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 №3**

Виконав:

ст. гр. КІ-201

Бовтач П.В

Прийняв:

ст. викладач   
каф. ЕОМ  
Козак Н. Б.

Львів 2024

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

**Виконання роботи:**

**Зображення, що містить схема, текст, План, Креслення

Автоматично згенерований опис**

*Рис. 1 – Top Level*

|  |
| --- |
| Файл ACC.vhd   1. LIBRARY ieee; 2. USE ieee.std\_logic\_1164.ALL; 4. -- Uncomment the following library declaration if using 5. -- arithmetic functions with Signed or Unsigned values 6. --USE ieee.numeric\_std.ALL; 8. ENTITY ACCTest IS 9. END ACCTest; 11. ARCHITECTURE behavior OF ACCTest IS 13. -- Component Declaration for the Unit Under Test (UUT) 15. COMPONENT ACC 16. PORT( 17. WR : IN std\_logic; 18. RESET : IN std\_logic; 19. CLK : IN std\_logic; 20. INPUT : IN std\_logic\_vector(7 downto 0); 21. OUTPUT : OUT std\_logic\_vector(7 downto 0) 22. ); 23. END COMPONENT;  26. --Inputs 27. signal WR : std\_logic := '0'; 28. signal RESET : std\_logic := '0'; 29. signal CLK : std\_logic := '0'; 30. signal INPUT : std\_logic\_vector(7 downto 0) := (others => '0'); 32. --Outputs 33. signal OUTPUT : std\_logic\_vector(7 downto 0); 35. -- Clock period definitions 36. constant CLKP: time := 2 ps; 38. BEGIN 40. -- Instantiate the Unit Under Test (UUT) 41. uut: ACC PORT MAP ( 42. WR => WR, 43. RESET => RESET, 44. CLK => CLK, 45. INPUT => INPUT, 46. OUTPUT => OUTPUT 47. ); 49. -- Clock process definitions 50. CLK\_process :process 51. begin 52. CLK <= '0'; 53. wait for CLKP/2; 54. CLK <= '1'; 55. wait for CLKP/2; 56. end process;  59. -- Stimulus process 60. stim\_proc: process 61. begin 62. RESET <= '1'; 63. wait for 4 \* CLKP; 64. RESET <= '0'; 65. WR <= '0'; 66. INPUT <= "00001111"; 67. wait for CLKP; 68. assert OUTPUT = "00000000" severity failure; 69. WR <= '1'; 70. wait for CLKP; 71. assert OUTPUT = "00001111" severity failure; 72. INPUT <= "11110000"; 73. wait for CLKP; 74. assert OUTPUT = "11110000" severity failure; 76. wait; 77. end process; 79. END; |

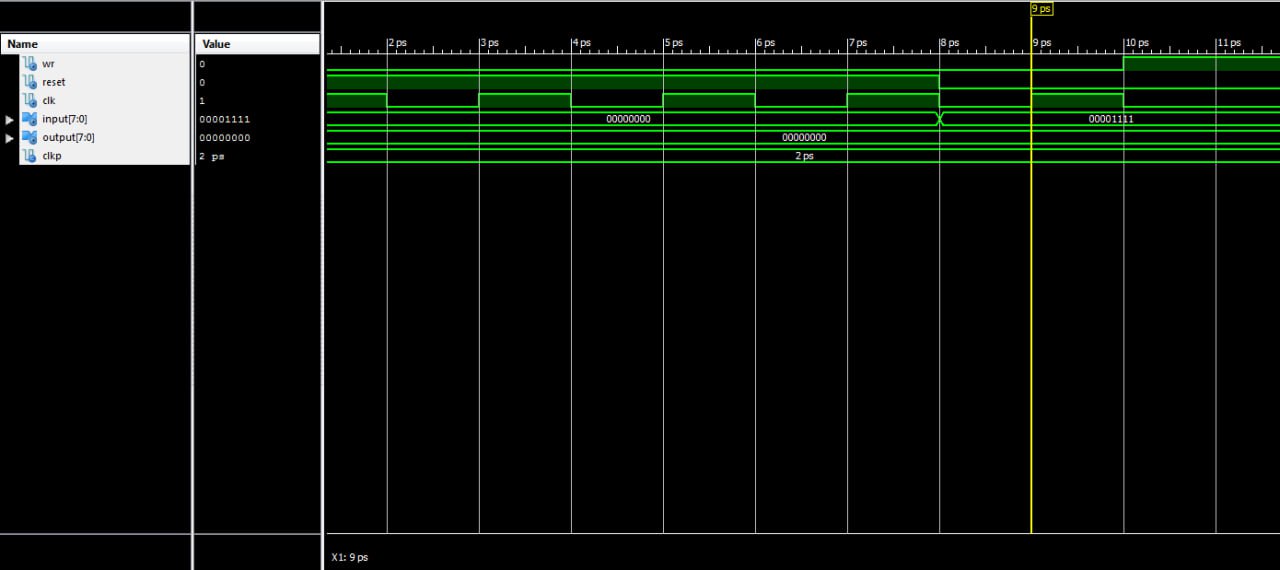
|  |
| --- |
| Файл ALU.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 4. -- Uncomment the following library declaration if using 5. -- arithmetic functions with Signed or Unsigned values 6. use IEEE.NUMERIC\_STD.ALL; 7. use IEEE.STD\_LOGIC\_UNSIGNED.ALL; 9. -- Uncomment the following library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

