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 №2**

Виконав:

ст. гр. КІ-201

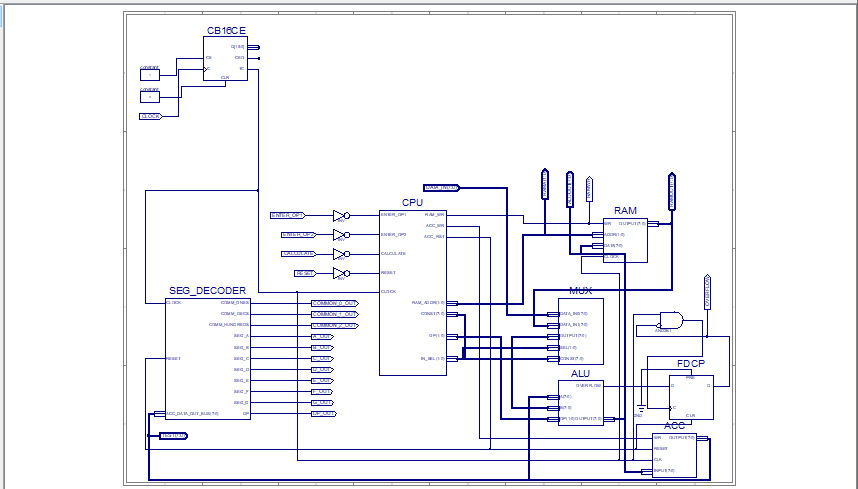
Бовтач О.М

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. --------------------------------------------------------------------------------- 2. -- Company: 3. -- Engineer: 4. -- 5. -- Create Date: 17:16:45 04/29/2024 6. -- Design Name: 7. -- Module Name: ACC - Behavioral 8. -- Project Name: 9. -- Target Devices: 10. -- Tool versions: 11. -- Description: 12. -- 13. -- Dependencies: 14. -- 15. -- Revision: 16. -- Revision 0.01 - File Created 17. -- Additional Comments: 18. -- 19. ---------------------------------------------------------------------------------- 20. library IEEE; 21. use IEEE.STD\_LOGIC\_1164.ALL; 22. -- Uncomment the following library declaration if using 23. -- arithmetic functions with Signed or Unsigned values 24. --use IEEE.NUMERIC\_STD.ALL; 25. -- Uncomment the following library declaration if instantiating 26. -- any Xilinx primitives in this code. 27. --library UNISIM; 28. --use UNISIM.VComponents.all; 29. entity ACC is 30. Port ( WR : in STD\_LOGIC; 31. RESET : in STD\_LOGIC; 32. CLK : in STD\_LOGIC; 33. INPUT : in STD\_LOGIC\_VECTOR (7 downto 0); 34. OUTPUT : out STD\_LOGIC\_VECTOR (7 downto 0)); 35. end ACC; 36. architecture ACC\_arch of ACC is 37. signal DATA : STD\_LOGIC\_VECTOR (7 downto 0); 38. begin 39. process (CLK) 40. begin 41. if rising\_edge(CLK) then 42. if RESET = '1' then 43. DATA <= (others => '0'); 44. elsif WR = '1' then 45. DATA <= INPUT; 46. end if; 47. end if; 48. end process; 50. OUTPUT <= DATA; 51. end ACC\_arch; |

