Joseph Bofani, 12/10/2018, EET 223, Tachometer Project

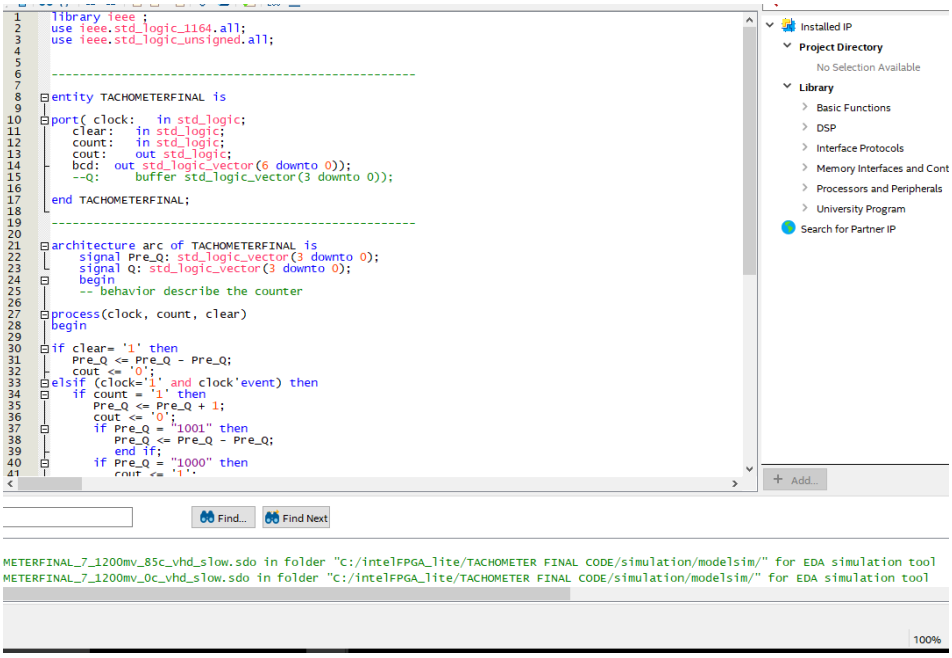
Objectives-

1) Simulate the CD4026 (BCD to 7-segment counter IC) in VHDL
2) Transfer the VHDL program into a block file (.qsf)
3) Simulate a bicycle tachometer in a Quartus schematic using the block file
4) Build the same bicycle tachometer on a breadboard, with a switch (a press of the button simulates a rotation of the wheel)

VHDL Design-

Narrative:

1) The flow of logic is as follows: Counter → Decoder → 7-segment display
2) Required inputs on counter: *clear*, *count*, & *clock*
3) Required outputs on decoder: *carry out*, two *7-segment displays*
4) Program a counter in VHDL that has all required inputs
5) Program a BCD to 7-segment decoder in the same VHDL program
6) The output of the counter feeds into the input of the decoder
7) The decoder uses case statements to assign 7-segment integers to the 7-segment display
8) Test the validity of the counter using a waveform
    a. The output should count from "0-9" and then reset to '0'
    b. Carry-out goes HIGH only when '9' is outputted HIGH from the

