Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра ЕОМ



Звіт

до лабораторної роботи № 3

з дисципліни «Моделювання комп’ютерних систем»

на тему:

«Поведінковий опис цифрового автомата Перевірка роботи автомата за допомогою стенда Elbert V2 – Spartan 3A FPGA»

**Варіант №10**

Виконав:

ст. гр. КІ-201

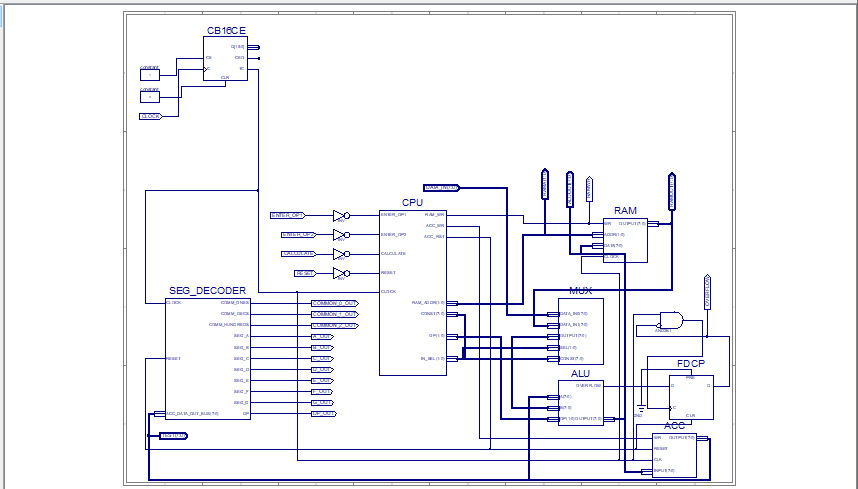
Гришканич А. М.

Прийняв:  
Козак Н. Б.

Львів 2024

**Мета роботи**: На базі стенда реалізувати цифровий автомат для обчислення значення виразів.

**Виконання роботи:**

****

*Рис. 1 – Top Level*

| Файл ACC.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. -- Uncomment the following library declaration if using 4. -- arithmetic functions with Signed or Unsigned values 5. --use IEEE.NUMERIC\_STD.ALL; 6. -- Uncomment the following library declaration if instantiating 7. -- any Xilinx primitives in this code. 8. --library UNISIM; 9. --use UNISIM.VComponents.all; 10. entity ACC is 11. Port ( WR : in STD\_LOGIC; 12. RESET : in STD\_LOGIC; 13. CLK : in STD\_LOGIC; 14. INPUT : in STD\_LOGIC\_VECTOR (7 downto 0); 15. OUTPUT : out STD\_LOGIC\_VECTOR (7 downto 0)); 16. end ACC; 17. architecture ACC\_arch of ACC is 18. signal DATA : STD\_LOGIC\_VECTOR (7 downto 0); 19. begin 20. process (CLK) 21. begin 22. if rising\_edge(CLK) then 23. if RESET = '1' then 24. DATA <= (others => '0'); 25. elsif WR = '1' then 26. DATA <= INPUT; 27. end if; 28. end if; 29. end process; 31. OUTPUT <= DATA; 32. end ACC\_arch; |
| --- |

