Supercomputing Frontiers 2017
A*STAR, Singapore

Simultac Fonton: A Fine-Grain Architecture for Extreme Performance beyond Moore's Law

Dr. Maciej Brodowicz
Research Scientist
Center for Research in Extreme Scale Technologies
School of Informatics and Computing
Indiana University

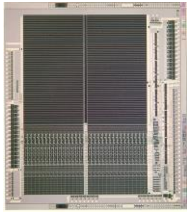
March 16, 2017

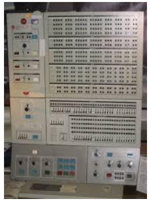
Topics

- Introduction
- Motivation
- Technical strategy
- CCA
- Simultac Fonton
- Properties
- Implementation
- Programming environment
- Conclusions

Opening Comments

- Commercial computers are predominantly von Neumann derivatives
  - MPPs, SIMD, vector, ILP, multithreading
  - Limited instruction issue
  - Optimize FPU/ALU utilization
  - Separation of processing from memory

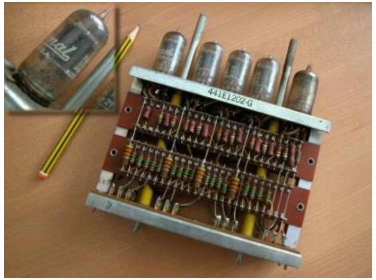




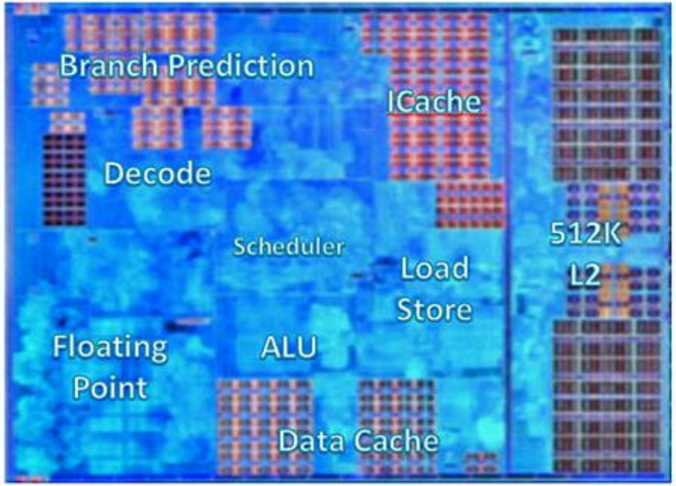
- Architectural changes
  - Epochs of computing (single issue only; pipelined; vector; SIMD array; MPPs/commodity clusters; multicore/GPU)
  - Barely respond to new technologies, opportunities, and challenges

- End of Moore’s Law
  - Approaching nano-scale
  - Power boundaries (end of Dennard scaling)
  - Where is the headroom to increase parallelism?
  - It may be necessary to go to non-von Neumann Architecture

Motivation



- Imbalance due to emphasis on ALU/FPU
- Poor die area efficiencies: caches, branch prediction, out of order and speculative execution, etc.
- Delay between when operation may be performed and actually is performed (satisfying precedent constraints)
- Von Neumann bottleneck
- Approaches demand data reuse
- Energy consumption
  - Logic
  - “Wires”: footprint, drivers, bandwidth matching
  - Communication
- Global synchronization (barriers)
- Must expose more parallelism!
- User productivity and performance portability (e.g., Titan vs Blue Gene)



Technology Demands new Response

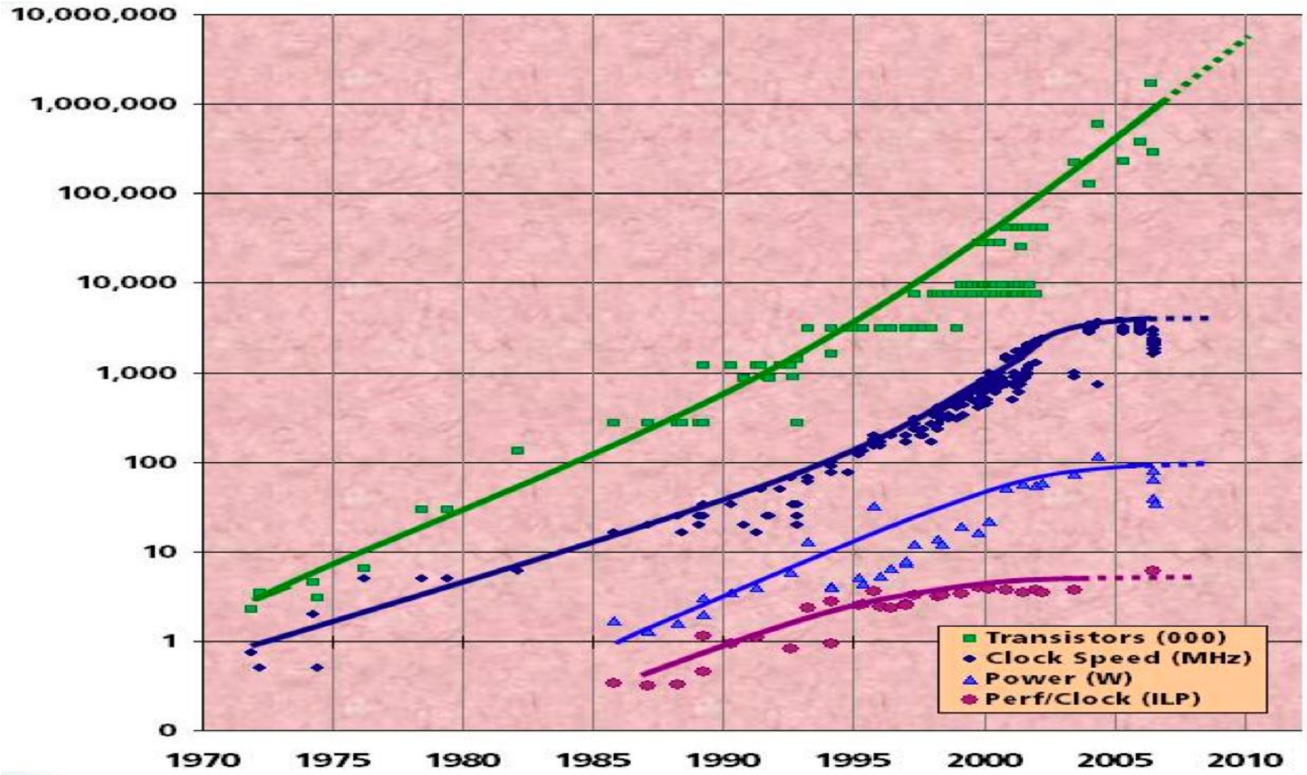


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Technical Strategy I: non-von Neumann Architecture

- Avoids “von Neumann bottleneck”
- Combines memory, execution logic, and communication in a single physical unit for
  - Reduced data access latency
  - Improved energy efficiency
  - High aggregate data bandwidth
- No explicit limitations of instruction issue

