Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?

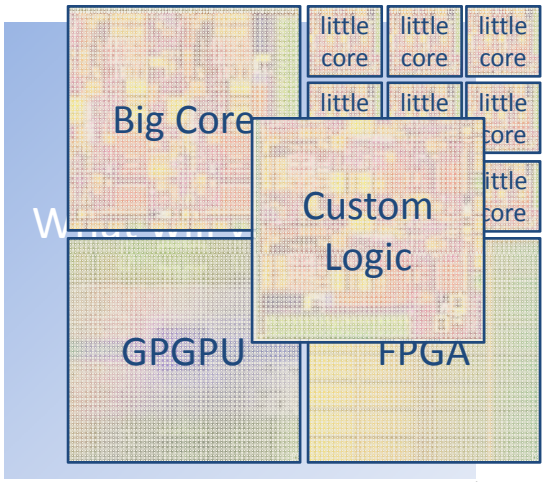
Eric S. Chung, Peter A. Milder,
James C. Hoe, Ken Mai

Computer Architecture Lab at
Carnegie Mellon

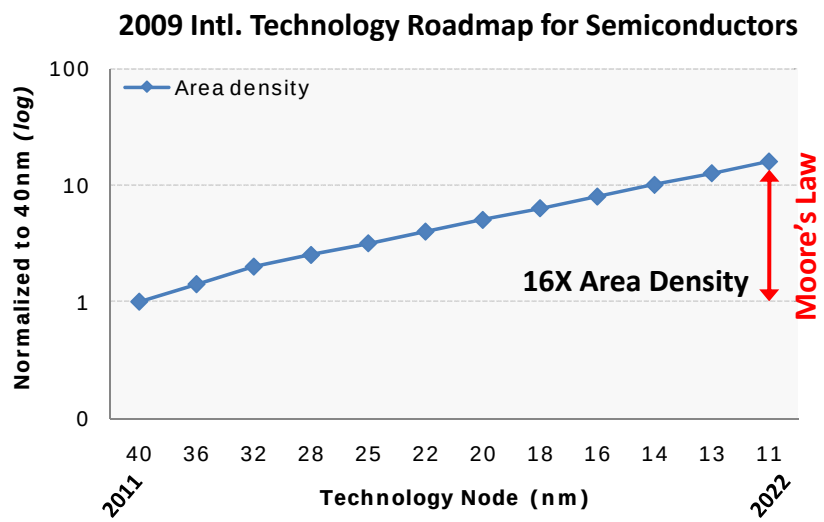
Based on Chung, et al. to appear in Proc. Intl. Symposium on Microarchitecture (MICRO), 2010.

Heterogeneous Potpourri

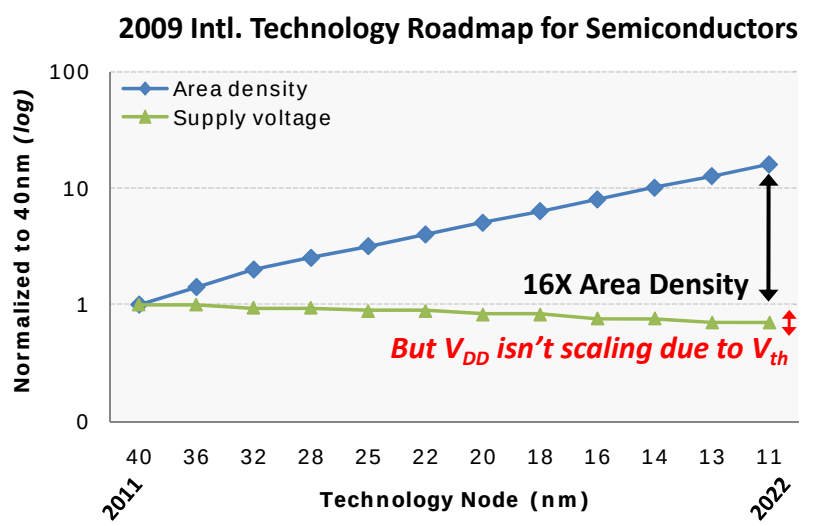
- By 2022 (11nm node), a large die (~500mm^2) will have over 10 billion transistors



