

OPEN

Compute Summit

March 10–11, 2015

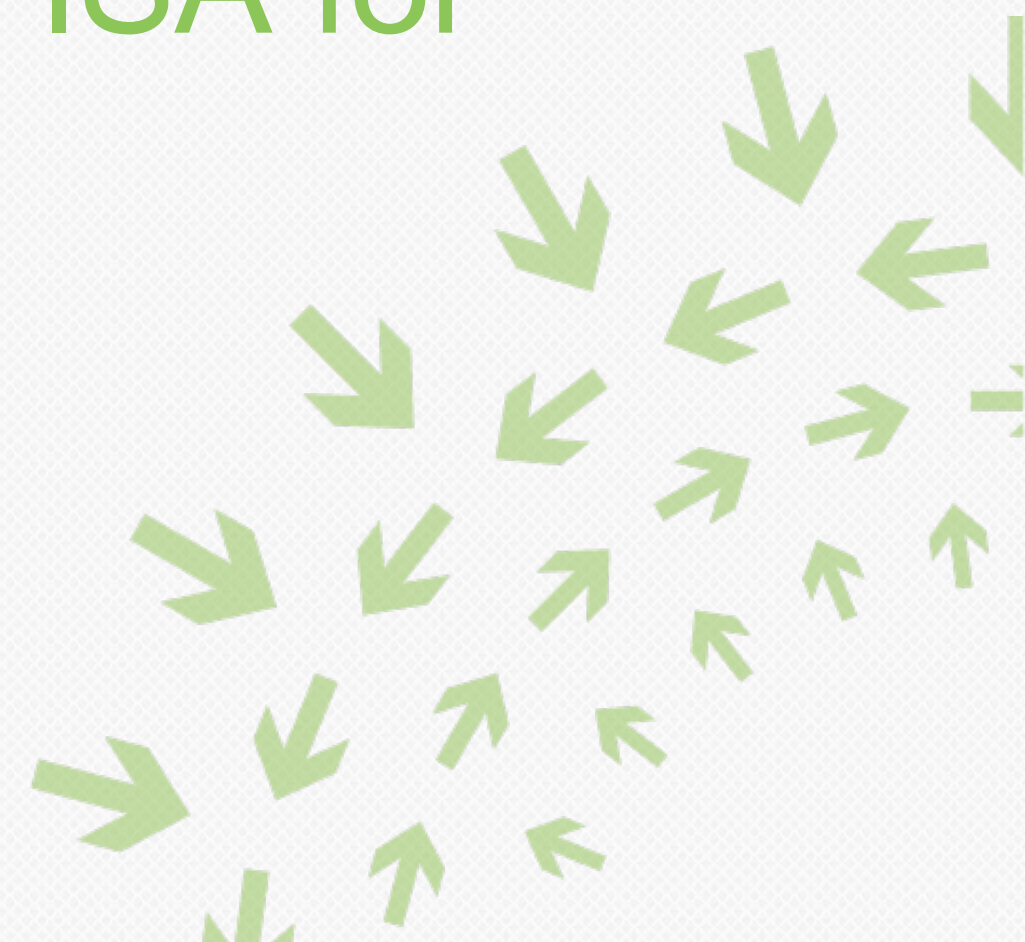
San Jose



Neo and Neo64

A new processor and open source ISA for
HPC

Thomas Sohmers
REX Computing
CEO



Current processors are built for an old paradigm

- 30 years ago, moving data was cheap (energy wise) and computation was expensive. This has now flipped, with moving data being over 40x more expensive.

| <i>Operation</i> | <i>Energy consumed</i> |
|-----------------------------------|------------------------|
| 64-bit fused multiply-add (DP FP) | 100 pJ |
| Read/store 64 bits register data | 640 pJ |
| Read 64 bits from DRAM | 4200 pJ |

- The #1 inefficiency in processors today is the standard hardware managed cache hierarchy, which can be removed and save over 50% of die area.

Existing architectures will not be able to get to Exascale and beyond without Moore's law or without major architectural changes.

| | | | |
|---------------------------|----------------------|------------------------------|-------------------|
| | Intel Xeon E5 | Intel Xeon Phi (2016) | NVIDIA K80 |
| Double-precision GFLOPS | 506 | ~3000 | 2,910 |
| Double-precision GFLOPS/W | 3.4 | ~12-15 | 9.7 |



What's so big about Exascale?

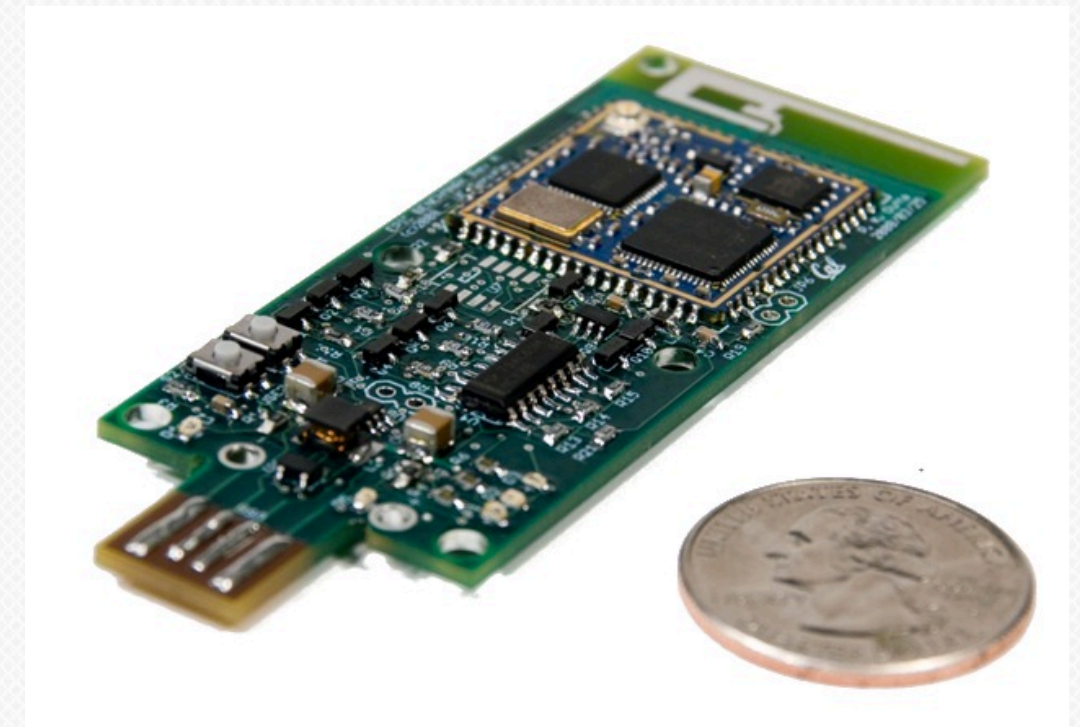
10^{18} FLOPs

How have we always picked HPC systems?

