E6h6101 std_logic; signal cout : std_logic; begin

Design, Architecture

E6H6101ElectronicsAssignment2 Part1Combinationaland Digital LogicIntroduction8-bit adder designmaking needs 1-bit adder, 4-bit adder and 8-bit adder. 1-bit adder VHDL codecan be made by full adder logic circuit. And then 4-bit adder code is producedby port mapping of 1-bit adder. Finally, 8-bit adder is designed with testbenches programs. Discussion Full adder – 1-bitadderVHDL entity for a 1-bit adder with carry library IEEE; use IEEE. STD_LOGIC_1164. ALL; entityfulladder is Port (A : inSTD_LOGIC; B : inSTD_LOGIC; Cin : inSTD_LOGIC; S : outSTD_LOGIC; Cout : outSTD_LOGIC); end fulladder; architecturegate_level of fulladder is begin S <= A XORB XOR Cin ; Cout <= (AAND B) OR (Cin AND A) OR (Cin AND B) ; endgate_level; Testbench program for a 1-bit adder LIBRARY ieee; USE ieee.

std_logic_1164. ALL; ENTITYTestbench_fulladder ISEND Testbench_fulladder; ARCHITECTUREbehavior OF Testbench_fulladder IS COMPONENTfulladder_vhdl_code PORT(A : INstd_logic; B : INstd_logic; Cin : INstd_logic; S : OUTstd_logic; Cout : OUTstd_logic); ENDCOMPONENT; signal A : std_logic := ‘ 0’; signal B : std_logic := ‘ 0’; signal Cin : std_logic := ‘ 0’; signal S : std_logic; signal Cout : std_logic; BEGIN uut: fulladder_vhdl_code PORT MAP (A => A, B => B, Cin => Cin, S => S, Cout => Cout); stim_proc: process begin wait for 100ns; A <= '1'; B <= '0'; Cin <='0'; wait for 10ns; A <= '0'; B <= '1'; Cin <='0'; wait for 10ns; A <= '1'; B <= '1'; Cin <='0'; wait for 10ns; A <= '0'; B <= '0'; Cin <='1'; wait for 10ns; A <= '1'; B <= '0'; Cin <='1'; wait for 10ns; A <= '0'; B <= '1'; Cin <='1'; wait for 10ns; A <= '1'; B <= '1'; Cin <='1'; wait for 10ns; end process; END; Simulation of programs using Intel

Quartus Prime andModelsim-Intel 1-bit adder designTo design 1-bit adder, the logic gates' function equations are made from this logic diagram. Threeinputs A, B, C in and 2 outputs S and C out are declared. Architecture of 1-bitadder is produced. Stimulus are inserted to run the VHDL program and testbenchis simulated. Output waveform of 1-bit adder (full adder) is produced.

Simulation waveform of 1-bit adder 4-bitadderVHDL entity for a 4-bit adder using 1-bit adder library IEEE; use IEEE. STD_LOGIC_1164. ALL; entity4bitAdder isPort (A : in STD_LOGIC_VECTOR (3 downto 0); B : in STD_LOGIC_VECTOR (3 downto 0); Cin : in STD_LOGIC; S : out STD_LOGIC_VECTOR (3 downto 0); Cout : out STD_LOGIC); end 4bitAdder; architectureBehavioral of 4bitAdder is componentfulladder_vhdl_codePort (A : in STD_LOGIC; B : in STD_LOGIC; Cin : in STD_LOGIC; S : out STD_LOGIC; Cout : out STD_LOGIC); end component; signal c1, c2, c3: STD_LOGIC; begin FA1: fulladder_vhdl_code port map(A(0), B(0), Cin, S(0), c1); FA2: fulladder_vhdl_code port map(A(1), B(1), c1, S(1), c2); FA3: fulladder_vhdl_code port map(A(2), B(2), c2, S(2), c3); FA4: fulladder_vhdl_code port map(A(3), B(3), c3, S(3), Cout); endBehavioral; Testbench program for a 4-bit adder LIBRARY ieee; USE ieee. std_logic_1164. ALL; ENTITY4bitAdder ISEND 4bitAdder; ARCHITECTUREbehavior OF 4bitAdder ISCOMPONENT 4bitAdderPORT(A : IN std_logic_vector(3 downto 0); B : IN std_logic_vector(3 downto 0); Cin : IN std_logic; S : OUT std_logic_vector(3 downto 0); Cout : OUT std_logic); END COMPONENT; signal A : std_logic_vector(3 downto 0) := (others => ‘ 0’); signal B : std_logic_vector(3 downto 0) := (others=> ‘ 0’);

