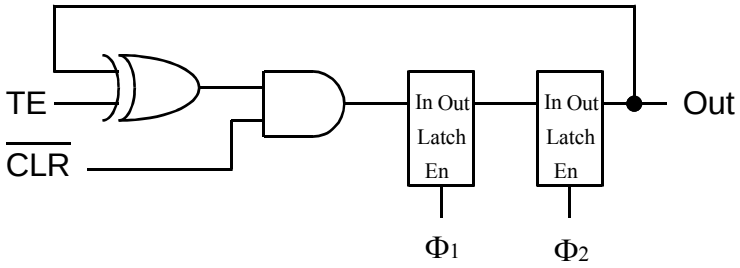
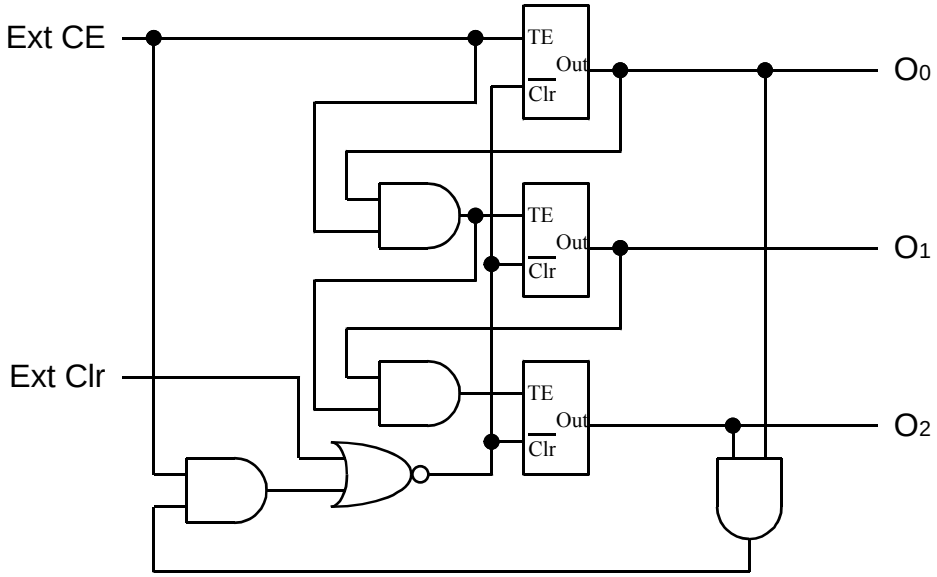
Problem 1 (2 parts, 24 points) Counters

Part A (12 points) Design a toggle cell using transparent latches and basic gates. Use an icon for the latch. 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**. Label all signals. Also complete the behavior table for the toggle cell.



TE	$\overline{CLR}$	CLK	Out
0	0	↑↓	**0**
1	0	↑↓	**0**
0	1	↑↓	$\mathbf{Q_o}$
1	1	↑↓	$\overline{\mathbf{Q_o}}$

Part B (12 points) Now combine these toggle cells to build a **divide by six** 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*.



Problem 2 (2 parts, 18 points) Datapath Elements

Part A (9 points) Consider the following input and output values for a shift operation. Determine the shift *type* and *amount* required to achieve the listed transformation. I/Os are in hexadecimal.

Input Value	Output Value	Shift Type	Shift Amount (signed decimal value)
87654321	43210000	arith / logical	-16 bits
87654321	76543218	rotate	-4 or +28 bits
87654321	00008765	logical	+16 bits

Part B (9 points) Consider the following input and output values for a logical operation. Determine the *logical function* and *function code* (in hexadecimal) required for the operation.

