Autumn Semester 2011-2012 (2 hours)

Answers to EEE411/6031 Advanced Computer Architectures 4, Questions 1…4

1. *A pipelined system is as shown in Figure 1:*

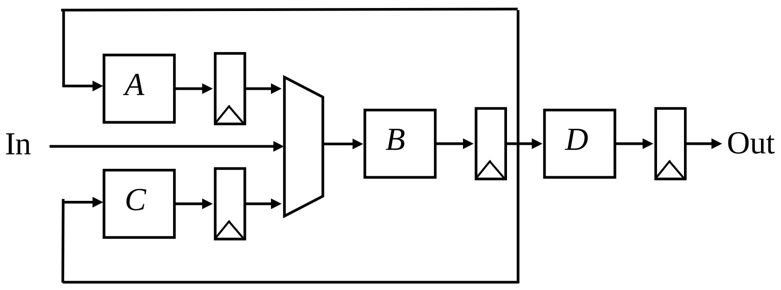


Figure 1: Pipelined System

and it is used to execute the sequence of operations:

Out = D(B(C(B(A(B(In))))))

a. *Draw the reservation table for the pipeline.*

time	**A**	**B**	**C**	**D**
0		D1		
	D1	D2		
	D2	D1		
		D2	D1	
		D1	D2	
		D2		D1
		D3		D2
	D3	D4		
	D4	D3		
		D4	D3	
		D3	D4	
		D4		D3
				D4

(8)

b. *Calculate the data throughput and utilisation of each processing block (assuming a greedy strategy).*

Throughput is 2/6 = 1/3 datum per clock cycle.

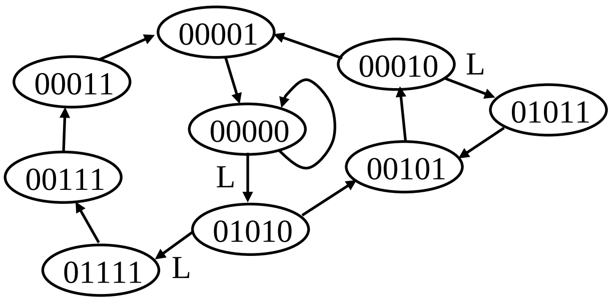
Utilisation A=33.3%, B=100%, C=33.3%, D=33.3% **(2)**

c. *Show that the Collision Vector for the operation is 01010.*

time	**A**	**B**	**C**	**D**
0		D1		
	D1			
		D1	3	
		1	D1	
		D1		
				D1

The gaps between use and reuse of B (the only one reused) are 1 and 3 and so bits 1 and 3 are set only i.e. 01010. **(2)**

d. *Using the Collision Vector, draw the State Table and, hence, show that there is another way to load data other than the greedy strategy.*

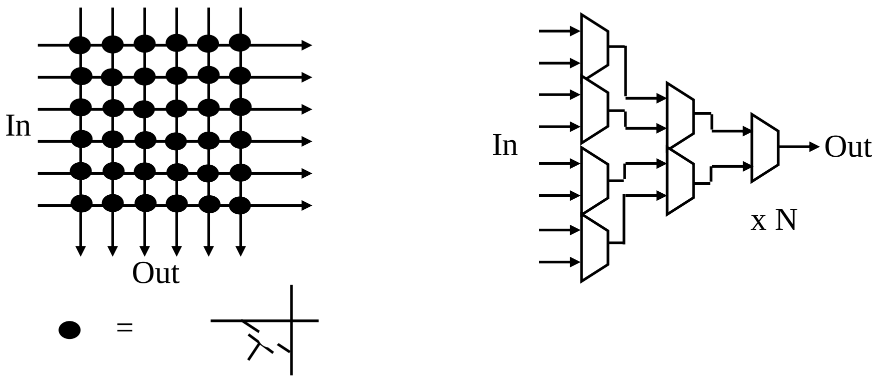


The greedy strategy is the left hand loop but by not loading after the initial load the loop on the right-hand side is reached. **(6)**

e. *Is this alternative method better or worse than the greedy strategy? Give a reason to support your answer.*

The throughput in either case is the same, 1/3 of a datum per clock cycle. However, the right-hand loop is much more regular i.e. a load every 3rd clock cycle as opposed to two loads and then a gap of four clock cycles **(2)**

2. a. i) *Draw two schematics to show two possible ways of making a cross-point switch.*



(4)

ii) *Of these two approaches to making a cross point switch, identify which is the most flexible (in terms of expansion, for example), why this is, and what its disadvantages are.*

