```
--      John Adams
--      11565123
--      WSU EE 324
--      VHDL StopWatch
--
----------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity stopwatch is
GENERIC (    clock_freq: INTEGER := 100000000 );

Port ( led : out  STD_LOGIC_VECTOR (3 downto 0);
       sw : in  STD_LOGIC_VECTOR (7 downto 0);
       an : out STD_LOGIC_VECTOR (3 downto 0) := "0000";
       seg : out STD_LOGIC_VECTOR (7 downto 0);
       btn : IN STD_LOGIC_VECTOR (3 downto 0);
       clk : in STD_LOGIC);
end stopwatch;

ARCHITECTURE Behavioral OF stopwatch IS     -- Define variables for
'stopwatch'

SIGNAL secs : INTEGER := 0;
SIGNAL tens_secs : INTEGER := 0;
SIGNAL millis : INTEGER := 0;
SIGNAL tens_millis : INTEGER := 0;
SIGNAL hundreds_millis : INTEGER := 0;
SIGNAL ssd_secs: STD_LOGIC_VECTOR (7 DOWNTO 0) := "11111111";
SIGNAL led_t_secs: STD_LOGIC_VECTOR (3 DOWNTO 0) := "0000";
SIGNAL ssd_millis: STD_LOGIC_VECTOR (7 DOWNTO 0) := "11111111";
SIGNAL ssd_tens_millis: STD_LOGIC_VECTOR (7 DOWNTO 0) :=
"11111111";
SIGNAL ssd_hundreds_millis: STD_LOGIC_VECTOR (7 DOWNTO 0) :=
```

```
"11111111";
SIGNAL m_clk: STD_LOGIC := '0';
SIGNAL millis_divisor : INTEGER := clock_freq/1000;
SIGNAL clk_counter :INTEGER := 0;
SIGNAL counter : INTEGER := 0;
SIGNAL rst : STD_LOGIC := '0';
SIGNAL incstate : INTEGER := 0;
SIGNAL en : STD_LOGIC := '1';
SIGNAL inc : STD_LOGIC := '0';
SIGNAL stop : STD_LOGIC := '0';
SIGNAL start : STD_LOGIC := '0';


BEGIN

led <= led_t_secs;
start <= btn(0);
stop <= btn(1);
inc <= btn(2);
rst <= btn(3);

PROCESS (clk)       -- Here is the clock divider to get it counting
in seconds
BEGIN
    IF (clk'EVENT and clk = '1') THEN
        IF (clk_counter = millis_divisor) THEN
            clk_counter <= 0;
            m_clk <= '1';
        ELSE
            clk_counter <= clk_counter + 1;
            m_clk <= '0';
        END IF;
    END IF;
END PROCESS;


PROCESS (m_clk, en, rst)        -- counter for the digits
```

```
BEGIN
    IF (rst = '1') THEN        --Set everything to zero to reset
the clock
        millis <= 0;
        tens_millis <= 0;
        hundreds_millis <= 0;
        secs <= 0;
        tens_secs <= 0;
    ELSIF(m_clk'EVENT and m_clk = '1') THEN
        IF (en = '1')
        THEN
            millis <= millis + 1;
            IF (millis = 9)
            THEN
                millis <= 0;
                tens_millis <= tens_millis + 1;
                IF (tens_millis = 9)
                THEN                                    --Each one of
these is when the next digit should be incrimented.
                    tens_millis <= 0;
                    hundreds_millis <= hundreds_millis + 1;
                    IF (hundreds_millis = 9)
                    THEN
                        secs <= secs + 1;
                        hundreds_millis <= 0;
                        IF (secs = 9)
                        THEN
                            tens_secs <= tens_secs + 1;
                            secs <= 0;
                            IF (tens_secs = 15)
                            THEN
                                tens_secs <= 0;    -- Reached the
max, now time to reset everything.
