From Moore's Law to Pace-Matching Data Access:

Thoughts on data-centric Computer Architectures

Xian-He Sun
Illinois Institute of Technology
sun@iit.edu

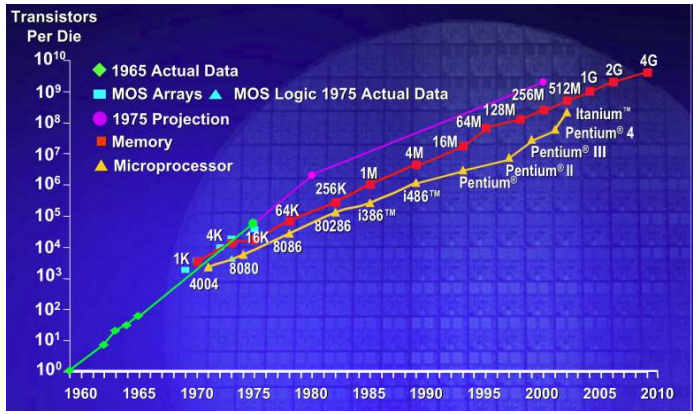
CNCC2018 Award Winning Speech, Oct. 2018

Thought

- Is **Moore's Law** (摩尔定律) ending?

Yes

- But,



Thought

- Is **Moore's Law** (摩尔定律) ending?

Yes

- But, it is not the transistors, it is **Dennard Scaling**

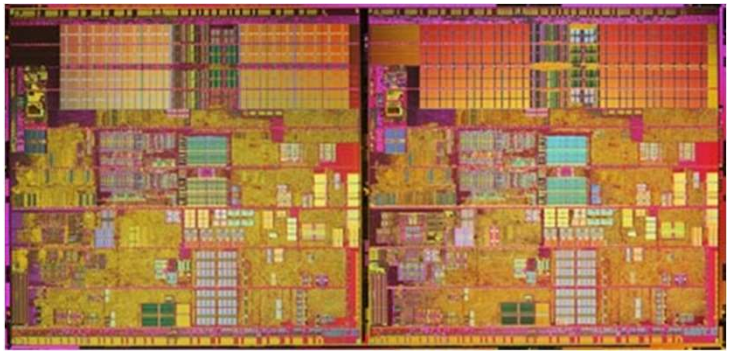
- **Dark Silicon**

Thought

- So, is power consumption (**Dennard Scaling)** the issue?

Yes

- But, but



Thought

- So, is power consumption (**Dennard Scaling)** the issue?

Yes

- But, we have the **many-core technologies**

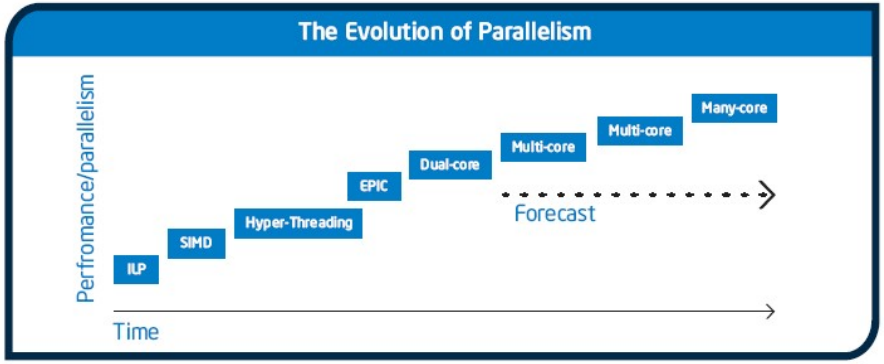
- Computing power still can increase

Thought

- Is **many-core technologies** a solution?

Yes

- But, but, but

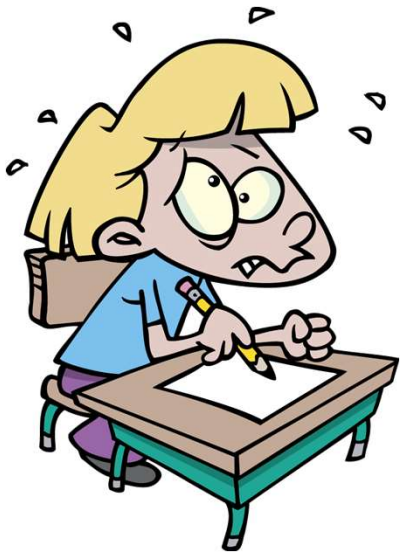


Thought

- Is **many-core technologies** a solution?

