МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ПО ВЫПОЛНЕНИЮ ДОМАШНЕГО ЗАДАНИЯ ПО КУРСУ «ЭВМ»

В ходе выполнения домашнего задания необходимо разработать устройство управления схемного типа, обрабатывающий входное командное слово C={ABCDEF} и выдающий сигналы управления $M=\{M_0,\ldots,M_{k-1}\}$ операционному блоку в соответствии с приведенной в индивидуальном задании логикой работы.
Домашнее задание выполняется в несколько этапов.

Этап 1.

А. По диаграмме переходов автомата (Приложение 1) и описанию условий переходов и активных сигналов (дополнительный файл варианты.pdf), определить тип управляющего автомата (автомат Мили или Мура, смешанный). Выбор обосновать.
В. Произвести кодирование состояний управляющего автомата. Составить схему переходов/состояний полученного автомата. Схему представить в отчете.

Этап 2.

Разработать описание устройства управления на языке VHDL, для чего использовать приведенные в Приложении 2 шаблоны для автоматов Мили и Мура.
Разработать тестовое описание для устройства, представляющее собой генератор входных сигналов (см. Приложение 3). Тестовое описание должно обеспечивать проверку всех ветвей автомата.

Этап 3.

А. Установить ПО ModelSim PE (или аналогичный продукт: Xilinx ISE, Altera Quartus), скачав дистрибутив по адресу:
http://s3.mentor.com/fv/modelsim-pe_student_edition.exe
Следовать инструкциям по установке.
Б. Выполнить моделирование полученного теста в ПО ModelSim PE. Результаты моделирования представить в отчете.

