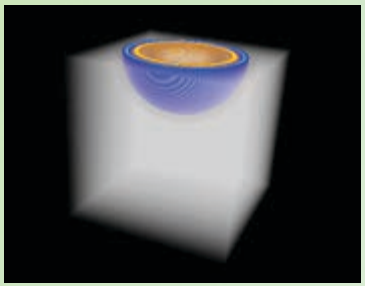


Build an Energy Efficient Supercomputer from Items You can Find in Your Home (Sort of)

Marty Deneroff
Chief Technology Officer
Green Wave Systems, Inc.
deneroff@grnwv.com

Using COTS Intellectual Property, mostly from the Consumer Electronics design space, and incorporating state of the art memory components and a bit of creativity, it is possible to design a computer that has better energy efficiency, compute density, and overall performance than can be achieved with traditional approaches.

How to Build a Supercomputer?

Since ~2000, “Supercomputers” have looked essentially identical to commercial servers.

- Intel Xeon™ or IBM Power™ processors
- Standard DDR DIMMs
- Small-way (<=8) SMPs connected with LAN networks
- Standard Interconnects (perhaps Infiniband™ instead of Ethernet)
- Recently we are adding GPGPUs

How to Build a Supercomputer?

Before that, there were

- Vector processors, perhaps interconnected as Small-way SMPs (*ie* Cray T90™)
- Collections of standard CPUs interconnected with specialized networks (*ie* Cray T3E™)

Are any of these approaches right for today?

How we got here

Transition is driven by economics!

Although the cost of producing a chip of a given size has remained fairly constant, the development cost has not.

The cost of developing a custom chip rises with each generation; the increase is tied to, but not the same as, Moore's Law

- As geometry shrinks, the cost of masks goes up – ~25% / generation
- If the number of transistors in a design doubles, more man-hours are required. Improved techniques help, but don't keep up with doubled transistors.

How we got here

By 2000, development cost of custom HPC computers was overwhelming. No one could do it at an affordable price and make a profit.

Those who attempted it couldn’t keep up with state-of-the-art in chip design, because they couldn’t support a large enough design team.

How we got here

Thus, the only viable choice was to use processor chips designed for a larger market than HPC.

Hence the transition to Xeons and GPGPUs.

Where do we go now?

Today, even Intel is having trouble affording new generations.

- Volume in PCs is flat, <10^7/year
- Moore's Law has arguably ended
  - Since 28 nm, $/transistor has been increasing each generation
  - Engineering effort continues to rise each generation

Where do we go now?

The Consumer Market* is much larger (~10X or more) than the PC market.

- Requires very fast product cycles -> simplified design processes
- Huge investments in off-the-shelf resalable IP

*smart phones, tablets, set-top boxes, game consoles, etc.

Can we now use IP from the Consumer space to do HPC, with better economics?

We will need the following items:

- ✓ CPU Core
- ✓ Memory Controller
- ✓ NOC – *yes, but may choose to design our own*
- ✓ External (Chip to Chip) Interconnect
- ❑ Memory System – *not a good match to Consumer space*
- ✓ IO Interfaces
- ❑ DMA Engine – *may have to design this one*

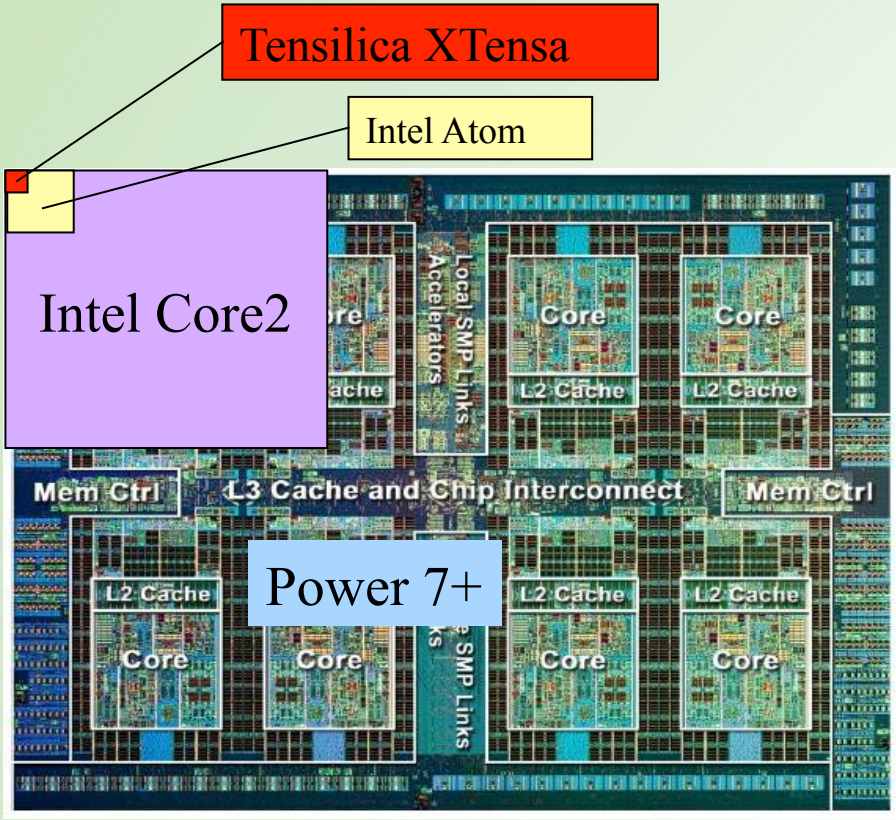
IP seems to exist – Does it meet our needs?