Yes

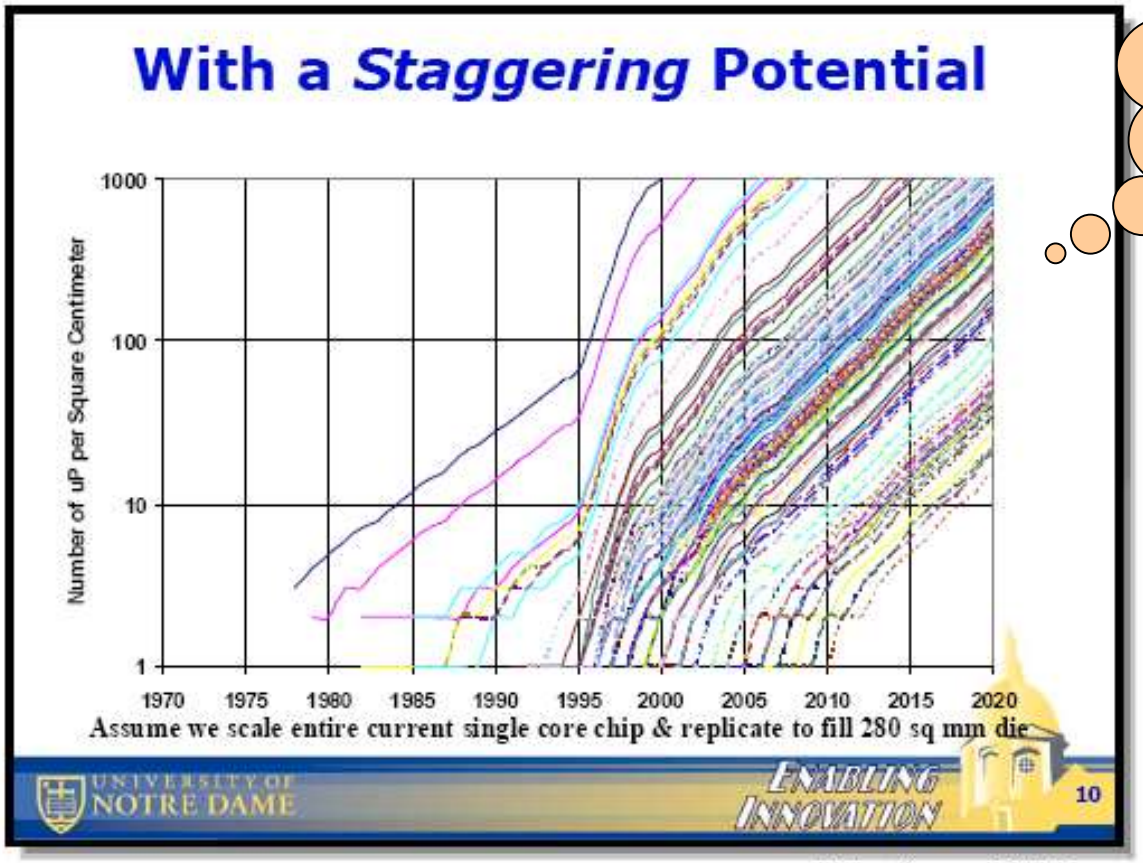
- But, it is **not scalable**



Why?

Why not Scale up the Number of Cores?

Perception/technology?



Peter Kogge, 2007

I told you
It is memory

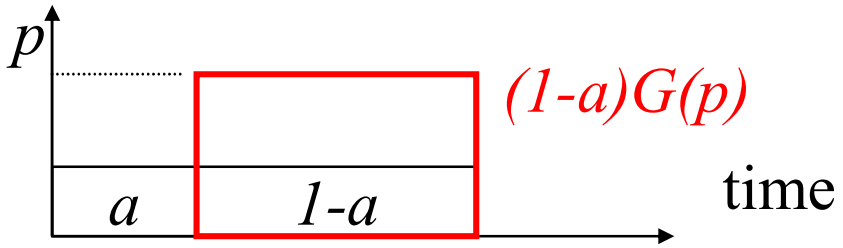
Stupid!

