Trip Report: Principles and Practice of Parallel Programming/**High-Performance Computer Architecture** 2019

PPoPP Sat 16 - Wed 20 February 2019 Washington, DC, United States

25th IEEE International Symposium on High-Performance Computer Architecture

February 16-20 2019, Washington D.C., USA

- Co-located with International Symposium on Code Generation and Optimization (CGO) and 28th International Conference on Compiler Construction (CC).

- Links:
  - PPoPP: https://ppopp19.sigplan.org/track/PPoPP-2019-papers#program
  - HPCA: http://hpca2019.seas.gwu.edu/keynote.html
  - CGO: http://cgo.org/cgo2019/
  - CC: https://cc-conference.github.io/19/

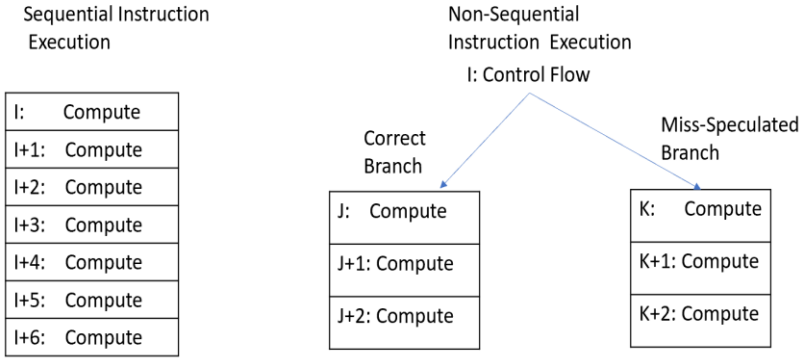
- I attended mostly HPCA sessions.
- Disclaimer: I have taken most diagrams and texts from the slides available or the papers.

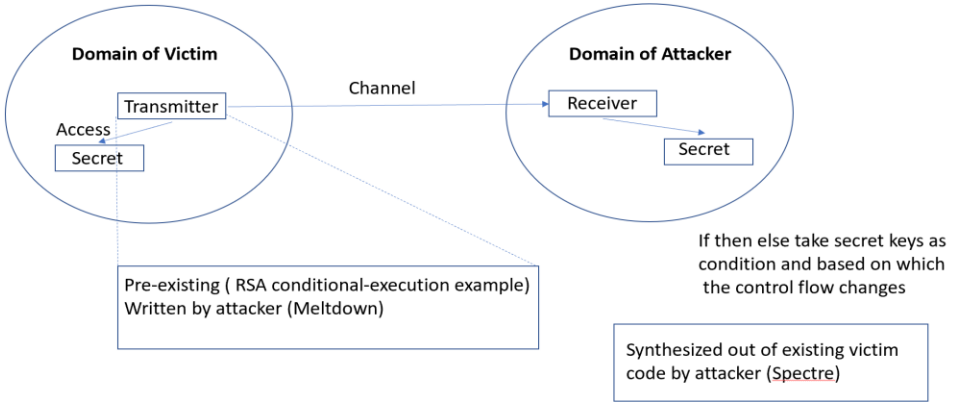
HPCA keynote: Towards Secure High-Performance Computer Architectures (Srini Devadas, MIT)

- Side channel attacks exploits microarchitectural optimizations like speculations to leak secrets.

- Isolation breaks because of shared microarchitectural states (shared last level cache, memory addresses, and cache addressing (set, tag), etc)
  - Rollback is perfect and restores back to the previous architectural states
  - But side channels can exploit mis-predicted branches and non-architectural states affected

- But can't do away with optimizations due to performance reasons.

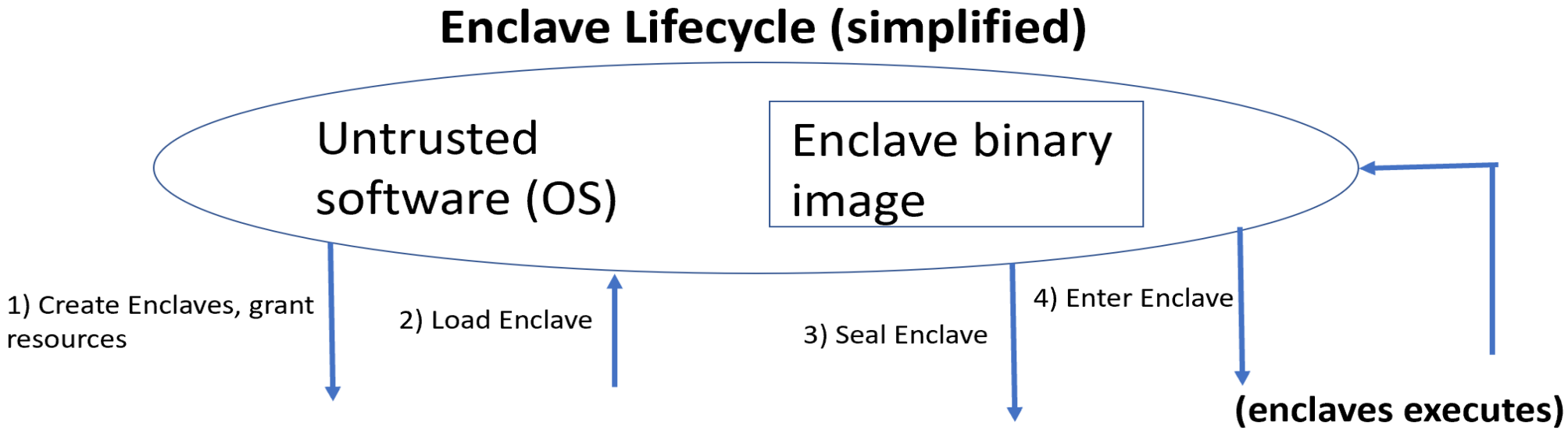




- Design processors to support *enclaves* (processes with associated security policy, microarchitectural isolation).

- No other program should be able to infer anything private about the enclave process through it's shared resources or shared micro-architectural states (separation of resources at all levels).

- All resources are isolated *spatially* and *temporally* as required by the threat model.



- *Strong timing independence*: Two programs are independent if a program's sequence and timing of microarchitectural events do not depend on another program.

- *Strong timing independence* implies *architectural independence.*

- Propose MI6, an out-of-order processor capable of providing secure enclaves.

- MI6 assures security in the scenario where the threat model consists of :-
  - an untrusted OS
  - attacker capable of mounting any practical software attacks.

- Enclaves run secure tasks side-by-side of ordinary processes.

- Use open source *RiscyOO* processor as a baseline. But *RiscyOO* also suffers from cache-timing side channels and is vulnerable to speculation based attacks like Spectre.

Architectural changes made

- Cache set partitioning (avoid cache timing side channels).
- MSHR partitioning, MSHRs sizing to prevent DRAM backpressure and isolation.
- Constant latency DRAM controller (to provide timing isolation).
- Flushing micro-architectural states (private cache, TLB, branch predictor states on context switch, restore later).
- Page-walk checks (DRAM region based partitioning and thus access translations in TLB).
- Turning off speculation and checking instruction fetches in the machine mode (high-level security monitor that helps assert resources allocated to an enclave).
- Have other functioning like TLB shootdowns, flush to ensure that speculative loads/stores don’t violet the security policy.

HPCA: Session 1: Best paper nominees

- **The Accelerator Wall: Limits of Chip Specialization** *Adi Fuchs and David Wentzlaff (Princeton University)*

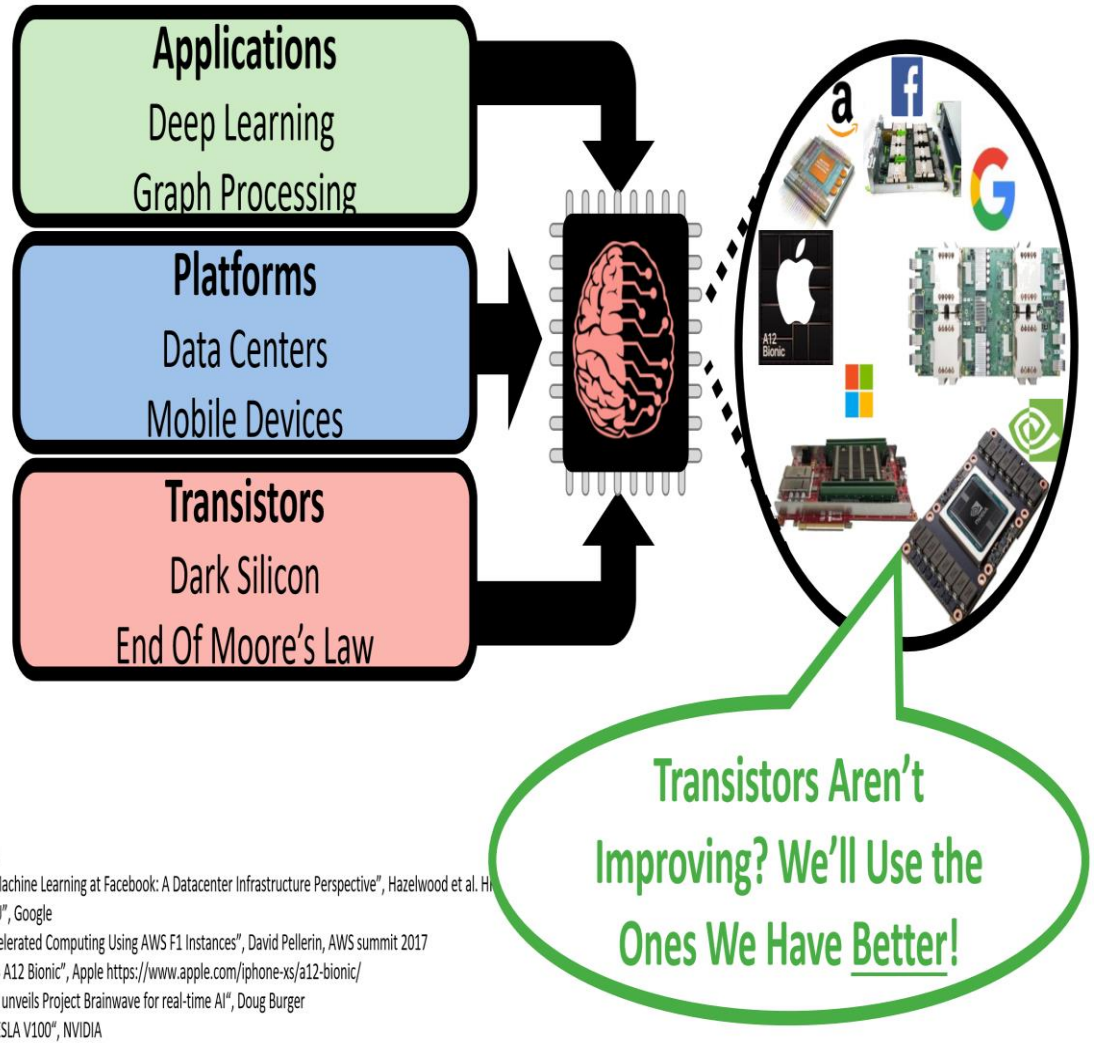
- **Stretch: Balancing QoS and Throughput for Colocated Server Workloads on SMT Cores** *Artemiy Margaritov (University of Edinburgh); Siddharth Gupta (EPFL); Rekai Gonzalez-Alberquilla (Arm Ltd, Cambridge, UK); Boris Grot (University of Edinburgh)*

- **CIDR: A Cost-Effective In-line Data Reduction System for Terabit-per-Second Scale SSD Arrays** *Mohammadamin Ajdari (POSTECH); Pyeongsu Park, Joonsung Kim, Dongup Kwon, and Jangwoo Kim (Seoul National University)*

- **Composite-ISA Cores: Enabling Multi-ISA Heterogeneity Using a Single ISA** *Ashish Venkat (UCSD/UVA); Harsha Basavaraj and Dean Tullsen (UCSD)*

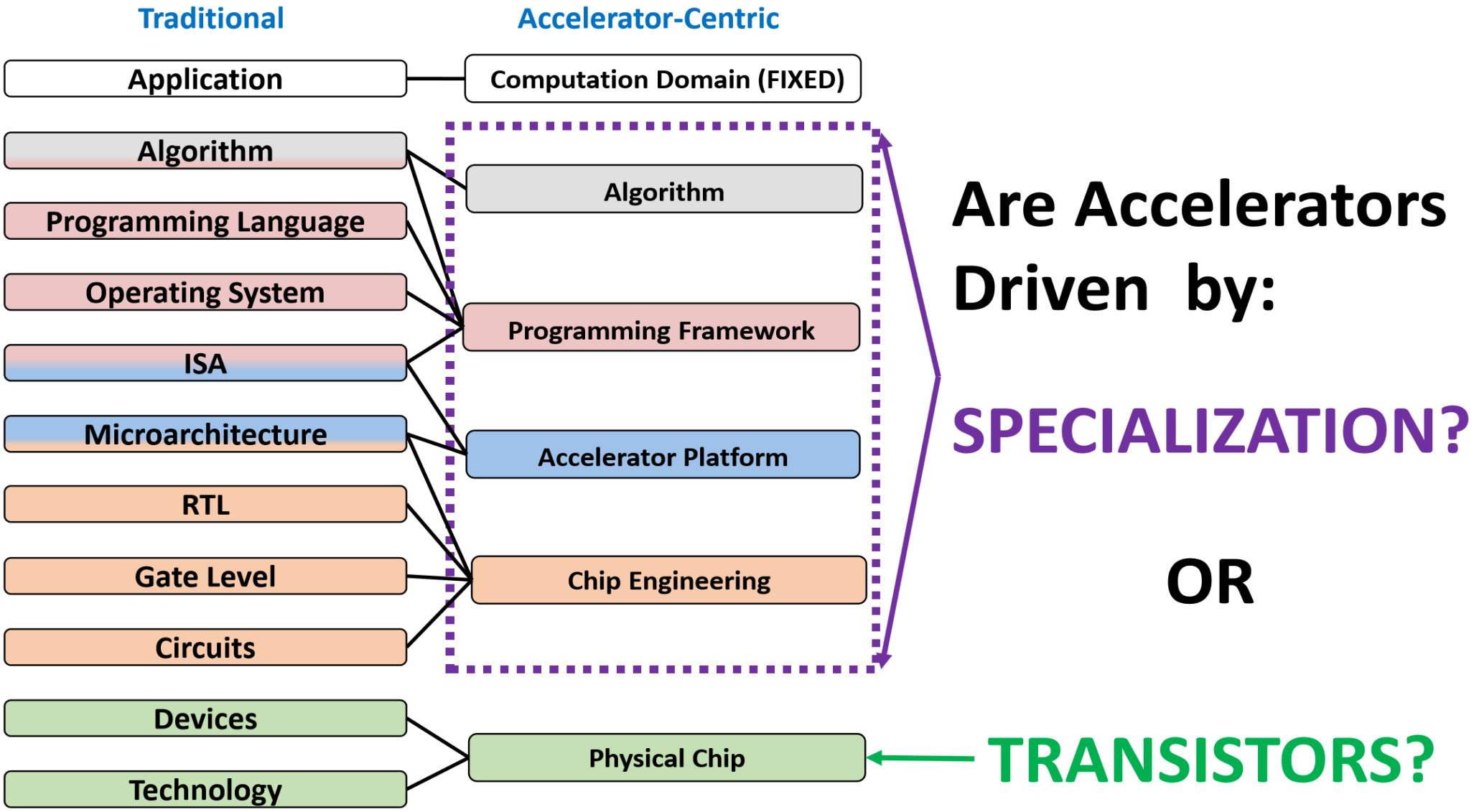
The Accelerator Wall: Limits of Chip Specialization

- Applications domains like Deep learning, Graph Processing running on Data centers and mobile devices are very common these days.
- Accelerators are specialized chips that get more out of the silicon than general purpose CPUs for particular functions (problems like dark silicon, End of Moore's Law).
- No more transistors -> No more cores (in a single chip) -> No more parallelism (within the chip)



Sources:
"Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective", Hazelwood et al. H
"Cloud TPU", Google
"FPGA Accelerated Computing Using AWS F1 Instances", David Pellerin, AWS summit 2017
"IPhone XS A12 Bionic", Apple https://www.apple.com/iphone-xs/a12-bionic/
"Microsoft unveils Project Brainwave for real-time AI", Doug Burger
"NVIDIA TESLA V100", NVIDIA

- What helps accelerators perform? Where does the benefit come from?