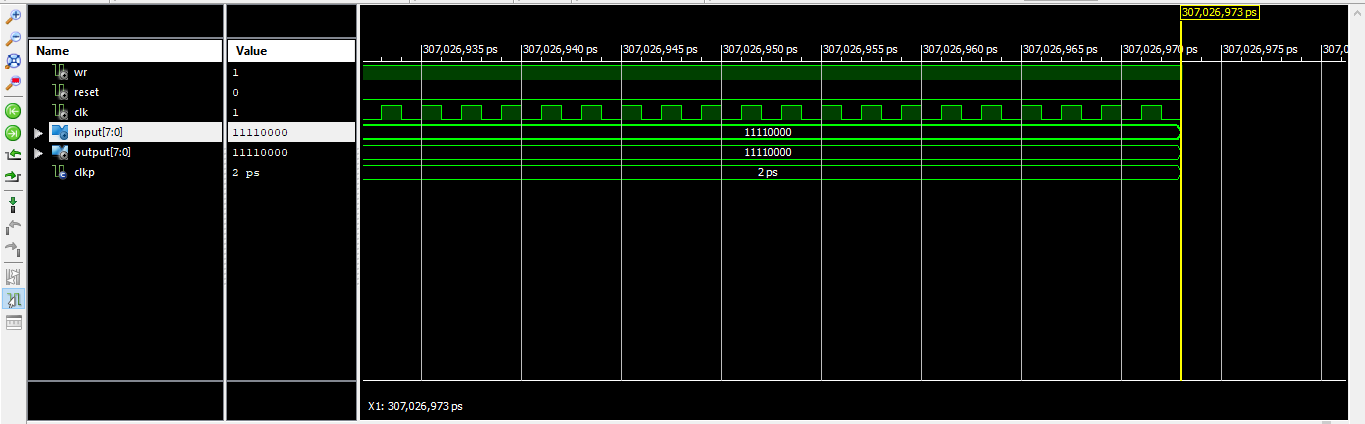
| Файл ALU.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. -- Uncomment the following library declaration if using 4. -- arithmetic functions with Signed or Unsigned values 5. use IEEE.NUMERIC\_STD.ALL; 6. use IEEE.STD\_LOGIC\_UNSIGNED.ALL; 7. -- Uncomment the following library declaration if instantiating 8. -- any Xilinx primitives in this code. 9. --library UNISIM; 10. --use UNISIM.VComponents.all; 11. entity ALU is 12. Port ( A : in STD\_LOGIC\_VECTOR(7 downto 0); 13. B : in STD\_LOGIC\_VECTOR(7 downto 0); 14. OP : in STD\_LOGIC\_VECTOR(1 downto 0); 15. OUTPUT : out STD\_LOGIC\_VECTOR(7 downto 0); 16. OVERFLOW: out STD\_LOGIC); 17. end ALU; 18. architecture ALU\_Behavioral of ALU is 19. signal ALUR: STD\_LOGIC\_VECTOR(15 downto 0) := (others => '0'); 20. signal Carry: STD\_LOGIC := '0'; 21. begin 22. process(A, B, OP) 23. begin 24. case (OP) is 25. when "01" => ALUR <= ("00000000" & A) + ("00000000" & B); 26. when "10" => ALUR <= ("00000000" & A) + ("11111111" & not B) + "0000000000000001"; 27. when "11" => 28. case(B) is 29. when x"00" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 0); 30. when x"01" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 1); 31. when x"02" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 2); 32. when x"03" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 3); 33. when x"04" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 4); 34. when x"05" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 5); 35. when x"06" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 6); 36. when x"07" => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 7); 37. when others => ALUR <= std\_logic\_vector(unsigned(("00000000" & A)) sll 0); 38. end case; 39. when others => ALUR <= ("00000000" & B); 40. end case; 41. end process; 42. OUTPUT <= ALUR(7 downto 0); 43. OVERFLOW <= ALUR(8) OR ALUR(9) OR ALUR(10) OR ALUR(11) OR ALUR(12) OR ALUR(13) OR ALUR(14) OR ALUR(15); 44. end ALU\_Behavioral; |
| --- |