Sun & Ni's Law

Xian-He Sun

存储受限理论
Memory Bounded Speedup

Lionel M. Ni

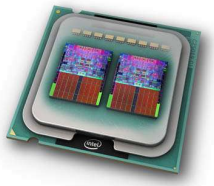


$$Speedup_{MB} = \frac{Work(p)/Time(p)}{Work(1)/Time(1)} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)G(p)/p}$$

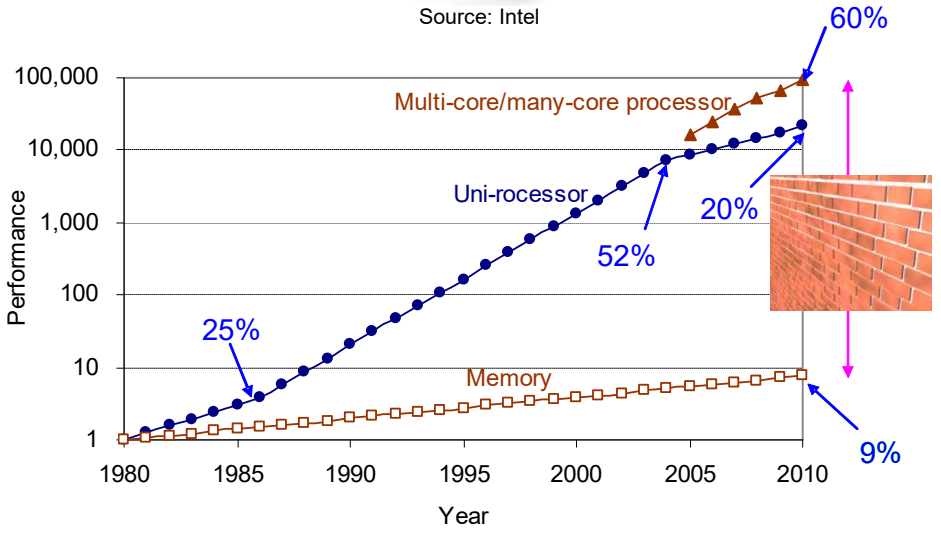
X.H. Sun, and L. Ni , *"Scalable Problems and Memory-Bounded Speedup," Journal of Parallel and Distributed Computing*, Vol. 19, pp.27-37, Sept. 1993 (SC90)

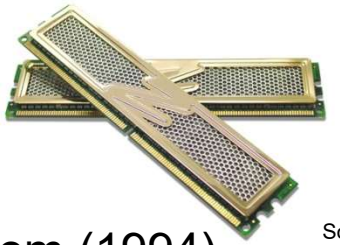
The Memory-wall Problem

- Processor performance increases rapidly
  - Uni-processor: ~52% until 2004
  - Aggregate multi-core/many-core processor performance even higher since 2004
- Memory: ~9% per year
  - Storage: ~6% per year
- Processor-memory speed gap keeps increasing



Source: Intel





Source: OCZ

Memory-bounded speedup (1990), Memory wall problem (1994)

Extension: Scalability of Manycore

- Based on Amdahl's law Multicore is not scalable

$$\frac{w_c}{perf(r)} + \frac{w_p}{perf(r)} = \frac{w_c}{perf(r)} + \frac{w_p'}{m \cdot perf(r)} \quad => \quad w_p' = m w_p$$

- Based on Gustafson and Sun-Ni's law, it scalable

$$\frac{\frac{w_c}{perf(r)} + \frac{w_p'}{m \cdot perf(r)}}{\frac{w_c}{perf(r)} + \frac{w_p}{perf(r)}} = \frac{w_c + m \cdot w_p}{w_c + w_p} = (1 - f') + m f' \qquad f' = \frac{w_p}{w_c + w_p}$$

- Based on Sun-Ni's law
  - *Multicore is scalable,* **if data access time is fixed** and does not increase with the amount of work and the number of cores
  - **Implication:** Data access is the bottleneck needs attention

X.-H. Sun and Y. Chen, "Reevaluating Amdahl's Law in the Multicore Era," Journal of Parallel and Distributed Computing, vol. 70, no. 2, pp. 183-188, Feb. 2010.

