E3.05
ISE3.19

IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2004

MSc and EEE/ISE PART III/IV: MEng, BEng and ACGI

DIGITAL SYSTEM DESIGN

Wednesday, 5 May 10:00 am

Time allowed: 3:00 hours

There are SEVEN questions on this paper.

Answer FOUR questions.

All questions carry equal marks

Corrected Copy

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : P.Y.K. Cheung
Second Marker(s) : D.M. Brookes

Special instructions for invigilators: None

Information for candidates:

In the figures showing digital circuits, all components have, unless explicitly indicated otherwise, been drawn with their inputs on the left and their outputs on the right. All signals labelled with the same name are connected together. All circuits use positive logic. The least significant bit of a bus signal is labelled as bit 0, and the most significant bit with the highest integer number. Therefore the signal X[7:0] is an eight bit bus with X7 being the MSB and X0 the LSB.

Hexadecimal numbers are prefixed with '$'. For example the decimal number 10 is written as $A.

In questions involving circuit design, you may use any standard digital circuits that are not explicitly forbidden by the question provided that you fully specify their operation.

Marks may be deducted for unnecessarily complex designs unless you are explicitly instructed not to simplify your solution.

1. Figure 1.1 shows a controller circuit for a crisp vending machine. The machine accepts *50p*, *20p* and 10p coins and, if excess money has been inserted, will retain credit for the next purchase.

Coins can only be inserted one at a time and, for each coin inserted, one of the signals *P50*, *P20* or *P10* goes high for about 20ms. There is at least a gap of 100ms between each coin insertion. A packet of crisp is released when *VEND* goes high.

The controller circuit consists of a FSM implemented with a ROM and a 5-bit register REGA clocked by the signal CLK. A 4-bit adder ADD4 and a second register REGB keep the current credit received so far. The carry input of the adder is connected to logical '0' and the carry out is one of the inputs to the multiplexer. The multiplier MUX41 selects *P50*, *P20*, *P10* or *COUT* to drive the FSM.

The contents of the ROM are as follows:

Addr[2:0]		*Data[7:0]*			
STATE[1:0]	*TEST*	*NXT_STATE[1:0]*	*CREDIT[3:0]*	*NXT_ADD*	*NXT_VEND*
0	0	1	0	0	0
0	1	0	5	1	0
1	0	2	0	0	0
1	1	1	2	1	0
2	0	3	7	0	0
2	1	2	1	1	0
3	0	0	0	0	0
3	1	0	7	1	1

a) Assuming that a 50p coin is inserted and by considering State 0 alone (i.e. *STATE[1:0]*=0), explain the entry and exit condition for State 0 and the action taken in this state.

[4 marks]

b) With the help of an ASM chart or a state transition diagram, shows the sequence of states occupied by the FSM, the factors that determine the state transition and the actions taken in all other states.

[8 marks]

c) Explain with justification how you deduce the price of a packet of crisp.

[4 marks]

d) Assuming that the worst case delays are given as follows: adder, 5 ns; multiplexer select to output, 4 ns; multiplexer input to output, 1 ns; ROM, 8 ns; register setup time, 2 ns; hold time 0 ns; clock-to-Q, 2 ns, what is the maximum frequency of the clock signal *CLK* for the controller to work correctly?

[4 marks]