- But: some elements remain unchanged, e.g., I/O

Technical Strategy II: Cellular Structures

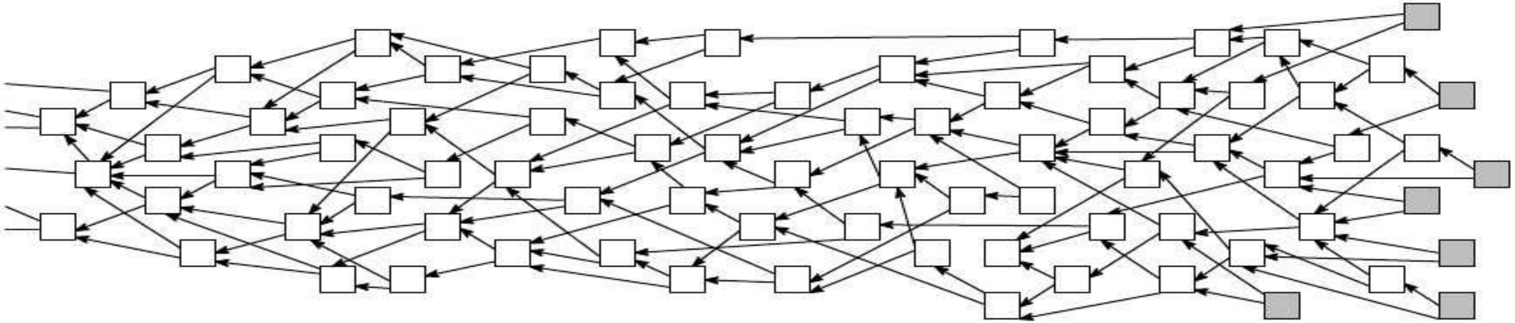
- Simplicity of design
- Lower complexity permits balancing of local operation latencies and bandwidths
- Required global characteristics met through simple aggregation
- High degree of replication affords:
  - Simplified physical placement and management of computation
  - Easier fault management due to redundancy
    - Replacement of failed units
    - Migrating the computation to “healthy” sites
  - Extreme availability
- Added bonus: no global clock

Technical Strategy III: Nearest Neighbor Access

- Minimal communication latency
- Bandwidth matched to that of local access
- Fan-in and fan-out structures possible
  - Near connection density dependent on link topology, but
  - May grow arbitrarily with radius (natural scaling)
- Facilitates:
  - Pipelining
  - Complex synchronization (e.g., compound atomic operations)
  - Complex function synthesis (tightly coupled ensembles)
  - Propagation of computation as wave-fronts (efficient for some problem classes)

Technical Strategy IV: Parallel Control Flow

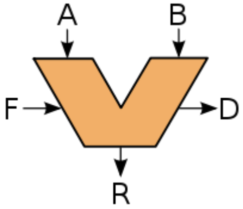
- Merging data and control parallelism
- Does not assume static dependency graph
- Enables storage of control flow information within metadata
- Encodes the parallelism discovery strategy as appropriate for specific type of data



Technical Strategy V: Objective Function Metrics

- Peak performance [ops per second]
- Total memory capacity [bytes]
- Total memory bandwidth [bytes/second]
- Communication bandwidth [bytes/second]
- Memory bandwidth to performance ratio [(bytes/s)/(ops/s)]
- Memory capacity to performance ratio [bytes/(ops/second)]
- Communication bandwidth to performance ratio [(bytes/s)/(ops/s)]
  - Internal
  - External
  - I/O

Technical Strategy VI: Not Maximizing FPU Utilization



- ALUs and FPUs no longer a precious resource
  - Early generations required maximum utilization of logic
  - VLSI dramatically shifted balance
  - But, even current multi-core architectures designed around FPU
- Emphasizes availability rather than utilization
  - FPUs are cheap
  - Latency and bandwidth are expensive
  - Permeate computing vehicle design with ALUs to minimize von Neumann bottleneck

Technical Strategy VII: Asynchrony Management and Message-Driven

- Eliminate global barriers
- Active message based
  - Remote actions with arguments
  - Action code embedded in messages or stored at remote site
  - May be fine-grain when needed
  - Pure data transport a special case
- Actions carried out in global namespace
- Supports realization of complex distributed control schemes

Technical Strategy VIII: Practicality

- Design:
  - Possible to build it using available technologies
  - Resource, cost, and energy effective
- Integration
  - Fits the existing infrastructure
  - Scales to higher level of performance
- Use:
  - Support for common programming languages and practices
  - Must not require the users to know every intricate detail of the architecture
    - Hardware support for important features
    - Support of practical software APIs
  - **Reduces the total time to solution**

Performance Factors - SLOWER

$$P = e(L,O,W) * S(s) * a(r) * U(E)$$

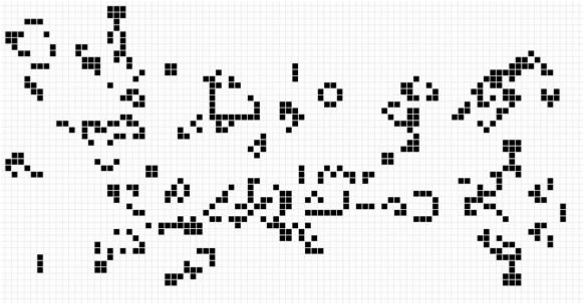
P – performance (ops)
e – efficiency (0 < e < 1)
s – application’s average parallelism,
a – availability (0 < a < 1)
U – normalization factor/compute unit
E – watts per average compute unit
r – reliability (0 < r < 1)

- Starvation
  - Insufficiency of concurrency of work
  - Impacts scalability and latency hiding
  - Effects programmability
- Latency
  - Time measured distance for remote access and services
  - Impacts efficiency
- Overhead
  - Critical time additional work to manage tasks & resources
  - Impacts efficiency and granularity for scalability
- Waiting for contention resolution
  - Delays due to simultaneous access requests to shared physical or logical resources

Technical Strategy IX: Satisfying SLOWER Model

- Starvation: hierarchies of parallelism
- Latency: new structures to reduce effects
- Overhead: architecture hardware mechanisms
- Waiting: functional availability
- Energy efficiency
- Resiliency: component redundancy and availability

Continuum Computer Architecture



- Discretization of continuous medium
- Genus of architecture
  - There are other solutions
- Ulam and Von Neumann invented the first CCA
  - Cellular automata
  - Some instances proven to be Turing-complete
  - Not practical (not general purpose, hard to program)
- Emergent behavior as a result of multitude of concurrent actions
- “No neuron knows you are playing a game of chess”