Solutions for Memory/Data Access

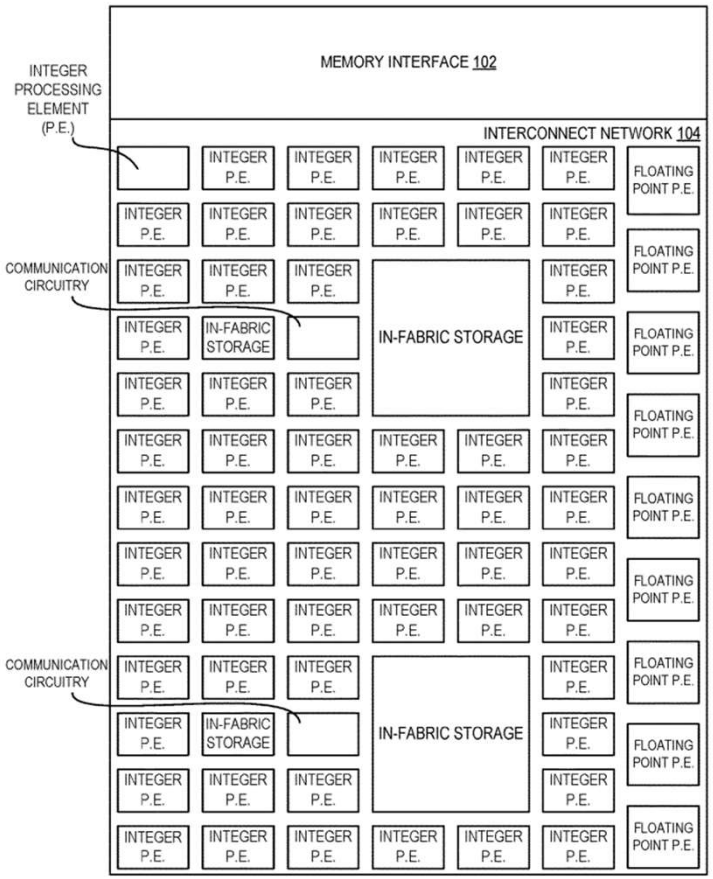
- New **Architecture** for Computing
  - GPU: data streaming
  - ASIC (Application Specific IC): not general, costly
  - Intel CSA (Configurable Spatial Accelerator): **Case study**
- New **Technology** for Memory Devices
  - 3D stacked DRAM (HBM), GDDR and multi-channel DRAM (MCDRAM), byte addressable non-volatile storage class memories (SCM) (phase-change memory (PCM), resistive RAM (ReRAM), 3D Xpoint), etc.: none can replace DRAM
- New Architecture for **Memory Systems**
  - Memory pool/memory segregation: **Case study**
  - Deep memory hierarchy: Under development
  - Elastic/Pace Matching data transfer: **Case study**

Configurable Spatial Accelerator (CSA)

1

Basic Feature

- Designed for the first US Exascale supercomputer, A21
- A grid of compute, switching, interconnect, storage elements on a die
- Dataflow Engines: *take the dataflow graph of a program, which is created by a compiler, and map that dataflow graph on an array of compute elements and interconnects*
- Region of code to be accelerated

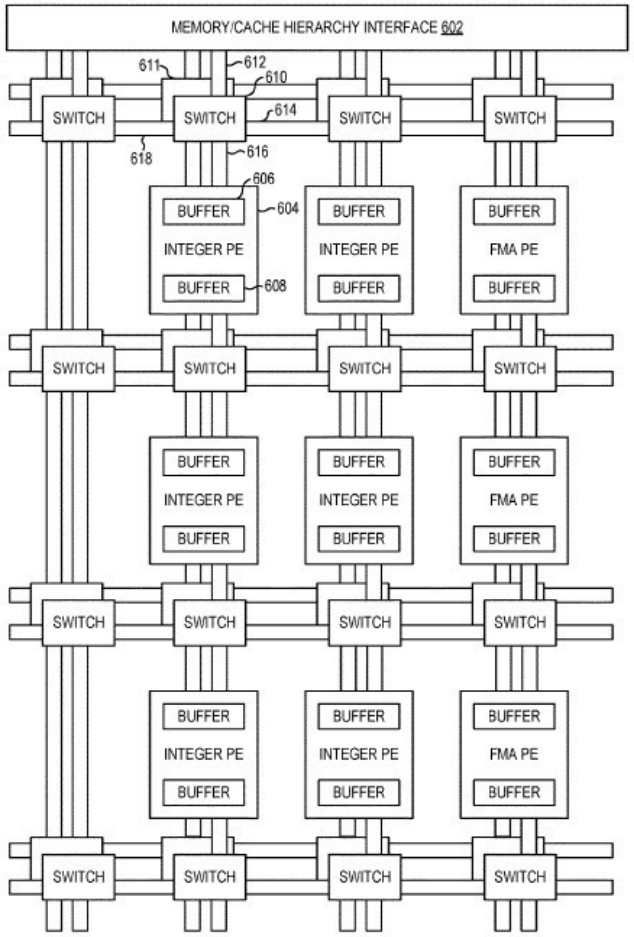


Intel CSA

Configurable Spatial Accelerator (CSA)

