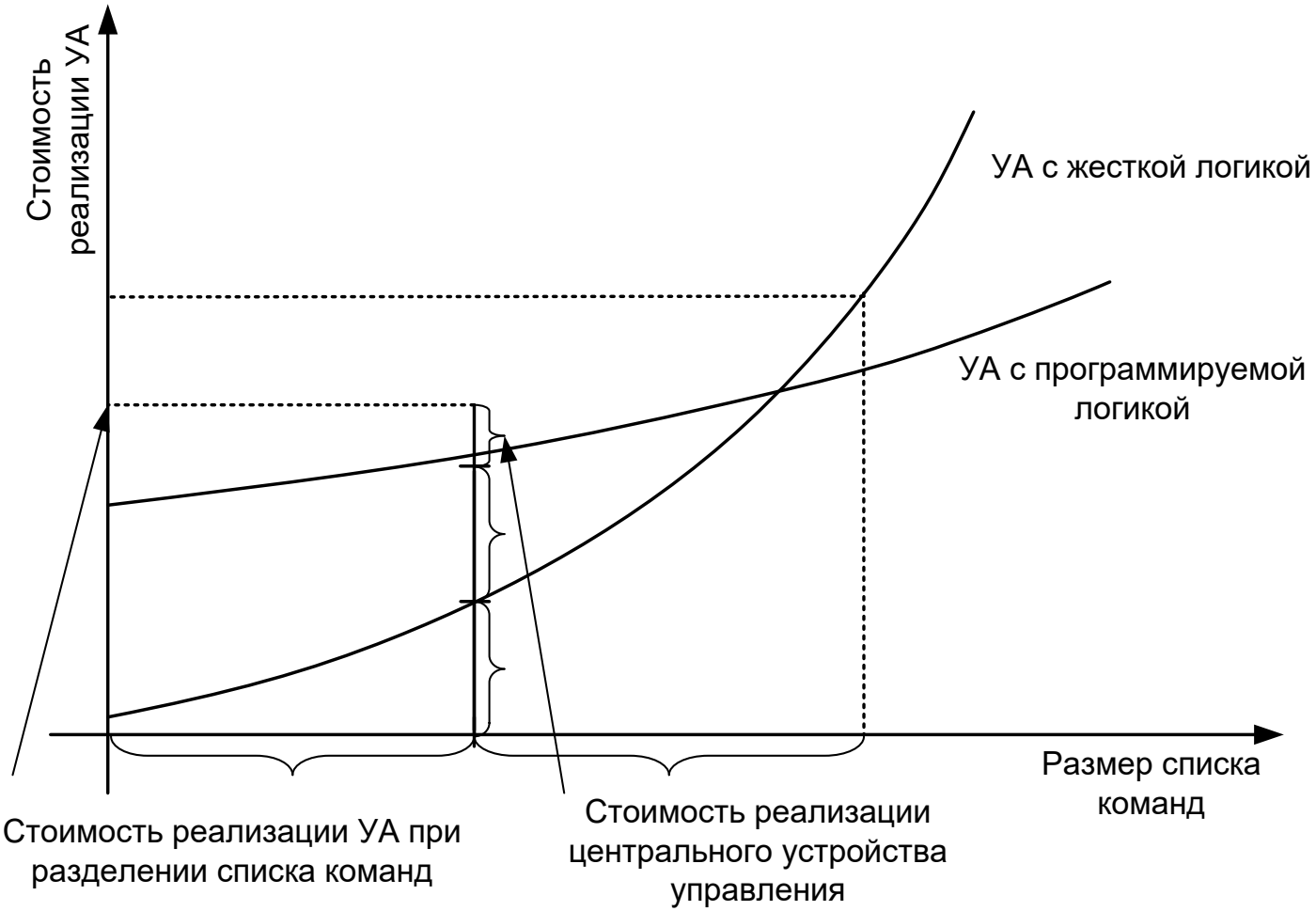
- Устройство управления с программируемой логикой.

Каждой выполняемой операции ставится в соответствие совокупность хранимых в памяти слов (микрокоманд), каждая из которых содержит информацию о микрооперациях, подлежащих исполнению в текущем такте.

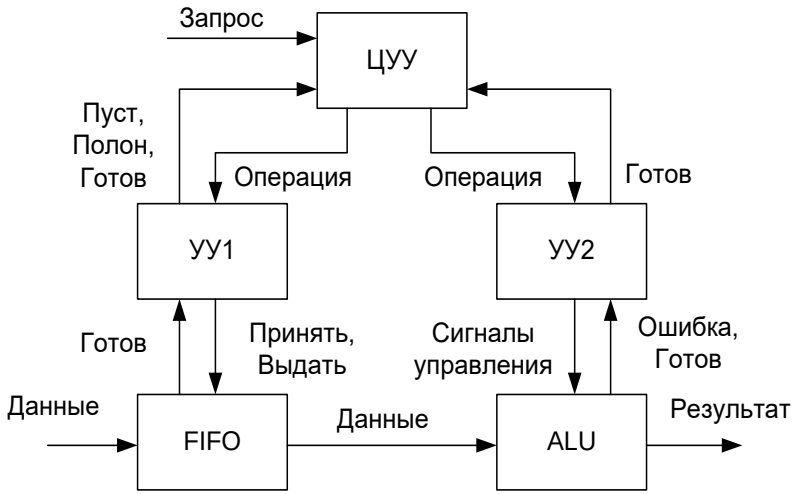
(+) Простота модификации и наращивания.

(-) Невысокое быстродействие для простых устройств.