Simultac Fonton

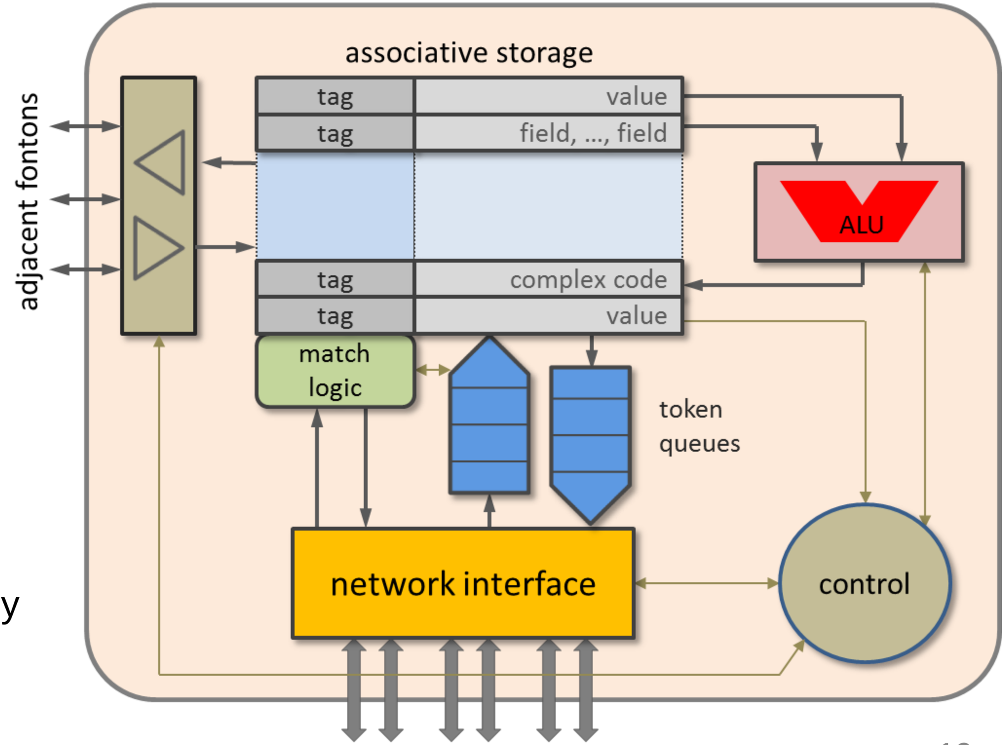
- Exploratory
  - No claim of guaranteed success
  - Big claim of interesting and potentially important possibilities
- Species of Continuum Computer Architecture
  - Not the only way to realize deep concepts associated with CCA
  - Exciting approach do to realizability with existing and near term technologies
- Stresses practical issues of design and implementation
  - Very simple design; less than a RISC core
  - Can be developed today with existing methodologies
  - Prototyping with current simulators and FPGA technologies
- Replicated and connected primitive elements

Fonton Attributes

- Fonton is a cell…
- And NOT a core
  - Much smaller
  - Limited functionality
  - Cannot execute program alone
- Incorporates basic primitive properties in a single unit
  - Unified
  - Logic + state
  - Adjacency interfaces
- High density as design goal
  - Over half of fonton area is state
  - Dramatic increase in system ALU count
  - Wide ALU
  - One cycle operation
  - Maximize fonton count on a die

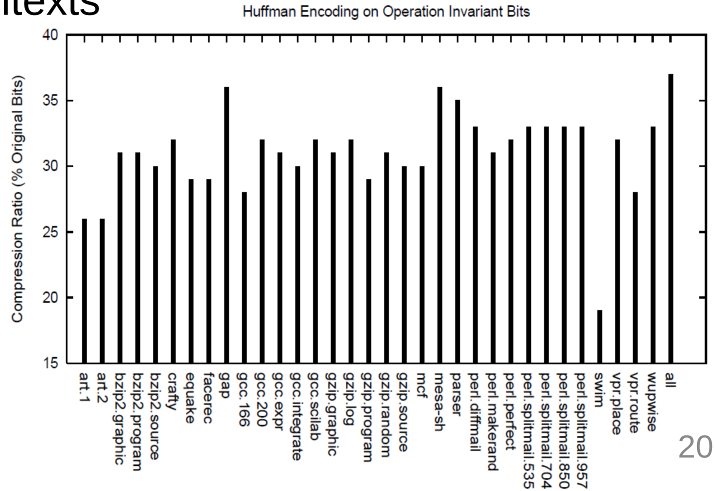
Fonton

- Storage space is not memory, registers or scratchpad
  - Registers and cache are not required
  - No register spilling
  - No replicated state
- Tagged architecture
- Typed data and operands
- Context tag
- PRECISE
- Not limited to data reuse
- Logic provides highest possible bandwidth (similar to PIM)
- Maximizes bandwidth and minimizes latency
- Avoids internal pipelining to reduce area
- Multiple ports to the external system



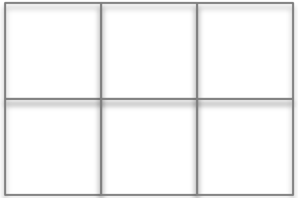
PRECISE

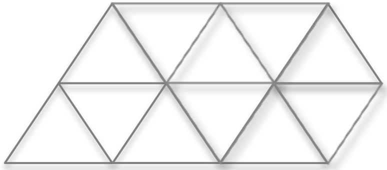
- *Processor Register Extensions for Collapsed Instruction Set Encoding*
- Compression of instruction stream
  - E.g., Huffman encoding
  - Different encodings for different execution contexts
  - About 5 bits per instruction usually suffice
- Reintroduce accumulator
  - Eliminates code of one operand
- Every operand is typed
  - No need for different instruction classes

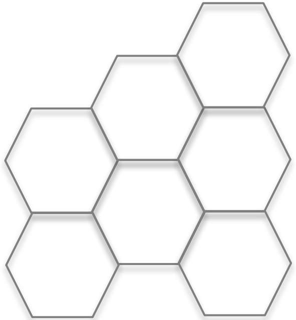


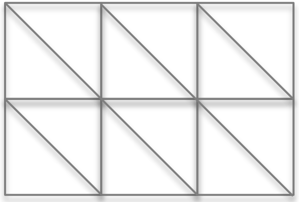
Simultac: Tessellation

- Uniform tessellation without gaps achieved with
  - Isosceles triangles
  - Equilateral triangles
  - Quadrilaterals
  - Hexagons
  - Other
- Compound atomic operations may be performed within local neighborhood
  - Small latencies
- May be extended to 3D
  - Beyond current scope
  - But 3D die stacks and NVRAM dies considered
- Hierarchy
  - Inter-fonton network
  - Provides added dimension