Basic Idea

- Only mapped elements active (dark silicon)
- Build application specific CSAs
- Grab explicitly the parallelism in the code (the dataflow graph)
- Treat memory operations as other dataflow operations
- It is a s*ystem on chip*
- as a co-processor, heterogeneous processor, or in group
- Not FPGA



Multiple Layers with Buffers

Configurable Spatial Accelerator (CSA)

3

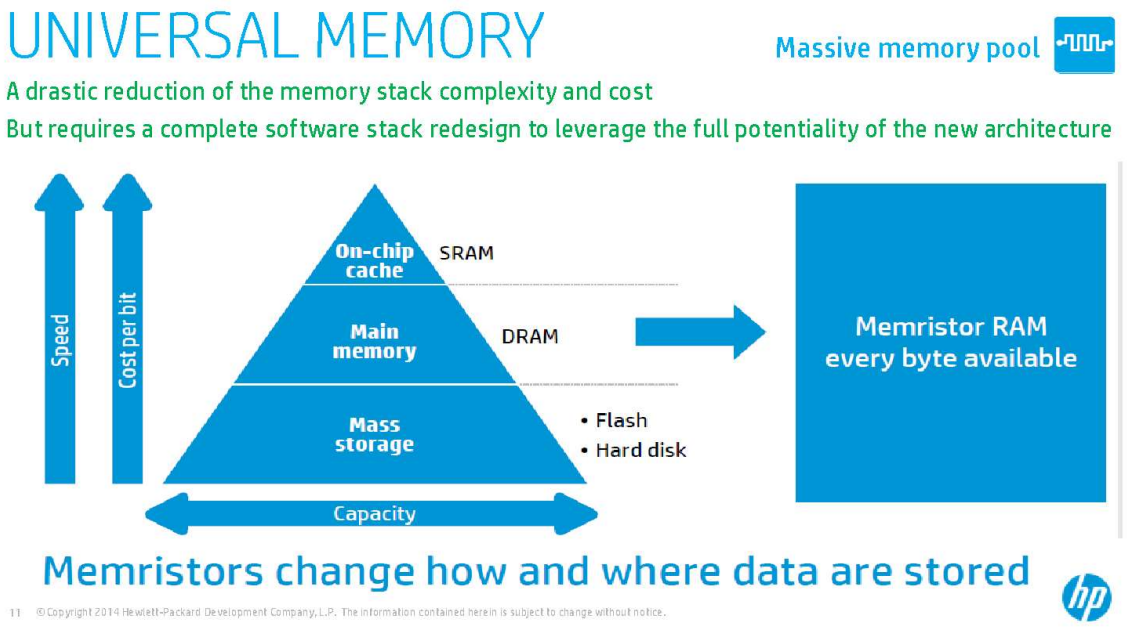
Discussion

- A dataflow machine on chip
  - But, historically dataflow concept only had very limited success
  - That is why it is called *Accelerator (work for some part of the code)*
- Build application specific CSAs
  - Must specific enough to be effective and general enough to keep cost low
- Grab explicitly the parallelism in the code for dataflow
  - Hard to achieve in general
- Accelerate what?
  - For compute-intensive application, GPU probably is a better accelerator
- So, they are (semi-)specific dataflow accelerators for some targeted non-compute-intensive application

HP: The Machine (universal memory)

Basic Feature

- Have a massive universal memory, available for every node
- Supported by photonic networks, for fast data transfer



HP: The Machine (universal memory)