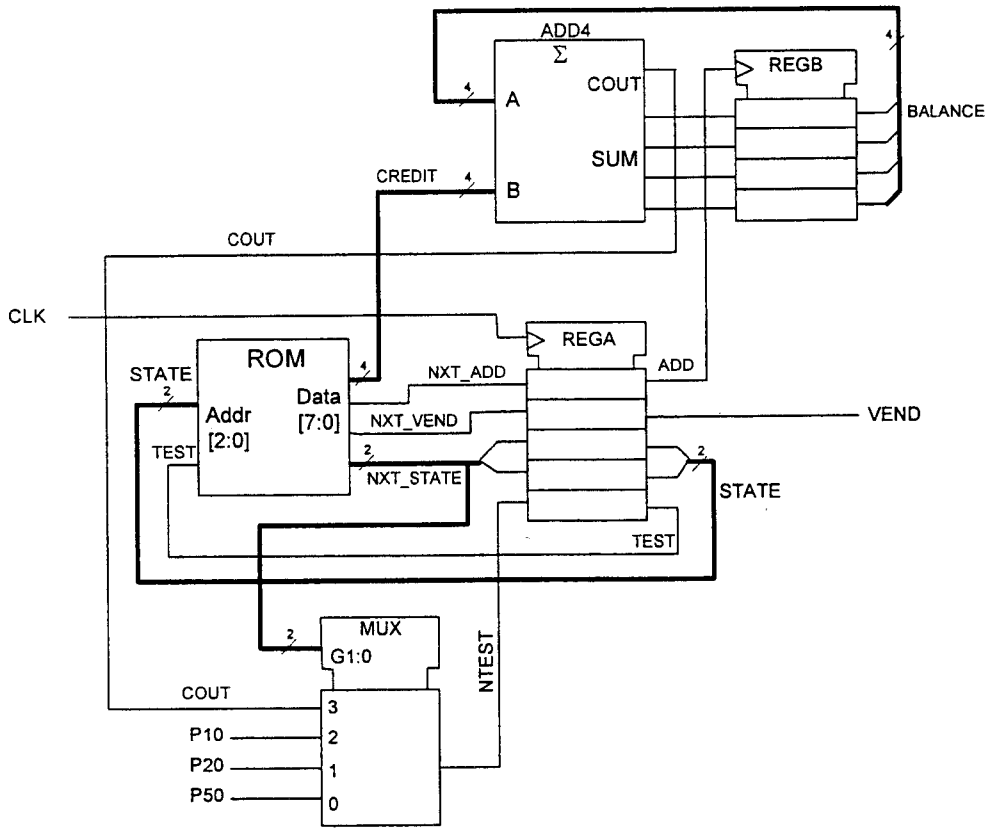


Figure 1.1

2. a) Design an 8-bit x 8-bit unsigned, parallel multiplier using only adders and four 4 x 4 unsigned parallel multipliers. Show your solution in the form of a clearly labelled schematic diagram.

[12 marks]

b) The Altera FLEX10K family of FPGAs contain Embedded Array Blocks (EABs) that can be configured as 2048 x 1-bit, 1024 x 2-bit, 512 x 4-bit or 256 x 8-bit ROM blocks. They also contain Logic Elements (LEs), each having a 4-input Look-Up Table (LUT) and a register. Each LE can implement one bit of a ripple carry, and each EAB is operated synchronously with a latency of 1 clock cycle.

The multiplier in (a) is modified to become a pipelined multiplier and is implemented using a FLEX10K device. How should the solution in (a) be modified? What is the pipeline latency of the multiplier? Estimate with justification the resources needed to implement this multiplier on a FLEX10K device.

[4 marks]

c) Assuming that the delays of the EAB, the LUT and the register inside a Logic Element are 4 ns, 2 ns and 1 ns respectively, and that the setup time of the register is 1 ns, estimate the maximum clock rate that this pipelined multiplier will operate. Ignore any interconnect delay.

[4 marks]

3. Figure 3.1 depicts a 256 x 32 bit synchronous stack or last-in-first-out (LIFO) memory module that is implemented inside an Altera FLEX10K FPGA device. All operations are synchronous to a system clock signal `clk`. The signal `reset` initializes the LIFO to an empty state. The LIFO is enabled by the `strobe` signal which goes high for one cycle when there is either a push or a pop operation.

On the rising edge of `clk`, input data `d_in` is pushed onto the LIFO if `push_pop` and `strobe` signals are both high. Similarly output data is popped onto `d_out` if `push_pop` is low and `strobe` is high. The first data words read out from the LIFO is the last data stored. If the number of data words remaining in the LIFO is ~~511~~, the signal `full` goes high and any further push operations will be ignored. Similar if the number of data words stored is 0, the signal `empty` goes high and any further pop operations will be ignored.

a) Using counters (as address pointers), static RAM and other circuit components, sketch the block level design of this synchronous LIFO. You may use any functional logic blocks such as adders, subtractors and comparators. Marks may be deducted for unnecessarily complicated circuit.

[12 marks]

b) Figure 3.2 depicts a Embedded Array Block (EAB) found in the FLEX10K family of FPGA. Each EAB can be configured as 2048 x 1, 1024 x 2, 512 x 4 or 256 x 8 block of RAM. The data output can be registered or unregistered depending on how the EAB is configured as shown in Figure 3.2.

(i) With the aid of a diagram, explain how you would use multiple EABs to implement the memory used in the LIFO.

[3 marks]

(ii) Estimate with justification the approximate number of logic elements (LEs) required to implement the LIFO. Each LE consists of a 4-inputs-1-output lookup table (LUT) and an optional 1-bit register.

[5 marks]

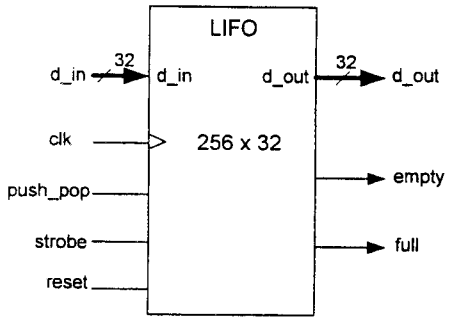


Figure 3.1

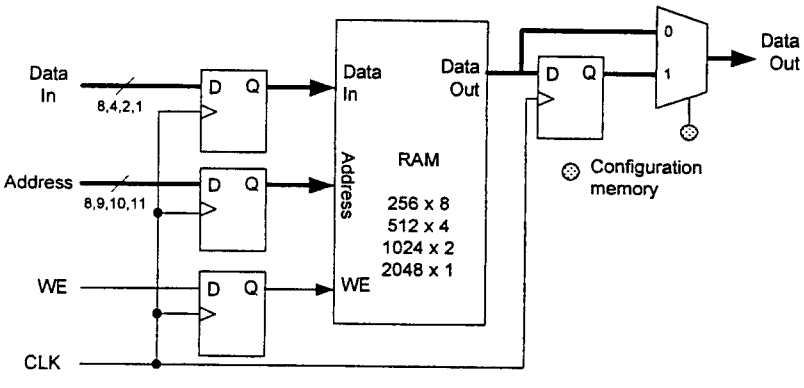


Figure 3.2

4. a) What is the meaning and function of a 'parity bit' in connection with storage of digital information in memory?