signal Cin : std_logic := ‘ 0’; signal S : std_logic_vector(3 downto 0); signal Cout : std_logic; BEGINuut: 4bitAdder PORT MAP (A => A, B => B, Cin => Cin, S => S, Cout => Cout); stim_proc: processbeginwait for 100 ns; A <= " 0110"; B <= " 1100"; wait for 100ns; A <= " 1111"; B <= " 1100"; wait for 100ns; A <= " 0110"; B <= " 0111"; wait for 100ns; A <= " 0110"; B <= " 1110"; wait for 100ns; A <= " 1111"; B <= " 1111"; wait; end process; END; Simulation of programs using Intel Quartus Prime andModelsim-Intel 4-bit adder designTo design 4-bit adder, the architecture of 1-bit adder is applied and port mapping is made 4 times.

Three inputs A, B, C in and 2 outputs S and C out are declared. Architecture of4-bit adder is produced. Stimulus are inserted to run the VHDL program andtestbench is simulated. Output waveform of 4-bit adder is produced. Simulation waveform of 4-bit adder

8-bit adderVHDL entity for an 8-bit adder using 4-bit adder library IEEE; use IEEE. STD_LOGIC_1164. ALL; entity 8bitadder isport (x : in std_logic_vector(7 downto 0); y : instd_logic_vector(7 downto 0); cin : instd_logic; f : outstd_logic_vector(7 downto 0); cout : outstd_logic); end 8bitadder; architecture structural of 8bitadder iscomponent fulladder_con is port (x, y, cin: in std_logic; f, cout : out std_logic); end component; signal carry : std_logic_vector(6 downto 0); beginU1 : fulladder_con port map(x(0), y(0), cin, f(0), carry(0)); U2 : for i in 1 to 6 generate U3 : fulladder_con port map (x(i), y(i), carry(i-1), f(i), carry(i)); end generate; U4 : fulladder_con port map(x(7), y(7), carry(6), f(7), cout); end structural; Testbench program for an 8-bit adder library IEEE; use IEEE.

STD_LOGIC_1164. ALL; entity 8bitadder_tst isend 8bitadder_tst; architecture beh of 8bitadder_tst iscomponent 8bitadder isport (x : in std_logic_vector(7 downto 0); y : instd_logic_vector(7 downto 0); cin : instd_logic; f : outstd_logic_vector(7 downto 0); cout : outstd_logic); end component; signal x_s, y_s, f_s : std_logic_vector (7 downto 0); signal cin_s, cout_s : std_logic; beginDUT : 8bitadder port map (x_s, y_s, cin_s, f_s, cout_s); process begin x_s <=" 10101010"; y_s <=" 01010101"; cin_s <='0'; wait for 10ns; x_s <=" 11001100"; y_s <=" 11110000"; cin_s <='1'; wait for 10ns; x_s <=" 11010111"; y_s <=" 01011010"; cin_s <='0'; wait for 10ns; x_s <=" 10110011"; y_s <=" 11111111"; cin_s <='1'; wait for 10ns; end process; end beh; Simulation of programs using Intel Quartus Prime andModelsim-Intel 8-bit adder designTo design 8-bit adder, the architecture of 4-bit adder is applied and port mapping is made. Threeinputs X, Y, C in and 2 outputs F and C out are declared. Architecture of 8-bitadder is produced. Stimulus are inserted to run the VHDL program and testbenchis simulated. Output waveform of 8-bit adder is produced. Simulation waveform of 8-bit adder SummaryAll waveform results are produced and they areoutputted like test benches' programs. Output waveform can be checked withinput data according to VHDL codes.

