CSSE2010 / CSSE7201 – Introduction to Computer Systems
Answers to Exercises – Week Five
ALU, Memory

Answers

Some of the questions below are taken from or based on questions in Tanenbaum, Structured Computer Organisation, 5th edition.

1. A small memory chip has 4096 cells each of 4 bits. How many address and data lines does it need? How many flip-flops are contained within this chip?
   It needs $\log_2(4096)$ = 12 address lines (2^{12} is 4096), and 4 data lines for the 4-bit cells. (Data lines for input and output are usually shared.) It needs 4096x4 = 16 384 flip-flops.

2. Which of the following are possible memory organisations? Which are reasonable? Explain.
   (a) 10-bit address, 1024 cells, 8-bit cell size
   2^{10} is 1024 so it is certainly possible to address 1024 cells. An 8-bit cell size is fairly normal. This system is both possible and reasonable.
   (b) 10-bit address, 1024 cells, 12-bit cell size
   This is old-fashioned (12-bit words aren't popular now), but it is certainly possible and reasonable
   (c) 9-bit address, 1024 cells, 10-bit cell size
   Impossible – 9 address bits will only allow us to address 2^9 = 512 cells, not 1024 cells.
   (d) 11-bit address, 1024 cells, 10-bit cell size
   This is certainly possible – but wastes address space. We could address up to 2048 cells with 11 address bits.
   (e) 10-bit address, 10 cells, 1024-bit cell size
   Possible, but wastes address space. 1024-bit cell size is unreasonable (no current machine uses anything like 1024 bits per cell).
   (f) 1024-bit address, 10 cells, 10-bit cell size
   Having 1024 address bits is unreasonable; this could address something like 10^{300} cells, and there are only 10 cells.

3. Sometimes it is useful for an 8-bit ALU such as that presented in the week 5 lecture (made up of eight 1-bit ALUs) to generate the constant –1 as output. Give two different ways this can be done. For each way, specify the values of the six control signals (F_1, F_0, ENA, ENB, INVA, INC).
   Remember that -1 is represented in two's complement binary as all ones – so we need each bit slice of the ALU to output a 1. Here are some possible solutions (X represents "don't cares" – i.e. could be a 0 or a 1):

Description	A's	B's	F1	F0	ENA	ENB	INVA	INC
Negate ENB so that B goes to 0. Then choose the function NOT(B)	X	X	**0**	**1**	X	**0**	X	X
			NOT(B)					
Negate ENA and ENB so that A and B both go to 0. Set INVA and choose OR.	X	X	**1**	**0**	**0**	**0**	**1**	X
			OR(A,B)					
Negate ENA and ENB so that both go to 0. Set INVA and choose ADD.	X	X	**1**	**1**	**0**	**0**	**1**	**0**
			ADD(A,B)					

4. For the ALU shown in lectures, give three different combinations of the six control signals (F_1, F_0, ENA, ENB, INVA, INC) which will result in the output being B. (Hint: one answer is given in the table in the lecture notes. Which of the other functions, besides OR can be utilised to produce B?)
   The answer given in the lecture notes is
   F_0=0, F_1=1, ENA=0, ENB=1, INVA=0, INC=0

This is the OR function with A disabled, i.e., the output is B or 0 = B. (INC could be 1 instead of 0 since its value is irrelevant when the OR function is used.)

A second option is to use the AND function, noting that B and 1 = B. To get a 1 on the other input of the AND gate, we disable the A input (i.e. always 0) and invert it (i.e. always 1):
F_0=0, F_1=0, ENA=0, ENB=1, INVA=1, INC=0 (or 1)

A third option is to use the PLUS function, noting that B plus 0 = B. To get a 0 on the other input of the adder, we disable the A input (with no inversion). We also must ensure that that the carry-in input is 0:
F_0=1, F_1=1, ENA=0, ENB=1, INVA=0, INC=0

There are other methods of generating B on the output if we assume particular values of A, e.g. if the A input is guaranteed to be 0, we don't need to disable the A input in the solutions above.

Additional Exercises

The exercise below goes beyond the learning objectives of the course but may help you develop a greater understanding of the course material.

5. A 16-bit ALU is built up of 16 1-bit ALUs, each having an add time of 10ns (nanosecond, 10^{-9} sec). If there is an additional 1ns delay for carry propagation time from one ALU to the next, how long does it take for the result of a 16-bit add to appear?
   There are 16 ALU delays and 15 propagation delays $= 16 \times 10 + 15 \times 1 = 175ns$