                            END IF;
                        END IF;
                    END IF;
                END IF;
```

```
            END IF;


        ELSIF(inc = '1' AND (incstate = 0))
        THEN                                        -- for the
incrament button. This was a b*tch in verilog but I got it working
here.
            incstate <= incstate + 1;
            millis <= millis + 1;
            IF (millis = 9)
            THEN                                -- Uses the same logic
as above for the increment button
                millis <= 0;
                tens_millis <= tens_millis + 1;
                IF (tens_millis = 9)
                THEN
                    tens_millis <= 0;
                    hundreds_millis <= hundreds_millis + 1;
                    IF (hundreds_millis = 9)
                    THEN
                        secs <= secs + 1;
                        hundreds_millis <= 0;
                        IF (secs = 9)
                        THEN
                            tens_secs <= tens_secs + 1;
                            secs <= 0;
                            IF (tens_secs = 15)
                            THEN
                                tens_secs <= 0;    -- Max count
reached, reset t_secs
                            END IF;
                        END IF;
                    END IF;
                END IF;
            END IF;

            ELSIF (inc = '1' AND (incstate > 0) AND (incstate <
```

```
30)) -- This gets fed in above and makes it so there is no switch
bouncing, only gets one incrament
            THEN
 -- per button push instead of mulitple buttons.
                incstate <= incstate + 1;

            ELSIF (inc = '0' AND (incstate > 0))
            THEN
                incstate <= incstate - 1;
        END IF;
    END IF;
END PROCESS;


WITH millis
SELECT          --Decoder for first digit
    ssd_millis <=
        "11000000" when 0,
        "11111001" when 1,
        "10100100" when 2,
        "10110000" when 3,
        "10011001" when 4,
        "10010010" when 5,
        "10000010" when 6,
        "11111000" when 7,
        "10000000" when 8,
        "10010000" when 9,
        "10000110" WHEN OTHERS;

WITH tens_millis
SELECT         -- Decoder for second digit
    ssd_tens_millis <=
        "11000000" when 0,
        "11111001" when 1,
        "10100100" when 2,
        "10110000" when 3,
        "10011001" when 4,
```

```
        "10010010" when 5,
        "10000010" when 6,
        "11111000" when 7,
        "10000000" when 8,
        "10010000" when 9,
        "10000110" WHEN OTHERS;

WITH hundreds_millis
SELECT          --Decoder for third digit
    ssd_hundreds_millis <=
        "11000000" when 0,
        "11111001" when 1,
        "10100100" when 2,
        "10110000" when 3,
        "10011001" when 4,
        "10010010" when 5,
        "10000010" when 6,
        "11111000" when 7,
        "10000000" when 8,
        "10010000" when 9,
        "10000110" WHEN OTHERS;

WITH secs
SELECT                  --Decoder for fourth digit
    ssd_secs <=
        "11000000" when 0,
        "11111001" when 1,
        "10100100" when 2,
        "10110000" when 3,
        "10011001" when 4,
        "10010010" when 5,
        "10000010" when 6,
        "11111000" when 7,
        "10000000" when 8,
        "10010000" when 9,
        "10000110" WHEN OTHERS;
```

```
WITH tens_secs
SELECT
    led_t_secs <=
        "0000" when 0,
        "0001" when 1,
        "0010" when 2,
        "0011" when 3,
        "0100" when 4,
        "0101" when 5,
        "0110" when 6,
        "0111" when 7,
        "1000" when 8,
        "1001" when 9,
        "1010" when 10,
        "1011" when 11,
        "1100" when 12,
        "1101" when 13,
        "1110" when 14,
        "1111" when 15,
        "1111" WHEN OTHERS;


PROCESS(clk)
BEGIN
    an <= "1111";
        IF (clk'EVENT AND clk='1')
        THEN
            counter <= counter + 1;

        IF(counter > 150 and counter < 200) -- Displays the first
digit
        THEN
            an <= "0111";
            seg <= ssd_secs AND "01111111";


        ELSIF (counter > 250 and counter < 300)
```

```
        THEN                                            -- Display
second digit
            an <= "1011";
            seg <= ssd_hundreds_millis;

        ELSIF (counter > 350 and counter < 400)
        THEN                                            -- Display
third digit
            an <= "1101";
            seg <= ssd_tens_millis;

        ELSIF (counter > 450 and counter < 500)
        THEN                                            -- Display
fourth digit
            an <= "1110";
            seg <= ssd_millis;

        ELSIF (counter >499) THEN           -- Reaches the max and
restarts the count variable.
        counter <= 1;

        ELSE
        an <= "1111";
        seg <= "11111111";
        END IF;
        END IF;
END PROCESS;


PROCESS (clk)
BEGIN
    IF (clk'EVENT and clk = '1')
    THEN
        IF (start = '1' AND stop = '0')
        THEN
            en <= '1';
        ELSIF (stop = '1' AND start = '0')
```

```
        THEN
            en <= '0';
        ELSE
            en <= en;
        END IF;
    END IF;
END PROCESS;

END Behavioral;
```