| Файл CPU.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. entity CPU is 4. port( ENTER\_OP1 : IN STD\_LOGIC; 5. ENTER\_OP2 : IN STD\_LOGIC; 6. CALCULATE : IN STD\_LOGIC; 7. RESET : IN STD\_LOGIC; 8. CLOCK : IN STD\_LOGIC; 9. RAM\_WR : OUT STD\_LOGIC; 10. RAM\_ADDR : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0); 11. CONST : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0); 12. ACC\_WR : OUT STD\_LOGIC; 13. ACC\_RST : OUT STD\_LOGIC; 14. IN\_SEL : OUT STD\_LOGIC\_VECTOR(1 downto 0); 15. OP : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0)); 16. end CPU; 18. architecture CPU\_arch of CPU is 19. type STATE\_TYPE is (RST, IDLE, LOAD\_OP1, LOAD\_OP2, RUN\_CALC0, RUN\_CALC1, RUN\_CALC2, RUN\_CALC3, RUN\_CALC4, FINISH); 20. signal CUR\_STATE : STATE\_TYPE; 21. signal NEXT\_STATE : STATE\_TYPE; 22. begin 23. SYNC\_PROC: process (CLOCK) 24. begin 25. if (rising\_edge(CLOCK)) then 26. if (RESET = '1') then 27. CUR\_STATE <= RST; 28. else 29. CUR\_STATE <= NEXT\_STATE; 30. end if; 31. end if; 32. end process;  35. NEXT\_STATE\_DECODE: process (CLOCK, ENTER\_OP1, ENTER\_OP2, CALCULATE) 36. begin 37. NEXT\_STATE <= CUR\_STATE; 39. case(CUR\_STATE) is 40. when RST => 41. NEXT\_STATE <= IDLE; 42. when IDLE => 43. if (ENTER\_OP1 = '1') then 44. NEXT\_STATE <= LOAD\_OP1; 45. elsif (ENTER\_OP2 = '1') then 46. NEXT\_STATE <= LOAD\_OP2; 47. elsif (CALCULATE = '1') then 48. NEXT\_STATE <= RUN\_CALC0; 49. else 50. NEXT\_STATE <= IDLE; 51. end if; 52. when LOAD\_OP1 => 53. NEXT\_STATE <= IDLE; 54. when LOAD\_OP2 => 55. NEXT\_STATE <= IDLE; 56. when RUN\_CALC0 => 57. NEXT\_STATE <= RUN\_CALC1; 58. when RUN\_CALC1 => 59. NEXT\_STATE <= RUN\_CALC2; 60. when RUN\_CALC2 => 61. NEXT\_STATE <= RUN\_CALC3; 62. when RUN\_CALC3 => 63. NEXT\_STATE <= RUN\_CALC4; 64. when RUN\_CALC4 => 65. NEXT\_STATE <= FINISH; 66. when FINISH => 67. NEXT\_STATE <= FINISH; 68. when others => 69. NEXT\_STATE <= IDLE; 70. end case; 71. end process; 72. OUTPUT\_DECODE: process (CUR\_STATE) 73. begin 74. case (CUR\_STATE) is 75. when RST => 76. RAM\_WR <= '0'; 77. RAM\_ADDR <= "00"; 78. CONST <= "00000000"; 79. ACC\_WR <= '0'; 80. ACC\_RST <= '1'; 81. IN\_SEL <= "00"; 82. OP <= "00"; 83. when LOAD\_OP1 => 84. RAM\_WR <= '1'; 85. RAM\_ADDR <= "00"; 86. CONST <= "00000000"; 87. ACC\_WR <= '0'; 88. ACC\_RST <= '1'; 89. IN\_SEL <= "00"; 90. OP <= "00"; 91. when LOAD\_OP2 => 92. RAM\_WR <= '1'; 93. RAM\_ADDR <= "01"; 94. CONST <= "00000000"; 95. ACC\_WR <= '0'; 96. ACC\_RST <= '1'; 97. IN\_SEL <= "00"; 98. OP <= "00"; 99. when RUN\_CALC0 => 100. RAM\_WR <= '0'; 101. RAM\_ADDR <= "01"; 102. CONST <= "00000000"; 103. ACC\_WR <= '1'; 104. ACC\_RST <= '0'; 105. IN\_SEL <= "01"; 106. OP <= "00"; 107. when RUN\_CALC1 => 108. RAM\_WR <= '0'; 109. RAM\_ADDR <= "00"; 110. CONST <= "00000000"; 111. ACC\_WR <= '1'; 112. ACC\_RST <= '0'; 113. IN\_SEL <= "01"; 114. OP <= "10"; 115. when RUN\_CALC2 => 116. RAM\_WR <= '0'; 117. RAM\_ADDR <= "00"; 118. CONST <= "00000001"; 119. ACC\_WR <= '1'; 120. ACC\_RST <= '0'; 121. IN\_SEL <= "10"; 122. OP <= "11"; 123. when RUN\_CALC3 => 124. RAM\_WR <= '0'; 125. RAM\_ADDR <= "00"; 126. CONST <= "00000000"; 127. ACC\_WR <= '1'; 128. ACC\_RST <= '0'; 129. IN\_SEL <= "01"; 130. OP <= "01"; 131. when RUN\_CALC4 => 132. RAM\_WR <= '0'; 133. RAM\_ADDR <= "00"; 134. CONST <= "00001010"; 135. ACC\_WR <= '1'; 136. ACC\_RST <= '0'; 137. IN\_SEL <= "10"; 138. OP <= "01"; 139. when IDLE => 140. RAM\_WR <= '0'; 141. RAM\_ADDR <= "00"; 142. CONST <= "00000000"; 143. ACC\_WR <= '0'; 144. ACC\_RST <= '0'; 145. IN\_SEL <= "00"; 146. OP <= "00"; 147. when others => 148. RAM\_WR <= '0'; 149. RAM\_ADDR <= "00"; 150. CONST <= "00000000"; 151. ACC\_WR <= '0'; 152. ACC\_RST <= '0'; 153. IN\_SEL <= "00"; 154. OP <= "00"; 155. end case; 156. end process; 157. end CPU\_arch; |
| --- |