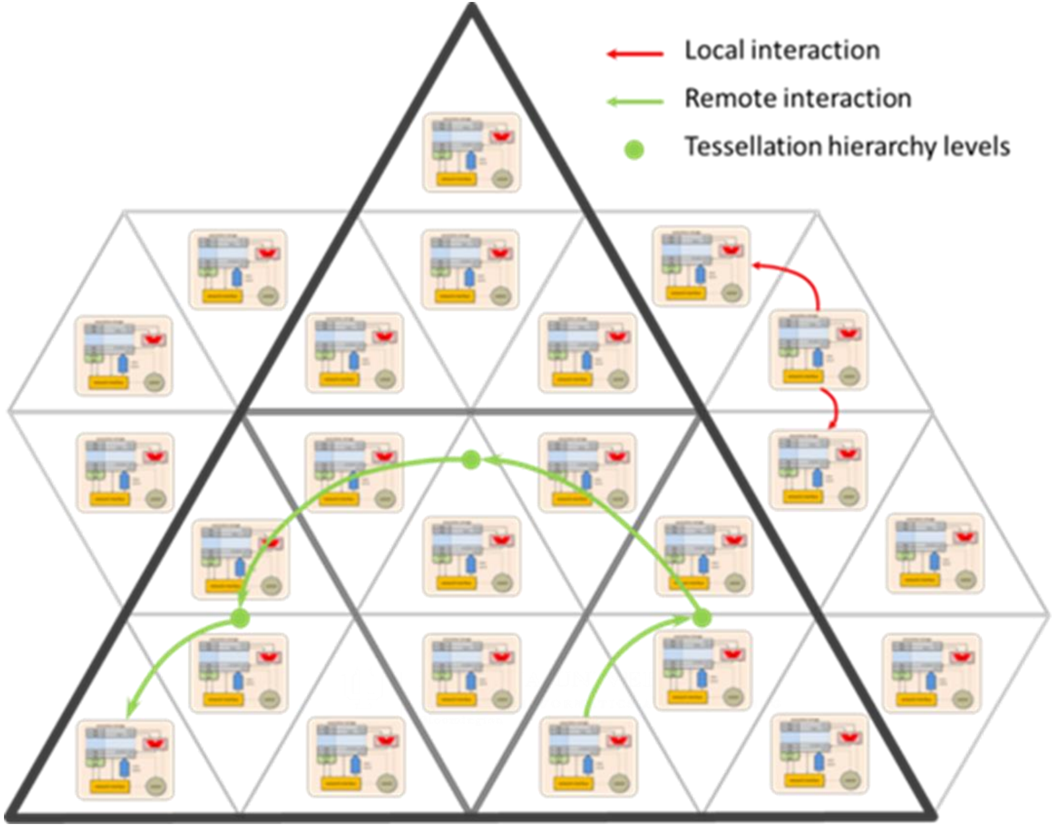








Example Triangular Tesselation

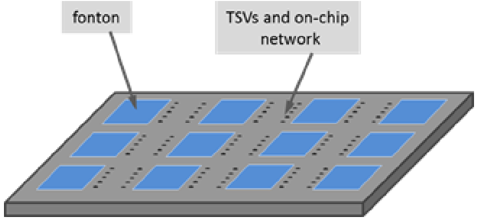


- Neighborhood routing
  - Wormhole routing
  - Dynamically modifiable on faults
  - About 1bit/hop
- Intra-die network
  - Hierarchy determined by tessellation boundaries
  - Fat tree – like
  - May be oversubscribed due to high-bandwidth adjacency traffic

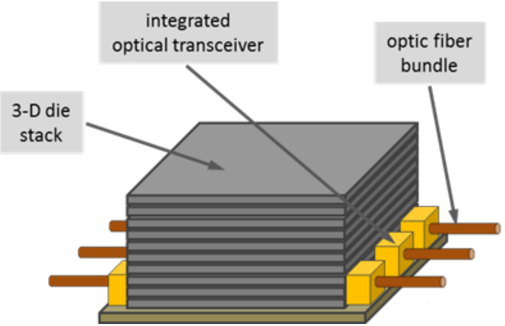
Trade-offs

- Word size
- Tagging and context overhead
  - Lookup tables may be shared by fonton groups (for PRECISE)
- Selection of functional units and operand sizes
  - Adder
  - Multiplier
  - Division support
  - Permutation network
- Wide vs. scalar ALU
- Dedicated code store (reduces #memory ports)
- Fraction of memory vs. logic
- Intra-chip interconnect
  - Type
  - Bisection bandwidth at every hierarchy level
  - Token buffer capacity

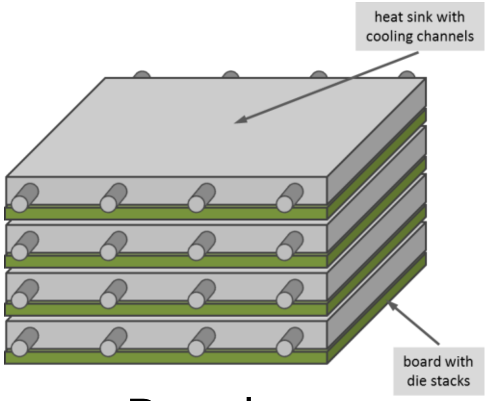
Simultac: Scaling in CMOS



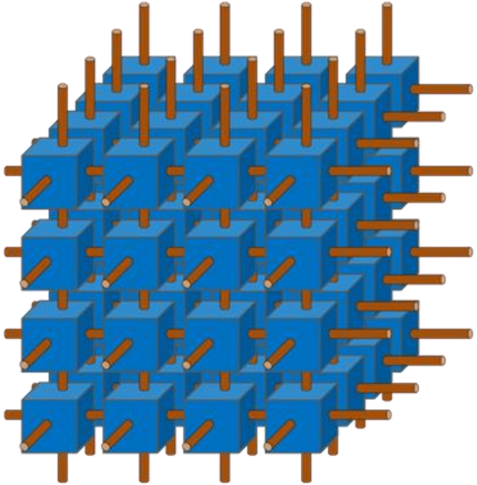
Single die



3D die stack connected
by through-silicon vias



Board
stack



System

Simultac vs. Prevalent Architectures

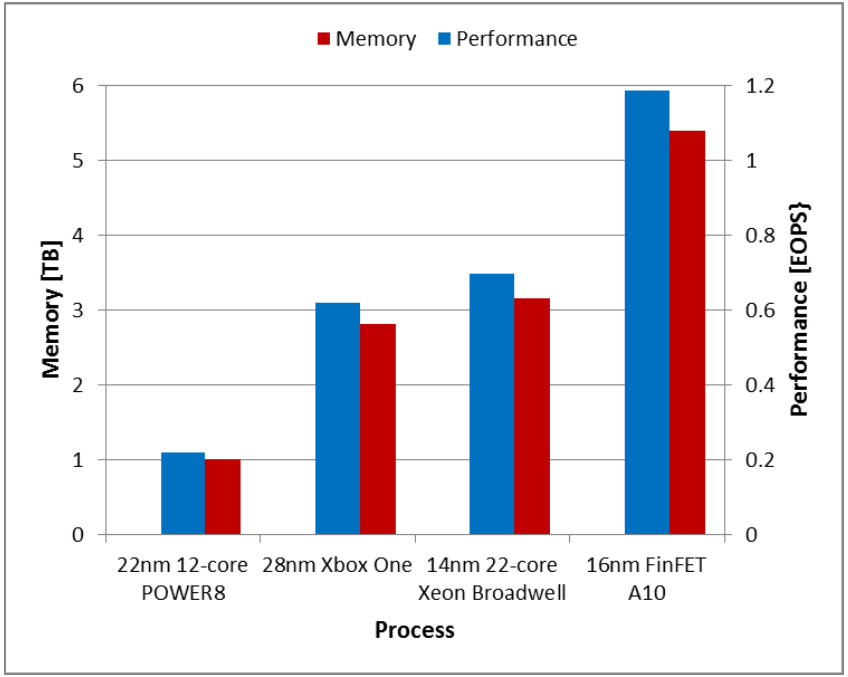
Property	CCA	Multi-core	GPU
Granularity of execution	Fine	Medium (disjoint cores)	Fine to medium (determined by SIMD)
On-die local store bandwidth	Very high (aggregate intra-fonton)	Medium (registers, caches)	High (registers, caches)
Attached memory bandwidth	Medium to high (optional external DRAM ensemble)	Low (shared DRAM banks)	Medium to high (GDDR, HBM); low to host DRAM
Memory addressing	Global, associative	Node local	Local only; node local with HSA
Memory coherence	Software controlled with hardware support	Node scope (NUMA)	GPU scope
On-die communication bandwidth	Very high	Medium (HT, QPI)	Medium-high (shared storage)
I/O latency	Low to medium, variable	Medium (PCIe)	Medium (PCIe)
I/O bandwidth	Medium to high	Medium (PCIe)	PCIe and host dependent
Fault recovery	Likely (replication, dense routing)	Highly limited	Limited