- Accelerators are good, but they are not solving the transistor problem.
- Use “throughput per Silicon” to build a CMOS potential model.
- *Device level scaling* takes into account frequency, leakage power, dynamic power (CMOS scaling study + Projections)
- *Chip Transistor Budget* using datasheets of thousands of commercial processors
- CMOS Potential= Device level scaling + Chip Transistor Budget

- **Formal Definition:**

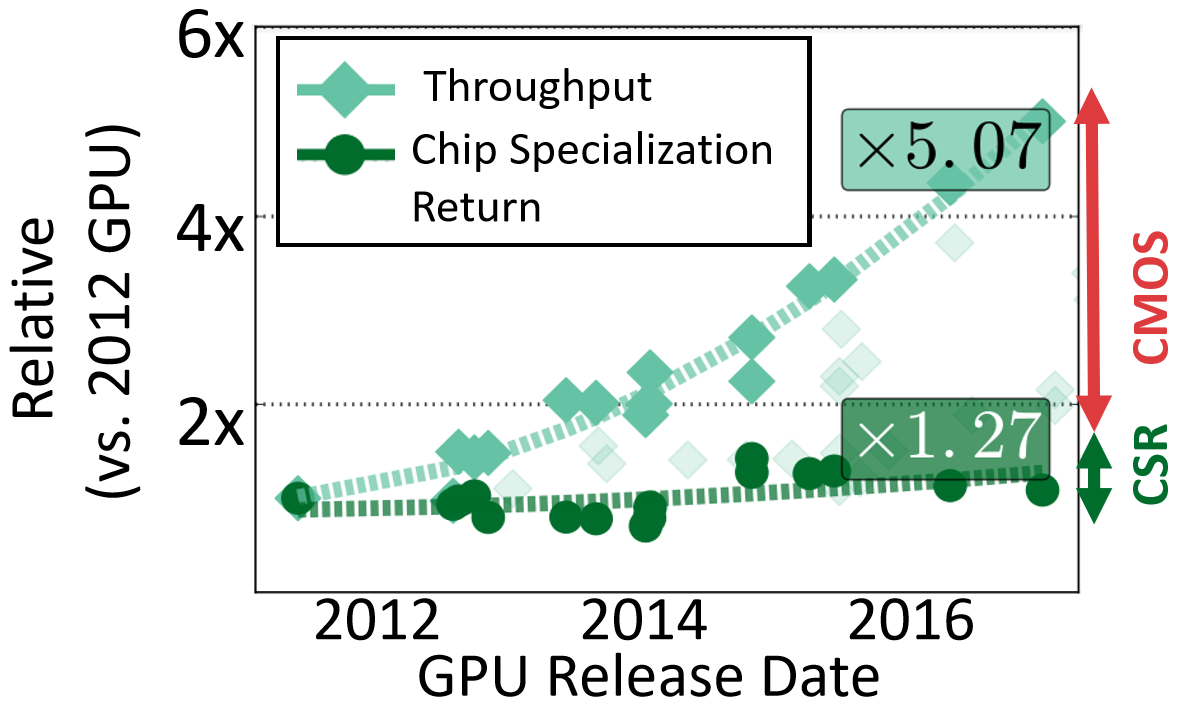
$$Gain = \boxed{\frac{Gain}{CMOS\ Potential}} \times CMOS\ Potential$$

Chip Specialization Return (CSR)

- **Comparing Accelerators:**

$$\frac{Gain_{ACCELERATOR\ B}}{Gain_{ACCELERATOR\ A}} = \frac{CSR_B}{CSR_A} \times \frac{CMOS\ Potential_B}{CMOS\ Potential_A}$$

- **Example:** Gaming Throughput on GPUs
  - **Throughput (Gain) Improvement: 5.07x**
  - **CMOS Scaling Contribution: 4x**
  - **Specialization Contribution: ONLY 1.27x**



Data Sources: www.techpowerup.com/gpudb
https://www.anandtech.com/bench/GPU18

- Chip specialization is not a long-term remedy for the end of Moore's Law.

- Need model to decouple *transistor returns* from *specialization returns* for accelerators.

- Introduce Rankine: A CMOS Potential Modeling Tools
  - Based on databased of thousands of commercial processor
  - Evaluate one accelerator vs other based on CMOS returns
  - Calculates CMOS Potential based on Physical Chip Properties e.g., CMOS Process, Die Size/Number of Transistors, etc.
  - GitHub Repo: https://github.com/PrincetonUniversity/accelerator-wall

Stretch: Balancing QoS and Throughput for Colocated Server Workloads on SMT Cores

- Data centers want to maximize performance per Watt and performance per Dollar value paid.
- To do that, they collocate latency sensitive and batch workloads on the same CMP as well as within a single SMT core.
- But at lower load rates, there is significant slack available from latency-sensitive workloads (still maintaining target QoS).
- Batch co-runners suffer more in terms of performance when they are co-located.
- Batch and latency sensitive workloads have different sensitivity to ROB capacity.

- Latency sensitive workloads don’t benefit much from larger ROB capacity;
  - this is due to frequent cache misses and data dependent computation that limits Instruction Level Parallelism (ILP) and memory level parallelism (MLP).

- Batch processes benefit from higher ROB capacity by exploiting ILP and MLP.

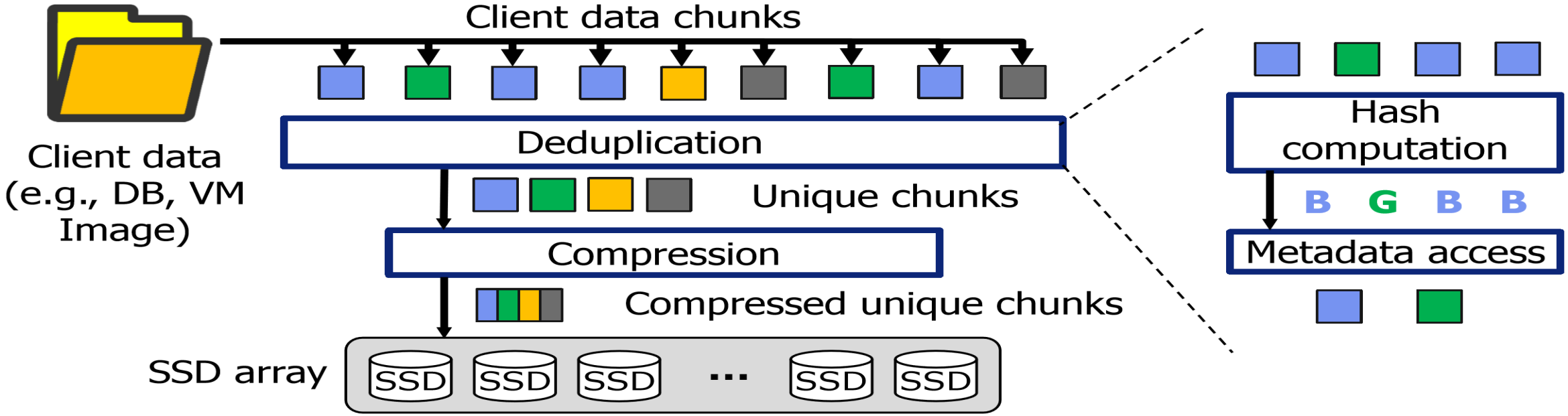
- Stretch is a simple mechanism that helps in boosting performance by providing ROB partitioning configurations.

- When latency-sensitive applications are running below their peak load, the excess slack can be used to shift ROB capacity to the batch applications.

- System software detects when latency-critical applications are running low loads and thus shift to different ROB partitioning configuration.

CIDR: A Cost-Effective In-Line Data Reduction System for Terabit-per-Second Scale SSD Arrays

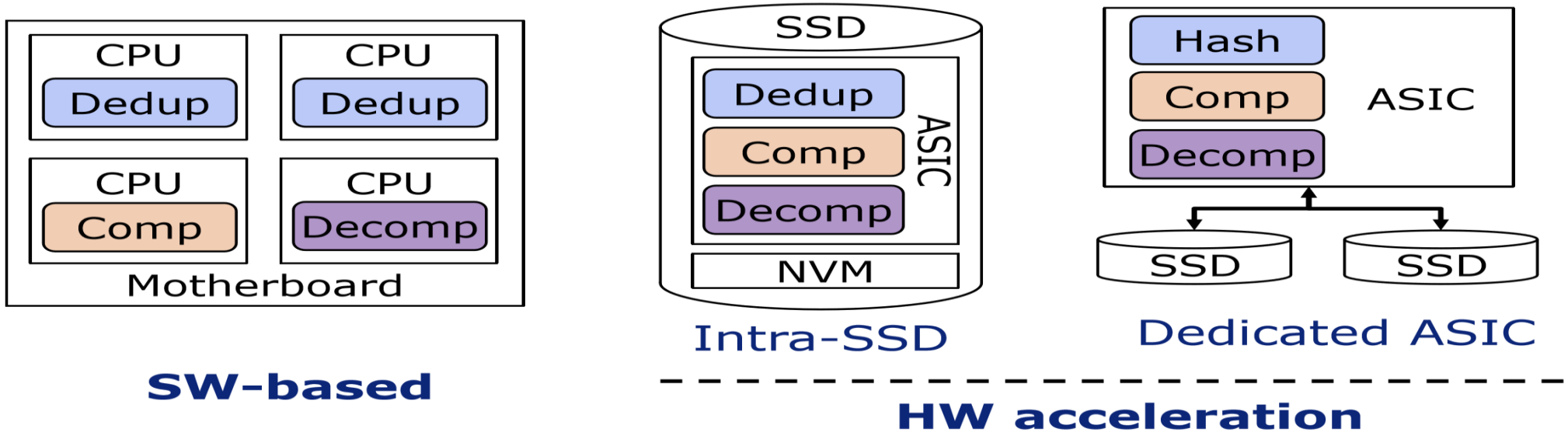
- Cost-effective ways required to store data in SSDs as they are expensive.
- Can use accelerators (FPGAs) for data compression, but need efficient mechanism.



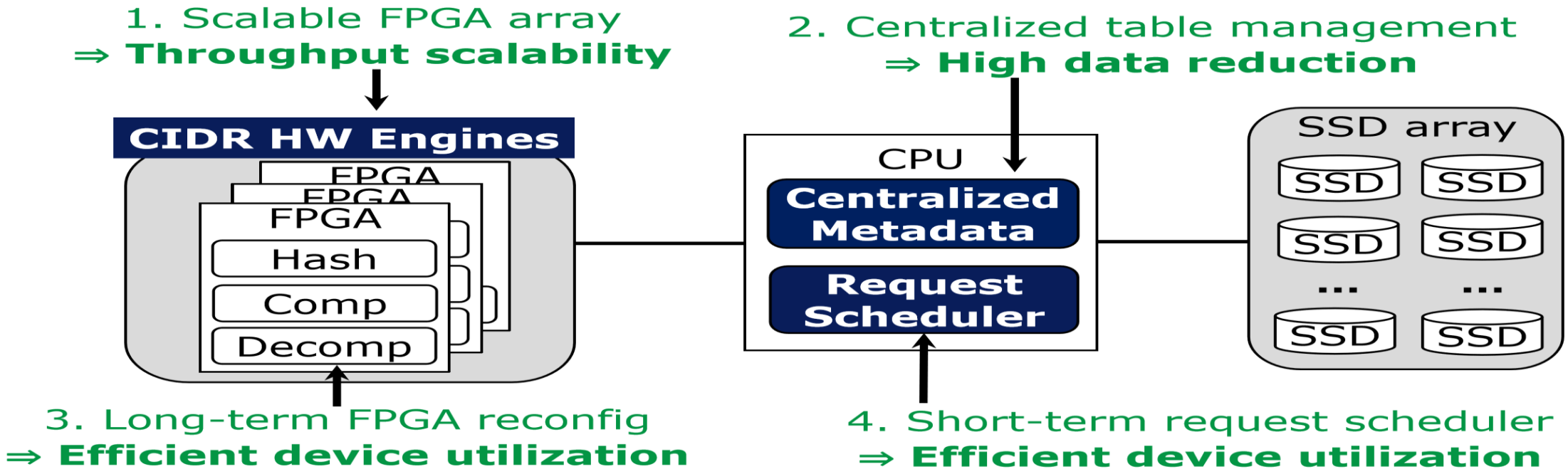
Deduplication + compression → 60-90% data reduction

Existing approaches for data reduction have limitations

- SW-based approach (CPUs run data reduction operations)
  - Low throughput and scalability due to CPU bottleneck
- Intra-SSD HW acceleration (accelerators in each SSD for scalable throughput)
  - Low data reduction due to **no** inter-SSD deduplication
- Dedicated HW acceleration
  - Low device utilization as separated from SSDs

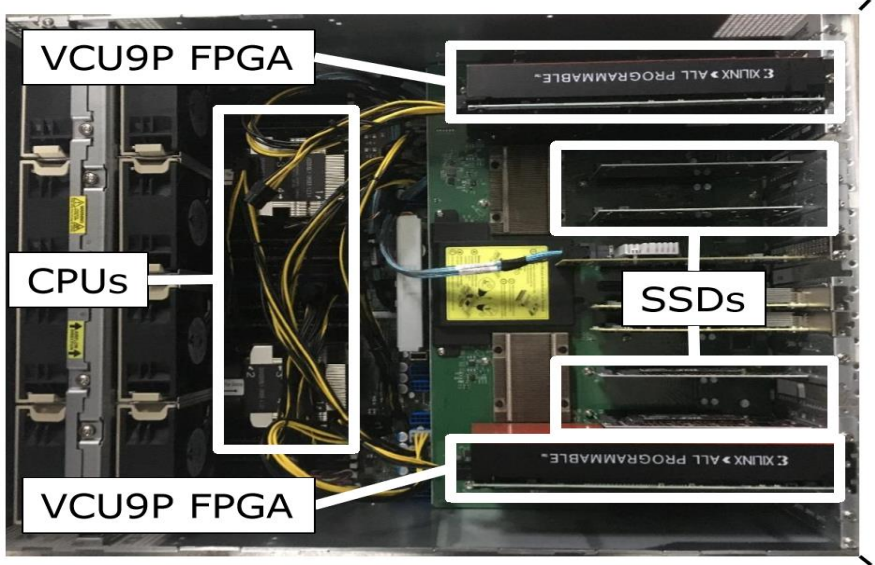


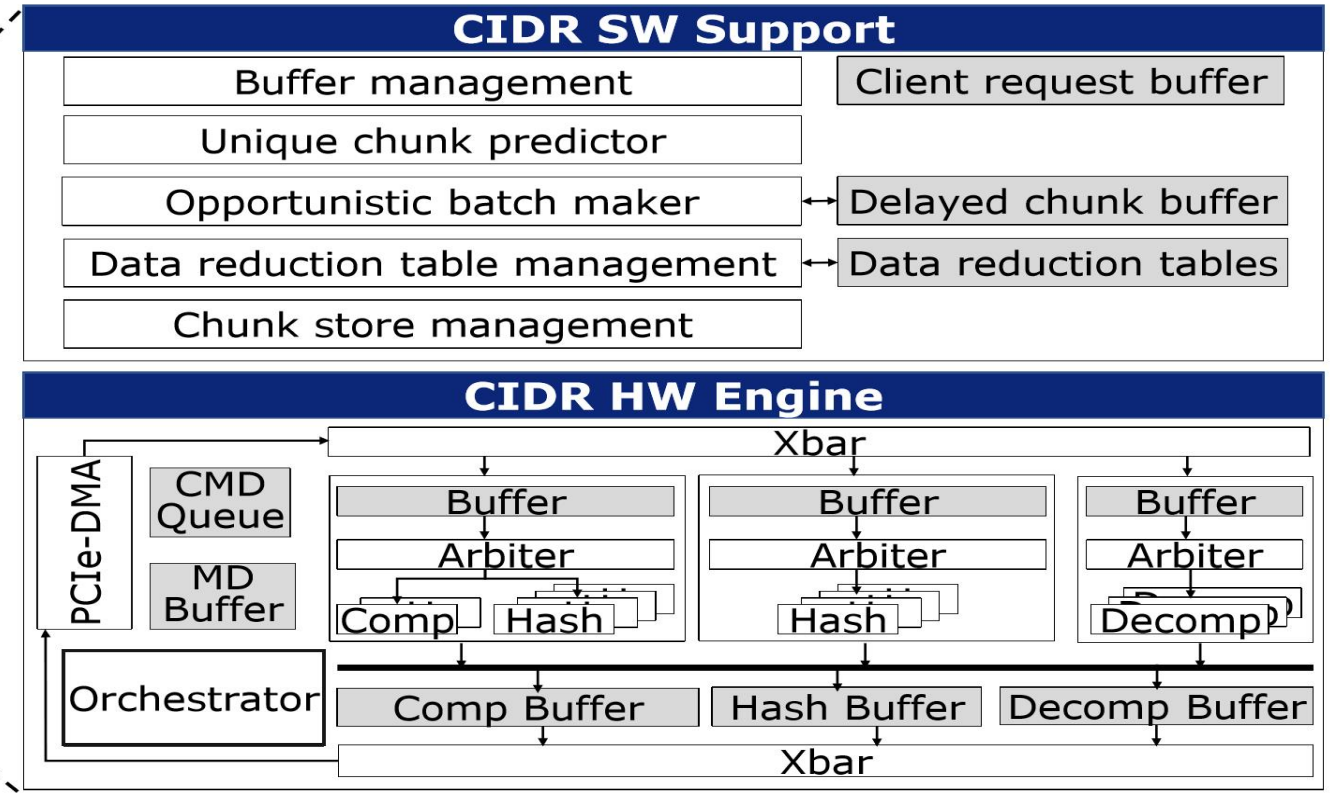
- Design goals to have, throughput and scalability, high data reduction and efficient device utilization.
- Key ideas for CIDR:-
  - Scalable FPGA array (deploy a CPU-free, scalable FPGA accelerator array)
  - Centralized table management (detect all duplicate chunks in a large SSD array)
  - Long-term FPGA reconfig (reconfigure FPGAs to workload's average behavior)
  - Short term request scheduler (schedule requests considering available HW resources)