- Applications are picked first, and hardware is designed to fit those.
- Most cases you have a hard set power/area/cost budget, but will try to get the best bang for your buck.
- For the majority of large problems, you are using the entire system.



COURTESY: PROF. JACK DONGARRA



A short history of caches

“You can’t fake memory bandwidth that isn’t there” –Seymour Cray

- Caches are great...
 - Hide latency due to slow main memory
 - CDC 6600 (1964) with instruction stack through the initial Motorola 68k (1982) instruction caches. It only started to get bad when chip designers had a bunch of “free” transistors.
- Except kills power efficiency when implementing virtual memory
 - IBM System 360 Model 67 (1967) was the first to implement Virtual Memory with what we would now call a MMU (at that point a “Dynamic Translation Box”)
 - Additional logic for TLBs and other logic on a modern chip uses 30%-50% of the die area.
 - **Virtual Memory translation and paging are two of the worst decisions in computing history**
 - Adds latency and power usage for making things a bit easier for a programmer... It may be “Better” programming, but it is not “Faster” or “Cheaper”.



A short history of caches cont.

- Chip designers got lazy with VLSI
 - Intel i286 (1982) was one of the first commercial chips to implement memory protection, and expanded upon the memory segmentation of the 8086.
 - Intel i386 (1986) was the first to implement an external cache (16 to 64K), but it was not replicating anything higher up in the hierarchy.
 - The 486 (1989) the first chip to implement an 8K on die cache.
 - The Pentium and Pentium Pro continued this trend, implementing a managed cache hierarchy followed by a second level of cache.
 - Introduction of hardware managed caching is what I consider “The beginning of the end”, in which additional hardware complexity was acceptable due to not caring how many transistors were used, as they were get faster, lower power, and cheaper every year.
 - This does not work well once the power wall is reached, and completely breaks down with the end of Moore’s law.

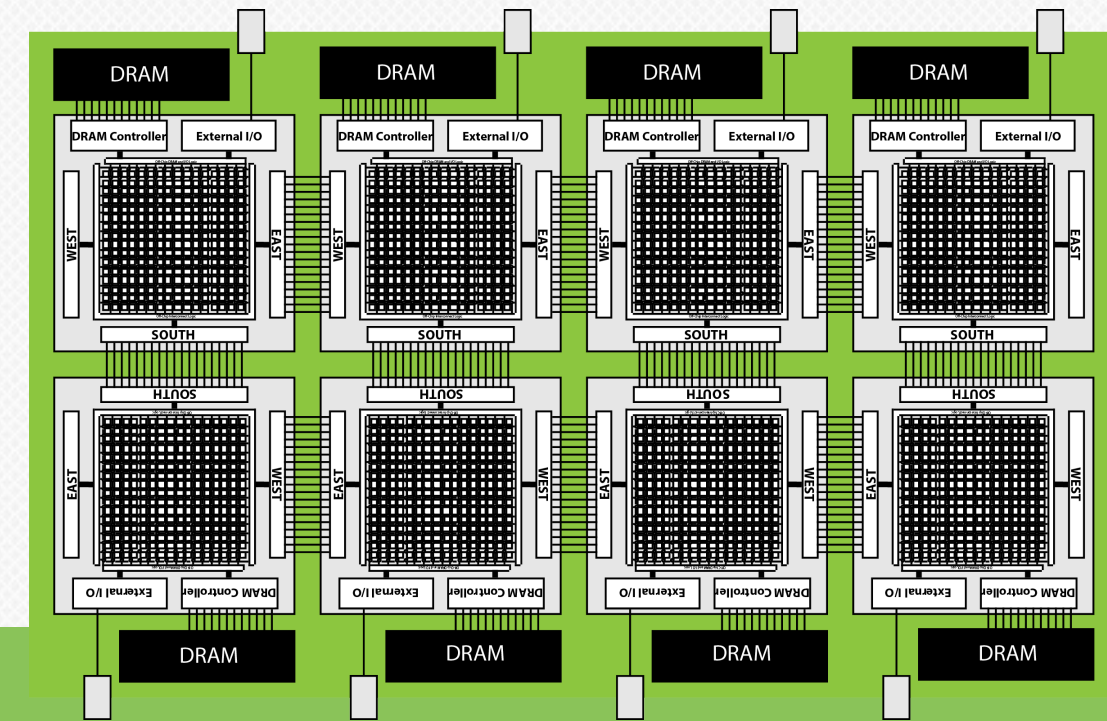
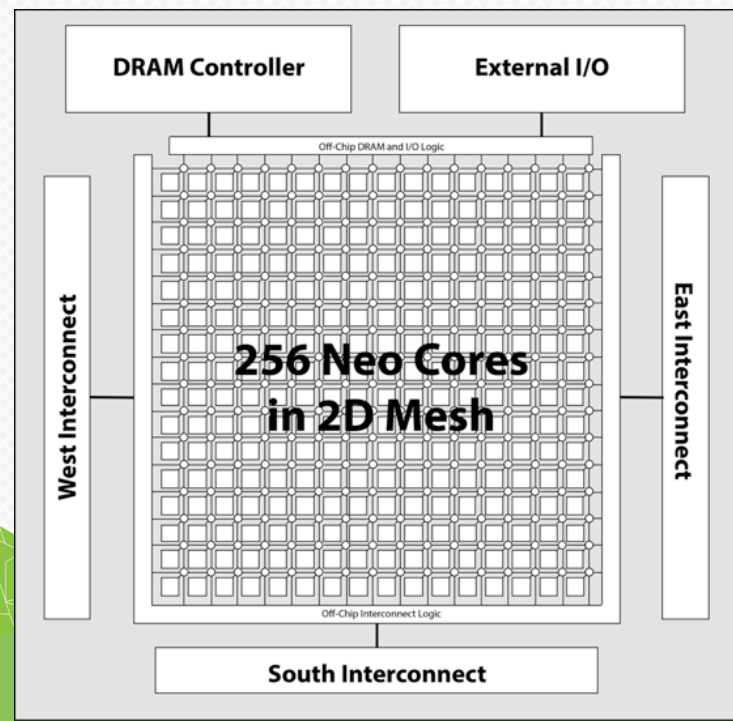


Neo: A massively parallel MIMD processor

256 cores per chip, scratchpad memory, a 2D-mesh interconnect, and a revolutionary high bandwidth chip-to-chip interconnect achieve:

256 GFLOPs DP or 512 GFLOPs SP
at **64 to 128 GFLOPs/Watt**

- Same performance for integer calculations.
- Balanced memory bandwidth allows near-theoretical peak performance.
- Extreme scalability: near limitless number of Neo chips per node.



