Advanced Computer Organization

Exam #3

Alex Salo

1. What are the main differences between USB 2 and USB 3?

USB 2.0

- Specification developed in 2000
- Provides a third (first two of USB are 12 Mbps and 1.5 Mbps) transfer rate of 480 Mbps while retaining backward compatibility.
- This yields ~32MB/s which was not enough for 2006's HDDs which was addressed in USB 3, with backward compatibility.

USB 3.0

- Ability to move data at rates up to 450MB/s while retaining backward compatibility with USB 2.0.
- Bus operations SuperSpeedPlus (or SSP) features or requirements – generally, "SuperSpeed".
- USB 3.1 extends the performance range of USB up to 1GB/s by doubling the SuperSpeed USB clock rate to 10Gbps and enhancing data encoding efficiency. Thus, USB 3.1 is primarily a performance enhancement to SuperSpeed USB 3.0 resulting in providing more than double the bandwidth for devices such as Solid State Drives and High Definition displays.

SuperSpeed Protocol vs USB 2.0 protocol:

- USB 2.0 uses a three-part transaction (Token, Data, and Handshake) while the EnhancedSuperSpeed protocol uses the same three parts differently. For OUTs, the token is incorporated in the data packet; while for INs, the Token is replaced by a handshake.
- USB 2.0 does not support bursting while the Enhanced SuperSpeed protocol supports continuous bursting.
- USB 2.0 is a half-duplex broadcast bus while the Enhanced SuperSpeed bus is a dual-simplex unicast bus which allows concurrent IN and OUT transactions.
- USB 2.0 uses a polling model while the Enhanced SuperSpeed protocol uses asynchronous notifications.
- USB 2.0 does not have a Streaming capability while the Enhanced SuperSpeed protocol supports Streaming for bulk endpoints.
- USB 2.0 offers no mechanism for isochronous capable devices to enter the low power USB bus state between service intervals. The Enhanced SuperSpeed bus allows isochronous capable devices to autonomously enter low-power link states between service intervals or within a service interval. An Enhanced SuperSpeed host shall transmit a PING packet to the targeted isochronous device before the service interval to allow time for the path to transition back to the active power state before initiating the isochronous transfer.
- USB 2.0 offers no mechanism for a device to inform the host how much latency the device can tolerate if the system enters lower system power states. Thus a host may not enter lower system power states as it might impact a device's performance because it lacks an understanding of a device's power policy. USB 3.1 provides a mechanism to allow Enhanced SuperSpeed devices to inform the host of their latency tolerance using Latency Tolerance Messaging. The host may use this information to establish a system power policy that accounts for the devices' latency tolerance.

Source: USB 3.1 Specs

2. What is the primary motivation for creating DRAM caches?

They could be significantly larger than traditional caches yet faster than just RAM memory on system!

Die-stacking technologies provide a way to tightly integrate multiple disparate silicon die with high-bandwidth, low-latency interconnects. A likely initial use of die stacking will be to integrate a large amount of DRAM with a conventional multicore processor chip. The implementation could involve vertical stacking or horizontal/2.5D stacking on an interposer, but in either case the processors are provided with a high bandwidth, low-latency path to the stacked memory.

DRAM capacity vary from a few tens or low hundreds of megabytes up to a few gigabytes. Even with a few gigabytes of stacked DRAM, this capacity may not be sufficient to support the entirety of a system's main memory. **Thus stacked DRAM could be used as a very large last-level cache.** This approach benefits from being software-transparent, and therefore it is not dependent on new versions of OS's, and could be deployed as quickly as a chip can be designed.

Source: Loh, Gabriel H., and Mark D. Hill. "Efficiently enabling conventional block sizes for very large die-stacked DRAM caches." *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2011.