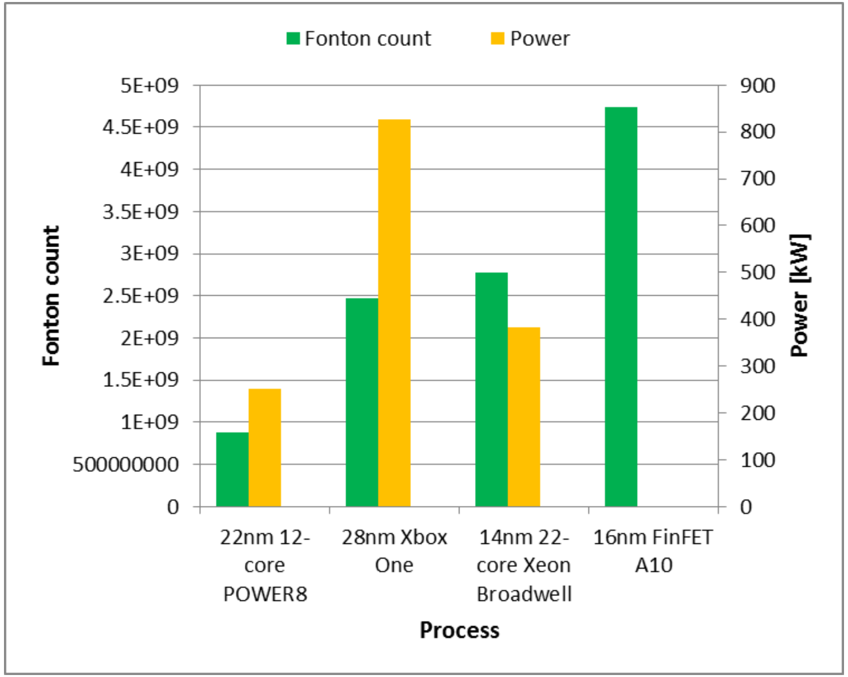
CMOS Implementation Assumptions

- Logic
  - 12 transistors per storage bit
  - 250 MHz clock
- Fonton
  - 16,000 6-transistor gates
  - 100mm^2 die
  - 40% logic, 50% memory, 10% interconnect
  - About 1KB effective storage
- Die stack
  - 4 dies

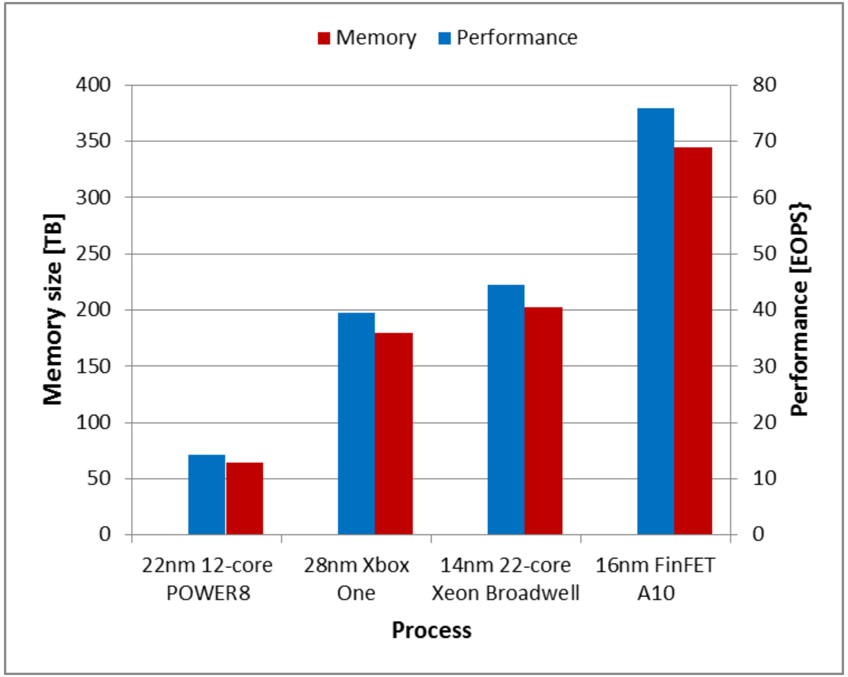
- Board
  - 1m x 1m
  - 10mm effective stack separation
  - 90% for stacks & board-level interconnect
- “Cube”
  - 20mm board spacing
  - 1m high
- System
  - 4 x 4 x 4 cubes