[2 marks]

b) Show how to correct single bit errors in data storage by adding a number of parity check bits to the data being stored. If N check bits are added, how many data bits is it possible to protect against single-bit errors? Illustrate your answer using 4-bit data as an example.

[3 marks]

c) Figure 4.1 shows an error-correcting memory module that can correct single bit error in the data `D7:0`. The signal `ERROR` at the output is high when an error is detected and low otherwise.

i) Design the parity bit generator circuit in the form of Boolean equations.

[5 marks]

ii) Design the parity checking circuit and the error correction circuit. Demonstrate that your design works properly with the example `D7:0` = 10100011 where an error has occurred in `D2`. Gate level design is not required.

[10 marks]

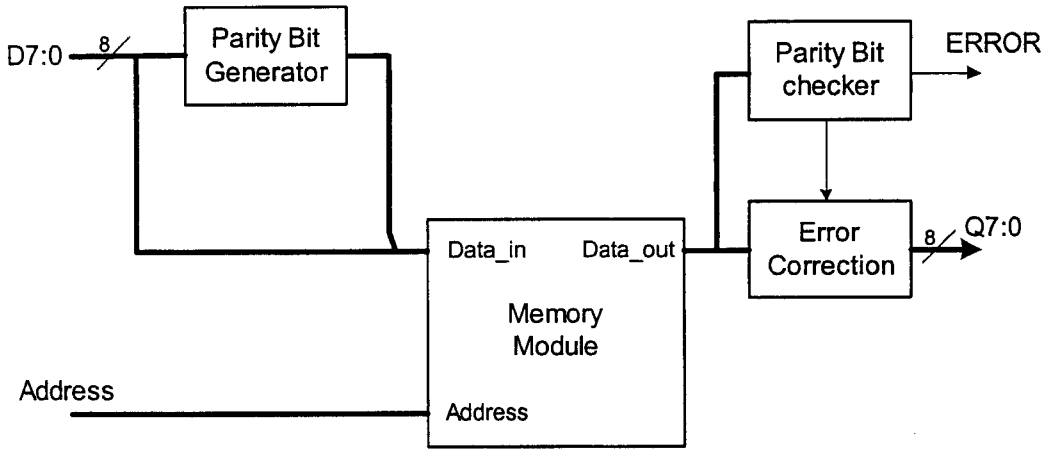


Figure 4.1

5. *Figure 5.1* shows an electronic system that counts the number of people entering a building using interruption of light on sensors A and B. The outputs of the sensors, `in_A` and `in_B` are high when the sensors are obscured. Two sensors are used to ensure that glitches on either `in_A` and `in_B` are ignored. This prevents a person dithering at the door from being counted more than once. The two sensors signals `in_A` and `in_B` are used to drive a finite state machine which generates a `ctr_enable` signal to a synchronous up counter as shown in the timing diagram shown in Figure 5.2. The system should ignore people leaving the store. You may also assume that the door is narrow enough that people enter or leave singly and there is always a gap between them.

a) Design the finite state machine in the form of a state diagram. State any assumptions that you make.

[12 Marks]

b) Using one-hot encoding, design the FSM in the form of registers and Boolean equations. A circuit diagram is not required.

[8 Marks]