Costs

Power consumption has become a real problem:

- What fits on a chip often limited by heat removal considerations, not available transistors
- Operation of a 3MW data center has $3M energy cost/year (comparable to the cost of the equipment)
- At Exascale, hard limits (~20 MW) on how much power can be delivered, at any cost, come into play

How do embedded cores compare to other cores?



IBM Power7+™ (server):
- 60W/core@4400MHz – 600W/chip
- 8 cores / chip

Intel Core2 SC™ (laptop):
- 5W/core@1000MHz – 120 W/chip
- ~16 cores fit in equivalent size

Intel Atom™ (embedded):
- 0.625W/core@800MHz – 62.5 W/chip
- ~100 cores fit in equivalent size

Tensilica XTensa LX5™ :
- 0.09W/core@1500MHz – 63 W/chip
- ~700 cores fit in equivalent size

Embedded is More Effective

	Freq. (MHz)	Peak Flop / Cyc.	Effic. Factor	Avg. GFlop / Core	Cores / Chip	Avg. GFlop / Chip	Avg. GFlop / Watt	Peak GFlop / Chip
Power 7+	4400	4	0.4	7.04	8	56.32	0.118	140
Tesla K40 + Xeon	745	2	0.05	0.075	2880	214.56	0.596	4291
Core2 SC	1000	4	0.25	1.0	16	16.0	0.200	64
Atom	800	2	0.25	0.4	100	40.0	0.64	160
XTensa LX5	1500	4	0.25	1.5	700	1050.0	16.667	4200

If we can program it effectively:

- *Tensilica LX5 achieves ~ 18x average flops/chip of Power7+*
- *Tensilica LX5 achieves ~ 177x average flops/watt of Power7+*

This is if we provide sufficient memory bandwidth to feed all those cores

What does an Exaflop look like?

	Avg. GFLOP / Core	Cores / Exaflop	Chips / Exaflop	Watts / Core	MWatts / Exaflop
Power 7+	7.04	142,245,454	17,755,682	60	1066
Tesla K40 with Xeon	0.0745	13,422,818,791	4,660,701*	0.078	1049
Core2 SC	1.0	1,000,000,000	62,500,000	5	5000
Atom	0.4	2,500,000,000	25,000,000	0.625	1562
XTensa LX5	1.5	666,666,666	952,381	0.09	60 !!!

* *GPGPU only – does not include host cpu in chip count*

Note: These cores are for different markets and implemented in different processes. There is no claim that this is truly apples to apples.