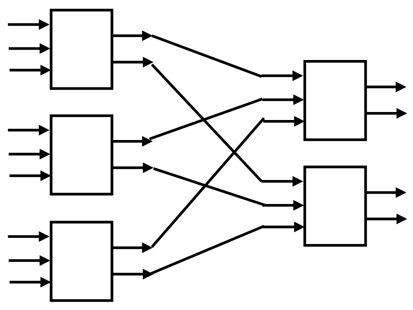
The LH, bus-based CPS can be extended by tiling the units, extending the buses horizontally and/or vertically to provide more inputs or outputs. Conversely, the RH multiplexer can be replicated to increase the number of outputs but more multiplexers must be added at the input side to increase the number of inputs. Additionally, the bus-based CPS can be made bi-directional. Consequently, it is more flexible. However, as the length of the bus and the loading is increased, the bus becomes slower and will consume more energy per transition. Consequently, this structure is not favoured for VLSI solutions. **(4)**

b. *A set of 9 processors must be connected to 4 disjoint memory blocks and it is decided that a Banyan network will be used.*

i) *What must the parameters of the Banyan network be (s, f, and l)*

A Banyan network has s^{l+1}=9 inputs and f^{l+1}=4 outputs and so *s* must equal 3 and *l* equals 1 whilst *f* must equal 2. **(3)**

ii) *Draw a schematic for this Banyan network.*



(6)

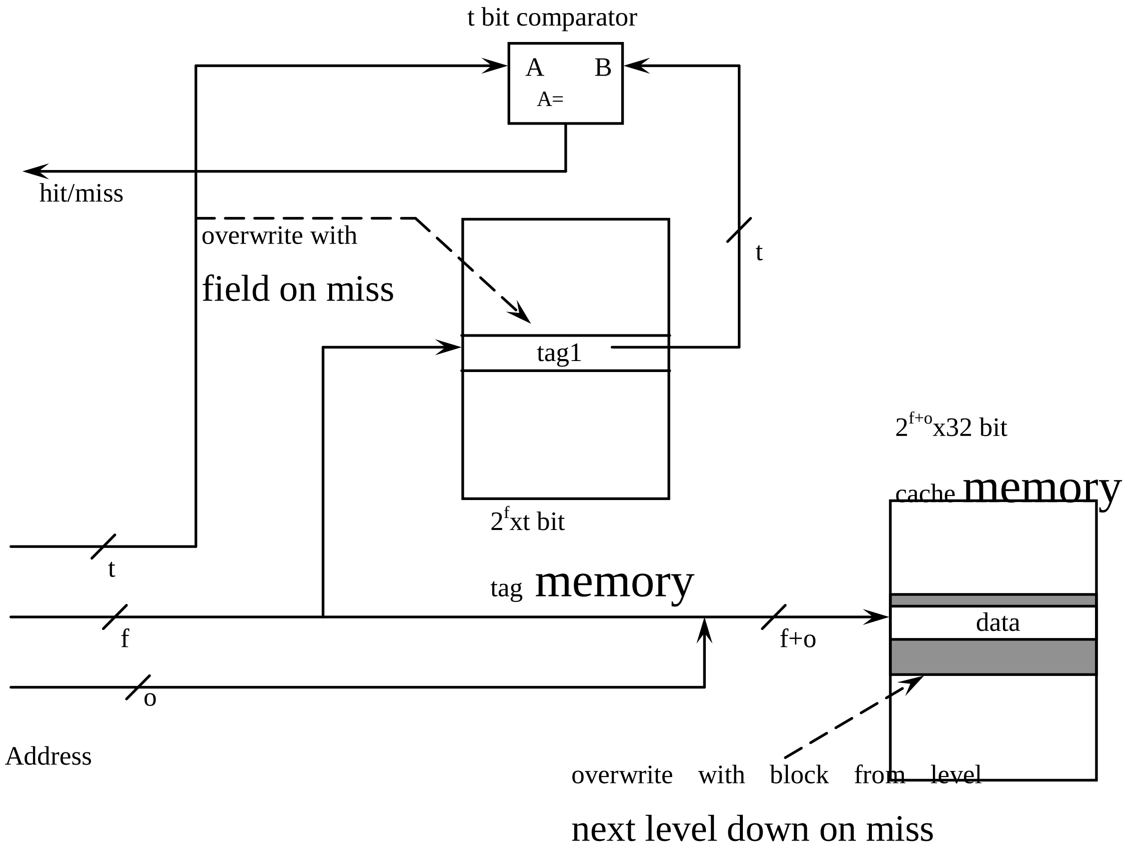
iii) *Calculate the efficiency of this network (in hardware terms) compared to a cross-point switch doing the same job.*

The complexity of a CPS with 9 inputs and 4 outputs is *k*x9x4 = 36*k*. The Banyan is composed of 3x2 CPS and the complexity of each of these is 6*k*. There are 5 of these CPS in the switch and so the overall cost is 30*k*. Consequently, the Banyan occupies 30/36 of the area of the corresponding CPS.