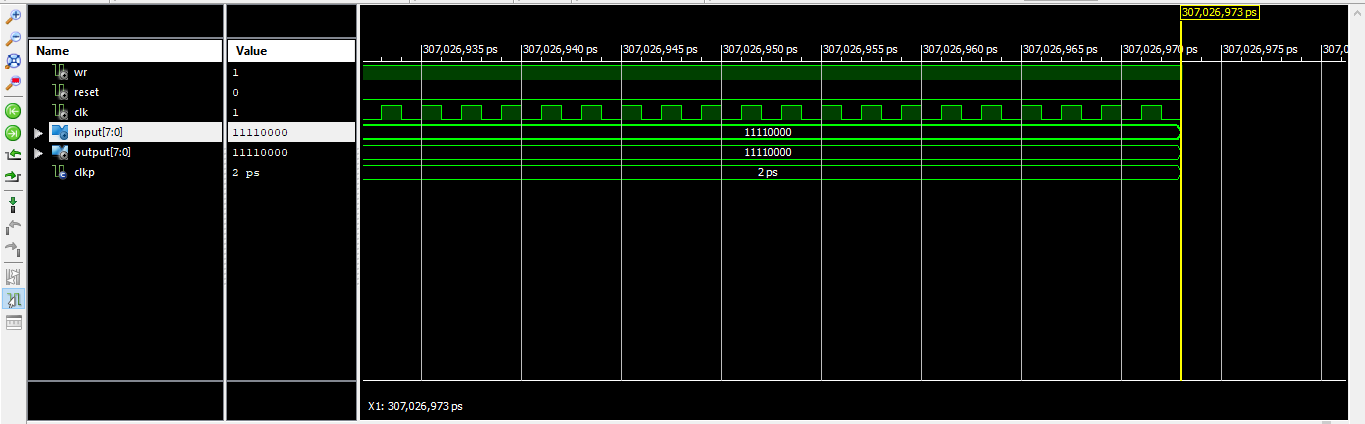
|  |
| --- |
| Файл ALU.vhd   1. - Company: 2. -- Engineer: 3. -- 4. -- Create Date: 11:09:43 05/07/2024 5. -- Design Name: 6. -- Module Name: C:/Users/User/Documents/Lab\_3\_v6/ACCTest.vhd 7. -- Project Name: Lab\_3\_Example 8. -- Target Device: 9. -- Tool versions: 10. -- Description: 11. -- 12. -- VHDL Test Bench Created by ISE for module: ACC 13. -- 14. -- Dependencies: 15. -- 16. -- Revision: 17. -- Revision 0.01 - File Created 18. -- Additional Comments: 19. -- 20. -- Notes: 21. -- This testbench has been automatically generated using types std\_logic and 22. -- std\_logic\_vector for the ports of the unit under test. Xilinx recommends 23. -- that these types always be used for the top-level I/O of a design in order 24. -- to guarantee that the testbench will bind correctly to the post-implementation 25. -- simulation model. 26. -------------------------------------------------------------------------------- 27. LIBRARY ieee; 28. USE ieee.std\_logic\_1164.ALL; 30. -- Uncomment the following library declaration if using 31. -- arithmetic functions with Signed or Unsigned values 32. --USE ieee.numeric\_std.ALL; 34. ENTITY ACCTest IS 35. END ACCTest; 37. ARCHITECTURE behavior OF ACCTest IS 39. -- Component Declaration for the Unit Under Test (UUT) 41. COMPONENT ACC 42. PORT( 43. WR : IN std\_logic; 44. RESET : IN std\_logic; 45. CLK : IN std\_logic; 46. INPUT : IN std\_logic\_vector(7 downto 0); 47. OUTPUT : OUT std\_logic\_vector(7 downto 0) 48. ); 49. END COMPONENT; 51. --Inputs 52. signal WR : std\_logic := '0'; 53. signal RESET : std\_logic := '0'; 54. signal CLK : std\_logic := '0'; 55. signal INPUT : std\_logic\_vector(7 downto 0) := (others => '0'); 56. --Outputs 57. signal OUTPUT : std\_logic\_vector(7 downto 0); 58. -- Clock period definitions 59. constant CLKP: time := 2 ps; 61. BEGIN 63. -- Instantiate the Unit Under Test (UUT) 64. uut: ACC PORT MAP ( 65. WR => WR, 66. RESET => RESET, 67. CLK => CLK, 68. INPUT => INPUT, 69. OUTPUT => OUTPUT 70. ); 71. -- Clock process definitions 72. CLK\_process :process 73. begin 74. CLK <= '0'; 75. wait for CLKP/2; 76. CLK <= '1'; 77. wait for CLKP/2; 78. end process; 80. -- Stimulus process 81. stim\_proc: process 82. begin 83. RESET <= '1'; 84. wait for 4 \* CLKP; 85. RESET <= '0'; 86. WR <= '0'; 87. INPUT <= "00001111"; 88. wait for CLKP; 89. assert OUTPUT = "00000000" severity failure; 90. WR <= '1'; 91. wait for CLKP; 92. assert OUTPUT = "00001111" severity failure; 93. INPUT <= "11110000"; 94. wait for CLKP; 95. assert OUTPUT = "11110000" severity failure; 97. wait; 98. end process; 99. END; |

|  |
| --- |
| 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 <= "00"; 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 <= "01"; 110. CONST <= "00000000"; 111. ACC\_WR <= '1'; 112. ACC\_RST <= '0'; 113. IN\_SEL <= "01"; 114. OP <= "11"; 115. when RUN\_CALC2 => 116. RAM\_WR <= '0'; 117. RAM\_ADDR <= "01"; 118. CONST <= "00000000"; 119. ACC\_WR <= '1'; 120. ACC\_RST <= '0'; 121. IN\_SEL <= "01"; 122. OP <= "01"; 123. when RUN\_CALC3 => 124. RAM\_WR <= '0'; 125. RAM\_ADDR <= "01"; 126. CONST <= "00001010"; 127. ACC\_WR <= '1'; 128. ACC\_RST <= '0'; 129. IN\_SEL <= "10"; 130. OP <= "01"; 131. when RUN\_CALC4 => 132. RAM\_WR <= '0'; 133. RAM\_ADDR <= "00"; 134. CONST <= "00000011"; 135. ACC\_WR <= '1'; 136. ACC\_RST <= '0'; 137. IN\_SEL <= "10"; 138. OP <= "10"; 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; |