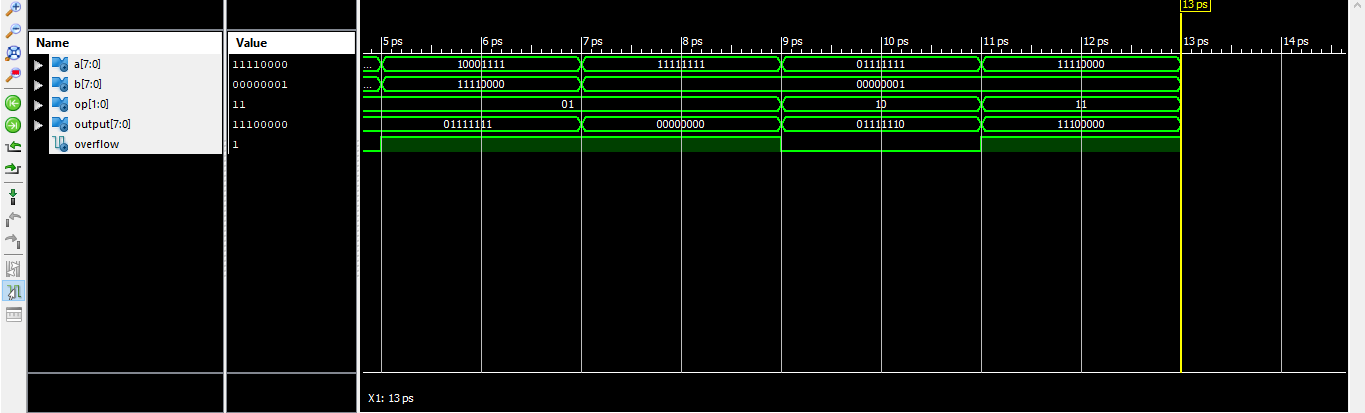
| Файл MUX.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. entity MUX is 4. PORT( 5. SEL: in STD\_LOGIC\_VECTOR(1 downto 0); 6. CONST: in STD\_LOGIC\_VECTOR(7 downto 0); 7. --CONST1: in STD\_LOGIC\_VECTOR() 8. DATA\_IN0: in STD\_LOGIC\_VECTOR(7 downto 0); 9. DATA\_IN1: in STD\_LOGIC\_VECTOR(7 downto 0); 10. OUTPUT: out STD\_LOGIC\_VECTOR(7 downto 0) 11. ); 12. end MUX; 13. architecture Behavioral of MUX is 14. begin 15. process (SEL, DATA\_IN0, DATA\_IN1, CONST) 16. begin 17. if (SEL = "00") then 18. OUTPUT <= DATA\_IN0; 19. elsif (SEL = "01") then 20. OUTPUT <= DATA\_IN1; 21. else 22. OUTPUT <= CONST; 23. end if; 24. end process; 25. end Behavioral; |
| --- |

| Файл RAM.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. use IEEE.NUMERIC\_STD.ALL; 4. use IEEE.STD\_LOGIC\_UNSIGNED.ALL; 5. entity RAM is 6. port( 7. WR : IN STD\_LOGIC; 8. ADDR : IN STD\_LOGIC\_VECTOR(1 DOWNTO 0); 9. DATA : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); 10. CLOCK: IN STD\_LOGIC; 11. OUTPUT : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) 12. ); 13. end RAM; 14. architecture RAM\_arch of RAM is 15. type ram\_type is array (3 downto 0) of STD\_LOGIC\_VECTOR(7 downto 0); 16. signal UNIT : ram\_type; 17. begin 18. process(ADDR, CLOCK, UNIT) 19. begin 20. if(rising\_edge(CLOCK)) then 21. if (WR = '1') then 22. UNIT(conv\_integer(ADDR)) <= DATA; 23. end if; 24. end if; 25. OUTPUT <= UNIT(conv\_integer(ADDR)); 26. end process; 27. end RAM\_arch; |
| --- |