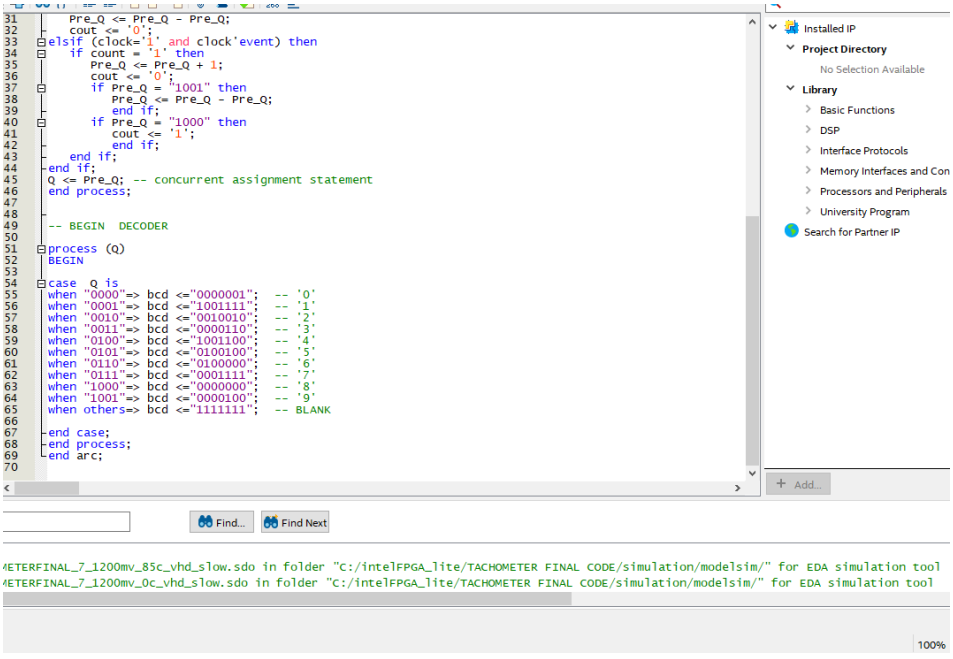


```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


-------------------------------------------------

entity TACHOMETERFINAL is

port( clock:   in std_logic;
    clear:   in std_logic;
    count:   in std_logic;
    cout:    out std_logic;
    bcd:  out std_logic_vector(6 downto 0));
    --Q:     buffer std_logic_vector(3 downto 0));

end TACHOMETERFINAL;

-------------------------------------------------

architecture arc of TACHOMETERFINAL is
    signal Pre_Q: std_logic_vector(3 downto 0);
    signal Q: std_logic_vector(3 downto 0);
    begin
    -- behavior describe the counter

process(clock, count, clear)
begin

if clear= '1' then
    Pre_Q <= Pre_Q - Pre_Q;
    cout <= '0';
elsif (clock='1' and clock'event) then
    if count = '1' then
        Pre_Q <= Pre_Q + 1;
        cout <= '0';
        if Pre_Q = "1001" then
            Pre_Q <= Pre_Q - Pre_Q;
            end if;
        if Pre_Q = "1000" then
```

Installed IP
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Cont
Processors and Peripherals
University Program
Search for Partner IP
Add...
Find...
Find Next

```
METERFINAL_7_1200mv_85c_vhd_slow.sdo in folder "C:/intelFPGA_lite/TACHOMETER FINAL CODE/simulation/modelsim/" for EDA simulation tool
METERFINAL_7_1200mv_0c_vhd_slow.sdo in folder "C:/intelFPGA_lite/TACHOMETER FINAL CODE/simulation/modelsim/" for EDA simulation tool
```

