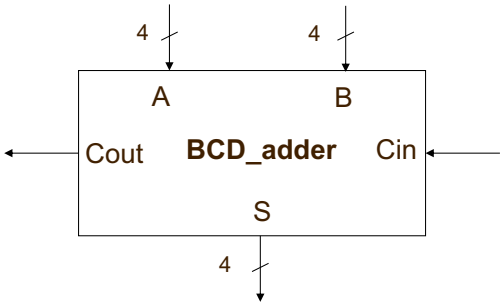
ECE 448
Spring 2021
Midterm Exam for the Lecture
Solutions

Problem 1

A. Symbol of a 1-digit BCD adder



B. RTL code a 1-digit BCD adder using the numeric_std package

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

ENTITY BCD_adder IS
  PORT(A, 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 BCD_adder;

ARCHITECTURE dataflow OF BCD_adder IS

  SIGNAL usum, usum_m10 : UNSIGNED(4 downto 0);
  SIGNAL uCin : UNSIGNED(0 downto 0);

BEGIN

  uCin(0) <= Cin;
  usum <= UNSIGNED('0' & A) + UNSIGNED(B) + uCin;
  usum_m10 <= usum - 10;
  S <= STD_LOGIC_VECTOR(usum(3 DOWNTO 0)) WHEN usum < 10 ELSE
       STD_LOGIC_VECTOR(usum_m10(3 DOWNTO 0));
  Cout <= '1' WHEN usum >= 10 ELSE
          '0';

END dataflow;
```

C. Structural description of a 3-digit BCD adder using the VHDL-93 convention for entity instantiation

Version 1, using FOR-GENERATE:

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

ENTITY Three_digit_BCD_adder IS
  PORT(A, B : IN STD_LOGIC_VECTOR(11 DOWNTO 0);
       Cin : IN STD_LOGIC;
       S : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);
       Cout : OUT STD_LOGIC);
END Three_digit_BCD_adder;

ARCHITECTURE structural OF Three_digit_BCD_adder IS

  SIGNAL Carry : STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN

  Carry(0) <= Cin;
  G: FOR i IN 0 TO 2 GENERATE
    BCD: ENTITY work.BCD_adder(dataflow)
      PORT MAP(
        A => A(4*i+3 DOWNTO 4*i),
        B => B(4*i+3 DOWNTO 4*i),
        Cin => Carry(i),
        S => S(4*i+3 DOWNTO 4*i),
        Cout => Carry(i+1)
      );
  END GENERATE;
  Cout <= Carry(3);

END structural;
```

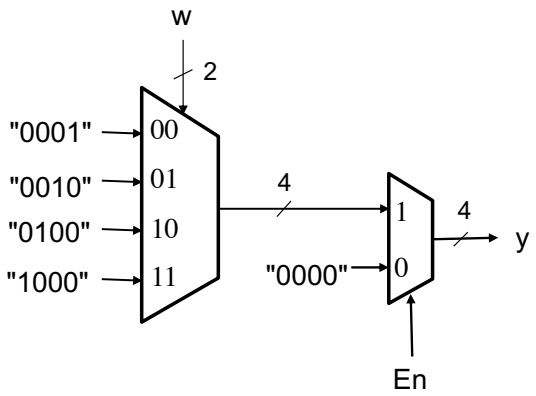
Version 2, without using FOR-GENERATE:

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

ENTITY Three_digit_BCD_adder IS
  PORT(A, B : IN STD_LOGIC_VECTOR(11 DOWNTO 0);
       Cin : IN STD_LOGIC;
       S : OUT STD_LOGIC_VECTOR(11 DOWNTO 0);
       Cout : OUT STD_LOGIC);
END Three_digit_BCD_adder;
```

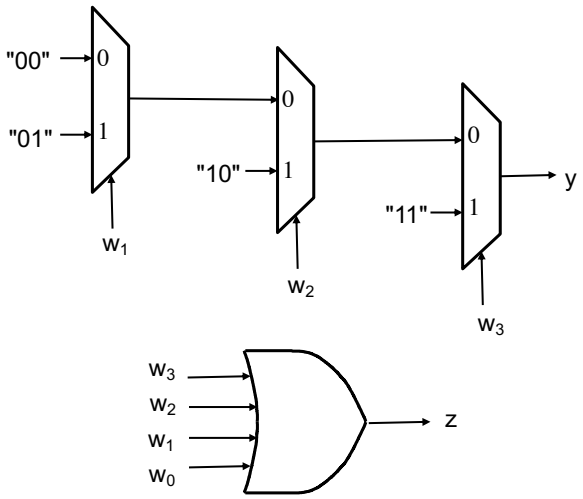
```
ARCHITECTURE structural OF Three_digit_BCD_adder IS