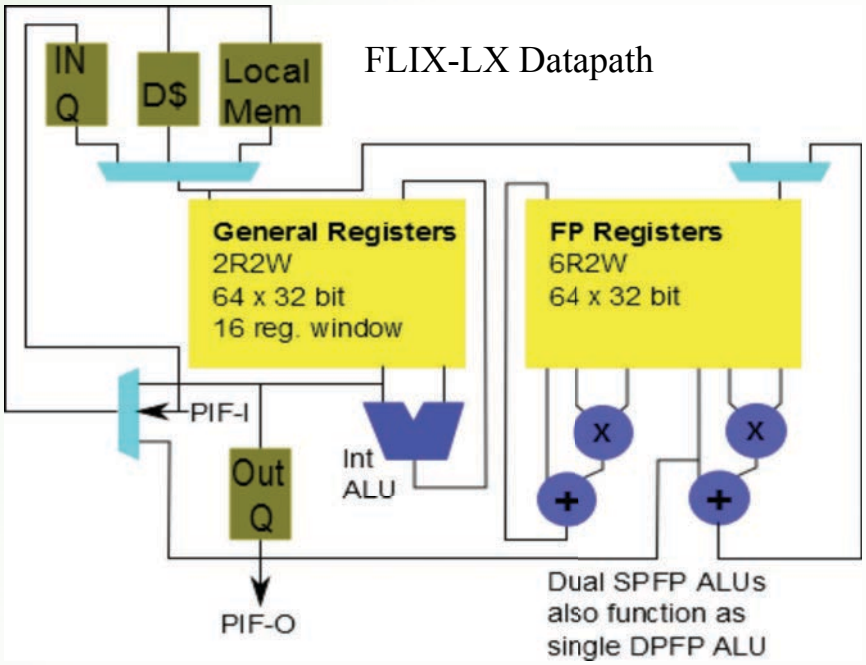
Why Does Tensilica Core Look This Good?

Core is designed for use in Cell Phones, Tablets, etc.

- Base design is minimalist, and extremely low power
- Includes capability to add functionality from “check-off” menu tool – can add VLIW FP datapaths, alter register file depths, add local memories and caches, special addressing modes, etc.
  - Appropriately targeted compiler automatically generated as core is configured
  - Selecting options doesn’t require a verification effort – tool guarantees correctness
- 3-way VLIW structure supports very high ratio of compute logic to control logic

Tensilica LX5-FLIX Compute Core

- 3-way VLIW with 2 Floating Point + 1 Integer instructions per clock (64 bit instruction bundles)
- 64 Floating Point Registers
- 64 General Registers, with 16 register rotating window
- 2 Single Precision or 1 Double Precision ops per clock, including MADD
- Specialized stencil address calculation instructions
- 2 Bank local store, 128k total
- 2-set associative I and D caches, 16 kb/cache
- Incoming and Outgoing Message Queues for on-chip communication
- Programmable clock rate for energy saving
- Software cache coherency
- 32 bit address space provides access to all on-chip resources and several large segments located in the connected HMCs
- Memory Mapped Register extension provides access to arbitrary 64 bit addresses
- 0.1 mm^2 area
- 90 mW power (including memories)
- 1500 MHz max clock rate



Building a system using these cores

The Green Wave Systems design:

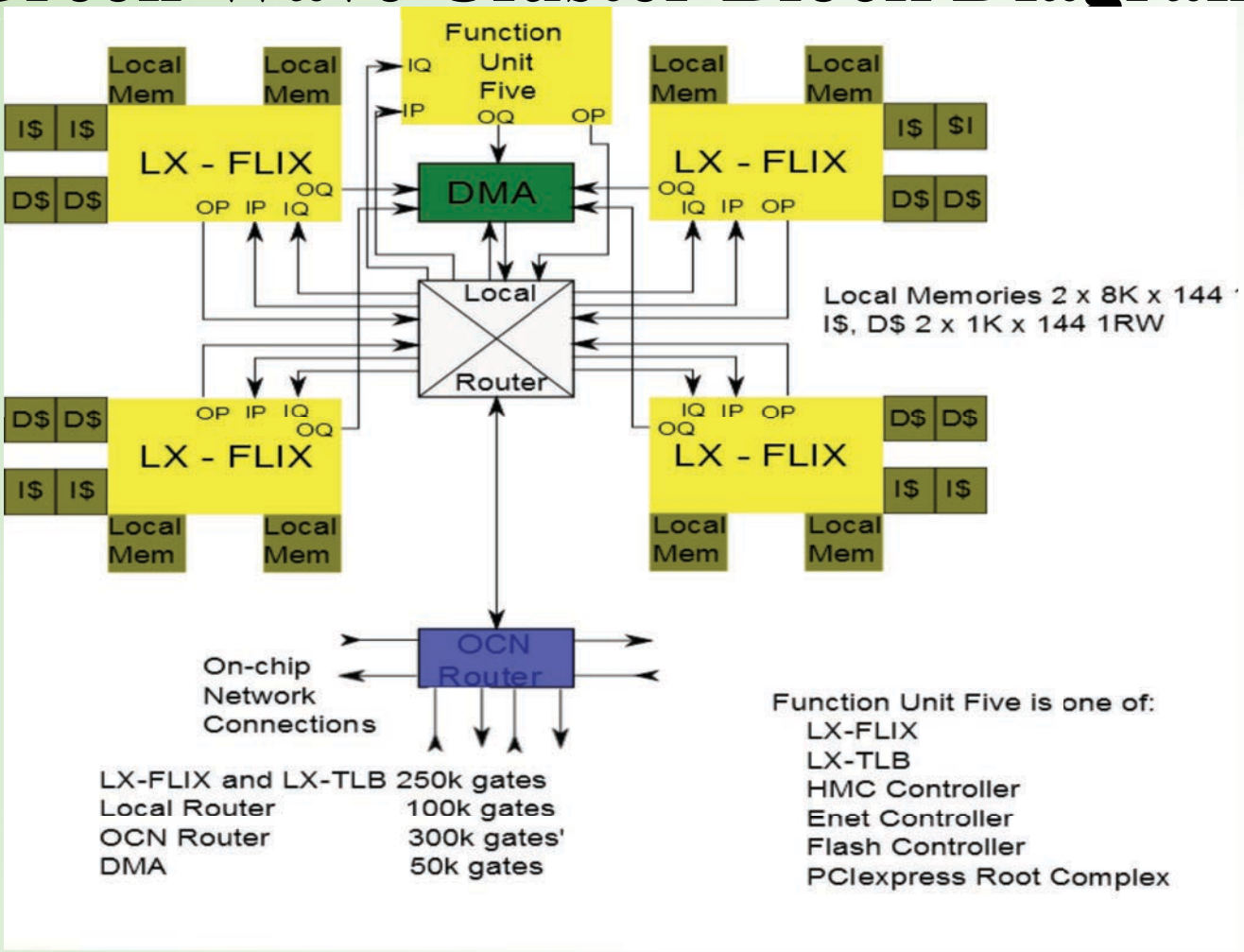
- Implement ~700 compute plus supervisory cores in a single 16nm FinFet ASIC, together with 2D Torus NOC, 8 HMC memory interfaces, and interfaces for PCIexpress IO and Flash memory.
- Implement main memory with Micron's Hybrid Memory Cube (HMC), providing 320 GB/s of bandwidth per ASIC, with very low power
- ASICs and HMCs are connected together in a 2D torus topology, using the HMCs as crossbars in the interconnect.
- Communication between chips shares the channels to the HMCs with memory traffic, maximizing efficiency and minimizing pin-count.

Building a system using these cores

The Green Wave Systems design – ASIC Architecture:

- A group of five cores, a local switch, and a DMA engine constitute a *Cluster*
- Clusters are connected to each other using the NOC, a *2D Folded Torus*
- Nearly everything on the chip is COTS IP. Only the DMA Engine and NOC are new design (IP for NOCs is available, but appears not to be up to this task at present time.).
- We expect to implement the entire chip and system with ~50 man-years effort. By comparison, implementing a sophisticated core like Xeon or P7+ requires hundreds of man-years for the core alone.

