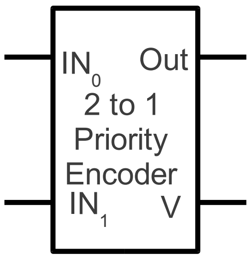
ECE 2030 B 12:00pm	Computer Engineering	Fall 2010
5 problems, 6 pages	Final Exam Solutions	13 December 2010

Problem 1 (3 parts, 24 points) — Design This

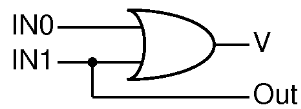
Complete each design below. Be sure to label all signals.

Part A: Define a 2 to 1 priority encoder, where $I_1 > I_0$, by completing the behavior table.

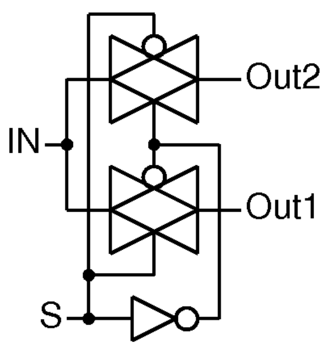


IN_0	IN_1	V	Out
0	0	0	X
1	0	1	0
X	1	1	1

Implement the 2 to 1 encoder using ***one*** basic gate. Only true (non-complemented) inputs are available. Label all inputs (IN0, IN1) and outputs (Out, V).



Part B: Implement a 1 to 2 demux using only pass gates and an inverter. Determine # of switches needed.



switches = 3 x 2 = 6T

Part C: Complete the truth table for even parity. Then write a sum of products (SOP) expression.

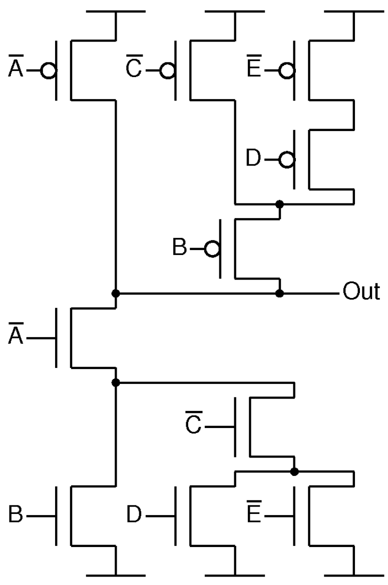
A	B	Out
0	0	1
1	0	0
0	1	0
1	1	1

$\overline{A \oplus B} = \overline{A}\cdot\overline{B} + A\cdot B$

Problem 2 (4 parts, 32 points) Design That

Complete each design below. Be sure to label all signals.

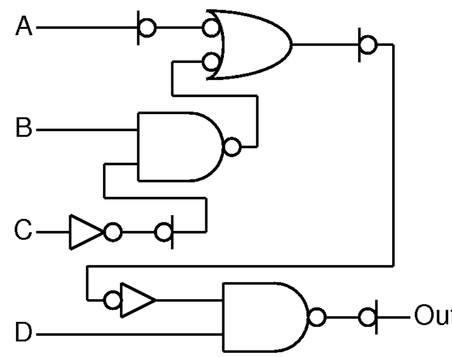
Part A: Complete the following CMOS design. Also express its behavior.



Out = $A+\overline{B}\cdot(C+\overline{D}\cdot E)$

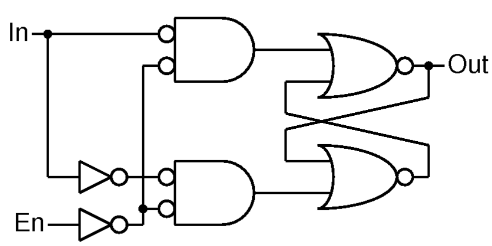
Part B: Implement the following expression using NAND and NOT gates. Use proper mixed logic design. Determine # of switches needed.

$$Out=\overline{\overline{\overline{A}+B\cdot\overline{C}}\cdot D}$$

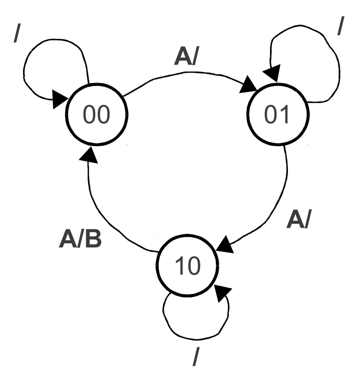


switches = 3 x 4 + 2 x 2 = 16T

Part C: Implement a transparent latch using only NOR and NOT gates.



