Assignment 5

IT451 : COA LAB

ANINDYA KUNDU
IT, 4th Semester
ID: 510817020 (Hx-19)

01-04-2019

Question 1

Design and simulate a 32 bit ALU with standard operations as given in Assignment II Q 5.

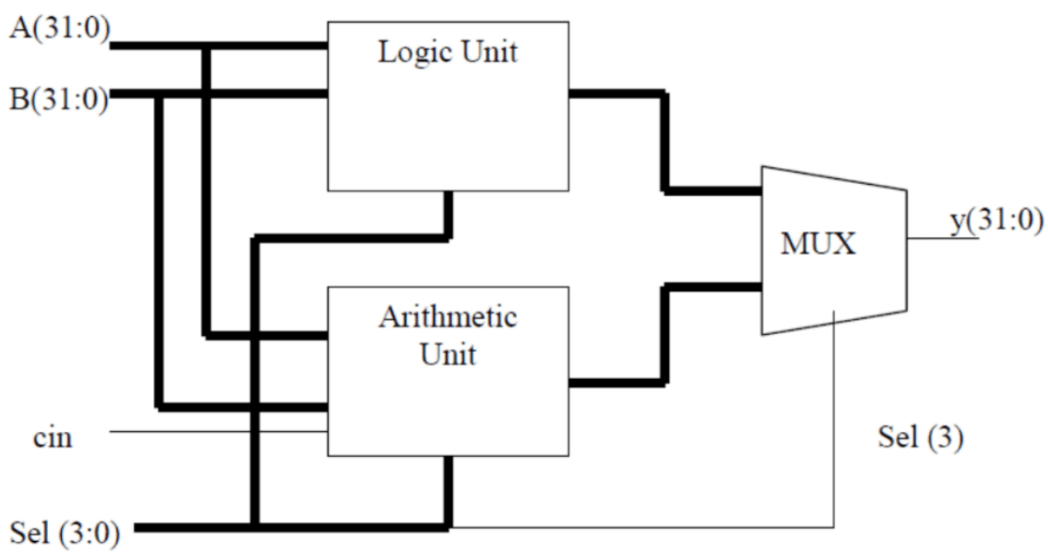
a. Synthesize it for Spartan 6. Report device utilization.

b. Synthesize the previously designed 16 bit ALU for the same device and report device utilization.

c. Create appropriate test benches to reflect its functional behavior for all possible cases. Include the snapshots in your report.

d. Prepare a comparative table between the two designs to reflect their difference in hardware complexities.

Unit	Function	Sel
Arithmetic	Transfer a	0000
	Increment a	0001
	Decrement a	0010
	Transfer b	0011
	Increment b	0100
	Decrement b	0101
	Add a and b	0110
	Add a and b with carry	0111
Logical	Complement a	1000
	Complement b	1001
	AND	1010
	OR	1011
	NAND	1100
	NOR	1101
	XOR	1110
	XNOR	1111



VHDL Module:

arithUnit.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity arithUnit is
    Port ( S : in       STD_LOGIC_VECTOR ( 3 downto 0);
           A : in       STD_LOGIC_VECTOR (31 downto 0);
           B : in       STD_LOGIC_VECTOR (31 downto 0);
           C : in       STD_LOGIC;
           Y : inout    STD_LOGIC_VECTOR (31 downto 0));
end arithUnit;

architecture Behavioral of arithUnit is

begin

process(S, A, B, C)
      variable uA, uB: unsigned(31 downto 0) := (others => '0');
begin
      uA := unsigned(A);
      uB := unsigned(B);

      if(S(3) = '0') then
            case S is
                  when "0000" => Y <= A;
                  when "0001" =>
                        Y <= STD_LOGIC_VECTOR(uA + 1);
                  when "0010" =>
                        Y <= STD_LOGIC_VECTOR(uA - 1);
                  when "0011" => Y <= B;
                  when "0100" =>
                        Y <= STD_LOGIC_VECTOR(uB + 1);
                  when "0101" =>
                        Y <= STD_LOGIC_VECTOR(uB - 1);
                  when "0110" =>
                        Y <= STD_LOGIC_VECTOR(uA + uB);
                  when "0111" =>
                        if(C = '1') then Y <= STD_LOGIC_VECTOR(uA + uB + 1);
                        else             Y <= STD_LOGIC_VECTOR(uA + uB + 0);
                        end if;
                  when others => Y <= "00000000000000000000000000000000";
```