X Input	Y Input	Output	Logical Function	Function Code
87654321	0000FFFF	00004321	AND	8
87654321	0000FFFF	8765FFFF	OR	E
87654321	0000FFFF	8765BCDE	XOR	6

Problem 3 (3 parts, 30 points) Memory Systems

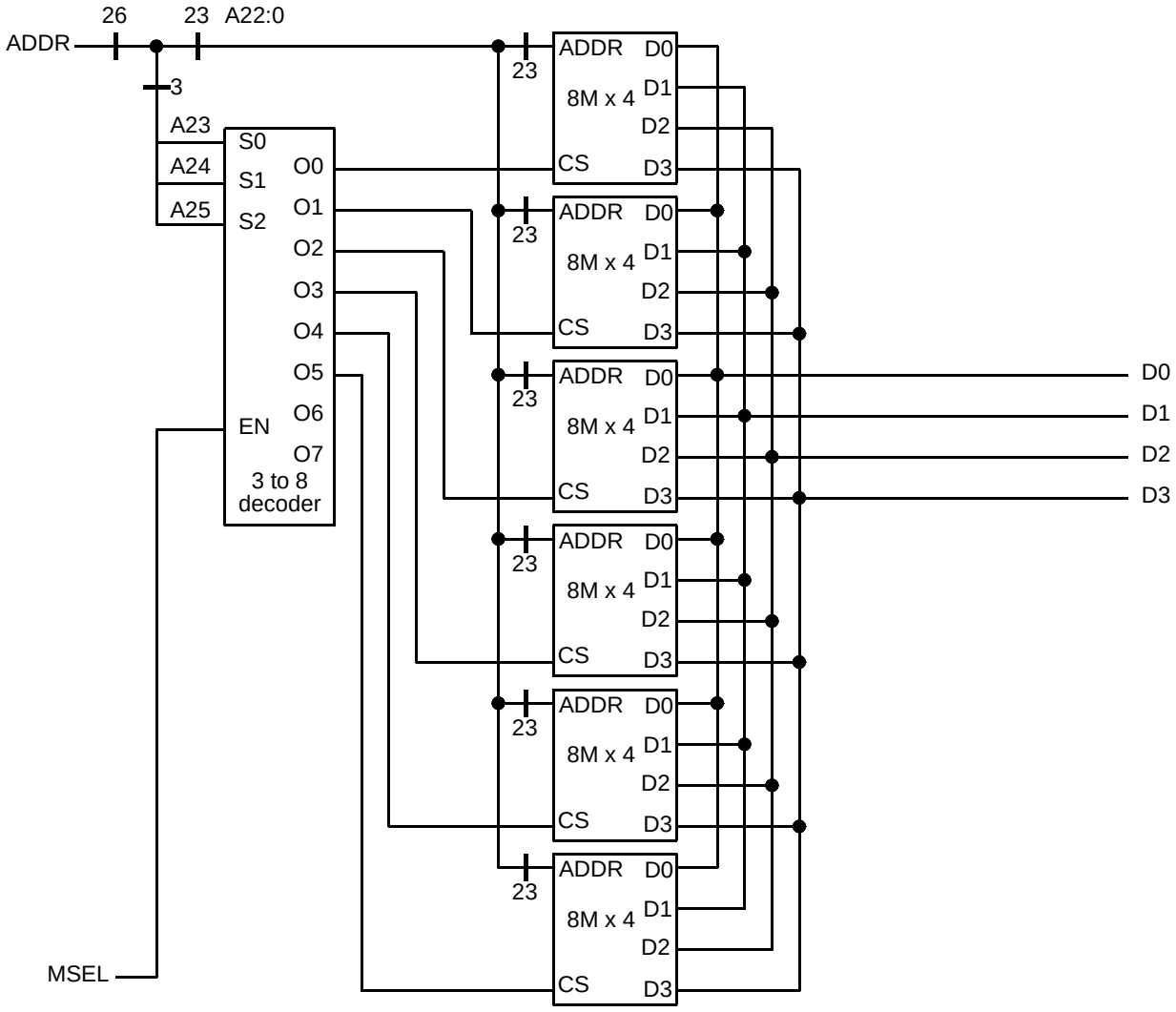
Part A (10 points) Consider a DRAM chip organized as **512 million addresses** of **eight bit words**. Assume both the DRAM cell and the DRAM chip is square. The column number and offset concatenate to form the memory address. Using the organization approach discussed in class, answer the following questions about the chip. **Express all answers in decimal**.