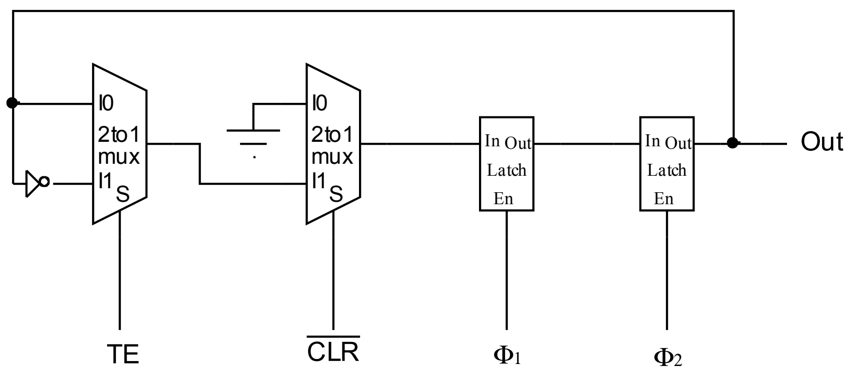
Part D: Draw the state table for the following state diagram.



A	S_1	S_0	NS_1	NS_0	B
0	0	0	0	0	0
1	0	0	0	1	0
0	0	1	0	1	0
1	0	1	1	0	0
0	1	0	1	0	0
1	1	0	0	0	1

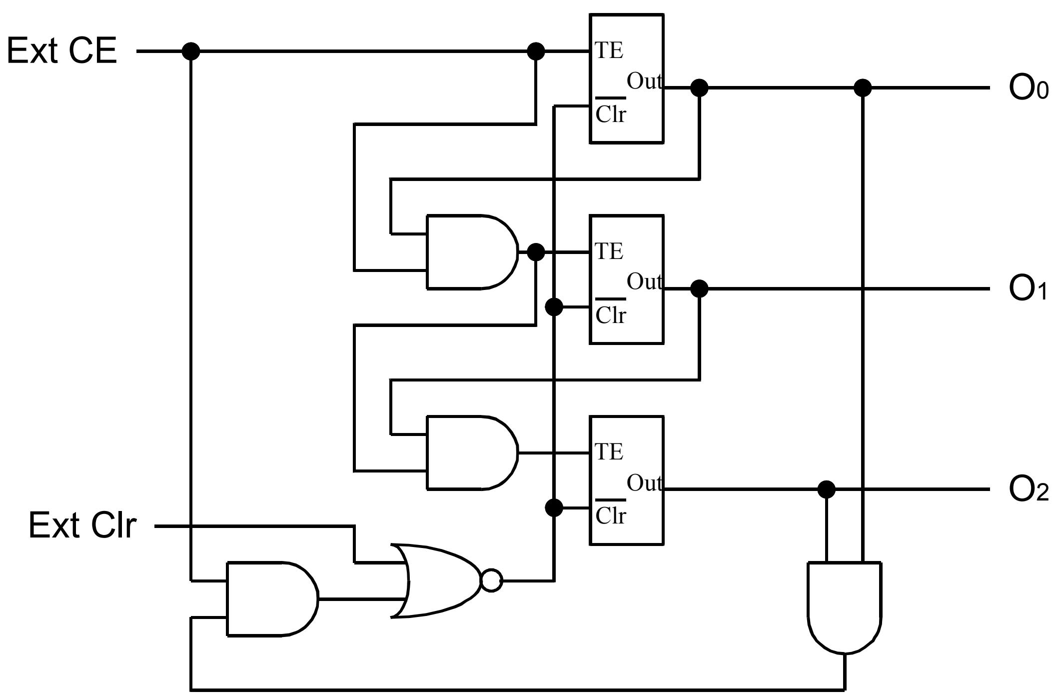
Problem 3 (3 parts, 30 points) Accountable

Part A (10 points) Design a toggle cell using **transparent latches**, **2to1 muxes**, and **inverters** (use icons, **labeling inputs & outputs**). Your toggle cell should have an active high toggle enable input **TE**, and an active low clear input $\overline{\textbf{CLR}}$, clock inputs Φ_1 and Φ_2, and an output **Out**. The $\overline{\textbf{CLR}}$ signal has precedence over **TE**. Also complete the behavior table for the toggle cell.