  SIGNAL Carry : STD_LOGIC_VECTOR(2 DOWNTO 1);

BEGIN

  BCD_0: ENTITY work.BCD_adder(dataflow)
    PORT MAP(
      A => A(3 DOWNTO 0),
      B => B(3 DOWNTO 0),
      Cin => Cin,
      S => S(3 DOWNTO 0),
      Cout => Carry(1)
    );

  BCD_1: ENTITY work.BCD_adder(dataflow)
    PORT MAP(
      A => A(7 DOWNTO 4),
      B => B(7 DOWNTO 4),
      Cin => Carry(1),
      S => S(7 DOWNTO 4),
      Cout => Carry(2)
    );

  BCD_2: ENTITY work.BCD_adder(dataflow)
    PORT MAP(
      A => A(11 DOWNTO 8),
      B => B(11 DOWNTO 8),
      Cin => Carry(2),
      S => S(11 DOWNTO 8),
      Cout => Cout
    );

END structural;
```

Problem 2

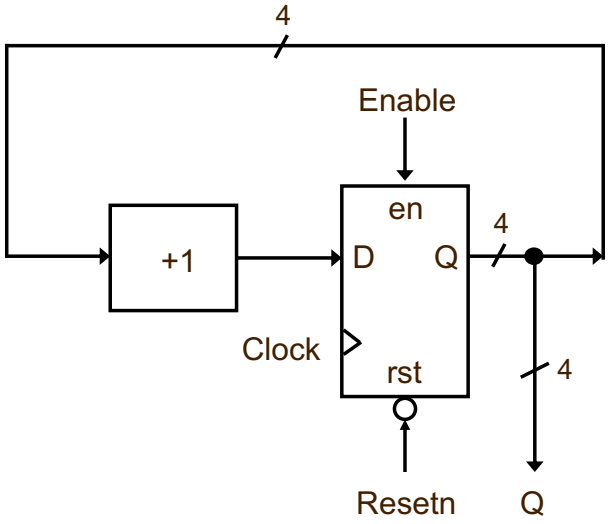
C. 2-to-4 Decoder with Enable



B. Priority Encoder (with the most significant bit of input having the highest priority)



C. 4-bit Up-Counter with Asynchronous Reset Active Low



Problem 3

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

ENTITY P3_Controller IS
    PORT ( clk, reset, go, gt1, gt2, gt3, zi : IN STD_LOGIC;
           esum, enc, en3, en2, en1, s3, s2, done : OUT STD_LOGIC);
END P3_Controller;
```

```
ARCHITECTURE behavioral OF P3_Controller IS

TYPE state IS (Waiting, Calculating, afterloop);
SIGNAL state_reg, state_next : state;

BEGIN

  PROCESS(reset, clk)
  BEGIN
    IF reset = '1' THEN
        state_reg <= Waiting;
    ELSIF rising_edge(clk) THEN
        state_reg <= state_next;
    END IF;
  END PROCESS;

  PROCESS(state_reg, go, gt1, gt2, gt3, zi)
  BEGIN
    state_next <= state_reg;
    esum <= '0';
    enc <= '0';
    en3 <= '0';
    en2 <= '0';
    en1 <= '0';
    s3 <= '0';
    s2 <= '0';
    done <= '0';

    CASE state_reg IS
      WHEN Waiting =>
        IF go = '1' THEN
          state_next <= Calculating;
        END IF;
      WHEN Calculating =>
        esum <= '1';
        IF gt1 = '1' THEN
           s3 <= '0';
           en3 <= '1';
           s2 <= '0';
           en2 <= '1';
           en1 <= '1';
        ELSIF gt2 = '1' THEN
           s3 <= '0';
           en3 <= '1';
           s2 <= '1';
           en2 <= '1';
        ELSIF gt3 = '1' THEN
           s3 <= '1';
           en3 <= '1';
        END IF;
```

```
        IF zi = '1' THEN
           state_next <= afterloop;
        ELSE
           enc <= '1';
           state_next <= Calculating;
        END IF;
      WHEN afterloop =>
        done <= '1';
        state_next <= afterloop;
    END CASE;
  END PROCESS;

END Behavioral;
```