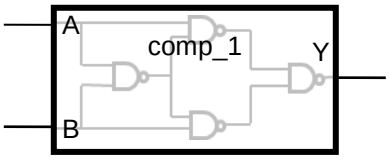
Lab05 Lecture Exercise

1. The following is a single-bit partial comparator similar to what you modeled and synthesized in week 4's lab. According to what you learned today, this comparator is a *component*.

```
ENTITY comp_1 IS
PORT (   A, B : IN      STD_LOGIC;
         Y  : OUT     STD_LOGIC
      );
END comp_1;
```

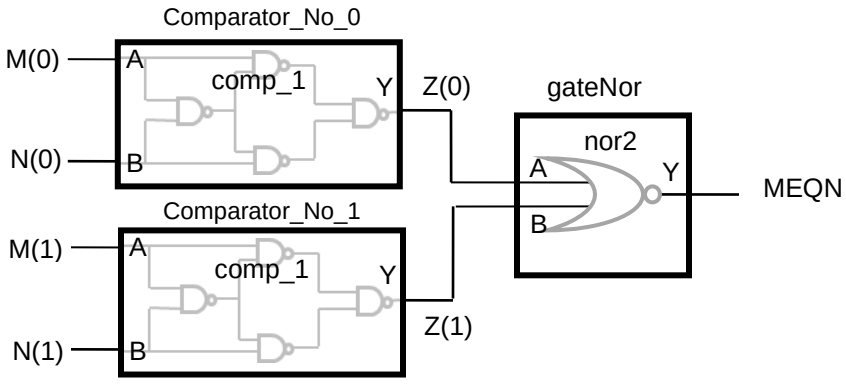



```
ARCHITECTURE Algebraic OF comp_1 IS      -- ARCHITECTURE tells us about the inside of circuit.
    SIGNAL T, U, W: STD_LOGIC;           -- Intermediate signals
BEGIN
    Y  <=  (U NAND W);
    W  <=  (B NAND T);
    U  <=  (A NAND T);
    T  <=  (A NAND B);
END Algebraic;
```

Based on what you see above:

- What is the *component's* name?
- What are the *formals* of the component?

Using the above component, you are going to *structurally* model a 2-bit partial comparator in VHDL. The structural model is *graphically* shown below. As you see, there are two instances of the above component in this 2-bit partial comparator. Additionally, a second component called nor2 is used with only one instance:



- What is the name of the first *instance* of comp_1 (the one on the top)?

- Write the *actuals* of the second instance of comp_1:

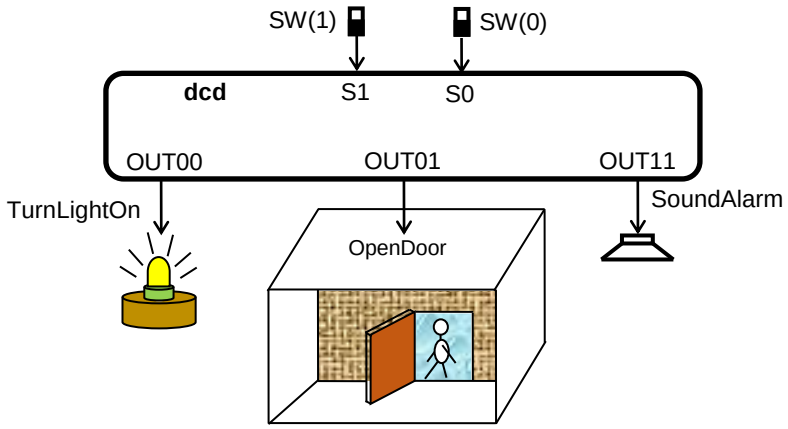
- A 4-bit partial comparator is **structurally** modeled here. Make all the necessary changes to convert it to a 2-bit comparator:

```
ENTITY comp_4h IS
PORT (M, N   : IN          STD_LOGIC_VECTOR (3 DOWNTO 0);
       MEQN : OUT       STD_LOGIC    -- Active-high output
       );
END comp_4h;

ARCHITECTURE Structure OF comp_4h IS

   SIGNAL Z: STD_LOGIC_VECTOR (3 DOWNTO 0);   -- Intermediate signals

   COMPONENT comp_1
   PORT (A,   B      : IN          STD_LOGIC;
              Y      : OUT       STD_LOGIC
         );
   END COMPONENT;

   COMPONENT nor4
   PORT (A, B, C, D: IN          STD_LOGIC;
                  Y : OUT       STD_LOGIC
        );
   END COMPONENT;
BEGIN
   First_Comparator: comp_1
   PORT MAP  (M(0),  N(0), Z(0));

   Second_Comparator: comp_1
   PORT MAP  (M(1), N(1), Z(1));

   Third_Comparator: comp_1
   PORT MAP  (M(2), N(2), Z(2));

   Fourth_Comparator: comp_1
   PORT MAP  (M(3), N(3), Z(3));

   -- NOR gate is added structurally (not algebraically)
   -- NOR gate needs its own file (entity and architecture) not shown here.
   -- You need to compile ALL the files when doing simulation or synthesis.

   gateNor: nor4
   PORT MAP       (Z(3),   Z(2),   Z(1),   Z(0), MEQN);

END Structure;
```

2. The entity of a component called dcd is shown here:

```
ENTITY dcd IS
PORT (    s1, s0      : IN        STD_LOGIC;
          OUT00,
          OUT01,
          OUT11       : OUT       STD_LOGIC
      );
END dcd;
```

This component asserts output OUT00, OUT01 or OUT11 when binary values 00, 01 or 11, respectively, are applied to inputs s1s0. Scott, who has not taken Digital Systems I yet (but digital design is his hobby), wants to use this dcd to control a light, a door and an alarm as shown in the following test bench. **Note** that the actuals associated with formals OUT00, OUT01 and OUT11 are TurnLightOn, OpenDoor and SoundAlarm, respectively:



Therefore,
If SW1:0 are set to 00, then the light should turn on.
If SW1:0 are set to 01, then the door should open.
If SW1:0 are set to 11, then the alarm should sound.

Here is how Scott has instantiated dcd:

```
Controller: dcd  PORT MAP (SW(1), SW(0), OpenDoor, SoundAlarm TurnLightOn);
```

Unfortunately, he realizes a weird operation when he tests his controller:
When he wants to open the door, the alarm sounds instead!
When he tries to sound the alarm, the light turns on instead!
When he wants to turn the light on, the door opens instead!

Help Scott figure out what is wrong with his design, and then correct it.