|  |
| --- |
| Файл CPU.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL;  5. entity CPU is 6. port( ENTER\_OP1 : IN STD\_LOGIC; 7. ENTER\_OP2 : IN STD\_LOGIC; 8. CALCULATE : IN STD\_LOGIC; 9. RESET : IN STD\_LOGIC; 10. CLOCK : IN STD\_LOGIC; 11. RAM\_WR : OUT STD\_LOGIC; 12. RAM\_ADDR : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0); 13. CONST : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0); 14. ACC\_WR : OUT STD\_LOGIC; 15. ACC\_RST : OUT STD\_LOGIC; 16. IN\_SEL : OUT STD\_LOGIC\_VECTOR(1 downto 0); 17. OP : OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0)); 18. end CPU; 20. architecture CPU\_arch of CPU is 22. type STATE\_TYPE is (RST, IDLE, LOAD\_OP1, LOAD\_OP2, RUN\_CALC0, RUN\_CALC1, RUN\_CALC2, RUN\_CALC3, RUN\_CALC4, FINISH); 23. signal CUR\_STATE : STATE\_TYPE; 24. signal NEXT\_STATE : STATE\_TYPE; 26. begin 27. SYNC\_PROC: process (CLOCK) 28. begin 29. if (rising\_edge(CLOCK)) then 30. if (RESET = '1') then 31. CUR\_STATE <= RST; 32. else 33. CUR\_STATE <= NEXT\_STATE; 34. end if; 35. end if; 36. end process;  39. NEXT\_STATE\_DECODE: process (CLOCK, ENTER\_OP1, ENTER\_OP2, CALCULATE) 40. begin 41. NEXT\_STATE <= CUR\_STATE; 43. case(CUR\_STATE) is 44. when RST => 45. NEXT\_STATE <= IDLE; 46. when IDLE => 47. if (ENTER\_OP1 = '1') then 48. NEXT\_STATE <= LOAD\_OP1; 49. elsif (ENTER\_OP2 = '1') then 50. NEXT\_STATE <= LOAD\_OP2; 51. elsif (CALCULATE = '1') then 52. NEXT\_STATE <= RUN\_CALC0; 53. else 54. NEXT\_STATE <= IDLE; 55. end if; 56. when LOAD\_OP1 => 57. NEXT\_STATE <= IDLE; 58. when LOAD\_OP2 => 59. NEXT\_STATE <= IDLE; 60. when RUN\_CALC0 => 61. NEXT\_STATE <= RUN\_CALC1; 62. when RUN\_CALC1 => 63. NEXT\_STATE <= RUN\_CALC2; 64. when RUN\_CALC2 => 65. NEXT\_STATE <= RUN\_CALC3; 66. when RUN\_CALC3 => 67. NEXT\_STATE <= RUN\_CALC4; 68. when RUN\_CALC4 => 69. NEXT\_STATE <= FINISH; 70. when FINISH => 71. NEXT\_STATE <= FINISH; 72. when others => 73. NEXT\_STATE <= IDLE; 74. end case; 75. end process; 77. OUTPUT\_DECODE: process (CUR\_STATE) 78. begin 79. case (CUR\_STATE) is 80. when RST => 81. RAM\_WR <= '0'; 82. RAM\_ADDR <= "00"; 83. CONST <= "00000000"; 84. ACC\_WR <= '0'; 85. ACC\_RST <= '1'; 86. IN\_SEL <= "00"; 87. OP <= "00"; 88. when LOAD\_OP1 => 89. RAM\_WR <= '1'; 90. RAM\_ADDR <= "00"; 91. CONST <= "00000000"; 92. ACC\_WR <= '0'; 93. ACC\_RST <= '1'; 94. IN\_SEL <= "00"; 95. OP <= "00"; 96. when LOAD\_OP2 => 97. RAM\_WR <= '1'; 98. RAM\_ADDR <= "01"; 99. CONST <= "00000000"; 100. ACC\_WR <= '0'; 101. ACC\_RST <= '1'; 102. IN\_SEL <= "00"; 103. OP <= "00"; 104. when RUN\_CALC0 => 105. RAM\_WR <= '0'; 106. RAM\_ADDR <= "01"; 107. CONST <= "00000000"; 108. ACC\_WR <= '1'; 109. ACC\_RST <= '0'; 110. IN\_SEL <= "01"; 111. OP <= "00"; 112. when RUN\_CALC1 => 113. RAM\_WR <= '0'; 114. RAM\_ADDR <= "00"; 115. CONST <= "00000101"; 116. ACC\_WR <= '1'; 117. ACC\_RST <= '0'; 118. IN\_SEL <= "10"; 119. OP <= "11"; 120. when RUN\_CALC2 => 121. RAM\_WR <= '0'; 122. RAM\_ADDR <= "01"; 123. CONST <= "00000000"; 124. ACC\_WR <= '1'; 125. ACC\_RST <= '0'; 126. IN\_SEL <= "01"; 127. OP <= "01"; 128. when RUN\_CALC3 => 129. RAM\_WR <= '0'; 130. RAM\_ADDR <= "00"; 131. CONST <= "00001010"; 132. ACC\_WR <= '1'; 133. ACC\_RST <= '0'; 134. IN\_SEL <= "10"; 135. OP <= "01"; 136. when RUN\_CALC4 => 137. RAM\_WR <= '0'; 138. RAM\_ADDR <= "00"; 139. CONST <= "00000000"; 140. ACC\_WR <= '1'; 141. ACC\_RST <= '0'; 142. IN\_SEL <= "01"; 143. OP <= "10"; 144. when IDLE => 145. RAM\_WR <= '0'; 146. RAM\_ADDR <= "00"; 147. CONST <= "00000000"; 148. ACC\_WR <= '0'; 149. ACC\_RST <= '0'; 150. IN\_SEL <= "00"; 151. OP <= "00"; 152. when others => 153. RAM\_WR <= '0'; 154. RAM\_ADDR <= "00"; 155. CONST <= "00000000"; 156. ACC\_WR <= '0'; 157. ACC\_RST <= '0'; 158. IN\_SEL <= "00"; 159. OP <= "00"; 160. end case; 161. end process; 162. end CPU\_arch; |