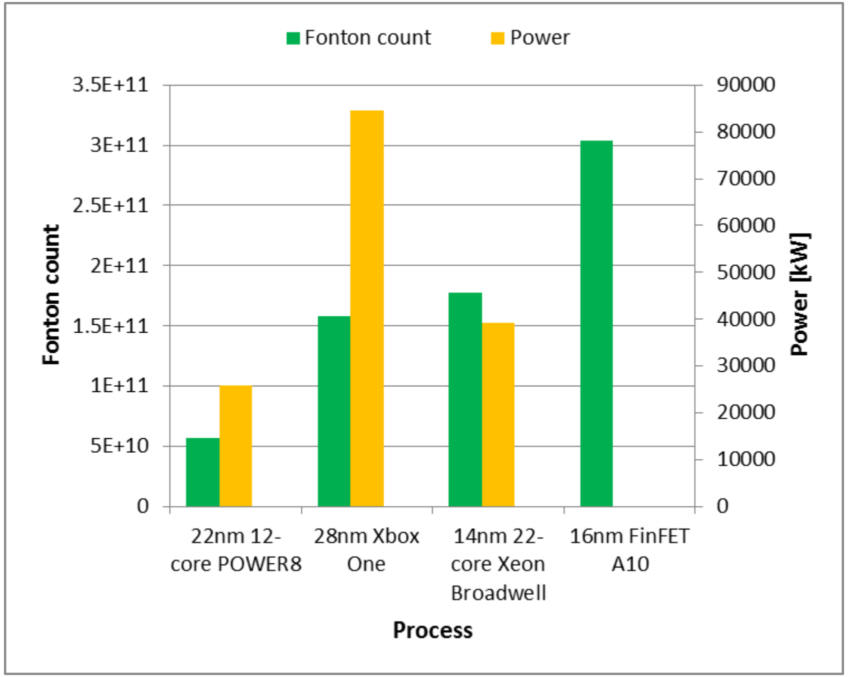
CMOS Implementation: Cube





CMOS Implementation: System



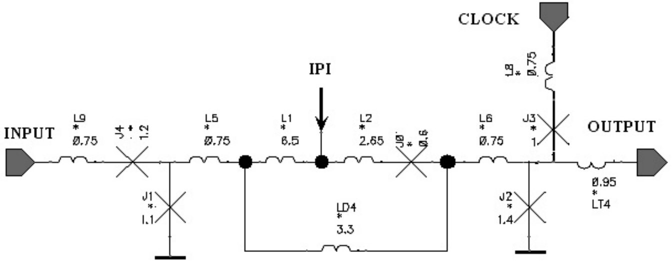


Reference Implementation Properties

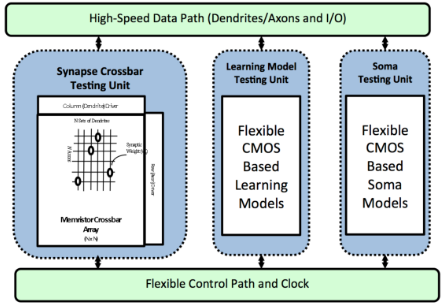
Parameter	Simultac	TaihuLight
Clock speed	250 MHz	1.45 GHz
Processing units	303.6 billion fontons	83.9 million FPUs
Peak performance	76 ExaOPS	125 PetaFLOPS
Total memory	345 TB	1.28 PB
Memory bandwidth	1821 EB/s	5.46 EB/s
Memory size to performance	0.0000044 bytes/OPS	0.01 bytes/FLOPS
Memory BW to performance	24 bytes/OP	0.044 bytes/FLOP
Footprint	25 m^2	605 m^2

Alternative Technologies

- Single Flux Quantum devices
  - Josephson junction based (superconducting)
  - Cryogenically cooled
  - Very low power demand for logic (~0.001% of CMOS)
  - Clock frequency demonstrated at few hundred GHz
  - Fastest-clocked logic at the SC'97



- Neuromorphic computing
  - Inspired by bio-neural processes and networks
  - Emulate select functions of nervous system
  - VLSI circuits
  - Analog, digital, and hybrid implementations
  - Utilize negative differential resistance or capacitance, and threshold switching circuits


High-Speed Data Path (Dendrites/Axons and I/O)
Synapse Crossbar Testing Unit
Column (Dendrite) Driver
N Sets of Dendrites
N Axons
Synaptic Weight (W)
Row (Axon) Driver
Memristor Crossbar Array (N x N)
Learning Model Testing Unit
Flexible CMOS Based Learning Models
Soma Testing Unit
Flexible CMOS Based Soma Models
Flexible Control Path and Clock

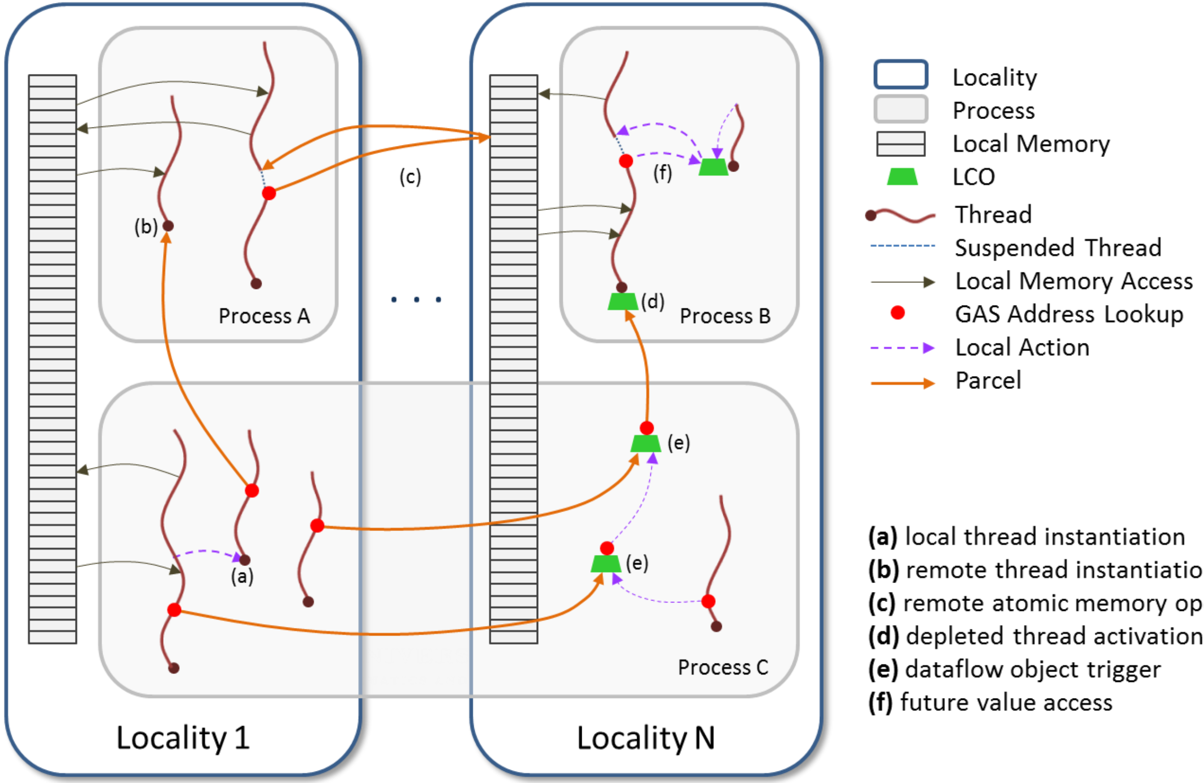
What about Programming?

