THE UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL ENGINEERING
ELEC 464 : Microcomputer System Design (Fall 1995)

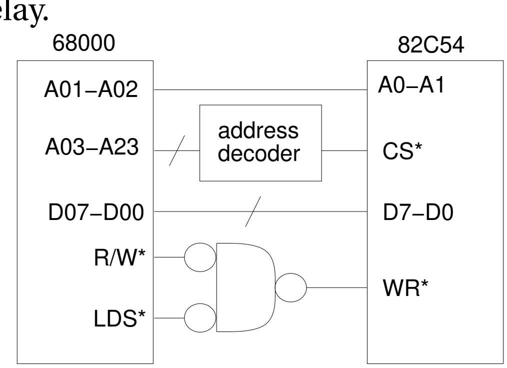
MID-TERM EXAMINATION
10:30 am – 11:30 am
October 31, 1995

This exam has three (3) questions. The marks for each question are as indicated. There are a total of 60 marks. Answer all questions. You may answer the questions in any order. Books, notes and calculators are allowed.

A copy of the MC68000 Programmer's Reference Guide (Appendix E from the text) and the MC68000 write cycle timing diagram and specifications (pages 182 and 183 in the text) will be provided on request.

Question 1 (19 marks)

The appendix gives a partial data sheet for an Intel 82C54 programmable interval timer chip. An 82C54 is interfaced to an 8 MHz Motorola 68000 CPU using the circuit shown below. You may assume the address decoder and the NAND gate have zero propagation delay.



For each 82C54 write-cycle timing requirement given in the table below:

- obtain an expression for the requirement in terms of 68000 timing specifications,
- compute the amount (in nanoseconds) by which the requirement is exceeded or is not met
- state whether the requirement is met or not

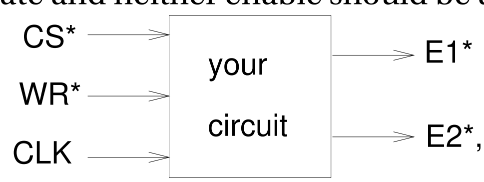
Symbol	Description	minimum	units
t_{SW}	CS* stable before WR* low	50	ns
t_{WA}	address hold after WR* high	10	ns
t_{WW}	WR* pulse width	80	ns
t_{DW}	data setup before WR* high	80	ns
t_{WD}	data hold after WR* high	30	ns

Question 2 (21 marks)

This question asks you to design a device that allows a CPU to read from two different input ports that share the same memory location. You circuit must alternate between the two ports so that successive CPU reads cause the value to be read from alternating devices. For example, the first time the CPU reads from this i/o memory location it will obtain the value on port 1. The next read from that location will obtain the value on port 2. The next read will obtain the value on port 1 and so on.

An additional feature is that the CPU can reset the circuit by doing a write to the same memory location that it reads the data from. After this reset operation the next read operation will read from port 1. The reset does not transfer data back to the input port.

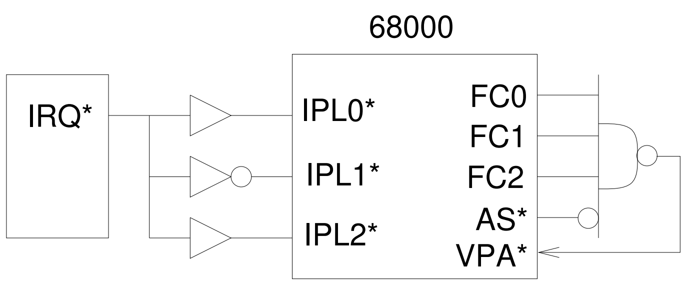
The inputs to your circuit are chip select CS* and WR*. The outputs are two enables (E1* and E2*) that control two i/o port chips. The enables should be asserted (one enable at a time) when CS* is asserted and WR* is not asserted. The device should be reset as described above and neither enable asserted when both CS* and WR* are asserted. You can also assume a clock signal (CLK) that will have exactly one rising edge during the time that CS* is asserted. If CS* is not asserted, the circuit should not change state and neither enable should be asserted.



Give the truth table for the device showing the values of the inputs, outputs and any state variables required. Write the CUPL `Pin` and "assignment" statements required to implement the device using an 18CV8 PAL. Your equations should define the values of the outputs (E1* and E2*) and the state variables (if any) required to implement the circuit.

Question 3 (20 marks)

A peripheral chip is connected to a Motorola 68000 CPU as shown in the following schematic.



(a) Answer the following questions:

1. When the IRQ* pin on the peripheral is asserted, what level of interrupt will be generated?
2. What signal levels (H or L) must appear on FC0, FC1, FC2, and AS* to assert VPA*?
3. What type of bus cycle would be taking place when VPA* is asserted?
4. At what address will the interrupt vector be found?

(b) Write an exception handler for this device. Assume the value of the exception vector is \$0000 2000. Your code:

- must include an `ORG` statement showing the location where the code starts
- should add 1 to the 4-byte value at memory location \$1000
- should then read one byte from memory location \$FE0000 and write it to the address given by memory location \$1000 (i.e. it should *not* write the byte to memory location \$1000 but to the memory location stored at \$1000)
- must be transparent to the program that was running when the interrupt occurred
- must return control properly to the program that was interrupted
- may assume it has at least 8 bytes available on the supervisor stack when the interrupt occurred