100%



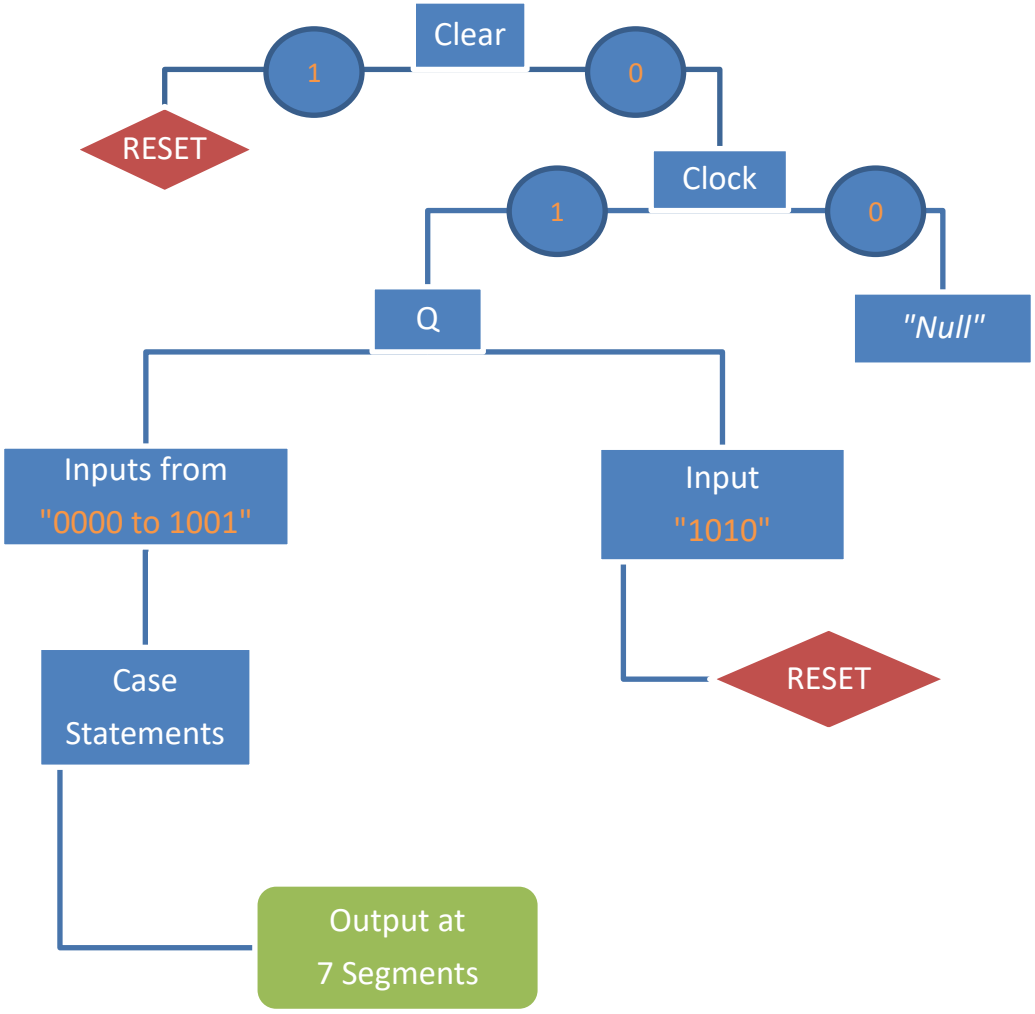
```
    Pre_Q <= Pre_Q - Pre_Q;
    cout <= '0';
elsif (clock='1' and clock'event) then
    if count = '1' then
        Pre_Q <= Pre_Q + 1;
        cout <= '0';
        if Pre_Q = "1001" then
            Pre_Q <= Pre_Q - Pre_Q;
            end if;
        if Pre_Q = "1000" then
            cout <= '1';
            end if;
    end if;
end if;
Q <= Pre_Q; -- concurrent assignment statement
end process;


-- BEGIN  DECODER

process (Q)
BEGIN

case  Q is
 when "0000"=> bcd <="0000001";   -- '0'
 when "0001"=> bcd <="1001111";   -- '1'
 when "0010"=> bcd <="0010010";   -- '2'
 when "0011"=> bcd <="0000110";   -- '3'
 when "0100"=> bcd <="1001100";   -- '4'
 when "0101"=> bcd <="0100100";   -- '5'
 when "0110"=> bcd <="0100000";   -- '6'
 when "0111"=> bcd <="0001111";   -- '7'
 when "1000"=> bcd <="0000000";   -- '8'
 when "1001"=> bcd <="0000100";   -- '9'
 when others=> bcd <="1111111";   -- BLANK

end case;
end process;
end arc;
```

Installed IP
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Con
Processors and Peripherals
University Program
Search for Partner IP
Add...
Find...
Find Next

```
METERFINAL_7_1200mv_85c_vhd_slow.sdo in folder "C:/intelFPGA_lite/TACHOMETER FINAL CODE/simulation/modelsim/" for EDA simulation tool
METERFINAL_7_1200mv_0c_vhd_slow.sdo in folder "C:/intelFPGA_lite/TACHOMETER FINAL CODE/simulation/modelsim/" for EDA simulation tool
```

100%

VHDL design of the bicycle tachometer. Clock, count, and clear (reset) are all inputs in the counter, which is dictated by the first process (VHDL line 27 – 46). The counter is a positive-edge triggered device with an Active-HIGH clear. The counter resets when integer '9' is reached (1001 in binary). The *cout* (carry-out) is activated when the counter reaches integer '8' so the *cout* will only be read at the subsequent positive edge (integer '9'). The decoder begins at the second process (VHDL lines 51 – 68). The decoder uses the counter's internal output, *Q*, as its input. Case statements are used to assign Active-LOW 7-segment display outputs (*bcd*) based on *Q*. If a number greater than 9 is counted, the 7-segment display will turn blank.