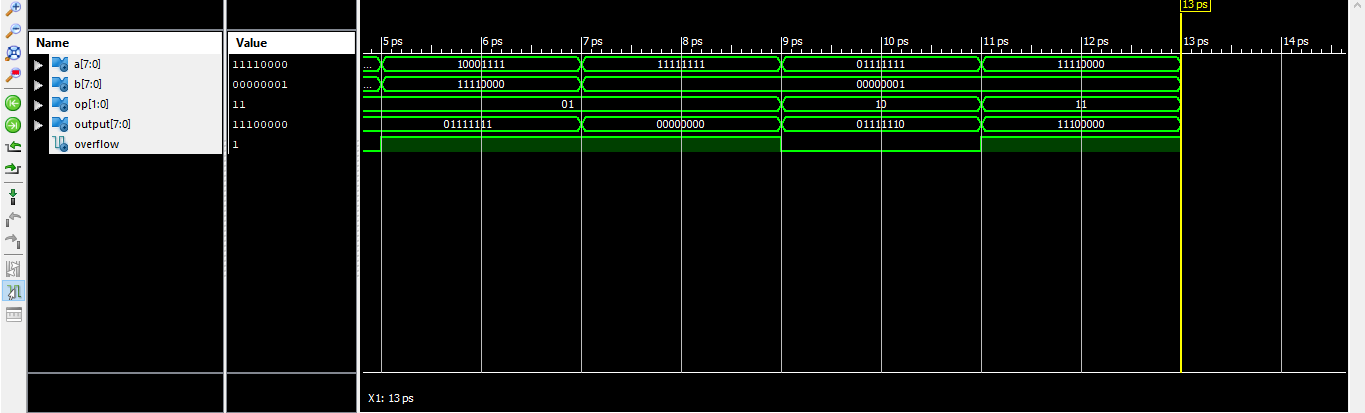
|  |
| --- |
| Файл MUX.vhd   1. library IEEE; 2. use IEEE.STD\_LOGIC\_1164.ALL; 3. entity MUX is 4. PORT( 5. SEL: in STD\_LOGIC\_VECTOR(1 downto 0); 6. CONST: in STD\_LOGIC\_VECTOR(7 downto 0); 7. --CONST1: in STD\_LOGIC\_VECTOR() 8. DATA\_IN0: in STD\_LOGIC\_VECTOR(7 downto 0); 9. DATA\_IN1: in STD\_LOGIC\_VECTOR(7 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. ---------------------------------------------------------------------------------- 2. -- Company: 3. -- Engineer: 4. -- 5. -- Create Date: 02:03:24 04/21/2024 6. -- Design Name: 7. -- Module Name: RAM - RAM\_arch 8. -- Project Name: 9. -- Target Devices: 10. -- Tool versions: 11. -- Description: 12. -- 13. -- Dependencies: 14. -- 15. -- Revision: 16. -- Revision 0.01 - File Created 17. -- Additional Comments: 18. -- 19. ---------------------------------------------------------------------------------- 20. library IEEE; 21. use IEEE.STD\_LOGIC\_1164.ALL; 22. use IEEE.NUMERIC\_STD.ALL; 23. use IEEE.STD\_LOGIC\_UNSIGNED.ALL; 24. entity RAM is 25. port( 26. WR : IN STD\_LOGIC; 27. ADDR : IN STD\_LOGIC\_VECTOR(1 DOWNTO 0); 28. DATA : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); 29. CLOCK: IN STD\_LOGIC; 30. OUTPUT : OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0) 31. ); 32. end RAM; 33. architecture RAM\_arch of RAM is 34. type ram\_type is array (3 downto 0) of STD\_LOGIC\_VECTOR(7 downto 0); 35. signal UNIT : ram\_type; 36. begin 37. process(ADDR, CLOCK, UNIT) 38. begin 39. if(rising\_edge(CLOCK)) then 40. if (WR = '1') then 41. UNIT(conv\_integer(ADDR)) <= DATA; 42. end if; 43. end if; 44. OUTPUT <= UNIT(conv\_integer(ADDR)); 45. end process; 46. end RAM\_arch; |