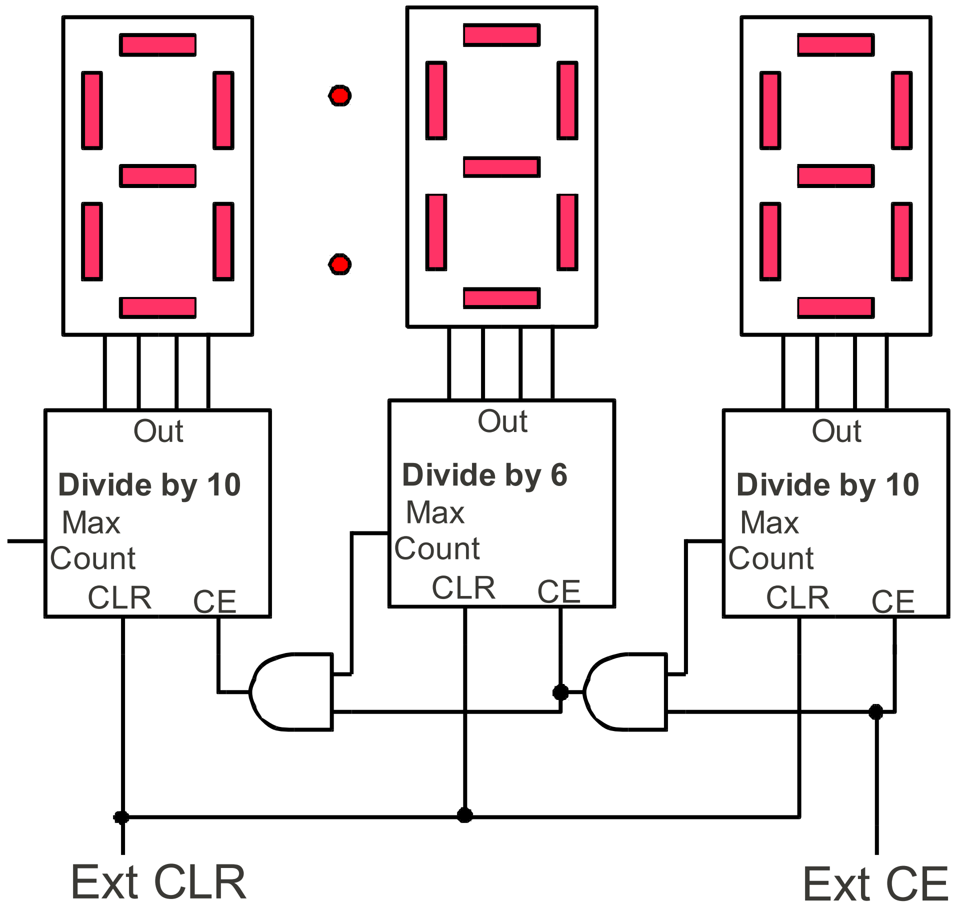


TE	$\overline{CLR}$	CLK	Out
0	0	↑↓	0
1	0	↑↓	0
0	1	↑↓	Q_0
1	1	↑↓	$\overline{Q_0}$

Part B (10 points) Now combine these toggle cells to build a **divide by 6** counter. Your counter should have an external clear, external count enable, and three count outputs O_2, O_1, O_0. Use any basic gates (AND, OR, NAND, NOR, & NOT) you require. Assume clock inputs to the toggle cells are already connected. *Your design should support multi-digit systems*.



Part C (10 points) Build a stopwatch that counts seconds and minutes using divide by N counters drawn below. ***Be sure to fill in the needed divider for seconds, tens of seconds, and minutes.*** Use any basic gates you require. Assume a one hertz clock is already connected.