3. How are the cache tags handled in DRAM caches? Give at least three methods, and describe each fully.

Emerging 3D-stacked memory technology can provide caches of hundreds of megabytes (or a few gigabytes) at almost an order of magnitude higher bandwidth compared to traditional DRAM. However, to get performance benefit from such large caches, one must first handle several key challenges, such as architecting **the tag store**, optimizing hit latency, and handling misses efficiently

1. **Storing tags in SRAM.** This approach stores tags in a separate SRAM structure. This design incurs an unacceptably high overhead since to obtain data, the access must first go through the tag-store, which incur the latency due to serialization of tag access “Tag Serialization Latency” (TSL). TSL directly impacts the cache hit latency, and hence must be minimized.

2. **Tags-in-DRAM: The LH-Cache.** The prohibitive overhead of storing tags in SRAM can be avoided by placing the tags directly in DRAM, but naively doing so doubles the latency, since it would require that each DRAM cache access incurs a latency of two accesses, one for tag and the other for data, further exacerbating the already high hit latency.
   The problem: DRAM caches are much slower than traditional caches, so optimizations that exacerbate the already high hit latency may degrade overall performance even if they provide a marginal improvement in hit rate.
   Improvement: co-locating the tags and data for the entire set in the same row reduces the access penalty of DRAM tags.

3. **Alloy Cache.** Alloy Cache tightly integrates or alloys tag and data into a single entity called TAD (Tag and Data). On an access to the Alloy Cache, it provides one TAD. If the tag obtained from the TAD matches with the given line address, it indicates a cache hit and the data line in the TAD is supplied. A tag mismatch indicates cache miss. Thus, instead of having two separate accesses (one to the “tag-store” and the other to the “data-store”), Alloy Cache tightly integrates those two accesses into a single unified access. On a cache miss, there is a minor cost in that bandwidth is consumed transferring a data line that is not used. Note that this overhead is still substantially less than the three tag lines that must be transferred for both hits and misses in the LH-Cache.
   The minimum size of a TAD is 72 bytes (64 bytes for data line and 8 bytes for tag). The Alloy Cache can store 28 lines in a row, reaching close to the 29-lines per row storage efficiency of the LH-Cache.
   Thus Alloy Cache is highly-effective latency-optimized cache architecture. Rather than splitting cache space into “tag store” and “data store,” it tightly integrates or alloys the tag and data into one unit (Tag and Data, TAD). Alloy Cache streams out a TAD unit on each cache access, thus avoiding the tag serialization penalty

Source: Qureshi, Moinuddin K., and Gabe H. Loh. "Fundamental latency trade-off in architecting dram caches: Outperforming impractical sram-tags with a simple and practical design." *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2012.

4. Is it possible to do away with DRAM cache tags completely? How?

The conventional die-stacked DRAM cache has both a TLB and a cache tag array, which are responsible for virtual-to-physical and physical-to-cache address translation, respectively.

Is possible to do without cache tags? Sure!

How? Align the granularity of caching with OS page size and take a unified approach to address translation and cache tag management:

- Use cache-map TLB (cTLB), which stores virtual-to-cache, instead of virtual-to-physical, address mappings.
    - At a TLB miss, the TLB miss handler allocates the requested block into the cache if it is not cached yet, and updates both the page table and cTLB with the virtual-to-cache address mapping.
    - Assuming the availability of large in-package DRAM caches, this ensures that an access to the memory region within the TLB reach always hits in the cache with low hit latency since a TLB access immediately returns the exact location of the requested block in the cache, hence saving a tag-checking operation.
    - The remaining cache space is used as victim cache for memory pages that are recently evicted from cTLB.

By completely eliminating data structures for cache tag management, from either on-die SRAM or in-package DRAM, described above DRAM cache achieves best scalability and hit latency, while maintaining high hit rate of a fully associative cache.

source: Lee, Yongjun, et al. "A fully associative, tagless DRAM cache." *Proceedings of the 42nd Annual International Symposium on Computer Architecture*. ACM, 2015.

5. What is bandwidth bloat?

Bloat Factor - is the ratio of the total bandwidth consumed by the DRAM cache to the bandwidth required for transferring only the data lines to the processor chip. It is used to measure the consumption of bandwidth due to secondary operations such as cache miss detection, fill on cache miss, and writeback lookup and content update on dirty evictions from the last-level on-chip cache.