Сравнение способов реализации УУ



Пример декомпозиции УУ



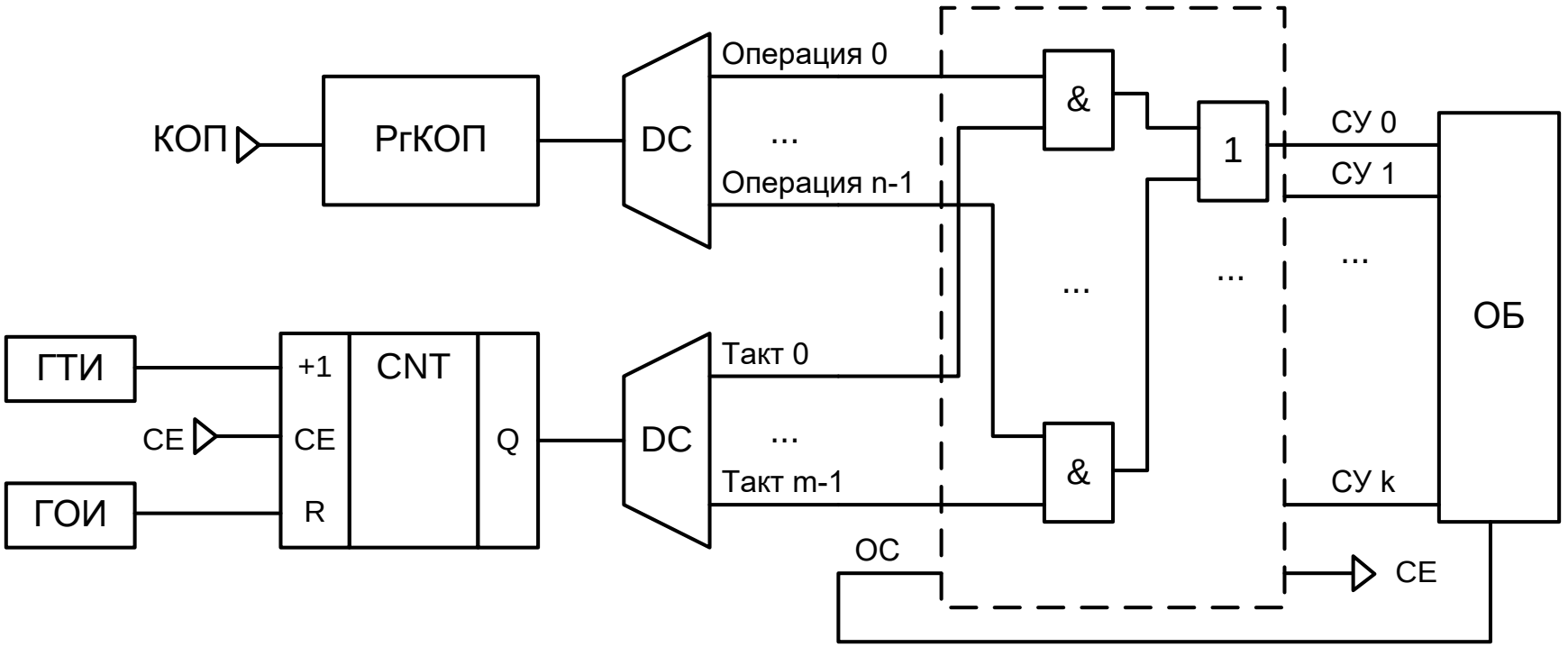
По способу кодирования микрокоманд:

- Минимальное кодирование (горизонтальное).
- Максимальное кодирование (вертикальное).
- Горизонтально-вертикальное кодирование.
- Вертикально-горизонтальное кодирование.
- Кодирование с помощью памяти нанокоманд.

По способу исполнения команд:

- Последовательные.
- Конвейерные.

Пример реализации управляющего автомата с жесткой логикой



Синтез управляющих автоматов с жесткой логикой

Исходные данные:

-Описание логики функционирования операционного блока (временные диаграммы, словесное описание, таблицы истинности, графы и т.д.).
-Сигналы управления.
-Осведомительные сигналы.
-Временные параметры.

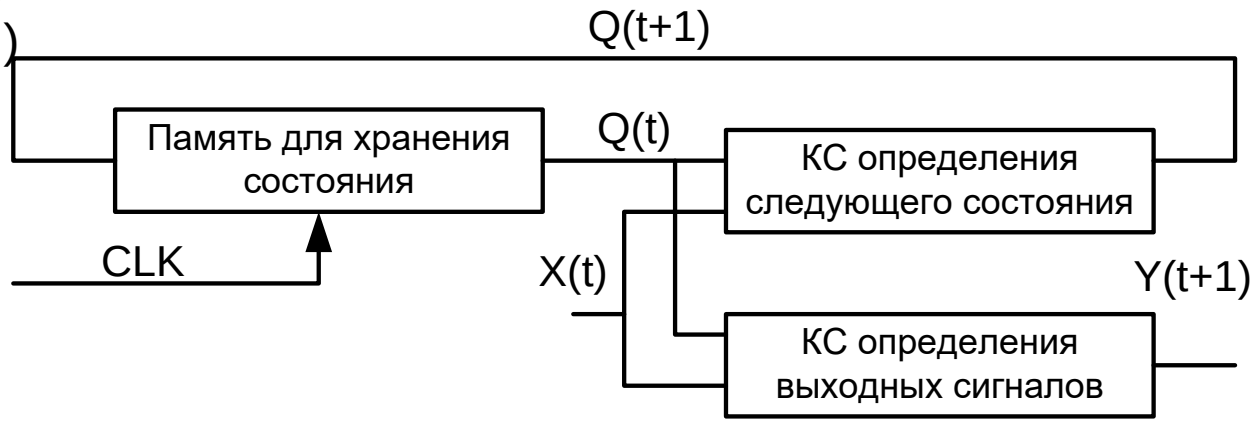
Процедура синтеза:

-Определение алгоритма функционирования управляющего автомата.
-Определение состояний с учетом выбранного типа автомата (Мили или Мура).
-Кодирование состояний автомата.
-Определение логических функций для сигналов управления и их минимизация.
-Определение логических функций перехода

