ECE 448
Midterm Exam – Solutions for Group 1
Monday, March 2, 2020

Problem 1

Perform the following tasks for

a combinational Variable Arithmetic Shifter Right,

with the block diagram shown in Fig. 1.

A. Fill in the blanks in the code of this component given on the next page.

B. Draw a symbol of this component with the correct
   - number
   - direction, and
   - widths

   of ALL inputs and outputs.

   Name each input and output, matching the VHDL code on the next page.

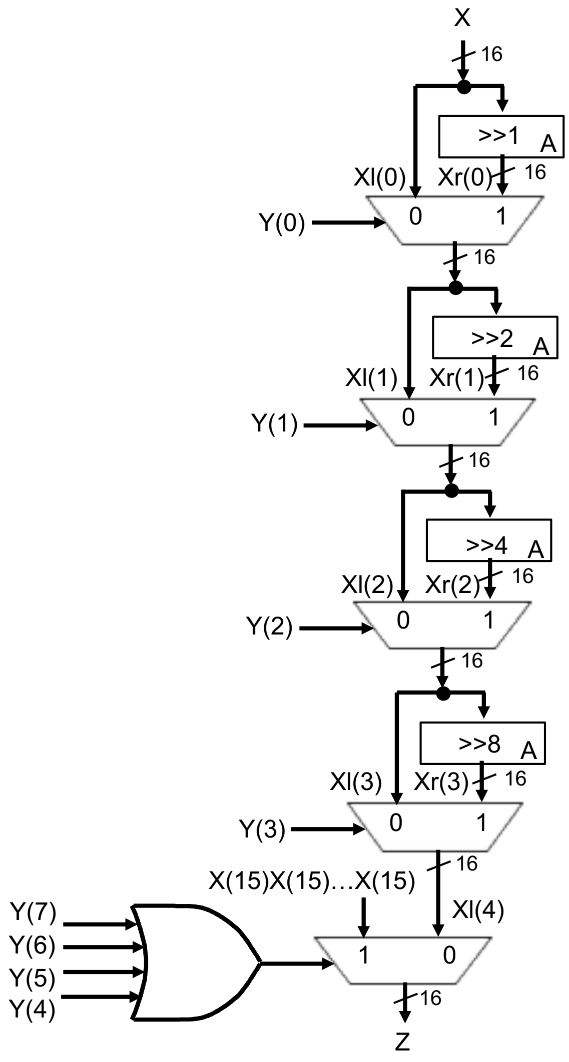


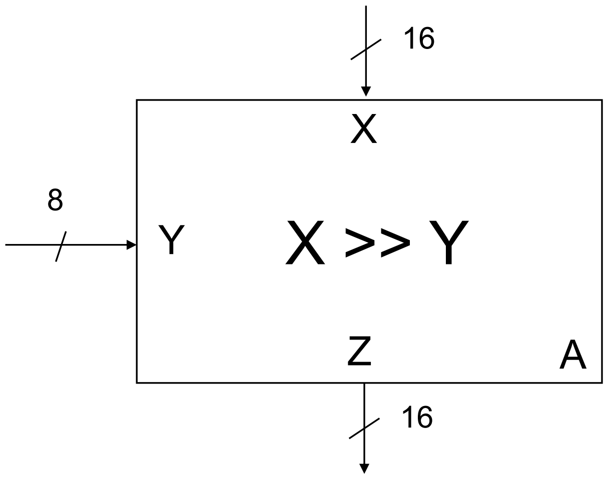
Fig. 1: Block diagram of a combinational Variable Arithmetic Shifter Right

Answer A:

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

ENTITY fixed_shifter_right IS
      GENERIC(L : INTEGER := 1);
      PORT(
           a : IN STD_LOGIC_VECTOR(15 downto 0);
           y : OUT STD_LOGIC_VECTOR(15 downto 0)
          );

END fixed_shifter_right;

ARCHITECTURE dataflow OF fixed_shifter_right IS
BEGIN
     y(15-L downto 0)    <= a(15 downto L);
     y(15 downto 15-L+1)<= OTHERS => a(15);
END dataflow;

library IEEE;
use IEEE.STD_LOGIC_1164.all;

ENTITY variable_shifter_right IS
      PORT(
          X : IN STD_LOGIC_VECTOR(15 downto 0);
          Y : IN STD_LOGIC_VECTOR(7 downto 0);
          Z : OUT STD_LOGIC_VECTOR(15 downto 0)
 );
END variable_shifter_right;

ARCHITECTURE mixed OF variable_shifter_right IS

TYPE array16 IS ARRAY (0 to 4) OF STD_LOGIC_VECTOR(15 DOWNTO 0);

SIGNAL Xl : array16;
SIGNAL Xr : array16;


BEGIN
     Xl(0) <= X;
     G: FOR i IN 0 TO 3 GENERATE

           SHIFT_I: ENTITY work.fixed_shifter_right(dataflow)
               GENERIC MAP (L => 2**i)
               PORT MAP ( a => Xl(i),
                          y => Xr(i));

          Xl(i+1) <=  Xr(i) WHEN Y(i)='1' ELSE Xl(i);

     END GENERATE;
```

```
Z <= OTHERS => X(15) WHEN (Y(7) OR Y(6) OR Y(5) OR Y(4)) = '1'
     ELSE X1(4);

END mixed;
```

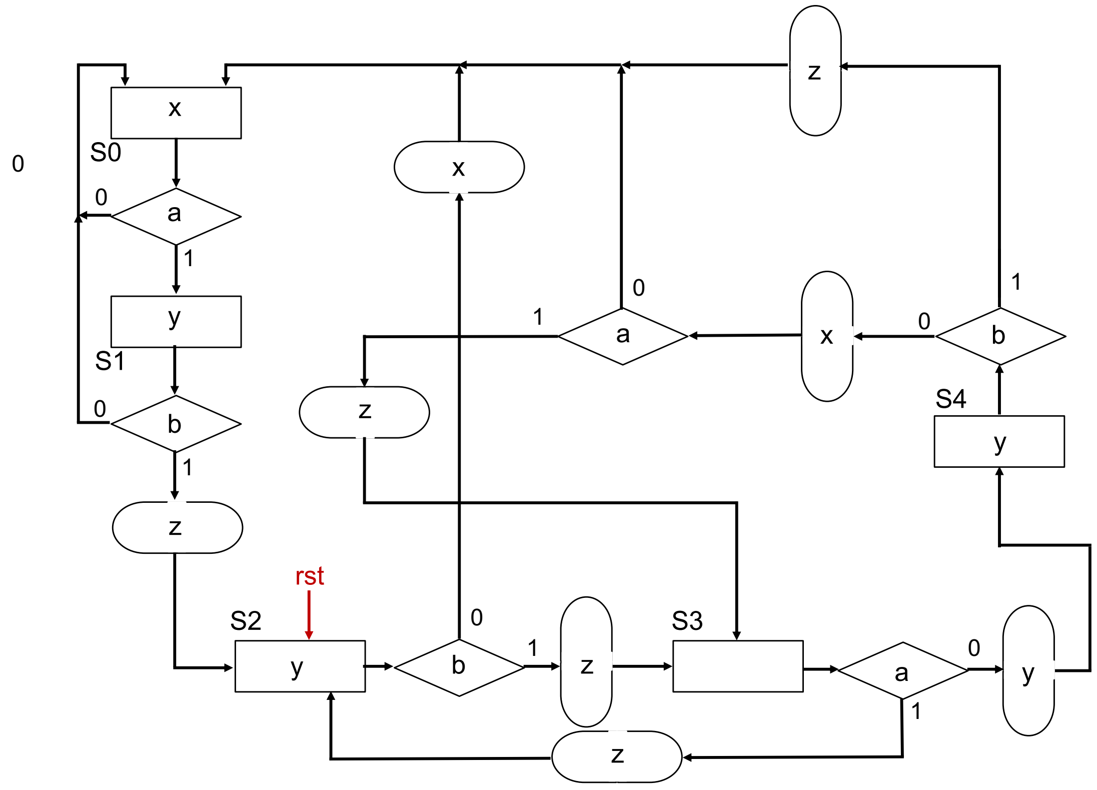
Answer B:



Problem 2