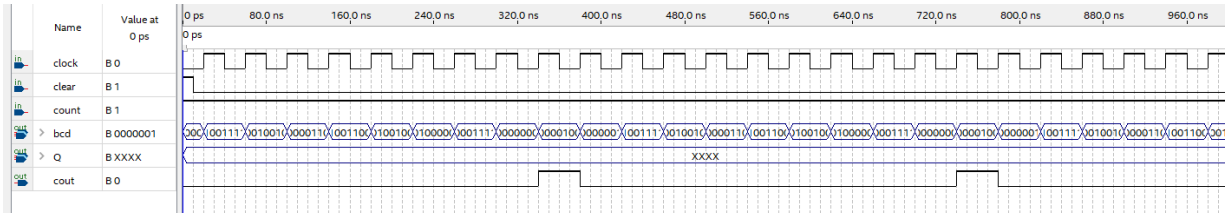


Logic hierarchy of the VHDL program. Orange text denotes a required input to continue down the current path. Orange ‘1’ equals HIGH input, and an orange ‘0’ equals LOW input.

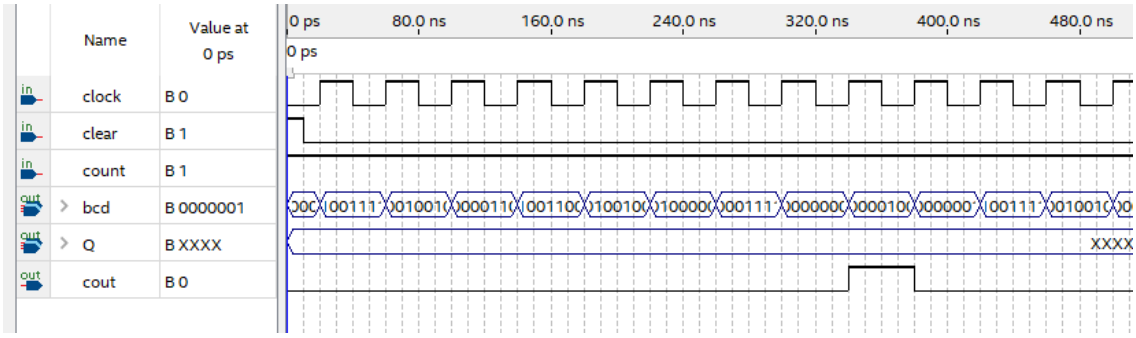
```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;


--------------------------------------------------


entity TACHOMETERFINAL is

port(       clock:      in std_logic;
            clear:      in std_logic;
            count:      in std_logic;
            cout:               out std_logic;
            bcd:        out std_logic_vector(6 downto 0));
            --Q:                buffer std_logic_vector(3 downto 0));

end TACHOMETERFINAL;

--------------------------------------------------

architecture arc of TACHOMETERFINAL is
   signal Pre_Q: std_logic_vector(3 downto 0);
             signal Q: std_logic_vector(3 downto 0);
             begin
   -- behavior describe the counter

process(clock, count, clear)
begin
```

```
if clear= '1' then
        Pre_Q <= Pre_Q - Pre_Q;
        cout <= '0';
elsif (clock='1' and clock'event) then
        if count = '1' then
                Pre_Q <= Pre_Q + 1;
                cout <= '0';
                if Pre_Q = "1001" then
                        Pre_Q <= Pre_Q - Pre_Q;
                        end if;
                if Pre_Q = "1000" then
                        cout <= '1';
                        end if;
        end if;
end if;
Q <= Pre_Q; -- concurrent assignment statement
end process;



-- BEGIN  DECODER


process (Q)
BEGIN


case  Q is
when "0000"=> bcd <="0000001";  -- '0'
when "0001"=> bcd <="1001111";  -- '1'
when "0010"=> bcd <="0010010";  -- '2'
when "0011"=> bcd <="0000110";  -- '3'
when "0100"=> bcd <="1001100";  -- '4'
```

```
when "0101"=> bcd <="0100100";  -- '5'

when "0110"=> bcd <="0100000";  -- '6'

when "0111"=> bcd <="0001111";  -- '7'

when "1000"=> bcd <="0000000";  -- '8'

when "1001"=> bcd <="0000100";  -- '9'

when others=> bcd <="1111111";  -- BLANK


end case;

end process;

end arc;
```



Waveform of the VHDL program. The count resets after the integer '9' is reached. The *clear* is Active-HIGH at the start of the waveform. It is also observed that *cout* is only HIGH when '9' is counted. The *Q* is the internal count of the design, and isn't accounted for in the results of this waveform (only the *bcd* and *cout* outputs matter).