Consumption of bandwidth due to secondary operations is undesirable - ideally, we want the bandwidth consumed for such secondary operations to be negligible, and have almost all the bandwidth be available for transfer of useful data from the DRAM cache to the processor, which should improve system performance by servicing data at a faster rate when the requested data is found in the cache, potentially increasing the memory bandwidth of the system by 4x-8x.

Thus, **bandwidth bloat –** is inability of DRAM cache to provide the performance increase due to the consumption of bandwidth by the secondary operations.

Source: Chou, Chiachen, Aamer Jaleel, and Moinuddin K. Qureshi. "BEAR: techniques for mitigating bandwidth bloat in gigascale DRAM caches." *Proceedings of the 42nd Annual International Symposium on Computer Architecture*. ACM, 2015.

6. What are the advantages of having redundant mapping for virtual memory?

Redundant mapping – is hardware/software co-designed implementation of virtual memory which allow efficient representation of many virtual-to-physical mappings.

Virtual memory: page-based virtual memory improves productivity, security and memory utilization.

Problem: page-based virtual memory incurs performance overhead (up to 50%) due to costly page table walks after TLB misses.

Advantages of redundant mapping in virtual memory:

- Redundant Memory Mapping (RMM) adds a redundant mapping in addition to page tables, that provides a more efficient representation of translation information for ranges of pages that are both physically and virtually contiguous. Investigations has showed that diverse workloads exhibit an abundance of contiguity in their virtual address space.
- RMM modifies OS memory allocation mechanism to use eager paging, which serves to allocate the largest possible range of contiguous virtual pages to contiguous physical pages. That allows to achieve a high hit ration in the range TLB and this low virtual memory overheads.

Summary:

Redundant Memory Mapping is hardware/software co-designed implementation of virtual memory which allow efficient representation of many virtual-to-physical mappings. That reduces costly page table walks after TLB misses and allows achieving a high hit ration in the range TLB and this low virtual memory overheads.

Source: Vasileios Karakostas, Jayneel Gandhi, Furkan Ayar, Adrián Cristal, Mark D. Hill, Kathryn S. McKinley, Mario Nemirovsky, Michael M. Swift, and Osman Ünsal. 2015. Redundant memory mappings for fast access to large memories. In Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA '15). ACM, New York, NY, USA, 66-78.
DOI=http://dx.doi.org/10.1145/2749469.2749471

7. What are PIM-enabled instructions, and why would one have such a thing? How would such a thing would be implemented?

PIM – is Processing-in-Memory – potentially viable solution for the memory wall crisis.

Why or what is the problem? During past decades there occurred a wide discrepancy between computation speed (fast) and data transfer speed (slow, relatively to computation). That phenomenon is commonly known as the *memory wall.*

What is PIM exactly? PIM is potentially capable of minimizing data movement by placing the computation close to where the data resides. This idea was researched decades ago but was impractical. However today, thanks to 3D stacking technology that enables cost-effective integration of logic and memory and due to the increase of data-intensity of many new applications (HD video, warehousing etc) which demand great amounts of memory bandwidth.

How could be implemented? Desirable features:

- Do not require new programming model for in-memory computation units
- Use on chip caches and virtual memory provided by host processors.

How?

- Enable PIM instructions by extending ISA of the host processor with PIM-enabled instructions (PEIs)
- Design an architecture that supports implementing PEIs as part of the host-PIM interface in order to provide the illusion that PIM operations are executed as if they were host processor instructions
- Track locality of data accessed by PEIs and execute PEIs with high data locality on host processors instead of offloading them to memory, in order to exploit large on-chip caches.
- Allow for memory coherence: design architecture that supports hardware cache coherence for PIM operations so that (1) PIM operations can access the latest versions of data even if the target data are stored in on-chip caches and (2) normal instructions can see the data modified by PIM operations. This allows programmers to mix normal instructions and PEIs in manipulating the same data without disabling caches, in contrast to many past PIM architectures
- Support memory atomicity: ensure atomicity between PEIs. In other words, a PEI that reads from and writes to its target cache block is not interrupted by other PEIs (possibly from other host processors) that access the same cache block.