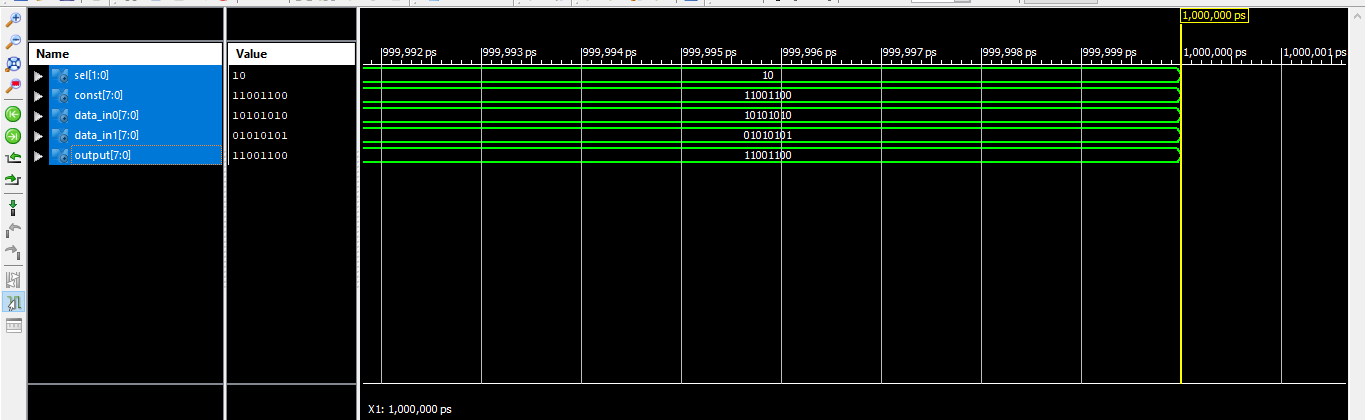
|  |
| --- |
| Файл SEG\_DECODER.vhd   1. ---------------------------------------------------------------------------------- 2. -- Company: 3. -- Engineer: 4. -- 5. -- Create Date: 02:34:19 04/21/2024 6. -- Design Name: 7. -- Module Name: BIN\_TO\_BCD - Behavioral 8. -- Project Name: 9. -- Target Devices: 10. -- Tool versions: 11. -- Description: 12. -- 13. -- Dependencies: 14. -- 15. -- Revision: 16. -- Revision 0.01 - File Created 17. -- Additional Comments: 18. -- 19. ---------------------------------------------------------------------------------- 20. library IEEE; 21. use IEEE.STD\_LOGIC\_1164.ALL; 22. use IEEE.STD\_LOGIC\_ARITH.ALL; 23. use IEEE.STD\_LOGIC\_UNSIGNED.ALL; 24. entity SEG\_DECODER is 25. port( CLOCK : IN STD\_LOGIC; 26. RESET : IN STD\_LOGIC; 27. ACC\_DATA\_OUT\_BUS : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0); 28. COMM\_ONES : OUT STD\_LOGIC; 29. COMM\_DECS : OUT STD\_LOGIC; 30. COMM\_HUNDREDS : OUT STD\_LOGIC; 31. SEG\_A : OUT STD\_LOGIC; 32. SEG\_B : OUT STD\_LOGIC; 33. SEG\_C : OUT STD\_LOGIC; 34. SEG\_D : OUT STD\_LOGIC; 35. SEG\_E : OUT STD\_LOGIC; 36. SEG\_F : OUT STD\_LOGIC; 37. SEG\_G : OUT STD\_LOGIC; 38. DP : OUT STD\_LOGIC); 39. end SEG\_DECODER; 40. architecture Behavioral of SEG\_DECODER is 41. signal ONES\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 42. signal DECS\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0001"; 43. signal HONDREDS\_BUS : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 44. begin 45. BIN\_TO\_BCD : process (ACC\_DATA\_OUT\_BUS) 46. variable hex\_src : STD\_LOGIC\_VECTOR(7 downto 0) ; 47. variable bcd : STD\_LOGIC\_VECTOR(11 downto 0) ; 48. begin 49. bcd := (others => '0') ; 50. hex\_src := ACC\_DATA\_OUT\_BUS; 51. for i in hex\_src'range loop 52. if bcd(3 downto 0) > "0100" then 53. bcd(3 downto 0) := bcd(3 downto 0) + "0011" ; 54. end if ; 55. if bcd(7 downto 4) > "0100" then 56. bcd(7 downto 4) := bcd(7 downto 4) + "0011" ; 57. end if ; 58. if bcd(11 downto 8) > "0100" then 59. bcd(11 downto 8) := bcd(11 downto 8) + "0011" ; 60. end if ; 62. bcd := bcd(10 downto 0) & hex\_src(hex\_src'left) ; -- shift bcd + 1 new entry 63. hex\_src := hex\_src(hex\_src'left - 1 downto hex\_src'right) & '0' ; -- shift src + pad with 0 64. end loop ; 65. HONDREDS\_BUS <= bcd (11 downto 8); 66. DECS\_BUS <= bcd (7 downto 4); 67. ONES\_BUS <= bcd (3 downto 0); 69. end process BIN\_TO\_BCD; 71. INDICATE : process(CLOCK) 72. type DIGIT\_TYPE is (ONES, DECS, HUNDREDS); 74. variable CUR\_DIGIT : DIGIT\_TYPE := ONES; 75. variable DIGIT\_VAL : STD\_LOGIC\_VECTOR(3 downto 0) := "0000"; 76. variable DIGIT\_CTRL : STD\_LOGIC\_VECTOR(6 downto 0) := "0000000"; 77. variable COMMONS\_CTRL : STD\_LOGIC\_VECTOR(2 downto 0) := "000"; 79. begin 80. if (rising\_edge(CLOCK)) then 81. if(RESET = '0') then 82. case CUR\_DIGIT is 83. when ONES => 84. DIGIT\_VAL := ONES\_BUS; 85. CUR\_DIGIT := DECS; 86. COMMONS\_CTRL := "001"; 87. when DECS => 88. DIGIT\_VAL := DECS\_BUS; 89. CUR\_DIGIT := HUNDREDS; 90. COMMONS\_CTRL := "010"; 91. when HUNDREDS => 92. DIGIT\_VAL := HONDREDS\_BUS; 93. CUR\_DIGIT := ONES; 94. COMMONS\_CTRL := "100"; 95. when others => 96. DIGIT\_VAL := ONES\_BUS; 97. CUR\_DIGIT := ONES; 98. COMMONS\_CTRL := "000"; 99. end case; 101. case DIGIT\_VAL is --abcdefg 102. when "0000" => DIGIT\_CTRL := "1111110"; 103. when "0001" => DIGIT\_CTRL := "0110000"; 104. when "0010" => DIGIT\_CTRL := "1101101"; 105. when "0011" => DIGIT\_CTRL := "1111001"; 106. when "0100" => DIGIT\_CTRL := "0110011"; 107. when "0101" => DIGIT\_CTRL := "1011011"; 108. when "0110" => DIGIT\_CTRL := "1011111"; 109. when "0111" => DIGIT\_CTRL := "1110000"; 110. when "1000" => DIGIT\_CTRL := "1111111"; 111. when "1001" => DIGIT\_CTRL := "1111011"; 112. when others => DIGIT\_CTRL := "0000000"; 113. end case; 114. else 115. DIGIT\_VAL := ONES\_BUS; 116. CUR\_DIGIT := ONES; 117. COMMONS\_CTRL := "000"; 118. end if; 120. COMM\_ONES <= not COMMONS\_CTRL(0); 121. COMM\_DECS <= not COMMONS\_CTRL(1); 122. COMM\_HUNDREDS <= not COMMONS\_CTRL(2); 124. SEG\_A <= not DIGIT\_CTRL(6); 125. SEG\_B <= not DIGIT\_CTRL(5); 126. SEG\_C <= not DIGIT\_CTRL(4); 127. SEG\_D <= not DIGIT\_CTRL(3); 128. SEG\_E <= not DIGIT\_CTRL(2); 129. SEG\_F <= not DIGIT\_CTRL(1); 130. SEG\_G <= not DIGIT\_CTRL(0); 131. DP <= '1'; 133. end if; 134. end process INDICATE; 135. end Behavioral; |