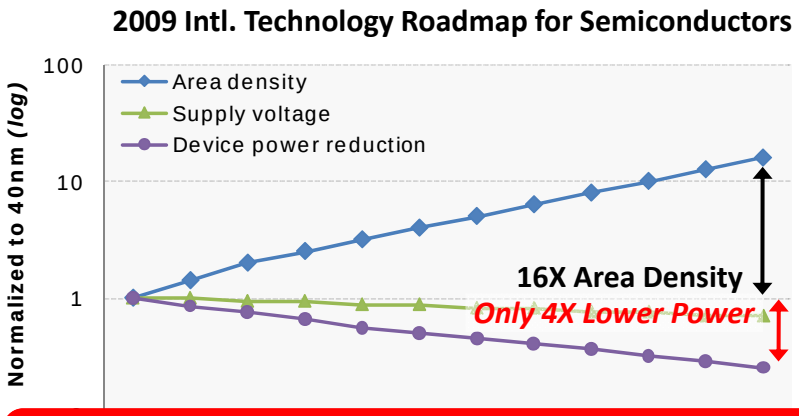
What we "know" about the future



What we "know" about the future



What we "know" about the future



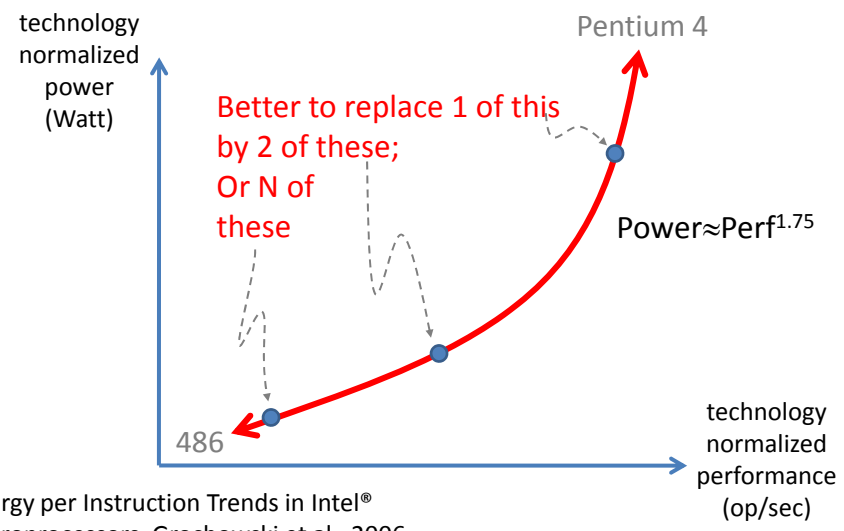
Area is free; power is not

Outline

- Future is about *perf-per-watt* and *op-per-Joule*
  - does the future need more than programmable processors
    Answer: Yes
- Where do FPGAs and GPUs stand today?
  - FPGAs suck at floating-point Answer: No
  - GPUs are power hogs Answer: Not entirely
- Do the future heterogeneous multicores include custom logic, FPGAs, and GPGPUs?
  Answer: . . . . . .

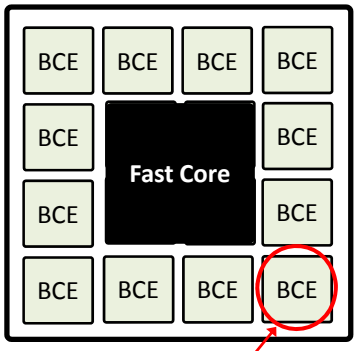
Disclosure: I am a big fan of reconfigurable logic

Energy Efficiency of Programmable Processors



Energy per Instruction Trends in Intel® Microprocessors, Grochowski et al., 2006

Asymmetric Multicores



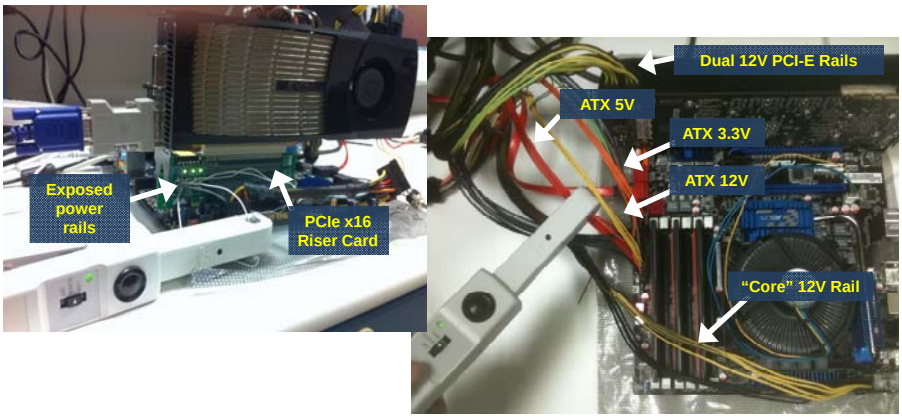
Base Core Equivalent (BCE) in [Hill and Marty, 2008]