The Neo Core: Simple, low power, highly scalable

Quad-issue VLIW (But functionally superscalar)

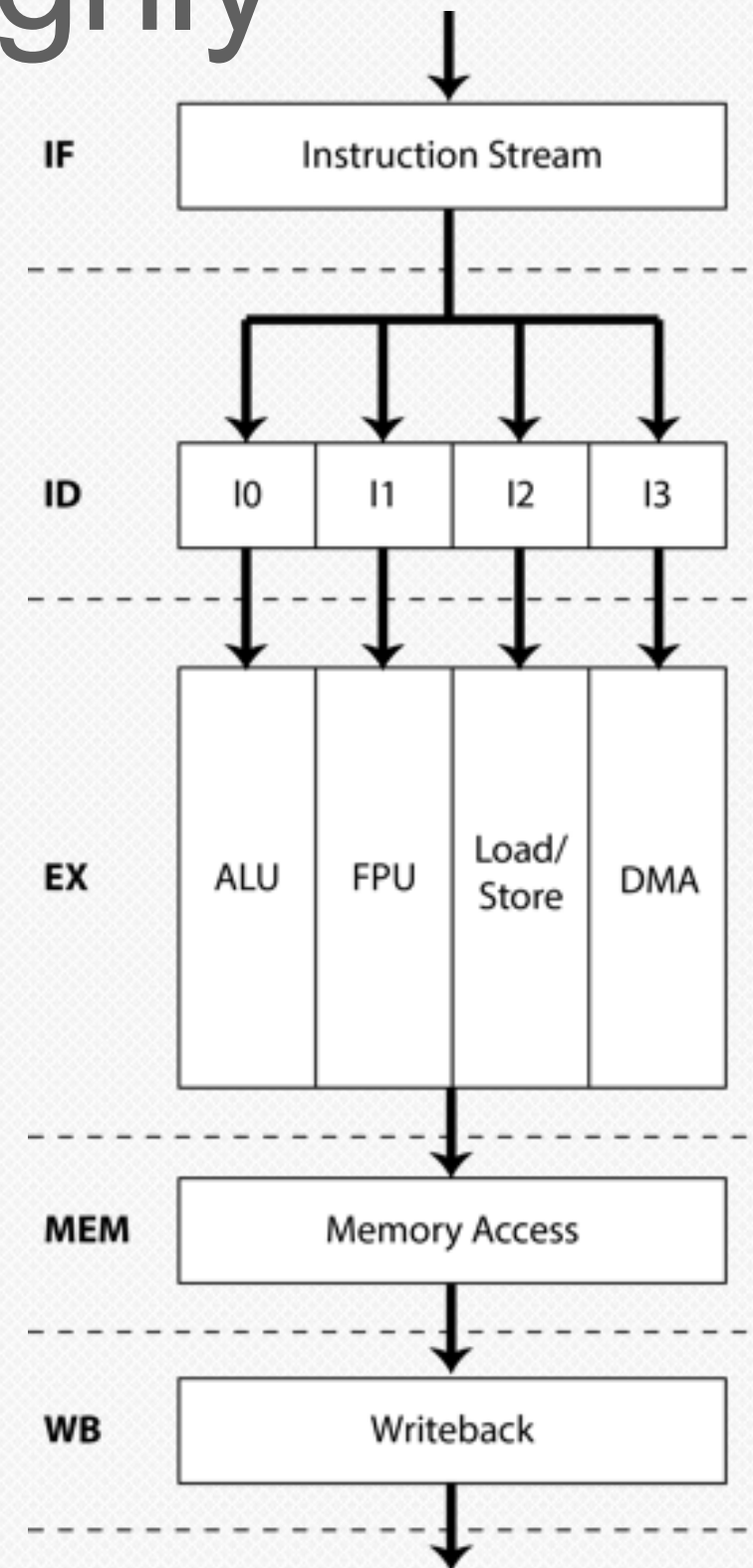
Fixed length 64 bit instructions per functional unit, packed into a 256 bit VLIW.

1. Integer: 64-bit ALU
2. Floating Point: 64-bit IEEE 754-2008 FPU; supports dual issue of 32-bit operations
3. Load / Store Unit
4. DMA Engine with Network-on-Chip Access