Optimizations

- Use SRAM instead of slow DRAM buffer
- Cluster units to make distribution network simple
- Remove the large SRAM queue by predicting uniqueness
- Delay over-subscribed requests into the delayed buffer





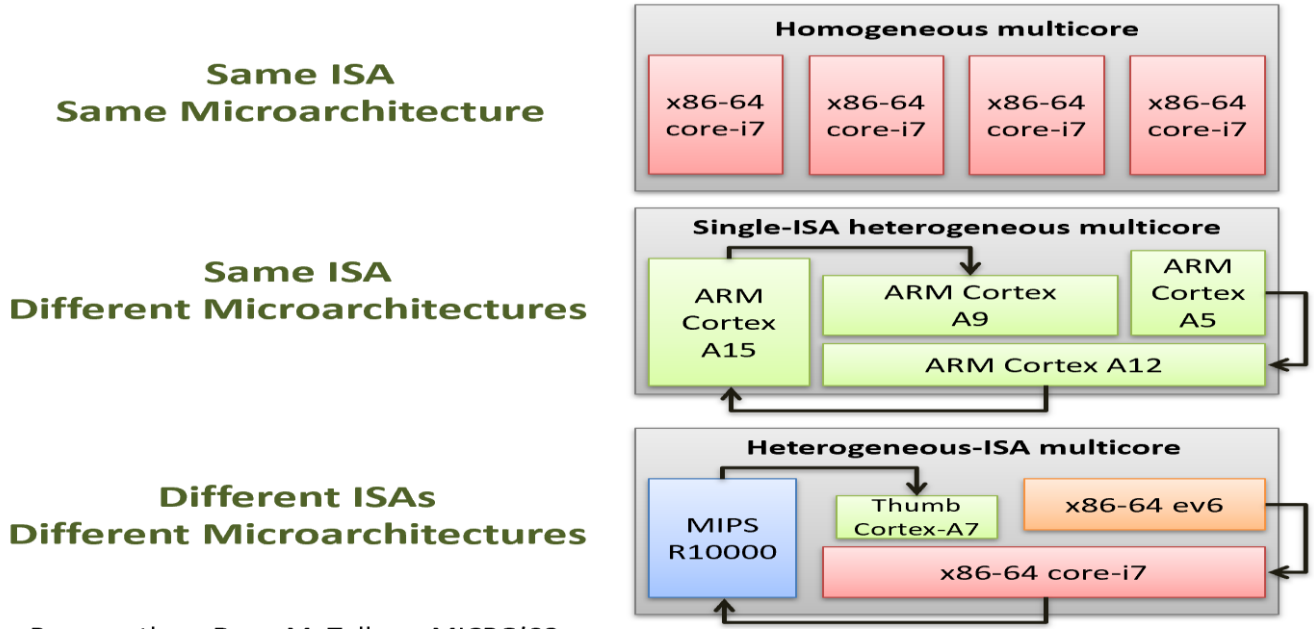
Composite-ISA Cores: Enabling Multi-ISA Heterogeneity using a Single ISA

- Hardware specialization can be provided based on two basic ideas :-
  - **Domain-specific specialization**: Accelerate the performance of a particular class of problems
  - **Microarchitectural heterogeneity**: Use a large number of small power-efficient diverse cores

- Is it possible to benefit from this heterogeneity and specialization while preserving the traditional programming model?

- Some tradeoffs, Speedup vs. energy savings with performance loss.

- ISA design metrics:-
  - High performance
  - Low power
  - Reduced code size
  - Domain-specific instructions, etc.



- Different configurations should be used based on the requirements.

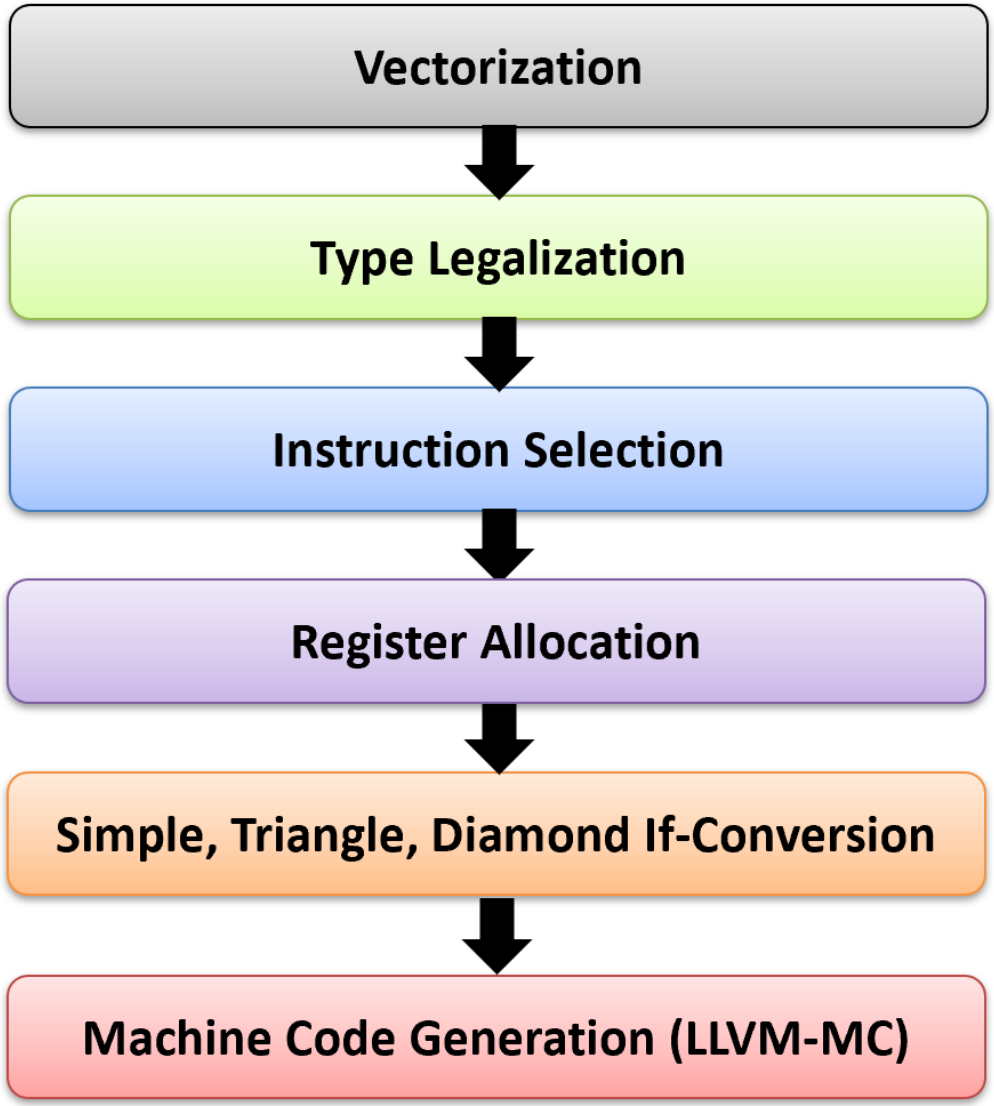
- Having the same ISA for all cores restricts heterogeneity and may not achieve full potential performance.

- Harnessing ISA Diversity helps:-
  - Exploit ISA affinity (applications have preference ISAs)
  - Enables ISA-microarchitecture co-design (helps overall energy consumption compared to homogeneous ones)

- Cross ISA migrations are difficult due to a variety of reasons ranging from different machine code, register file width, and data formats. Also there is a multi-vendor licensing issue.

- The paper avoids multi-vendor licensing issues by using a single vendor (considering a single vendor has the best chips in all domain, not true, greatly reduces the challenges).

- Three basic strategies in achieving Composite ISA cores:-
  - ISA feature set derivation
    - Start with base ISA (x86), customize along 5 dimensions (next slide)
    - 26 different composite ISAs
  - Compiler Strategy
  - Design Space exploration (choice of microarchitectural parameters, 4680 single core design points, San Diego Supercomputer Center)

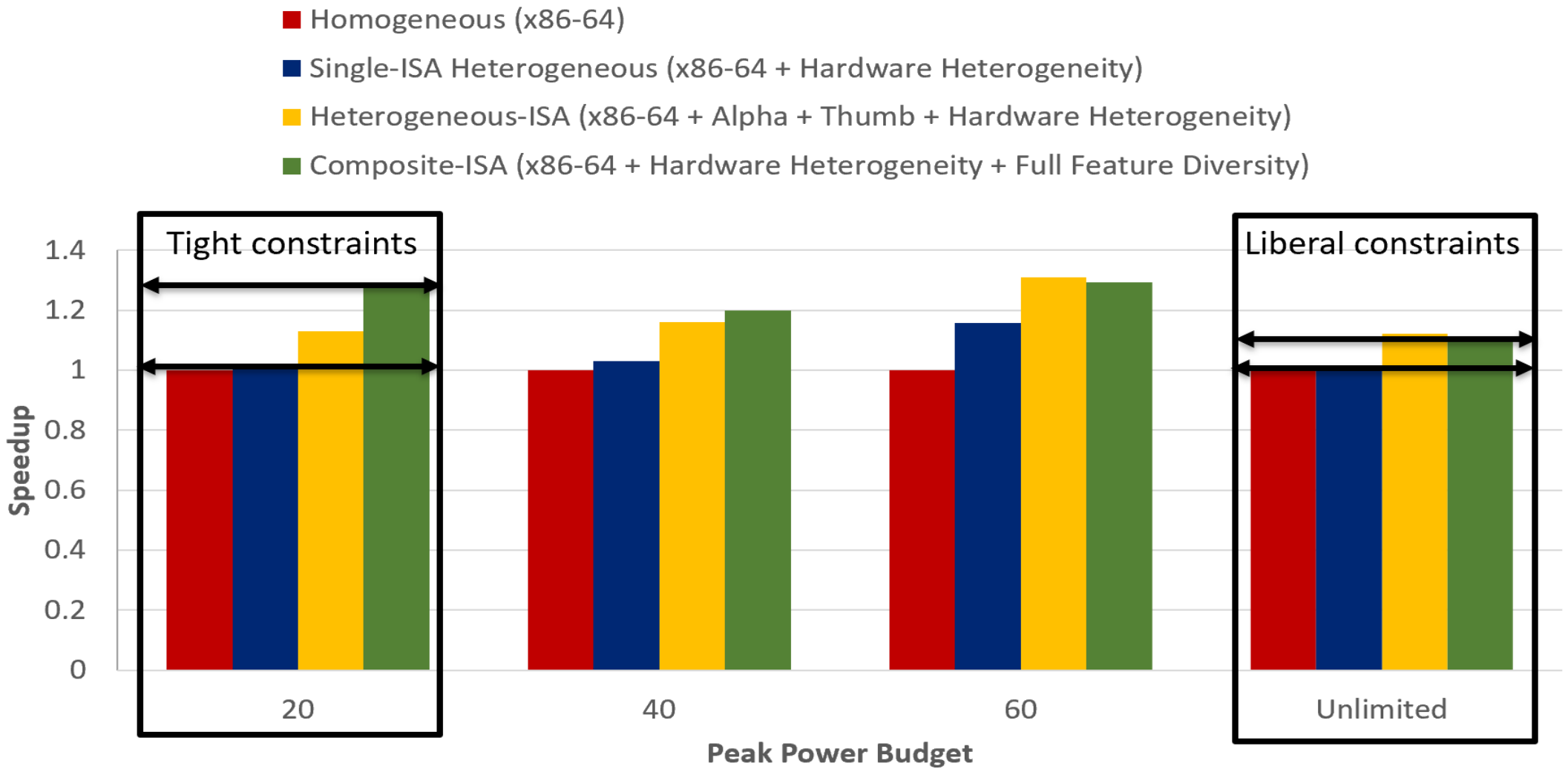


Composite-ISA Features:

- Data Parallelism: {SIMD, no SIMD}
- Register Width: {32-bit, 64-bit}
- Addressing Mode Options: {x86, microx86}
- Register Depth: {8, 16, 32, 64 registers}
- Predication: {partial (CMOV), full predication}

Composite-ISA Encoding Prefixes and Options

- Power constrains determine performance for composite ISA



Session 3B: Emerging technologies

- **The What's Next Intermittent Computing Architecture**
  Karthik Ganesan (University of Toronto); Joshua San Miguel (University of Wisconsin-Madison); Natalie Enright Jerger (University of Toronto)

