Problem

Design and implement a state machine with 3-bit inputs which detects a sequence of inputs with 2 ones, 1 one and 2 ones.

```
entity seq_det_wt is
port (              x : in std_logic_vector(2 downto 0);
  reset, ck, b0, b1 : in std_logic;
                    z : out std_logic
      );
```

Verify correctness using single-step mechanism buttons b0 and b1. After a reset apply a sequence of the inputs (switches) and observe the output z (LED)

x :	“100”,	“101”,	“001”,	“110”,	“010”,
# of 1's :	1	2	1	2	1
z :	0	0	0	1	0

```
-----------------------
-- Solution without # of 1's combinational block
-----------------------
--process(en)
--begin
--if en='1' and en'event then
-- if reset = '1' then
--  n_s <= s0; z <= '0'; else
--  case n_s is
--   when s0 =>
--    if x = "011" or x = "101" or x = "110" then
--    n_s <= s1; z <= '0'; else z <= '0'; end if;
--   when s1 =>
--    if x = "001" or x = "010" or x = "100" then
--    n_s <= s2; z <= '0'; else z <= '0'; end if;
--   when s2 =>
--    if x = "011" or x = "101" or x = "110" then
--    n_s <= s3; z <= '1'; else z <= '0'; end if;
--   when s3 =>
--    if x = "001" or x = "010" or x = "100" then
--    n_s <= s2; z <= '0'; else z <= '0'; end if;
--   when others => null;
--  end case;
-- end if;
--end if;
--end process;
-----------------------
```

```
----------------------------------------------------
-- Company: Drexel ECE
-- Engineer: Prawat
-- State machine with 3-bit inputs which detects a
-- sequence of inputs with 2 ones, 1 one and 2 ones.
----------------------------------------    --------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity seq_det_wt is
port (          x : in  std_logic_vector(2 downto 0);
reset,ck,btn0,btn1: in  std_logic;
                z : out std_logic
     );
end seq_det_wt;

architecture Behavioral of seq_det_wt is
type my_state is (s0, s1, s2, s3);
signal n_s : my_state;
signal en: std_logic;

begin
```

```
-- Solution with # of 1's combinational block
--      --------       -------
--     |        |     |       |
-- -->|# of 1's|---->|State M|--> z
-- x   |        |  w  |       |
--      --------       -------

-- combinational block for # of 1's
-- w is the wire connected to state machine
process(x)
variable wt : my_int;
begin
wt := 0;
for i in 0 to 2 loop
if x(i) = '1' then wt := wt + 1; end if;
end loop;
w <= wt;
end process;
```

```
-- State Machine
process(en)
begin
if en='1' and en'event then
 if reset = '1' then
  n_s <= s0; z <= '0'; else
  case n_s is
   when s0 =>
    if w = 2 then
    n_s <= s1; z <= '0'; else z <= '0'; end if;
   when s1 =>
    if w = 1 then
    n_s <= s2; z <= '0'; else z <= '0'; end if;
   when s2 =>
    if w = 2 then
    n_s <= s3; z <= '1'; else z <= '0'; end if;
   when s3 =>
    if w = 2 then
    n_s <= s2; z <= '0'; else z <= '0'; end if;
   when others => null;
  end case;
 end if;
end if;
end process;
```

```
-- single step, debounce (db)
-- btn0 to enter and btn1 to reset
process(ck)
type db_state is (not_rdy, rdy, pulse);
variable db_ns: db_state;
begin
if ck='1' and ck'event then
  case db_ns is
   when not_rdy => en <= '0';
   if btn1 = '1' then db_ns := rdy; end if;
   when rdy => en <= '0';
   if btn0 = '1' then db_ns := pulse; end if;
   when pulse => en <= '1';
   db_ns := not_rdy;
   when others => null;
  end case;
end if;
end process;

end Behavioral;
```