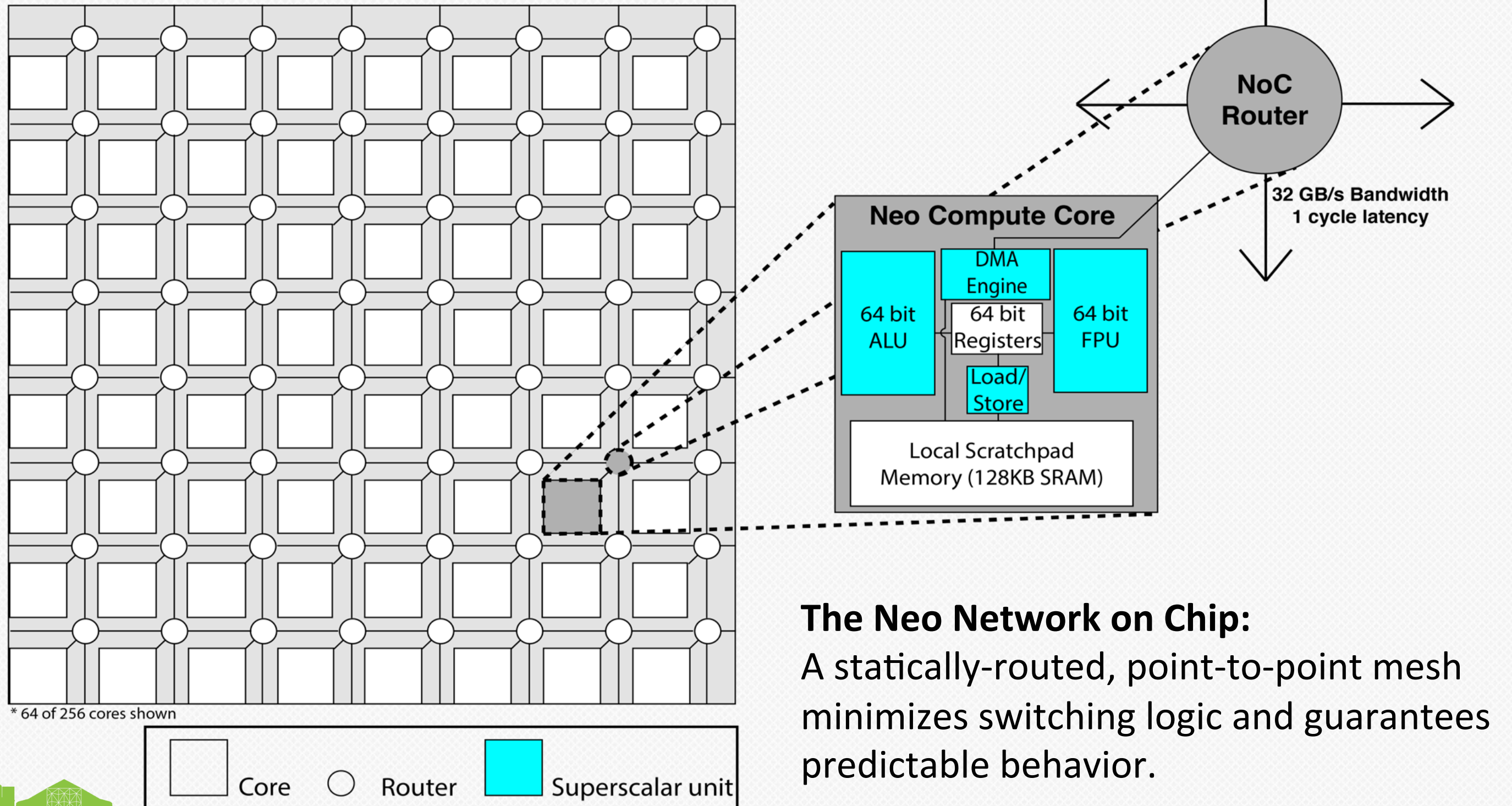
Large software-managed scratchpad

- 128 kbytes of local memory (SRAM) per core
- Partitioned Global Address Space (PGAS) across chips
- Excluding hardware-managed cache hierarchy enables 5x higher memory energy efficiency.

Designed for easy debugging: Includes interrupt controller and debug unit with multiple timers.



2D-mesh network scales to any problem size



The Neo Network on Chip:

A statically-routed, point-to-point mesh minimizes switching logic and guarantees predictable behavior.



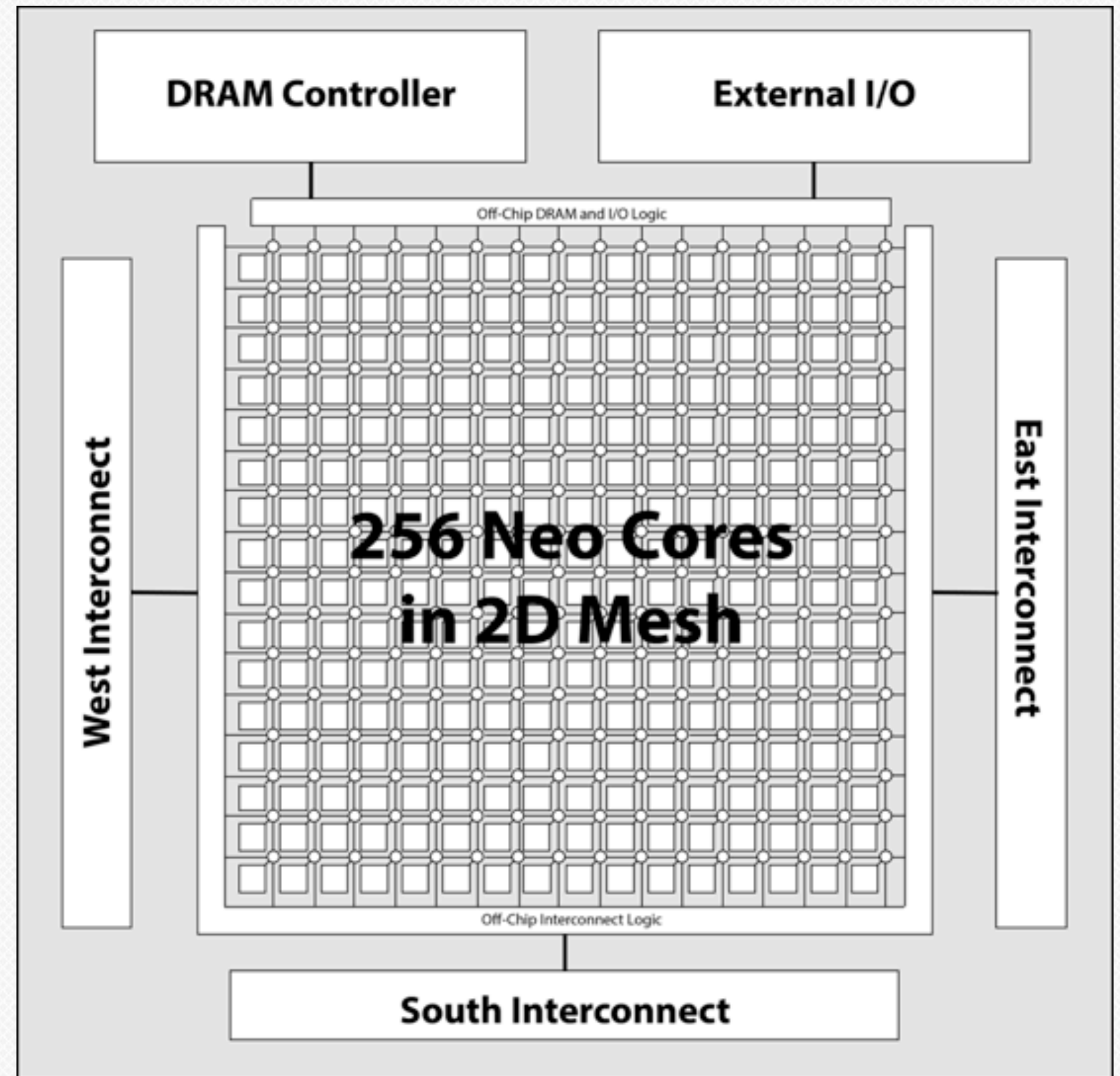
The Neo chip balanced compute with memory and I/O

Software-managed memory hierarchy:

- 128K local scratchpad per core (32 megabytes L1 equivalent per chip)
- DMA access to any core on any connected chip
- Chip-attached DRAM (DDR4/HMC/HMB/Tezzaron/Other)

Flexible, manycore MIMD design supports many programming models, including PGAS, SHMEM, and Systolic Array.

High Bandwidth chip-to-chip interconnect moves up to **288 GB/s** (aggregate chip bandwidth).