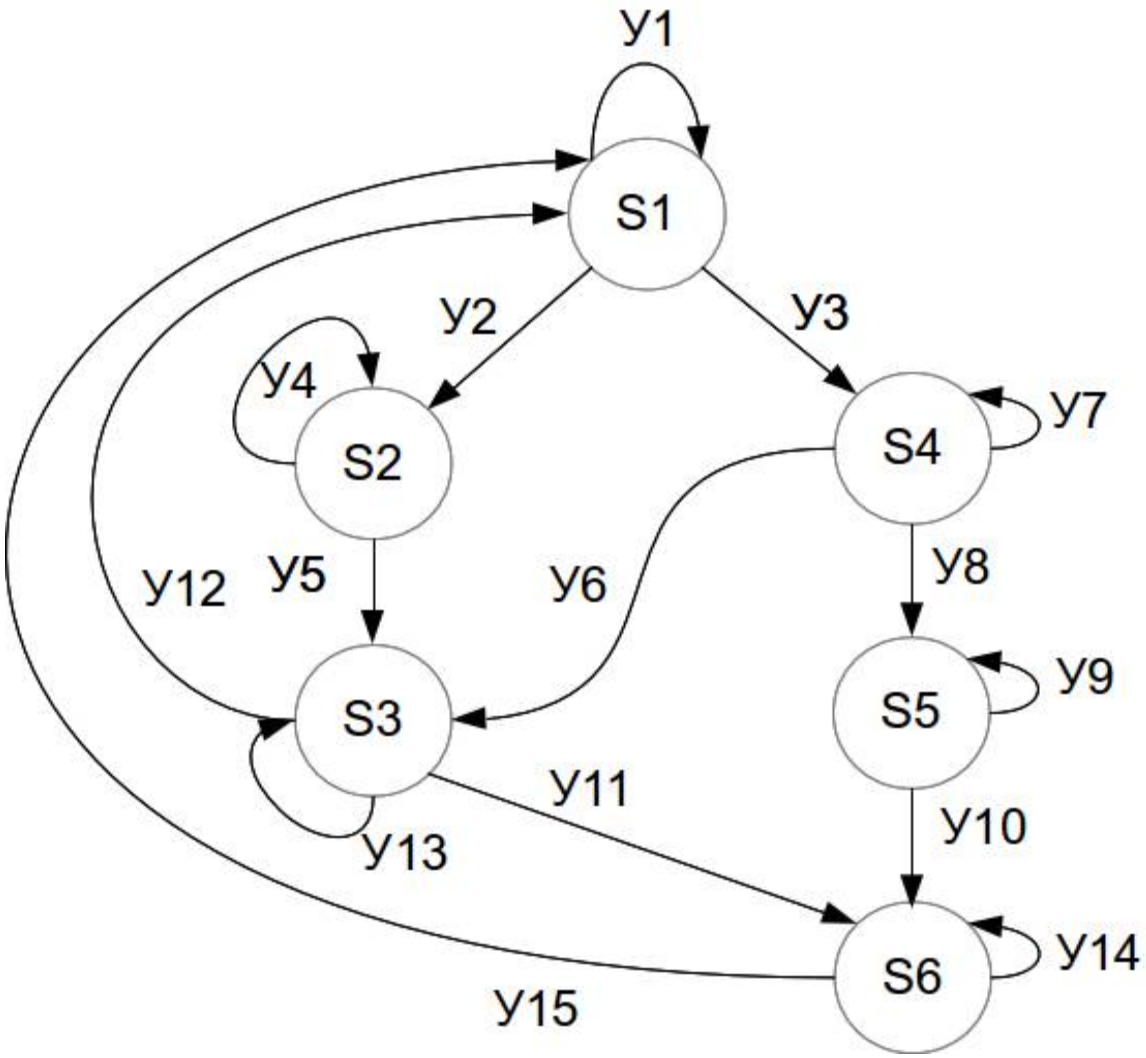
Содержание отчета

1. Исходное задание для проведения лабораторной работы.
2. Диаграмма переходов состояний управляющего автомата с условиями, указанными в индивидуальном задании.
3. Листинг VHDL описания управляющего автомата.
4. Листинг VHDL тестового описания.
5. Результат моделирования автомата