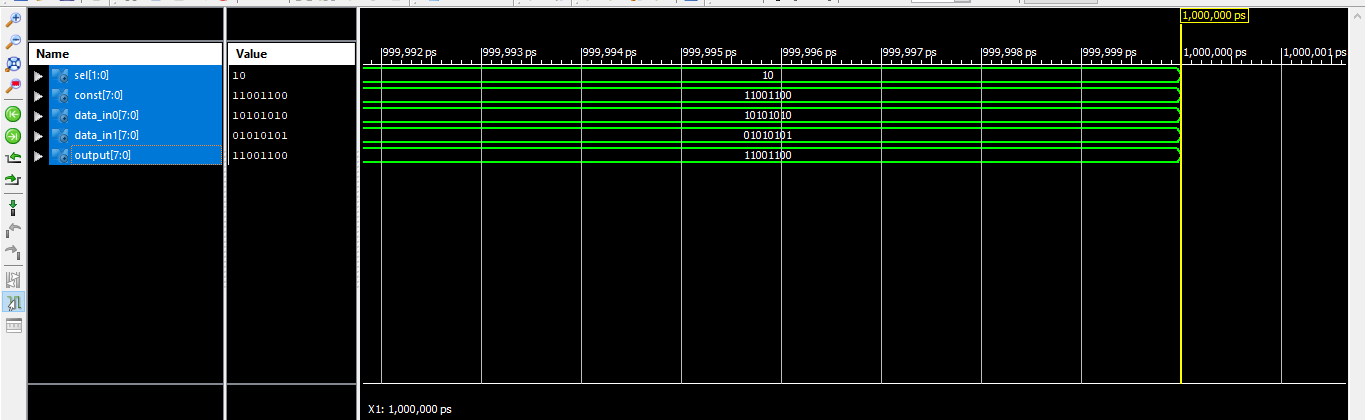
| Файл SEG\_DECODER.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. use IEEE.STD\_LOGIC\_ARITH.ALL; 4. use IEEE.STD\_LOGIC\_UNSIGNED.ALL; 5. entity SEG\_DECODER is 6. port( CLOCK : IN STD\_LOGIC; 7. RESET : IN STD\_LOGIC; 8. ACC\_DATA\_OUT\_BUS : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); 9. COMM\_ONES : OUT STD\_LOGIC; 10. COMM\_DECS : OUT STD\_LOGIC; 11. COMM\_HUNDREDS : OUT STD\_LOGIC; 12. SEG\_A : OUT STD\_LOGIC; 13. SEG\_B : OUT STD\_LOGIC; 14. SEG\_C : OUT STD\_LOGIC; 15. SEG\_D : OUT STD\_LOGIC; 16. SEG\_E : OUT STD\_LOGIC; 17. SEG\_F : OUT STD\_LOGIC; 18. SEG\_G : OUT STD\_LOGIC; 19. DP : OUT STD\_LOGIC); 20. end SEG\_DECODER; 21. architecture Behavioral of SEG\_DECODER is 22. signal ONES\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 23. signal DECS\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0001"; 24. signal HONDREDS\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 25. begin 26. BIN\_TO\_BCD : process (ACC\_DATA\_OUT\_BUS) 27. variable hex\_src : STD\_LOGIC\_VECTOR(7 downto 0) ; 28. variable bcd : STD\_LOGIC\_VECTOR(11 downto 0) ; 29. begin 30. bcd := (others => '0') ; 31. hex\_src := ACC\_DATA\_OUT\_BUS; 32. for i in hex\_src'range loop 33. if bcd(3 downto 0) > "0100" then 34. bcd(3 downto 0) := bcd(3 downto 0) + "0011" ; 35. end if ; 36. if bcd(7 downto 4) > "0100" then 37. bcd(7 downto 4) := bcd(7 downto 4) + "0011" ; 38. end if ; 39. if bcd(11 downto 8) > "0100" then 40. bcd(11 downto 8) := bcd(11 downto 8) + "0011" ; 41. end if ; 43. bcd := bcd(10 downto 0) & hex\_src(hex\_src'left) ; -- shift bcd + 1 new entry 44. hex\_src := hex\_src(hex\_src'left - 1 downto hex\_src'right) & '0' ; -- shift src + pad with 0 45. end loop ; 46. HONDREDS\_BUS <= bcd (11 downto 8); 47. DECS\_BUS <= bcd (7 downto 4); 48. ONES\_BUS <= bcd (3 downto 0); 50. end process BIN\_TO\_BCD; 52. INDICATE : process(CLOCK) 53. type DIGIT\_TYPE is (ONES, DECS, HUNDREDS); 55. variable CUR\_DIGIT : DIGIT\_TYPE := ONES; 56. variable DIGIT\_VAL : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 57. variable DIGIT\_CTRL : STD\_LOGIC\_VECTOR(6 downto 0) := "0000000"; 58. variable COMMONS\_CTRL : STD\_LOGIC\_VECTOR(2 downto 0) := "000"; 60. begin 61. if (rising\_edge(CLOCK)) then 62. if(RESET = '0') then 63. case CUR\_DIGIT is 64. when ONES => 65. DIGIT\_VAL := ONES\_BUS; 66. CUR\_DIGIT := DECS; 67. COMMONS\_CTRL := "001"; 68. when DECS => 69. DIGIT\_VAL := DECS\_BUS; 70. CUR\_DIGIT := HUNDREDS; 71. COMMONS\_CTRL := "010"; 72. when HUNDREDS => 73. DIGIT\_VAL := HONDREDS\_BUS; 74. CUR\_DIGIT := ONES; 75. COMMONS\_CTRL := "100"; 76. when others => 77. DIGIT\_VAL := ONES\_BUS; 78. CUR\_DIGIT := ONES; 79. COMMONS\_CTRL := "000"; 80. end case; 82. case DIGIT\_VAL is --abcdefg 83. when "0000" => DIGIT\_CTRL := "1111110"; 84. when "0001" => DIGIT\_CTRL := "0110000"; 85. when "0010" => DIGIT\_CTRL := "1101101"; 86. when "0011" => DIGIT\_CTRL := "1111001"; 87. when "0100" => DIGIT\_CTRL := "0110011"; 88. when "0101" => DIGIT\_CTRL := "1011011"; 89. when "0110" => DIGIT\_CTRL := "1011111"; 90. when "0111" => DIGIT\_CTRL := "1110000"; 91. when "1000" => DIGIT\_CTRL := "1111111"; 92. when "1001" => DIGIT\_CTRL := "1111011"; 93. when others => DIGIT\_CTRL := "0000000"; 94. end case; 95. else 96. DIGIT\_VAL := ONES\_BUS; 97. CUR\_DIGIT := ONES; 98. COMMONS\_CTRL := "000"; 99. end if; 101. COMM\_ONES <= not COMMONS\_CTRL(0); 102. COMM\_DECS <= not COMMONS\_CTRL(1); 103. COMM\_HUNDREDS <= not COMMONS\_CTRL(2); 105. SEG\_A <= not DIGIT\_CTRL(6); 106. SEG\_B <= not DIGIT\_CTRL(5); 107. SEG\_C <= not DIGIT\_CTRL(4); 108. SEG\_D <= not DIGIT\_CTRL(3); 109. SEG\_E <= not DIGIT\_CTRL(2); 110. SEG\_F <= not DIGIT\_CTRL(1); 111. SEG\_G <= not DIGIT\_CTRL(0); 112. DP <= '1'; 114. end if; 115. end process INDICATE; 116. end Behavioral; |
| --- |