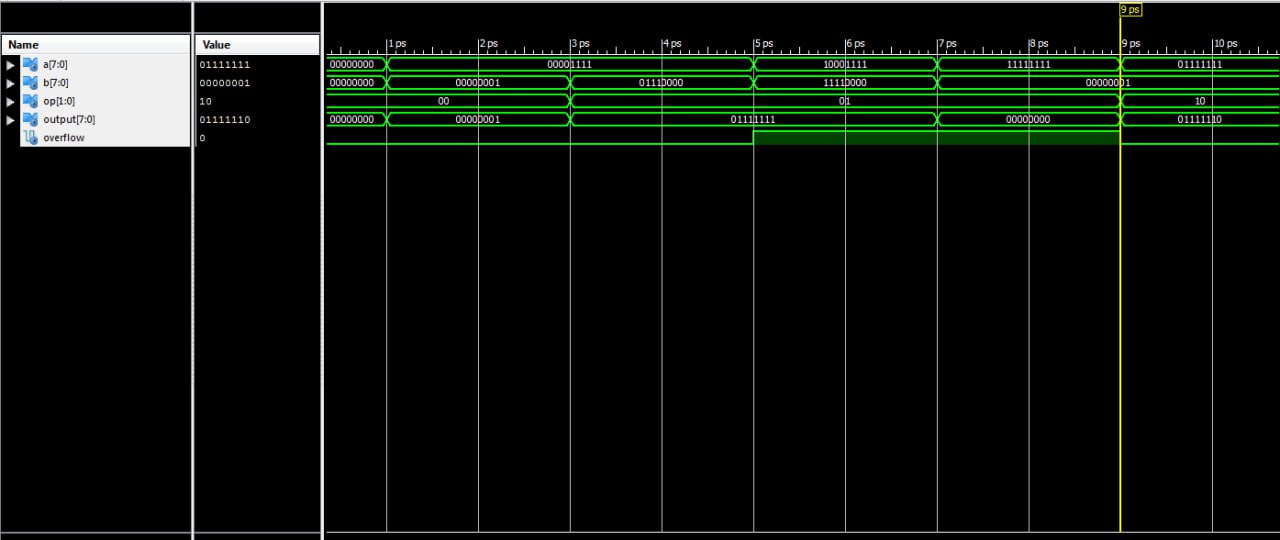
|  |
| --- |
| Файл MUX.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 4. entity MUX is 5. PORT( 6. SEL: in STD\_LOGIC\_VECTOR(1 downto 0); 7. CONST: in STD\_LOGIC\_VECTOR(7 downto 0); 8. --CONST1: in STD\_LOGIC\_VECTOR() 9. DATA\_IN0: in STD\_LOGIC\_VECTOR(7 downto 0); 10. DATA\_IN1: in STD\_LOGIC\_VECTOR(7 downto 0); 11. OUTPUT: out STD\_LOGIC\_VECTOR(7 downto 0) 12. ); 13. end MUX; 15. architecture Behavioral of MUX is 16. begin 17. process (SEL, DATA\_IN0, DATA\_IN1, CONST) 18. begin 19. if (SEL = "00") then 20. OUTPUT <= DATA\_IN0; 21. elsif (SEL = "01") then 22. OUTPUT <= DATA\_IN1; 23. else 24. OUTPUT <= CONST; 25. end if; 26. end process; 27. 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;  7. entity RAM is 8. port( 9. WR : IN STD\_LOGIC; 10. ADDR : IN STD\_LOGIC\_VECTOR(1 DOWNTO 0); 11. DATA : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); 12. CLOCK: IN STD\_LOGIC; 13. OUTPUT : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) 14. ); 15. end RAM; 17. architecture RAM\_arch of RAM is 18. type ram\_type is array (3 downto 0) of STD\_LOGIC\_VECTOR(7 downto 0); 19. signal UNIT : ram\_type; 21. begin 22. process(ADDR, CLOCK, UNIT) 23. begin 24. if(rising\_edge(CLOCK)) then 25. if (WR = '1') then 26. UNIT(conv\_integer(ADDR)) <= DATA; 27. end if; 28. end if; 29. OUTPUT <= UNIT(conv\_integer(ADDR)); 30. end process; 31. end RAM\_arch; |