Автомат Мили

$$\begin{cases} Q(t+1) = A(Q(t), x(t)) \\ Y(t+1) = B(Q(t), x(t)) \end{cases}$$

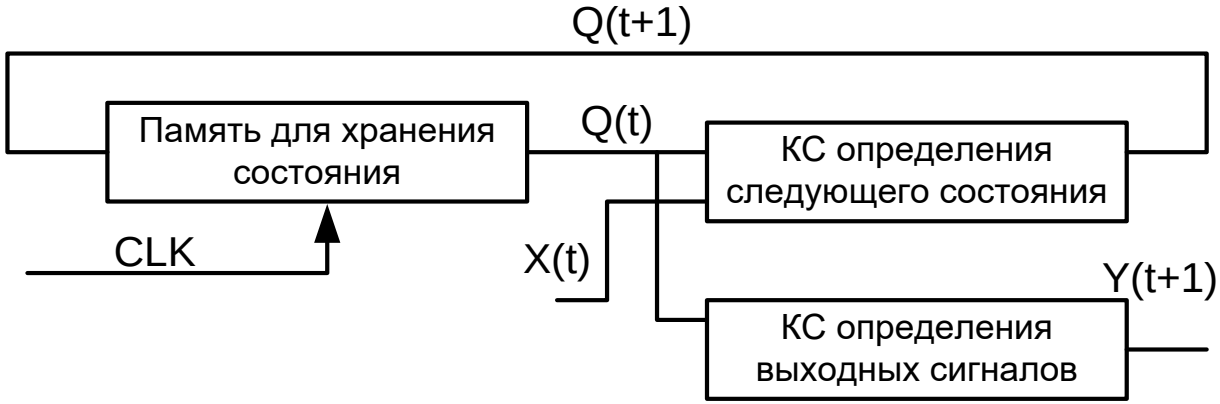
Схема автомата Мили



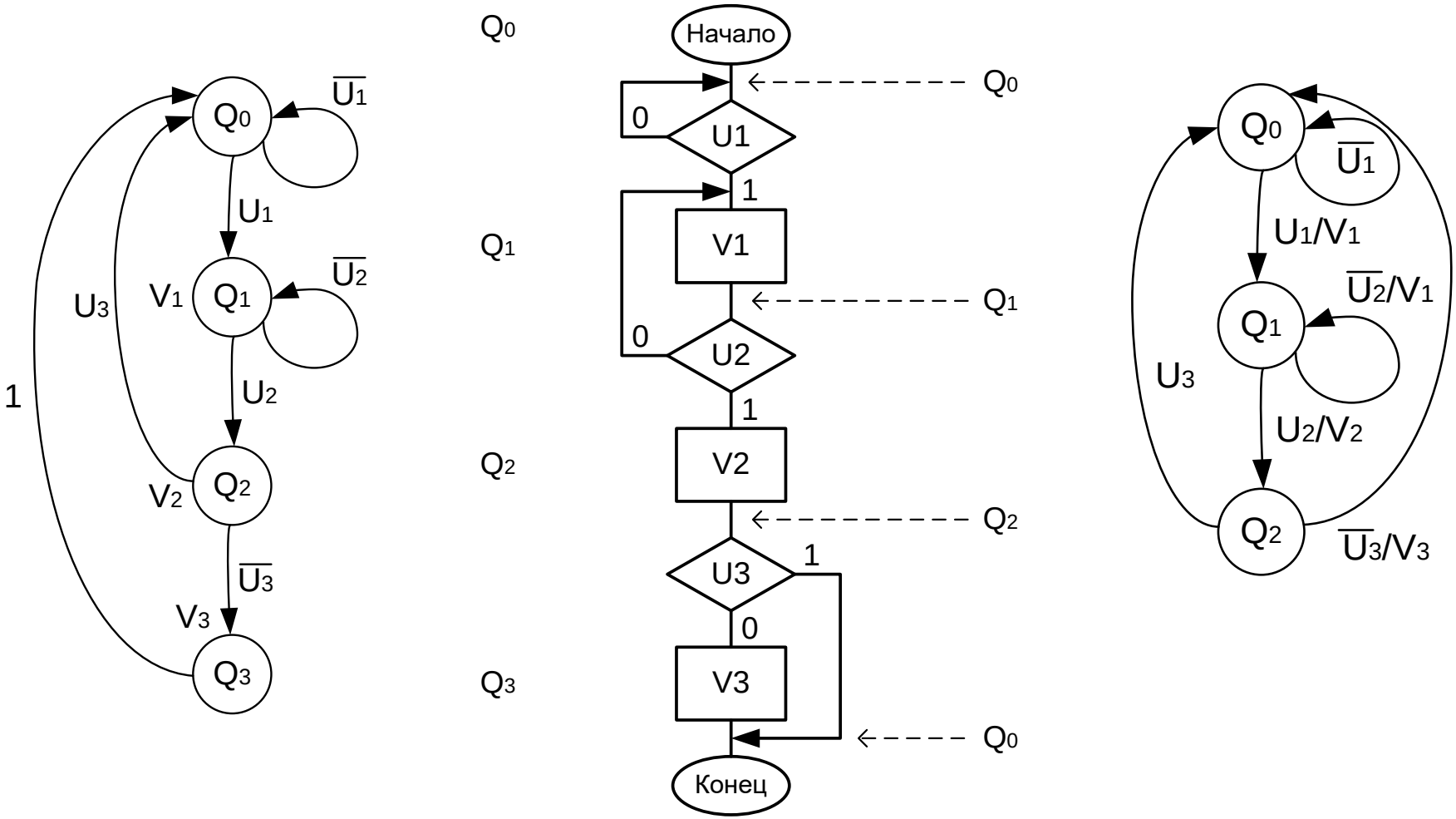
Автомат Мура

$$\begin{cases} Q(t+1) = A(Q(t), x(t)) \\ Y(t+1) = B(Q(t)) \end{cases}$$

Схема автомата Мура



Пример синтеза УУ с ЖЛ на основе автоматов Мили и Мура



Автомат Мура

$V_i = U\ Q_{Vi}$, где Q_{Vi} – состояния автомата, в которых сигнал V_i активен.

$V_1 = Q_1$; $V_2 = Q_2$; $V_3 = Q_3$;

$Q_0 = Q_0!U_1\ U\ Q_2U_3\ U\ Q_3$;

