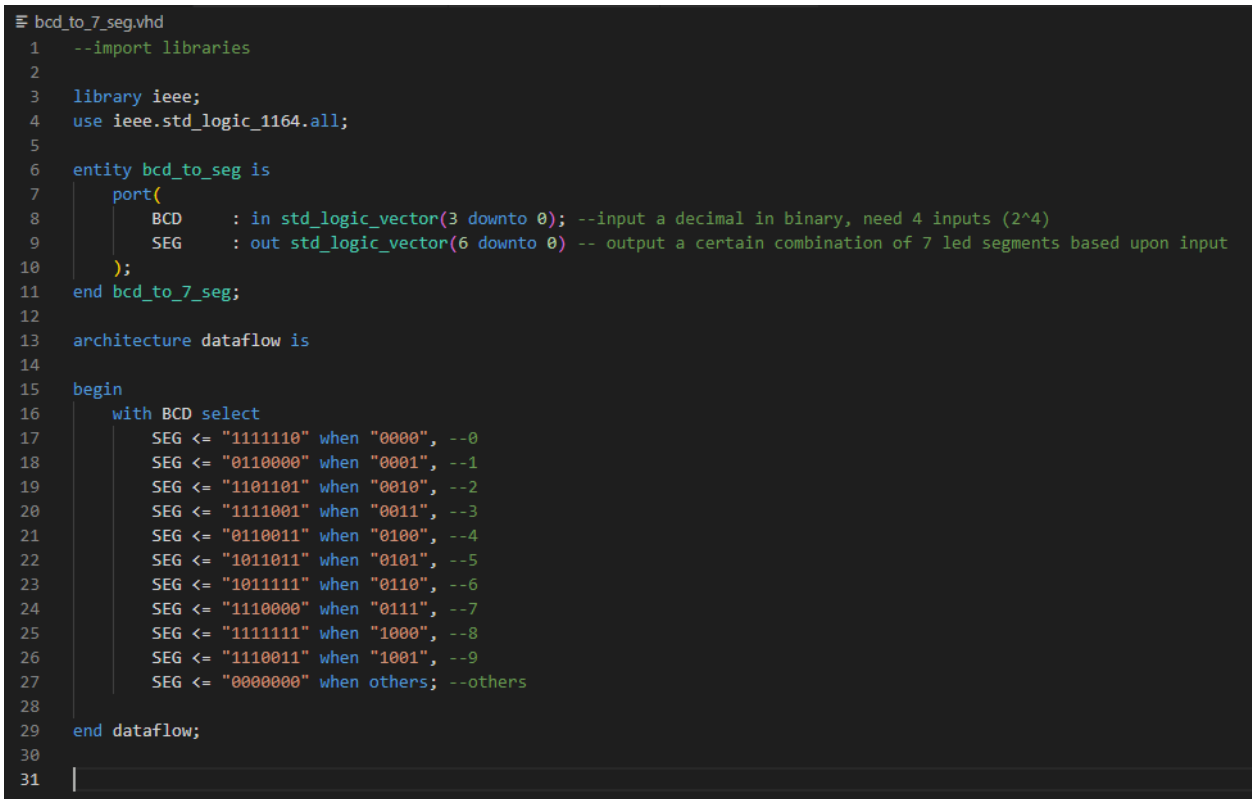
Jonathan Mazurkiewicz

Homework #2

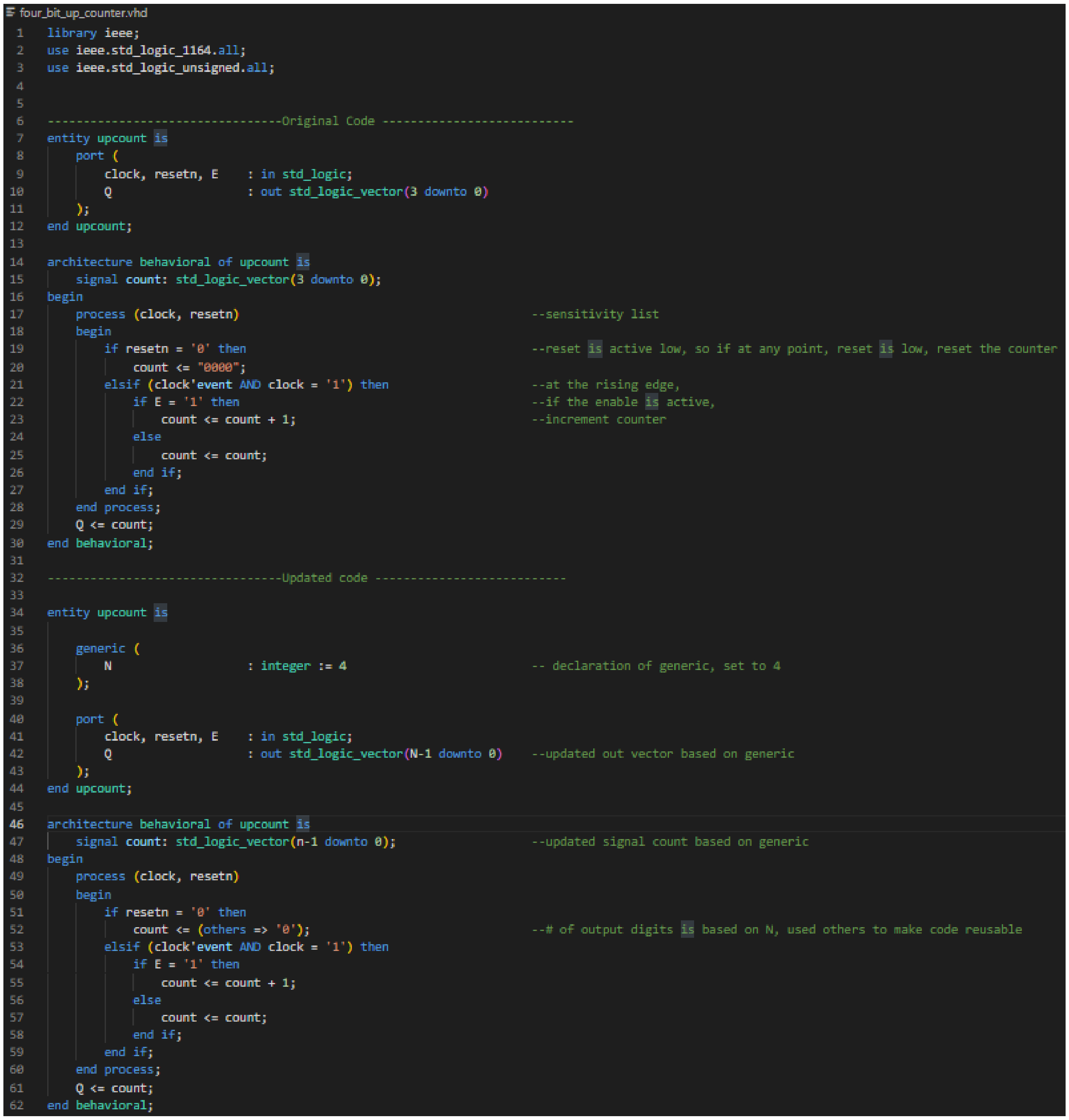
Problem 1:



bcd_to_7_seg.vhd

```vhdl
--import libraries

library ieee;
use ieee.std_logic_1164.all;

entity bcd_to_seg is
    port(
        BCD     : in std_logic_vector(3 downto 0); --input a decimal in binary, need 4 inputs (2^4)
        SEG     : out std_logic_vector(6 downto 0) -- output a certain combination of 7 led segments based upon input
    );
end bcd_to_7_seg;

architecture dataflow is

begin
    with BCD select
