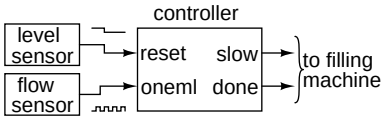
Solutions to Final Exam

Question 1

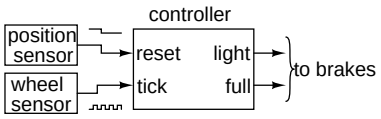
There were three versions of this question. Each design was a synchronous counter with reset. The values of the two outputs are determined by the counter and input values. For versions 1 and 2 the reset is `reset` and the clock is `oneml` or `tick`. For version 3 the timer is reset when the `drogue` output is not asserted and the clock is `clock`. Sample entities and architectures for the three versions are given below.

For version 1 the counter is 9 bits wide and the `slow` and `done` outputs are set high whenever the count is more than 400 and 502 respectively (since the counter is reset for the first two ml):



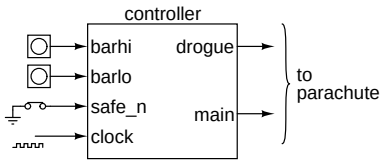

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity controller1 is

  port (
    reset, oneml : in  std_logic;
    slow, done   : out std_logic);

end controller1;

architecture rtl of controller1 is

  signal level, next_level : unsigned (8 downto 0);

begin

  next_level <=
    to_unsigned(0,level'length) when reset = '1' else
    level + 1 ;

  process (oneml)
  begin
    if oneml'event and oneml = '1' then
      level <= next_level ;
    end if;
  end process;

  slow <= '1' when level >= 400 else '0' ;
  done <= '1' when level >= 500 else '0' ;

end rtl;
```

For version 2 the counter is 8 bits wide and the `light` and `full` outputs are set high whenever the count is more than 200 and 250 respectively (since the counter is reset for the first two meters):




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

entity controller2 is

  port (
    reset, tick : in  std_logic;
    light, full : out std_logic);

end controller2;

architecture rtl of controller2 is

  signal dist, next_dist : unsigned (7 downto 0);

begin

  next_dist <=
    to_unsigned(0,dist'length) when reset = '1' else
    dist + 1 ;

  process (tick)
  begin
    if tick'event and tick = '1' then
      dist <= next_dist ;
    end if;
  end process;

  light <= '1' when dist >= 200 else '0' ;
  full  <= '1' when dist >= 250 else '0' ;

end rtl;
```

For version 3 the counter is 3 bits wide. The `drogue` output is asserted when `safe_n` is high and either `barhi` or `barlo` are high. `main` is asserted if `drogue` is asserted or the count is more than 5. An internal version of the `drogue` output is used to avoid reading an output signal.

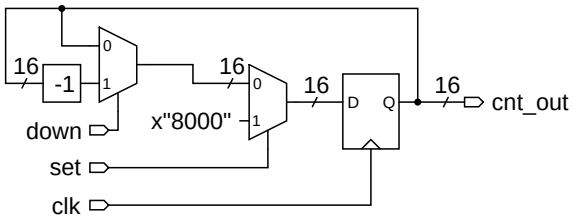



```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity controller3 is

  port (
    barhi, barlo, safe_n, clock : in  std_logic;
    drogue, main : out std_logic);

end controller3;

architecture rtl of controller3 is

  signal delay, next_delay : unsigned (2 downto 0);
  signal drogue_i : std_logic;

begin

  drogue_i <=  safe_n and ( barhi or barlo ) ;

  next_delay <=
   to_unsigned(0,delay'length) when drogue_i = '0' else
    delay + 1 ;

  process (clock)
  begin
    if clock'event and clock = '1' then
      delay <= next_delay ;
    end if;
  end process;

  main  <=
    '1' when safe_n = '1' and
        ( drogue_i = '1' or ( delay > 5 ) ) else
    '0' ;

  drogue <= drogue_i ;

end rtl;
```

Question 2

The three solutions are very similar. Each schematic consists of a synchronous counter that is set/reset/incremented by two control signals.

