1. TRUE or FALSE and SHORT ANSWERS:

a) Label each statement with "T" or "F". Be very clear. Answers that look like a combination of "T" and "F" will be marked wrong. [1 mark each]

Statement	T/F
An 8-bit multiplier can be described in a single synthesizable VHDL process.	T
In Quartus II (which we have been using in the lab), you can use a .mif file to indicate the values that are to be stored in a memory.	T
A 4-input lookup table can implement $(2^4)^4 = 65536$ different functions.	
In VHDL, a "generate" statement can only be used within a process.	F

b) In class, we saw five types of adders: ripple adder, carry lookahead adder, carry-skip adder, carry-save adder, carry-select adder

i) Suppose we wish to add two 32-bit numbers. Of these adders, which is the fastest? [1 mark]

carry lookahead (also accepted carry skip, since carry-lookahead is not realistic for a 32 bit adder)

ii) Suppose we wish to add two 32-bit numbers. Of these adders, which will use the least hardware? [1 mark]

ripple

iii) Which of these adders would be most appropriate if we wish to add eight 32-bit numbers together? [1 mark]

carry save

2. In this question you are to describe an 8-bit shift register using synthesizable VHDL. Each rising clock edge, the shift register will do one of four things, depending on the values of two control signals *c1* and *c2*:

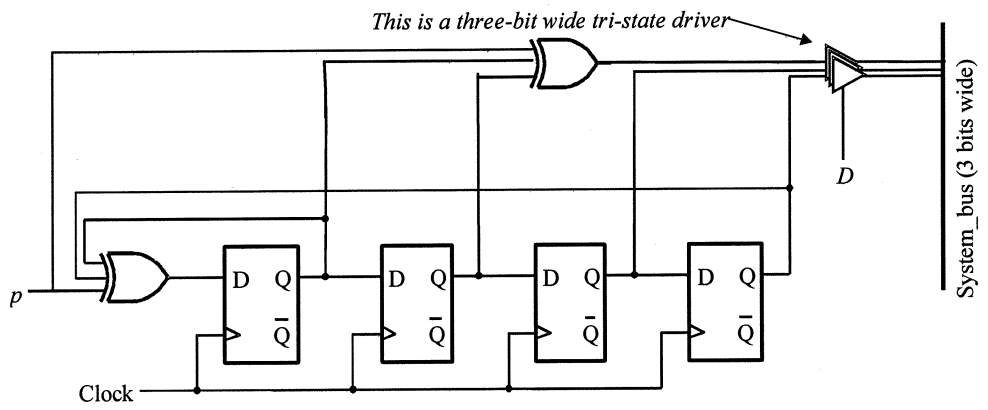
c1	*c2*	function
0	0	shift left by 1 bit
0	1	shift right by 1 bit
1	0	parallel load
1	1	reset to 0

At any other time (other than a rising clock edge) the shift register maintains its value. You can assume that when the register is shifted to the right, a '0' is shifted into the most-significant bit, and when the register is shifted to the left, a '0' is shifted into the least-significant bit.

Write a synthesizable VHDL description of this block, using as few processes as possible. Marks will be deducted for solutions which are overly long or complex. [6 marks]

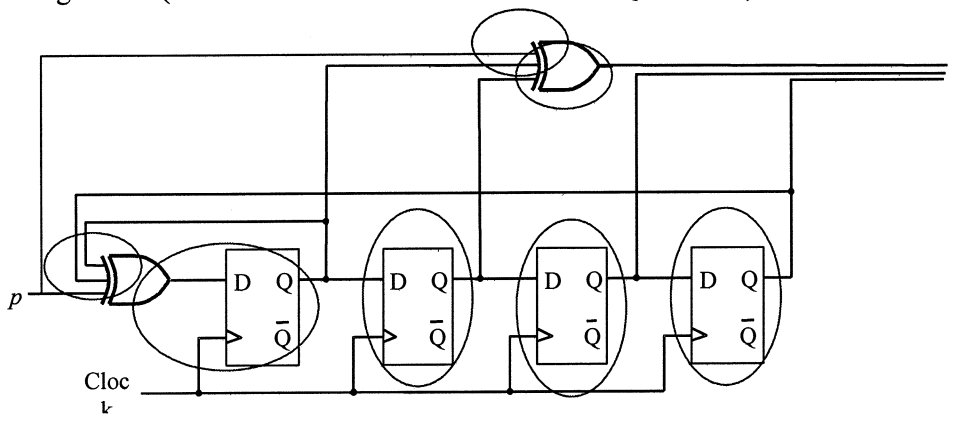
```
entity SHIFTBLOCK is
        port ( c1, c2, clk : in std_logic;
               parallel_in : in std_logic_vector(7 downto 0);
               Q: out std_logic_vector(7 downto 0));
end SHIFTBLOCK;

architecture behavioural of SHIFTBLOCK is

begin
   process (clk)
   variable int_value : std_logic_vector (7 downto 0);
   begin
       if (clk='1' and clk'event) then
              if (c1 = '0' and c2 = '0') then
                    int_value := int_value(6 downto 0) & '0';
              elsif (c1 = '0' and c2 = '1') then
                    int_value := '0' & int_value(7 downto 1);
              elsif (c1 = '1' and c2 = '0') then
                    int_value := parallel_in;
              else
                    int_value := "00000000";
              end if;

         Q <= int_value;
       end if;
     end process;
end behavioural;
```

3. Consider the following circuit. **Write a synthesizable VHDL description** of this block, using as few processes as possible. Marks will be deducted for solutions which are overly long or complex. [8 marks]

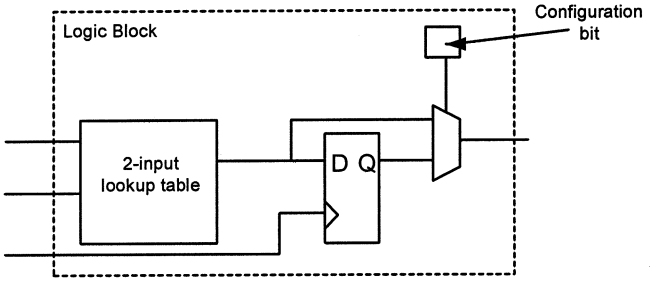



```
entity SIGBLOCK is
         port ( p, clock, D : in std_logic;
                system_bus : out std_logic_vector(2 downto 0));
      end SIGBLOCK;

architecture behavioural of SIGBLOCK is


signal temp : std_logic_vector(3 downto 0);
begin
   process (clock)
   variable int_value : std_logic_vector (3 downto 0);
   begin
       if (clock='1' and clock'event) then
              int_value := (int_value(3) xor int_value(0) xor p) &
                                        int_value(3 downto 1);
        end if;
        temp <= int_value;
     end process;

     process(p, temp, D)
     begin
         if (D = '0') then
             system_bus <= "ZZZ";
         else
             system_bus <= (p xor temp(3) xor temp(2)) & temp(1 downto 0);
         end if;
     end process;
end behavioural;
```

4. Consider the following circuit (which is almost the same as that from Question 3).



Suppose this circuit is to be implemented on an FPGA in which each logic block is as follows:



a) How many logic blocks are required? Assume we want the smallest number of logic blocks possible. Show your work (possibly by drawing circles on the above circuit diagram). [3 marks]

From above, 7 logic blocks required

b) If the delay of each lookup table is D, write an expression for the maximum clock frequency of the circuit. Connection delays are negligible. Assume input ***p*** is driven by a flip-flop, and the output lines drive flip-flops. Show your work. [2 marks]

1/(2D)

5. Consider the following circuit, which is common in communications-related applications to filter out high frequency noise. The circuit receives a series of data elements, one per clock cycle. The circuit produces a series of output data elements, one per clock cycle. Each output data element is the *average* of the most-recently received four input data elements. In other words:

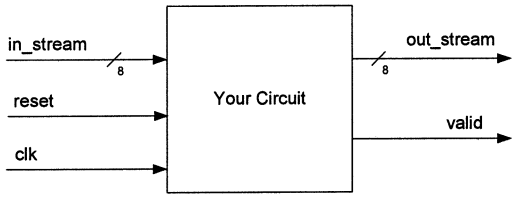
$$out_i = (in_i + in_{i-1} + in_{i-2} + in_{i-3}) / 4$$

where:

out_i is the output data element produced during cycle i
in_i is the input data element received during cycle i
in_{i-1} is the input data element received during cycle i-1
in_{i-2} is the input data element received during cycle i-2
in_{i-3} is the input data element received during cycle i-3

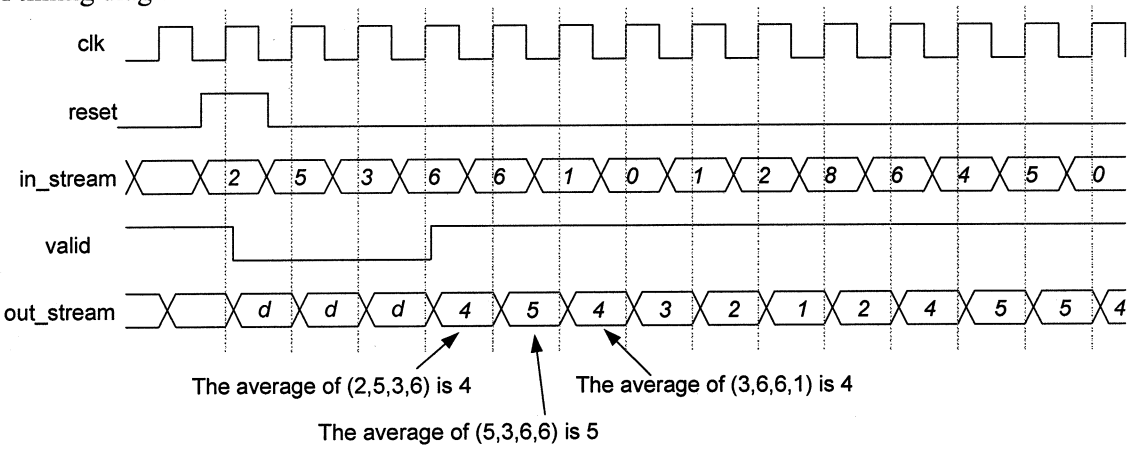
If the result is not an integer, it is rounded down (so if the average is 3.6, it is rounded down to 3).

A top-level diagram of your circuit is as follows:



Each cycle one 8-bit input data element appears on *in_stream*, and each cycle, one 8-bit result is produced on *out_stream*. There is a synchronous reset that indicates the start of the data stream. The *valid* output is low during the first three cycles after the reset, and goes high after the first three cycles (the output is not valid for the first three cycles, because there are fewer than four data elements to average).

A timing diagram is as follows:



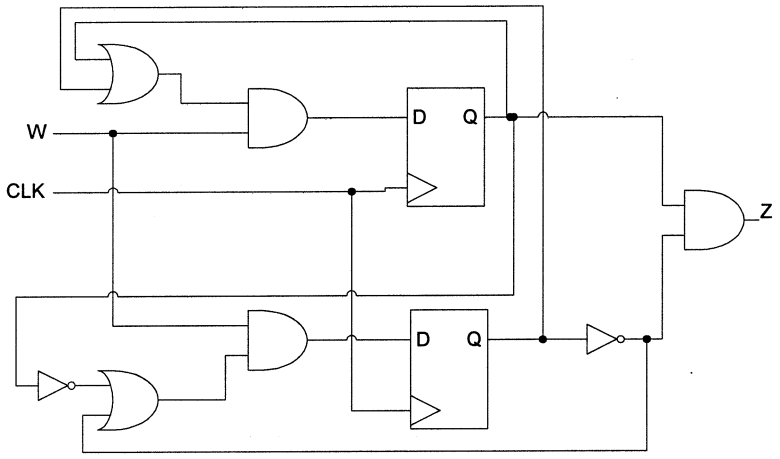
Explanation of the above timing diagram: during the first three cycles after the synchronous reset, the valid output is low, and the *out_stream* can be anything (don't care, denoted *d* in the diagram). During the fourth cycle, the *valid* output goes high, and the number 4 appears on *out_stream* (this is the average of 2,5,3, and 6). Similarly, for all remaining cycles, the value appearing on *out_stream* is the average of the inputs received during the current cycle and the previous three cycles. Note that all outputs change aligned to the positive (0-to-1) clock edge.

Write synthesizable VHDL code to specify the behaviour of this circuit. Marks will be deducted for solutions which are overly long or complex. You can assume that the division operator ("/") is synthesizable, and that if the result of a divide is fractional, it is rounded down. [8 marks]

Write your answer on the next page. You can remove this page from the exam book if you like. No writing on this page will be marked.

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

entity avg is
   port ( clk, reset : in std_logic;
          instream : in std_logic_vector(7 downto 0);
          valid : out std_logic;
          outstream : out std_logic_vector(7 downto 0));
end avg;

architecture behavioural of avg is

begin
 process (clk)
 variable int_value1, int_value2, int_value3, int_value4 : std_logic_vector (7 downto 0);
 variable samples : integer;

 begin
  if (clk = '1' and clk'event) then
        if (reset = '1') then
            samples := 1;
            valid <= '0';
            outstream <= "00000000";
            int_value4 := "00000000";
            int_value3 := "00000000";
            int_value2 := "00000000";
            int_value1 := instream;
       else
            samples := samples + 1;
            if (samples > 3) then
                valid <= '1';
            else
               valid <= '0';
            end if;

            int_value4 := int_value3;
            int_value3 := int_value2;
            int_value2 := int_value1;
            int_value1 := instream;

            outstream <= (int_value1 + int_value2 + int_value3 + int_value4) / 4;
     end if;
  end if;
 end process;
end behavioural;
```

6. Consider the following circuit:



Assume the following:

Clk-to-Q delay of each flip-flop: 0.5 ns
Delay of each logic gate: 1 ns
Hold time of each flip-flop: 0.2 ns
Set up time of each flip-flop: 0.3 ns

a) What is the maximum frequency of the clock in this circuit (in Mhz) ? [2 marks]

0.5 ns + 3 ns + 0.3 ns = 3.8 ns

(1/3.8 ns)

b) Suppose the hold time is not 0.2ns. What is the largest value for the hold time for which this circuit will still function properly? [2 marks]

fastest path is through two gates:

fail if 0.5 ns + 2 ns < hold time

fail if hold time > 2.5 ns

c) Would it be possible to describe this circuit in a single synthesizable VHDL process? Circle Yes or No. You do not need to actually describe it in VHDL. [1 mark]

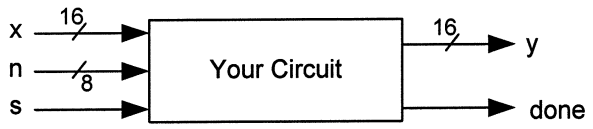
YES

7. In this question, you are to design a circuit to take the square root of a number. You are given the following algorithm for approximating the square root of a number as the basis for the datapath of your circuit:

$y = x$
for ($i = 0$ to n-1) {
  $y = ½ * (y + x/y)$
}
the result is in y

In each of the n iterations, the above algorithm computes a new value for y, based on the old value of y and the value of x. At the end of the n iterations, y contains an approximation to the square root of x. The higher the value of n (i.e. the more times you iterate), the better the approximation.

As shown below, your circuit will have two input buses: one for x (16 bits) and one for n (8 bits), and a start input s. When s goes high, the calculation starts in the next clock cycle. Once the calculation is done, the circuit presents the result on the y output bus, and asserts the done signal. This is the same timing behaviour we assumed in class when we talked about datapath circuits.



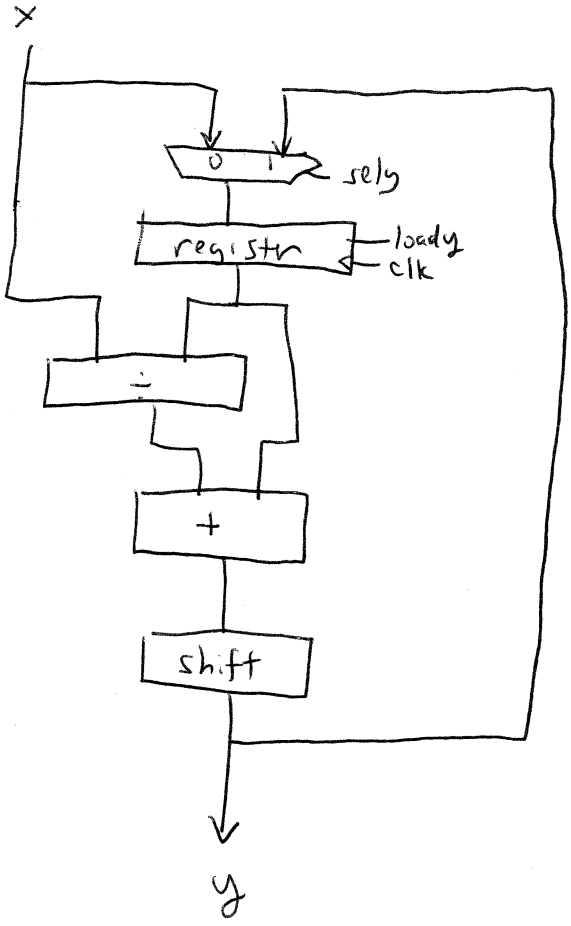
Although this would normally be done with floating point arithmetic, for the purposes of this question, assume we are using integer arithmetic only. Ignore overflows and underflows.

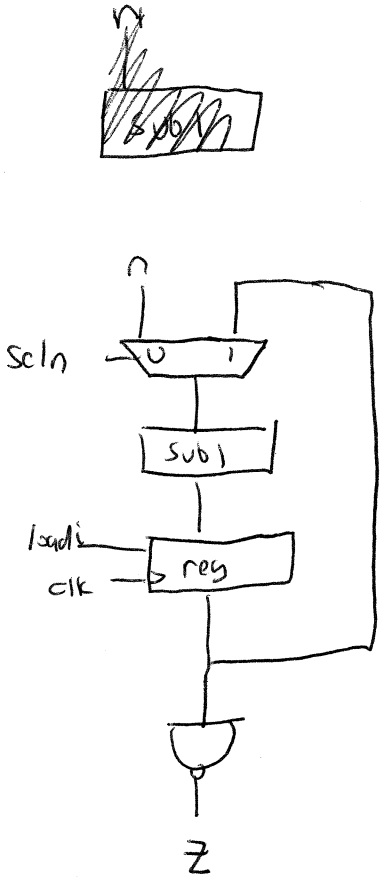
Assume you have the following component available:
- Basic gates, flip-flops, registers, multiplexers
- Any number of 8 or 16-bit adders and subtractors
- Any number of 8 or 16-bit dividers that can divide two numbers in a single clock cycle
- Any number of 8 or 16-bit shift blocks (to divide by 2, for example)
- Any number of tri-state drivers

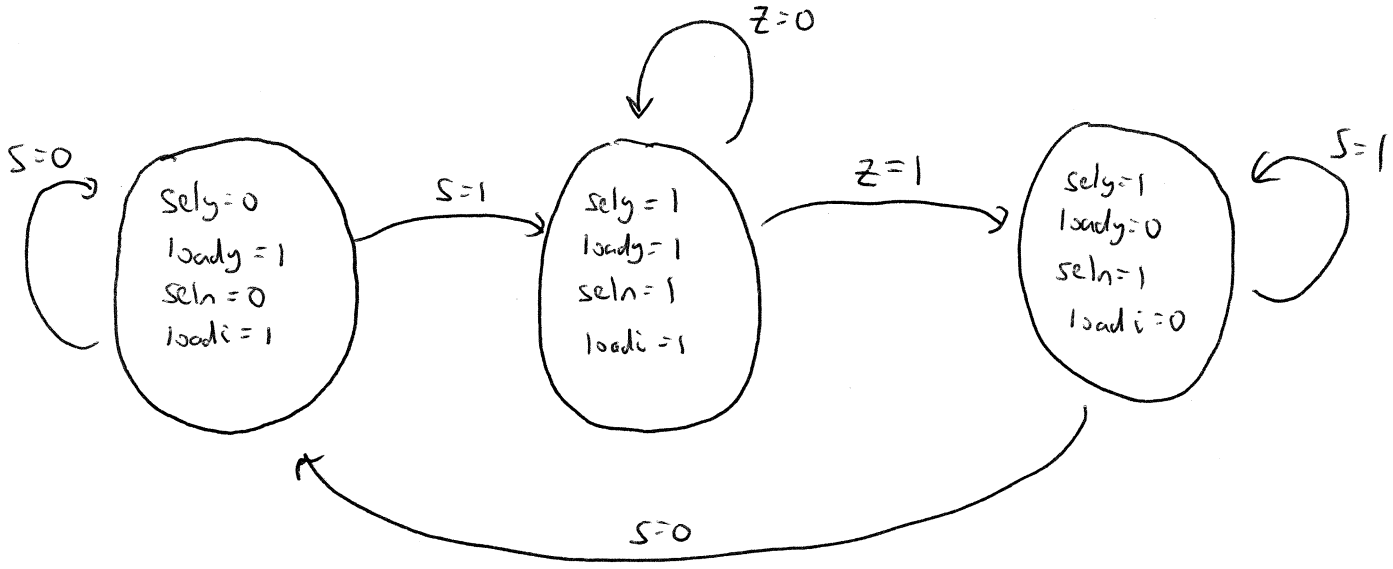
Design a datapath and accompanying state machine that would implement the above algorithm. Present the datapath as a schematic and the state machine as a bubble (state) diagram. Do not use VHDL. Clearly state any assumptions. [8 marks]

Write your answer on the next page. You can remove this page from the exam book if you like.
No writing on this page will be marked.

Answer Question 7 on this page.

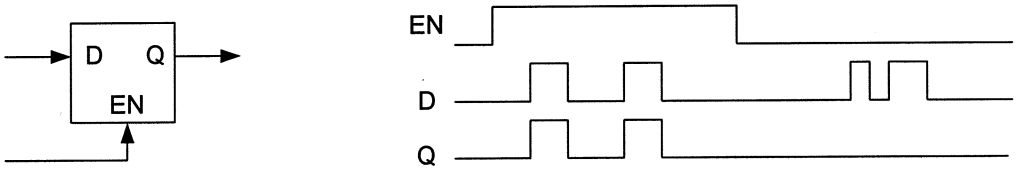






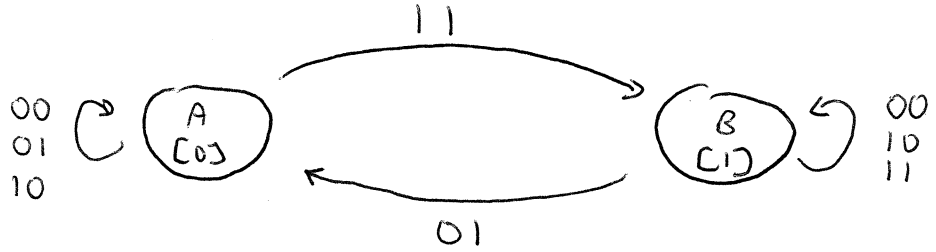
Other answers are possible.

8. Consider a level-sensitive latch. Recall that a level-sensitive latch has an Enable signal (which we will call EN). When EN is high, the output Q is equal to the input D. When EN is low, the output Q maintains its value. A timing diagram is as below.



a) Suppose you are to design this circuit using asynchronous state machine design techniques. Draw a state diagram (bubble diagram) that describes the behaviour of this level-sensitive latch. Clearly label all transitions and input and output values. You can assume either a Moore Machine or a Mealy Machine structure. [2 marks]

all labels: D E



Other state diagrams are possible.

b) Given your state diagram, choose a suitable state assignment (binary encoding of state bits for each state). Remember this is an asynchronous machine. If you can not find a suitable state assignment, indicate why not, and how you can modify your state diagram so that you can find a suitable state assignment. [2 marks]

A: 0

B: 1

c) Write logic equations for the next state logic and the output logic. Show your work clearly, and present your answer in terms of equations (you do not need to draw the schematic). Do not use VHDL. [3 marks]

Y \ DE	00	01	11	10
0	0	0	1	0
1	1	0	1	1

note order.

hazard cover.

$$Y = DE + Dy + \overline{E}y$$

$$Q = y. \quad \text{by observation}$$

Other answers are possible.