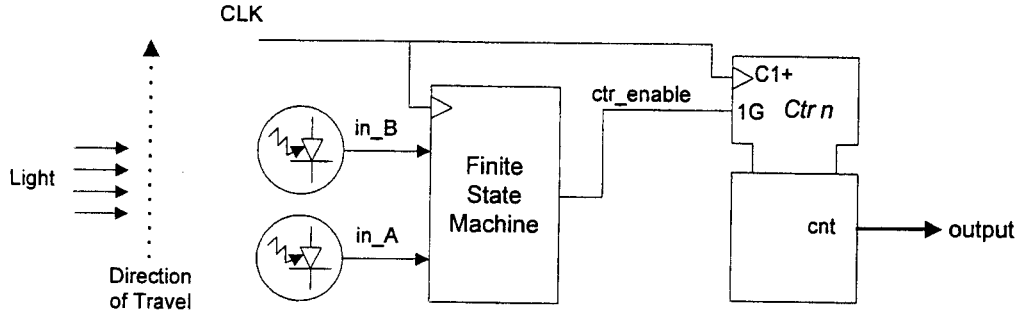


Figure 5.1

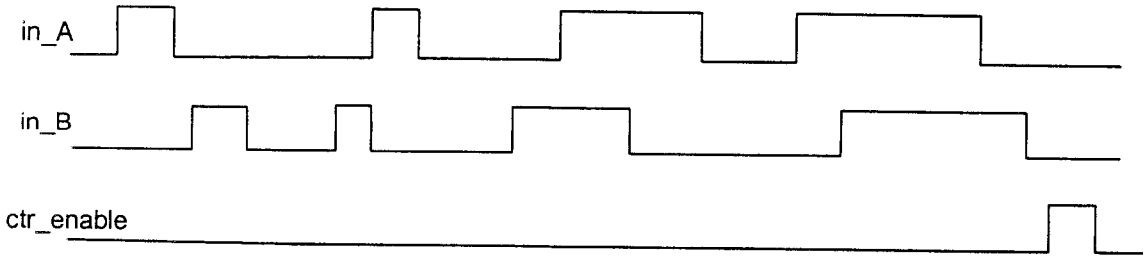


Figure 5.2

6. *Figure 6.1* shows a microprocessor connected to a 16M x 32 bit SDRAM module through an address decoder and an interface circuit. The timing of the microprocessor and the SDRAM for reading a burst of four memory words is shown in *Figure 6.2*. The address bus `A31:0` becomes valid and the address strobe signal $\overline{AS}$ is asserted at time B after the rising edge of the clock signal `CLK` during the clock cycle T0. The period of the clock is A. At time C after $\overline{AS}$ is asserted, the address decoder circuit produces a chip select signal $\overline{CS}$, which is asserted low for addresses in the range $00000000 to $00FFFFFF provided that $\overline{AS}$ is also low. The row address strobe signal $\overline{RAS}$ and the column address strobe signal $\overline{CAS}$ are asserted by the SDRAM interface circuit for one clock cycle on the falling edge of the clock during T1 and T4 respectively.

   The microprocessor reads four consecutive words from SDRAM on the rising edge of the clock if the $\overline{DTACK}$ signal is sampled low.

   Design the address decoder and the SDRAM interface circuit to generate $\overline{CS}$, $\overline{RAS}$, $\overline{CAS}$, $\overline{DTACK}$ and the memory address bus signals `AD11:0`. State the relationship between the time periods A, B and C, and any other constraints or assumptions under which your design is valid.

   [20 marks]

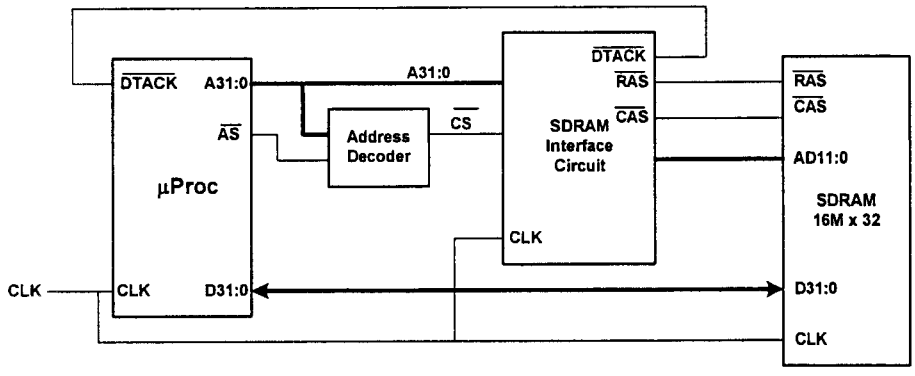


Figure 6.1

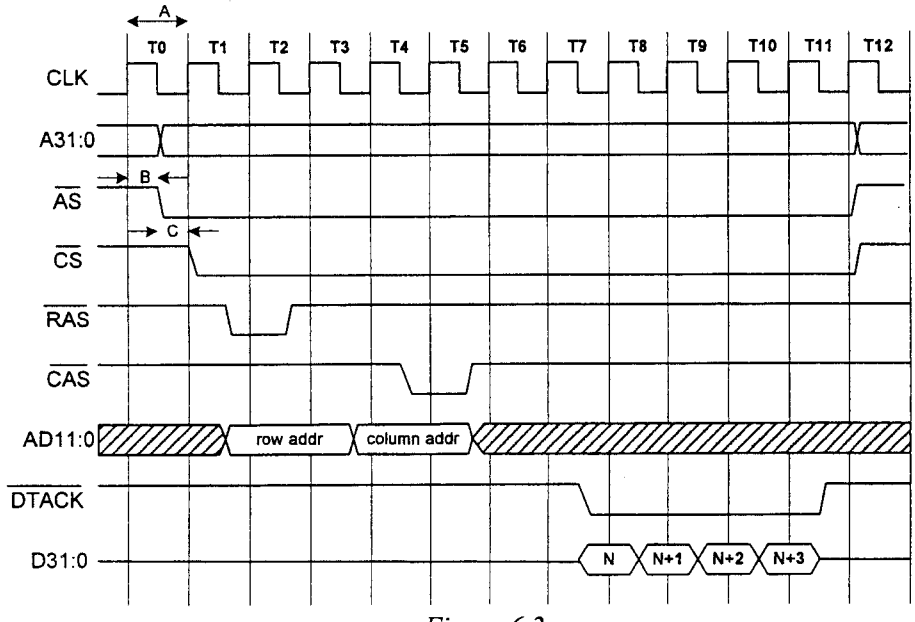


Figure 6.2

7. A 2-dimensional vector with coordinate (x,y) is rotated by 30° counter clockwise to (x',y') by applying equation 1:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad [1]$$

where $a = d = 0.1101111_2$, $b = 0.1_2$ and $c = -0.1_2$. The input coordinates x and y lie in the range +511 to –512, and are expressed in 2's complement form.

a) By employing distributed arithmetic, design at architectural level a circuit to implement this coordinate rotation engine. All inputs and outputs are synchronous to a system clock signal CLK and are available as parallel data. Show the widths of the datapaths and the positions of the binary points in your design so that there is no loss of precision throughout the circuit.