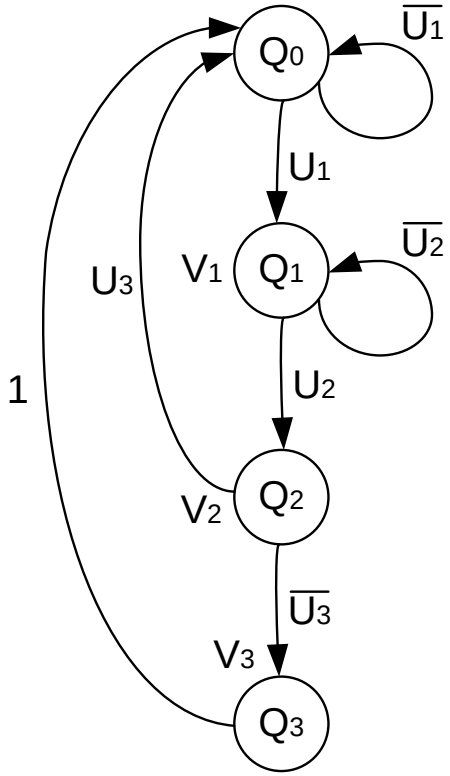
$Q_1 = Q_1!U_2\ U\ Q_0U_1$

$Q_2 = Q_1U_2$

$Q_3 = Q_2!U_3$

Таблица кодирования состояний

	D1	D0
Q0	0	0
Q1	0	1
Q2	1	0
Q3	1	1

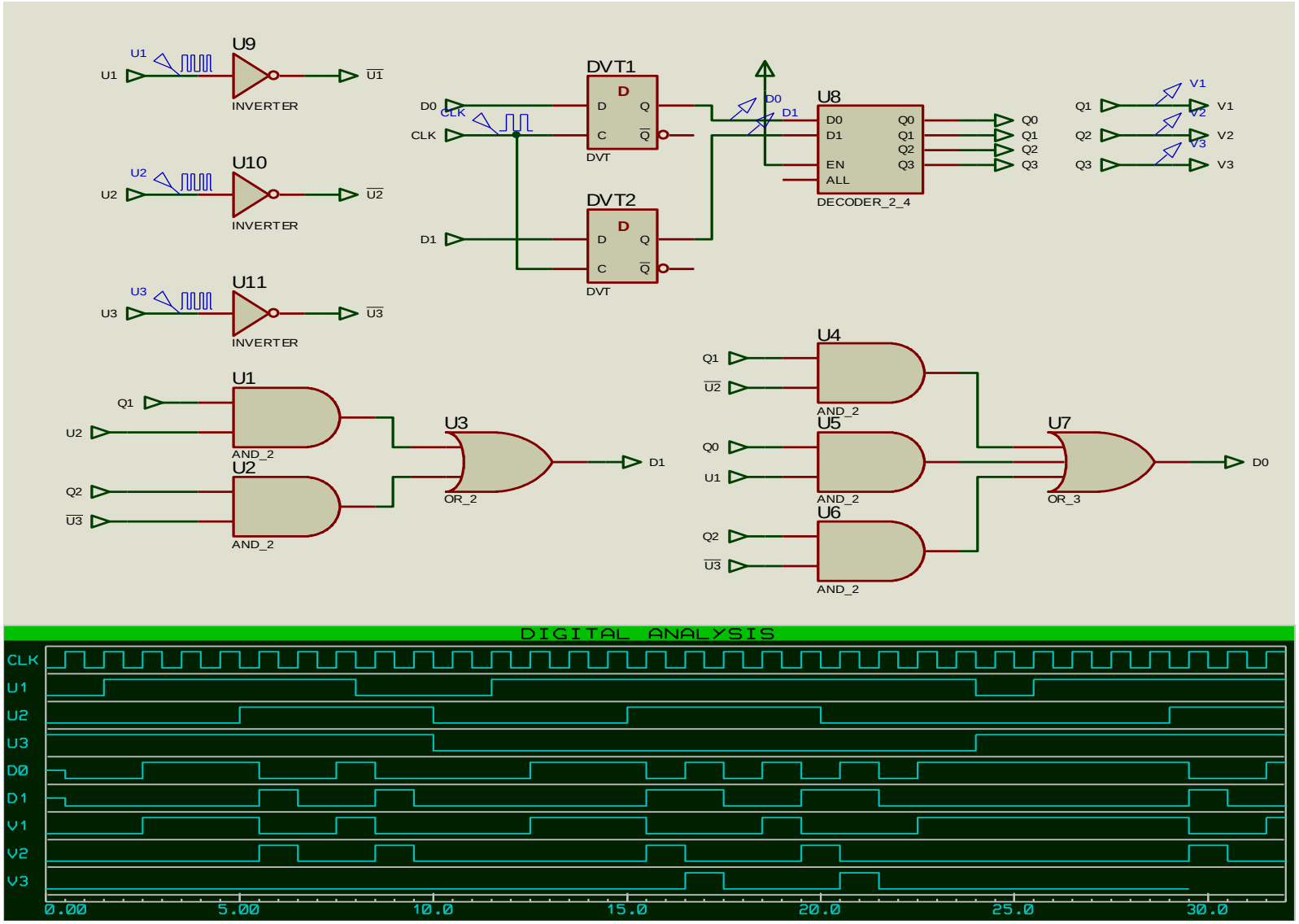


Наиболее используемые способы кодирования состояний: двоичное, One-Hot, Код Грея

$D_0(t+1) = Q_1(t)\ U\ Q_3(t) = Q_1!U_2\ U\ Q_0U_1\ U\ Q_2!U_3$;

$D_1(t+1) = Q_2(t)\ U\ Q_3(t) = Q_1U_2\ U\ Q_2!U_3$;

Автомат Мура



Автомат Мили

$V_i = U(Q_{Vi}U_j)$, где Q_{Vi} – состояния автомата, в которых сигнал V_i активен, U_j - условие перехода.

$$V_1 = Q_0U_1 \cup Q_1!U_2;\ V_2 = Q_1U_2;\ V_3 = Q_2!U_3;$$

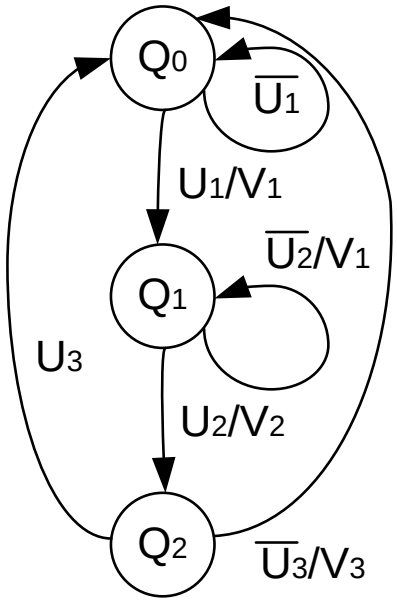
$$Q_0 = Q_0!U_1 \cup Q_2U_3 \cup Q_2!U_3;$$

