Solutions to Assignment 2

Question 1

A possible VHDL description of a 74’161 counter is:

```
-- ELEC 464 Assignment 2
-- 74LS161: 4-bit counter with async reset
--   and parallel load
-- Ed Casas, September 17 1996

entity LS161 is
   port ( cep, cet, cp, pe, mr : in bit ;
          d : in bit_vector (3 downto 0) ;
          q : out bit_vector (3 downto 0) ;
          tc :  out bit ) ;
end ;


architecture rtl of LS161 is
   signal nextc : bit_vector (3 downto 0) ;
   signal count : bit_vector (3 downto 0) ;
begin

   --  combinational logic to compute next count
   --  from current count and control inputs
   process(d,pe,cep,cet,count)
   begin

       -- parallel load if pe low
       if pe = '0' then

          nextc <= d ;

       -- increment if both count enables high
       elsif cep = '1' and cet = '1' then

          case count is
             when "0000" => nextc <= "0001" ;
             when "0001" => nextc <= "0010" ;
             when "0010" => nextc <= "0011" ;
             when "0011" => nextc <= "0100" ;
             when "0100" => nextc <= "0101" ;
             when "0101" => nextc <= "0110" ;
             when "0110" => nextc <= "0111" ;
             when "0111" => nextc <= "1000" ;
             when "1000" => nextc <= "1001" ;
             when "1001" => nextc <= "1010" ;
             when "1010" => nextc <= "1011" ;
             when "1011" => nextc <= "1100" ;
             when "1100" => nextc <= "1101" ;
             when "1101" => nextc <= "1110" ;
             when "1110" => nextc <= "1111" ;
             when "1111" => nextc <= "0000" ;
          end case ;

       -- otherwise retain count
       else

          nextc <= count ;

       end if ;

   end process ;


   -- sequential logic
   process(cp,mr,nextc)
   begin
      -- async reset if mr low
      if mr = '0' then
         count <= "0000" ;
      else
         -- clock loads next count
         if cp'event and cp = '1' then
            count <= nextc ;
         end if ;
      end if ;
   end process ;


   -- combinational logic for outputs
   process(count,cet)
   begin
      q <= count ;

      if cet = '1' and count = "1111" then
         tc <= '1' ;
      else
         tc <= '0' ;
      end if ;
   end process ;

end ;
```

The result of running the test-bench is:

```
# run
 Test vector 0: Q=0000 TC=0. Output Q=0000 TC=0
 Test vector 1: Q=0001 TC=0. Output Q=0001 TC=0
 Test vector 2: Q=0010 TC=0. Output Q=0010 TC=0
 Test vector 3: Q=0010 TC=0. Output Q=0010 TC=0
 Test vector 4: Q=0010 TC=0. Output Q=0010 TC=0
 Test vector 5: Q=0010 TC=0. Output Q=0010 TC=0
 Test vector 6: Q=1110 TC=0. Output Q=1110 TC=0
 Test vector 7: Q=1111 TC=1. Output Q=1111 TC=1
(vhdlsim): Simulation complete, time is 16 FS.
#
```

The result of synthesizing this description is shown in Figure **??**.

Question 2

An 8088 assembly-language program to print the upper-case version of a string stored in memory might be as follows:

```
; ELEC 464 Assignment 2 Question 2
; Prints the upper-case version of a string
;   stored in memory.
; Ed Casas, September 27 1996

code segment public  ; set-up for .COM file
     assume  cs:code,ds:code
     org  100h

; the program

start:
     mov  bx,offset msg   ; bx points to string
loop:
     mov  al,[bx] ; load a character into al
     cmp  al,0    ; check for terminating zero
     jz   done    ; quit if so
     cmp  al,'a'  ; compare to lower limit
     jl   skip    ; don't convet if <'a'
     cmp  al,'z'  ; compare to upper limit
     jg   skip    ; don't convert if >'z'
     sub  al,20H  ; else convert to upper case
skip:
     call printc
     inc  bx      ; point to next character
     jmp  loop    ; and loop back

done:
     int  20h     ; return to DOS

; subroutine to print the byte in al

printc:
     push ax     ; save ax and dx on stack
     push dx
     mov  dl,al  ; DOS print-character function
     mov  ah,02H
     int  21H
     pop  dx     ; restore registers ax and dx
     pop  ax
     ret

; the string to convert to print in upper case

msg  db  'AZaz Ed Casas 12345678',0

code ends
     end start
```

And the output is:

```
AZAZ ED CASAS 12345678
```

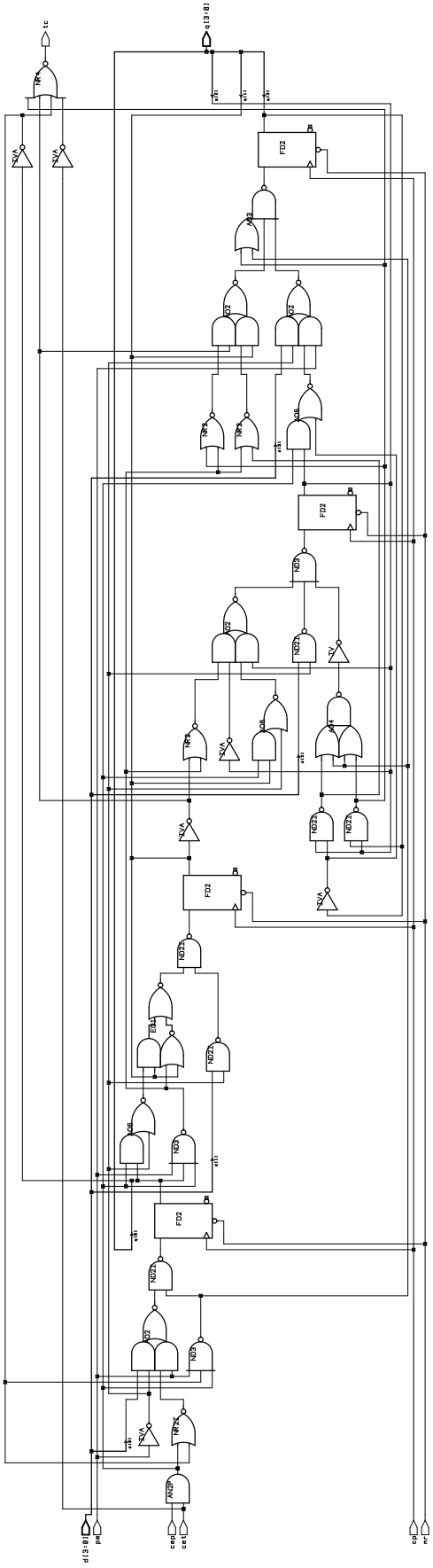

Figure 1: Schematic of 74'161.