[12 marks]

b) Determine the contents of all ROMs used in your design.

[8 marks]

[END]

EJ.05
ISC 3.19
A~~C~~2

IMPERIAL COLLEGE OF SCIENCE TECHNOLOGY AND MEDICINE
UNIVERSITY OF LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
M.Eng., B.Eng., B.Sc(Eng.) and A.C.G.I. EXAMINATIONS 2004

PART III and PART IV

DIGITAL SYSTEM DESIGN

SOLUTIONS

First Marker: *Peter Y. K. Cheung*

Second Marker: *Mike Brookes*

22/3/04

Answer to Question 1

This question tests students' understanding of FSM as implemented using lookup table and register. It is also complicated by the fact that the FSM is link to a adder which keeps track of the current credit balance.

a) State 0 is entered either from State 0 or State 3. Before entering State 0, NXT_State[1:0] = 0, selecting P50 as output of MUX41 to drive TEST. We are therefore testing for a 50p coin in State 0. If a 50p is detected (i.e. P50 = '1'), three things happens: 1) the machine stays in State 0 until P50 goes low, 2) it outputs a value 5 (for 50p) on CREDIT[3:0] and, 3) it produces a rising edit on the signal ADD to update BALANCE stored in REGB. If a 50p is not detected, it proceeds to State 1.

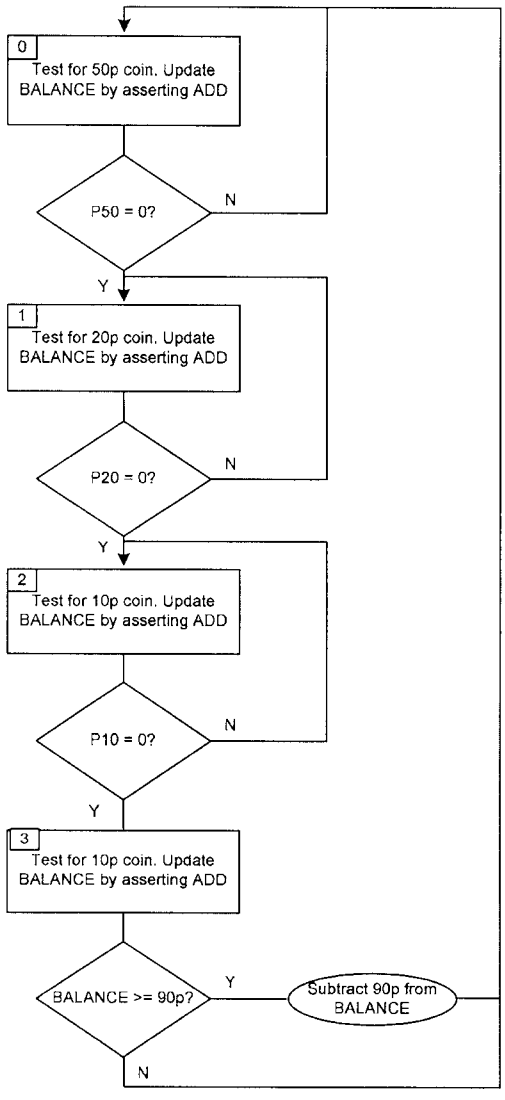
[4]

b) State 1: the same as State 0 except that it tests for a 20p coin.

State 2: almost the same as State 1, again it tests for a 10p coin. However, something else happens in State 2. The value of CREDIT is set to 7 if P10 is low. The adder will add 7 to the current BALANCE value. This will cause COUT to go high if BALANCE is 9 or higher before entering State 3. Note that although the adder performs the addition, the result sum is NOT saved in REGB because the NXT_ADD signal is low. This is merely a test!

State 3: NXT_State is always 0. However, if TEST is low (no COUT), nothing else happens. If TEST is high (i.e. COUT='1'), it indicates that BALANCE is 9 or higher. It asserts NXT_VEND and add 7 to BALANCE, which is the same as subtracting 9!

(I wonder how many students will get this!)



[8]

c) See b) above. Crisp costs 90p.

[4]

d) Worst case path is:

CLk on REGA to Q (2ns)

-> to ROM output (8 ns)

-> ADDER COUT (5 ns)

-> MUX in-out (1 ns)

-> REGA setup (2 ns)

Therefore CLK period must be at least:

(2 + 8 + 5 + 1 + 2) ns. Max frequency = 55.6 MHz.