$$Q_1 = Q_1!U_2 \cup Q_0U_1$$

$$Q_2 = Q_1U_2$$

Таблица кодирования состояний

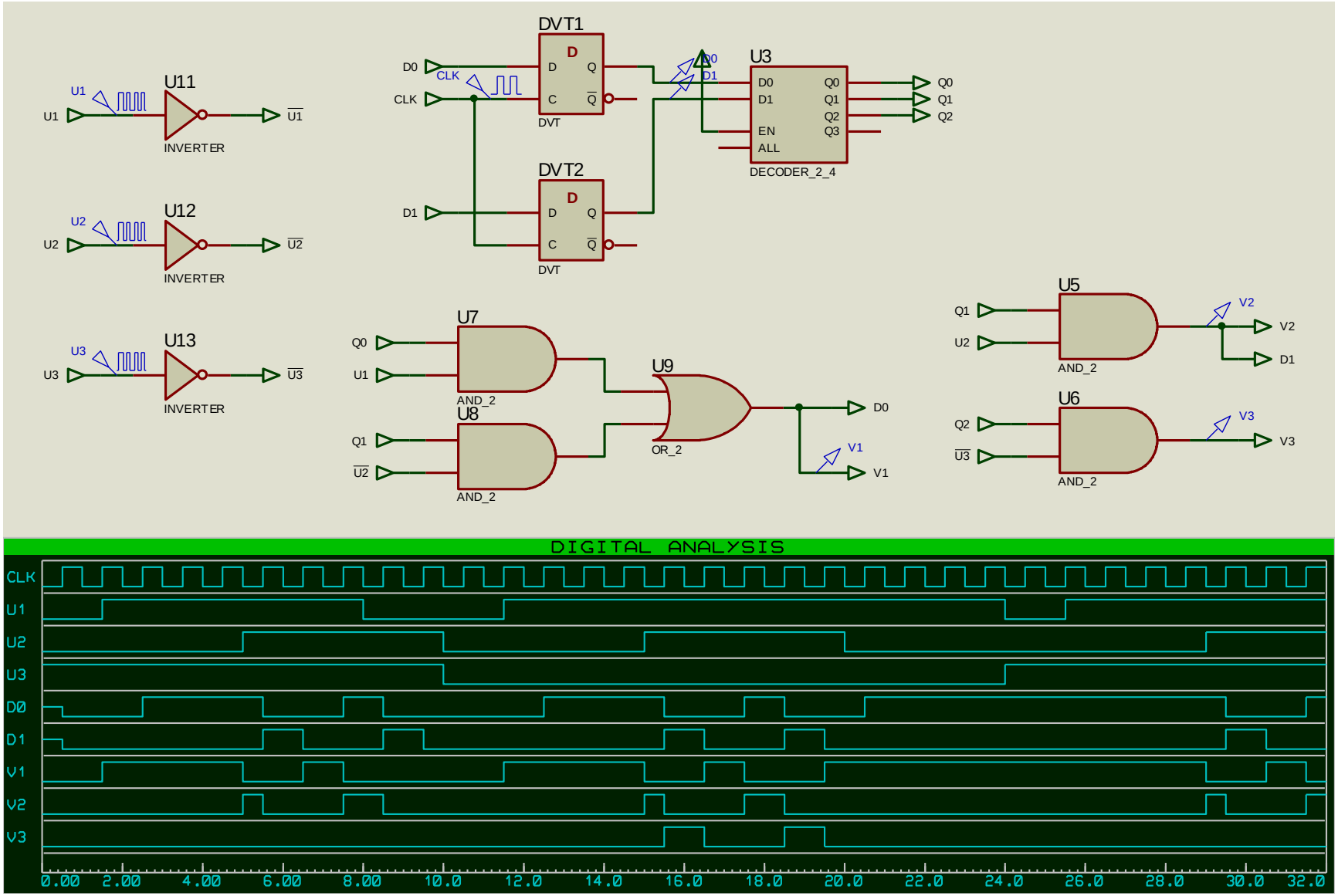
	D1	D0
Q0	0	0
Q1	0	1
Q2	1	0



$$D_0(t+1) = Q_1(t) = Q_1!U_2 \cup Q_0U_1;$$

$$D_1(t+1) = Q_2(t) = Q_1U_2;$$

Автомат Мили



Автоматы Мили и Мура с синхронными входами и выходами

Схема автомата Мили с синхронным выходом

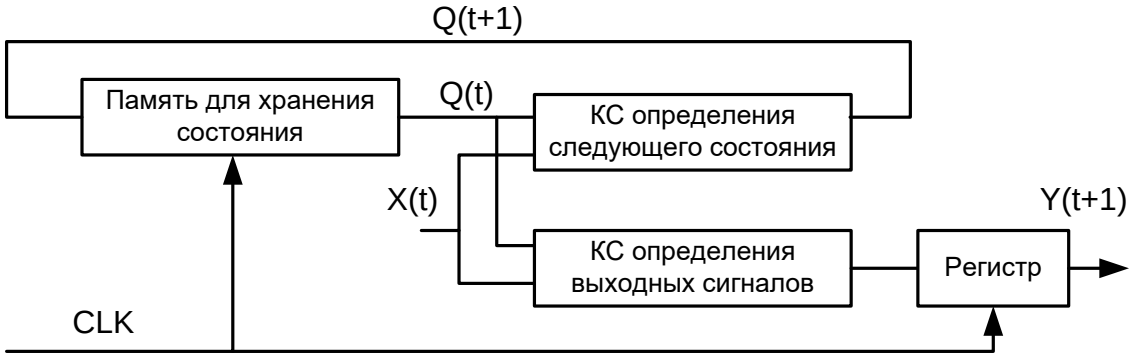
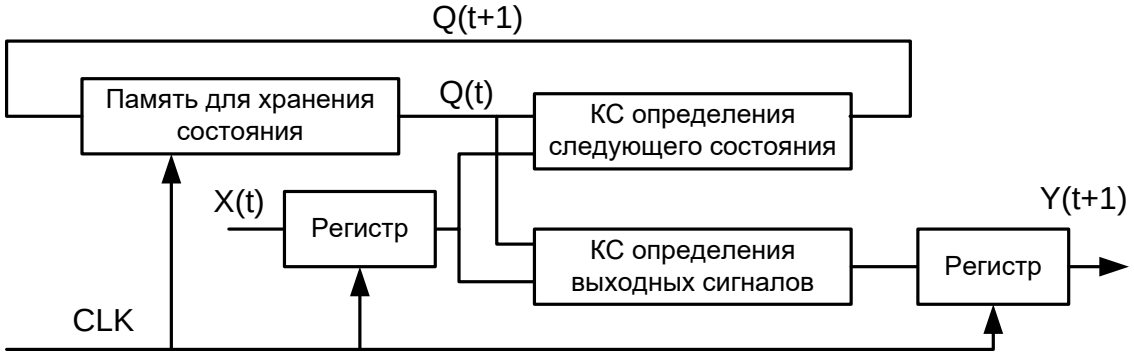


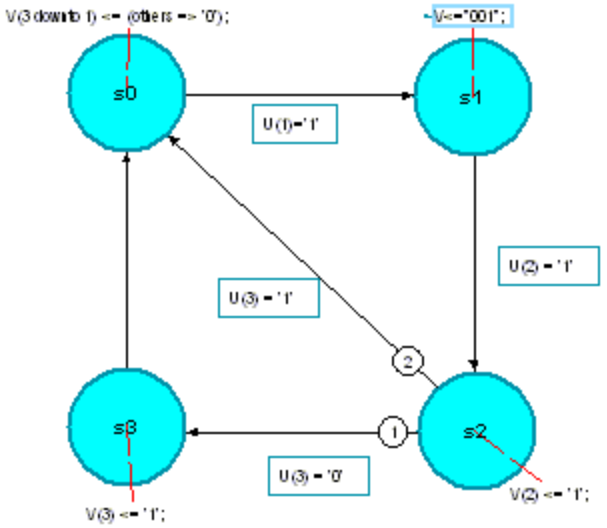
Схема автомата Мили с синхронными входами/выходами