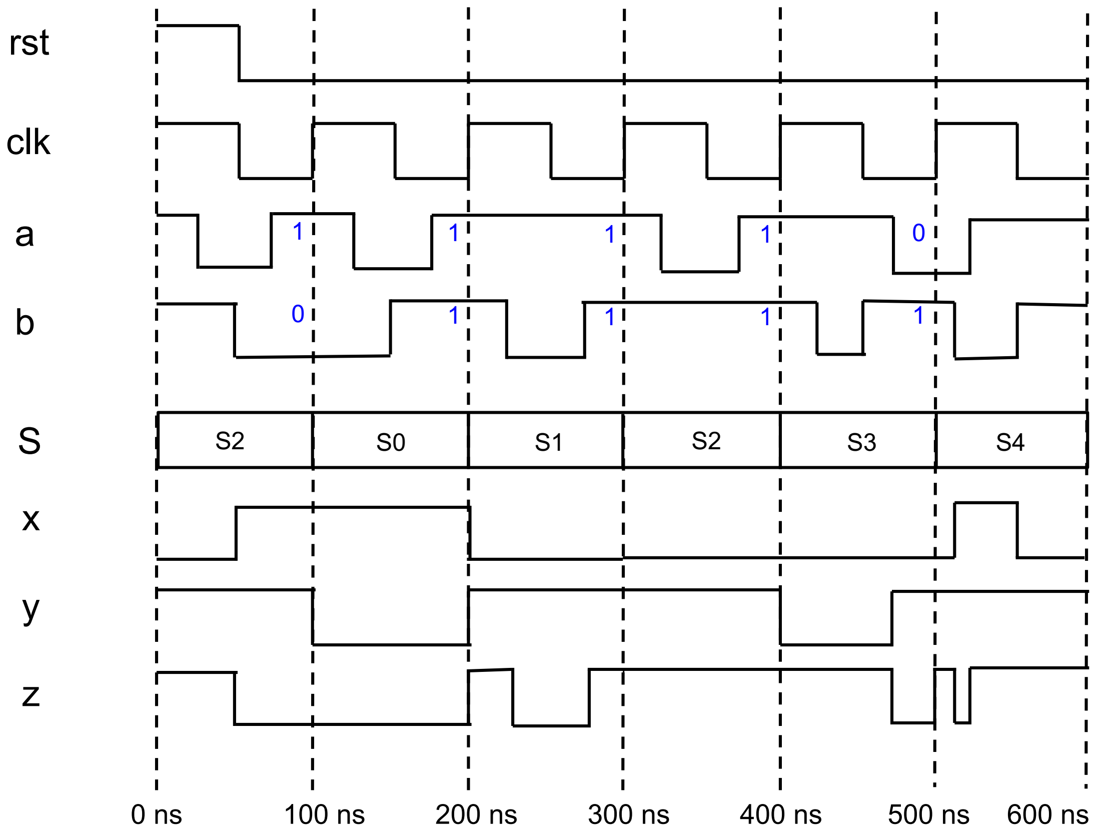
Assuming the controller described using the given below ASM chart:

A. Supplement timing waveforms given on the next page with the values of the **state S**, and the values of the **outputs x**, **y**, and **z.**
B. Fill in the blanks in the fragment of the code of this component given on the next page.



Answer A:

Be careful regarding the initial value of the state S after reset!



Answer B:

```
ARCHITECTURE behavioral of Controller IS
TYPE state IS (S0, S1, S2, S3, S4);
SIGNAL state_reg, state_next: state;

BEGIN
P1: PROCESS(clk, rst)
BEGIN
     IF(rst = '1') THEN
          state_reg <= S2;
     ELSIF rising_edge(clk) THEN
          state_reg <= state_next;
       END IF;
END PROCESS;

Next_State_Output:
PROCESS (state_reg, a, b)
BEGIN
     state_next <= state_reg;
     x <= '0';
     y <= '0';
     z <= '0';
     CASE state_reg IS