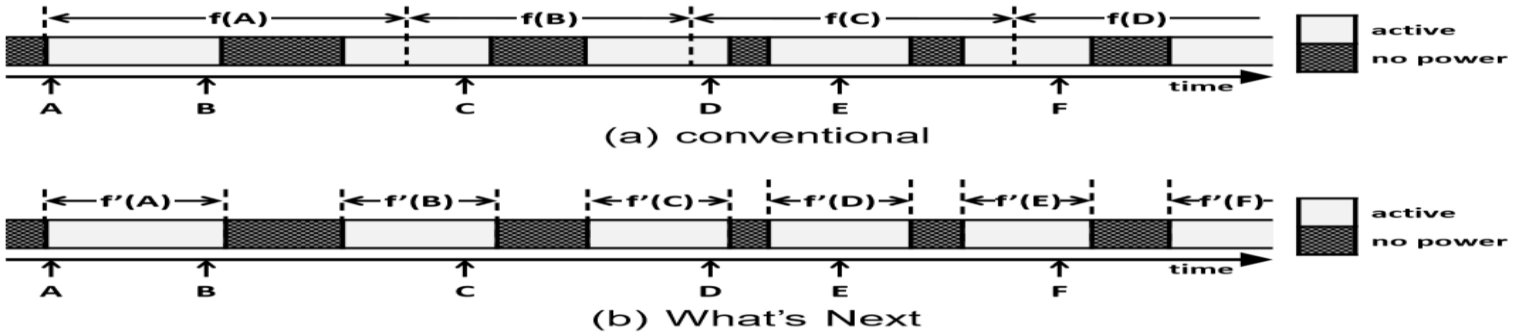
- **eQASM: An Executable Quantum Instruction Set Architecture**
  Xiang Fu (QuTech, Delft University of Technology; Quantum Computer Architecture Lab, Delft University of Technology); Leon Riesebos (Quantum Computer Architecture Lab and QuTech, Delft University of Technology); M. A. Rol (QuTech, Delft University of Technology; Kavli Institute of Nanoscience, Delft University of Technology); Jeroen van Straten (Computer Engineering Lab, Delft University of Technology); Hans van Someren, Nader Khammassi, and Imran Ashraf (Quantum Computer Architecture Lab and QuTech, Delft University of Technology); Raymond Vermeulen (QuTech, Delft University of Technology; Kavli Institute of Nanoscience, Delft University of Technology); Vincent Newsum and Kelvin Loh (Netherlands Organisation for Applied Scientific Research (TNO); QuTech, Delft University of Technology); Jacob de Sterke (Topic Embedded Systems; QuTech, Delft University of Technology); Wouter Vlothuizen (Netherlands Organisation for Applied Scientific Research (TNO); QuTech, Delft University of Technology); Raymond Schouten (QuTech, Delft University of Technology; Kavli Institute of Nanoscience, Delft University of Technology); Carmina G. Almudever (Quantum Computer Architecture Lab and QuTech, Delft University of Technology); Leo DiCarlo (QuTech, Delft University of Technology; Kavli Institute of Nanoscience, Delft University of Technology); Koen Bertels (Quantum Computer Architecture Lab and QuTech, Delft University of Technology)

- **Reliability Evaluation of Mixed-Precision Architectures**
  Fernando Fernandes dos Santos, Daniel Oliveira, Caio Lunardi, Fabiano Pereira Libano, and Paolo Rech (UFRGS)

- **Architecting Waferscale Processors - A GPU Case Study**
  Saptadeep Pal (UCLA); Daniel Petrisko and Matthew Tomei (UIUC); Puneet Gupta and Subramanian S. Iyer (UCLA); Rakesh Kumar (UIUC)

The What's Next Intermittent Computing Architecture

- Energy harvesting devices that run on non-conventional sources of energy like solar, RF, piezo-electricity, operate in the *intermittent computing* paradigm (frequent power outages).
- The systems use non-volatile processors or periodic checkpointing.
- When new data arrives for processing, need to decide whether to continue processing the old data or start with the new one.
- Provides partial answer (approximate) when energy is limited but can refine answer when more energy is available.
- Uses *interruptibility* (processing can be halted while still providing an approximate output) and *flexibility* (if greater accuracy required then can run longer) from *anytime automation model.*



- Process subwords from MSB to LSB, processing each subword generates approximate results.
- Two basic methods for supporting subword operations: *subword pipelining* and *subword vectorization*
- Subword pipelining decomposes high-latency operations like (multiplication) into smaller subwords operations.
- Subword vectorization fuse low-latency instructions (add, load, store) such the most significant subwords of different data elements are processed in parallel.
- Skim points introduced to decouple restore PC from backup PC to skip remaining subwords if the approximate result is good enough.

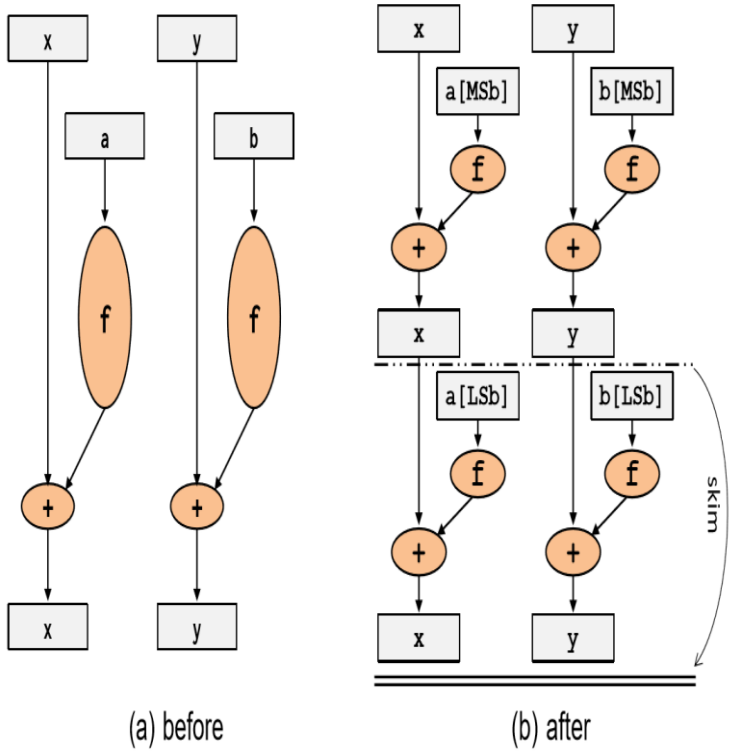


Figure 5: IR transformation for anytime SWP.

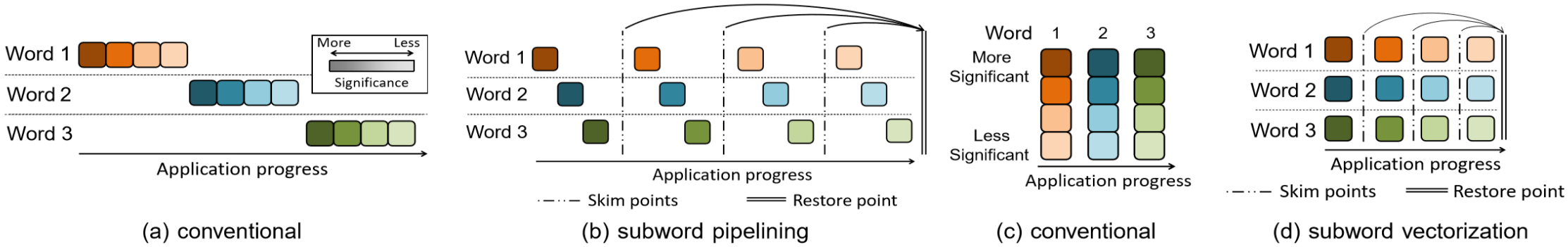


Figure 4: Anytime subword pipelining and vectorization for long-latency and short-latency operations.

eQASM: An Executable Quantum Instruction Set Architecture

- Need to provide a reliable and convenient instruction set architecture for bridging the gap between quantum hardware and software.

- Previous suggested quantum microinstruction set (QuMIS) suffers from multiple issues:-
  - Doesn't support feedback on QuBit measurement results, which is required for active reset in circuit-model-based quantum computing
  - Has low instruction information density
  - Have limited flexibility because it is very low-level and is bound to electronic hardware implementation

- Thus this paper proposes eQASM based on QISA, that contains both quantum and classical instructions.

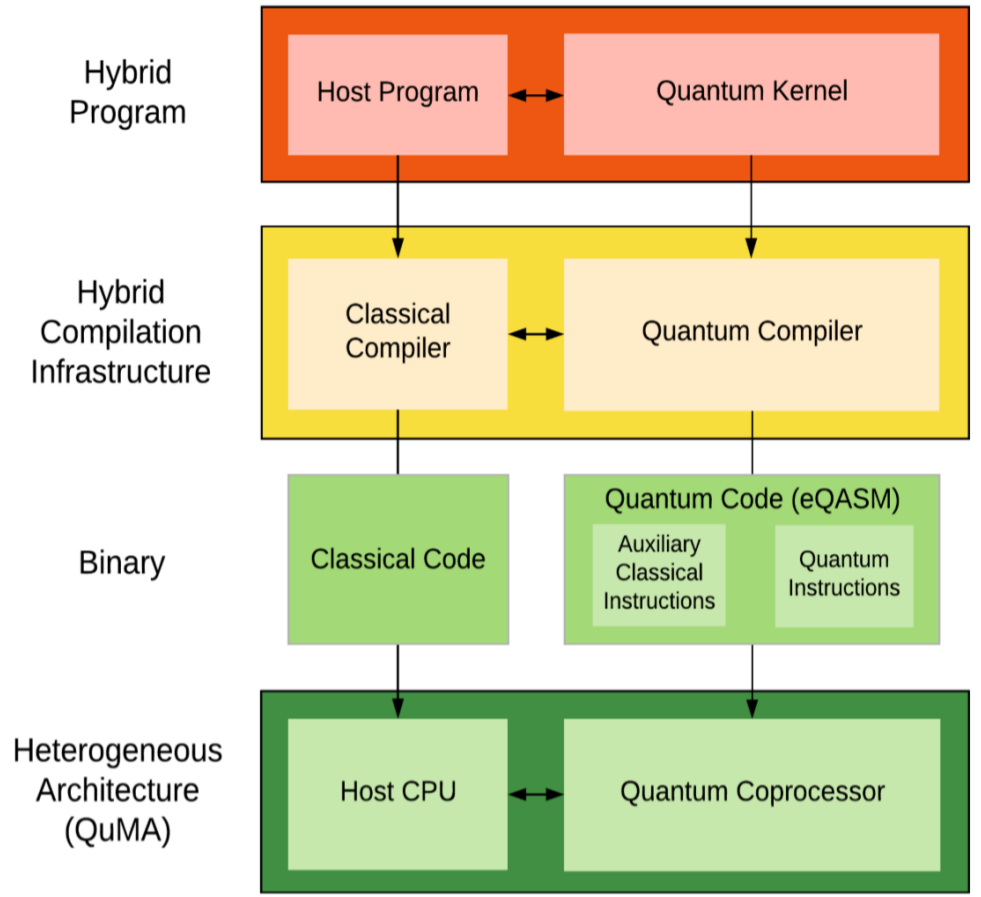


Fig. 1. Heterogeneous quantum programming and compilation model.

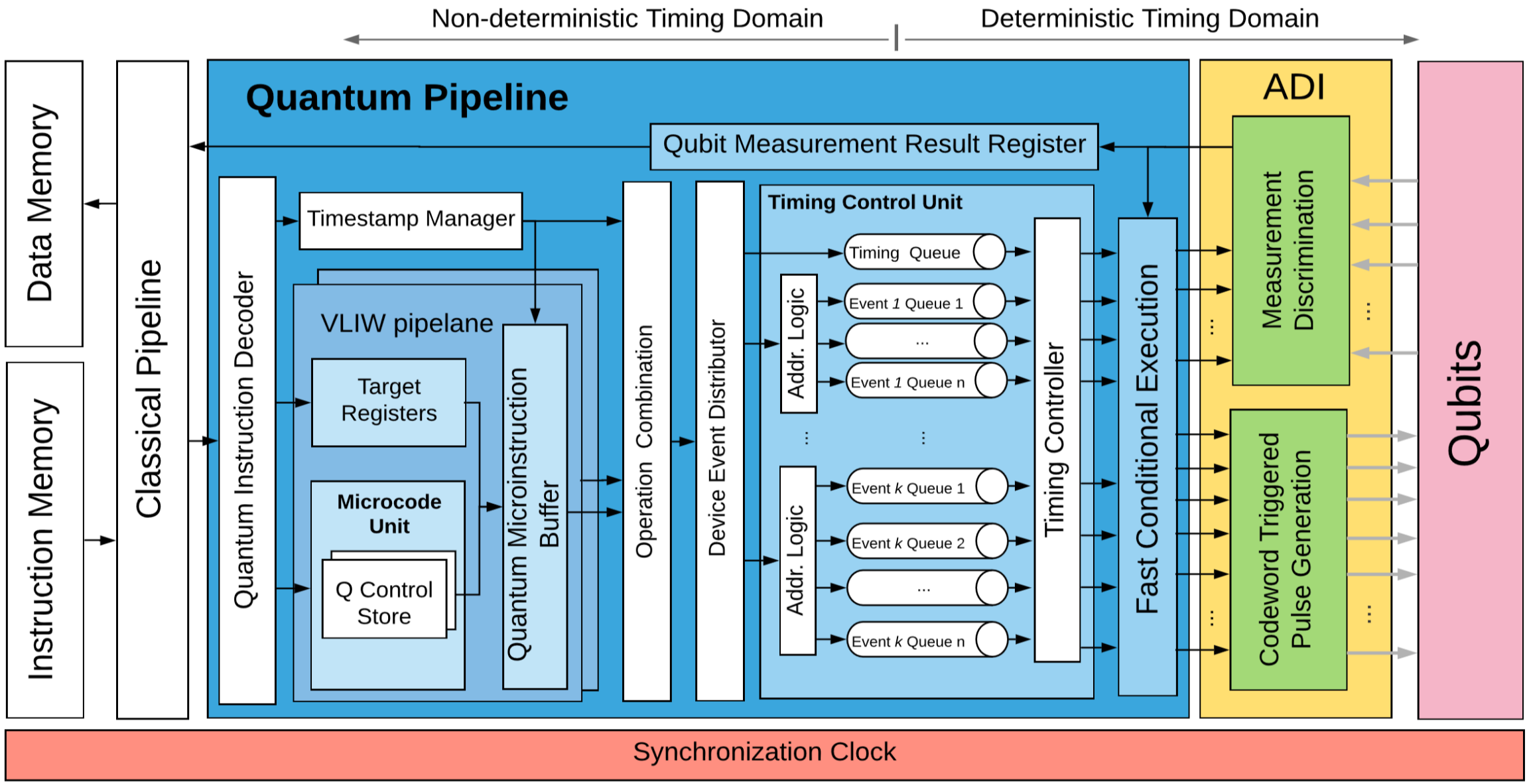


Fig. 9. Quantum microarchitecture implementing the instantiated eQASM for the seven-qubit superconducting quantum processor.

Table 1. Overview of eQASM Instructions.

Type	Syntax	Description
Control	`CMP Rs, Rt`	Compare GPR Rs and Rt and store the result into the comparison flags.
	`BR <Comp. Flag>, Offset`	Jump to PC + Offset if the specified comparison flag is '1'.
Data Transfer	`FBR <Comp. Flag>, Rd`	Fetch the specified comparison flag into GPR Rd.
	`LDI Rd, Imm`	Rd = sign_ext(Imm[19..0], 32).
	`LDUI Rd, Imm, Rs`	Rd = Imm[14..0]::Rs[16..0].
	`LD Rd, Rt(Imm)`	Load data from memory address Rt + Imm into GPR Rd.
	`ST Rs, Rt(Imm)`	Store the value of GPR Rs in memory address Rt + Imm.
	`FMR Rd, Qi`	Fetch the result of the last measurement instruction on qubit i into GPR Rd.
Logical	`AND/OR/XOR Rd, Rs, Rt` `NOT Rd, Rt`	Logical and, or, exclusive or, not.
Arithmetic	`ADD/SUB Rd, Rs, Rt`	Addition and subtraction.
Waiting	`QWAIT Imm` `QWAITR Rs`	Specify a timing point by waiting for the number of cycles indicated by the immediate value Imm or the value of GPR Rs.
Target Specify	`SMIS Sd, <Qubit List>` `SMIT Td, <Qubit Pair List>`	Update the single- (two-)qubit operation target register Sd (Td).
Q. Bundle	`[PI,] Q_Op [| Q_Op]*`	Applying operations on qubits after waiting for a small number of cycles indicated by PI.