Familiar tools and software environment

Portable Applications

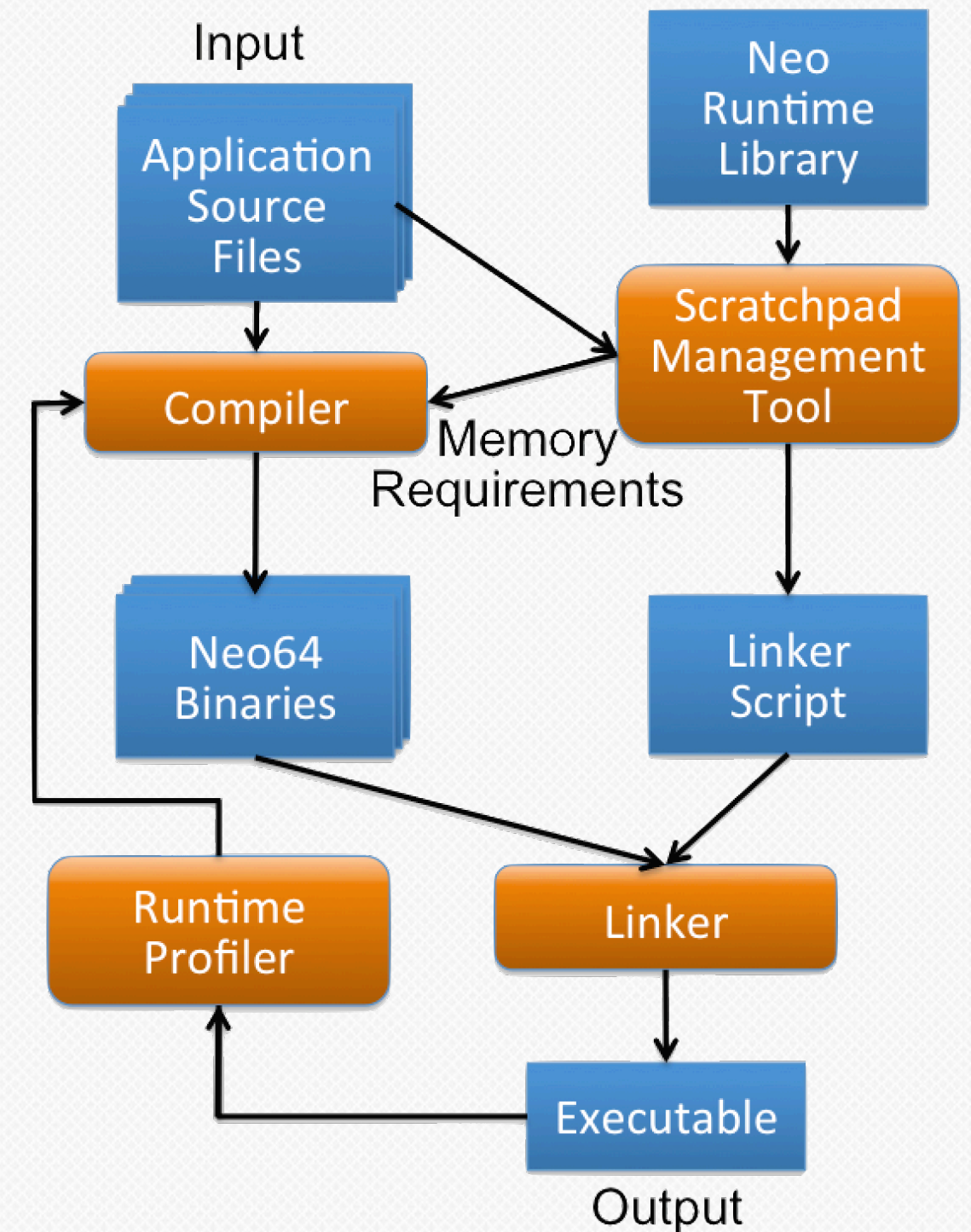
- Neo cores are capable of running a minimal (μ C) Linux environment.
- Targeting the MIMD Neo architecture can be as simple as cross-compiling existing code.
- Compatible with existing IDEs and debugging tools.

Performance Tuning Suite

- Compiler support for advanced automatic scratchpad management (integrated with GCC and LLVM).
- Code profiling tools enable evidence-driven optimization.

Complete Control Over Hardware

- Optional access to unique features, including explicit data flow management and static scheduling.



Scratchpad management software

Automated Scratchpad Management Tool

- Treats scratchpad as a software-managed memory space with similar capabilities as a traditional L1 cache, but with explicit data movement.
- Integrated statically at the compiler level, and dynamically at runtime.
- Capable of estimating and managing memory requirements for stack and heap data.
- Memory space information is passed on internally to the compiler to support runtime integration logic and externally, in the form of linker description specifications.

Stack and Heap management

- Stack utilization is implemented dynamically (e.g. consider the runtime allocation of a function frame) and must be available in the local memory space of an executing application.
- Smart Stack Data Management extends the management granularity to the entire stack space (from function-level), removes the pointer threat problem by providing automated pointer management, simplifies the library to use a linear queue to automate insertion of API calls.
- Heap data is also dynamically managed, and has initially been implemented by modifying the *_malloc* and *_free* functions in GCC and adding API functions to manage pointer conversions between local and global address spaces according to local memory availability.



Further software plans

Memory related language extensions

- An extension of the C (and/or other languages) to enable programmers to unambiguously record assumptions about the memory hierarchy of the architecture and give feedback regarding efficiency of memory usage by a combination of compiler additions and static analysis of source code.
- This may be thought of as a collection of features similar to the `__builtin_expect` construct offered by GCC or as an expanded set of code assertions like those supported by many C, C++, and Java compilers – inclusion in code should be easily understood and low-effort, but the conveyed meaning unambiguous and powerful. The hierarchy will likely include registers, scratchpad memory, main DRAM, and mass storage. The declaration of data structures as being in one of those regimes makes the communication and I/O costs explicit and possible to analyze, allowing the programmer and the compiler to be able to make decisions based on the power and time costs for fetching memory that is far away.

Full LLVM backend

- The advantages of LLVM are clear, there is just a switchover cost, which is relatively minimum at the moment. This will most likely happen before further software tool progress.



Ecosystem development

To compete in an ecosystem with 90%+ marketshare of one ISA,

- REX plans on open sourcing the Neo64 ISA through the Open Compute Project. Licensing terms are TBD, but will be on the permissive side.
- Is fully extendable, but many architectural design decisions are based on scratchpad memory and specific HPC requirements
- ISA spec will be released soon™... latest realistic release is ~June (when we anticipate a lock down on design)
- Read: “Instruction Sets Should be Free: The Case for RISC-V”

Open hardware companies can exist (and be profitable!)