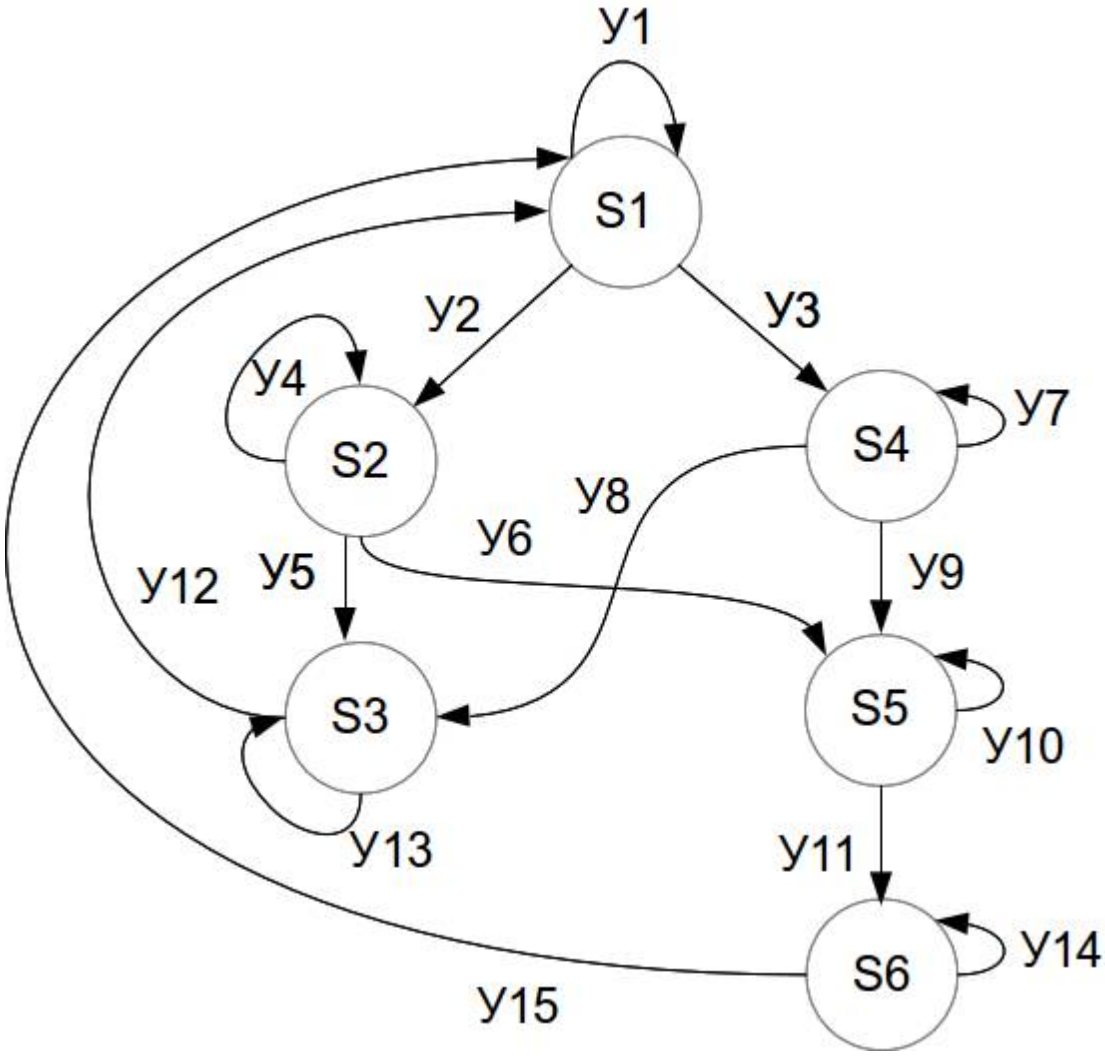
ПРИЛОЖЕНИЕ 1

Варианты диаграмм переходов (индивидуальные варианты диаграмм указаны в файле «варианты.pdf»).