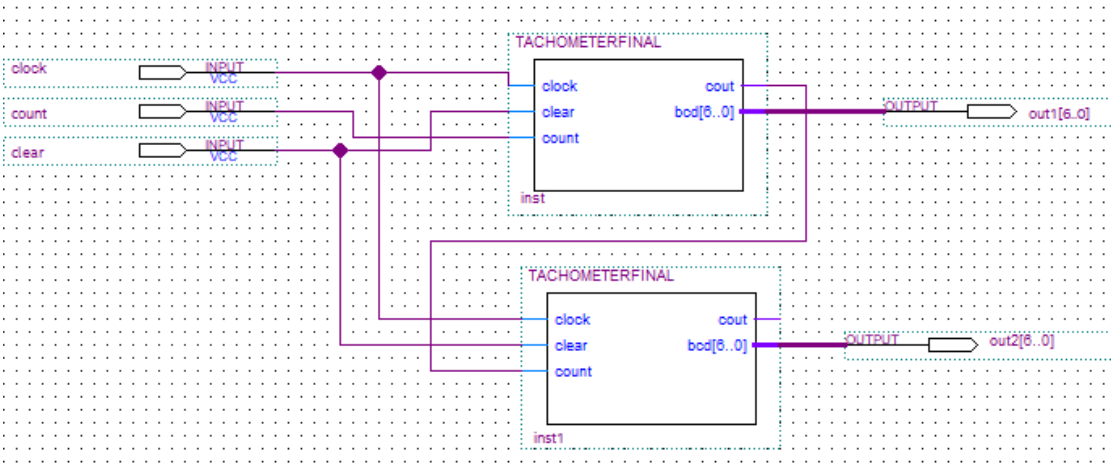


Zoomed in version of the VHDL program's waveform.

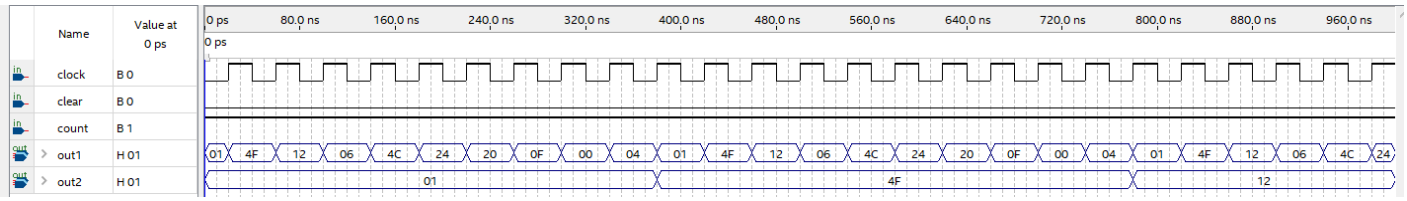
Quartus Schematic-

Narrative:

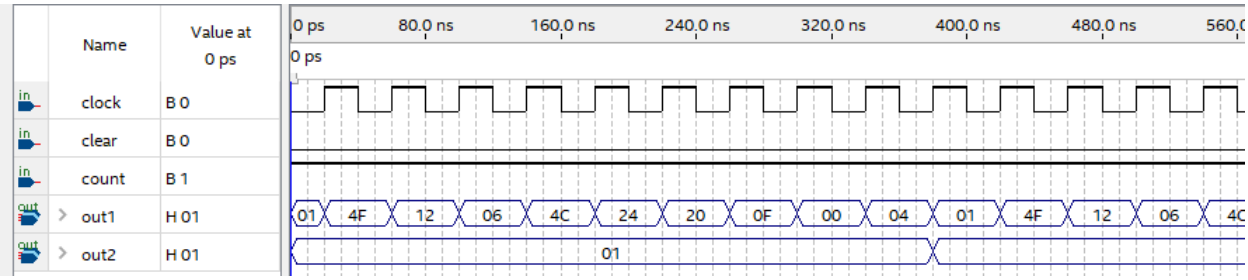
1) Convert VHDL into a .qsf block file
2) Build a bicycle tachometer by combining 2 newly created block files together in a schematic
3) Test the validity of the tachometer by using a waveform
    a. The two outputs of the 7-segment displays should count from "0 – 99"



Quartus schematic file that connects two custom-created chips that emulate the CD4026 ICs. Notice the *cout* of the first IC is the *count* of the second IC. The result is a tachometer that is hooked into two Active-LOW 7-segment displays, represented as *out1* and *out2*.



