CDA Lab

Assignment/Experiment-5 : Arithmetic Unit

Q) Write a VHDL program to implement Arithmetic Unit using data modelling.

Code)
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity ARITHMETIC_UNIT is
Port
(
a,b: in std_logic_vector(3 downto 0);
op: in std_logic_vector(2 downto 0);
--zero: out std_logic;
f: out std_logic_vector(3 downto 0)
);
end ARITHMETIC_UNIT;
architecture bhv of ARITHMETIC_UNIT is
begin
process(op,a,b);
variable temp: std_logic_vector(3 downto 0);
begin
case op is
when "000" =>
temp:= a+b;
when "001" =>
temp:= a+b+1;
when "010" =>
temp:= a+(not b)+1;
when "011" =>
temp:= a+(not b);
when "100" =>
temp:= a+1;
when "101" =>
temp:= a-1;
when "110" =>
temp:= a;
when "111" =>
temp:= b;
```

```
when others =>
NULL;
end case;
f<=temp;
end process;
end bhv;
```

Assignment-2 : Logical Unit

Q) Write a VHDL code program to implement Logical Unit using data model

Code

```
library ieee;
  use ieee.std_logic_1164.all
  use ieee.std_logic_unsigned.all;
  use ieee.std_logic_arith.all;
entity LOGICAL_UNIT is
port
(
a,b: in std_logic_vector(3 downto 0);
op: in std_logic_vector(2 downto 0);
zero: out std_logic;
f: out std_logic_vector(3 downto 0);
);
end LOGICAL_UNIT;
architecture bhv of LOGICAL_UNIT is
begin
process(op,a,b)
variable temp: std_logic_vector(3 downto 0);
begin
case op is
when "000" =>
temp:=a OR b;
when "001" =>
temp:= a AND b;
when "010" =>
temp:= NOT a;
when "011" =>
temp:= a NOR b;
when "100" =>
temp:= a NAND b;
when "101" =>
temp:= a XOR b;
when "110" =>
temp:= a XNOR b;
```

```
when "111" =>
if a<b then
temp := "1111";
else
temp := "0000";
end if;
when others =>
NULL;
end case;
if temp = "0000" then
zero <= '1';
else
zero <= '0';
end if
c <= temp;
end Process;
end bhv;
```

Assignment 3: Arithmetic Logic Unit (ALU)

Code 3) Write a VHDL code program to implement Arithmetic logical Unit using data modelling.

Code)
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity ALU is
port(
A : in std_logic_vector(1 downto 0);
B : in std_logic_vector(1 downto 0);
Sel : in std_logic_vector(1 downto 0);
Res : out std_logic_vector(1 downto 0));
);
end ALU;
architecture behv of ALU is
begin
process(A,B,Sel)
begin
-- use case statement to archieve
-- different operations of ALU
case Sel is
when "00" =>
Res <= A+B;
when "01" =>
Res <= A+(not B)+1;
when "10" =>
```

```
Res <= A and B;
when "11" =>
Res <= A or B;
when others =>
Res <= "XX";
end case;
end Process;
end behv;
```

Theory : An Arithmetic Logic unit is the part of a central processing unit that carries out arithmetic and logic operations on the operands in computer instruction words. In some processors, the ALU is divided into two units:

i) an Arithmetic Unit (AU) and
ii) a Logical Unit (LU)

The Logical Unit handles logical operations like AND, OR and XOR instead of arithmetic operations. It also performs numerical tests - for examples, it checks whether the number is negative or not. It also controls if the output of the ALU is zero Arithmetic logical Unit (ALU) value is zero or not.

Assignment 4: Shift Unit

Write a VHDL program code to implement Shift unit using data modelling.

Code>

```
library ieee;
  use ieee.std_logic_1164.all;
  use ieee.std_logic_unsigned.all;
  use ieee.std_logic_arith.all;
entity shifter_unit is
 port(
 (a: in std_logic_vector(3 downto 0);
  op: in std_logic;
  f: out std_logic_vector(3 downto 0));
end shifter_unit;

architecture bhv of shifter_unit is
   begin
   process(op,a)
     variable temp: std_logic_vector(3 downto 0);
     begin
        case op is
          when '0' =>
             -- left shift
            temp(3 downto 1):= a(2 downto 0);
            temp(0):= '0';
         when '1' =>
             -- right shift
           temp(2 downto 0) := a(3 downto 1);
           temp(3):='0';
           when others =>
             NULL;
      end case;
     f<=temp;
 end process;
end bhv;
```

Assignment-5: Single Port RAM

Q) Write a VHDL Code program to implement single Port RAM using data modelling.