**

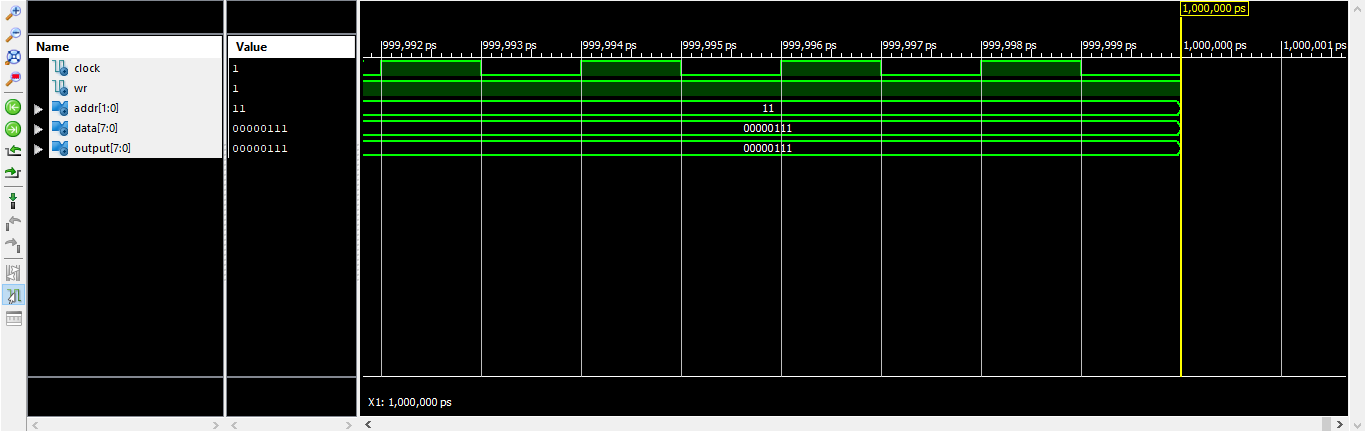
*Рис. 2 – Часова діаграма ACC*

**

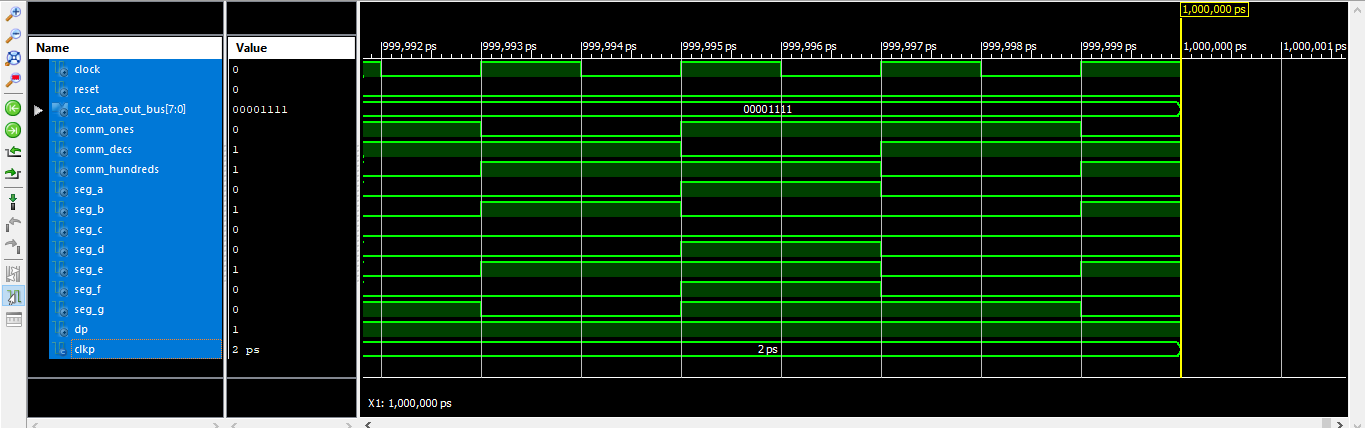
*Рис. 3 – Часова діаграма ALU*

**

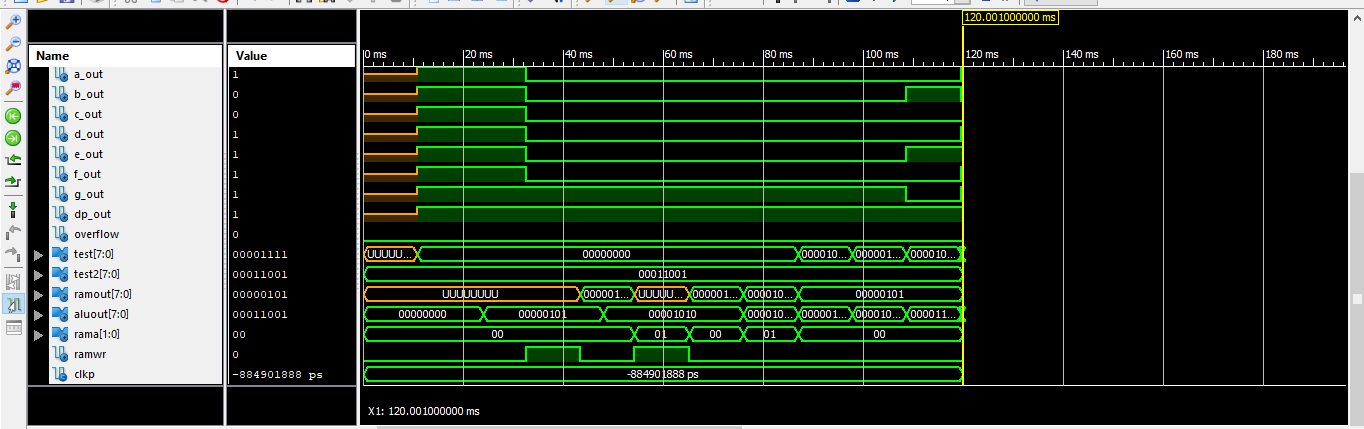
*Рис. 4 – Часова діаграма MUX*

**

*Рис. 5 – Часова діаграма RAM*

**

*Рис 6. – Часова діграма SEG\_DECODER*

**

*Рис 7. – Часова діграма TopLevel*

| *Файл TopLevelTest.vhd*   1. LIBRARY ieee; 2. USE ieee.std\_logic\_1164.ALL; 3. USE ieee.numeric\_std.ALL; 4. LIBRARY UNISIM; 5. USE UNISIM.Vcomponents.ALL; 6. ENTITY TopLevel\_TopLevel\_sch\_tb IS 7. END TopLevel\_TopLevel\_sch\_tb; 8. ARCHITECTURE behavioral OF TopLevel\_TopLevel\_sch\_tb IS 9. COMPONENT TopLevel 10. PORT( CLOCK : IN STD\_LOGIC; 11. RESET : IN STD\_LOGIC; 12. ENTER\_OP1 : IN STD\_LOGIC; 13. ENTER\_OP2 : IN STD\_LOGIC; 14. CALCULATE : IN STD\_LOGIC; 15. DATA\_IN : IN STD\_LOGIC\_VECTOR (7 DOWNTO 0); 16. COMMON\_0\_OUT : OUT STD\_LOGIC; 17. COMMON\_1\_OUT : OUT STD\_LOGIC; 18. COMMON\_2\_OUT : OUT STD\_LOGIC; 19. TEST: OUT STD\_LOGIC\_VECTOR(7 downto 0); 20. A\_OUT : OUT STD\_LOGIC; 21. B\_OUT : OUT STD\_LOGIC; 22. C\_OUT : OUT STD\_LOGIC; 23. D\_OUT : OUT STD\_LOGIC; 24. E\_OUT : OUT STD\_LOGIC; 25. F\_OUT : OUT STD\_LOGIC; 26. G\_OUT : OUT STD\_LOGIC; 27. DP\_OUT : OUT STD\_LOGIC; 28. RAMOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0); 29. ALUOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0); 30. RAMA: OUT STD\_LOGIC\_VECTOR(1 downto 0); 31. RAMWR: OUT STD\_LOGIC; 32. OVERFLOW : OUT STD\_LOGIC); 33. END COMPONENT; 34. SIGNAL CLOCK : STD\_LOGIC := '0'; 