- Dynamic adaptive model of computation
- Asynchrony management
- Fine-grain capable
- Able to support massive amounts of parallelism
- Lightweight synchronization
- Naturally message-driven
- First class objects reachable in global namespace
- Supports migration to other physical locales
- Manages hierarchy of computational states
- Scalable
- Spatially aware

ParalleX Model of Execution

- Model for guided computation, not “ballistic”
- Processes
  - Management of parallel computation hierarchy
- Message-driven computation mediated by parcels
  - Active messages with continuations
  - Carry actions, arguments, and data to destination objects described by global addresses
- Active Global Address Space
  - Permit access and migration of first class objects in physical space
- Compute complexes
  - Control flow constrained by data dependencies
  - Superset of conventional threads
- Local control objects (LCOs)
  - Rely on atomic updates of local state
  - Futures, dataflow, …

Interaction of ParalleX Elements



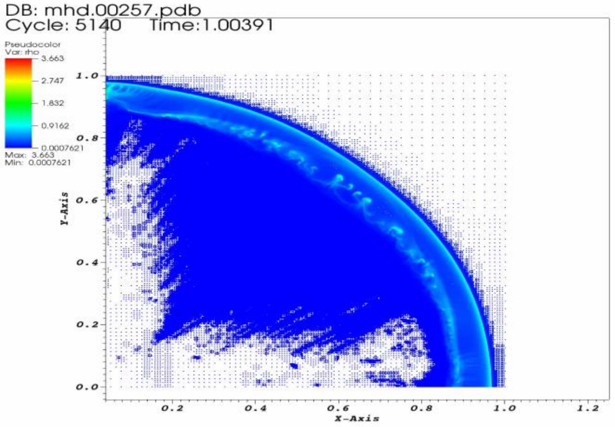
(a) local thread instantiation
(b) remote thread instantiation
(c) remote atomic memory operation
(d) depleted thread activation
(e) dataflow object trigger
(f) future value access

ParalleX Mapping to Simultac

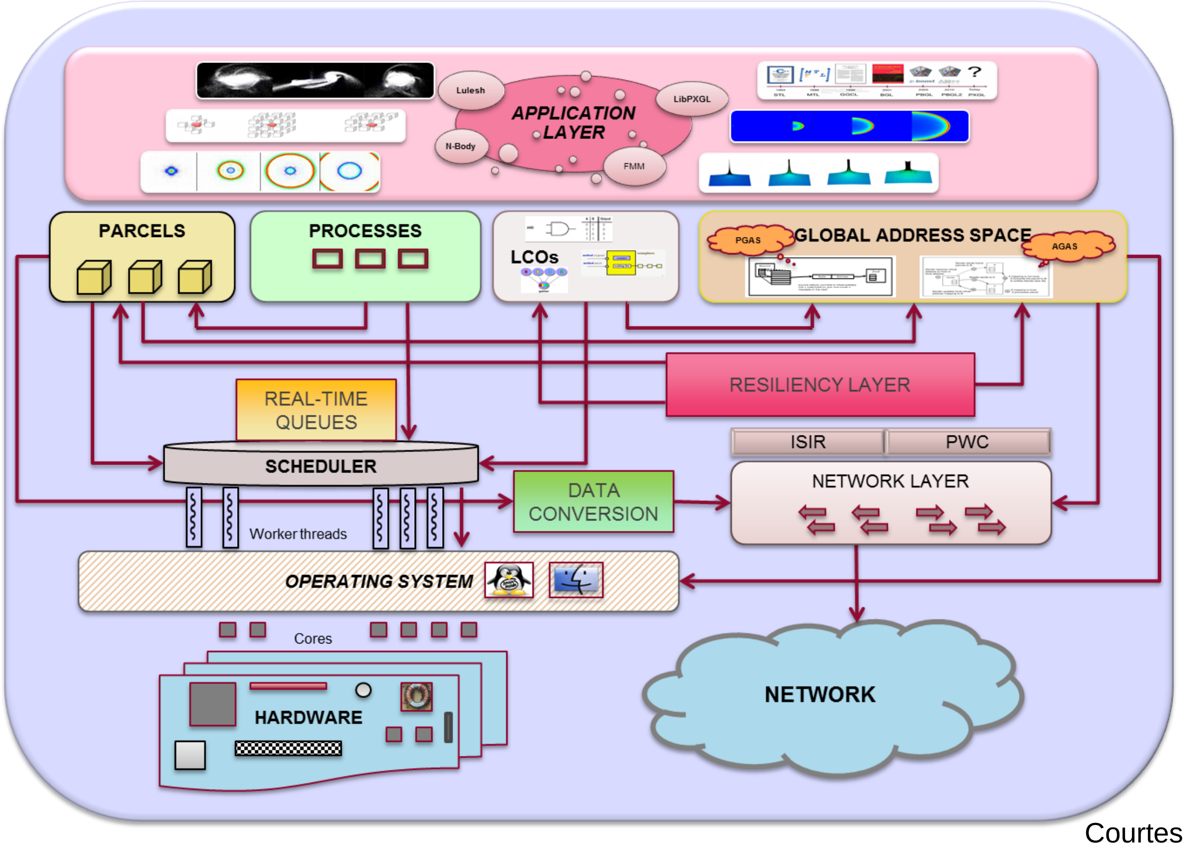
- Micro-complexes to perform fine-grain static dataflow computations
  - Play the role of threads on conventional processor cores
  - Stored in fontons
  - Carried in tokens
- Most LCO types directly supported by fonton logic
  - Guarantee of local atomicity of operation
  - Fonton aggregations deliver distributed control of arbitrary shape and scope
- Parcels represented by single tokens and token groups
- Union of memory tags comprises GAS
- Processes used for
  - Establishing of spatial and functional hierarchy
  - Allocation of medium for macro-computations
  - Relocation of computation as in whole units (optimization and fault tolerance)

HPX+: Runtime Software System Development

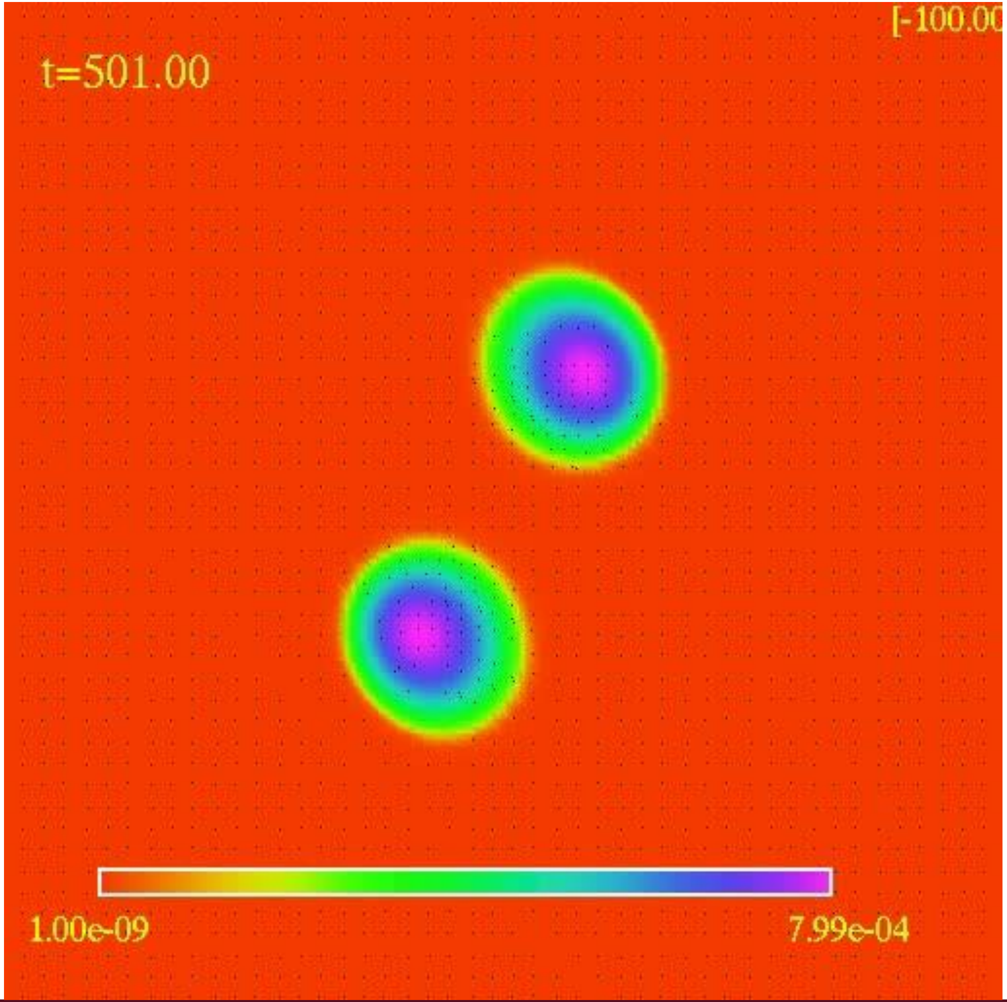
- Reduction to practice of ParalleX execution model
- Thread scheduler
- Global address system (AGAS)
- Message-driven computation
- Multi-locality dynamic processes
- Futures/dataflow synchronization and continuation
- Percolation for heterogeneous computation
- Introspection data acquisition and policy-based control
- Load balancing hooks/stubs
- Low level intermediate representation for source to source compilation and heroic users/experimenters
- Driver for architecture investigations