- Combine a power-hungry "fast" core with lots of simple, power-efficient "slow" cores
  - fast core for sequential code
  - slow cores for parallel sections
- [Hill and Marty, 2008]

$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{(n-r)+perf_{seq}}}$$

  - solve for optimal die area allocation depending on parallelizability f
- But, up to 95% of energy of von Neuman processors is "overhead" [Hameed, et al., ISCA 2010]

A hard look at the performance and ops-per-Joule of today's non-von Neuman alternatives



GPUs, FPGAs and all that

	CPU	GPUs			FPGA	ASIC
	Intel Core i7-960	Nvidia GTX285	Nvidia GTX480	ATI R5870	Xilinx V6-LX760	Std. Cell
Year	2009	2008	2010	2009	2009	2007
Node	45nm	55nm	40nm	40nm	40nm	65nm
Die area	263mm^2	470mm^2	529mm^2	334mm^2	-	-
Clock rate	3.2GHz	1.5GHz	1.4GHz	1.5GHz	0.3GHz	-
Single-prec. floating-point apps						
M-M-Mult	MKL 10.2.3 multithreaded	CUBLAS 2.3	CUBLAS 3.1	CAL++	hand-coded	
FFT	Spiral.net multithreaded	CUFFT 2.3 3.0/3.1	CUFFT 3.0	-	Spiral.net	
Black-Scholes	PARSEC multithreaded	CUDA 2.3	-	-	hand-coded	

In-Core Performance and Energy

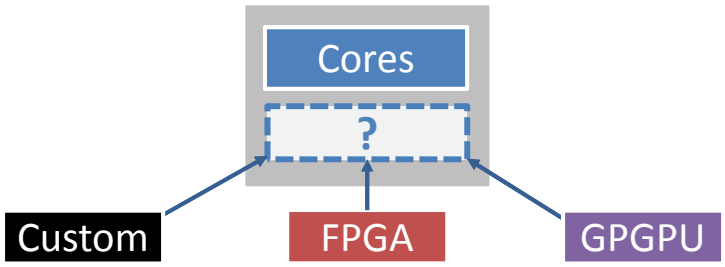
	Device	GFLOP/s actual	(GFLOP/s)/mm² normalized to 40nm	GFLOP/J normalized to 40nm
MMM	Intel Core i7 (45nm)	96	0.50	1.14
	Nvidia GTX285 (55nm)	425	2.40	6.78
	Nvidia GTX480 (40nm)	541	1.28	3.52
	ATI R5870 (40nm)	1491	5.95	9.87
	Xilinx V6-LX760 (40nm)	204	0.53	3.62
	same RTL std cell (65nm)	694	19.28	50.73

- CPU and GPU benchmarking was compute-bound; FPGA and Std Cell effectively compute-bound (no off-chip I/O)
- Power (switching+leakage) measurements isolated the core from the system
- For detail see [Chung, et al. MICRO 2010]

More of the Same

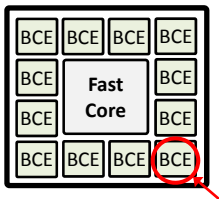
		GFLOP/s	(GFLOP/s)/mm²	GFLOP/J
FFT-2^{10}	Intel Core i7 (45nm)	67	0.35	0.71
	Nvidia GTX285 (55nm)	250	1.41	4.2
	Nvidia GTX480 (40nm)	453	1.08	4.3
	ATI R5870 (40nm)	-	-	-
	Xilinx V6-LX760 (40nm)	380	0.99	6.5
	same RTL std cell (65nm)	952	239	90
		Mopt/s	**(Mopts/s)/mm²**	**Mopts/J**
Black-Scholes	Intel Core i7 (45nm)	487	2.52	4.88
	Nvidia GTX285 (55nm)	10756	60.72	189
	Nvidia GTX480 (40nm)	-	-	-
	ATI R5870 (40nm)	-	-	-
	Xilinx V6-LX760 (40nm)	7800	20.26	138
	same RTL std cell (65nm)	25532	1719	642.5