Contributions:-

- Can be supported using OpenCL, different operations SOMQ (Single Operation Multiple QuBits), FMR (Fetch Measurement Result)
- Runtime feedback: fast conditional execution for simple and fast feedback and comprehensive feedback control (CFC) for user-defined feedback
- Increase the quantum operation issue rate
- Configurable QISA at compile time

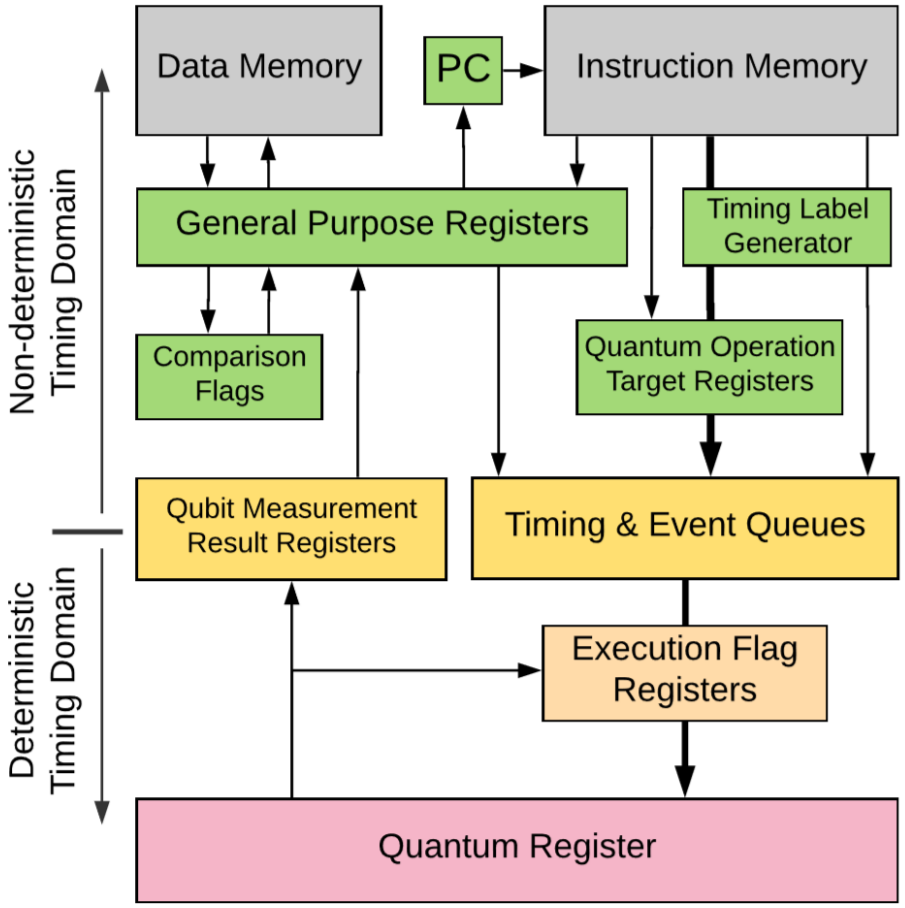
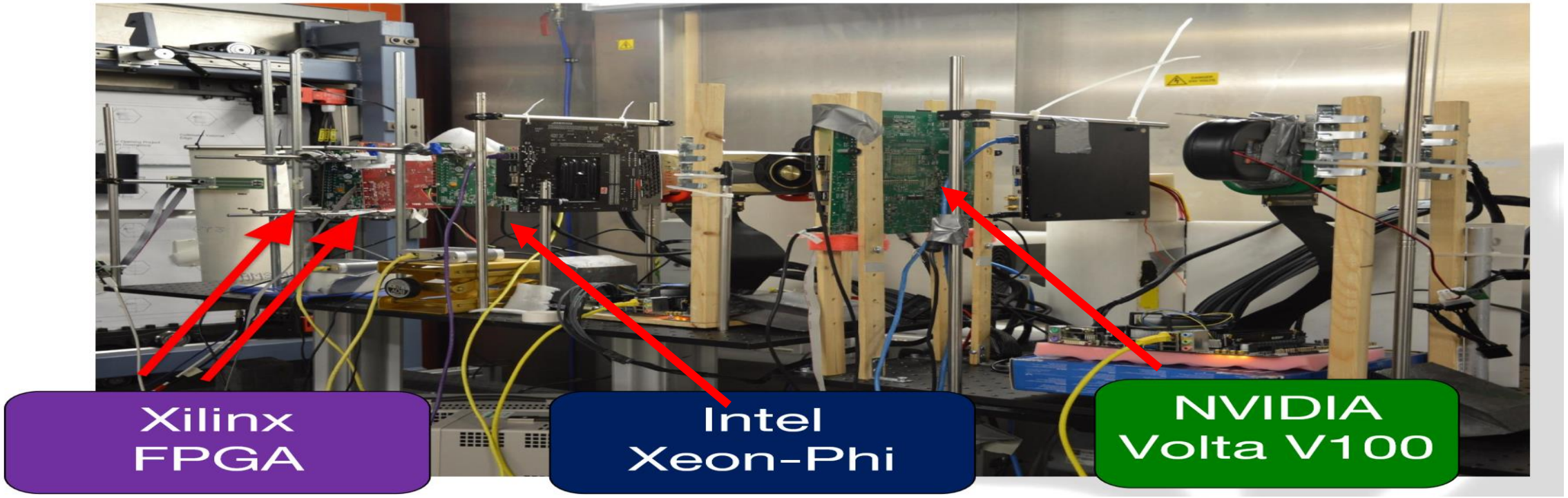


Fig. 2. Architectural state of eQASM. Arrows indicates the possible information flow. The thick arrows represent quantum operations, which reads information from the modules passed through.

Reliability Evaluation of Mixed-Precision Architectures

- Floating points helps improve graphics quality, physical simulation, etc (the more precise, the better).
- Precision comes with a cost, including area (circuit complexity), power consumption and execution time.
- Approximate computing proves that some operations in some algorithms can be approximated without affecting the results significantly.
- Mixed-precision architectures improve performance and energy savings. Helpful mostly in neural network training and execution.

- How is system reliability affected in Mixed-precision environments?
- Cosmic rays produce Muons, protons, neutrons, Gamma rays, etc, that can intorduce soft errors (bit-flips in memory/logic error, not permanent)
- Use FPGA, X86(Xeon Phi) and GPU (Volta V100) for evaluation
- Fault injection using Neutron beam, measure Failure in time (FIT)



Observations

- Doubles have a much larger area than half, thus more likely to be corrupted.

- However, a fault in double value is much less critical than a fault in half
  - Double 81% bits are mantissa, half 60% are mantissa

- Case study for FPGA, Xeon Phi, and GPU (find the rate of decrement of FIT rate)
  - FPGA Half-precision FIT reduction is slower: faults are more critical
  - Xeon Phi doesn’t have dedicated HW for double/single precision, the compiler decides how to use the functional units, single has higher error rates, but criticality is similar
  - GPUs, error criticality increases with a decrease in precision

- In general low precision can improve reliability: more data can be processed before experiencing an error.

Architecting Waferscale Processors – A GPU Case Study

- Parallel hardware needs low overhead communication, but the area and power overhead for chip-to-chip communication is 25-30%
- Waferscale processors help alleviate issues of communication overheads while scaling.
- Historically had yield issues, but with recent integration technology such as Silicon-Interconnect Fabric (Si-IF) (pre-manufactured dies are bonded to a silicon wafer) it seems possible.
- The wafer is the processor, either monolithic processor or a set of processors are designed on the wafer and are connected using low-cost on-wafer interconnect.

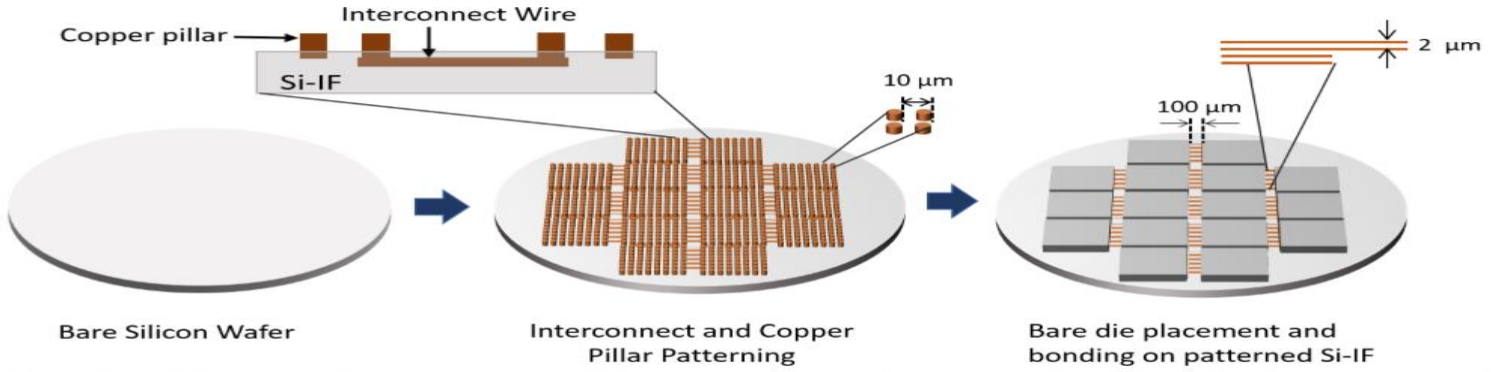


Fig. 3: The system assembly process flow is shown. Interconnect layers and copper pillars are made by processing the bare silicon wafer. Bare dies are then bonded on the wafer using TCB

Contributions of the paper

- Answers the question whether building a waferscale the GPU system on 300mm wafer with 100 GPU modules possible.

- Waferscale GPUs are area-constrained due to power delivery network overheads, not thermally-constrained.

- Techniques for thread group scheduling and data partitioning.

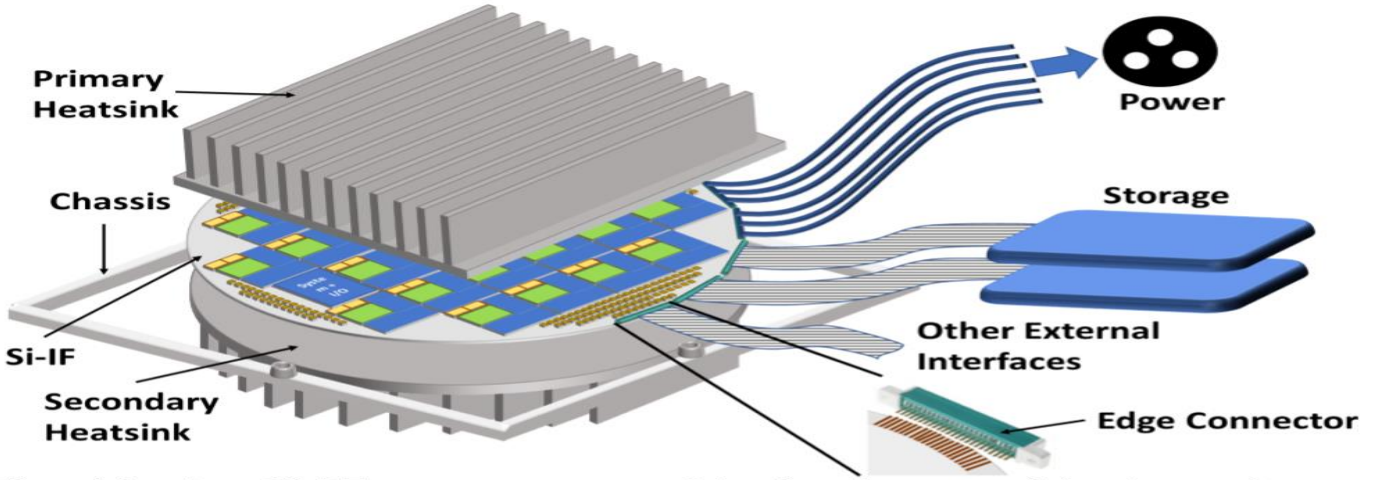


Fig. 10: An Si-IF system assembly is shown with the primary and backside secondary heat sinks. The whole system is bolted to a chassis. The host CPU could either be connected externally or reside on the wafer itself.

- Performance and energy efficiency depends on how the compute threads and data is placed on the waferscale chip.

- Consecutive Thread Block (TB) can benefit from data locality; distributed scheduling used rather than centralized round-robin scheduling.

- TBs placed on GPM array starting from the corner and moving row-wise.

- First memory access to a page moves that page to the local DRAM of the GPM.

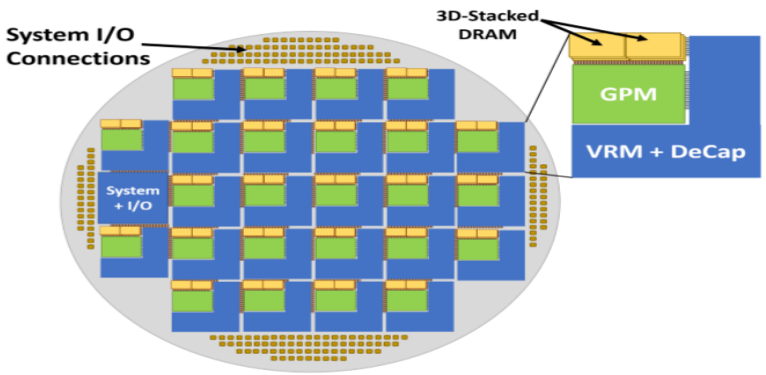


Fig. 11: Waferscale GPU with 25 GPM units (1 redundant unit) comprising of two 3D-stacked DRAM per unit, VRM unit and decoupling capacitors.

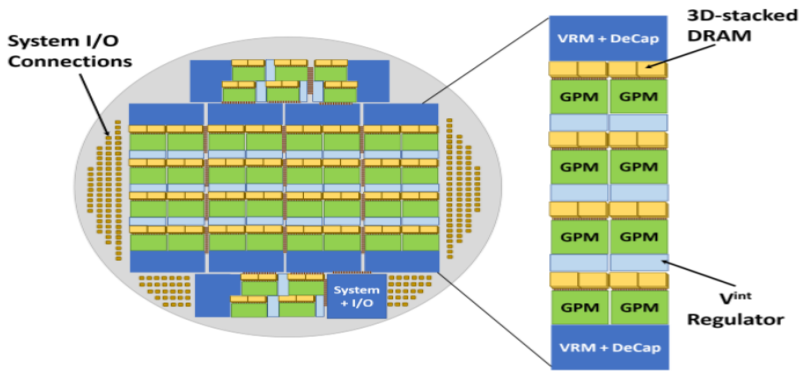


Fig. 12: Waferscale GPU with 42 GPM units (2 redundant units) comprising of two 3D-stacked DRAM per unit, VRM unit and decoupling capacitors.

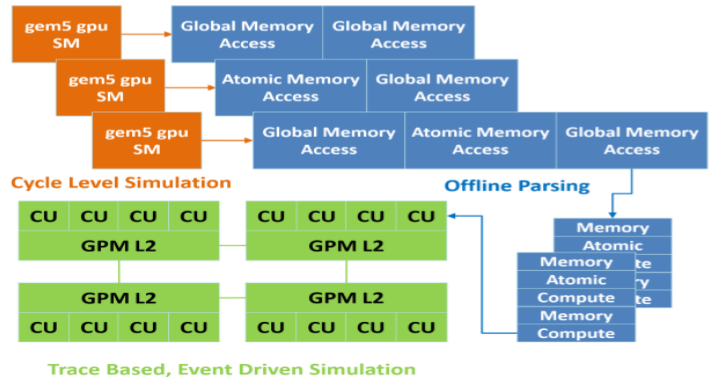


Fig. 13: Simulator Workflow

Session 4B: Industry Session 1, Mobile and Low power

- **Killi: Runtime Fault Classification to Deploy Low Voltage Caches without MBIST**
  Shrikanth Ganapathy (AMD Research), John Kalamatianos (AMD Research), Brad Beckmann (AMD Research), Steven Raasch (AMD Research), Lukasz Szafaryn (Intel)

(SRAM cells fail exponentially at low-voltages, Memory Built-in-Self Tests are not scalable for large shared L2 , completely rely on on-line adaptive Low-voltage technique to classify faults)

- **Gables: A Roofline Model for Mobile SoCs with Many Accelerators and Ceilings**
  Mark Hill (Google), Vijay Janapa Reddi (Google) ("From a department chair to an intern at Google is a promotion", roofline model for accelerators, pre-silicon model for understanding usage of IPs and maximum performance that can be achieved for workloads)

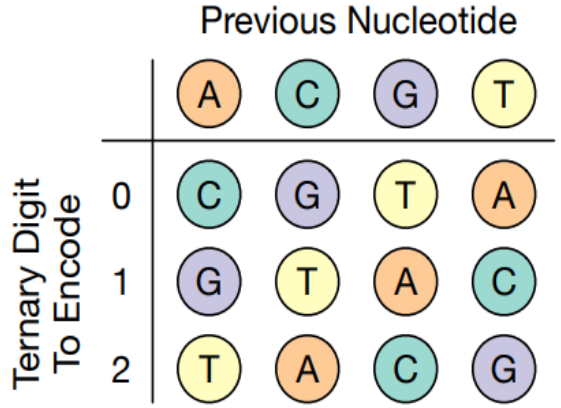
- **Machine Learning at Facebook: Understanding Inference at the Edge**
  Carole-Jean Wu (Facebook), David Brooks (Facebook), Kevin Chen (Facebook), Douglas Chen (Facebook), Sy Choudhury (Facebook), Marat Dukhan (Facebook), Kim Hazelwood (Facebook), Eldad Isaac (Facebook), Yangqing Jia (Facebook), Bill Jia (Facebook), Tommer Leyvand (Facebook), Hao Lu (Facebook), Yang Lu (Facebook), Lin Qiao (Facebook), Brandon Reagen (Facebook), Joe Spisak (Facebook), Fei Sun (Facebook), Andrew Tulloch (Facebook), Peter Vajda (Facebook), Xiaodong Wang (Facebook), Yanghan Wang (Facebook), Bram Wasti (Facebook), Yiming Wu (Facebook), Ran Xian (Facebook), Sungjoo Yoo (Facebook), Peizhao Zhang (Facebook)

PPoPP keynote: When Moore met Feynman: Ultra-dense data storage and extreme parallelism with electronic-molecular systems (Karin Strauss, Microsoft Research)

- Data storage for huge data centers is becoming a concern, need high-density storage devices that can alleviate the real-estate needs of data storage.
- Digital files are segmented and encoded into different sequences.
- Individual files can be retrieved from a mixed database using PCR (Polymerase chain reaction)-based random access.