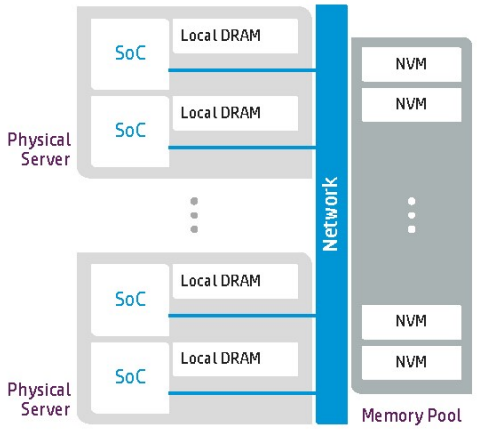
Basic Structure

- Have a massive universal memory, available for every node
- Supported by photonic networks, for fast data transfer

Basic Idea

- Fast network is available
- NVRAM are available
- With a lower software cost
- Quicker access for massive data

Essential characteristics of The Machine



Converging memory and storage

- Byte-addressable non-volatile memory (NVM) replaces hard drives and SSDs

Shared memory pool

- NVM pool is accessible by all compute resources
- Optical networking advances provide near-uniform low latency
- Local memory provides lower latency, high performance tier

Heterogeneous compute resources distributed closer to data

The Machine (universal memory)

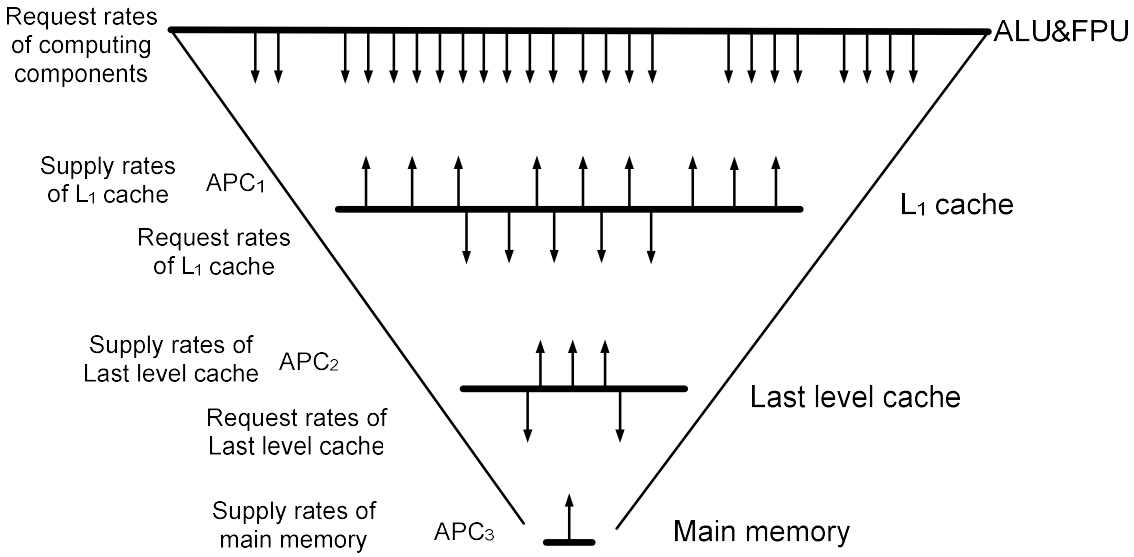
Discussion

- Each node still have its **local** memory and caches
  - Fast network does not solve the memory-wall problem
- There is a pool of global memory after local memory
  - For large data and data sharing
  - Higher bandwidth, higher latency
- Converging the global memory with storage (NVRAM)
  - Manage NVRAM as extended memory
  - Selective in accessing global memory or storage file system
- The effective of the global pool concept **needs to be verified**
- It **does not address** how to access local memory fast

Pace Matching Data Access（搏动数据获取）

Basic Feature

- Build an elastic memory hierarchy to dynamically **match** the data supply with the data request
- A methodology is developed to perform the match to remove memory-wall effect