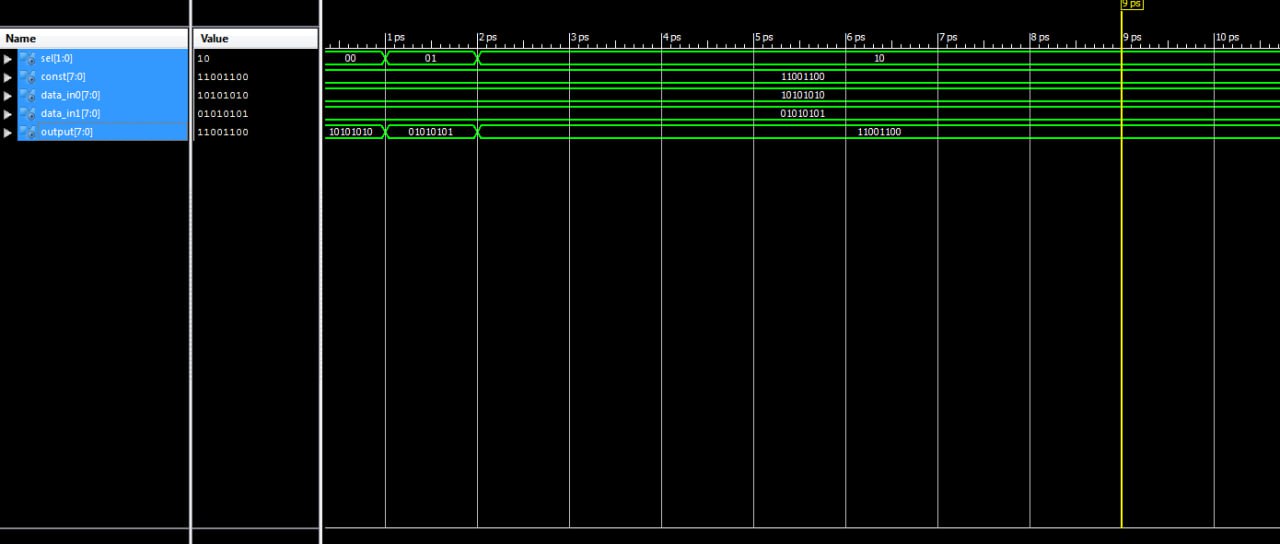
|  |
| --- |
| Файл SEG\_DECODER.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. use IEEE.STD\_LOGIC\_ARITH.ALL; 4. use IEEE.STD\_LOGIC\_UNSIGNED.ALL;  7. entity SEG\_DECODER is 8. port( CLOCK : IN STD\_LOGIC; 9. RESET : IN STD\_LOGIC; 10. ACC\_DATA\_OUT\_BUS : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); 11. COMM\_ONES : OUT STD\_LOGIC; 12. COMM\_DECS : OUT STD\_LOGIC; 13. COMM\_HUNDREDS : OUT STD\_LOGIC; 14. SEG\_A : OUT STD\_LOGIC; 15. SEG\_B : OUT STD\_LOGIC; 16. SEG\_C : OUT STD\_LOGIC; 17. SEG\_D : OUT STD\_LOGIC; 18. SEG\_E : OUT STD\_LOGIC; 19. SEG\_F : OUT STD\_LOGIC; 20. SEG\_G : OUT STD\_LOGIC; 21. DP : OUT STD\_LOGIC); 22. end SEG\_DECODER; 24. architecture Behavioral of SEG\_DECODER is 26. signal ONES\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 27. signal DECS\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0001"; 28. signal HONDREDS\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 30. begin 31. BIN\_TO\_BCD : process (ACC\_DATA\_OUT\_BUS) 32. variable hex\_src : STD\_LOGIC\_VECTOR(7 downto 0) ; 33. variable bcd : STD\_LOGIC\_VECTOR(11 downto 0) ; 34. begin 35. bcd := (others => '0') ; 36. hex\_src := ACC\_DATA\_OUT\_BUS; 38. for i in hex\_src'range loop 39. if bcd(3 downto 0) > "0100" then 40. bcd(3 downto 0) := bcd(3 downto 0) + "0011" ; 41. end if ; 42. if bcd(7 downto 4) > "0100" then 43. bcd(7 downto 4) := bcd(7 downto 4) + "0011" ; 44. end if ; 45. if bcd(11 downto 8) > "0100" then 46. bcd(11 downto 8) := bcd(11 downto 8) + "0011" ; 47. end if ; 49. bcd := bcd(10 downto 0) & hex\_src(hex\_src'left) ; -- shift bcd + 1 new entry 50. hex\_src := hex\_src(hex\_src'left - 1 downto hex\_src'right) & '0' ; -- shift src + pad with 0 51. end loop ; 53. HONDREDS\_BUS <= bcd (11 downto 8); 54. DECS\_BUS <= bcd (7 downto 4); 55. ONES\_BUS <= bcd (3 downto 0); 57. end process BIN\_TO\_BCD; 59. INDICATE : process(CLOCK) 