**

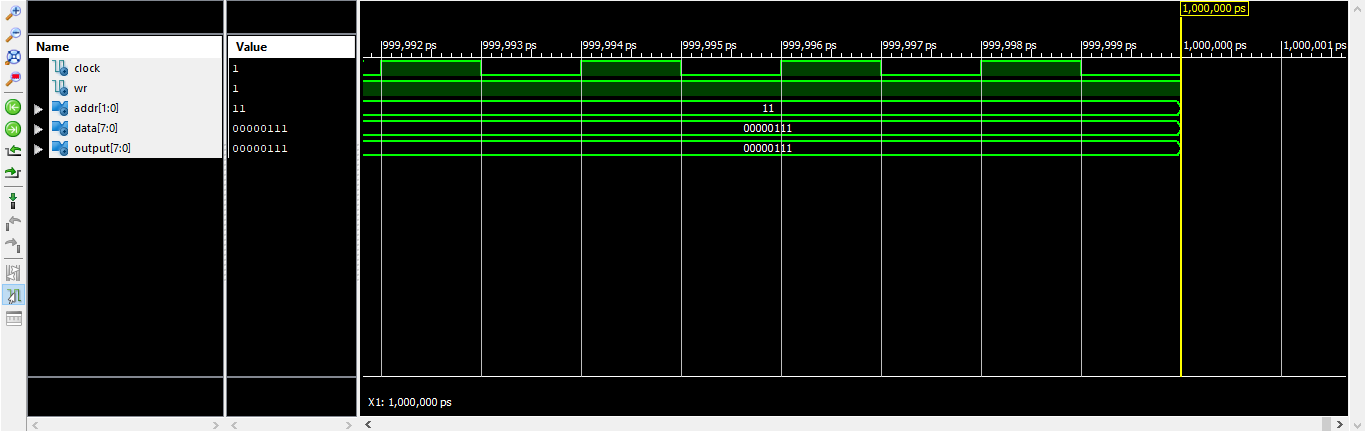
*Рис. 2 – Часова діаграма ACC*

**

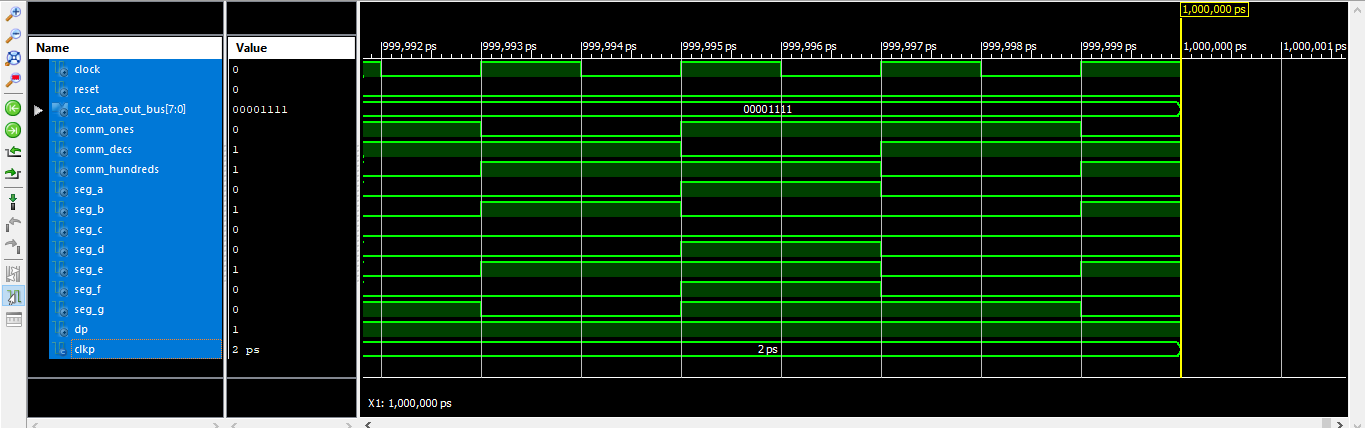
*Рис. 3 – Часова діаграма ALU*

**

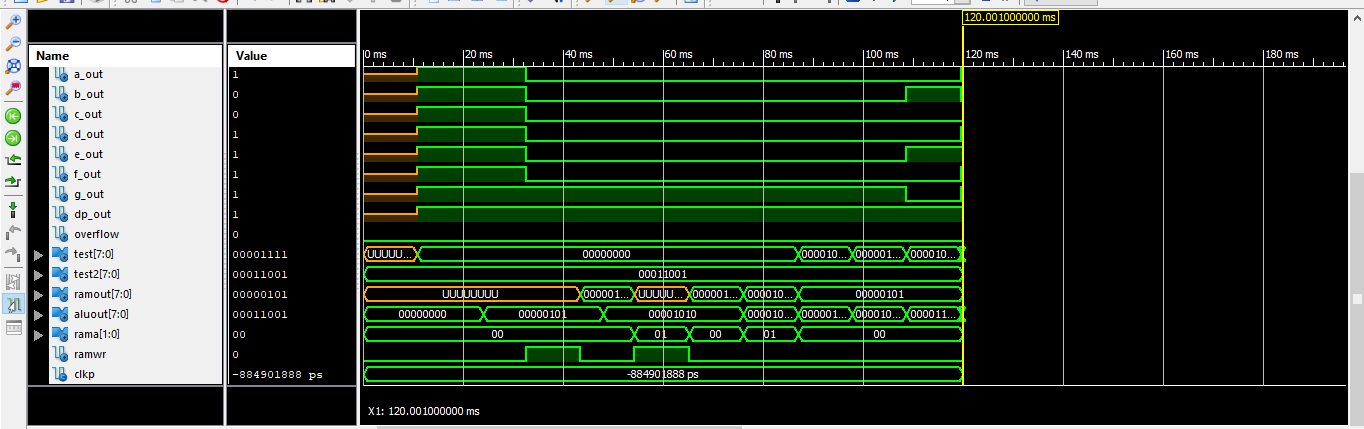
*Рис. 4 – Часова діаграма MUX*

**

*Рис. 5 – Часова діаграма RAM*

**

*Рис 6. – Часова діграма SEG\_DECODER*

**

*Рис 7. – Часова діграма TopLevel*

|  |
| --- |
| *Файл TopLevelTest.vhd*   1. -- Vhdl test bench created from schematic C:\Users\User\Documents\Lab\_3\TopLevel.sch - Mon Apr 29 19:00:56 2024 2. -- 3. -- Notes: 4. -- 1) This testbench template has been automatically generated using types 5. -- std\_logic and std\_logic\_vector for the ports of the unit under test. 6. -- Xilinx recommends that these types always be used for the top-level 7. -- I/O of a design in order to guarantee that the testbench will bind 8. -- correctly to the timing (post-route) simulation model. 9. -- 2) To use this template as your testbench, change the filename to any 10. -- name of your choice with the extension .vhd, and use the "Source->Add" 11. -- menu in Project Navigator to import the testbench. Then 12. -- edit the user defined section below, adding code to generate the 13. -- stimulus for your design. 14. -- 15. LIBRARY ieee; 16. USE ieee.std\_logic\_1164.ALL; 17. USE ieee.numeric\_std.ALL; 18. LIBRARY UNISIM; 19. USE UNISIM.Vcomponents.ALL; 20. ENTITY TopLevel\_TopLevel\_sch\_tb IS 21. END TopLevel\_TopLevel\_sch\_tb; 22. ARCHITECTURE behavioral OF TopLevel\_TopLevel\_sch\_tb IS 23. COMPONENT TopLevel 24. PORT( CLOCK : IN STD\_LOGIC; 25. RESET : IN STD\_LOGIC; 26. ENTER\_OP1 : IN STD\_LOGIC; 27. ENTER\_OP2 : IN STD\_LOGIC; 28. CALCULATE : IN STD\_LOGIC; 29. DATA\_IN : IN STD\_LOGIC\_VECTOR (7 DOWNTO 0); 30. COMMON\_0\_OUT : OUT STD\_LOGIC; 31. COMMON\_1\_OUT : OUT STD\_LOGIC; 32. COMMON\_2\_OUT : OUT STD\_LOGIC; 33. TEST: OUT STD\_LOGIC\_VECTOR(7 downto 0); 34. A\_OUT : OUT STD\_LOGIC; 35. B\_OUT : OUT STD\_LOGIC; 36. C\_OUT : OUT STD\_LOGIC; 37. D\_OUT : OUT STD\_LOGIC; 38. E\_OUT : OUT STD\_LOGIC; 39. F\_OUT : OUT STD\_LOGIC; 40. G\_OUT : OUT STD\_LOGIC; 41. DP\_OUT : OUT STD\_LOGIC; 42. RAMOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0); 43. ALUOUT: OUT STD\_LOGIC\_VECTOR(7 downto 0); 44. RAMA: OUT STD\_LOGIC\_VECTOR(1 downto 0); 45. RAMWR: OUT STD\_LOGIC; 46. OVERFLOW : OUT STD\_LOGIC); 47. END COMPONENT; 48. SIGNAL CLOCK : STD\_LOGIC := '0'; 49. SIGNAL RESET : STD\_LOGIC; 50. SIGNAL ENTER\_OP1 : STD\_LOGIC; 51. SIGNAL ENTER\_OP2 : STD\_LOGIC; 52. SIGNAL CALCULATE : STD\_LOGIC; 53. SIGNAL DATA\_IN : STD\_LOGIC\_VECTOR (7 DOWNTO 0); 54. SIGNAL COMMON\_0\_OUT : STD\_LOGIC; 55. SIGNAL COMMON\_1\_OUT : STD\_LOGIC; 56. SIGNAL COMMON\_2\_OUT : STD\_LOGIC; 57. SIGNAL A\_OUT : STD\_LOGIC; 58. SIGNAL B\_OUT : STD\_LOGIC; 59. SIGNAL C\_OUT : STD\_LOGIC; 60. SIGNAL D\_OUT : STD\_LOGIC; 61. SIGNAL E\_OUT : STD\_LOGIC; 62. SIGNAL F\_OUT : STD\_LOGIC; 63. SIGNAL G\_OUT : STD\_LOGIC; 64. SIGNAL DP\_OUT : STD\_LOGIC; 65. SIGNAL OVERFLOW : STD\_LOGIC; 66. SIGNAL TEST: STD\_LOGIC\_VECTOR(7 downto 0); 67. SIGNAL TEST2: STD\_LOGIC\_VECTOR(7 downto 0); 68. signal RAMOUT: STD\_LOGIC\_VECTOR(7 downto 0); 69. signal ALUOUT: STD\_LOGIC\_VECTOR(7 downto 0); 70. signal RAMA: STD\_LOGIC\_VECTOR(1 downto 0); 71. signal RAMWR: STD\_LOGIC; 73. -- constant CLOCK\_period : time := 166ns; 74. constant CLKP: time := 12ms;--24ms; 75. BEGIN 76. UUT: TopLevel PORT MAP( 77. CLOCK => CLOCK, 78. RESET => RESET, 79. ENTER\_OP1 => ENTER\_OP1, 80. ENTER\_OP2 => ENTER\_OP2, 