Norwegian University of Science and Technology (NTNU)
DEPT. OF COMPUTER AND INFORMATION SCIENCE (IDI)

Contact person for questions regarding exam exercises:
Name: Lasse Natvig
Phone: 906 44 580

EXAM IN COURSE TDT4260 COMPUTER ARCHITECTURE

Monday 26th of May 2008
Time: 0900 – 1300

Solution sketches in blue text

Supporting materials: No handwritten or printed materials allowed, simple specified calculator is allowed.

By answering in short sentences it is easier to cover all exercises within the duration of the exam. The numbers in parenthesis indicate the maximum score for each exercise. We recommend that you start by reading through all the sub questions before answering each exercise.

The exam counts for 80% of the total evaluation in the course. Maximum score is therefore 80 points.

Exercise 1) Parallel Architecture (Max 25 points)

a) (Max 5 points) The *feature size* of integrated circuits is now often 65 nanometres or smaller, and it is still decreasing. Explain briefly how the number of transistors on a chip and the wire delay changes with shrinking feature size.
The number of transistors can be 4 times larger when the feature size is halved. However the wire delay does *not* improve (scales poorly). (The textbook page 17 gives more details, but we here ask for the main trends)

b) (Max 5 points) In a cache coherent multiprocessor, the concepts *migration* and *replication* of shared data items are central. Explain both concepts briefly and also how they influence on latency to access of shared data and the bandwidth demand on the shared memory.
Migration means that data move to a place closer to requesting/accessing unit. *Replication* just means storing several copies. Having a local copy in general means faster access, and it is harmelss to have several copies of read-only data. (Textbook page 207)

c) (Max 5 points) Explain briefly how a write buffer can be used in cache systems to increase performance. Explain also what "write merging" is in this context.
The main purpose of the write buffer is to temporarily store data that are evicted from the cache so new data can reuse the cache space as fast as possible, i.e. to avoid waiting for the latency of the memory one level further away from the processor. If more writes are to the same cache block (adress) these writes can be combined, resulting in a reduced traffic towards the next memory level. (Textbook page 300)
((Also slides 11-6-3)). // Retting: 3 poeng for skrive-buffer-forståelse og 2 for skrive-fletting.

d) (Max 5 points) Sketch a figure that shows how a hypercube with 16 nodes are built by combining two smaller hypercubes. Compare the hypercube-topology with the 2-dimensional mesh topology with respect to connectivity and node cost (number of links/ports per node).
(Figure E-14 c) A mesh has a fixed degree of connectivity and becomes slower in general when the number of nodes is increased, since the number of hops needed for reaching another node on average is increasing. For a hypercube it is the other way around, the connectivity increase for larger networks, so the communication time does not increase much, but the node cost does also increase. When going to a larger network, increasing the

dimension, every node must be extended with a new port, and this is a drawback when it comes to building computers using such networks.

e) (Max 5 points) When messages are sent between nodes in a multiprocessor two possible strategies are *source routing* and *distributed routing*. Explain the difference between these two.
For *source routing*, the entire routing path is precomputed by the source (possibly by table lookup—and placed in the packet header). This usually consists of the output port or ports supplied for each switch along the predetermined path from the source to the destination, (which can be stripped off by the routing control mechanism at each switch. An additional bit field can be included in the header to signify whether adaptive routing is allowed (i.e., that any one of the supplied output ports can be used).
For *distributed routing*, the routing information usually consists of the destination address. This is used by the routing control mechanism in each switch along the path to determine the next output port, either by computing it using a finite-state machine or by looking it up in a local routing table (i.e., forwarding table). (Textbook page E-48)

Exercise 2) Parallel processing (Max 15 points)

a) (Max 5 points) Explain briefly the main difference between a VLIW processor and a dynamically scheduled superscalar processor. Include the role of the compiler in your explanation.
Parallel execution of several operations is scheduled (analysed and planned) at compile time and assembled into very long/broad instructions for VLIW. (Such work done at compile time is often called static). In a dynamically scheduled superscalar processor dependency and resource analysis are done at run time (dynamically) to find opportunities to do operations in parallell. (Textbook page 114 -> and VLIW paper)

b) (Max 5 points) What function has the vector mask register in a vector processor?
If you want to update just some subset of the elements in a vector register, i.e. to implement
IF A[i] != 0 THEN A[i] = A[i] – B[i] for (i=0..n) in a simple way, this can be done by setting the vector mask register to 1 only for the elements with A[i] != 0. In this way, the vectorinstruction A = A - B can be performed without testing every element explicitly.