Binary data	P 01010000	o 01101111	l 01101100	y 01111001	a 01100001	; 00111011
Base 3 Huffman code	12011	02110	02101	222111	01112	222021
DNA nucleotides	GCGAG	TGAGT	ATCGA	TGCTCT	AGAGC	ATGTGA

(a) Translating binary data to DNA nucleotides via a Huffman code.



(b) A rotating encoding to nucleotides avoids homopolymers (repetitions of the same nucleotide), which are error-prone.

Figure 5. Encoding a stream of binary data as a stream of nucleotides. A Huffman code translates binary to ternary digits, and a rotating encoding translates ternary digits to nucleotides.

A Content-Addressable DNA Database with Learned Sequence Encoding

- Pair-wise matching forms complete double helix structure as some of the (A, G, T, C) attract and some repel.

```
ATTGCAGTGATCG      ATTGCAGT-GATCG
|||||    |||||      |||||| |||||
ATTGCGTCGATCG      ATTGC-GTCGATCG
```

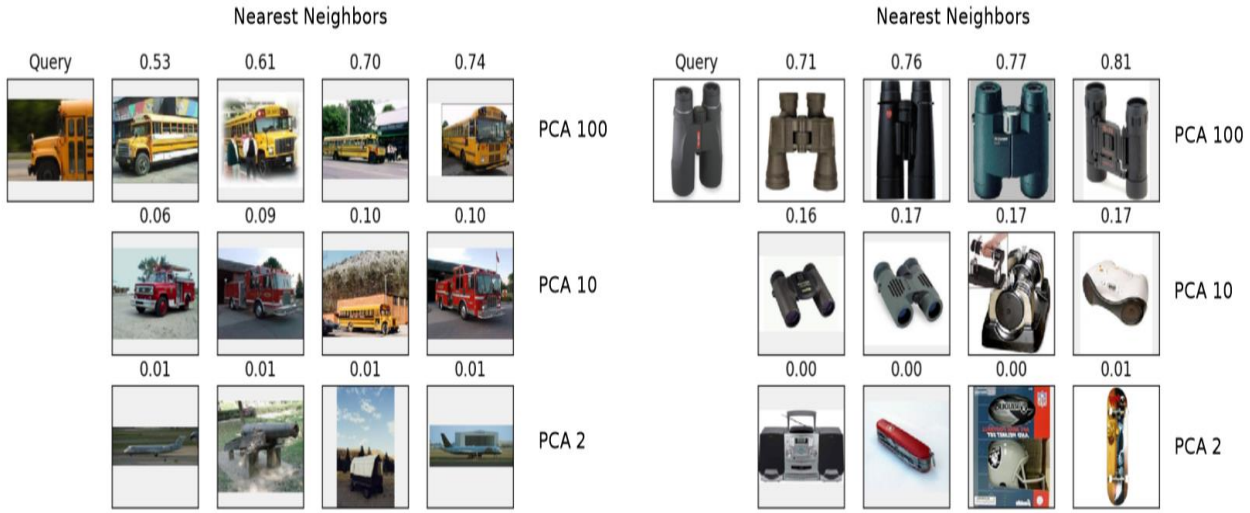


Fig. 1: A pair of sample queries from the Caltech-256 dataset, showing the four nearest neighbors in three different feature spaces. Each neighbor is annotated with its Euclidean distance to the query in that space.

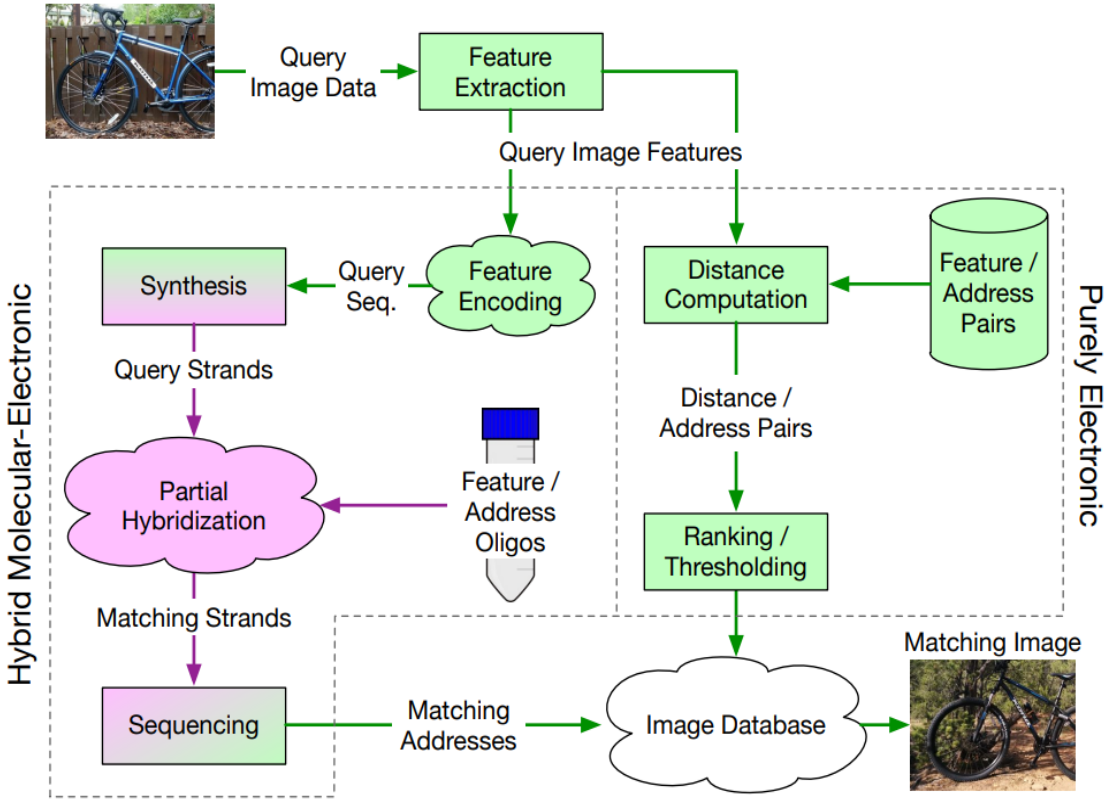


Fig. 7: Stack diagram for a hybrid and a purely electronic content-based image retrieval system. Electronic components are green; molecular components are pink.

Session 5B: Memory Hierarchy management

- **Bingo Spatial Data Prefetcher** *Mohammad Bakhshalipour (Sharif University of Technology); Mehran Shakerinava (Sharif University of Technology); Pejman Lotfi-Kamran (Institute for Research in Fundamental Sciences (IPM)); Hamid Sarbazi-Azad (Sharif University of Technology)*

- **NoMap: Speeding-Up JavaScript Using Hardware Transactional Memory**
  Thomas Shull and Jiho Choi (University of Illinois Urbana-Champaign); Maria Garzaran (Intel); Josep Torrellas (University of Illinois Urbana-Champaign)

- **FUSE: Fusing STT-MRAM into GPUs to Alleviate Off-Chip Memory Access Overheads**
  Jie Zhang and Myoungsoo Jung (Yonsei University); Mahmut Kandemir (Penn State University)

- **Featherlight Reuse-distance Measurement** *Qingsen Wang (College of William & Mary); Millind Chabbi (Uber); Xu Liu (College of William & Mary)*

Bingo Spatial Data Prefetcher

- Spatial data prefetchers exploit spatial locality patterns to prefetch data.
- Relies on spatial address correlation: the similarity of access patterns among multiple pages of memory.
- Record accesses to a page (record a footprint), footprints associated to event(s).
- Using a single event (say PC) instead of (PC+offset, PC+Address , etc) can be efficient but may not be highly accurate.
- TAGE (TAgged GEometric length)-like tables occupy space and may have redundant information.
- Propose a single history table that is looked up multiple times to find prediction with the longest event sequence.

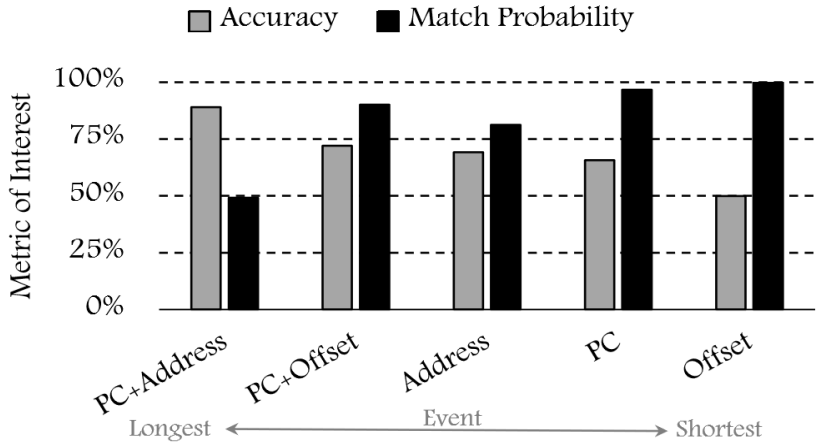


Figure 2. Accuracy and match probability of various heuristics as the event to which access history of pages are associated.

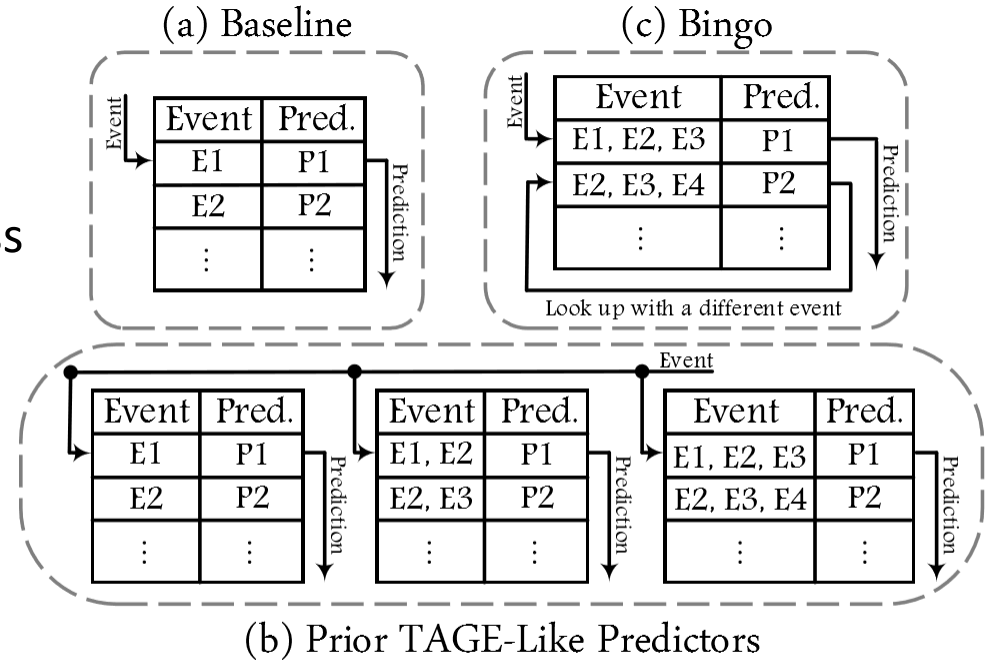


Figure 1. Comparison of proposals for predictor-based hardware optimizers.

- The idea is that short events are carried in long events.
- History table stores history of long events but is looked up using both short and long events.
- Table indexed with the hash of short events but tagged with the longest event.
- The history table is looked up with the longest event if match found then the prediction is made, otherwise, the table is looked up again with the next-longest event in the same set as the set is indexed using short events.

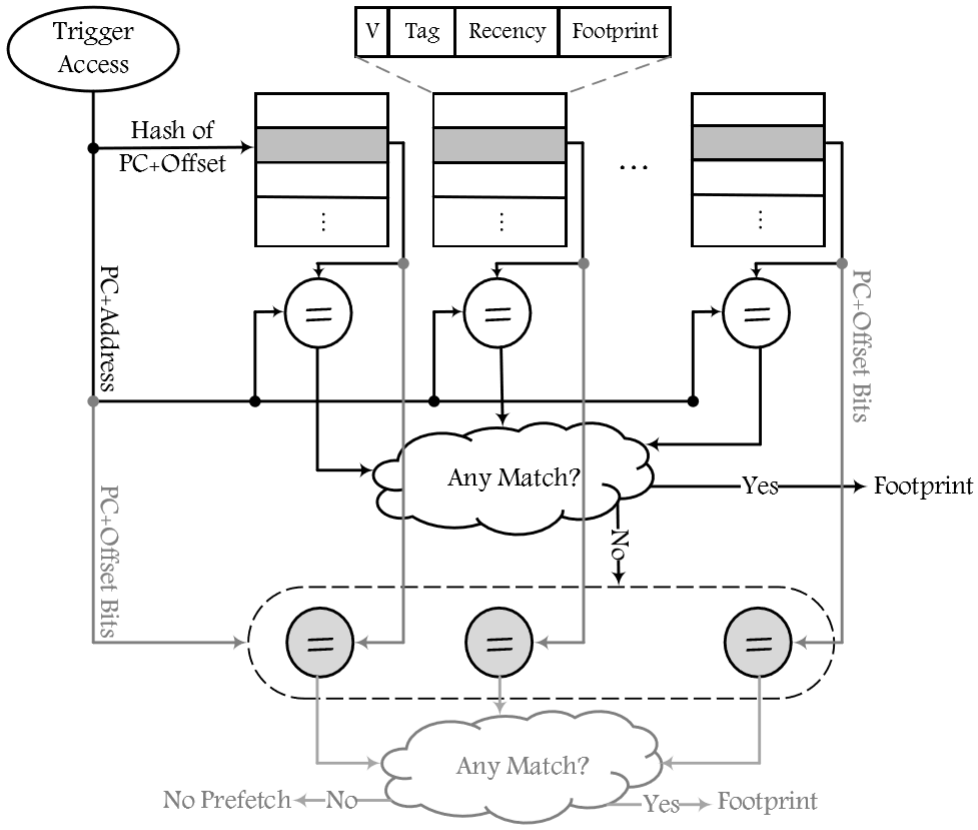


Figure 5. The details of the history table lookup in Bingo prefetcher. Gray parts indicate the case where lookup with long event fails to find a match. Each large rectangle indicates a physical way of the history table.