Waveform for the schematic tachometer. *out1* represents the one's place, and *out2* represents the ten's place on the tachometer. The waveform can be observed to count from 0 – 35. *out1* takes 1 clock pulse to count up, whereas *out2* takes 10 clock pulses to count up.

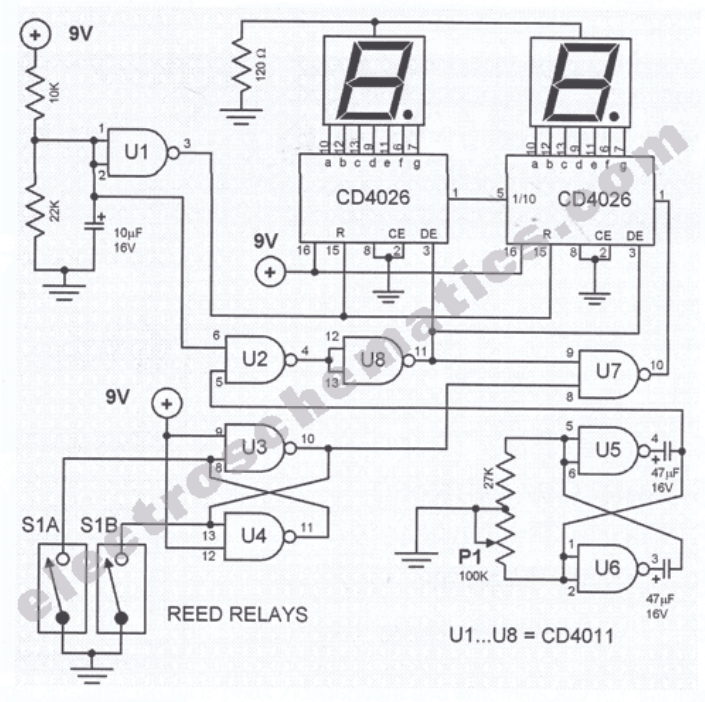


Zoomed in version of the tachometer's (schematic) waveform.

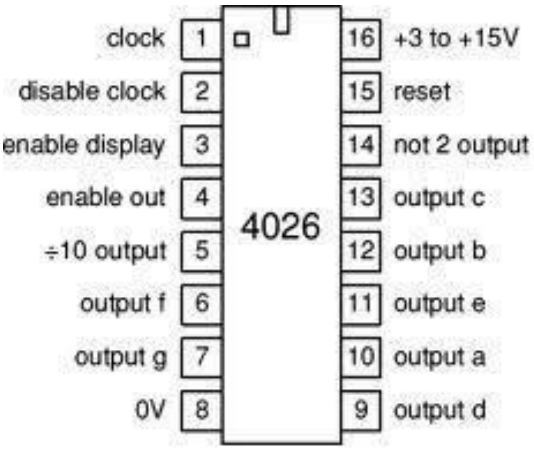
Breadboard Circuit-

Narrative:

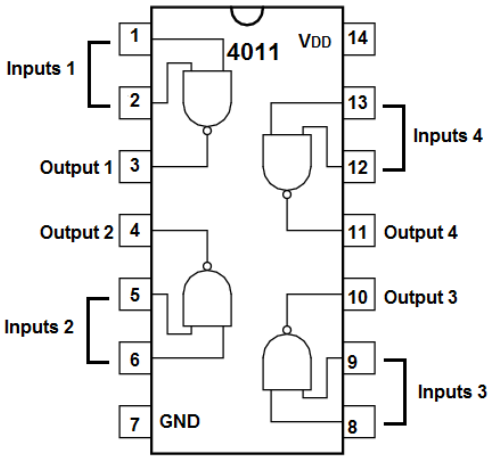
1) Build the tachometer circuit according to the below schematic
    a. Test IC's and 7-segments to make sure they are functioning properly
    b. 7-segments must be common cathode
    c. Two CD4011 IC's are needed for the NAND gates
2) Attach power supply (9V and 500mA) and Ground to circuit
3) Test validity of tachometer
    a. Circuit should display '0' at startup and consecutively count from "0-99" with each switch action
    b. Circuit resets to '0' after "99" is displayed