81. CALCULATE => CALCULATE, 82. DATA\_IN => DATA\_IN, 83. COMMON\_0\_OUT => COMMON\_0\_OUT, 84. COMMON\_1\_OUT => COMMON\_1\_OUT, 85. COMMON\_2\_OUT => COMMON\_2\_OUT, 86. A\_OUT => A\_OUT, 87. B\_OUT => B\_OUT, 88. C\_OUT => C\_OUT, 89. D\_OUT => D\_OUT, 90. E\_OUT => E\_OUT, 91. F\_OUT => F\_OUT, 92. G\_OUT => G\_OUT, 93. DP\_OUT => DP\_OUT, 94. OVERFLOW => OVERFLOW, 95. TEST => TEST, 96. RAMOUT => RAMOUT, 97. ALUOUT => ALUOUT, 98. RAMA => RAMA, 99. RAMWR => RAMWR 100. ); 102. CLOCK\_process: process 103. begin 104. CLOCK <= '0'; 105. wait for 83ns; 106. CLOCK <= '1'; 107. wait for 83ns; 108. end process; 109. -- \*\*\* Test Bench - User Defined Section \*\*\* 110. tb : PROCESS 111. BEGIN 112. lp1: for i in 4 to 4 loop 113. lp2: for j in 2 to 2 loop 114. TEST2 <= std\_logic\_vector(to\_unsigned(to\_integer(signed(std\_logic\_vector(to\_unsigned(j, 8)) or std\_logic\_vector(to\_unsigned(i, 8)))) + j + 10 - 3, 8)); 115. ENTER\_OP1 <= '1'; 116. ENTER\_OP2 <= '1'; 117. CALCULATE <= '1'; 118. DATA\_IN <= (others => '0'); 119. RESET <= '0'; 120. wait for CLKP; 121. RESET <= '1'; 122. wait for CLKP; 123. DATA\_IN <= std\_logic\_vector(to\_unsigned(i, 8)); -- A 124. ENTER\_OP1 <= '0'; 125. wait for CLKP; 126. ENTER\_OP1 <= '1'; 127. wait for CLKP; 128. DATA\_IN <= std\_logic\_vector(to\_unsigned(j, 8)); -- B 129. ENTER\_OP2 <= '0'; 130. wait for CLKP; 131. ENTER\_OP2 <= '1'; 132. wait for CLKP; 133. CALCULATE <= '0'; -- START CALCULATION 134. wait for CLKP\* 7; 135. assert TEST = TEST2 severity FAILURE; 136. wait for CLKP; 137. end loop; 138. end loop; 140. WAIT; -- will wait forever 141. END PROCESS; 142. -- \*\*\* End Test Bench - User Defined Section \*\*\* 143. 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.