35. SIGNAL RESET : STD\_LOGIC; 36. SIGNAL ENTER\_OP1 : STD\_LOGIC; 37. SIGNAL ENTER\_OP2 : STD\_LOGIC; 38. SIGNAL CALCULATE : STD\_LOGIC; 39. SIGNAL DATA\_IN : STD\_LOGIC\_VECTOR (7 DOWNTO 0); 40. SIGNAL COMMON\_0\_OUT : STD\_LOGIC; 41. SIGNAL COMMON\_1\_OUT : STD\_LOGIC; 42. SIGNAL COMMON\_2\_OUT : STD\_LOGIC; 43. SIGNAL A\_OUT : STD\_LOGIC; 44. SIGNAL B\_OUT : STD\_LOGIC; 45. SIGNAL C\_OUT : STD\_LOGIC; 46. SIGNAL D\_OUT : STD\_LOGIC; 47. SIGNAL E\_OUT : STD\_LOGIC; 48. SIGNAL F\_OUT : STD\_LOGIC; 49. SIGNAL G\_OUT : STD\_LOGIC; 50. SIGNAL DP\_OUT : STD\_LOGIC; 51. SIGNAL OVERFLOW : STD\_LOGIC; 52. SIGNAL TEST: STD\_LOGIC\_VECTOR(7 downto 0); 53. SIGNAL TEST2: STD\_LOGIC\_VECTOR(7 downto 0); 54. signal RAMOUT: STD\_LOGIC\_VECTOR(7 downto 0); 55. signal ALUOUT: STD\_LOGIC\_VECTOR(7 downto 0); 56. signal RAMA: STD\_LOGIC\_VECTOR(1 downto 0); 57. signal RAMWR: STD\_LOGIC; 59. -- constant CLOCK\_period : time := 166ns; 60. constant CLKP: time := 12ms;--24ms; 61. BEGIN 62. UUT: TopLevel PORT MAP( 63. CLOCK => CLOCK, 64. RESET => RESET, 65. ENTER\_OP1 => ENTER\_OP1, 66. ENTER\_OP2 => ENTER\_OP2, 67. CALCULATE => CALCULATE, 68. DATA\_IN => DATA\_IN, 69. COMMON\_0\_OUT => COMMON\_0\_OUT, 70. COMMON\_1\_OUT => COMMON\_1\_OUT, 71. COMMON\_2\_OUT => COMMON\_2\_OUT, 72. A\_OUT => A\_OUT, 73. B\_OUT => B\_OUT, 74. C\_OUT => C\_OUT, 75. D\_OUT => D\_OUT, 76. E\_OUT => E\_OUT, 77. F\_OUT => F\_OUT, 78. G\_OUT => G\_OUT, 79. DP\_OUT => DP\_OUT, 80. OVERFLOW => OVERFLOW, 81. TEST => TEST, 82. RAMOUT => RAMOUT, 83. ALUOUT => ALUOUT, 84. RAMA => RAMA, 