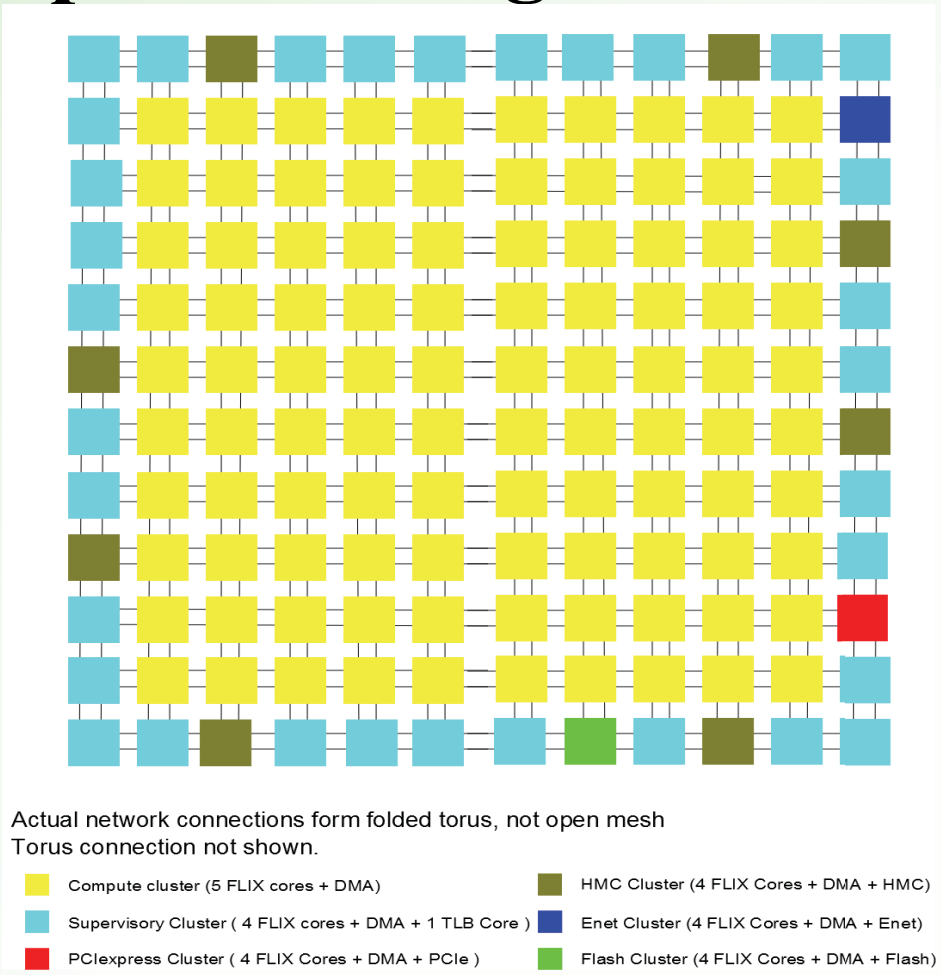
Green Wave Cluster Block Diagram



Green Wave Chip Block Diagram

- 12 x 12 2D on-chip torus network
- 676 Compute cores (500 in compute clusters, 176 in peripheral clusters)
- 33 Supervisory cores
- 1 PCIexpress interface (16 bit Gen 3)
- 8 Hybrid Memory Cube (HMC) interfaces
- 1 Flash controller
- 1 1000BaseT ethernet controller

It is not anticipated that all cores will be utilized – some are spares for yield enhancement.



Inter-Chip Communications

2D Torus implemented using store-and-forward mailbox areas in HMCs

- Shares HMC channels for memory and communications
- Managed by one or more supervisory cores on each chip
- Uses DMA engines for large transfers
- Capable of 40 GB/s between neighbors
- Emulates standard networking protocols at much higher bandwidth/lower latency

The Green Wave Computer

ASIC : 676 Compute Cores,
33 Supervisory cores

2U chassis : 32 GW ASICs, 512 GB Mem

High Speed Interconnect (in chassis):
2 D Torus, 40 GB/S/link (HMC channels)
Infiniband or 10 G enet between chassis

16 chassis/rack

Rack Performance: **~ 16 x 32 x 4.3TF/s (SP) > 2.0 Pflop**

Rack Power: **<48 KW**

Efficiency: **<24 KW/Pflop!**

Conclusion

- It is feasible and compelling to build a supercomputer using commercially available IP, primarily from the consumer product space.
- Development effort of such a system is significantly smaller than required using traditional approaches.
- Achievable energy efficiency approaches the requirements for proposed exascale systems that target 2022 deployment, using technology available now.