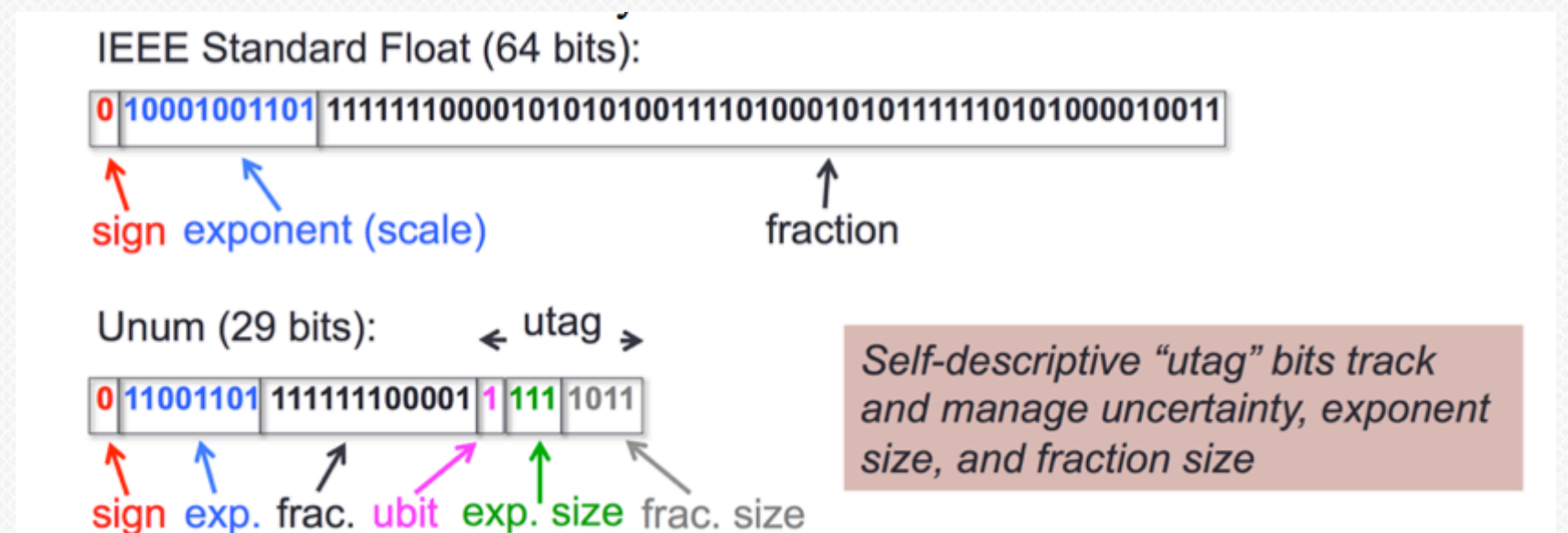
- The Red Hat model is to provide enterprise support for free software. How is it possible for a company to succeed if it opens itself to more competition?
 - It needs to have a technical advantage.
- The benefits of growing the ecosystem (and thus making it a lower risk to change architectures) outweigh the chance of losing a technical advantage early in the game.
- Licensing is always an option



Further hardware plans

(Potential) change from IEEE Float to Unum (<http://tiny.cc/unumSOS>)

- Designed by John Gustafson, and described by him as “Unum is to floating point what floating point is to integer”
- Unum is a superset of the IEEE 754-2008 and 1788 standards.
- Removes inexact rounding and many unnecessary “features” of IEEE Float
- In most scenarios, uses fewer bits than IEEE Float
- There is a potential ~30% power savings on all FLOPs by using Unum, while also increasing precision



Wrap around NoC

- Diagonal wiring of the network on chip could allow a 1D torus going around the Network on Chip
- Minimal increase in wiring while reducing maximum number of hops in half.



Neo Development Kit expected in mid-2016

First shuttle run (planned for Q1 2016) will cost ~\$250k and produce 100 16-core prototype chips.

These chips (estimated die size of 12 mm²) will validate the system in silicon, and pave the way to the first full production run of the 256 core Neo chips 6 to 9 months later.

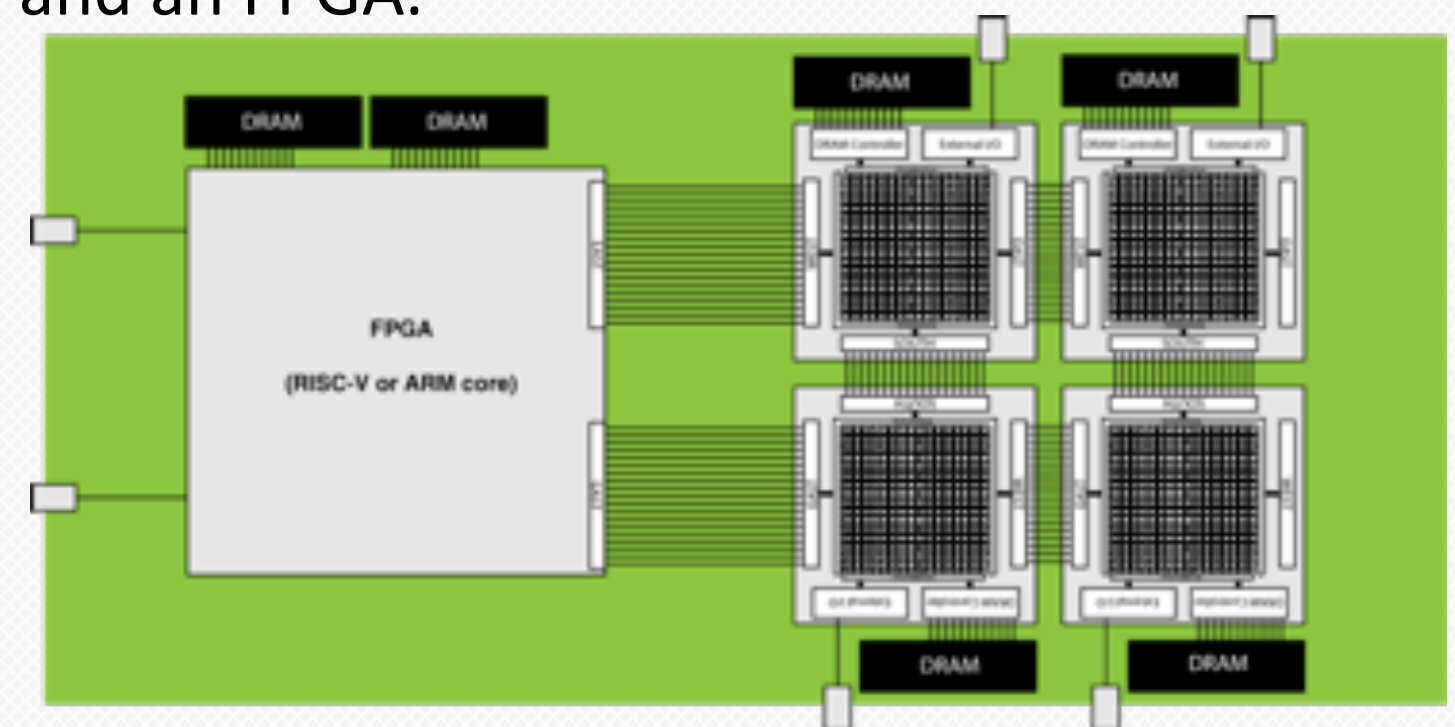
Neo Evaluation and Development Kit:

Select customers will be eligible to purchase or lease development units and software, which will include four 16-core prototype chips and an FPGA.