```
            end case;
      else Y <= "00000000000000000000000000000000";
      end if;
end process;

end Behavioral;
```

logicUnit.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity logicUnit is
    Port ( S : in       STD_LOGIC_VECTOR ( 3 downto 0);
                  A : in          STD_LOGIC_VECTOR (31 downto 0);
           B : in       STD_LOGIC_VECTOR (31 downto 0);
           Y : inout    STD_LOGIC_VECTOR (31 downto 0));
end logicUnit;

architecture Behavioral of logicUnit is

begin

process(S, A, B, Y)
begin
      if(S(3) = '1') then
            case S is
                  when "1000" => Y <= not A;
                  when "1001" => Y <= not B;
                  when "1010" => Y <= A and  B;
                  when "1011" => Y <= A or   B;
                  when "1100" => Y <= A nand B;
                  when "1101" => Y <= A nor  B;
                  when "1110" => Y <= A xor  B;
                  when "1111" => Y <= A xnor B;
                  when others => Y <= "00000000000000000000000000000000";
            end case;
      else Y <= "00000000000000000000000000000000";
      end if;
end process;

end Behavioral;
```

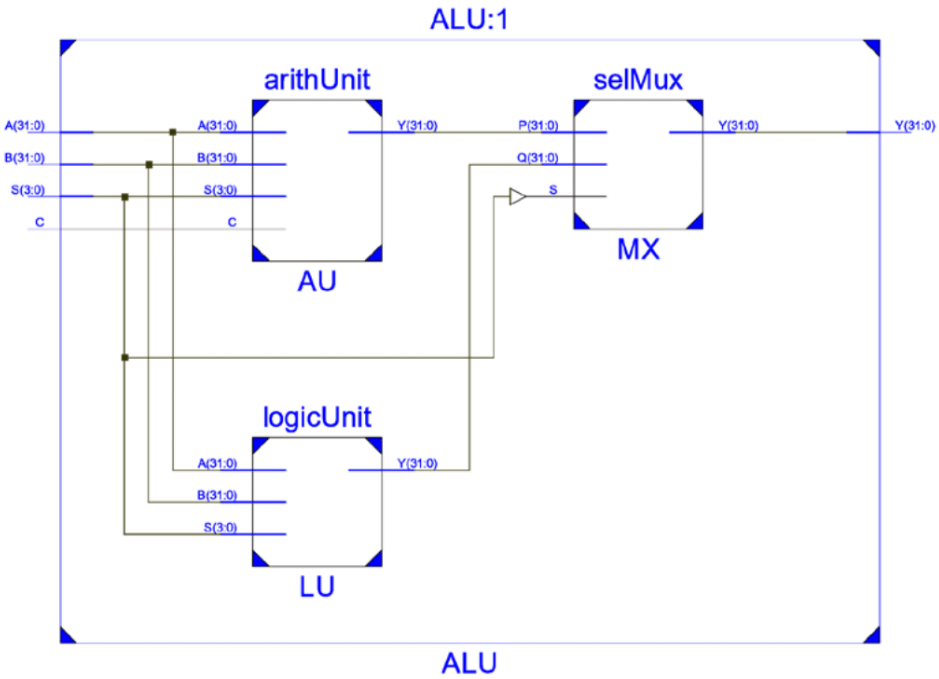
selMux.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity selMux is
    Port ( S : in  STD_LOGIC;
           P : in  STD_LOGIC_VECTOR (31 downto 0);
           Q : in  STD_LOGIC_VECTOR (31 downto 0);
           Y : out STD_LOGIC_VECTOR (31 downto 0));
end selMux;