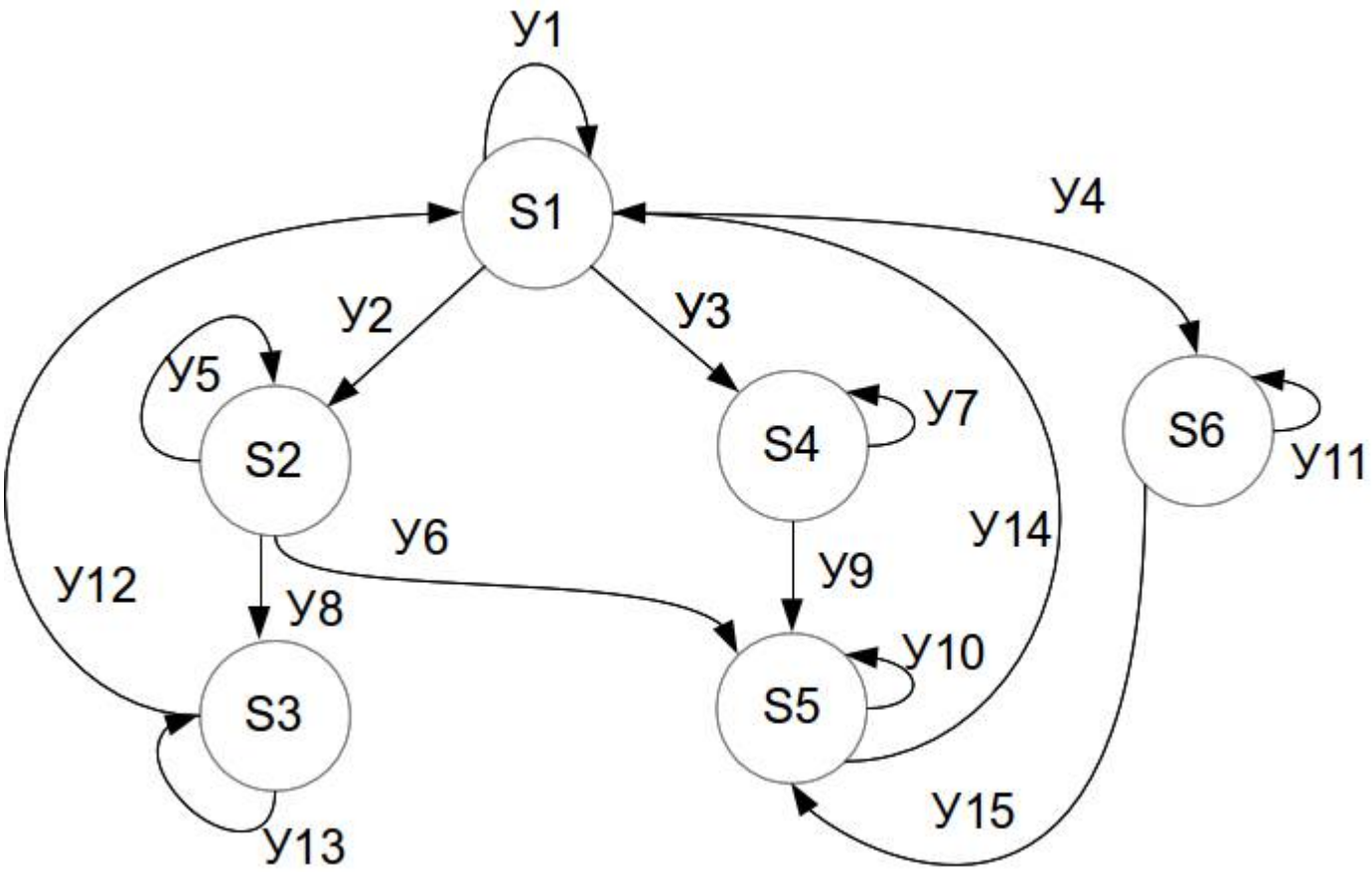
Вариант 1



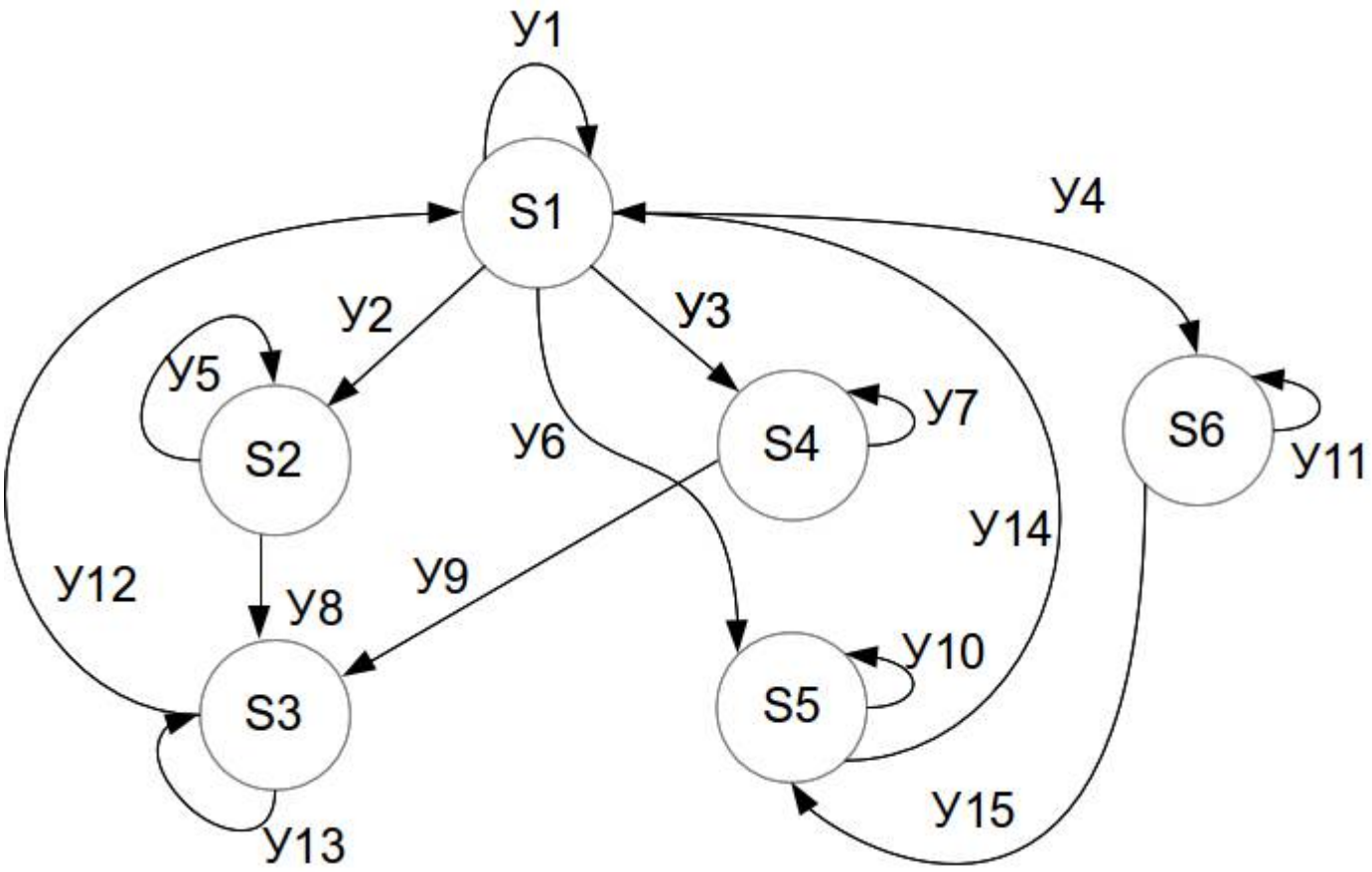
Вариант 2



Вариант 3