Early Access Tools:

(Available mid 2015)

1. Neo ISA simulator
2. Neo compute core FPGA implementation
3. Cycle-accurate Neo chip simulator
4. QEMU Neo full-system virtualization



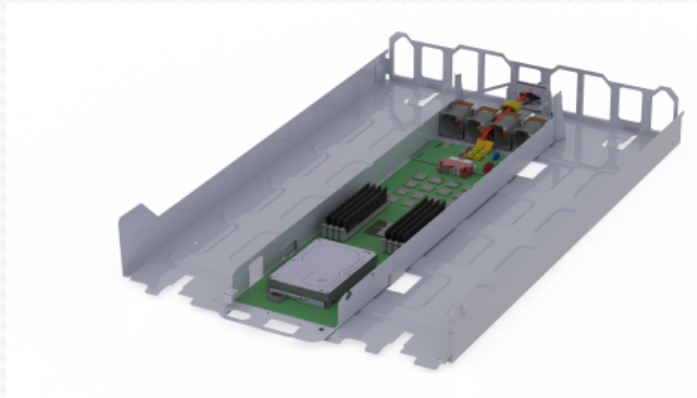
Full 256 core Neo expected in 2017

The Neo Compute Chip (1GHz, 256 core, 128k scratchpad per core, TSMC 28nm):

- 4 Watt TDP
- Estimated core size: 0.1 mm²
- Estimated scratchpad size .15 mm²
- Estimated full die size: 80 mm²-100 mm² (smaller than a modern smartphone processor)
- 256 GFLOPs (DP/64)
 - Same for integer
 - **64 GFLOPs/Watt**
- 512 GFLOPs (SP/32)
 - Same for integer
 - **128 GFLOPs/Watt**
- 32GB/s core to core bandwidth; 1 cycle per hop for Network on Chip
- Balanced off-chip bandwidth:
 - 32 to 48 GB/s per direction, per interface.
 - 192 to 288 GB/s aggregate chip-to-chip bandwidth
- DDR3/DDR4/GDDR5/HMC/Other stacked DRAM (TBD). Anywhere from 25GB/s to 512GB/s.
- External I/O is dependent on customer. Can be changed between 10/40/100gbit, Ethernet, Infiniband, SRIO, or raw SerDes.



(A very conservative) Neo Rack



One Neo Node (1/3rd width 10U)

- Fills 1/6 of 20U shelf
- 16 Neo processors
- 4096 cores
- 256 GB of DDR4 memory

4 DP TFLOPS

80 Watts



One Neo Rack (Open Compute)

- 90 Neo nodes
- 1,440 Neo processors
- 369,000 cores
- 22.5 TB of DDR4 memory

360 DP TFLOPS


7.2 Kilowatts

50 GFLOPS/Watt (DP) node level




REX Neo: Evolving to Exascale by 2020

Projected Neo evolution:

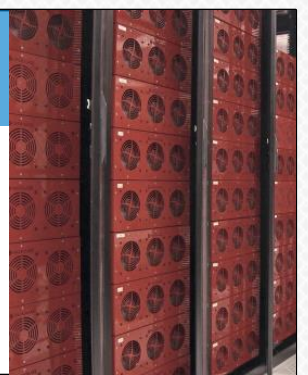


| | Racks | PFLOPS | Power (MW) | GFLOPS /W |
|--|-------|--------|------------|-----------|
| REX Neo (3 rd Gen) (circa 2020) | 350 | 1000 | 10 | 100 |
| | ▲ | ▲ | ▲ | ▲ |
| REX Neo (2 nd Gen) (2018/19) | 2800 | 1000 | 15 | 65 |
| | ▲ | ▲ | ▲ | ▲ |
| REX Neo (2017) | 10 | 3.6 | .072 | 50 |



The first practical exascale solution.

- 350 Neo racks
- 50 petabytes of storage
- Power budget of 10MW




REX Computing is a new approach to Fabless Semi

Why do we think we can do this?

New tools:

Chisel (UC Berkeley ASPIRE Lab) allowed us to have a prototype core's RTL be produced in under 4 months.

Simplicity significantly reduces development time:

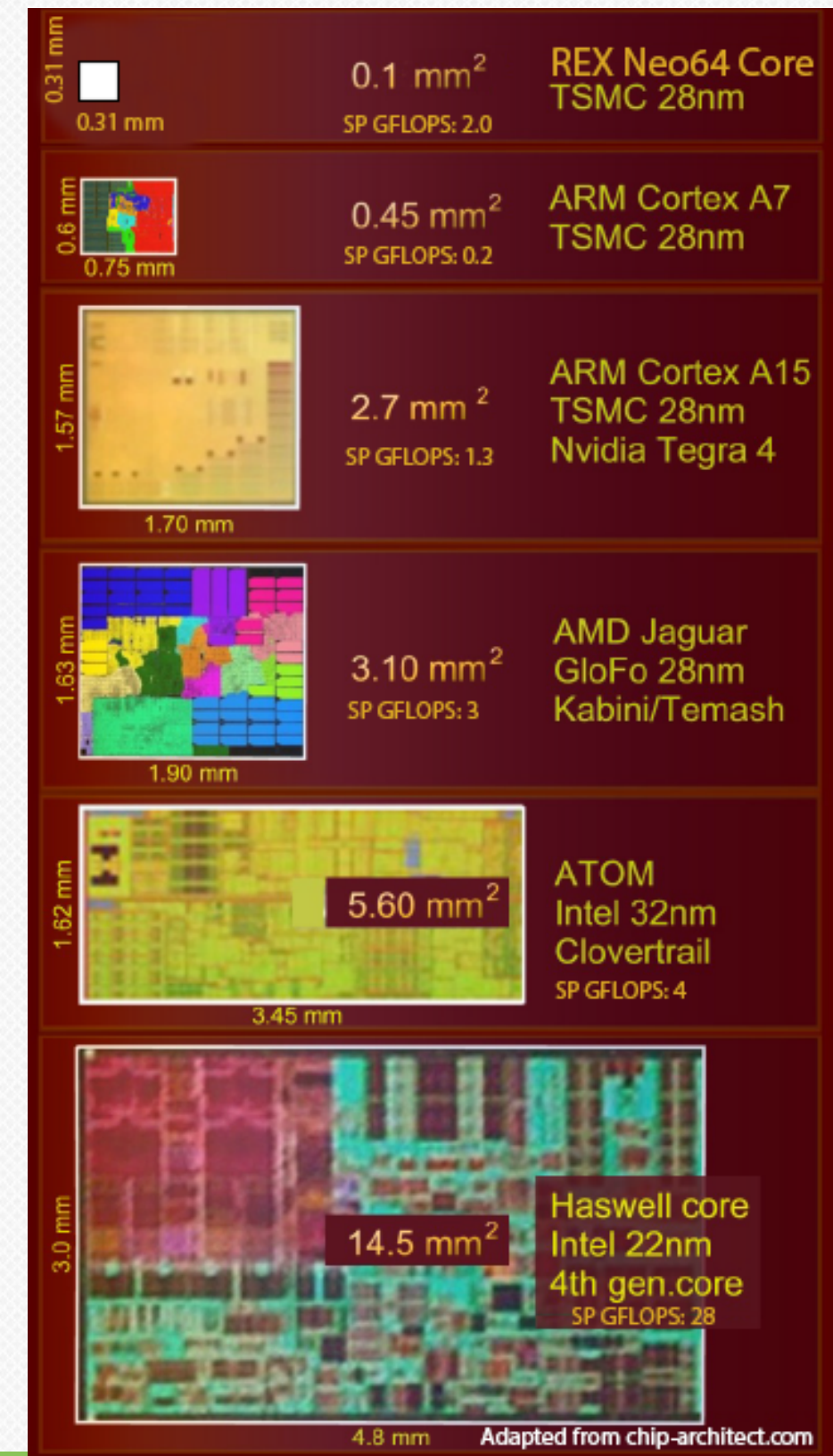
The 64-bit Neo core is 1/145th the size of a 4th-gen Intel Haswell core or 1/27th the size of a 32-bit ARM Cortex A15 core.