architecture Behavioral of selMux is

begin
      Y <= P when S = '0' else Q;
end Behavioral;
```

ALU.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ALU is
    Port ( A : in  STD_LOGIC_VECTOR (31 downto 0);
           B : in  STD_LOGIC_VECTOR (31 downto 0);
           C : in  STD_LOGIC;
           S : in  STD_LOGIC_VECTOR ( 3 downto 0);
           Y : out STD_LOGIC_VECTOR (31 downto 0));
end ALU;

architecture Structural of ALU is
      signal M, N, O : STD_LOGIC_VECTOR (31 downto 0);

begin

AU : entity work.arithUnit port map(S, A, B, C, M);
LU : entity work.logicUnit port map(S, A, B, N);
MX : entity work.selMux       port map(S(3), M, N, Y);

end Structural;
```

RTL Schematic:

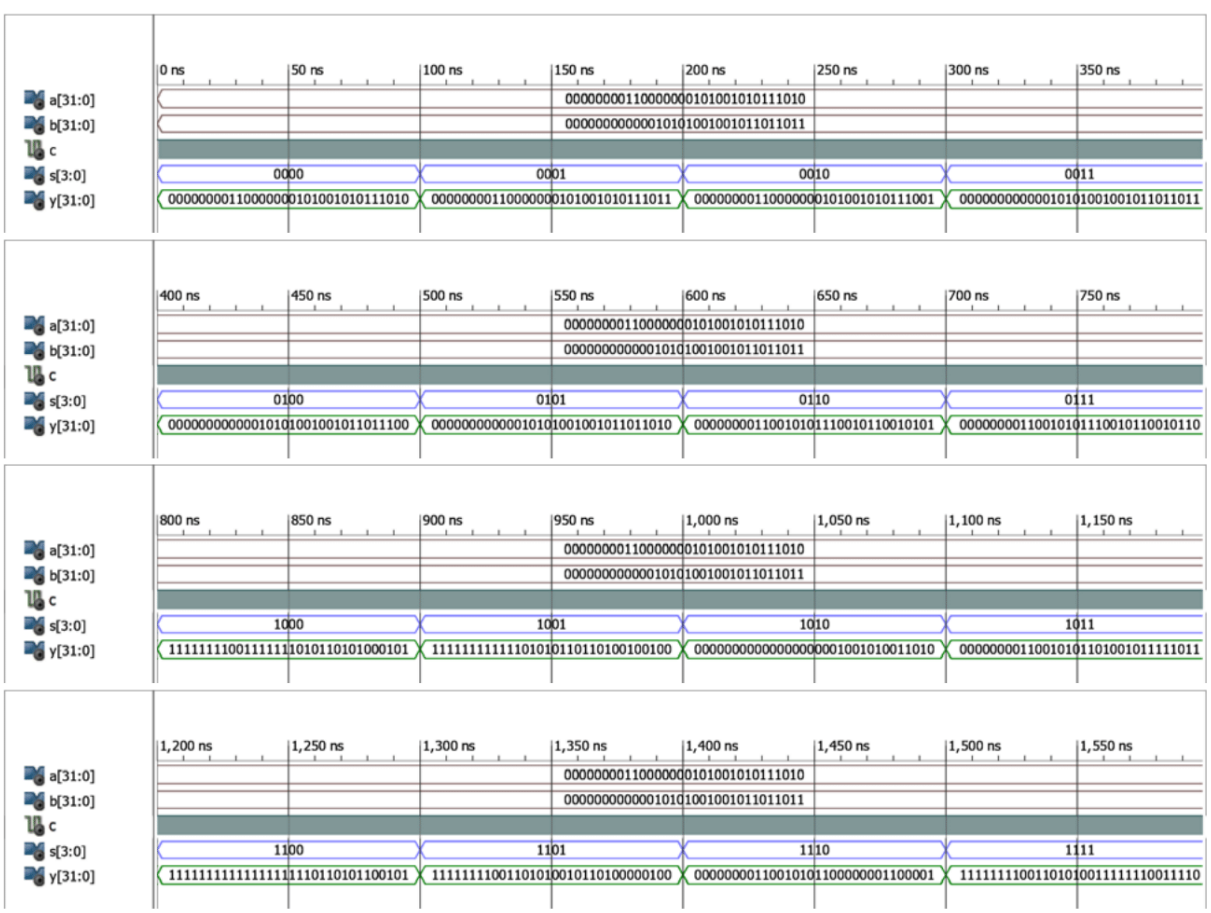


VHDL Test Bench: *tb_ALU.vhd*

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

ENTITY tb_ALU IS
END tb_ALU;

ARCHITECTURE behavior OF tb_ALU IS
	COMPONENT ALU
		PORT ( A : in  STD_LOGIC_VECTOR (31 downto 0);
		       B : in  STD_LOGIC_VECTOR (31 downto 0);
		       C : in  STD_LOGIC;
		       S : in  STD_LOGIC_VECTOR ( 3 downto 0);
		       Y : out STD_LOGIC_VECTOR (31 downto 0));
	END COMPONENT;

SIGNAL a : STD_LOGIC_VECTOR (31 downto 0) := (others => '0');
SIGNAL b : STD_LOGIC_VECTOR (31 downto 0) := (others => '0');
SIGNAL c : STD_LOGIC := '0';
SIGNAL s : STD_LOGIC_VECTOR ( 3 downto 0) := (others => '0');
SIGNAL y : STD_LOGIC_VECTOR (31 downto 0);

BEGIN
	uut: ALU PORT MAP(
```

```
            A => a,
            B => b,
            C => c,
            S => s,
            Y => y
      );

process
begin
      a <= "00000000110000000101001010111010";
      b <= "00000000000001010100100101011011011";