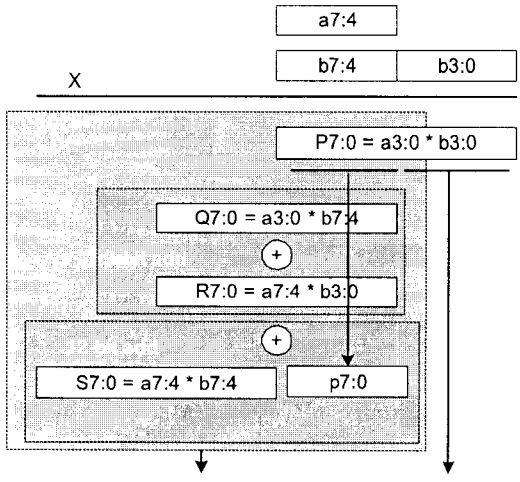
[4]

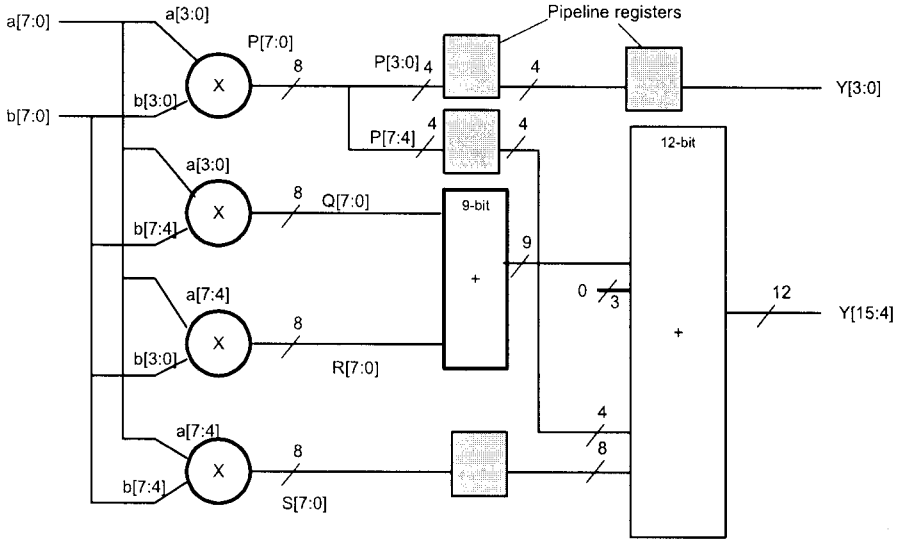
Answer to Question 2

a) The following diagram explains how 8 x 8 unsigned multiplication can be performed using 4 4x4 multipliers. The 8-bit numbers are divided into lower and upper nibbles. Use cross multiplication to obtain partial products and add these together. The method used here uses minimum size adders. I will accept designs using less optimal configurations.

Be careful about bit alignment and possible overflow.

This map to the following circuit:





[12]

b) Use registers inside EAB and adders for pipelining. Add pipeline registers at locations shown in shaded boxes. Pipeline latency = 3 cycles. Resources used: 4 EAB, 41 LEs.

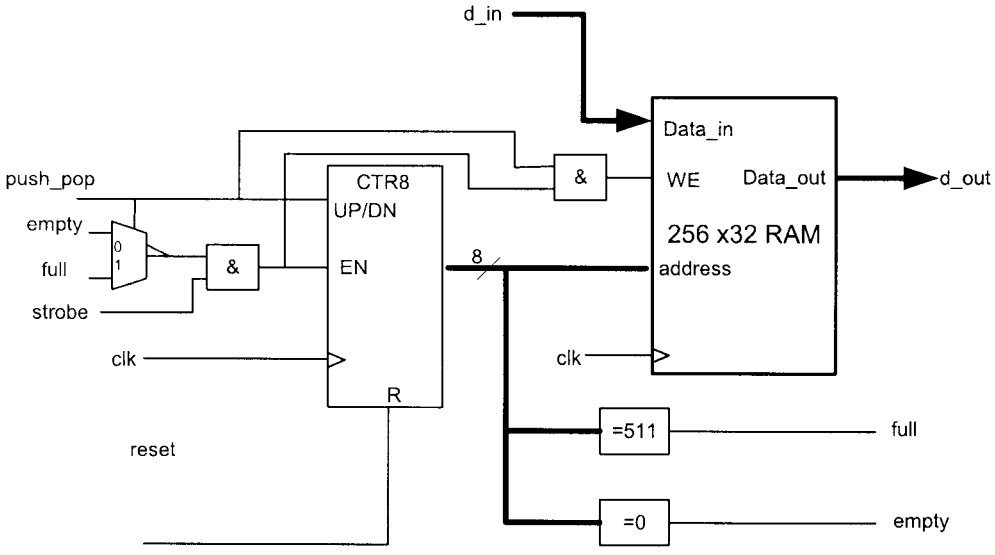
[4]

c) Worst case delay between each pipeline stage = clk-reg_out (1 ns) + slowest adder (12 x 2 ns) + setup time (1 ns) = 26 ns or 38.46 MHz.

[4]

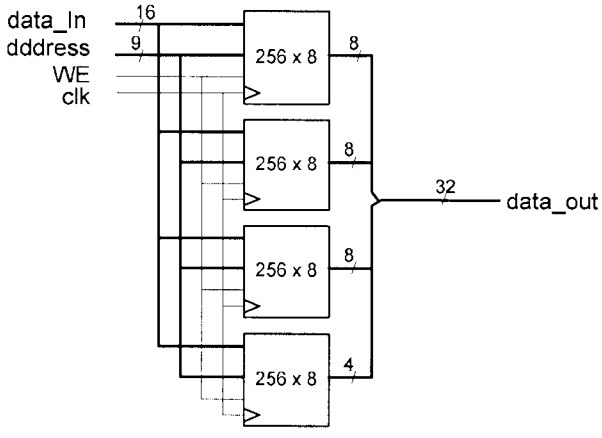
Answer to Question 3

a)



[12]

b) i)



[3]

(ii)

Component	Resources
Memory	4 EABs
Stack UP/DN counter	8 LEs
= 511 detect	2 LEs
= 0 detect	2 LEs
Other gates	2 LEs
Total	**4 EABs, 14 LEs**

[5]

Answer to Question 4

a)

Parity bit is an extra bit that can be added to a group of "0" bits and "1" bits to make the parity of the group odd or even. Odd and even parity means that there are odd or even number of "1" s in the group including the parity bit. Parity bit can be used to detect single bit error.