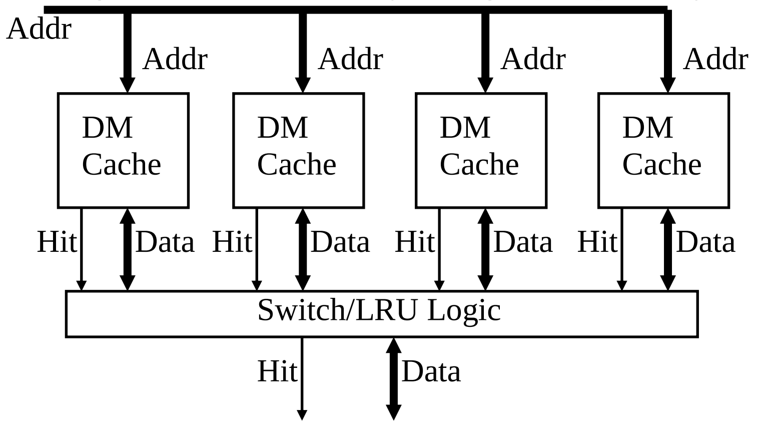
(3)

3. a. i) *Draw a schematic diagram for a 4-way set associative cache.*

A direct mapped cache is the basic block used:



A 4-way SA cache is formed from 4 of these DM caches in parallel with an additional set of logic to deal with memory management (probably LRU).



(5)

ii) *What would the approach be and the costs (in memory terms) of implementing an LRU memory policy on a 4-way set associative cache.*

A 6 bit memory with an entry for each set in the cache is needed. With 4 lines in each set, a 2 bit descriptor is required for each line in the set and to implement LRU a queue of the descriptors is needed to decide which is th least recently used. Nominally, this queue has 4 descriptors on it but only 3 are needed (the 4th is implied – hence 6 bits per set. When a hit occurs, the entry in the memory is logically combined with the line that hit to generate a new 6 bit value. **(3)**

b. *Why do we use a memory hierarchy in practical systems?*

The cost per bit and density of memory of the speed necessary to meet the speed requirements of a practical processor and size of memory required by a practical system makes its use prohibitively expensive. By using a small amount of such memory (as a cache) underpinned by a larger, cheaper and slower memory, and relying on locality, the speed of the memory system resembles the upper layer but the volume is determined by the lower layer. This argument can be extended to multiple layers where speed increases up the hierarchy and volume increases down the hierarchy. **(2)**

c. A particular processing system with an on-chip clock frequency of 2GHz utilises a single write-back cache level (on-chip) with the main memory implemented using synchronous DRAM (you can assume that the time taken to read or write a cache line to the SDRAM is 32ns). Empirical studies identify that, on average:

- the probability of a hit is 0.96 for a read and 0.98 for a write;
- the proportion of memory accesses that are reads is 0.7;
- the proportion of dirty blocks is 0.4;
- the access time to the cache (on hit) is 1 clock cycle for a read and 2 clock cycles for a write.
- All accesses to the SDRAM are hits.

i) *Estimate the average access time achieved by the processor.*

Splitting reads and writes:

The overall cost of a read is 1 clock cycle (0.5ns) on a hit (96% of the time) but on a miss (4% of the time) the cost is the time taken to fetch the missing block (32ns) but in 40% of the cases, the replaced block also needs to be written back to the SDRAM. Consequently, read-access = 0.5*0.96 + 0.04*(32 + 0.4*32)

The overall cost of a write is 2 clock cycles (1ns) on a hit (98% of the time) but on a miss (2% of the time) the cost is the time taken to fetch the missing block (32ns) but in 40% of the cases, the replaced block also needs to be written back to the SDRAM. Consequently, write-access = 1*0.98 + 0.02*(32 + 0.4*32)

Read accesses occur 70% of the time and writes 30% of the time. Consequently:

memory-access = 0.7*read-access + 0.3*write-access =
0.7*(0.5*0.96+0.04*32*1.4) + 0.3*(0.98+0.02*32*1.4) = 2.15ns **(8)**

ii) *How does this access time compare against the best possible access time that could be achieved*