Part2SequentialDigital SystemIntroductionInup-down counter design, it is needed to decide conditions and inputs/outputsdeclaration to write VHDL code. Case states changes are produced with truthtable and test bench code is written to produce output values. Up-down counter state machine DiscussionVHDL program of the up-down counter as a statemachine library ieee; use ieee. std_logic_1164. all; entity state isport

(clock, reset, input: in std_logic; output: out std_logic_vector (3 downto 0)); end state; architectire behavioural of state istype state_t is (0, 1, 2, 3, 4, 5, 6, 7, 8, 9); signal state: state_t; process (clock, reset)beginif (reset=’1′)thenstate <= 0; elsif rising-edge (clock) thencase state iswhen 0 => if input = ‘ 1’ then state <= 1; elsif input = '0' then state <= 9; end if; when 1 => if input = ‘ 1’ then state <= 2; elsif input = '0' then state <= 0; end if; when 2 => if input = ‘ 1’ then state <= 3; elsif input = '0' then state <= 1; end if; when 3 => if input = ‘ 1’ then state <= 4; elsif input = '0' then state <= 2; end if; when 4 => if input = ‘ 1’ then state <= 5; elsif input = '0' then state <= 3; end if; when 5 => if input = ‘ 1’ then state <= 6; elsif input = '0' then state <= 4; end if; when 6 => if input = ‘ 1’ then state <= 7; elsif input = '0' then state <= 5; end if; when 7 => if input = ‘ 1’ then state <= 8; elsif input = '0' then state <= 6; end if; when 8 => if input = ‘ 1’ then state <= 9; elsif input = '0' then state <= 7; end if; when 9 => if input = ‘ 1’ then state <= 0; elsif input = '0' then state <= 8; end if; end case; end if; End process; output <= " 0001" when state = 1 else" 0010" when state = 2 else" 0011" when state = 3 else" 0100" when state = 4 else" 0101" when state = 5 else" 0110" when state = 6 else" 0111" when state = 7 else" 1000" when state = 8 else" 1001" when state = 9 else" 0000"; end behavioural; Testbench program of the up-down counter Library ieee; use ieee. std_logic_1164. all; entity state isport (clock, reset, input: in std_logic; output: out std_logic_vector (3 downto 0); end state; architecture behavioural of state istype state_t is (0, 1, 2, 3, 4, 5, 6, 7, 8, 9); signal state: state_t; process (clock, reset)beginif (reset= '1')thenstate <= 0; elsif rising_edge (clock) thenif (input= '1') thencase state iswhen 0 => state <=

1; when 1 => state <= 2; when 2 => state <= 3; when 3 => state <= 4; when 4 => state <= 5; when 5 => state <= 6; when 6 => state <= 7; when 7 => state <= 8; when 8 => state <= 9; when 9 => state <= 0; end case; elsif (input = '0') thencase state iswhen 0 => state <= 9; when 1 => state <= 0; when 2 => state <= 1; when 3 => state <= 2; when 4 => state <= 3; when 5 => state <= 4; when 6 => state <= 5; when 7 => state <= 6; when 8 => state <= 7; when 9 => state <= 8; end case; end if; End process; output <= " 0001" when state = 1 else" 0010" when state = 2 else" 0011" when state = 3 else" 0100" when state = 4 else" 0101" when state = 5 else" 0110" when state = 6 else" 0111" when state = 7 else" 1000" when state = 8 else" 1001" when state = 9 else" 0000"; end behavioural; Simulation of programs using Intel Quartus Prime andModelsim-Intel Up-down counter designTo design up-downcounter, this state machine of 10 states is made as state VHDL code.

Three inputsclock, reset, input and one output are declared. Architecture of state isproduced. Case states and output description are written to run the VHDLprogram and test bench is simulated.

If reset is given by 1, the state willreturn to 0 and output will be 0000. If reset is given by 0, the state willperform like up-down counter functions. If rising-edge (clock), case stateprocesses will start. States will change to next state as ascending ordescending according to input controlling 1 or 0. This states are zero to nine(10 states) and outputs will be 4-bit binary numbers (0000 to 1001). SummaryThis state machine mustwork correctly with VHDL code like up-down counter.

In both up or downcondition, this state must change truly count up or count down and the outputvalues are produced respectively according to output states assign.