             WHEN S4 =>
               y <='1';
               if b='1' then
                    z <='1';
                    state_next <= S0;
               else
                    x <='1';
                    if a ='1' then
                         z <='1';
                         state_next <= S3;
                    else
                         state_next <= S0;
                    end if;
               end if;
```

__**Please provide the code only for the case when state_reg = S4.**__

Problem 3

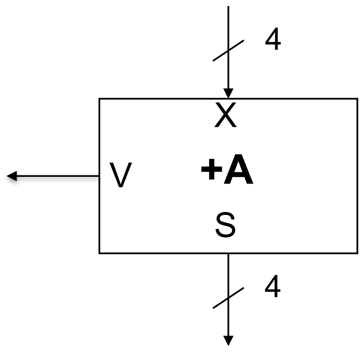
Perform the following tasks for a component

taking a 4-bit input X of the type std_logic_vector, adding to it a constant, given by the generic A of the type integer (in the range -8 to 7), and returning the sum S of the type std_logic_vector and the overflow V of the type std_logic.

A. Draw a symbol of this component with the correct
   - number
   - direction, and
   - widths

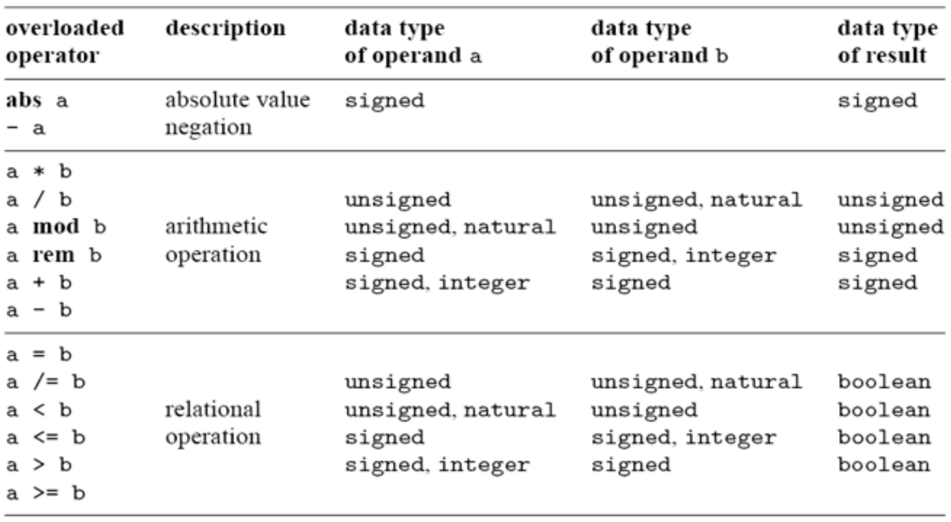
   of ALL inputs and outputs.
   Name each input and output.
B. Write the full synthesizable code of this design entity, including entity declaration and architecture body, using the package numeric_std.
C. **(bonus)** Draw an internal block diagram of this component, composed of medium logic components and/or basic logic gates for the case of A=-6.

Answer A:



Answer B:

You can use the following table of overloaded operators in the IEEE numeric_std package

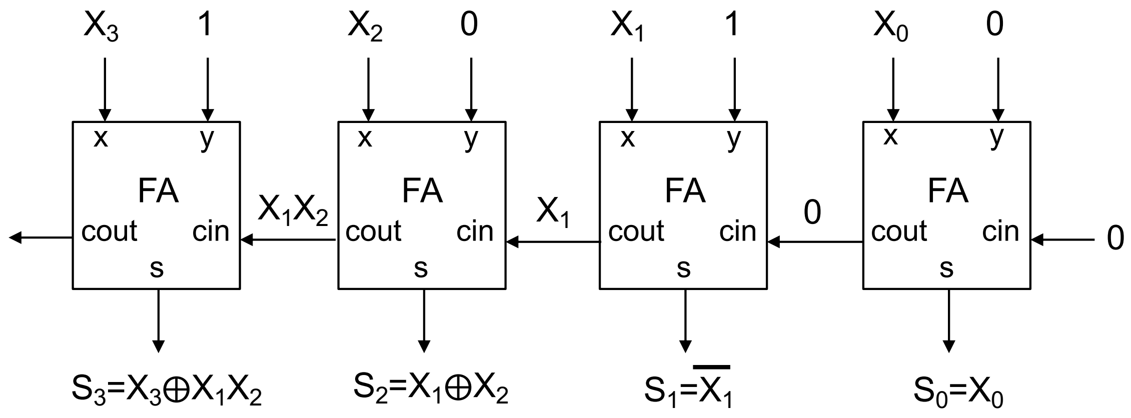


overloaded operator	description	data type of operand a	data type of operand b	data type of result
abs a	absolute value	signed		signed
− a	negation			
a ∗ b				
a / b		unsigned	unsigned, natural	unsigned
a **mod** b	arithmetic	unsigned, natural	unsigned	unsigned
a **rem** b	operation	signed	signed, integer	signed
a + b		signed, integer	signed	signed
a − b				
a = b				
a /= b		unsigned	unsigned, natural	boolean
a < b	relational	unsigned, natural	unsigned	boolean
a <= b	operation	signed	signed, integer	boolean
a > b		signed, integer	signed	boolean
a >= b				

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

ENTITY plusA IS
   GENERIC ( A : INTEGER RANGE -8 TO 7 := -6 ) ;
   PORT ( X : IN STD_LOGIC_VECTOR(3 downto 0);
          S : OUT STD_LOGIC_VECTOR(3 downto 0);
          V : OUT STD_LOGIC);
END plusA;

ARCHITECTURE dataflow OF plusA IS
  SIGNAL Sum : SIGNED(3 downto 0);
BEGIN
  Sum <= signed(X) + A;
  S <= std_logic_vector(Sum);
  V <= (X(3) and to_signed(A, 4)(3) and not Sum(3)) or
       (not X(3) and not to_signed(A, 4)(3) and Sum(3));
END dataflow;
```

