18-742 Advanced Computer Architecture

Test 2 ---
November 19, 1997

SOLUTION

Problem 1: Physical Memory

A computer system uses 5-way interleaved memory, with interleaving performed on an 8-byte word basis (incrementing the address by 8 goes to the next higher bank, with wrap-around to bank 0). There is no skew factor used to address the memory (skew=0). (Note: we'll use the term "bank" to refer to what is sometimes called a "module" in interleaved memory).

(6 points)
a) Given the physical memory address for an 8-byte word, write the equations to compute the bank number (starting at bank 0) and the byte address of a word within a bank. You will recall from the lectures that for skewed addressing on *word-addressed* interleaved memory:

Word address within bank = address / #banks

Bank # = ((address/#banks) + (address * skew)) MOD #banks

The byte-addressed equations you are being asked to write need only be valid on 8-byte aligned accesses, and will use integer arithmetic for implicit truncation of intermediate results. They should contain no redundant terms.

Byte address = word address * 8 = 8 * ((address/8) /#banks)

Bank# = ((address / 8) / #banks) MOD #banks = (address / 8) mod #banks

(9 points)
b) You have a C program with an array definition of:

```
long x[4][4];
```

This machine uses 64-bit longs. Assume that x[0][0] is at address 0 and fill in the blanks in the picture below that depicts how the array elements map into the memory banks.

	BANK 0	BANK 1	BANK 2	BANK 3	BANK 4
0	x[0][0]	x[0][1]	x[0][2]	x[0][3]	x[1][0]
8	x[1][1]	x[1][2]	x[1][3]	x[2][0]	x[2][1]
16	x[2][2]	x[2][3]	x[3][0]	x[3][1]	x[3][2]
24	x[3][3]				

Problem 2

A vector computer has the following characteristics:

- 3 memory pipes (two VAGs for loading, one VAG for storing) . Memory accesses complete in-order **on a per-VAG** basis (so, VAGs can get ahead of or behind each other).
- Two buses that can each carry one 8-byte word per clock tick (all numbers are in 8-byte words) and operate at 50 MHz
- No cache is involved
- There are 5 interleaved memory banks addressed sequentially at 8-byte word boundaries (zero skew factor).
- There are as many VRF registers of infinite length as necessary (to simplify problem); up to three data transfers can be completed at the VRF each clock cycle with 1 clock cycle latency to access the VRF
- VDS can transfer three words concurrently on each clock cycle from any point to any other points (3 inputs routed to as many outputs as desired on each clock), and presents inputs to both sides of the adder on the same clock cycle (slightly different than the class example, which assumed an internal buffer register for one input).
- Bus reads are given priority over writes (so writes wait if there is any read pending)
- Vector instructions are dispatched by a scalar processor at one clock per instruction.
- Vector chaining as described in class is supported and should be used in your solutions. All resources are pipelined except for memory banks.

Latency	Clock ticks
Vector instruction dispatch	1
VAG setup	1
Address bus latency	2
DRAM read latency (cycle time is 1 clock longer than this)	3
Bus latency to return read data	2
VDS delay	1
Adder delay (starting when both operands available)	2
VDS delay	1
Data write latency for bus	2
Data written in to DRAM (cycle time 1 clock longer than this)	2

A 4-element vector addition takes 4 clock cycles to issue ("vload", "vload", "vadd", "vstore"). Assume that all three vectors have their first element mapping into memory bank 0 and are accessed with a stride of 1 word. Assume that data arrives at the DRAM from the bus during the RAS cycle. Complete the table for a 4-element vector addition of z[i] = x[i] + y[i]. Note that some columns are missing and some are filled in to reduce the amount of work needed for this exercise. Each entry in the table should be for the clock cycle when an operation starts.

(11 points) a) fill in the memory bank columns
(6 points) b) fill in the VDS column
(6 points) c) fill in the VRF column

Beat	ADR	BANK 0	BANK 1	BANK 2	BANK 3	BANK 4	BUS	VDS	ADD	VRF
3	X[0]									
4	X[1]									
5	X[2]	rX[0]								
6	X[3]	cX[0]	rX[1]							
7	Y[0]	X[0]	cX[1]	rX[2]						
8	Y[1]	cyc	X[1]	cX[2]	rX[3]		X[0]			
9	Y[2]	rY[0]	cyc	X[2]	cX[3]		X[1]			
10	Y[3]	cY[0]	rY[1]	cyc	X[3]		X[2]	X[0]		
11		Y[0]	cY[1]	rY[2]	cyc		X[3]	X[1]		X[0]
12		cyc	Y[1]	cY[2]	rY[3]		Y[0]	X[2]		X[1]
13			cyc	Y[2]	cY[3]		Y[1]	X[3]		X[2], X[0]
14				cyc	Y[3]		Y[2]	Y[0], X[0]		X[3], X[1]
15					cyc		Y[3]	Y[1], X[1]	X[0], Y[0]	Y[0], X[2]
16								Y[2], X[2]	X[1], Y[1]	Y[1], X[3]
17								Y[3], X[3], Z[0]	X[2], Y[2]	Y[2]
18	Z[0]						Z[0]	Z[1]	X[3], Y[3]	Y[3], Z[0]
19	Z[1]						Z[1]	Z[2]		Z[1]
20	Z[2]	rZ[0]					Z[2]	Z[3]		Z[2]
21	Z[3]	Z[0]	rZ[1]				Z[3]			Z[3]
22		cyc	Z[1]	rZ[2]						
23			cyc	Z[2]	rZ[3]					
24				cyc	Z[3]					
25					cyc					

