```
library IEEE;
use IEEE.NUMERIC_STD.All;
use IEEE.STD_LOGIC_1164.ALL;

entity Stopwatch is port ( CEN, RST : in STD_LOGIC;
                           CLK: in STD_LOGIC; CATH : out STD_LOGIC_VECTOR(7
downto 0);
                           AN : out STD_LOGIC_VECTOR(3 downto 0));
end Stopwatch;

architecture behavioral of Stopwatch is

    --new clock signals driven by the Basys3 internal 100MHz clock
    signal CLK_1cs : STD_LOGIC := '0';
    signal CLK_240Hz : STD_LOGIC := '0';

    --signal that starts and stops the cathode counting logic
    signal ENABLE : STD_LOGIC := '0';

    --signals used to sense button rising edges
    signal PB_start, PB_reset, PB_sig_1, PB_sig_2, stop_sig : STD_LOGIC :=
'0';

    -- cathode signal buses including the decimal point
    signal CATHODE1 : STD_LOGIC_VECTOR(7 downto 0)  := "00000010";
    signal CATHODE2 : STD_LOGIC_VECTOR(7 downto 0)  := "00000010";
    signal CATHODE3 : STD_LOGIC_VECTOR(7 downto 0)  := "00000011";
    signal CATHODE4 : STD_LOGIC_VECTOR(7 downto 0)  := "00000010";

begin

process(CLK_1cs) begin
    if(rising_edge(CLK_1cs)) then
        if(CEN = '1') then -- START/STOP button rising-edge toggle logic
            PB_start <= '1';
        elsif(CEN = '0') then
            PB_start <= '0';
        end if;

        PB_sig_1 <= PB_start;
        if(PB_sig_1 = '0' and PB_start = '1') then
            ENABLE <= not ENABLE;
        end if;

        if(stop_sig = '1') then --stop_sig senses 60.00 has been reached
            ENABLE <= '0'; --stop cathode counting logic
        end if;
    end if;
end process;

--SET UP CENTISECOND AND 240Hz CLOCK SIGNALS
process(CLK) --Set up clock signal CLK_1s
    variable count_1cs : unsigned(18 downto 0) := to_unsigned(0,19); --first
digit clock counter, running at 1 centisecond)
    variable count_240Hz : unsigned(15 downto 0) := to_unsigned(0,16); --
anode rotate frequency
begin
```

```
    if(rising_edge(CLK)) then
        count_1cs := count_1cs + 1; --add to count_1cs on every CLK rising
edge
            if(count_1cs = to_unsigned(500000,19)) then -- check count for
desired clock frequencies
                CLK_1cs <= not CLK_1cs; --toggle CLK_1cs
                count_1cs := to_unsigned(0,19); -- reset count_1cs to 0 when
CLK_1cs toggle has been reached
            end if;

        count_240Hz := count_240Hz + 1; --add to count_240Hz on every CLK
rising edge
            if(count_240Hz = to_unsigned(41667,16)) then -- anode rotation
frequency
                CLK_240Hz <= not CLK_240Hz; --toggle CLK_240Hz
                count_240Hz := to_unsigned(0,16); -- reset variable to 0 when
CLK_240Hz toggle has been reached
            end if;
    end if;
end process;

--CATHODE LOGIC
process(CLK, CLK_1cs, RST, ENABLE)
    variable count1 : unsigned := "0000"; --1st digit count variable
    variable count2 : unsigned := "0000"; --2nd digit count variable
    variable count3 : unsigned := "0000"; --3rd digit count variable
    variable count4 : unsigned := "0000"; --4th digit count variable
begin

    if(rising_edge(CLK_1cs)) then
        if(ENABLE = '0') then -- if counting is paused
            if(RST = '1') then -- reset button-press rising edge detection
logic
                PB_reset <= '1';
            elsif(RST = '0') then
                PB_reset <= '0';
            end if;

            PB_sig_2 <= PB_reset;
            if(PB_sig_2 = '0' and PB_reset = '1') then
                count1 := "0000"; --reset all count variables to 0
                count2 := "0000";
                count3 := "0000";
                count4 := "0000";
                stop_sig <= '0'; --reset the stop signal to 0
            end if;

        elsif(ENABLE = '1') then --cathode counting logic won't run unless
ENABLE is '1'
            count1 := count1 + 1; --counting logic

            --use the CLK_1cs signal for the first digit to run the counting
for the rest of the digits
            if(count1 = "1010") then --when count1 is 10...