c) (Max 5 points) Explain briefly the principle of vector chaining in vector processors.
The execution of instructions using several/different functional and memory pipelines can be chained together directly or by using vector registers. The chaining forms one longer pipeline. (This is the technique of forwarding (used in processor, as in Tomasulos algorithm) extended to vector registers (Textbook F-23)
((Slides forel-9, slide 20)) – bør sjekkes

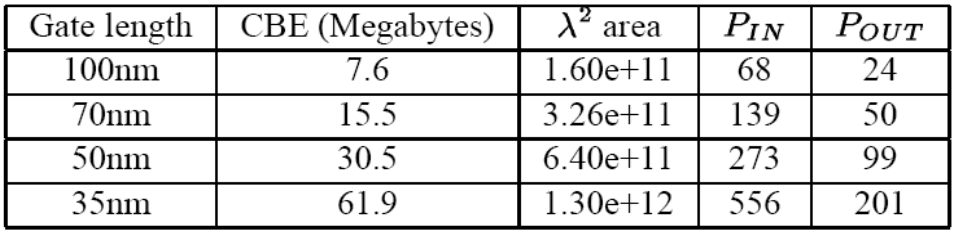
Exercise 3) Multicore processors (Max 20 points)

a) (Max 5 points) In the paper *Chip Multithreading: Opportunities and Challenges*, by Spracklen & Abraham is the concept Chip Multithreaded processor (CMT) described. The authors describe three generations of CMT processors. Describe each of these briefly. Make simple drawings if you like.
a) 1. generation: typically 2 cores pr. chip, every core is a traditional processor-core, no shared resources except the off-chip bandwidth. 2.generation: Shared L2 cache, but still traditional processor cores. 3. generation: as 2. gen., but the cores are now custom-made for being used in a CMP, and might also use simultaneous multithreading (SMT). (This description is a bit ”biased” and colored by the backgorund of the authors (in Sun Microsystems) that was involved in the design of Niagara 1 og 2 (T1))
// fig. 1 i artikkel, og slides // Var deloppgave mai 2007,

b) (Max 5 points) Outline the main architecture in SUN’s T1 (Niagara) multicore processor. Describe the placement of L1 and L2 cache, as well as how the L1 caches are kept coherent.
Fig 4.24 at page 250 in the textbook, that shows 8 cores, each with its own L1-cache (described in the text), 4 x L2 cache banks, each having a channel to external memory, 1x FPU unit, crossbar as interconnection. Coherence

is maintained by a catalog associated with each L2 cache. This knows which L1-caches that havbe a copy of data in the L2 cache.
// Læreboka side 249-250, også forelsning

c) (Max 6 points) In the paper *Exploring the Design Space of Future CMP's* the authors perform a design space exploration where several main architectural parameters are varied assuming a fixed total chip area of 400mm^2. Outline the approach by explaining the following figure;



Gate length	CBE (Megabytes)	λ^2 area	P_{IN}	P_{OUT}
100nm	7.6	1.60e+11	68	24
70nm	15.5	3.26e+11	139	50
50nm	30.5	6.40e+11	273	99
35nm	61.9	1.30e+12	556	201

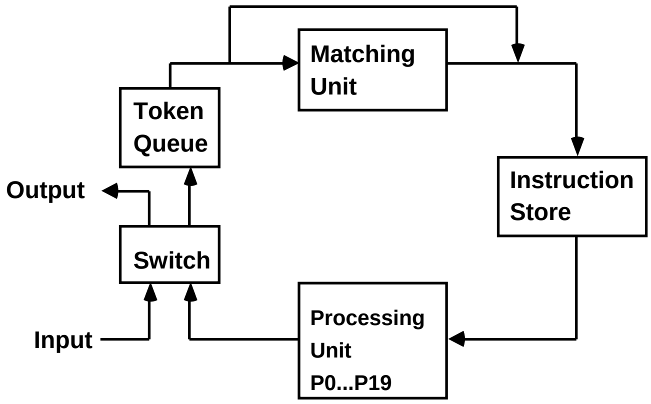
Table 3. Total Chip Area.