Problem 3

You have a vector computer with the following bandwidths available for 8-byte data values:

- 5 memory banks at 120 Million B/sec each
- 1 read-only bus and 1 write-only bus at 480 Million B/sec each
- a VDS with three concurrent connections at 480 Million B/sec each
- a pipelined vector adder at 60 MFLOPS; 3 clock latency; 1 result per clock throughput
- a VRF with three ports at 480 Million B/sec each

The following table gives the latency for vector additions at different vector sizes:

Vector Length	Total Latency (clocks)	Achieved MFLOPS
1	29	2.07
2	31	3.87
3	33	5.45
4	35	6.86
5	37	8.11
6	39	9.23
7	41	10.24
8	43	11.16
9	45	12.00

(10 points)
a) Fill in the Achieved MFLOPS column for this table. (Assume no other operations are concurrent with the addition.) Write the computation for vector length 3 below:

MFLOPS = one ADD per total latency clocks
3 adds / 33 clocks = (1 add / 11 clocks) * 60 MHz = 5.45 MFLOPS

(6 points)
b) If the scalar floating point coprocessor can compute a vector sum at the rate of 8 clocks per result, what is N_v on this architecture? Use a linear interpolation between the nearest pair of data points in the table.

8 clocks / result = 7.50 MFLOPS for the scalar coprocessor

Using a linear interpolation:

X = 4 + ((7.50 - 6.86) / (8.11 - 6.86)) = N_v is 4.51

(12 points)
c) For vector addition, what is $R_{infinity}$ on this architecture?

60 MFLOPS * 8 bytes = 480 Million B/sec per data stream

Vector addition uses 3 operand ("data") streams; 480*3 = 1440 Million B/sec

Memory -- supports 1.25 data streams
Bus -- 1 data stream read; 1 data stream write
VDS -- 3 data streams
VRF -- 3 data streams
Adder -- 3 data streams (2 in, 1 out)

Therefore, $R_{infinity}$ is limited by memory bandwidth at
(1.25 data items per clock / 3 data items per result) * 60 MHz = 25 MFLOPS

Problem 4

You are designing a computer system bus under the following constraints:

- Each connection costs $.10 for each socket *contact* ("pin") and $.02 for an edge connector *contact* ("pin") on a parallel bus at 60 MHz with a maximum of 10 card slots.
- Your initial design multiplexes 32 address lines with 32 data lines for a total of 60 bus signal lines (the address and data lines share the same 32 out of 60 lines).
- A bus transaction takes 4 clocks to complete (*i.e.*, 4 clocks per 32-bit word transfer)

You wish to increase system performance by using transfers of 128 bits. Assume that data can be transferred at the rate of one data chunk per bus clock after the first data chunk is returned with latency 4 clocks including addressing delay. What is the total fully populated (10 card) system cost, latency (in ns), and sustained bandwidth assuming full utilization for the following configurations at 128-bit data transfers?

(10 points)
a) multiplexed address/data lines 32 bit data lines (Cost, latency, sustained bandwidth)

Cost = $.12 * 60 * 10 = $72.00

Latency = 4 clocks + 1 + 1 + 1 clock => 7 clocks / 60 MHz =117 ns

Bandwidth = 128-bit word / 7 clocks = (16 bytes / 7 clocks) * 60 MHz
= 137 million B/sec = 130.8 MB/sec

(12 points)
b) non-multiplexed address/data lines; 128 bit data lines (Cost, latency, sustained bandwidth). Assume all control lines remain unchanged.

Cost = $.12 * (60+128) * 10 = $225.60

Latency = 4 clocks => 4 clocks / 60 MHz = 67 ns

Bandwidth = 128-bit word / 4 clocks = (16 bytes / 4 clocks) * 60 MHz
= 240 million B/sec = 228.9 MB/sec

Problem 5

(12 points)
A Hamming SEC code uses 4 data bits and 3 check bits with the following syndrome $[S_1\ S_2\ S_3]$:

$S_1 = d_1 \oplus d_2 \oplus d_3 \oplus c_1$
$S_2 = d_1 \oplus d_3 \oplus d_4 \oplus c_2$
$S_3 = d_2 \oplus d_3 \oplus d_4 \oplus c_3$

given that:
$d_1=1$
$d_2=1$
$d_3=0$
$d_4=0$
what are the check bit values?

For no errors, the Syndrome bits all equal zero, so the check bits equal the xor'ed data bits

$c_1 = d_1 \oplus d_2 \oplus d_3 = 1 \oplus 1 \oplus 0 = 0$

$c_2 = d_1 \oplus d_3 \oplus d_4 = 1 \oplus 0 \oplus 0 = 1$

$c_3 = d_2 \oplus d_3 \oplus d_4 = 1 \oplus 0 \oplus 0 = 1$