Описание автомата Мура на языке VHDL

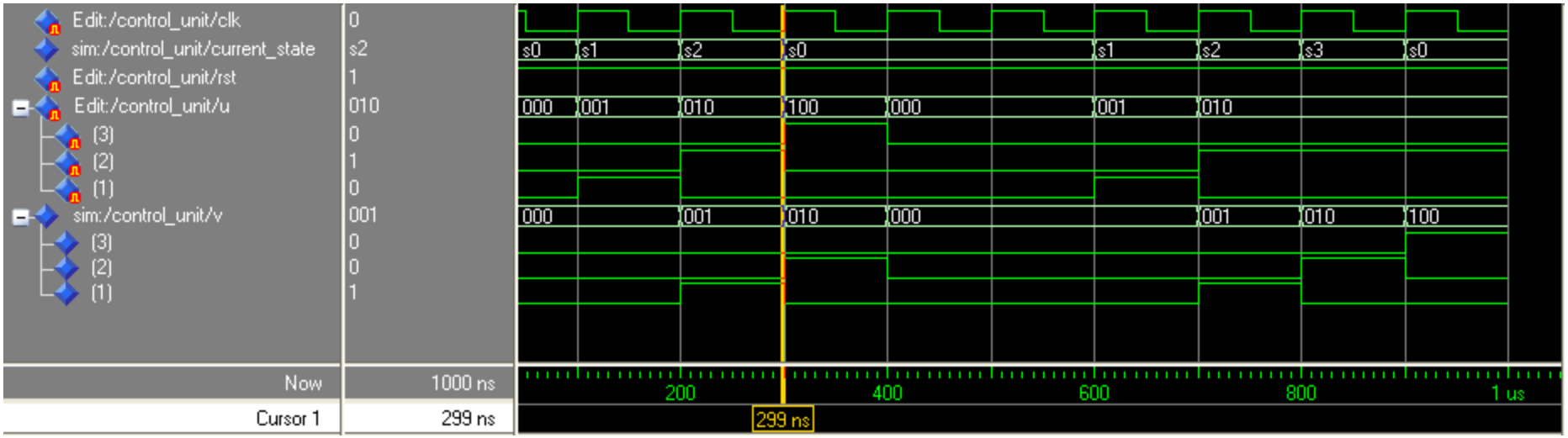
(вариант с синхронными входами и выходами)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
ENTITY control_unit IS
  PORT(      U   : IN     std_logic_vector ( 3 DOWNTO 1 );
      clk : IN     std_logic;
      rst : IN     std_logic;
      V   : OUT    std_logic_vector ( 3 DOWNTO 1 ) );
END control_unit;
ARCHITECTURE  moore OF control_unit IS
  TYPE STATE_TYPE  IS (s0, s1,s2,s3);
  SIGNAL current_state : STATE_TYPE;
BEGIN
  clocked_proc : PROCESS (clk, rst)
  BEGIN
    IF (rst = '0') THEN
      current_state <= s0;
    ELSIF (clk'EVENT AND clk = '1') THEN
```



```
CASE current_state IS
        WHEN s0 =>
          V(3 downto 1) <= (others => '0');
          IF (U(1)='1') THEN current_state <= s1;
          ELSE current_state <= s0; END IF;
        WHEN s1 =>
          V<= "001";
          IF (U(2) = '1') THEN current_state <= s2;
          ELSE current_state <= s1; END IF;
        WHEN s2 =>
          V <= "010";
          IF (U(3) = '0') THEN  current_state <= s3;
          ELSE current_state <= s0; END IF;
        WHEN s3 =>
          V <= "100";
          current_state <= s0;
        WHEN OTHERS =>
          current_state <= s0;
      END CASE;
    END IF;
  END PROCESS clocked_proc;
END moore;
```

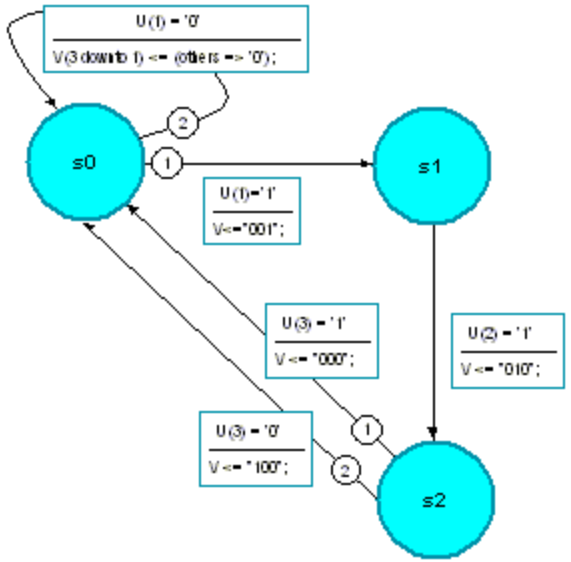
Тест автомат Мура в ModelSim 6



Описание автомата Мили на языке VHDL

(вариант с синхронными выходами)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
ENTITY control_unit IS
  PORT(      U   : IN     std_logic_vector ( 3 DOWNTO 1 );
      clk : IN     std_logic;
      rst : IN     std_logic;
      V   : OUT    std_logic_vector ( 3 DOWNTO 1 ) );
END control_unit;
ARCHITECTURE  mielie OF control_unit IS
  TYPE STATE_TYPE  IS (s0, s1,s2);
  SIGNAL current_state : STATE_TYPE;
BEGIN
  clocked_proc : PROCESS (clk, rst)
  BEGIN
    IF (rst = '0') THEN
      current_state <= s0;
    ELSIF (clk'EVENT AND clk = '1') THEN
```

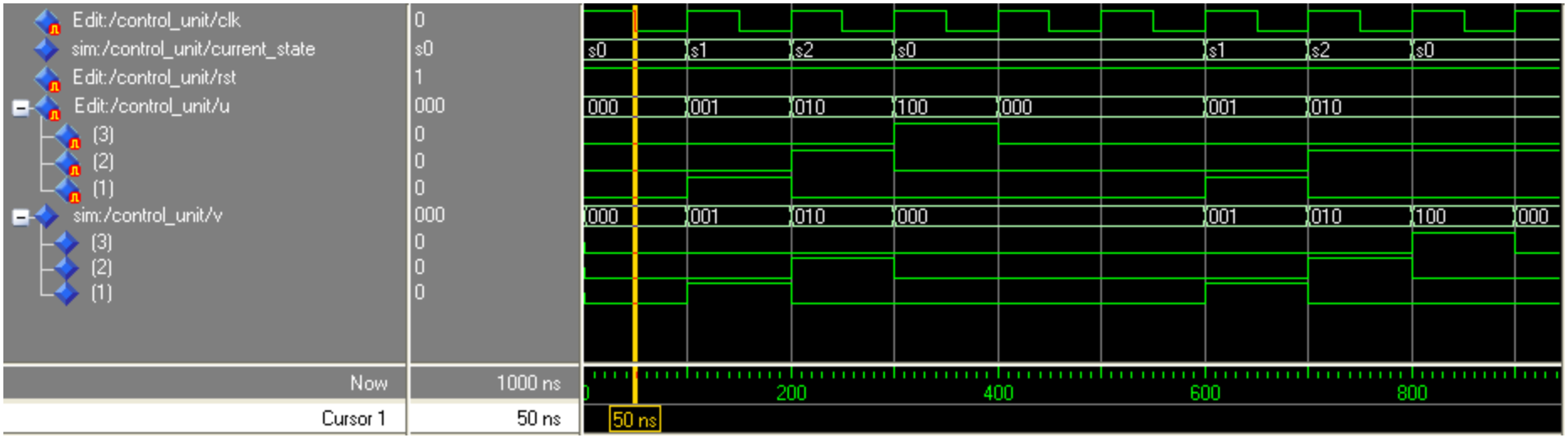


Описание автомата Мили на языке VHDL (окончание)