85. RAMWR => RAMWR 86. ); 88. CLOCK\_process: process 89. begin 90. CLOCK <= '0'; 91. wait for 83ns; 92. CLOCK <= '1'; 93. wait for 83ns; 94. end process; 95. -- \*\*\* Test Bench - User Defined Section \*\*\* 96. tb : PROCESS 97. BEGIN 98. lp1: for i in 2 to 2 loop 99. lp2: for j in 4 to 4 loop 100. TEST2 <= std\_logic\_vector((to\_signed(j - i, 8) sll 1) + i + 10); 101. ENTER\_OP1 <= '1'; 102. ENTER\_OP2 <= '1'; 103. CALCULATE <= '1'; 104. DATA\_IN <= (others => '0'); 105. RESET <= '0'; 106. wait for CLKP; 107. RESET <= '1'; 108. wait for CLKP; 109. DATA\_IN <= std\_logic\_vector(to\_unsigned(i, 8)); -- A 110. ENTER\_OP1 <= '0'; 111. wait for CLKP; 112. ENTER\_OP1 <= '1'; 113. wait for CLKP; 114. DATA\_IN <= std\_logic\_vector(to\_unsigned(j, 8)); -- B 115. ENTER\_OP2 <= '0'; 116. wait for CLKP; 117. ENTER\_OP2 <= '1'; 118. wait for CLKP; 119. CALCULATE <= '0'; -- START CALCULATION 120. wait for CLKP\* 7; 121. assert TEST = TEST2 severity FAILURE; 122. wait for CLKP; 123. end loop; 124. end loop; 126. WAIT; -- will wait forever 127. END PROCESS; 128. -- \*\*\* End Test Bench - User Defined Section \*\*\* 129. END; |
| --- |

*Зображення, що містить текст, знімок екрана, схема, Шрифт

Автоматично згенерований опис*

*Рис.8 – 7-сегментний індикатор*

Зображення, що містить текст, знімок екрана, монітор, програмне забезпечення

Автоматично згенерований опис

*Рис.9 – Успішна прошивка*

**Висновок:** Виконуючи дану лабораторну роботу я навчився реалізовувати цифровий автомат для обчислення значення виразів використовуючи засоби VHDL.