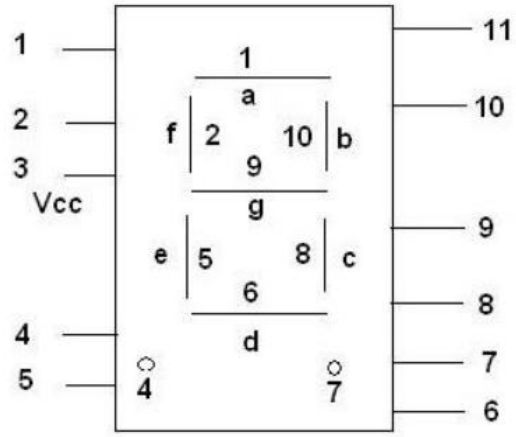
Tachometer circuit schematic

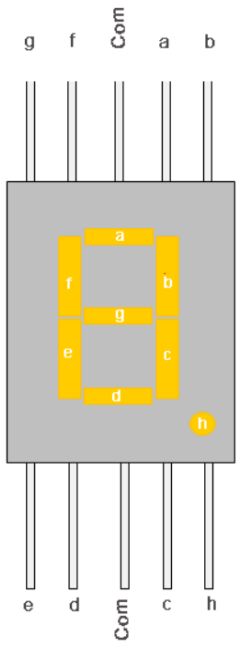


CD4026 7-segment decoder pinout

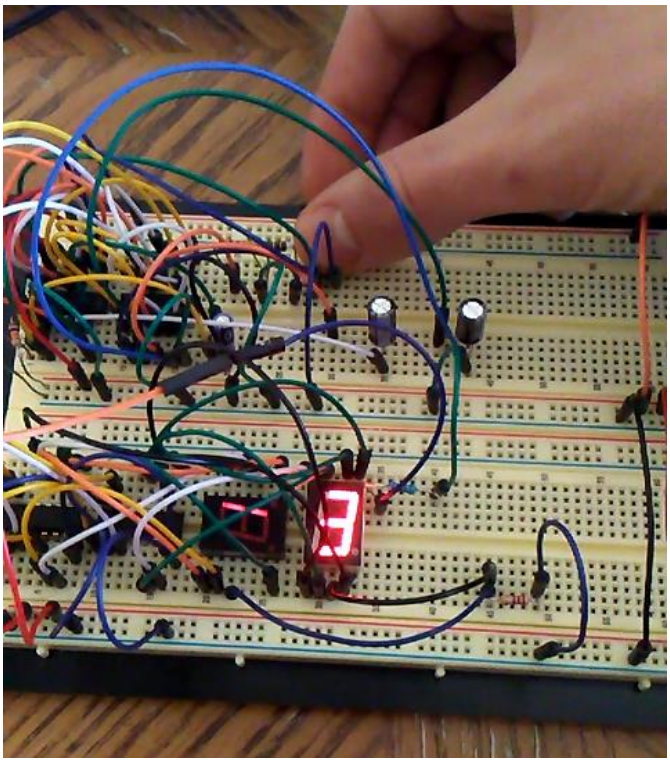


CD4011 NAND gate pinout





Pinouts for two common types of 7-segments, which are both used in my circuit. Pinouts will vary between different types of 7-segments. For this project, the 7-segments must be common-cathode. Ground must be connected to where the above pinouts read "V_{CC}" or "Com."



Fully functional tachometer circuit that's displaying "31." The '3' is in the ten's place, and the '1' is in the one's place. The circuit boots-up with a '0,' counts to "99," and then resets to '0'. The CD4011's are grouped adjacent on the top left, and the CD4026's are grouped adjacent to the left of the 7-segments.

Conclusion-

Coding the VHDL was best done in two programs; building the counter and decoder separately. Once they were both functioning, they could be programmed together with a common in and out line, which was *Q* in my code. The .bsf file and its respective VHDL file must be in the same computer folder in order for the block file to compile in a Quartus schematic. Before creating the physical circuit, it's crucial to test for broken parts. Pins on IC's can be broken, or components could be shorted from previous use. Two 7-segments I originally picked up had LED's that weren't working, so I had to swap out components. Also make sure that wires and pins do not overlap, and create an error in the logic. The logic of both the virtual and physical tachometer replicates each other. Both types of circuits have input flow through a counter, which passes data to a 7-segment decoder. From the decoders, the logic goes into two common-cathode (Active-LOW) 7-segment displays. Both types of circuits also reset to the startup value of '0' after the max number "99" is counted.