Answer C:

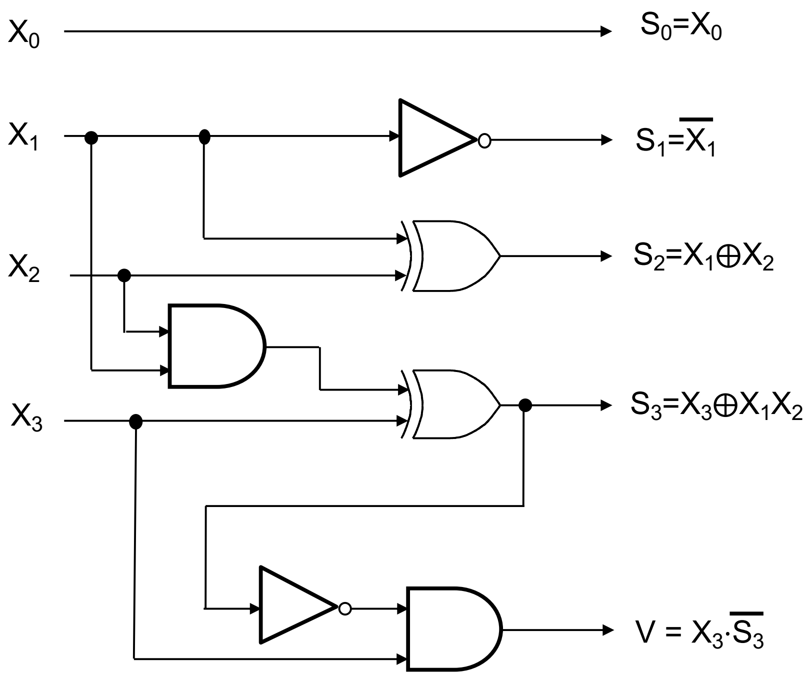
Derivation:



V = 1 when X negative and S positive

$$V = X_3\cdot\overline{S_3} = X_3\cdot\overline{(X_3\oplus X_1X_2)}$$

Final Optimized Circuit:



X_0
X_1
X_2
X_3
$S_0=X_0$
$S_1=\overline{X_1}$
$S_2=X_1 \oplus X_2$
$S_3=X_3 \oplus X_1X_2$
$V = X_3 \cdot \overline{S_3}$