60. type DIGIT\_TYPE is (ONES, DECS, HUNDREDS); 62. variable CUR\_DIGIT : DIGIT\_TYPE := ONES; 63. variable DIGIT\_VAL : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 64. variable DIGIT\_CTRL : STD\_LOGIC\_VECTOR(6 downto 0) := "0000000"; 65. variable COMMONS\_CTRL : STD\_LOGIC\_VECTOR(2 downto 0) := "000"; 67. begin 68. if (rising\_edge(CLOCK)) then 69. if(RESET = '0') then 70. case CUR\_DIGIT is 71. when ONES => 72. DIGIT\_VAL := ONES\_BUS; 73. CUR\_DIGIT := DECS; 74. COMMONS\_CTRL := "001"; 75. when DECS => 76. DIGIT\_VAL := DECS\_BUS; 77. CUR\_DIGIT := HUNDREDS; 78. COMMONS\_CTRL := "010"; 79. when HUNDREDS => 80. DIGIT\_VAL := HONDREDS\_BUS; 81. CUR\_DIGIT := ONES; 82. COMMONS\_CTRL := "100"; 83. when others => 84. DIGIT\_VAL := ONES\_BUS; 85. CUR\_DIGIT := ONES; 86. COMMONS\_CTRL := "000"; 87. end case; 89. case DIGIT\_VAL is --abcdefg 90. when "0000" => DIGIT\_CTRL := "1111110"; 91. when "0001" => DIGIT\_CTRL := "0110000"; 92. when "0010" => DIGIT\_CTRL := "1101101"; 93. when "0011" => DIGIT\_CTRL := "1111001"; 94. when "0100" => DIGIT\_CTRL := "0110011"; 95. when "0101" => DIGIT\_CTRL := "1011011"; 96. when "0110" => DIGIT\_CTRL := "1011111"; 97. when "0111" => DIGIT\_CTRL := "1110000"; 98. when "1000" => DIGIT\_CTRL := "1111111"; 99. when "1001" => DIGIT\_CTRL := "1111011"; 100. when others => DIGIT\_CTRL := "0000000"; 101. end case; 102. else 103. DIGIT\_VAL := ONES\_BUS; 104. CUR\_DIGIT := ONES; 105. COMMONS\_CTRL := "000"; 106. end if; 108. COMM\_ONES <= not COMMONS\_CTRL(0); 109. COMM\_DECS <= not COMMONS\_CTRL(1); 110. COMM\_HUNDREDS <= not COMMONS\_CTRL(2); 112. SEG\_A <= not DIGIT\_CTRL(6); 113. SEG\_B <= not DIGIT\_CTRL(5); 114. SEG\_C <= not DIGIT\_CTRL(4); 115. SEG\_D <= not DIGIT\_CTRL(3); 116. SEG\_E <= not DIGIT\_CTRL(2); 117. SEG\_F <= not DIGIT\_CTRL(1); 118. SEG\_G <= not DIGIT\_CTRL(0); 119. DP <= '1'; 121. end if; 122. end process INDICATE; 124. end Behavioral; |