Problem 4 (2 parts, 48 points) Microcode in Reverse

Part A (20 points) Translate this undocumented microcode fragment (in hexidecimal) to corresponding MIPS assembly instructions. Also include comments summarizing the instruction.

#	X	Y	Z	rwe	im en	im va	au en	- a/s	lu en	lf	su en	st	ld en	st en	r/-w	msel
1	5	0	7	1	0	0	0	0	0	0	0	0	1	0	1	1
2	7	0	9	1	1	C	0	0	0	0	1	0	0	0	0	0
3	9	0	9	1	1	FFF	0	0	1	8	0	0	0	0	0	0
4	9	A	A	1	0	0	1	0	0	0	0	0	0	0	0	0
5	8	A	0	0	0	0	0	0	0	0	0	0	0	1	0	1

1	`lw   $7, 0($5)`	`# $7 ← mem[pointer $5]`
2	`srl  $9, $7, 12`	`# $9 ← $7 >> 12`
3	`andi $9, $9, 0xFFF`	`# $9 ← $9 & 0xFFF`
4	`add  $10, $9, $10`	`# $10 ← $10 + $9`
5	`sw   $10, 0($8)`	`# mem[pointer $8] ← $10`

Part B (28 points) Complete a recursive subroutine that computes the factorial of N. Assume N is received in $1 and N! is returned in $2. $29 is the stack pointer.

label	*instruction*	*comment*
`Fact:`	`addi $02, $00, 1`	`# init result to 1`
	`slti $03, $01, 2`	`# if N < 2`
	`bne  $03, $00, Done`	`# you're done`
	`addi $29, $29, -8`	`# allocate stack space`
	`sw   $31, 4($29)`	`# push return address`
	`sw   $01, 0($29)`	`# push N`
	`addi $01, $01, -1`	`# decrement N`
	`jal  Fact`	`# call Fact(N-1)`
	`lw   $01, 0($29)`	`# pop N`
	`lw   $31, 4($29)`	`# pop return address`
	`addi $29, $29, 8`	`# deallocate stack space`
	`mult $01, $02`	`# N * Fact(N-1)`
	`mflo $2`	`# place result in $2`
`Done:`	`jr $31`	`# return to caller`

Problem 5 (4 parts, 39 points) "Random Bits"

Part A (9 points) Consider the instruction set architecture below with fields containing zeros.

0 0000	0000	0000	0000 0000 0000 0000 0000
opcode	dest. reg.	source 1 reg.	immediate value

What is the maximum number of opcodes? $2^5 = 32$

What is the number of registers? $2^4 = 16$

What is the range of the signed immediate value? $2^{20} = \pm 512K$

Part B (9 points) For the representations below, determine the most positive value and the step size (difference between sequential values). **All answers should be expressed in decimal notation**. Fractions (e.g., 3/16ths) may be used. Signed representations are two's complement.

representation	most positive value	step size
signed integer (15 bits) . (0 bits)	$2^{15} = 16K$	1
unsigned fixed-point (10 bits) . (5 bits)	$2^{10} = 1024 \approx 1K$	1/32
signed fixed-point (5 bits) . (10 bits)	15 999/1000	1/1K = .001

Part C (9 points) A 16 bit floating point representation has a 10 bit mantissa field, a 5 bit exponent field, and one sign bit. *Express all answers in decimal.* Fractions (e.g., 3/8) are okay.

What is the largest value that can be represented (closest to infinity)? 32K

What is the smallest value that can be represented (closest to zero)? 1/64K

How many decimal significant figures are supported? 3

Part D (12 points) For each problem below, compute the operations using the rules of arithmetic, and indicate whether an overflow occurs assuming all numbers are expressed using a **five bit unsigned fixed-point** and **five bit two's complement fixed-point** representations.

	11.1 + 111.1	11.011 + 10.101	10101 - 1010	100.00 + .01
result	1011.0	10.000	1011	100.01
unsigned error?	■ no □ yes	□ no ■ yes	■ no □ yes	■ no □ yes
signed error?	□ no ■ yes	■ no □ yes	□ no ■ yes	■ no □ yes