Unique focus on software:

REX understands that software is key to driving customer adoption; hardware is only as valuable as the applications it enables.

A fresh perspective:

REX is free from the popular misconception that semiconductor development requires large teams and complex bureaucracy, and that all the easy solutions in chip design have been found.





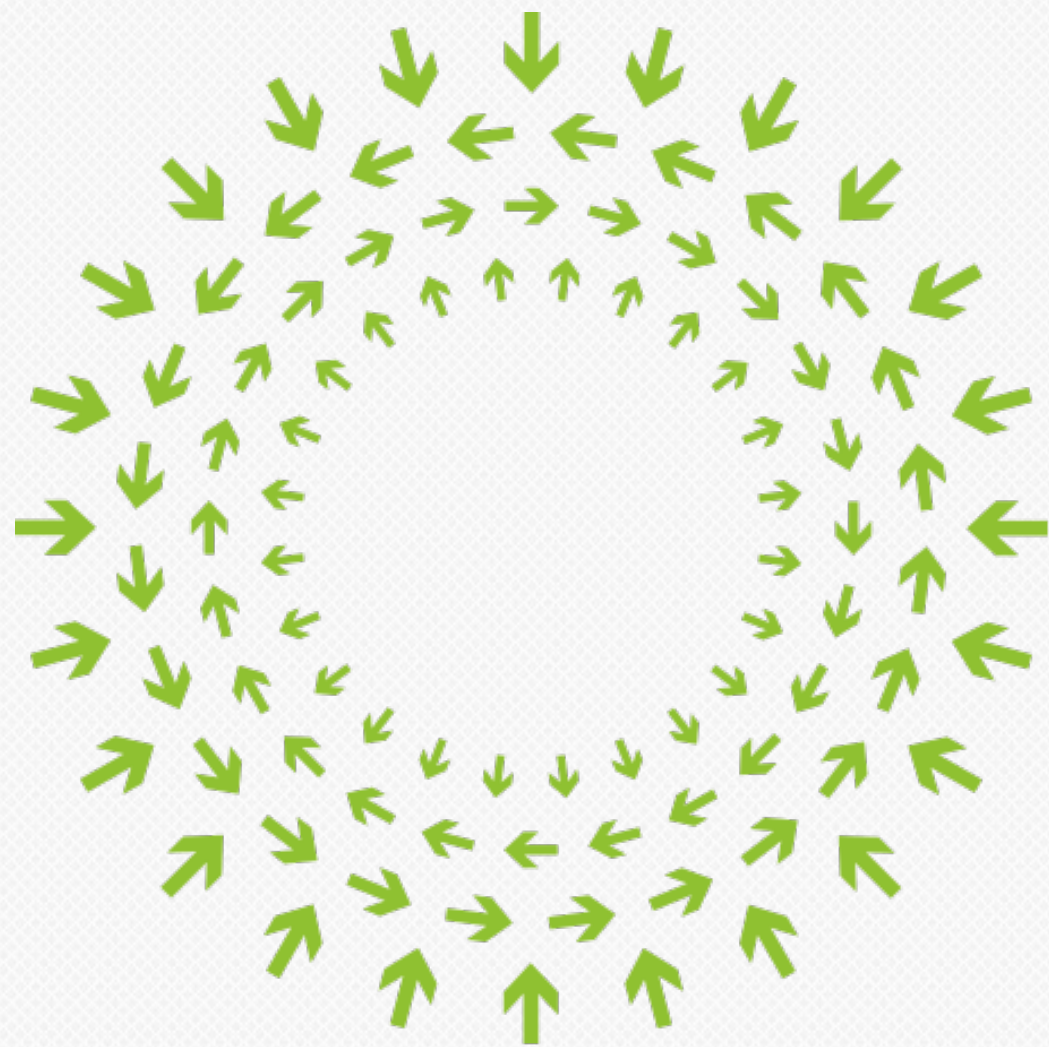
Why a new ISA?

(why not ARM?)



Software?

| Chip | Node | Cores | GF (FP32) | GF (FP64) | Memory Bandwidth | Watt | GF/W (32) | GF/W (64) | Price |
|---------------------------------------|-------------|--------------------------|--------------|--------------|-----------------------|----------|--------------|--------------|---------|
| Intel Xeon E5 (2014) | 22nm | 18 | 1012 | 506 | 68 GB/s | 145 | 6.9 | 3.4 | \$4,500 |
| Xeon Phi 7120 (2014) | 22nm | 61 | 1900 | 1200 | 352 GB/s | 300 | 6.3 | 4 | \$4,235 |
| Xeon Phi Knights Landing (2015) | 14nm | 72 | ~5000 | ~3000 | ~600 GB/s | 215 | 23.2 | 13.9 | 2016 |
| NVIDIA K40 (2013) | 28nm | 2880 | 4290 | 1430 | 288 GB/s | 245 | 17.5 | 5.8 | \$3,000 |
| NVIDIA K80 (2015) | 28nm | 2496*2 (multi GPU) | 8740 | 2910 | 240 GB/s (per GPU) | 300 | 29.1 | 9.7 | \$5,000 |
| Adapteva Epiphany IV | 28nm | 64 | 102 | N/A | 6.25 GB/s | 2 | 51 | N/A | \$750 |
| Kalray MPPA-256 | 28nm | 256 | 230 | 43 | 180 GB/s | 11 | 20.9 | 3.9 | \$2,000 |
| REX Neo | 28nm | 256 | 512 | 256 | 288 GB/s | 4 | 128 | 64 | TBD |



OPEN

Compute Summit

March 10–11, 2015

San Jose