Source: Ahn, Junwhan, et al. "PIM-enabled instructions: a low-overhead, locality-aware processing-in-memory architecture." *Proceedings of the 42nd Annual International Symposium on Computer Architecture*. ACM, 2015.

8. What is quantum computing, and what are its potential advantages over conventional computing?

Quantum computing – use quantum instead of traditional transistors.

The rules governing the world of very things smaller than 10^9 (nano) are called quantum mechanics and significantly differ from that of the classical physics that we have applied in the macro world. If we can successfully exploit quantum mechanics for communication and computing purposes, then we will have the chance to construct more efficient algorithms than those that are based on more traditional means. Furthermore, nanoscale computing and communications will provide the opportunity to exceed the classical capacity of communication channels.

How?

- Using quantum mechanical properties, such as the polarization of a photon or the spin of an electron, which might carry information
- Using cubits $|\phi >= a_0|0 > \ +a_1|1 >$ - two dimensional vector. Qubit contains the superposition of both logical values at the same time.
- By merging qubits into registers. If we consider the fact that a qregister of n = 500 qubits may yield classical numbers on a scale larger than all of the atoms in the known universe in total, we can then imagine the parallel information storage capacity offered by quantum mechanics. The major challenge however for quantum algorithms is to modify the probability distribution of the measurement results in such a way that the requested result of the computation problem will be obtained with high probability.

Advantages over conventional computing:

- Cryptography: traditional cyphers (like RSA, SHA-256) become very easy to break. However, using quantum algorithms better and more secure algorithms for protection could be used.
- ~sqrt(N) search time in database of N items (compare ~N time for conventional)
- Multiple optimization problems could be solved much faster
- Any probabilistic algorithms in general could be done much faster
- Improve communication protocols (e.g., the strong doubt surrounding public key cryptography can be eliminated in worldwide networks).
- Better protection from unsanctioned copying due to the “no cloning” theorem.

9. What are Automatic Custom Instructions, and how are they implemented?

The application-specific instruction set processors **(ASIPs)** make trade-offs between flexibility and performance by extending the base instruction set of a general-purpose processor with custom functional units (CFUs). Custom instructions **(CIs)**, executed on CFUs, improve performance and achieve flexibility for extensible processors. CI is a complex instruction that encapsulates several basic instructions. A CFU is the hardwired implementation of a custom instruction. Generally, application programs from the same domain have similar structure. Using the same set of custom instructions across different application programs from the same domain is the flexibility provided by ASIPs.

The key steps involved in a custom instruction synthesis flow for ASIPs are custom instruction enumeration and custom instruction selection. However, it is really time-consuming to enumerate custom instructions and select custom instruction from a given program's data-flow graph (DFG). Hence, a **fully automated custom instruction synthesis flow is necessary**.

Automatic Custom Instructions - are automatically enumerated suitable CIs that exploit the features and bandwidth offered by the statically defined LSU and AGU, i.e. we want to detect memory access patterns in the application, to configure the AGUs correspondingly, and to extract the affected data flow as CIs

How?

1. (based on compiler IR, which is represented as one combined control and data flow graph (CDFG) per function in the source code):
   The CDFG consists of basic blocks (BBs) in which the total order of evaluation can be relaxed to a partial order allowing for a high degree of instruction level parallelism which will be utilized for generating CIs. Operations that alias a memory location (like two pointers to the same address) are still kept in correct order by extra memory dependency edges and, if unavoidable, synchronization points. The control flow determines the BB execution sequence and within a BB the data flow between operations is given. Techniques:
   a. Increase the BB size and thus their inherent parallelism by applying (on the IR):
      i. loop unrolling
      ii. IF-conversion (i.e. replacing basic if-then-else control flows by data flow with a multiplexer node that selects the output)
   b. Profiling is used to identify and ensure that only computationally relevant BBs are processed, and it is ensured that no forbidden nodes are part of the BB.
   c. Function calls are in-lined (where possible). BBs that contain a function call which could not be inlined (e.g. library calls or system calls) cannot be transformed into a CI, as it is not possible to switch execution control to the CPU before the active CI is finished. Thus remaining call nodes are treated as forbidden nodes.
2. (based in compiler GECOS):
   a. The application program is firstly translated to a control data-flow graph (CDFG). A CDFG is a graph that represents the control dependencies among a number of basic blocks and the data dependencies inside the basic blocks.
   b. Then, all the subgraphs (custom instructions are represented as subgraphs) satisfying the architectural constraints are enumerated.