                count2 := count2 + 1; --add 1 to count2
                count1 := "0000"; --and reset count1 back to 0
            end if;

            if(count2 = "1010") then --use previous logic to run count3
                count3 := count3 + 1;
                count2 := "0000";
            end if;

            if(count3 = "1010") then --and so on
                count4 := count4 + 1;
                count3 := "0000";
            end if;

            if(count4 = "0101" and count3 = "1001" and count2 = "1001" and
count1 = "1001") then --stop signal logic
                stop_sig <= '1'; --if statement senses 59.99 so that stop_sig
stops the stopwatch on 60.00
            end if;
        end if;
    end if;

    -- CASE STATEMENTS TO DEFINE CATHODE OUTPUT BASED ON COUNT VARIABLE
VALUES
    case count1 is
        when "0000" => CATHODE1 <= "00000011"; --0
        when "0001" => CATHODE1 <= "10011111"; --1
        when "0010" => CATHODE1 <= "00100101"; --2
        when "0011" => CATHODE1 <= "00001101"; --3
        when "0100" => CATHODE1 <= "10011001"; --4
        when "0101" => CATHODE1 <= "01001001"; --5
        when "0110" => CATHODE1 <= "01000001"; --6
        when "0111" => CATHODE1 <= "00011111"; --7
        when "1000" => CATHODE1 <= "00000001"; --8
        when "1001" => CATHODE1 <= "00011001"; --9
        when others => CATHODE1 <= "11111111"; --nothing
    end case;

    case count2 is
        when "0000" => CATHODE2 <= "00000011"; --0
        when "0001" => CATHODE2 <= "10011111"; --1
        when "0010" => CATHODE2 <= "00100101"; --2
        when "0011" => CATHODE2 <= "00001101"; --3
        when "0100" => CATHODE2 <= "10011001"; --4
        when "0101" => CATHODE2 <= "01001001"; --5
        when "0110" => CATHODE2 <= "01000001"; --6
        when "0111" => CATHODE2 <= "00011111"; --7
        when "1000" => CATHODE2 <= "00000001"; --8
        when "1001" => CATHODE2 <= "00011001"; --9
        when others => CATHODE2 <= "11111111"; --nothing
    end case;

    case count3 is -- all of CATHODE3(0) are 0 because the decimal point for
this digit needs to be lit
        when "0000" => CATHODE3 <= "00000010"; --0
        when "0001" => CATHODE3 <= "10011110"; --1
        when "0010" => CATHODE3 <= "00100100"; --2
        when "0011" => CATHODE3 <= "00001100"; --3
        when "0100" => CATHODE3 <= "10011000"; --4
        when "0101" => CATHODE3 <= "01001000"; --5
        when "0110" => CATHODE3 <= "01000000"; --6
        when "0111" => CATHODE3 <= "00011110"; --7
        when "1000" => CATHODE3 <= "00000000"; --8
        when "1001" => CATHODE3 <= "00011000"; --9
        when others => CATHODE3 <= "11111110"; --nothing
    end case;

    case count4 is
        when "0000" => CATHODE4 <= "11111111"; --0
        when "0001" => CATHODE4 <= "10011111"; --1
        when "0010" => CATHODE4 <= "00100101"; --2
        when "0011" => CATHODE4 <= "00001101"; --3
        when "0100" => CATHODE4 <= "10011001"; --4
        when "0101" => CATHODE4 <= "01001001"; --5
        when "0110" => CATHODE4 <= "01000001"; --6
        when "0111" => CATHODE4 <= "00011111"; --7
        when "1000" => CATHODE4 <= "00000001"; --8
        when "1001" => CATHODE4 <= "00011001"; --9
        when others => CATHODE4 <= "11111111"; --nothing
    end case;
end process;

--ANODE ROTATION LOGIC & PUSH CATHODE SIGNALS TO CATHODE OUTPUT
process(CLK_240Hz)
    variable AN_count : unsigned(1 downto 0) := "00"; --set up ANODE counter
begin

    if(rising_edge(CLK_240Hz)) then
        AN_count := AN_count + 1; --no AN_count reset is needed because 2-
bits covers exactly 4 anodes
    end if;

    case AN_count is --push final logic to the outputs AN and CATH
        when "00" => AN <= "1110"; CATH <= CATHODE1;
        when "01" => AN <= "1101"; CATH <= CATHODE2;
        when "10" => AN <= "1011"; CATH <= CATHODE3;
        when "11" => AN <= "0111"; CATH <= CATHODE4;
        when others =>
    end case;
end process;

end Behavioral;
```