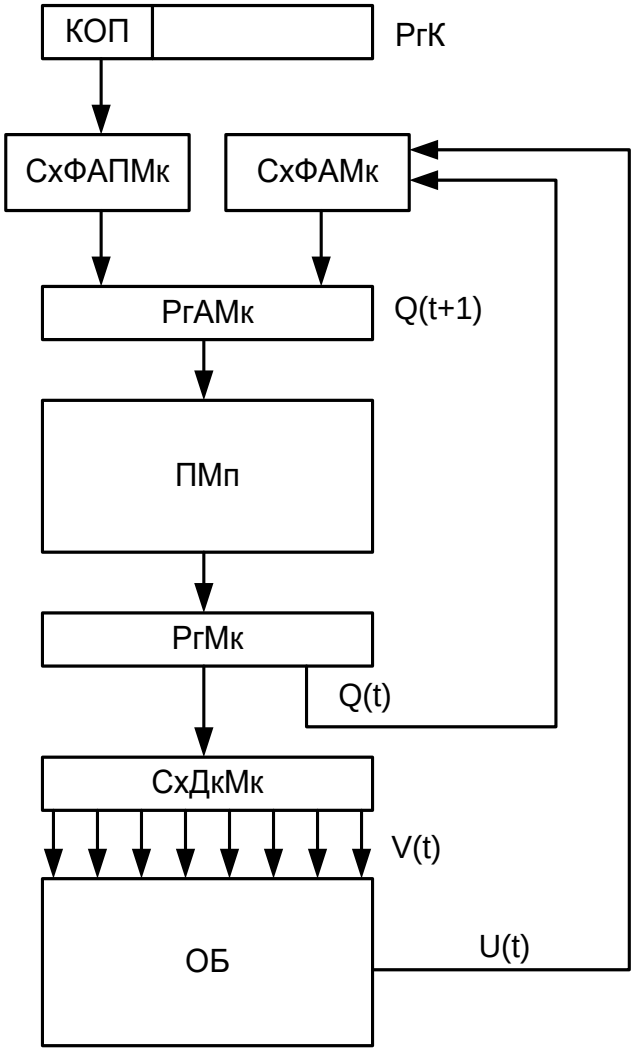
```
CASE current_state IS
        WHEN s0 =>
          IF (U(1)='1') THEN
  V<="001";
            current_state <= s1;
          ELSIF (U(1) = '0') THEN
            V(3 downto 1) <= (others => '0');
            current_state <= s0;
          ELSE
            current_state <= s0;
          END IF;
        WHEN s1 =>
          IF (U(2) = '1') THEN
            V <= "010";
            current_state <= s2;
          ELSE
            current_state <= s1;
          END IF;
        WHEN s2 =>
          IF (U(3) = '1') THEN
            V <= "000";
            current_state <= s0;
          ELSIF (U(3) = '0') THEN
            V <= "100";
            current_state <= s0;
          ELSE
            current_state <= s2;
          END IF;
        WHEN OTHERS =>
          current_state <= s0;
      END CASE;
    END IF;
  END PROCESS clocked_proc;

END mielie;
```

Тест автомат Мили в ModelSim 6



Управляющие устройства с программируемой логикой



РгК – регистр команды;
КОП – код операции;
СхФАПМк – схема формирования адреса первой микрокоманды;
СхФАМк – схема формирования адреса микрокоманды;
РгАМк – регистр адреса микрокоманды;
ПМп – память микропрограмм;
РгМк – регистр микрокоманды;
СхДкМк – схема декодирования микрокоманд;
ОБ – операционный блок.