HPX+ Runtime Software Architecture

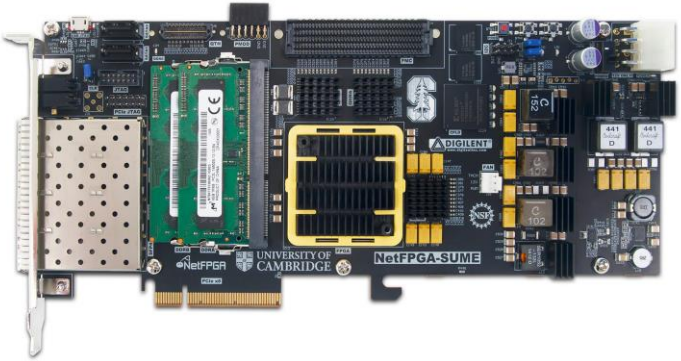


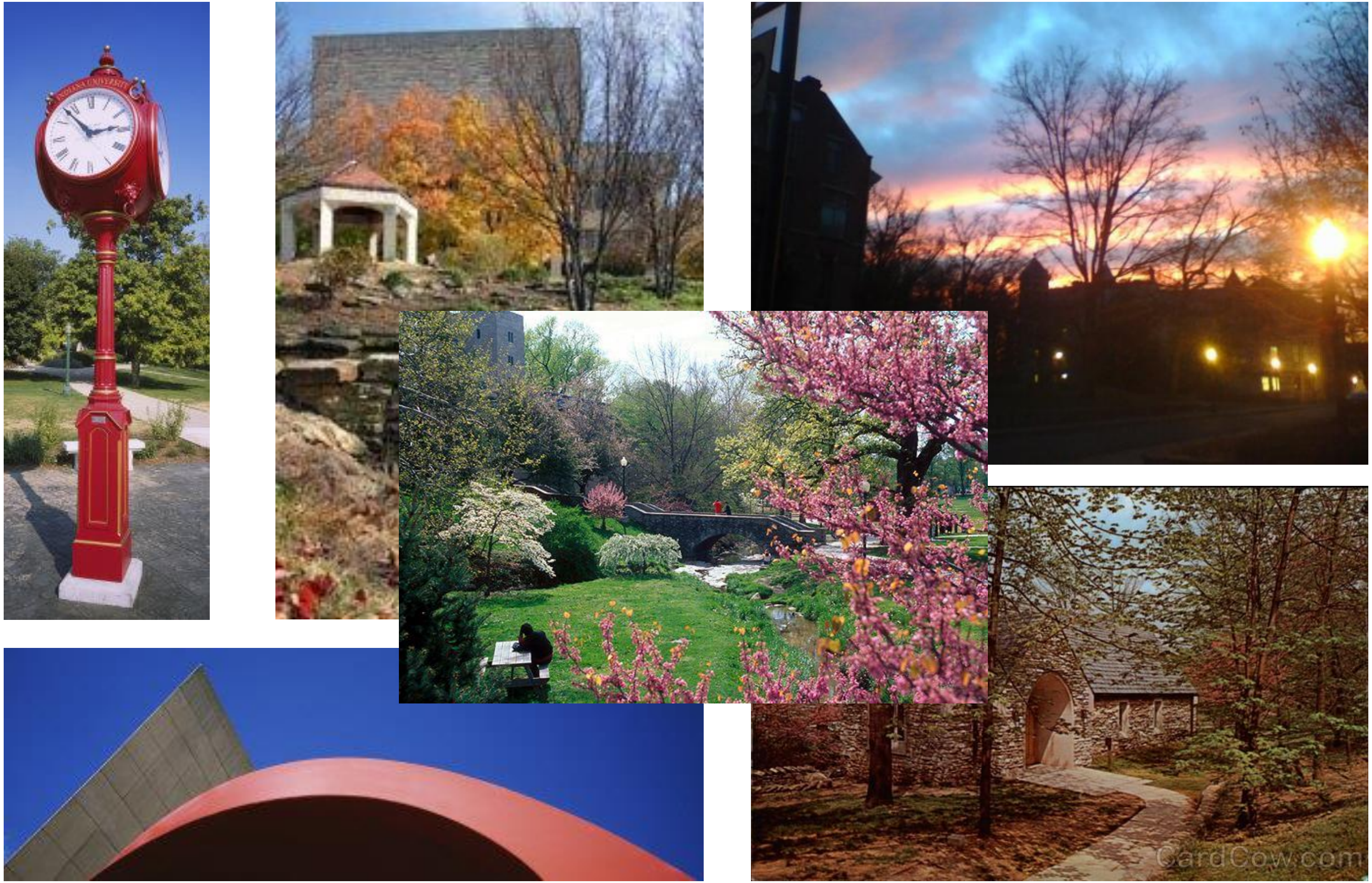
Courtesy of Jayashree Candadai, IU


t=501.00
[-100.00
1.00e-09
7.99e-04

Closing Remarks

- Von Neumann architectures no longer satisfy the constraints of CMOS
- CCA demonstrates feasibility of non von Neumann architectures for extreme scale computing
- Property of emergent behavior from replicated function units powerful construct
- CMOS implementation of Simultac plausible using current technology
- Memory starved
  - NVRAM on die stack for permanent storage, fault tolerance, increased capacity
  - May use attached memory dies as a part of stack
- Future work
  - Fonton ISA
  - Selection of optimal tessellation
  - Interconnect parameters
  - FPGA proof-of-concept fonton




INDIANA UNIVERSITY
CardCow.com