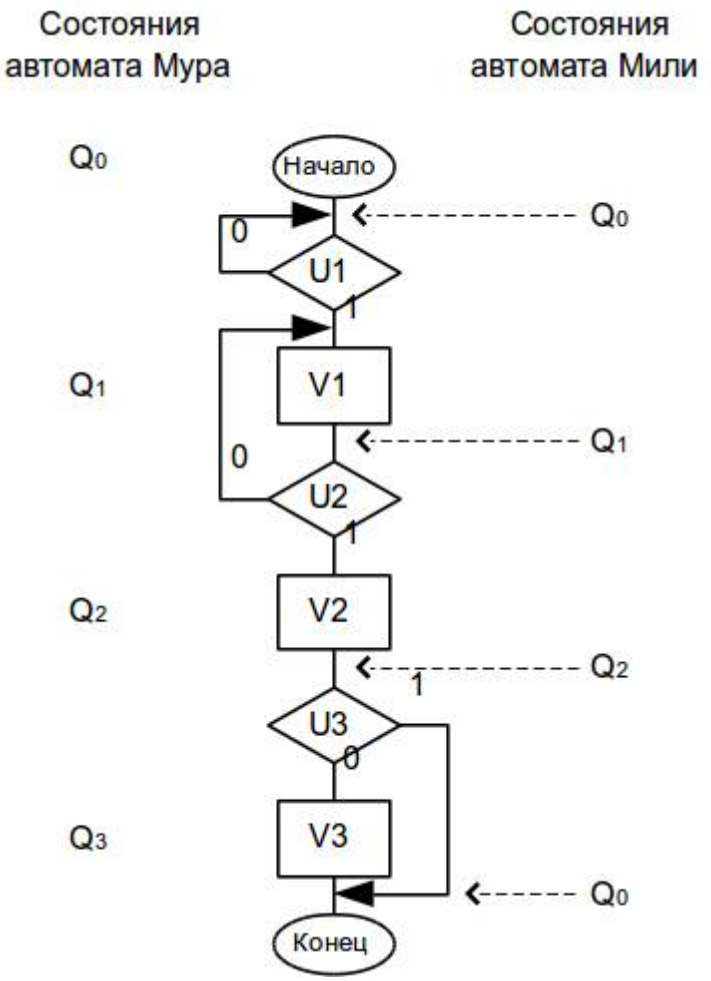


Вариант 4



ПРИЛОЖЕНИЕ 2
Пример синтеза устройства управления с жесткой логикой на основе автоматов Мили и Мура