FUSE: Fusing STT-MRAM into GPUs to Alleviate Off-Chip Memory Access Overheads

- GPUs achieve outstanding performance with low power, they employ streaming multiprocessors (SMs) with large private register files.
- Register files occupy 62% of the total private memory. The on-chip L1D cache also suffers from thrashing due to irregular access patterns and task switches.
- Employing a larger L1D cache can help in reducing off-chip access to the DRAM and improve performance, but on-chip area remains a bottleneck.
- Use NVM technology like Spin-Transfer Torque Magnetic RAM which offer higher density to build bigger L1D caches.

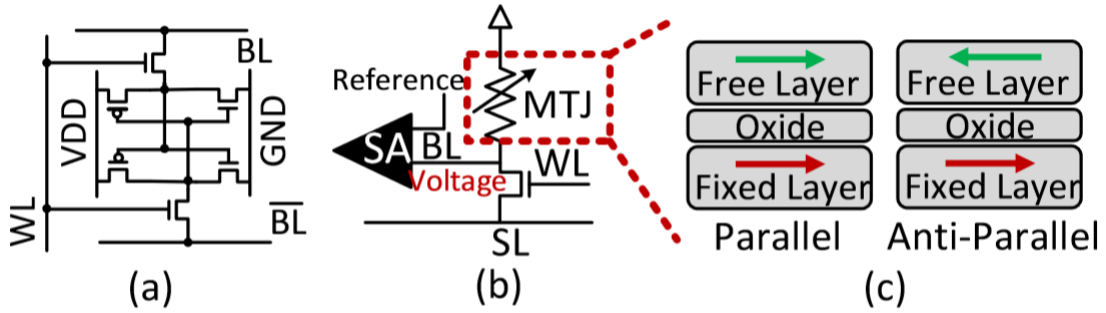


Figure 4: (a) SRAM memory cell, (b) STT-MRAM memory cell and (c) MTJ structure.

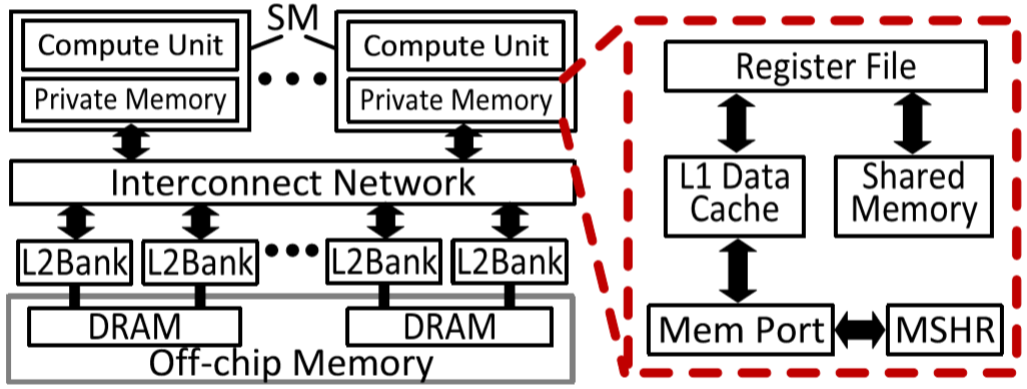


Figure 2: GPU baseline architecture.

Contributions of the paper

- Efficient heterogeneous GPU Caches: STT-MRAM banks along with SRAM banks. STT-MRAM suffer from high write latency, thus require swapping mechanism to faster SRAM when hot blocks are detected.
- WORM (Write once read multiple) blocks can be speculatively predicted and kept in the larger STT-MRAM banks, the majority of data blocks (90%) under GPU applications experience this.
- Smart data placement strategy: use a read-level predictor that detects WORM data blocks.
- Approximate STT-MRAM as fully set-associative, use a counting bloom filter for faster lookup.

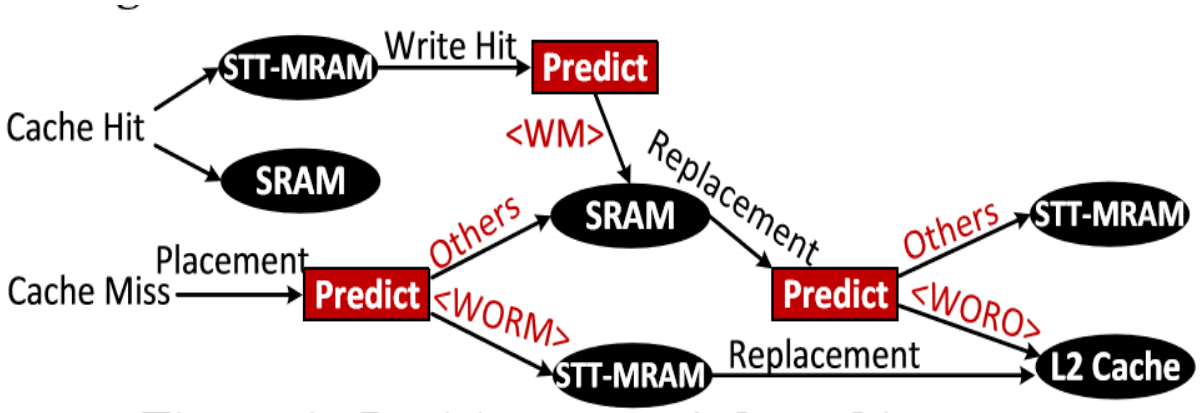


Figure 9: Decision tree of the arbitrator.

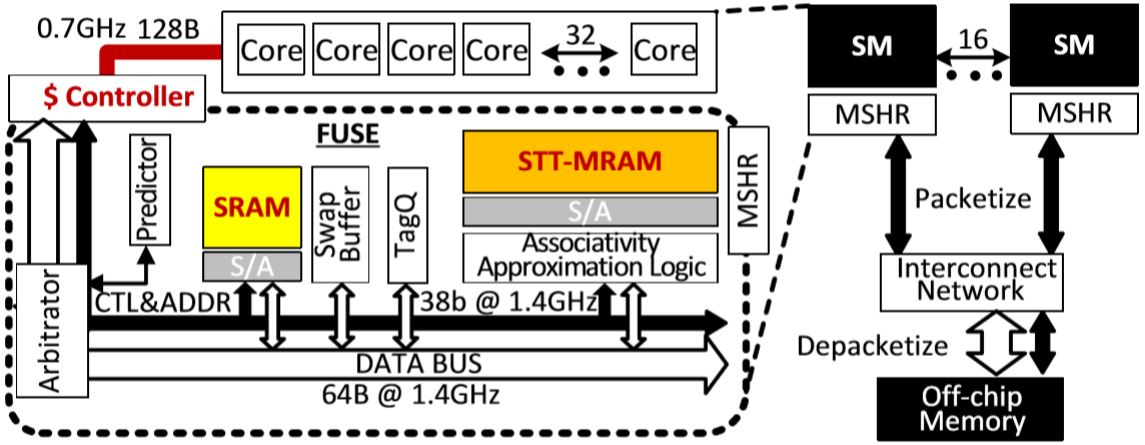


Figure 5: High-level view of the proposed FUSE.

Featherlight Reuse-distance Measurement

- Reuse distance: the number of distinct memory location between two consecutive uses of the same memory location.

- Reuse distance used in quantifying data locality (cache simulation, code optimization, program phase prediction, etc).

- But profiling reuse distance for the whole program is costly (instrumentation 100x to 1000x slow down).

- Solution use RDX :-
  - Sampling based profiler
  - Low overhead (5% performance (time)) (7% memory)
  - High accuracy > 90%

- Three basic components:-
  - Sample Memory access address
    - Use Performance Monitoring Units to sample loads and stores
    - Record effective address of each access
  - Measure time distance (number of memory accesses since last use)
    - Use debug registers to detect the reuse position
  - Time distance -> reuse distance
    - Each data location is accessed independently
    - Statistically estimate reuse distance histogram from time distance

Locality Approximation Using Time (POPL'07)	
Assumption	A data element is accessed independently from others, which is a Bernoulli process.
Input	Time distance histogram, max working size
Output	Stack distance histogram

Challenges:-

- Sampling may miss opportunities.
- Deal with limited number of debug registers, use Reservoir sampling, if a free register is available use it otherwise probabilistically replace one of the monitored addresses.

Session 6A: Industry Session 2, Microarchitecture

- **Efficient Load Value Prediction using Multiple Predictors and Filters**
  Rami Sheikh (Qualcomm), Derek Hower (Qualcomm)

- **BRB: Mitigating Branch Predictor Side-Channels**
  Ilias Vougioukas (ARM Research/U. of Southampton), Nikos Nikoleris (ARM Research), Andreas Sandberg (ARM Research), Stephan Diestelhorst (ARM Research), Bashir M. Al-Hashimi (U. of Southampton), Geoff V. Merrett (U. of Southampton)

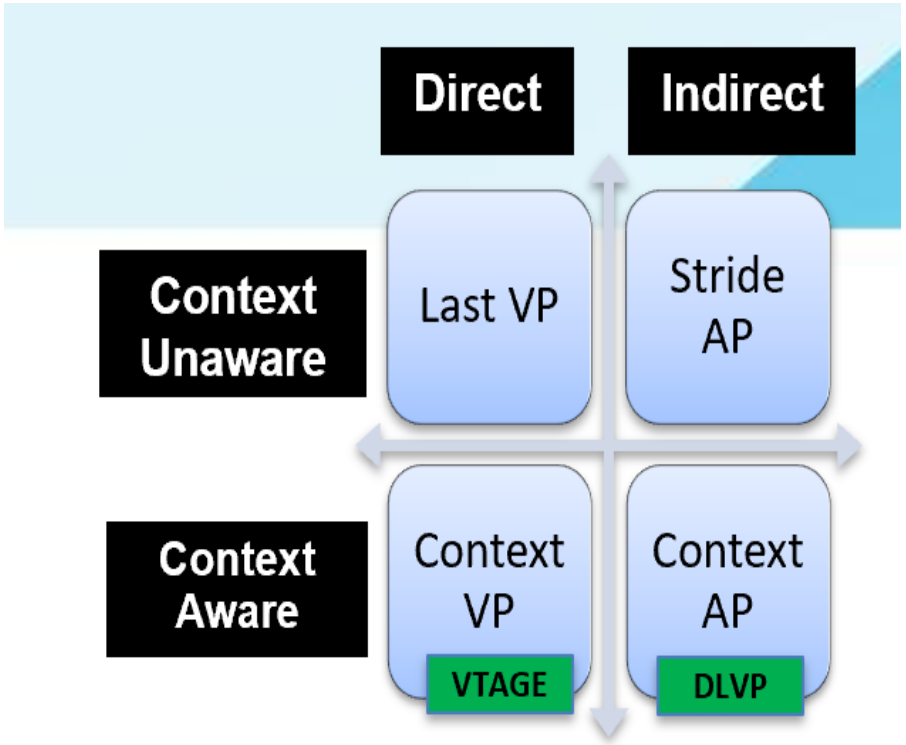
- **Elastic Instruction Fetching**
  Arthur Perais (Qualcomm), Rami Sheikh (Qualcomm), Luke Yen (Qualcomm), Michael McIlvaine (Qualcomm), Robert D. Clancy (Qualcomm)

Efficient load value predictor using Multiple predictors and filters

- Value prediction helps in breaking true data dependency, thus improving performance.

- Most value predictors have high budget requirements, and no single value prediction scheme can predict all load values effectively.

- Variations in value prediction schemes:-
  - Direct: predict value
  - Indirect: predict address, and read the value from that address
  - Context-aware
  - Context unaware

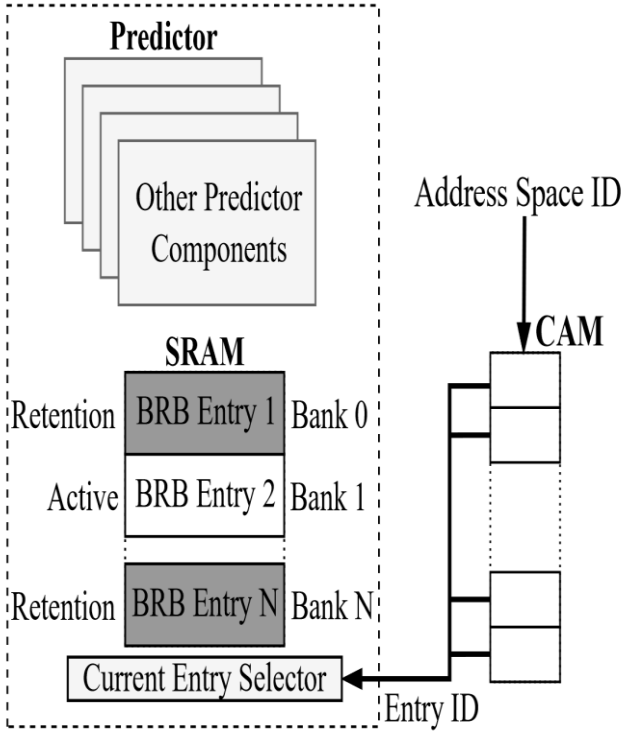


- A composite value predictor design by Predictor Fusion and heterogeneous predictors.

- Optimizations on top of that:-
  - Accuracy monitors: Mitigate pathological cases
    - Track prediction accuracy
    - Squash predictions if inaccurate
  - Smart training: Eliminate overlap
    - Train all predictors, if mispredicts training is necessary
    - Prefer value over address and context-unaware over aware
  - Predictor fusion: improve utilization
    - Identify under delivering predictors and re-purpose them

Mitigating Branch Predictor Side-Channels

- Threat model, victim and attacker run on the same core share the same branch predictor.
- An attacker can poison branch predictor entries (BTBs and other tables) to have victim execute vulnerable code or mispredict.
- Flush BP contents on context switch (expensive) and degrades performance.
- Use Branch Retention Buffer (BRB) to store partial states
  - Retain states per context
  - Activated on ASID



- Focus on components of the TAGE (Tagged Geometric length predictor), and check, saving which components of the predictor help to mitigate performance issues the most (no need to store all states)

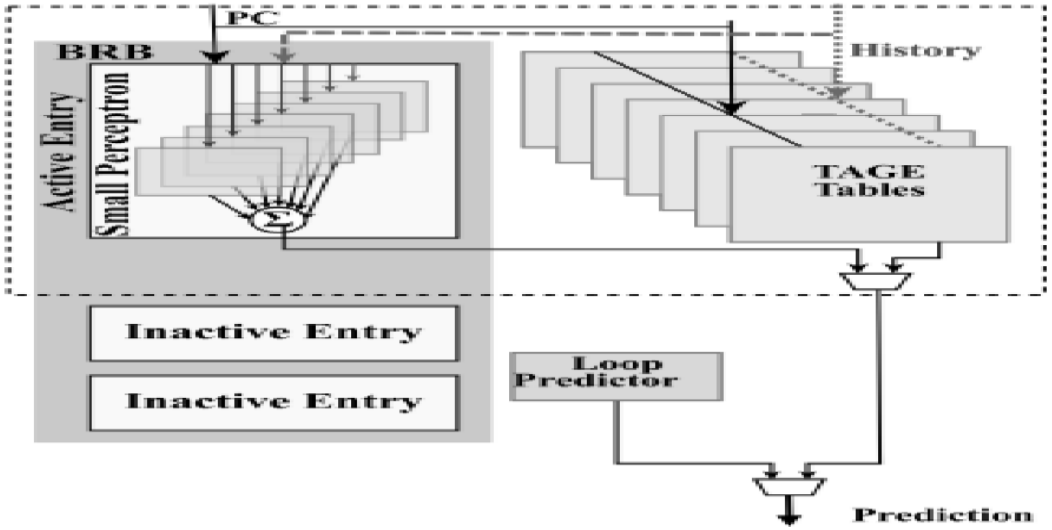
- Retaining no states doubles mispredictions, need to reduce that by selecting which of the components bimodal, loop predictor, stats corrector or the Tage tables.

- Found that the bimodal component helps in saving the most transient states, use branch retention buffer to save it.

- But still, accuracy is not great, need to get a better bimodal component that can retain steady states.

- ParTAGE swaps bimodal for a perceptron.

- Better than TAGE with BRB.