c. Based on the enumerated subgraphs, only a subset of subgraphs are selected as custom instructions according to different strategies (minimum run time, minimum number of custom instructions, minimum power consumption, etc.).
d. After selecting the most profitable custom instructions, the source code is transformed to a new code by incorporating the selected custom instructions and this new code is provided to the compilation process.
e. Finally, the set of selected custom instructions is provided to the hardware synthesis process to produce the corresponding hardware (CFU) that will be added to the ASIPs.

sources:

1. Haaß, Martin, Lars Bauer, and Jörg Henkel. "Automatic custom instruction identification in memory streaming algorithms." *Proceedings of the 2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. ACM, 2014.

2. Xiao, Chenglong, et al. "Automatic custom instruction identification for application-specific instruction set processors." *Microprocessors and Microsystems* 38.8 (2014): 1012-1024.

10. What is a residue number system, and what advantages does such a system have over conventional number system?

Residue Number System (RNS) is based on a puzzle Chinese Remainder Theorem (CRT) introduced by the Chinese mathematician Sun-Tzu. It has several interesting number theoretic properties and unique features that can be used to boost up the speed of certain electronic computations. The carry-propagation chain of conventional binary number system was the main bottleneck of fast arithmetic operation and became the key motivation driving researchers to venture into this alternative number system for which the residue arithmetic operation in each modulus channel is independent and carry-free.

An RNS is characterized by a set of N pairwise relatively prime numbers known as moduli m_i for i=1,2,...N. The dynamic range M of data representable in RNS is determined by the product of all the moduli. An unsigned integer X within M can be uniquely represented using residue digits which are computed by taking the least positive number of the division of X by m_i. To represent a signed integer, M is divided into two sub-ranges. The lower half and upper half ranges are used to represent positive and negative integers, respectively.

Advantages:

- Thus **smaller modular arithmetic operations can be carried out in parallel and at a faster speed than in two's complement number system** (TCS) for the identical dynamic range.
- Moduli set with specific power of two moduli (i.e. $2^n \pm 1$) offers attractive mathematical properties for manipulation to simplify the arithmetic units and converter designs.
- In each modulus channel, modular arithmetic operations are performed on the corresponding residue digits independently and their carry outputs do not propagate across modulus channels. Using features of RNS it is possible to design a digital correlator with ten times faster speed than that based on conventional binary number system.
- Special-purpose digital computer for solving simultaneous equations using RNS with a great speed advantage.
- A Redundant RNS (RRNS) with error detection and correction capability can be formed by adding redundant moduli into an existing moduli set to extend the legitimate range of the original information moduli.

Advantages in applications in which RNS enables efficiency boost:

- Digital signal processing
- Communication and networking: to reduce the number of bits needed to be forwarded by each next-hop node, a set of small consecutive primes that satisfy the condition M >= 2^w can be selected as the moduli without increasing the number of forwarders and yet the message can still be reconstructed if one or more remainders are lost.
- Cloud services to provide greater reliability in terms of long-term availability and privacy: split file into p + r residue-segments based on RRNS, where r is the number of redundant moduli. Each chunk is encoded by BASE-64 before it is encrypted with a symmetric algorithm to encapsulate the binary data in the payload of an XML wrapper file. Finally, each encrypted chunk is sent to a different cloud storage provider.
- Fault tolerant computing; Cryptography and cyphers

This is the place for a Christmas gift +10!