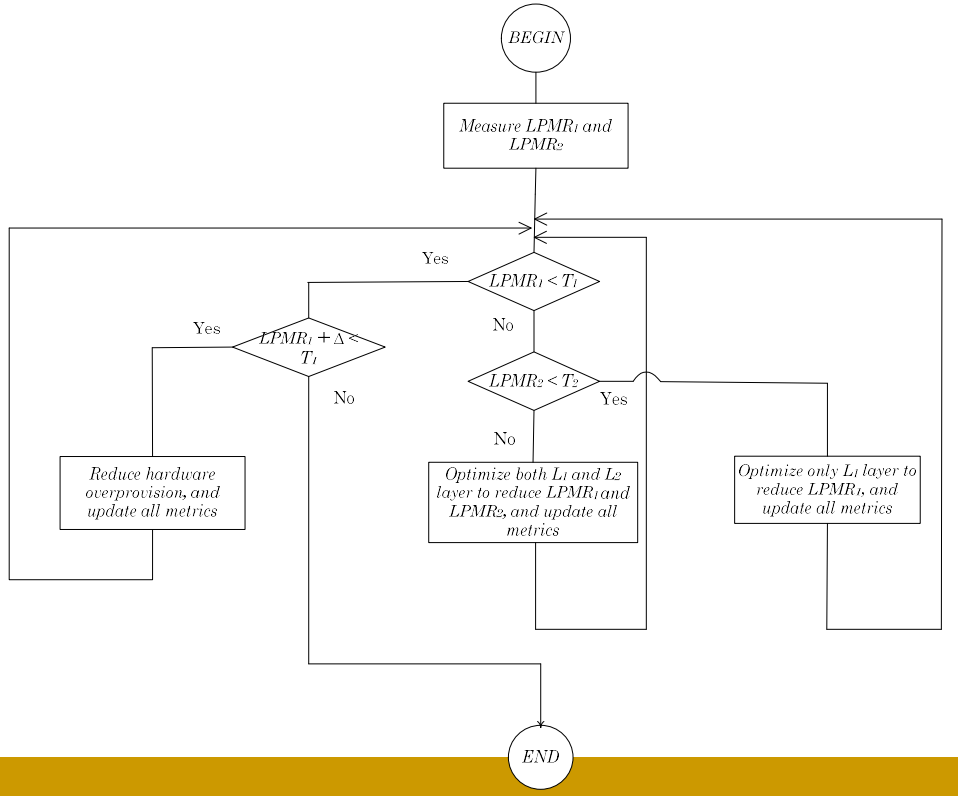
Pace Matching Data Access（搏动数据获取）

Basic Feature

- Build an elastic memory hierarchy to dynamically **match** the data supply with the data request
- A methodology is developed to perform the match to remove memory-wall effect

Basic Idea

- If a memory system matches data request, there is not data access delay
- The match can be measured and controlled dynamically



Pace Matching Data Access（搏动数据获取）

Discussion

- The C-AMAT model and LPM algorithm are developed to support Pace Matching
- Analysis and experimental results show the Pace Matching approach is correct and feasible
- Matching can be achieve through
  - Reduce request: Improve locality, **in memory computing**, etc
  - Improve data access: memory concurrency , memory technology, etc
  - Mask the difference: Overlapping computing with data access delay (pure miss), prefetch, etc.
- Like Intel CSA, HP the machine, it requests the support of
  - Hardware technology, compiler technology, application algorithm design, system scheduling

Conclusion

- Intel CSA's dataflow engine is only effective for certain code regions and needs to be specified to reduce dark silicon
- HP The Machine is designed for handling huge data, not designed for solving the memory-wall problem
- Pace Matching removes the memory-wall effect via matching data supply with data request. It is mathematically sound.

- **The combined solution**: use CSA to support Pace Matching, including using memory concurrency, memory hierarchy, and dataflow; the matching ends at the HP memory pool.
- **The combined solution**: The matching can help on designing effective CSA and managing the HP memory pool effectively

Background Literature

- Intel CSA
  https://www.nextplatform.com/2018/08/30/intels-exascale-dataflow-engine-drops-x86-and-von-neuman/amp/?__twitter_impression=true
- HP The Machine
  https://www.youtube.com/watch?v=S--Kgseuy0Q
- Pace Matching Data Transfer
  - http://www.cs.iit.edu/~scs/psfiles/hpc-china-2015-2.pdf
  - http://www.cs.iit.edu/~scs/psfiles/Sluice_CCCF.pdf
  - X.-H. Sun and Y.-H. Liu, "Utilizing Concurrency Data Access: A New Theory," in Proc. of the 29th International Workshop on Languages and Compilers for Parallel Computing (LCPC2016) (a position paper), Sept, 2016, New York, USA.
  - Yu-Hang Liu and Xian-He Sun, "LPM: Concurrency-driven Layered Performance Matching," in Proc. of the 44th International Conference on Parallel Processing (ICPP'15), Beijing, China, Sept. 2015