

Course Examinations 2002-2003

Course Code & Title : **CEG 5010 Rapid Prototyping of Digital Systems**

Time allowed : **2** hours **0** minutes

Student I.D. No. : Seat No. :

Open book examination. Please answer all questions.

1. General FPGA questions:

- (a) (5 marks) Explain how multiplexers can be implemented using tristate buffers. What might be an advantage of this approach?
- (b) (5 marks) Give two advantages and two disadvantages of using FPGAs to perform computation over an ASIC.
- (c) (5 marks) We wish to implement the boolean logic function

$$f = (ab + c) + d$$

using 4 input lookup tables. Write out the contents of an FPGA 16×1 ROM which can be used to implement this equation.

- (d) (5 marks) We wish to implement a 7 bit parity function, the output of which is equal to '1' iff the number of set bits in the input are even. Explain how this can be implemented using 2 lookup tables and draw a circuit of their implementation which includes the values of the ROMs for both tables.
 - (e) (5 marks) Explain using a diagram how a double data rate transfer scheme can be used to double the throughput of a normal edge triggered scheme without using a double frequency clock.
2. Bit Serial Design. In the questions below, assume that the bit-serial inputs, x and y , are two's complement numbers which are 4-bits in length. Also assume that there is an input signal, LSB which is 1 when the least significant number is being transmitted, 0 otherwise. Your answer should only use standard logic gates (AND, OR, XOR etc), multiplexers, adders, subtractors, flip-flops and ROMs. In your answer, try to minimize the amount of resources and latency of the design.
- (a) (10 marks) Draw the full schematic of a fully pipelined bit-serial subtractor. Explain how it works. The input should be the LSB signal, x and y . The output should be $x - y$.
 - (b) (10 marks) Draw the full schematic of a fully pipelined bit-serial max function:

$$max(x, y) = \begin{cases} x & \text{if } x \geq y \\ y & \text{otherwise} \end{cases} \quad (1)$$

The input should be the LSB signal, x and y . The output should be $\max(x, y)$. You can use the subtractor of the previous question as a block in your design. What is the latency of your implementation?

- (c) (5 marks) Estimate whether a bit-parallel or bit-serial implementation of the \max function would give a higher throughput per slice? Throughput per slice is the throughput (outputs per second) divided by the number of slices used. Justify your answer.

3. ROM-based Numerically Controlled Oscillator. A ROM-based approach is used to generate a sine wave output. Such a system is called a numerically controlled oscillator (NCO) and has no inputs (apart from a clock). The outputs are two's complement digital values which, if passed to an 8-bit digital to analog converter (DAC) produces an analog sine wave output.

Components to be used for this design are a 16×8 -bit ROM, a counter for the address and a DAC which is used to form the analog output. You can assume that the DAC takes two's complement numbers as inputs. The only synchronous element in the design is the counter. Your answers should only use standard logic gates (AND, OR, XOR etc), multiplexers, adders, subtractors, flip-flops and ROMs. Try to minimize the amount of resources used.

- (a) (10 marks) Draw a full schematic diagram of the NCO, showing how the counter, ROM and DAC are connected together. The symmetry of the sin function **should not** be used in your answer to this question. If the output of the ROM should be the maximum possible value when $\sin(x) = 1$, produce a table which has the columns:

- i. *address* (as a 4-bit unsigned decimal number)
- ii. *ROM contents* (in decimal numbers between -128 and 127 inclusive)

for all addresses of the ROM. Explain how you calculated the contents of the ROM.

- (b) (5 marks) What is the minimum number of Virtex slices your design will occupy? Justify your answer and show a breakdown of the usage.
- (c) (10 marks) A more efficient scheme would be to store $1/4$ of the sin wave table in ROM (i.e. the values for angles $0 \leq x < 90$, and use symmetry to obtain $\sin(x)$ for the other angles (i.e. $90 < x < 360$). To produce identical results to the previous part of the question, a 4×8 - bit ROM is required. Explain with the aid of diagrams how the circuit must be changed so that it produces exactly the same output as the previous approach, but uses a much smaller ROM. Draw complete circuits for the parts which are changed.

4. A CORDIC scheme can be also be used to compute $\sin(x)$. The iterations of the CORDIC operation are:

$$x_{i+1} = x_i - (y_i d_i 2^{-i}) \quad (2)$$

$$y_{i+1} = y_i + (x_i d_i 2^{-i}) \quad (3)$$

$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i}) \quad (4)$$

$$(5)$$

where

$$d_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{otherwise} \end{cases} \quad (6)$$

After the CORDIC algorithm completes,

$$x_n = \frac{1}{K} (x_0 \cos(z_0) - y_0 \sin(z_0)) \quad (7)$$

$$y_n = \frac{1}{K} (y_0 \cos(z_0) + x_0 \sin(z_0)) \quad (8)$$

$$z_n \approx 0. \quad (9)$$

- (a) (10 marks) Write the VHDL description of a single CORDIC element which calculates an iteration of this algorithm (you can ignore the effect of K in the equations above). The VHDL entity description is given below (note that “tanval” contains $\tan^{-1}(2^{-i})$):

```
entity cordicsin is
port (
  clk: in std_logic;
  i: in std_logic_vector(1 downto 0); -- iteration number
  xi: in std_logic_vector(3 downto 0);
  yi: in std_logic_vector(3 downto 0);
  zi: in std_logic_vector(3 downto 0);
  tanval: in std_logic_vector(3 downto 0); -- value of atan(2^{-i})
  xiplus1: out std_logic_vector(3 downto 0);
  yiplus1: out std_logic_vector(3 downto 0);
  ziplus1: out std_logic_vector(3 downto 0);
);
end cordicsin;
```

- (b) (10 marks) How many Virtex slices does your design occupy? Justify your answer and show a breakdown of the usage.
- (c) (5 marks) Explain a situation in which the CORDIC approach occupies much less resources than the ROM based approach of the previous question.

-End-