[2]

b)

Book work – show this using a 4-bit example adding 3 check bits. If there are N check bit, can correct single bit error in $2^N - N - 1$ bit wide data.

[3]

c) i)

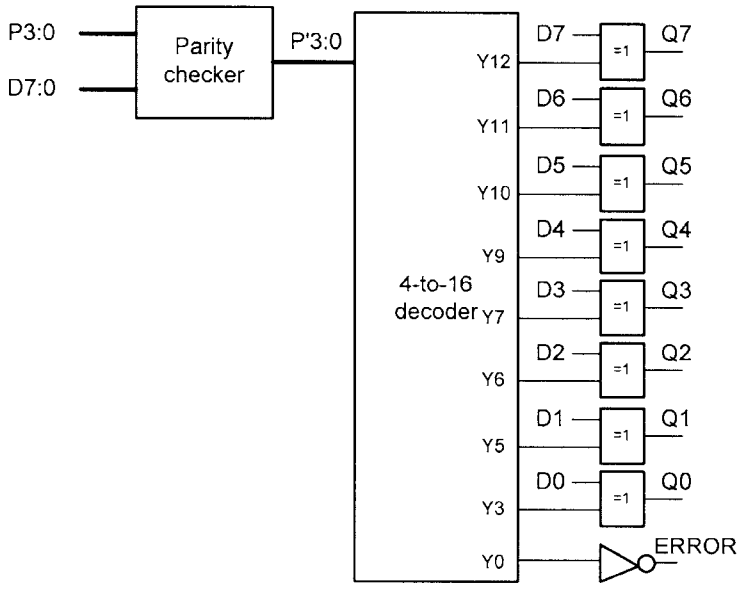
4 parity check bits are required to correct single bit error in 8-bit data. The 4 parity bits could be assigned according to this table:

	d7	d6	d5	d4	d3	d2	d1	d0
P3	x	x	x	x				
P2	x				x	x	x	
P1		x	x		x	x		x
P0		x		x	x		x	x
Code	12	11	10	9	7	6	5	3

P3 = d7 ⊕ d6 ⊕ d5 ⊕ d4
P2 = d7 ⊕ d3 ⊕ d2 ⊕ d1
P1 = d6 ⊕ d5 ⊕ d3 ⊕ d2 ⊕ d0
P0 = d6 ⊕ d4 ⊕ d3 ⊕ d1 ⊕ d0

[5]

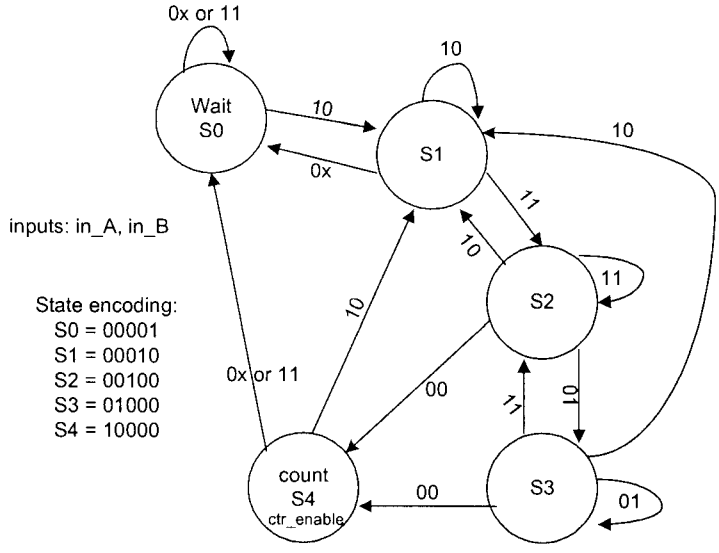
ii)



[10]

Answer to Question 5

a) Assert ctr_enable in S4 only.