Code:

```
library ieee;
use ieee.std_logic_1164.all;
entity single_port_ram is
port
(data: in std_logic_vector(7 downto 0);
addr: in natural range 0 to 63;
we : in std_logic := '1';
clk : in std_logic ;
q : out std_logic_vector(7 downto 0)
);
end entity;

architecture rtl of single_port_ram is
subtype word_t is std_logic_vector(7 downto 0);
type memory_t is array(63 downto 0) of word_t;
signal ram: memory_t;
signal addr_reg : natural range 0 to 63;
begin
    if (rising_edge(clk)) then
    if(we='1') then
    ram(addr)<= data;
end if;
addr_reg <= addr;
end if;
end processor;
q <= ram(addr_reg);
end rtl;
```

Assignment 6: Booth

Q) Write a VHDL program code to implement Booth using data modelling

Code)
```
library ieee;
use ieee.std_logic_1164.all
use ieee.std_logic_unsigned.all;

entity BOOTH is
port(m: in std_logic_vector(4 downto 1);
     mc: in std_logic_vector(4 downto 1);
     o: out std_logic_vector(3 downto 1));
end BOOTH;

architecture arc of BOOTH is
begin
process(m,mc)
variable ea: std_logic_vector(4 downto 1)
variable b: std_logic_vector(4 downto 1)
variable q: std_logic_vector(4 downto 1)
variable qn: std_logic_vector(2 downto 1)
begin
ea:="0000";
b:=m;
q:=mc;
qn:=q(1)&'0';
for i in 4 downto 1 loop
if qn="01" then
  ea:=ea(4 downto 1)+b(4 downto 1);
  q:=ea(1)&q(4 downto 2);
  ea:=ea(4)&ea(4 downto 2);
  qn:=q(1)&qn(2);
elsif qn:"10" then
  ea:=ea(4 downto 1)+not(b(4 downto 1))+1;
  q:=ea(1)&q(4 downto 2);
  ea:=ea(4)&ea(4 downto 2);
  qn:=q(1)&qn(2);
else
  q:=ea(1)&q(4 downto 2);
  ea:=ea(4)&ea(4 downto 2);
  qn:=q(1)&qn(2);
```

```
end if;
end loop;
Q <= eq(4 downto 1) & q(4 downto 1);
end process;
end arc;
```

8) Write a VHDL code program to implement and verify data flow modelling.

Code: library ieee;
use ieee.std_logic_1164.all
use ieee.std_logic_unsigned.all
entity mod is
port(
q: in std_logic_vector(4 downto 1);
m: in std_logic_vector(4 downto 1);
r: out std_logic_vector(4 downto 1);
qo: out std_logic_vector(4 downto 1);
);
end mod;
architecture arc of mod is
begin
Process(q,m);
variable qe: std_logic_vector(4 downto 1);
variable me: std_logic_vector(4 downto 1);
variable a: std_logic_vector(4 downto 1);
begin
qe := q;
me := m;
a := "0000";
for i in 4 downto 1 loop
if(a(4)='0') then
a(4 downto 2) := a(3 downto 1);
a(1) := qe(4);
qe(4 downto 2) := qe(3 downto 1);
a(4 downto 1) := a(4 downto 1) + (not me(4 downto 1)) + 1;
elsif (a(4)='1') then
a(4 downto 2) := a(3 downto 1);
a(1) := qe(4);
qe(4 downto 2) := qe(3 downto 1);
a(4 downto 1) := a(4 downto 1) + me(4 downto 1);
end if;
if(a(4)='0') then

```
        qv(1) := '1';
    elsif(a(4) = '1') then
        qv(1) := '0';
    end if;
end loop;
if (a(4) = '1') then
    a(4 downto 1) := a(4 downto 1) + mv(4 downto 1);
end if;
q0 <= qv;
a <= a;
end process;
end arc;
```

8) Write a VHDL program code to implement rd using data modelling.

```
code> library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_unsigned.all;
    entity rd is
    port
    (a : in std_logic_vector(4 downto 1);
    m: in std_logic_vector(4 downto 1);
    q: out std_logic_vector(4 downto 1);
    q0: out std_logic_vector(4 downto 1)
    );
end rd
architecture arc of rd is
begin
process(a,m)
variable a : std_logic_vector(4 downto 1);
variable qv: std_logic_vector(4 downto 1);
variable mv: std_logic_vector(4 downto 1);
begin
qv := q;
mv := m;
a := "0000";
for i in 4 downto 1 loop
  a(4 downto 2) := a(3 downto 1);
  a(1) := qv(4);
qv(4 downto 2) := qv(3 downto 1);
```

```
a(4 downto 1) := a(4 downto 1) + (not mv(4 downto 1)) + 1;
  if (a(4) = '0') then
     qv(i) := '1';
  elsif (a(4) = '1') then
     qv(i) := '0';
     a(4 downto i) := a(4 downto 1) + mv(4 downto 1);
  end if;
end loop;
qo <= qv;
cr <= a;
end process;
end arc;
```