number of columns	$512M \times 8 = 2^{29} \times 2^3 = 2^{32}$; $2^{16} = 64K$
column decoder required (*n* to *m*)	16 to 64K
type of mux required (*n* to *m*)	64K/8 = 8K; 8K to 1
number of muxes required	8
number of address lines in column number	16
number of address lines in column offset	13

Part B (10 points) Consider a **4 Gbyte** memory system with **256 million addresses** of **16 byte words** using DRAM chips organized as **64 million** addresses by **16 bit words.**

word address lines for memory system	$\log_2(256M) = 28$
chips needed in one bank	16 bytes / 16 bits = 8 chips
banks for memory system	256M / 64M = 4 banks
memory decoder required (*n* to *m*)	2 to 4
DRAM chips required	8 x 4 = 32 chips

Part C (10 points) Design a 48M address x 4 bit memory system with six 8M address x 4 bit memory chips. **Label all busses and indicate bit width**. Assume R/W is connected and not shown here. Use a decoder if necessary. Place a star on the chip(s) that contain address 26,000,000.



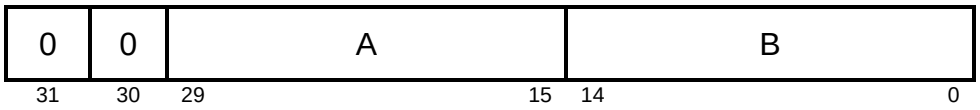
Problem 4 (2 parts, 28 points) Microcode

Using the supplied datapath, write microcode fragments to accomplish the following procedures. Express all values in hexadecimal notation. Use 'X' when a value is don't cared. For maximum credit, complete the description field.

Part A (14 points) $R_7 = \frac{3 \times R_5}{16} - 256 \times R_6$ **Modify only R_5, R_6 and R_7.**

#	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	description
1	5	5	7	1	0	X	1	0	0	X	0	X	0	0	X	0	R7 ← R5 + R5
2	7	7	5	1	0	X	1	0	0	X	0	X	0	0	X	0	R7 ← R7 + R5
3	7	X	7	1	1	4	0	X	0	X	1	1	0	0	X	0	R7 ← R7 >> 4
4	6	X	6	1	1	FFF8	0	X	0	X	1	1	0	0	X	0	R6 ← R6 << 8
5	7	6	7	1	0	X	1	1	0	X	0	X	0	0	X	0	R7 ← R7 + R6
6																	
7																	

Part B (14 points) Write a microcode sequence that loads a 32 bit word from memory location 0x4000, unpacks and averages two 15 bit unsigned values (A and B), and then stores the result back to memory location 0x4000. Assume the most significant two bits of the register are zero. **Modify only R_1, R_2, and R_3.**



#	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	description
1	X	X	1	1	1	4000	0	X	1	C	0	X	0	0	X	0	R1 ← 0x4000
2	1	X	2	1	0	X	0	X	0	X	0	X	1	0	1	1	R2 ← mem[R1]
3	2	X	3	1	1	7FFF	0	X	1	8	0	X	0	0	X	0	R3 ← R2 & 0x7FFF
4	2	X	2	1	1	F	0	X	0	X	1	0	0	0	X	0	R2 ← R2 >> 15
5	2	3	2	1	0	X	1	0	0	X	0	X	0	0	X	0	R2 ← R2 + R3
6	2	X	2	1	1	1	0	X	0	X	1	1	0	0	X	0	R2 ← R2 >> 1
7	1	2	X	0	0	X	0	X	0	X	0	X	0	1	0	1	mem[R1] ← R2
8																	