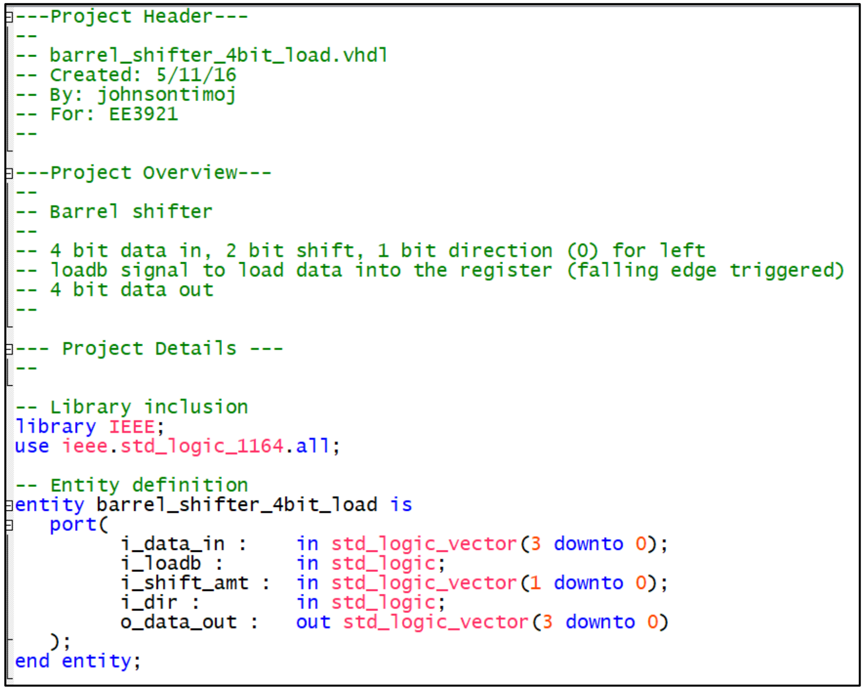
EE 3921

Dr. Johnson

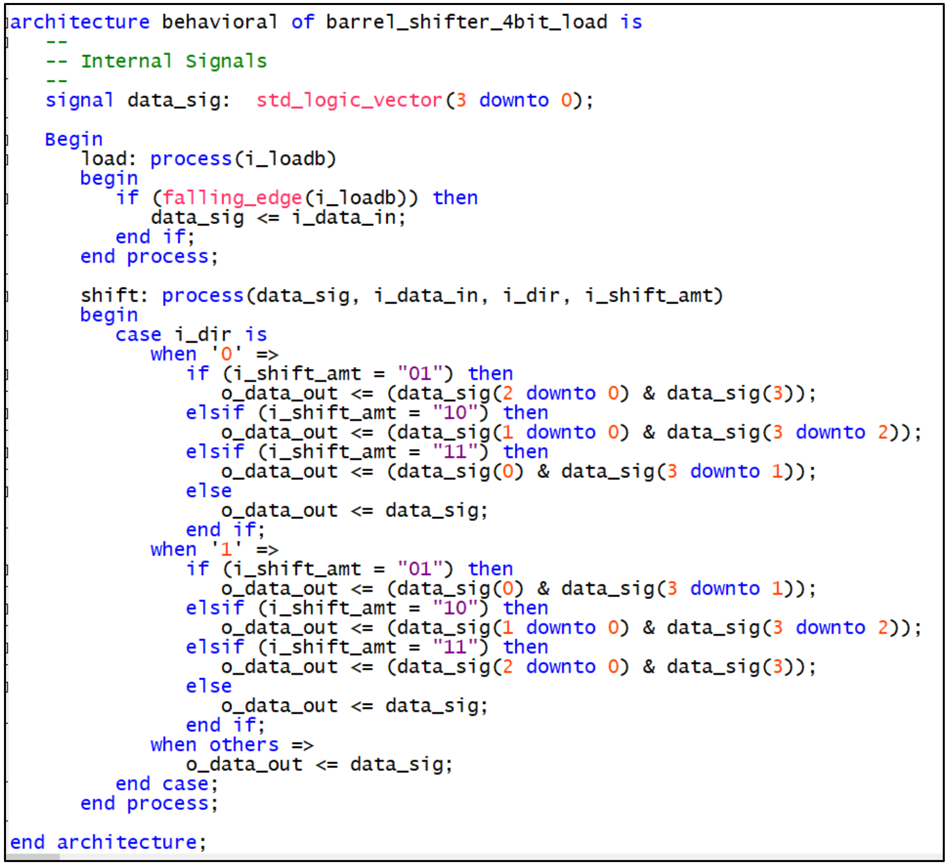
Homework 2

1) Create code for an asynchronous 4 bit barrel shifter with a falling edge load_bar signal to initialize the shifter. Inputs: 4 bit data word to be loaded, loadb signal to load the input, 2 bit shift value, 1 bit direction value. Outputs: 4 bit shifted output.

40pts



```vhdl
---Project Header---
--
-- barrel_shifter_4bit_load.vhdl
-- Created: 5/11/16
-- By: johnsontimoj
-- For: EE3921
--

---Project Overview---
--
-- Barrel shifter
--
-- 4 bit data in, 2 bit shift, 1 bit direction (0) for left
-- loadb signal to load data into the register (falling edge triggered)
-- 4 bit data out
--

--- Project Details ---
--

-- Library inclusion
library IEEE;
use ieee.std_logic_1164.all;

-- Entity definition
entity barrel_shifter_4bit_load is
   port(
         i_data_in :    in std_logic_vector(3 downto 0);
         i_loadb :      in std_logic;
         i_shift_amt :  in std_logic_vector(1 downto 0);
         i_dir :        in std_logic;
         o_data_out :   out std_logic_vector(3 downto 0)
   );
end entity;
```



```vhdl
architecture behavioral of barrel_shifter_4bit_load is
   --
   -- Internal Signals
   --
   signal data_sig:  std_logic_vector(3 downto 0);

   Begin
      load: process(i_loadb)
      begin
         if (falling_edge(i_loadb)) then
            data_sig <= i_data_in;
         end if;
      end process;

      shift: process(data_sig, i_data_in, i_dir, i_shift_amt)
      begin
         case i_dir is
            when '0' =>
               if (i_shift_amt = "01") then
                  o_data_out <= (data_sig(2 downto 0) & data_sig(3));
               elsif (i_shift_amt = "10") then
                  o_data_out <= (data_sig(1 downto 0) & data_sig(3 downto 2));
               elsif (i_shift_amt = "11") then
                  o_data_out <= (data_sig(0) & data_sig(3 downto 1));
               else
                  o_data_out <= data_sig;
               end if;
            when '1' =>
               if (i_shift_amt = "01") then
                  o_data_out <= (data_sig(0) & data_sig(3 downto 1));
               elsif (i_shift_amt = "10") then
                  o_data_out <= (data_sig(1 downto 0) & data_sig(3 downto 2));
               elsif (i_shift_amt = "11") then
                  o_data_out <= (data_sig(2 downto 0) & data_sig(3));
               else
                  o_data_out <= data_sig;
               end if;
            when others =>
               o_data_out <= data_sig;
         end case;
      end process;

end architecture;
```

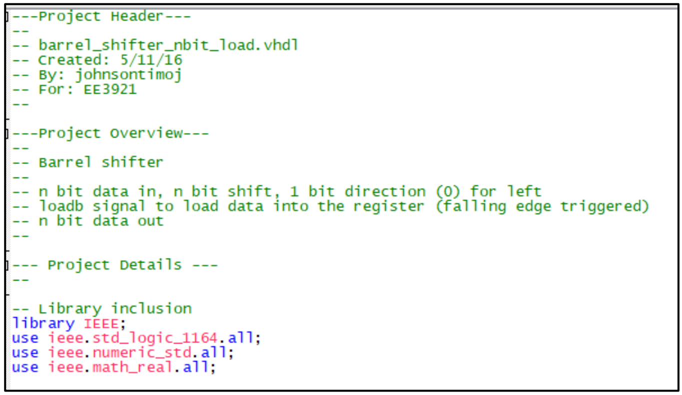
There are lot of more efficient ways to do this – but sometimes extra coding helps clarify the problem and allows others to better understand what is happening

e.g. note rt sh 1 and lt sh 3 are the same
rt sh 2 and lt sh 2 are the same

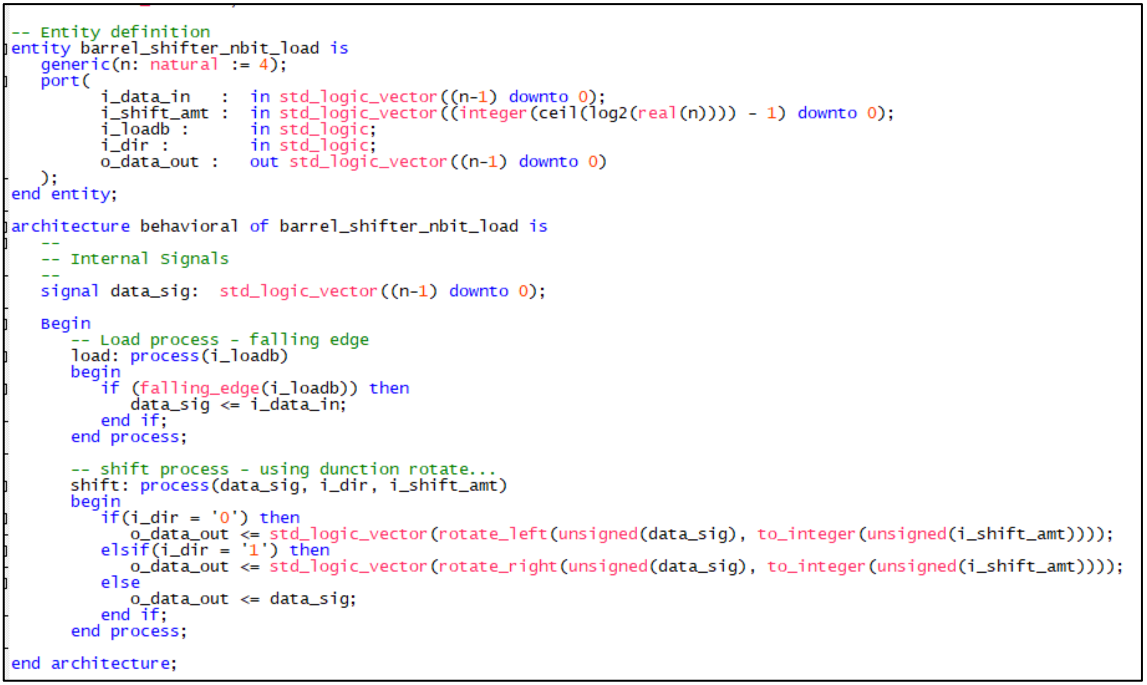
There is a left_shift and rt_shift function
in numeric standard (allows for an Nbit shifter)

) Create code for an asynchronous 4 bit barrel shifter with a falling edge load_bar signal to initialize the shifter. Inputs: 4 bit data word to be loaded, loadb signal to load the input, 2 bit shift value, 1 bit direction value. Outputs: 4 bit shifted output.

40pts



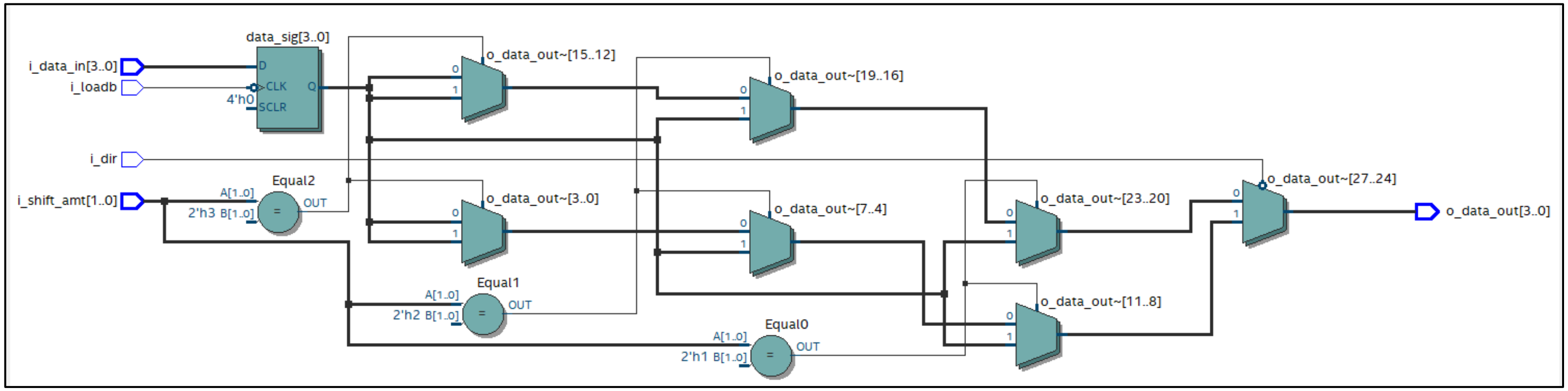
```
---Project Header---
--
-- barrel_shifter_nbit_load.vhdl
-- Created: 5/11/16
-- By: johnsontimoj
-- For: EE3921
--

---Project Overview---
--
-- Barrel shifter
--
-- n bit data in, n bit shift, 1 bit direction (0) for left
-- loadb signal to load data into the register (falling edge triggered)
-- n bit data out
--

--- Project Details ---
--

-- Library inclusion
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;
```



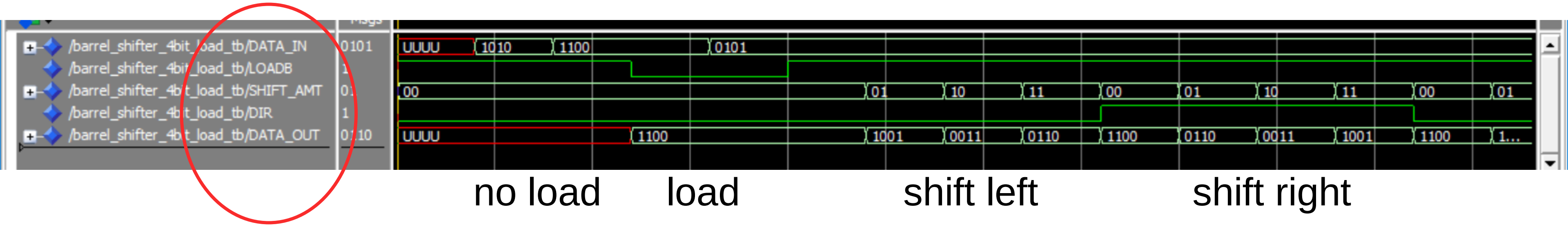
```
-- Entity definition
entity barrel_shifter_nbit_load is
   generic(n: natural := 4);
   port(
         i_data_in    :  in std_logic_vector((n-1) downto 0);
         i_shift_amt :  in std_logic_vector((integer(ceil(log2(real(n)))) - 1) downto 0);
         i_loadb :       in std_logic;
         i_dir :         in std_logic;
         o_data_out :    out std_logic_vector((n-1) downto 0)
   );
end entity;

architecture behavioral of barrel_shifter_nbit_load is
   --
   -- Internal Signals
   --
   signal data_sig:  std_logic_vector((n-1) downto 0);

   Begin
      -- Load process - falling edge
      load: process(i_loadb)
      begin
         if (falling_edge(i_loadb)) then
            data_sig <= i_data_in;
         end if;
      end process;

      -- shift process - using dunction rotate...
      shift: process(data_sig, i_dir, i_shift_amt)
      begin
         if(i_dir = '0') then
            o_data_out <= std_logic_vector(rotate_left(unsigned(data_sig), to_integer(unsigned(i_shift_amt))));
         elsif(i_dir = '1') then
            o_data_out <= std_logic_vector(rotate_right(unsigned(data_sig), to_integer(unsigned(i_shift_amt))));
         else
            o_data_out <= data_sig;
         end if;
      end process;

end architecture;
```

2 Synthesize your barrel shifter and provide the RTL

20pts



3) Simulate your barrel shifter, provide the testbench and simulation results 40pts



```
/barrel_shifter_4bit_load_tb/DATA_IN     0101
/barrel_shifter_4bit_load_tb/LOADB       1
/barrel_shifter_4bit_load_tb/SHIFT_AMT   01
/barrel_shifter_4bit_load_tb/DIR         1
/barrel_shifter_4bit_load_tb/DATA_OUT    0110
```

no load load shift left shift right