[12]

b)

Current State					Inputs		Next State					
D4	D3	D2	D1	D0	in_A	in_B	Q4	Q3	Q2	Q1	Q0	ctr_enabl
0	0	0	0	1	0	X	0	0	0	0	1	0
0	0	0	0	1	1	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0	0	0	0	1	0
0	0	0	1	0	0	X	0	0	0	0	1	0
0	0	0	1	0	1	0	0	0	0	1	0	0
0	0	0	1	0	1	1	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	1	0	1	0	0	0	0
0	0	1	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	1	1	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	0	0	1	0	0
0	1	0	0	0	1	1	0	0	1	0	0	0
1	0	0	0	0	0	X	0	0	0	0	1	1
1	0	0	0	0	1	0	0	0	0	1	0	1
1	0	0	0	0	1	1	0	0	0	0	1	1

Q0 = D0&!in_A + D0&in_A&in_B + D1&!in_A + D4&!in_A + D4&in_A&in_B

Q1 = D0&in_A&!in_B + D1&in_A&!in_B + D2&in_A&!in_B + D3&in_A&!in_B + D4&in_A&!in_B

Q2 = D1&in_A&in_B + D2&i_nA&in_B + D3&in_A&in_B

Q3 = D2&!in_A&in_B + D3&!in_A&in_B

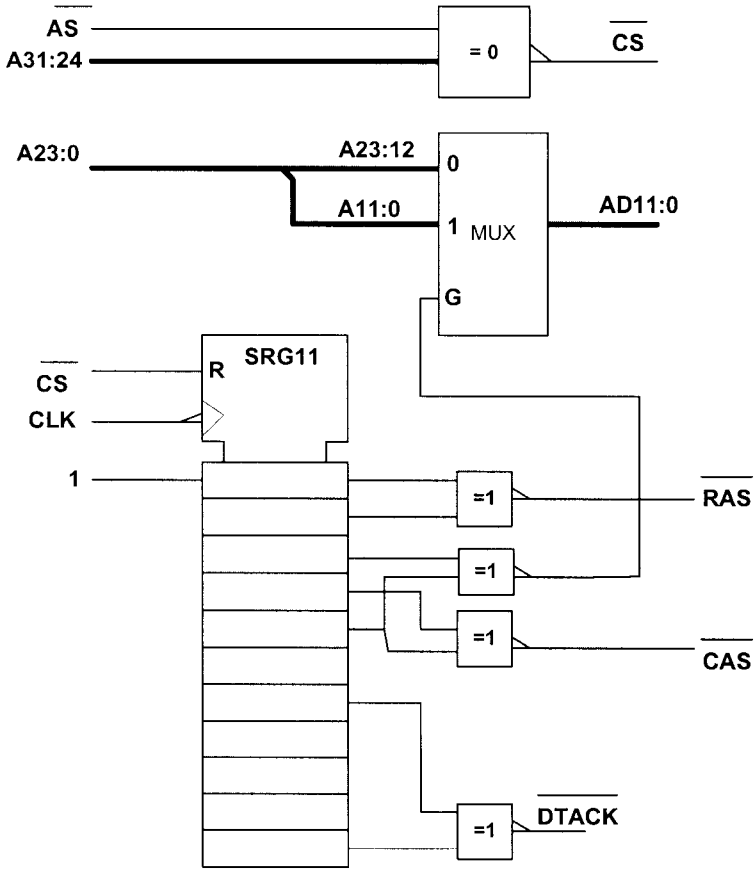
Q4 = D2&!in_A&!in_B + D3&!in_A&!in_B

ctr_enable = D4

[8]

Answer to Question 6

Shift register based design is probably the simplest:



[20]

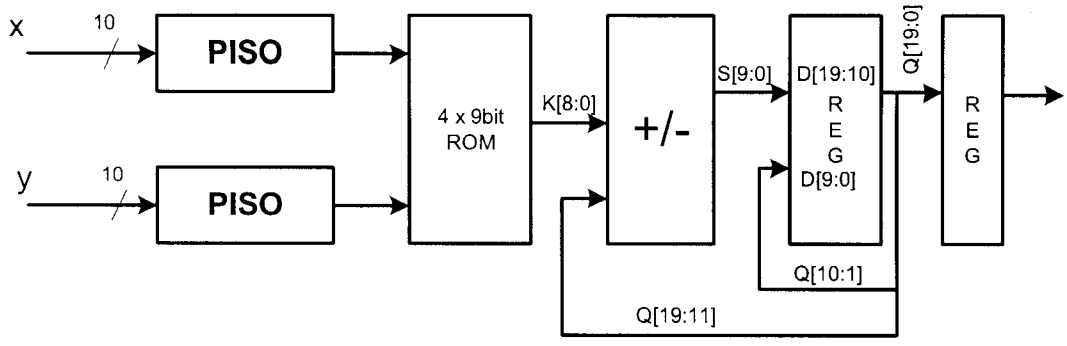
Answer to Question 7

Students have designed a digital filter using distributed arithmetics as the course work for this course. This question tests their ability to apply what they have learned in this course work to a problem they have not come across before.

(a) Two inner products:-

$$x' = [a \quad b] \times \begin{bmatrix} x \\ y \end{bmatrix} \qquad y' = [c \quad d] \times \begin{bmatrix} x \\ y \end{bmatrix}$$

Each inner product can be computed using a distributed arithmetic engine as shown below.



This circuit takes 10 cycles to compute each coordinate transformation. During the last cycle the adder/subtractor works as a subtractor.

[15]

(b)

Use 9-bit ROM in 2's complement fix-point format: S I . I I I I I I I, where S= sign, I = 0 or 1.

ROM1 ($ax+by$)

Address	Contents
0	0
1	a = 00.1101111
2	b = 00.1000000
3	a+b = 01.0101111

ROM2 ($cx+dy$)

Address	Contents
0	0
1	c = 11.1000000
2	d = 00.1000000
3	c+d = 00.0101111

[10]

[END]