Acknowledgements

Lawrence Berkeley National Lab:
John Shalf
David Donofrio

Fraunhoffer Institute:
Franz-Josef Pfreund

Green Wave Systems, Inc.:
Ken Jacobsen
Valerie Deneroff

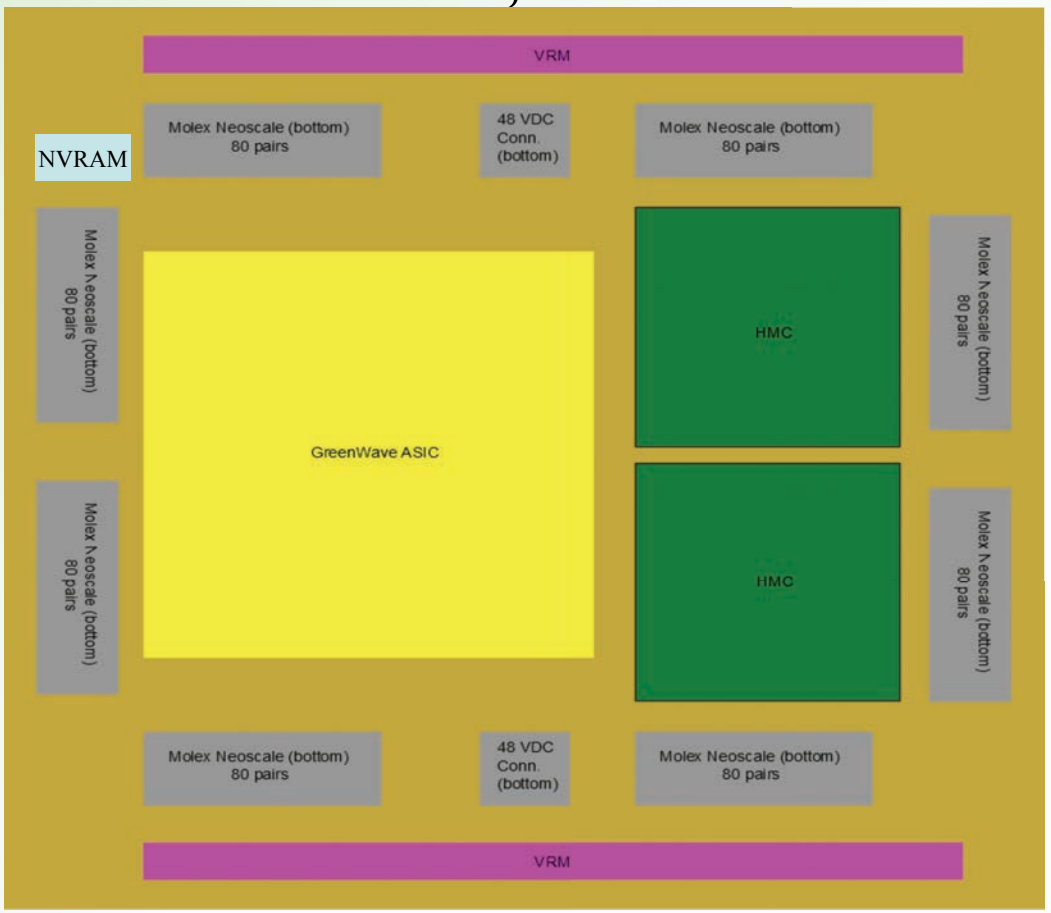
If we build it, can it be programmed?

Okay, the simple embedded cores give us both more FLOPs/chip and more FLOPs/watt. That's good. But we will have many more total cores. Can we program this effectively?