The best possible access time would be achieved if there were no misses. In this case, the access time would be 0.7*1+0.4*2 = 1.3ns i.e. it is 60% longer than the best possible access time. **(2)**

4. **a.** *Describe the organisation of a square-connected SIMD processor.*

The array is made up of a set of processing elements consisting (usually) of a datapath capable of processing a few bits (1,2, or 4) with a small amount of local memory (64-1024 bits). Each PE is connected to its local neighbours in a N, S, E, and W direction allowing data to be broadcast from a central PE and broadcast data to be read by neighbouring PEs. The PEs are all controlled centrally and constrained to execute the same instructions at the same time. However, this instruction is executed locally i.e. using a PE's local memory and connections. **(4)**

b. *A SIMD processor uses a corner-turn buffer (connected to the West side of the array) to allow data to be loaded into the array. The CTB is byte-wide with the LSB of the data at the tail-end of the serial registers making up the CTB.*

Assuming that the SIMD array has 256 columns of Processing Elements (PE) and supports the following instructions:

```
WAITCTB      ; wait until the CTB is full and ready to transfer data
SHIFTCTB     ; shift the data, serially, by one place in the CTB
BCAST    x   ; broadcast bit address x to N,S,E, and W
INy      x   ; input from direction y (N,S,E, or W) to bit address x
```

i) *write a short program that will shift a column of data from the CTB into the array of PEs so that the LSB of the data is in bit address 0 of a PE's memory and the MSB of the data is in bit address 7 of the PE's memory.*

The first data read from the W input would be the MSB of the data byte being read and each time the SHIFTCTB instruction is executed the next bit would be presented.

```
WAITCTB   ; this allows the array to synchronise with the filling of the CTB
INW     7 ; put the MSB into address 7
SHIFTCTB  ; get the next bit
INW     6 ; put the next bit into address 6
SHIFTCTB  ; get the next bit
INW     5 ; put the next bit into address 5
SHIFTCTB  ; get the next bit
INW     4 ; put the next bit into address 4
SHIFTCTB  ; get the next bit
INW     3 ; put the next bit into address 3
SHIFTCTB  ; get the next bit
INW     2 ; put the next bit into address 2
SHIFTCTB  ; get the next bit
INW     1 ; put the next bit into address 1
SHIFTCTB  ; get the LSB
INW     0 ; put the LSB into address 0
```
(6)

ii) *How would this short program be amended/repeated to load a full array (256 columns of data) into the array of PEs? State any assumptions that you make.*

This program needs to be amended such that when the data is read in to the LH column, it is also copied to the next column – we'll make it a function.

```
function MOVECOLUMN
WAITCTB    ; this allows the array to synchronise with the filling of the CTB
BCAST    7 ; output the existing bit from address 7
INW      7 ; put the MSB into address 7
SHIFTCTB   ; get the next bit
BCAST    6 ; output the existing bit from address 6
INW      6 ; put the next bit into address 6
SHIFTCTB   ; get the next bit
BCAST    5 ; output the existing bit from address 5
INW      5 ; put the next bit into address 5
SHIFTCTB   ; get the next bit
BCAST    4 ; output the existing bit from address 4
INW      4 ; put the next bit into address 4
SHIFTCTB   ; get the next bit
BCAST    3 ; output the existing bit from address 3
INW      3 ; put the next bit into address 3
SHIFTCTB   ; get the next bit
BCAST    2 ; output the existing bit from address 2
INW      2 ; put the next bit into address 2
SHIFTCTB   ; get the next bit
BCAST    1 ; output the existing bit from address 1
INW      1 ; put the next bit into address 1
SHIFTCTB   ; get the LSB
BCAST    0 ; output the existing bit from address 0
INW      0 ; put the LSB into address 0
end
```

This function needs to be called 256 times:

```
for i=1 to 256
    MOVECOLUMN
```

(6)

c. *What is the disadvantage of connecting a CTB to the inter-PE connections (as in part* ***b.*** *above)? Is there a better solution?*

The disadvantage is that the array must execute all of the instructions to load the data and this time is not available for data processing. So, in the above example, there are 24*256 instructions (assuming that the cost of refilling the CTB is less) and once the data is loaded the instructions consumed in processing the data must be considerably greater of this overhead is burdensome. A better method is to set up a string of registers running from W to E connected to the CTB output allowing a plane of data bits to be loaded across the array whilst the array is still processing previous data. Once the registers are loaded then only a single instruction would be required to drop the data into the PE's memory. Clearly, this still requires the data processing time to be at least greater than the time taken to load these registers or the loading time will still dominate. **(4)**

NLS