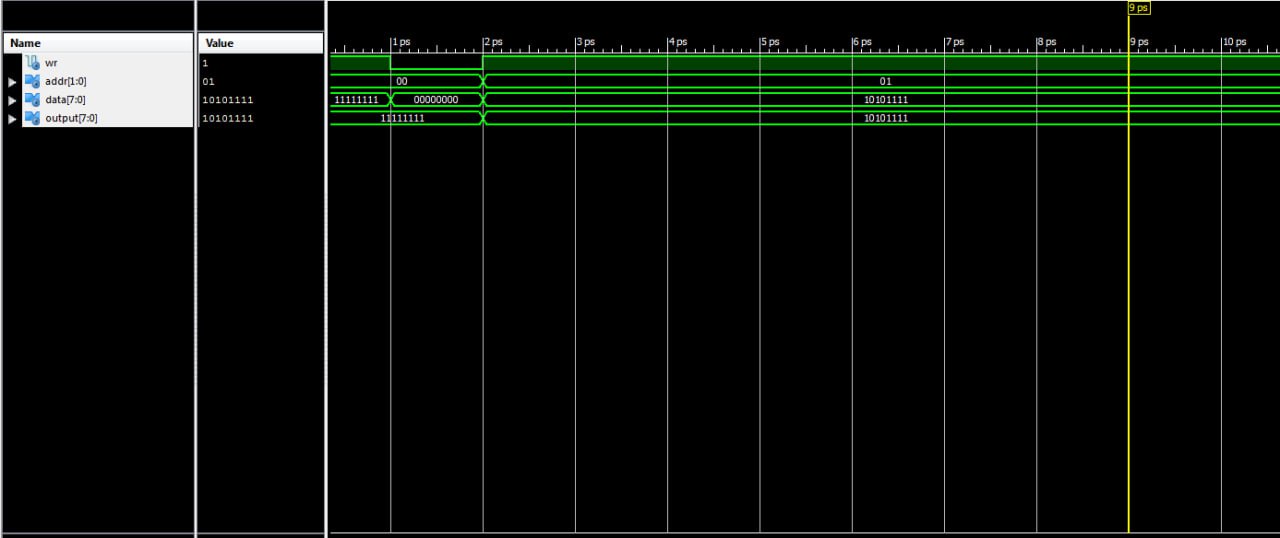
*Рис. 2 – Часова діаграма ACC*

**

*Рис. 3 – Часова діаграма ALU*

**

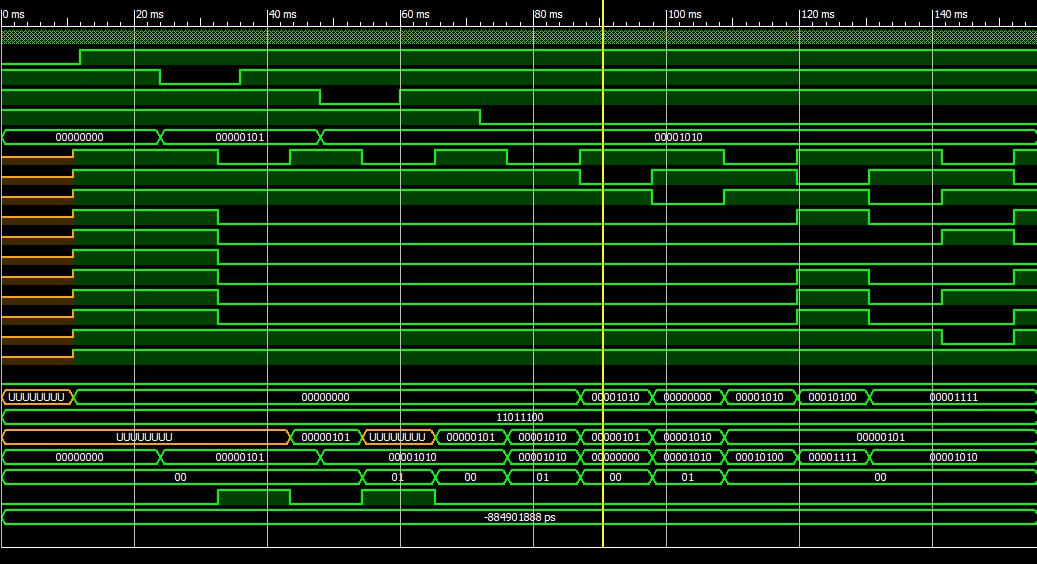
*Рис. 4 – Часова діаграма MUX*

**

*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 10. COMPONENT TopLevel 11. PORT( CLOCK : IN STD\_LOGIC; 12. RESET : IN STD\_LOGIC; 13. ENTER\_OP1 : IN STD\_LOGIC; 14. ENTER\_OP2 : IN STD\_LOGIC; 15. CALCULATE : IN STD\_LOGIC; 16. DATA\_IN : IN STD\_LOGIC\_VECTOR (7 DOWNTO 0); 17. COMMON\_0\_OUT : OUT STD\_LOGIC; 18. COMMON\_1\_OUT : OUT STD\_LOGIC; 19. COMMON\_2\_OUT : OUT STD\_LOGIC; 20. TEST: OUT STD\_LOGIC\_VECTOR(7 downto 0); 21. A\_OUT : OUT STD\_LOGIC; 22. B\_OUT : OUT STD\_LOGIC; 23. C\_OUT : OUT STD\_LOGIC; 24. D\_OUT : OUT STD\_LOGIC; 25. E\_OUT : OUT STD\_LOGIC; 26. F\_OUT : OUT STD\_LOGIC; 27. G\_OUT : OUT STD\_LOGIC; 28. DP\_OUT : OUT STD\_LOGIC; 29. RAMOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0); 30. ALUOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0); 31. RAMA: OUT STD\_LOGIC\_VECTOR(1 downto 0); 32. RAMWR: OUT STD\_LOGIC; 33. OVERFLOW : OUT STD\_LOGIC); 34. END COMPONENT; 36. SIGNAL CLOCK : STD\_LOGIC := '0'; 37. SIGNAL RESET : STD\_LOGIC; 38. SIGNAL ENTER\_OP1 : STD\_LOGIC; 39. SIGNAL ENTER\_OP2 : STD\_LOGIC; 40. SIGNAL CALCULATE : STD\_LOGIC; 41. SIGNAL DATA\_IN : STD\_LOGIC\_VECTOR (7 DOWNTO 0); 42. SIGNAL COMMON\_0\_OUT : STD\_LOGIC; 43. SIGNAL COMMON\_1\_OUT : STD\_LOGIC; 44. SIGNAL COMMON\_2\_OUT : STD\_LOGIC; 45. SIGNAL A\_OUT : STD\_LOGIC; 46. SIGNAL B\_OUT : STD\_LOGIC; 47. SIGNAL C\_OUT : STD\_LOGIC; 48. SIGNAL D\_OUT : STD\_LOGIC; 49. SIGNAL E\_OUT : STD\_LOGIC; 50. SIGNAL F\_OUT : STD\_LOGIC; 51. SIGNAL G\_OUT : STD\_LOGIC; 52. SIGNAL DP\_OUT : STD\_LOGIC; 53. SIGNAL OVERFLOW : STD\_LOGIC; 54. SIGNAL TEST: STD\_LOGIC\_VECTOR(7 downto 0); 55. SIGNAL TEST2: STD\_LOGIC\_VECTOR(7 downto 0); 56. signal RAMOUT: STD\_LOGIC\_VECTOR(7 downto 0); 57. signal ALUOUT: STD\_LOGIC\_VECTOR(7 downto 0); 58. signal RAMA: STD\_LOGIC\_VECTOR(1 downto 0); 59. signal RAMWR: STD\_LOGIC; 61. -- constant CLOCK\_period : time := 166ns; 62. constant CLKP: time := 12ms;--24ms; 64. BEGIN 66. UUT: TopLevel PORT MAP( 67. CLOCK => CLOCK, 68. RESET => RESET, 69. ENTER\_OP1 => ENTER\_OP1, 70. ENTER\_OP2 => ENTER\_OP2, 71. CALCULATE => CALCULATE, 72. DATA\_IN => DATA\_IN, 73. COMMON\_0\_OUT => COMMON\_0\_OUT, 74. COMMON\_1\_OUT => COMMON\_1\_OUT, 75. COMMON\_2\_OUT => COMMON\_2\_OUT, 76. A\_OUT => A\_OUT, 77. B\_OUT => B\_OUT, 78. C\_OUT => C\_OUT, 79. D\_OUT => D\_OUT, 80. E\_OUT => E\_OUT, 81. F\_OUT => F\_OUT, 82. G\_OUT => G\_OUT, 83. DP\_OUT => DP\_OUT, 84. OVERFLOW => OVERFLOW, 85. TEST => TEST, 86. RAMOUT => RAMOUT, 87. ALUOUT => ALUOUT, 88. RAMA => RAMA, 89. RAMWR => RAMWR 90. ); 92. CLOCK\_process: process 93. begin 94. CLOCK <= '0'; 95. wait for 83ns; 96. CLOCK <= '1'; 97. wait for 83ns; 98. end process; 100. -- \*\*\* Test Bench - User Defined Section \*\*\* 101. tb : PROCESS 102. BEGIN 103. lp1: for i in 5 to 5 loop 104. lp2: for j in 10 to 10 loop 105. TEST2 <= std\_logic\_vector(to\_unsigned(to\_integer(to\_unsigned(i, 8) sll 2) \* j + j + 10, 8)); 106. ENTER\_OP1 <= '1'; 107. ENTER\_OP2 <= '1'; 108. CALCULATE <= '1'; 109. DATA\_IN <= (others => '0'); 110. RESET <= '0'; 111. wait for CLKP; 112. RESET <= '1'; 113. wait for CLKP; 114. DATA\_IN <= std\_logic\_vector(to\_unsigned(i, 8)); -- A 115. ENTER\_OP1 <= '0'; 116. wait for CLKP; 117. ENTER\_OP1 <= '1'; 118. wait for CLKP; 119. DATA\_IN <= std\_logic\_vector(to\_unsigned(j, 8)); -- B 120. ENTER\_OP2 <= '0'; 121. wait for CLKP; 122. ENTER\_OP2 <= '1'; 123. wait for CLKP; 124. CALCULATE <= '0'; -- START CALCULATION 125. wait for CLKP\* 7; 126. assert TEST = TEST2 severity FAILURE; 127. wait for CLKP; 128. end loop; 129. end loop; 131. WAIT; -- will wait forever 132. END PROCESS; 133. -- \*\*\* End Test Bench - User Defined Section \*\*\* 135. 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.