      c <= '1';
      s <= "0000";
      while s >= "0000" loop
            wait for 100ns;
            s <= STD_LOGIC_VECTOR(unsigned(s) + 1);
      end loop;
      wait;
end process;
END;
```

Simulation:



Device Utilisation Summary:

32-bit ALU

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	384	27288	1%
Number of fully used LUT-FF pairs	0	384	0%
Number of bonded IOBs	101	296	34%

16-bit ALU

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	192	27288	0%
Number of fully used LUT-FF pairs	0	192	0%
Number of bonded IOBs	53	296	17%

Hardware Complexity Comparison:

	32-bit ALU			16-bit ALU		
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Used**	**Available**	**Utilization**
Number of Slice Registers	0	54,576	0%	0	54,576	0%
Number of Slice LUTs	384	27,288	1%	192	27,288	1%
Number used as logic	381	27,288	1%	189	27,288	1%
Number using O6 output only	226			114		
Number using O5 output only	92			44		
Number using O5 and O6	63			31		
Number used as ROM	0			0		
Number used as Memory	0	6,408	0%	0	6,408	0%
Number used exclusively as route-thrus	3			3		
Number with same-slice register load	0			0		
Number with same-slice carry load	3			3		
Number with other load	0			0		
Number of occupied Slices	119	6,822	1%	67	6,822	1%
Number of MUXCYs used	192	13,644	1%	96	13,644	1%
Number of LUT Flip Flop pairs used	384			192		
Number with an unused Flip Flop	384	384	100%	192	192	100%
Number with an unused LUT	0	384	0%	0	192	0%
Number of fully used LUT-FF pairs	0	384	0%	0	192	0%
Number of slice register sites lost to control set restrictions	0	54,576	0%	0	54,576	0%
Number of bonded IOBs	101	296	34%	53	296	17%