Technology independent area models – found empirically, – core area and cache area measured in cache byte equivalents (CBE). Study the relative costs in area versus the associated performance gains --- maximize performance per unit area for future technology generations. With smaller feature sizes, the available area for cache banks and processing cores increases. Table 3 displays die area in terms of the cache-byte-equivalents (CBE), and P_{IN} and P_{OUT} columns show how many of each type of processor with 32KB separate L1 instruction and data caches could be implemented on the chip if no L2 cache area were required. (P_{IN} is a simple in-order-execution processor, P_{OUT} is a larger out-of-order exec processor). And, for reference, Lambda-squared where lambda is equal to one half of the feature size. The primary goal of this paper is to determine the best balance between per-processor cache area, area consumed by different processor organizations, and the number of cores on a single die.
LF; Ny oppgave / Middels/vanskelig / foil 1-6, og 2-3

d) (Max 4 points) Explain the argument of the authors of the paper *Exploring the Design Space of Future CMP's* that we in the future may have chips with useless area on the chip that performs no other function than as a placeholder for pin area?
As applications become bandwidth bound, and global wire delays increase, an interesting scenario may arise. It is likely that monolithic caches cannot be grown past a certain point in 50 or 35nm technologies, since the wire delays will make them too slow. It is also likely that, given a ceiling on cache size, off-chip bandwidth will limit the number of cores. Thus, there may be useless area on the chip which cannot be used for cache or processing logic, and which performs no function other than as a placeholder for pin area. That area may be useful to use for compression engines, or intelligent controllers to manage the caches and memory channels.
(Fra forel 8, slide 6 på side 4)

Exercise 4) Research prototypes (Max 20 points)

a) (Max 5 points) Sketch a figure of the main system structure of the Manchester Dataflow Machine (MDM). Include the following units: Matching unit, Token Queue, IO switch, Instruction store, Overflow unit and Processing unit. Show also how these are connected.
See figure 5 in the paper, and slides. The Overflow unit is coupled to the matching unit, in parallel..



b) (Max 5 points) What was the function of the overflow unit in MDM and explain very briefly how it was implemented.

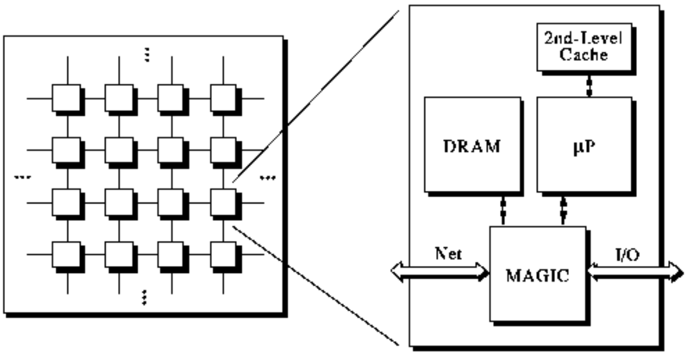
If an operand does not find its corresponding operand in the Matching Unit (MU), and it is not space in MU to store it (for waiting on the other operand), the operand is stored in the overflow store. This is a separate and much slower subsystem with much larger storage capcity. It is composed of a separate overflow-bus, memory and a microcoded processors, in other words a SW-solution. See also figure 7 in the paper.

c) (Max 5 points) In the paper *The Stanford FLASH Multiprocessor* by Kuskin et.al., the FLASH computer is described. FLASH is an abbreviation for *FLexible Architecture for SHared memory*. What kind of flexibility was the main goal for the project?

Programming paradigm, flexibility in the choice between distributed shared memory (DSM) i.e. cache coherent shared memory and message passing, but also other alternative ways of communication between the nodes could be explored.

d) (Max 5 points) Outline the main architecture of a node in a FLASH system. What was the most central design choice to achieve this flexibility?

Fig. 2.1 explain much of this



Interconnection of PE’s in a mesh. The most central design choice was the MAGIC unit, a specially designed node controller. All memory accesses goes through this, and it can as an example realise a cache-coherence protocol. Every Node is identical. The whole computer has one single adress space, but the memory is physically distributed.

---oooOOOooo---