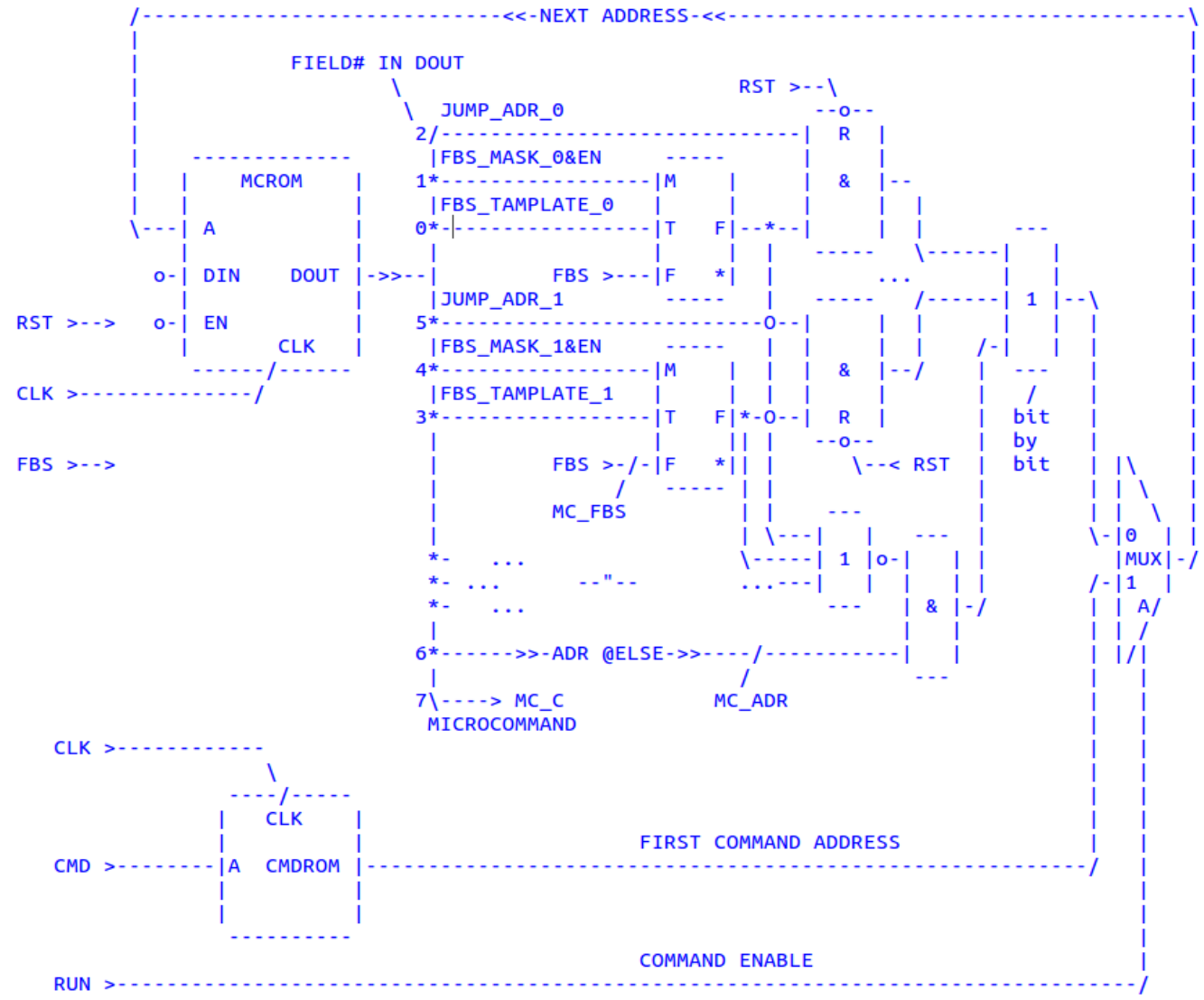
<u>В зависимости от типа ПМп:</u>

-Статическое микропрограммирование (ROM);
-Динамическое программирование (RAM).

<u>По способу адресации микрокоманд:</u>

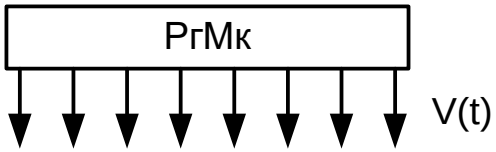
-Принудительная адресация;
-Естественная адресация.

Управляющие устройства с программируемой логикой

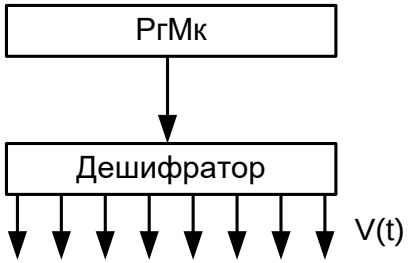


Способы кодирования микрокоманд

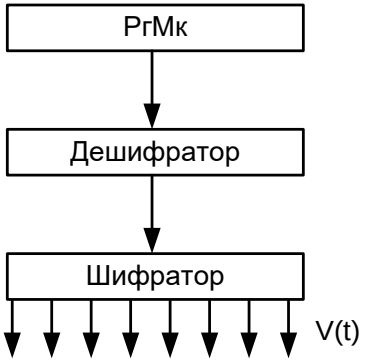
Минимальное кодирование (горизонтальное).



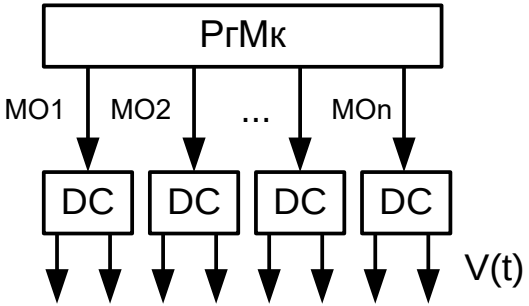
Максимальное кодирование (вертикальное).



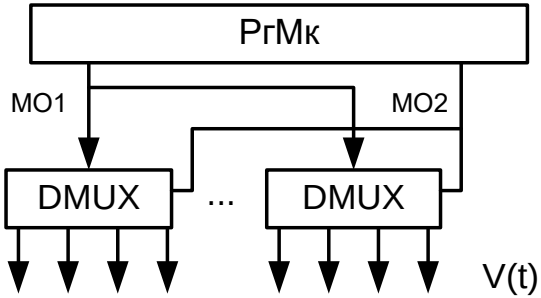
Максимальное кодирование с шифратором.



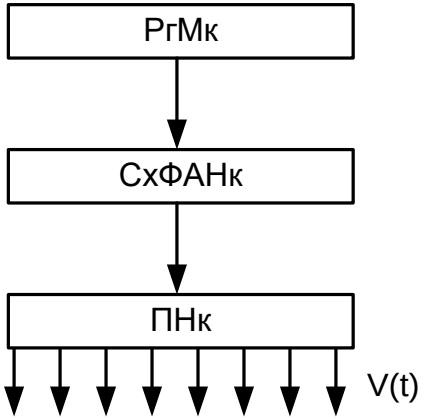
Вертикально-горизонтальное кодирование



Горизонтально-вертикальное кодирование.



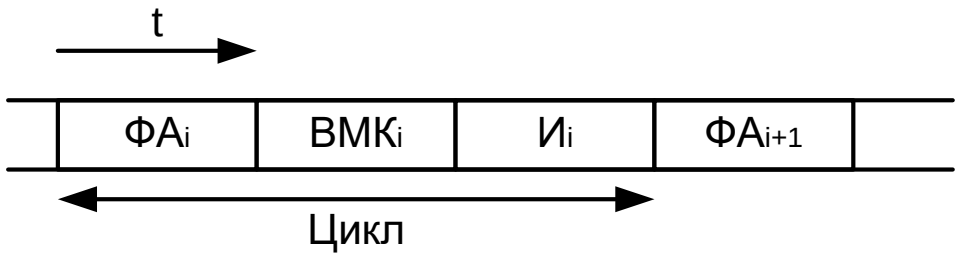
Кодирование с помощью памяти нанокоманд



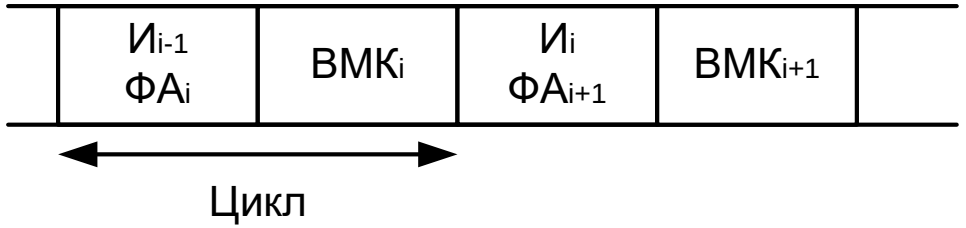
Цикл микрокоманды

Цикл исполнения микрокоманды;
-Формирование адреса микрокоманды.
-Выборка микрокоманды.
-Исполнение микрокоманды.

При последовательном исполнении цикла:
$T_{МК} = T_{ФА} + T_{ВМК} + T_{И}$.



При совмещении исполнения и формирования адреса следующей микрокоманды:
$T_{МК} = MAX((T_{ФА} + T_{ВМК}), T_{И})$.



При совмещении всех действий:
$T_{МК} = MAX(T_{ФА}, T_{ВМК}, T_{И})$.