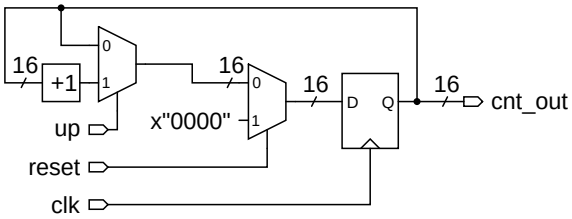
The VHDL code corresponding to:



is:

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

entity question2a is

  port (
    down, set, clk : in  std_logic;
    cnt_out : out unsigned (15 downto 0) );
end question2a ;

architecture rtl of question2a is

  signal cnt, next_cnt : unsigned (15 downto 0);

begin

  next_cnt <=
    x"8000" when set = '1' else
    cnt - 1 when down = '1' else
    cnt ;

  process (clk)
  begin
    if clk'event and clk = '1' then
      cnt <= next_cnt ;
    end if;
  end process;

  cnt_out  <= cnt ;

end rtl;
```

The VHDL code corresponding to:

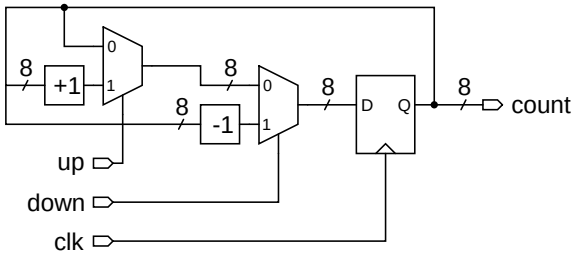


is:

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

entity question2b is

  port (
    down, reset, clk : in  std_logic;
    cnt_out : out unsigned (15 downto 0) );
end question2b ;

architecture rtl of question2b is

  signal cnt, next_cnt : unsigned (15 downto 0);

begin
```

```
  next_cnt <=
    x"0000" when reset = '1' else
    cnt + 1 when down = '1' else
    cnt ;

  process (clk)
  begin
    if clk'event and clk = '1' then
      cnt <= next_cnt ;
    end if;
  end process;

  cnt_out  <= cnt ;

end rtl;
```

The VHDL code corresponding to:



is:

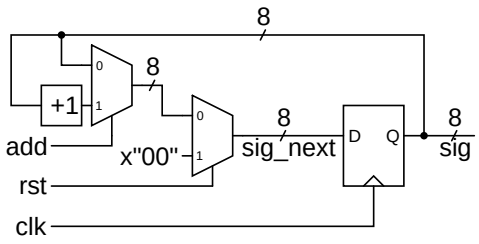
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity question2c is

  port (
    up, down, clk : in  std_logic;
    count : out unsigned (7 downto 0) );
end question2c ;

architecture rtl of question2c is

  signal cnt, next_cnt : unsigned (7 downto 0);

begin

  next_cnt <=
    cnt - 1 when down = '1' else
    cnt + 1 when up = '1' else
    cnt ;

  process (clk)
  begin
    if clk'event and clk = '1' then
      cnt <= next_cnt ;
    end if;
  end process;

  count  <= cnt ;

end rtl;
```

Question 3

A schematic for the following VHDL code:

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

architecture rtl1 of exam is
   signal sig, sig_next : unsigned (7 downto 0) ;
   signal rst, add, clk : std_logic ;
begin

  sig_next <=
    x"00" when rst = '1' else
    sig+1 when add = '1' else
    sig ;

  process(clk)
  begin
    if clk'event and clk = '1' then
       sig <= sig_next ;
    end if ;
  end process ;

end rtl1 ;
```

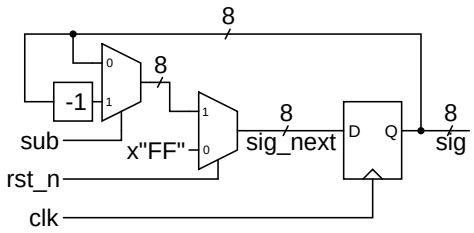
might be:



A schematic for the following VHDL code:

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

architecture rtl2 of exam is
   signal sig, sig_next : unsigned (7 downto 0) ;
   signal rst_n, sub, clk : std_logic ;
begin

  sig_next <=
    x"FF" when rst_n = '0' else
    sig-1 when sub = '1' else
    sig ;

  process(clk)
  begin
    if clk'event and clk = '1' then
       sig <= sig_next ;
    end if ;
  end process ;

end rtl2 ;
```

might be:



A schematic for the following VHDL code:

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

architecture behav of exam is
   signal wave, waven : unsigned (14 downto 0) ;
   signal set, up, tick : std_logic ;
begin

  waven <=
    to_unsigned(1234,15) when set = '1' else
    wave+2 when up = '1' else
    wave ;

  process(tick)
  begin
    if tick'event and tick = '1' then
       wave <= waven ;
    end if ;
  end process ;

end behav;
```

might be: