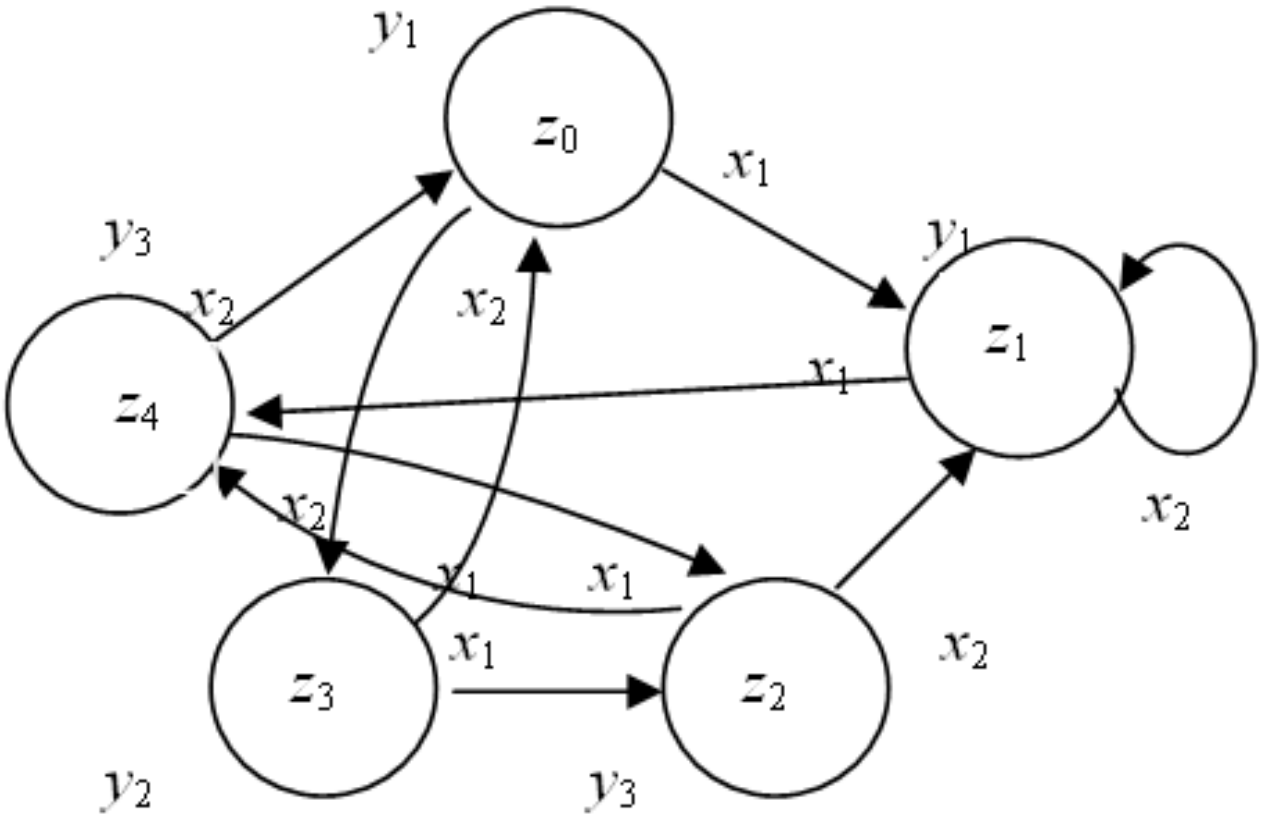
17 НОЯБРЯ 2015 Г.



Практика по VHDL

Лабораторная работа №2

ЮРИЙ МИХАЙЛУЦ

ИММиКН ЮФУ

НЕОБХОДИМЫЕ ТЕОРЕТИЧЕСКИЕ ЗНАНИЯ

Реализация автомата на VHDL

Объявление компонента.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_arith.all;
4
5  entity machine is
6  port(
7      -- ============================
8      -- <<Вспомогательные сигналы>>
9      -- тактовый сигнал
10     clk: in std_logic;
11     -- сигнал сброса
12     rst: std_logic;
13     -- ============================
14     -- <<Входной символ автомата>>
15     -- сигнал выключения автомата
16     input_off: in std_logic;
17     -- сигнал включения автомата
18     input_on: in std_logic;
19     -- входное слово
20     input_word: in std_logic_vector(7 downto 0);
21     -- ============================
22     -- <<Выходной символ автомата>>
23     -- сигнал готовности выходных данных
24     output_valid: out std_logic;
25     -- выходное слово
26     output_word: out std_logic_vector(31 downto 0);
27     );
28 end machine;
```

Сигналы и типы, необходимые для реализации автомата.

```
30 architecture machine_synth of machine is
31     -- Объявление типа, перечисляющего состояния нашего автомата
32     type machine_state is (Init, Read1, Read2, Read3, Read4, Write1);
33     -- ============================
34     -- <<Текущее состояние автомата>>
35     -- Текущее состояние (скомпилируется в регистр)
36     signal present_state : machine_state;
37     -- сигнал для хранения выходного порта
38     signal output_word_int: std_logic_vector(31 downto 0);
39     -- ============================
40     -- <<Функция состояния автомата>>
41     -- Следующее состояние (скомпилируется в логическую функцию)
42     signal next_state : machine_state;
43     -- Сигналы, соответствующие определённым байтам выходного слова,
44     signal to_output_byte0: std_logic_vector(7 downto 0);
45     signal to_output_byte1: std_logic_vector(7 downto 0);
46     signal to_output_byte2: std_logic_vector(7 downto 0);
47     signal to_output_byte3: std_logic_vector(7 downto 0);
48
49 begin
```

Процесс записи выходного сигнала в порт и процесс, устанавливающий новое состояние автомата. Первый процесс – асинхронный, второй – синхронный.

```
51      -- Запись выходного порта
52      output_word <= output_word_int;
53
54      -- Каждый такт, елси нет сигнала сброса, устанавливает новое состояние
55      PresentStateProc: process (clk, rst, Rg1_Ready)
56      begin
57          if rst = '1' then
58              present_state <= Init;
59              output_word_int (7 downto 0) <= (others => '0');
60              output_word_int (15 downto 8) <= (others => '0');
61              output_word_int (23 downto 16) <= (others => '0');
62              output_word_int (31 downto 24) <= (others => '0');
63          elsif clk'event and clk = '1' then
64              present_state <= next_state;
65              output_word_int (7 downto 0) <= to_output_byte0;
66              output_word_int (15 downto 8) <= to_output_byte1;
67              output_word_int (23 downto 16) <= to_output_byte2;
68              output_word_int (31 downto 24) <= to_output_byte3;
69          end if;
70      end process;
```

Данный процесс описывает таблицу истинности для функции, вычисляющей следующее состояние автомата, он будет скомпилирован в логическую таблицу.

```
72      -- Вычисляет следующее состояние
73      NextStateProc: process (present_state, input_on, input_off)
74      begin
75          case present_state is
76              when Init =>
77                  if input_on = '1' then
78                      next_state <= Read1;
79                  else
80                      next_state <= Init;
81                  end if;
82              when Read1 =>
83                  next_state <= Read2;
84              when Read2 =>
85                  next_state <= Read3;
86              when Read3 =>
87                  next_state <= Read4;
88              when Read4 =>
89                  next_state <= Write1;
90              when Write1 =>
91                  if input_off = '1' then
92                      next_state <= Init;
93                  else
94                      next_state <= Read1;
95                  end if;
96              when others =>
97                  next_state <= Init;
98          end case;
99      end process;
```

Ещё один процесс, описывающий таблицу истинности, но на этот раз для выходного сигнала.

```
101     -- Вычисляет выходные значения, в зависимости от текущего состоя
102     OutputFuncProc: process (present_state)
103     begin
104         case present_state is
105             when Init =>
106                 to_output_byte0 <= (others => '0');
107                 to_output_byte1 <= (others => '0');
108                 to_output_byte2 <= (others => '0');
109                 to_output_byte3 <= (others => '0');
110 
111                 output_valid <= '0';
112             when Read1 =>
113                 to_output_byte0 <= to_output_byte0;
114                 to_output_byte1 <= (others => '0');
115                 to_output_byte2 <= (others => '0');
116                 to_output_byte3 <= (others => '0');
117 
118                 output_valid <= '0';
119             when Read2 =>
120                 to_output_byte0 <= output_word_int (7 downto 0) ;
121                 to_output_byte1 <= to_output_byte1;
122                 to_output_byte2 <= (others => '0');
123                 to_output_byte3 <= (others => '0');
124 
125                 output_valid <= '0';
126             when Read3 =>
127                 to_output_byte0 <= output_word_int (7 downto 0) ;
128                 to_output_byte1 <= output_word_int (15 downto 8) ;
129                 to_output_byte2 <= to_output_byte2;
130                 to_output_byte3 <= (others => '0');
131 
132                 output_valid <= '0';
133             when Read4 =>
134                 to_output_byte0 <= output_word_int (7 downto 0) ;
135                 to_output_byte1 <= output_word_int (15 downto 8) ;
136                 to_output_byte2 <= output_word_int (23 downto 16);
137                 to_output_byte3 <= to_output_byte3;
138 
139                 output_valid <= '0';
140             when Write1 =>
141                 to_output_byte0 <= output_word_int (7 downto 0) ;
142                 to_output_byte1 <= output_word_int (15 downto 8) ;
143                 to_output_byte2 <= output_word_int (23 downto 16);
144                 to_output_byte3 <= output_word_int (31 downto 24);
145 
146                 output_valid <= '1';
147             when others =>
148                 to_output_byte0 <= (others => '0');
149                 to_output_byte1 <= (others => '0');
150                 to_output_byte2 <= (others => '0');
151                 to_output_byte3 <= (others => '0');
152 
153                 output_valid <= '0';
154         end case;
155     end process;
```

ЗАДАЧА

Построить автомат, в соответствии с условием.

ВАРИАНТЫ

- А. Разработать АЛУ. Для выполнения арифметических операций использовать встроенные в стандартную библиотеку перегруженные операторы для типа unsigned.
- Б. Разработать автомат, реализующий последовательный приёмник. Например, UART, но можете предложить свой протокол.
- В. Разработать автомат, реализующий последовательный передатчик. Например, UART, но можете предложить свой протокол.
- Г. Разработать автомат для голосования. Может быть несколько вариантов ответа. Голосуют по очереди. После каждого автомат выдаёт лидера, на текущий момент.