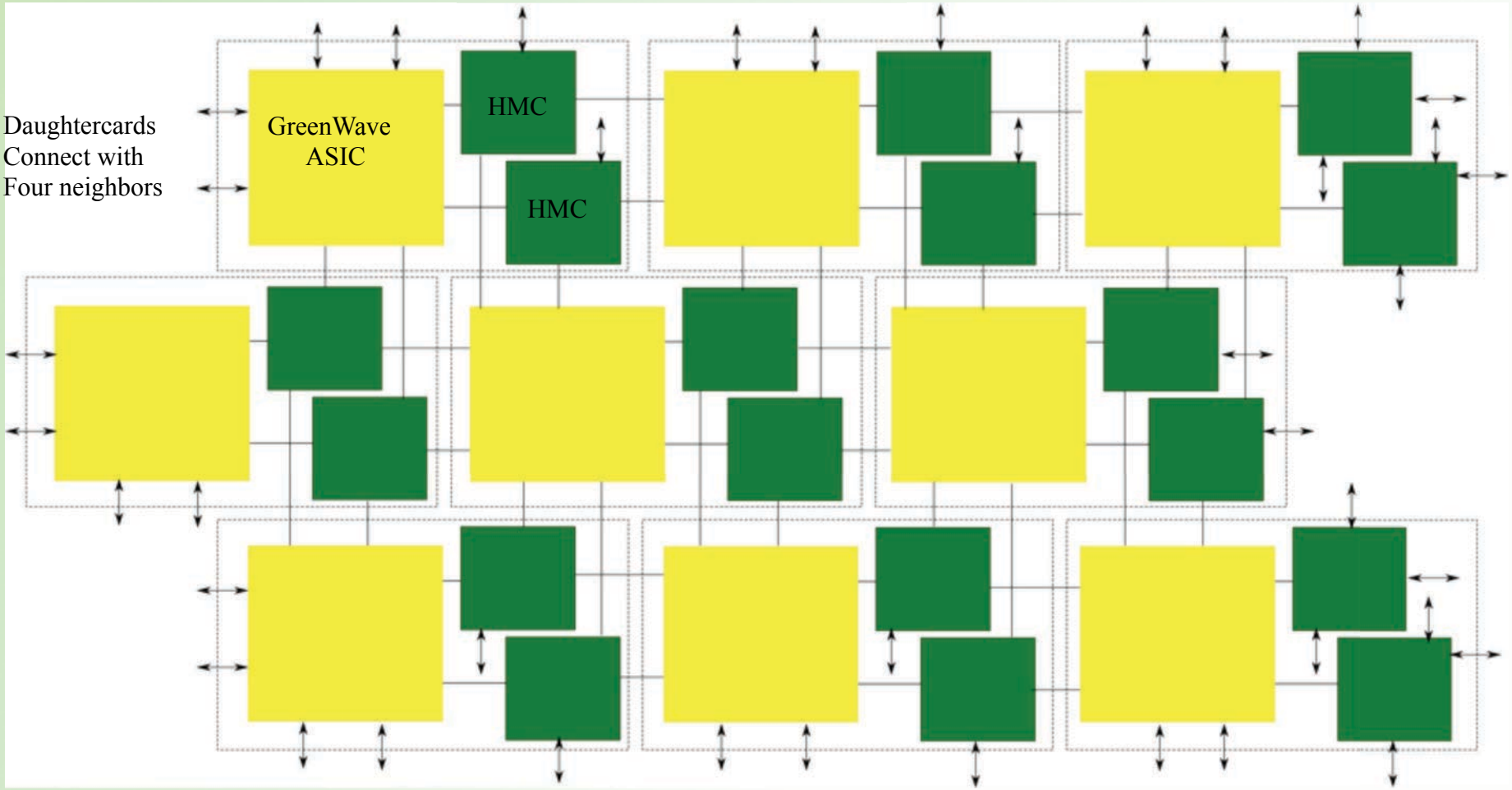
- Clearly 2 cores are harder than 1, and 100 is harder than 10. But *how much harder is 650,000 than 150,000?*
- GPGPUs have even more cores , plus a heterogeneous programming environment (GPGPU + host cpu) and a fragmented memory map (host memory vs "graphics" memory). We seem to be managing to program this.
- To be successful, programming abstractions must treat collections of threads, rather than individuals. Having a homogeneous architecture facilitates these abstractions (but does not solve the problem!).

Green Wave Daughtercard:
1 ASIC, 2 HMC, DC-DC Converters, NVRAM

- Contains Voltage Regulators for direct conversion from 48 VDC to voltages required by ASIC and HMCs
- NVRAM allows implementation of Solid State Disk in software (dedicated cores for this purpose)
- HMC channels connect adjacent daughterboards in X and Y dimensions
- PCIe connections brought off chip to connect IO devices
- ~5” x 5”
- <~70 W peak power, inc. conversion loss
- Small, inexpensive Field Replaceable Unit



Green Wave Interconnect Topology:



Software Environment

- Linux operating system runs on Supervisory Cores
- Lightweight run-time (Open Community Runtime?) on Compute Cores
- Tensilica C/C++ compiler (based on Open64)
- OpenMP, MPI libraries, software based (CRF) cache coherency
- Future LLVM-based compiler to support additional languages, improved optimization
- Reservoir Labs R-Stream auto-parallelizing compiler available
- Possible support for UPC and/or Chapel

Energy Saving Features

- **HMC Memory** uses < 20% energy of DDR4 Memory
- **HMC Channels** shared for inter-ASIC communications
- **Simple In-order VLIW Cores** at moderate clock rate deliver high efficiency at very lower power
- **Adjustable Clock Dividers** per core under program control – reduce clock rate when off critical path or communications/memory bound to further reduce power usage
- **Working Set DMA'd into Local Store** (*ala* Cell) for efficient processing
- **16 FinFET ASIC technology** drastically reduces static power
- **Local Flash Memory** reduces/largely avoids use of spinning disk, provides faster access to data

Green Wave Added Value

- Green Wave will be a homogenious multiprocessor taking advantage of consumer technology
- Green Wave gives >10x improvement in density and power consumption
- Initial implementation is adapted to stencil operations used in wide variety of signal processing, CFD, climate, weather and image applications and can be applied to a variety of big data problems