        SEG <= "1111110" when "0000", --0
        SEG <= "0110000" when "0001", --1
        SEG <= "1101101" when "0010", --2
        SEG <= "1111001" when "0011", --3
        SEG <= "0110011" when "0100", --4
        SEG <= "1011011" when "0101", --5
        SEG <= "1011111" when "0110", --6
        SEG <= "1110000" when "0111", --7
        SEG <= "1111111" when "1000", --8
        SEG <= "1110011" when "1001", --9
        SEG <= "0000000" when others; --others

end dataflow;

```

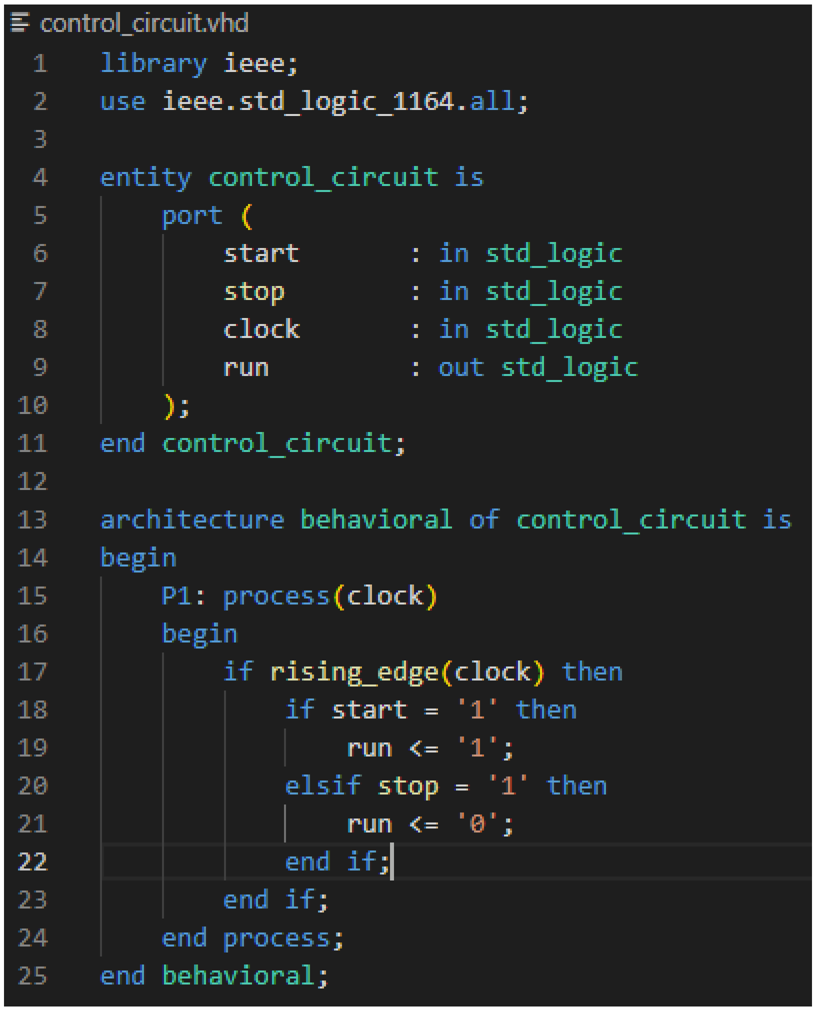
Problem 2:



four_bit_up_counter.vhd

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


--------------------------------Original Code --------------------------
entity upcount is
    port (
        clock, resetn, E    : in std_logic;
        Q                   : out std_logic_vector(3 downto 0)
    );
end upcount;

architecture behavioral of upcount is
    signal count: std_logic_vector(3 downto 0);
begin
    process (clock, resetn)                                         --sensitivity list
    begin
        if resetn = '0' then                                        --reset is active low, so if at any point, reset is low, reset the counter
            count <= "0000";
        elsif (clock'event AND clock = '1') then                    --at the rising edge,
            if E = '1' then                                         --if the enable is active,
                count <= count + 1;                                 --increment counter
            else
                count <= count;
            end if;
        end if;
    end process;
    Q <= count;
end behavioral;

--------------------------------Updated code --------------------------

entity upcount is

    generic (
        N                   : integer := 4                          -- declaration of generic, set to 4
    );

    port (
        clock, resetn, E    : in std_logic;
        Q                   : out std_logic_vector(N-1 downto 0)    --updated out vector based on generic
    );
end upcount;

architecture behavioral of upcount is
    signal count: std_logic_vector(n-1 downto 0);                   --updated signal count based on generic
begin
    process (clock, resetn)
    begin
        if resetn = '0' then
            count <= (others => '0');                               --# of output digits is based on N, used others to make code reusable
        elsif (clock'event AND clock = '1') then
            if E = '1' then
                count <= count + 1;
            else
                count <= count;
            end if;
        end if;
    end process;
    Q <= count;
end behavioral;
```

Problem 3:

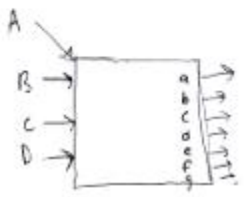


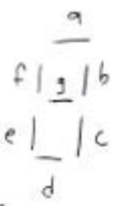
control_circuit.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity control_circuit is
    port (
        start       : in std_logic
        stop        : in std_logic
        clock       : in std_logic
        run         : out std_logic
    );
end control_circuit;

architecture behavioral of control_circuit is
begin
    P1: process(clock)
    begin
        if rising_edge(clock) then
            if start = '1' then
                run <= '1';
            elsif stop = '1' then
                run <= '0';
            end if;
        end if;
    end process;
end behavioral;
```

WORK:

Problem 1:





* Similar to a 4-to-16 decoder, but with ~~re~~ ignoring 10-15 outputs.

1	b, c
2	a, b, g, e, d
3	a, b, g, c, d
4	f, g, b, c
5	a, f, g, c, d
6	a, f, g, e, c, d
7	a, b, c
8	ALL
9	ALL except 'e'
0	ALL except 'g'

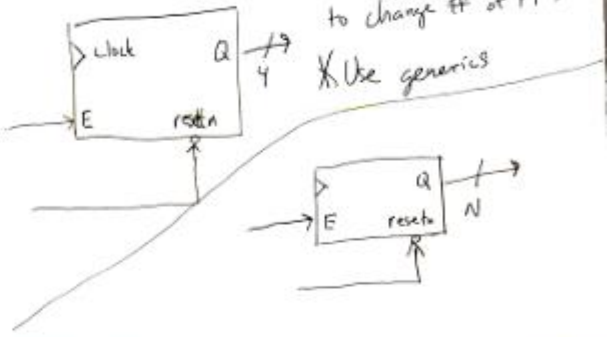
none when others

Problem 2

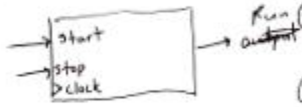
* Asynchronous reset

* Need to be able to change # of FF's

* Use generics



Problem 3:



① when start high, run = 1

Run ~~output~~ ② Stay high until stop is high, then go back low.

③ Sync w/ clock.

thoughts:

- output needs to be based on rising edge of clock.
- stop is like a reset, but active high.
- sounds like behavioral architecture.
- start can change after the beginning and output is not affected.
- will be synchronous FF
  - does not work, because start changes,

Desired truth table

start	stop	run
0	0	0
1	0	1
-	1	0

clk

clk↑, start high, store run high

clk↑, stop high store run low