Алгоритм работы устройства управления:



Пример описания автомата Мура на языке VHDL (вариант с асинхронными 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, next_state : STATE_TYPE;
BEGIN
```

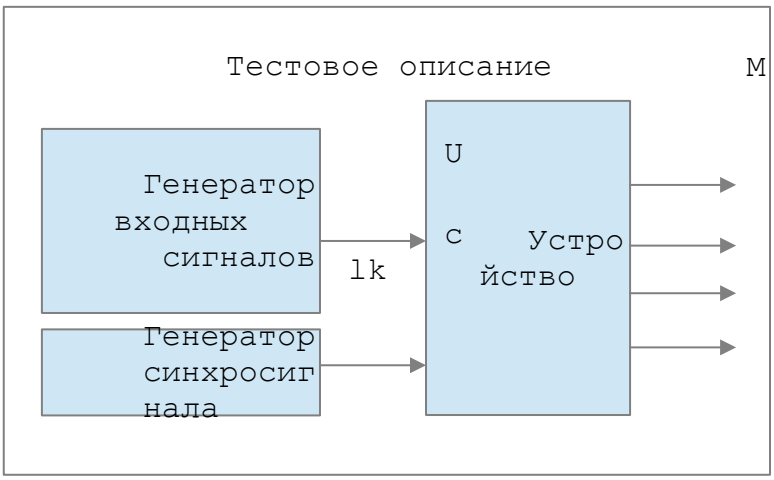
```
clocked_proc: PROCESS(clk)
BEGIN
      IF(rising_edge(clk)) THEN
            IF (reset='1') THEN
                  current_state <= s0;
            ELSE
                  current_state <= next_state;
            END IF;
      END IF;
END PROCESS;

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

Пример описания автомата Мили на языке VHDL (вариант с асинхронными 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 mielie OF control_unit IS
   TYPE STATE_TYPE IS (s0, s1,s2,s3);
   SIGNAL current_state, next_state : STATE_TYPE;
```

```
BEGIN

clocked_proc: PROCESS(clk)
BEGIN
      IF(rising_edge(clk)) THEN
            IF (reset='1') THEN
                  current_state <= s0;
            ELSE
                  current_state <= next_state;
            END IF;
      END IF;
END PROCESS clocked_proc;

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

ПРИЛОЖЕНИЕ 2
Пример тестового описания для устройства управления с жесткой логикой на основе автоматов Мили




```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY test IS
END test;


ARCHITECTURE behavior OF test IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT control_unit
    PORT(
         U : IN  std_logic_vector(3 downto 1);
         clk : IN  std_logic;
         rst : IN  std_logic;
         M : OUT  std_logic_vector(3 downto 1)
        );
    END COMPONENT;


   --Inputs
   signal U : std_logic_vector(3 downto 1) := (others => '0');
   signal clk : std_logic := '0';
   signal rst : std_logic := '0';

      --Outputs
   signal M : std_logic_vector(3 downto 1);

   -- Clock period definitions
   constant clk_period : time := 10ns;

BEGIN
```

```
      -- Instantiate the Unit Under Test (UUT)
   uut: control_unit PORT MAP (
          U => U,
          clk => clk,
          rst => rst,
          M => M
        );

   -- Clock process definitions
   clk_process :process
   begin
            clk <= '0';
            wait for clk_period/2;
            clk <= '1';
            wait for clk_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100ns.
            rst<='1';
      wait for 100ns;

            rst<='0';

      wait for clk_period*10;

      -- insert stimulus here

            U<="000";
            wait for clk_period;

            U<="001";
            wait for clk_period;


            U<="010";
            wait for clk_period;

            U<="011";
            wait for clk_period;

      wait;
   end process;

END;
```