Elastic Instruction Fetching

- Two primary modes of instruction fetching:-
  - Coupled Instruction fetching: On a I-Cache miss next PC generation is stalled.
  - Decoupled Instruction fetching: Branch predictor used to generate next PC, decoupled queue is filled up

- Workloads which have large I-cache footprint benefit from Decoupled instruction fetching.

- But the ones which have smaller I-cache footprint don’t benefit much from the presence or absence of decoupled fetching.

- Thus propose Elastic fetching (ELF) which alternates between two modes of fetching.

Idea behind ELF

- When branch misprediction is corrected, the next correct PC is already available, no need to wait for BP to catch up

- Restart both branch prediction and instruction fetch concurrently
  - Front end behaves in “coupled mode” after pipeline restart
  - Switch to “decoupled mode” when branch predictor catches up
  - Counters to change mode

- Two versions L-ELF (limited) upon decoding a branch fetch stalls, U-ELF (Unlimited)

Session 7B: Microarchitecture

- **R3-DLA (Reduce, Reuse, Recycle): A More Efficient Approach to Decoupled Look-Ahead Architectures**
  Sushant Kondguli and Michael Huang (University of Rochester)

(Look ahead thread helps warm up structures associated with running the main thread, lighter look-ahead thread (36%))

- **Recycling Data Slack in Out-of-Order Cores**
  Gokul Subramanian Ravi and Mikko Lipasti (University of Wisconsin - Madison)

(Clock cycle dedicated to data and environment safety, harness data slack from different operations depending on data width and type, accumulate slack and use efficient scheduling techniques by introducing new entries in reservation stations)

- **Freeway: Maximizing MLP for Slice-Out-of-Order Execution**
  Rakesh Kumar (Norwegian University of Science and Technology (NTNU), Norway); Mehdi Alipour and David Black-Schaffer (Uppsala University, Sweden)

Session 8A: Memory

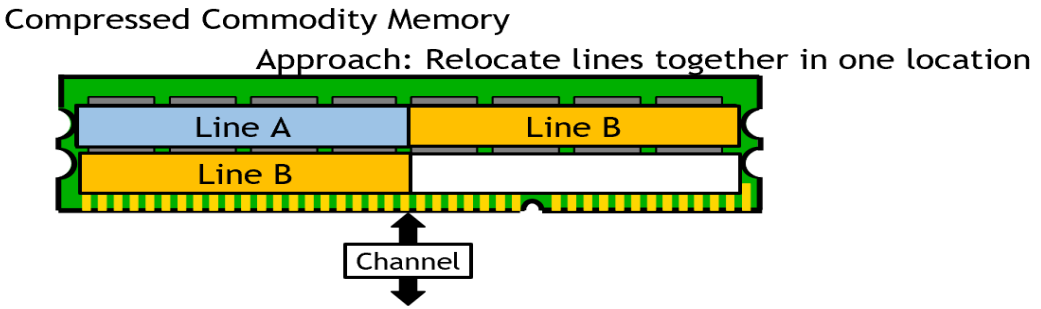
- **Enabling Transparent Memory-Compression for Commodity Memory Systems**
  Vinson Young, Sanjay Kariyappa, and Moinuddin Qureshi (Georgia Institute of Technology)

- **D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput**
  Jeremie S Kim (Carnegie Mellon University; ETH Zurich); Minesh Patel and Hasan Hassan (ETH Zurich); Lois Orosa (ETH Zurich; Universidade Estadual de Campinas); Onur Mutlu (ETH Zurich; Carnegie Mellon University)

- **PageSeer: Using Page Walks to Trigger Page Swaps in Hybrid Memory Systems**
  Apostolos Kokolis, Dimitrios Skarlatos, and Josep Torrellas (University of Illinois, Urbana-Champaign)

Enabling Transparent Memory-Compression for Commodity Memory Systems

- Need practical and efficient solutions for scaling memory bandwidth.
- Compression helps with memory bandwidth scaling but would need OS support as memory requirement and compressibility varies.
- Transparent Memory Compression (TMC): HW compression for Bandwidth scaling without OS support.
- Challenges with TMC:-
  - Transfer in chunks (64 Bytes each) even if data acquiring less space due to compression.
  - If able to relocate lines together then would help reduce bandwidth issues (pair-wise remapping).

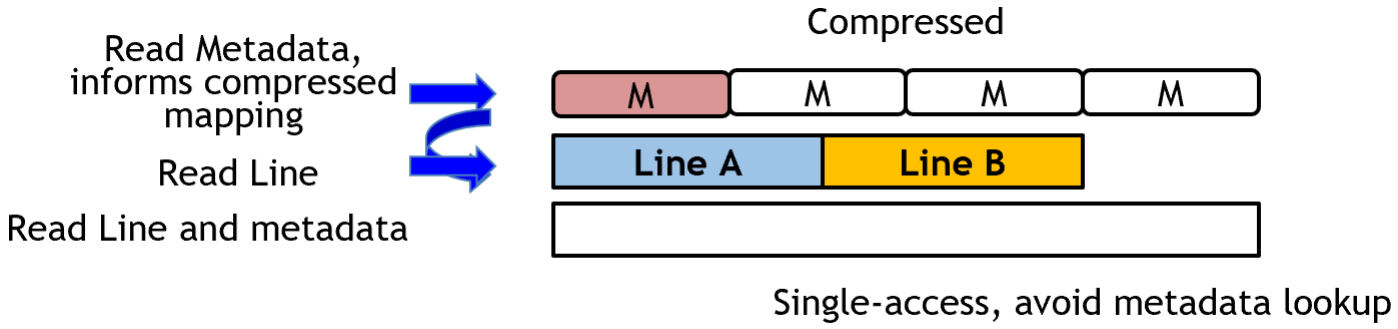


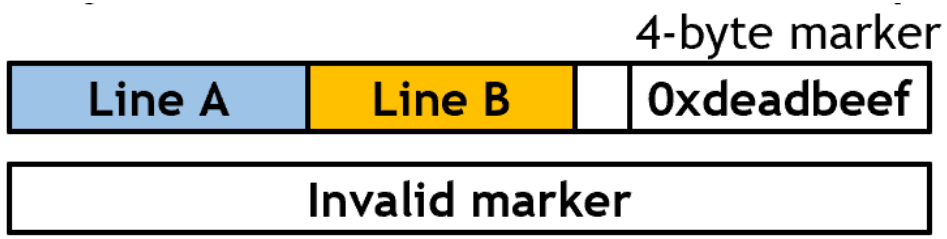
- Can use metadata to store information about the compressed/uncompressed lines, suffers from performance issue as you always need to read meta-data first and then read the appropriate line.

- Storing metadata within the line helps single access reads.

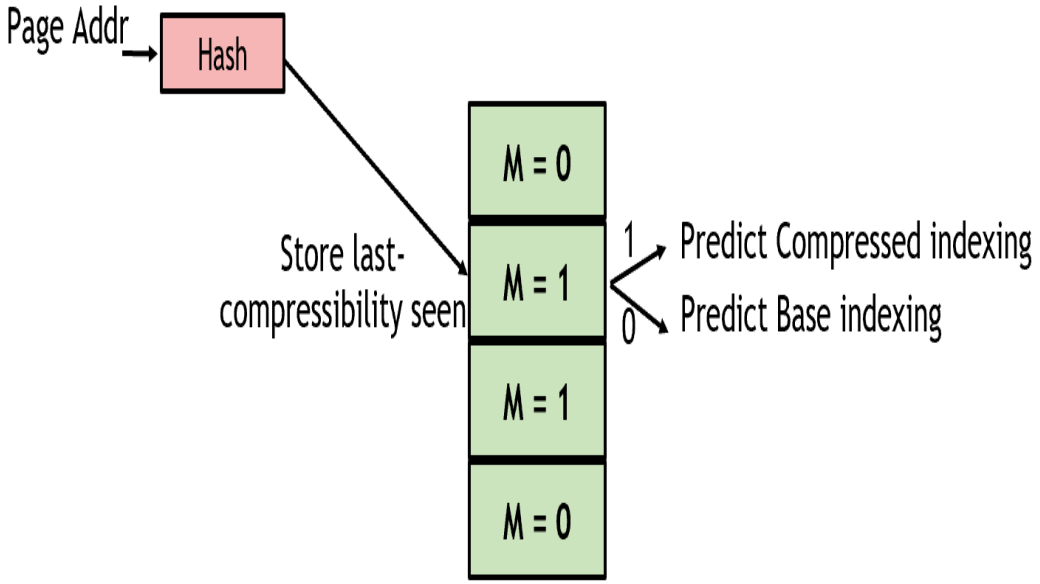
- Use 4-byte maker to denote compressibility of lines (whether the entire line has been used or not)

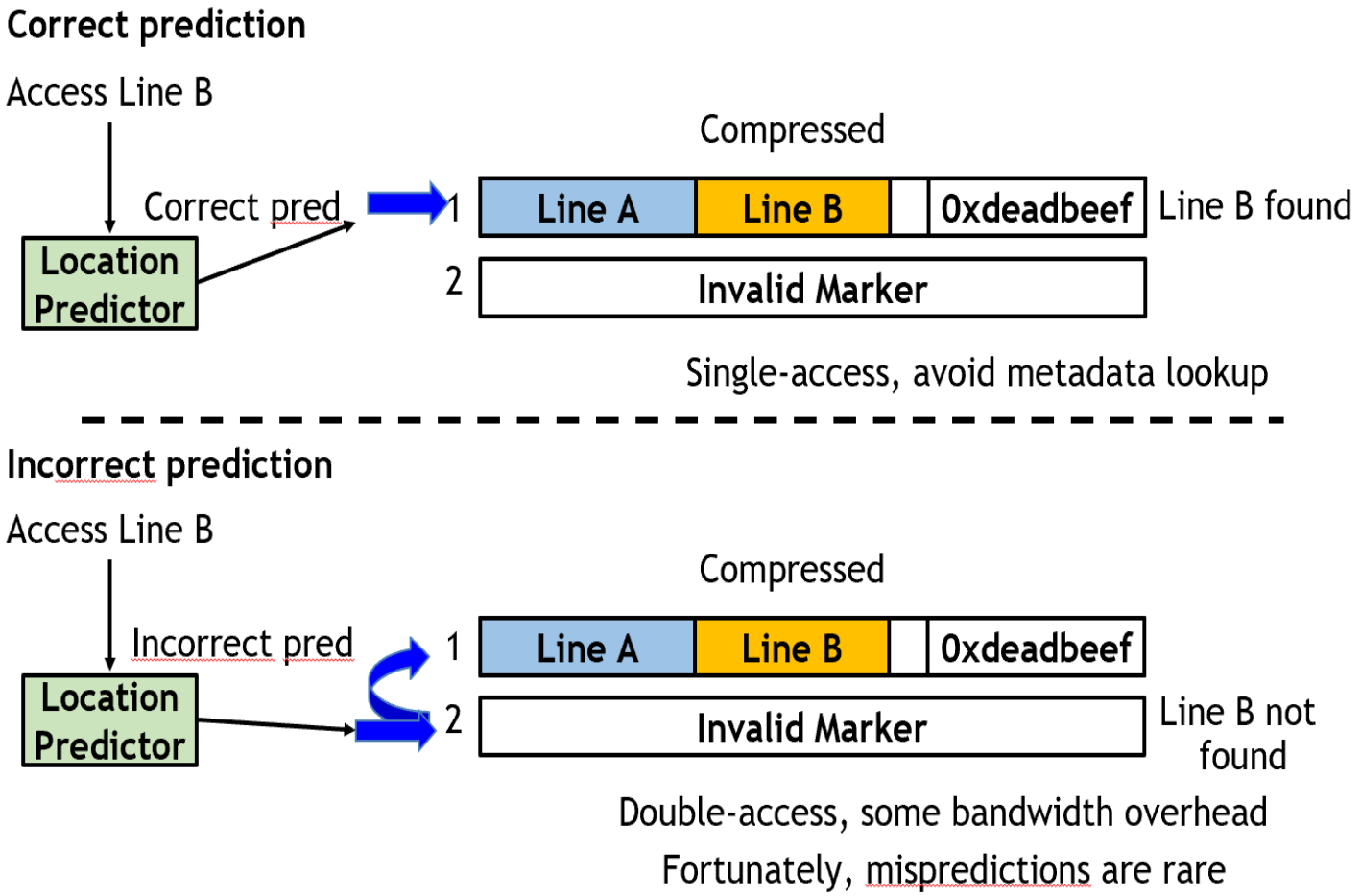
- But uncompressed lines can collide with marker tags, store small SRAM based Collision table for this purpose.





- But how do you find a line with? Reading all possible lines is not a solution.
- Predict location and compressibility of line to enabling reading in a single access.
- Use a hash-based prediction table.





D-Range: Using commodity DRAM devices to generate true random numbers with low latency and high throughput

- Random numbers are used for a variety of security and randomized algorithm applications.
- There is a need for hardware based True Random Number Generator (TRNG) with existing devices.
- True Random Numbers can only be generated using a physical process example radioactive decay, thermal noise, etc.
- Goal is to provide TRNG using DRAM devices, using cell's latency failure probability.
- Latency failure is related to random process variation during manufacturing.

- Random values can be extracted by sampling DRAM cells that fail truly and the cell's latency failure probability.

- Access cells with reduced t_{RCD} (time from activation to read) cause more cells to latency fail.

- Use multiple DRAM banks in parallel to generate high throughput random numbers.

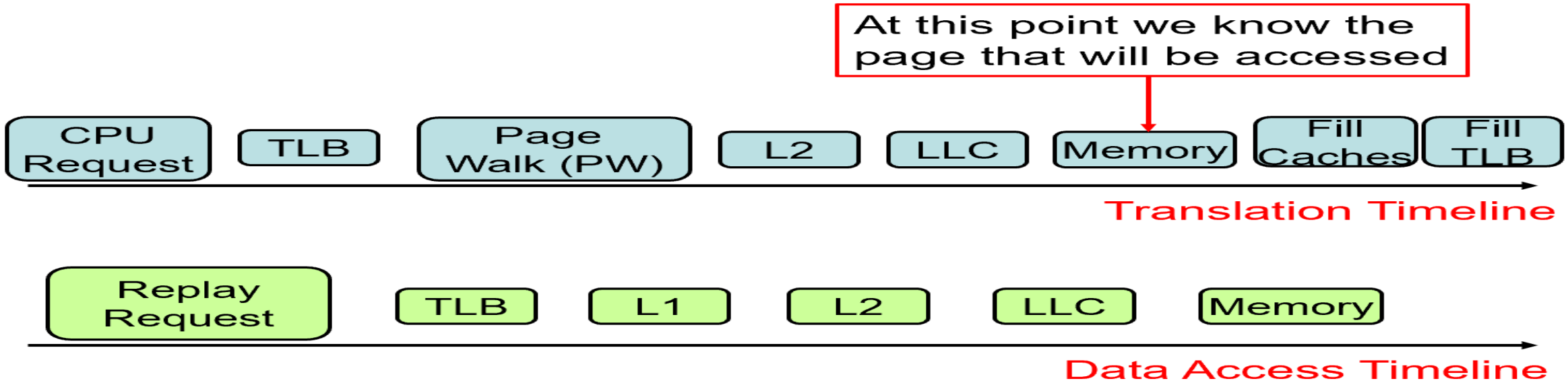
- Exclusive access to RNG cells and reserve rows that have RNG cells to minimize interference.

PageSeer: Using Page Walks to Trigger Page Swaps in Hybrid Memory Systems

- Memory capacity becoming a bottleneck with ever-evolving memory intensive applications.
- DRAM can no longer provide capacity while being power efficient.
- Non-volatile Memory (NVMs) are becoming more and more common due to high density and low power requirements.
- But they suffer for high access latency; thus hybrid memory system design is required to harness the best of both worlds.

Challenges involve:-

- Managing pages (decide which pages to swap to or from the faster DRAM based region compared to the slower NVM based one)
- Track page activity (frequently written pages, hot pages etc)
- Record the page mapping/remapping
- Swap is costly, need to predict future access patterns accurately and as early as possible
- Can take hints at translation time for memory access



- Use page-walks to trigger swaps
- Use page-correlation mechanism to prefetch page swaps
- Track hot pages
- Hybrid Memory Control (HMC) leverages information from MMU after about forth coming memory operations