Onward with a Future of Heterogeneous Multicores



Modeling Heterogeneous Multicores

Asymmetric



Base Core Equivalent

$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{(n-r)}}$$

[Hill and Marty, 2008] simplified

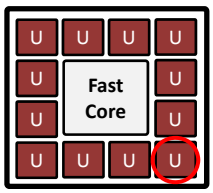
f is fraction parallelizable

n is total die area in BCE units

r is fast core area in BCE units

$perf_{seq}(r)$ is fast core perf. relative to BCE

Heterogeneous



$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{\mu \times (n-r)}}$$

For the sake of analysis, break the area for GPU/FPGA/etc. into units of **U-cores** that are the same size as BCEs. Each U-core type is characterized by a relative performance μ and relative power φ compared to a BCE

φ and μ example values

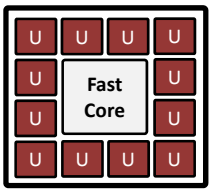
		MMM	Black-Scholes	FFT-2^{10}
Nvidia GTX285	Φ	0.74	0.57	0.63
	μ	3.41	17.0	2.88
Nvidia GTX480	Φ	0.77	[illegible]	0.47
	μ	1.83	[illegible]	[illegible]
ATI R5870	Φ	1.27	[illegible]	[illegible]
	μ	8.47	[illegible]	[illegible]
Xilinx LX760	Φ	0.31	[illegible]	0.29
	μ	0.75	5.68	2.02
Custom Logic	Φ	0.79	4.75	4.96
	μ	27.4	482	489

On equal area basis, 3.41x performance at 0.74x power relative a BCE

Nominal BCE based on an Intel Atom in-order processor, 26mm^2 in a 45nm process

Modeling Power and Bandwidth Budgets

Heterogeneous

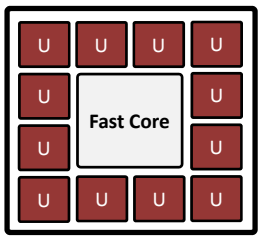


$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{\mu \times (n-r)}}$$

- The above is based on area alone
- Power or bandwidth budget limits the usable die area
  - if P is total power budget expressed as a multiple of a BCE's power, then usable U-core area $n - r \le P/\phi$
  - if B is total memory bandwidth expressed also as a multiple of BCEs, then usable U-core area $n - r \le B/\mu$

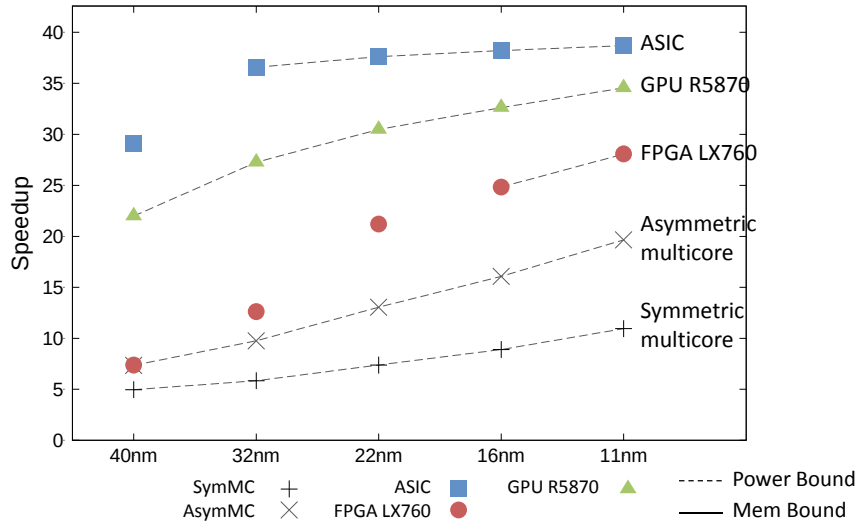
Combine Model with ITRS Trends

Year	2011	2013	2016	2019	2022
Technology	40nm	32nm	22nm	16nm	11nm
Core die budget (mm^2)	432	432	432	432	432
Normalized area (BCE)	19	37	75	149	298 (16x)
Core power (W)	100	100	100	100	100
Bandwidth (GB/s)	180	198	234	234	252 (1.4x)
Rel pwr per device	1X	0.75X	0.5X	0.36X	0.25X

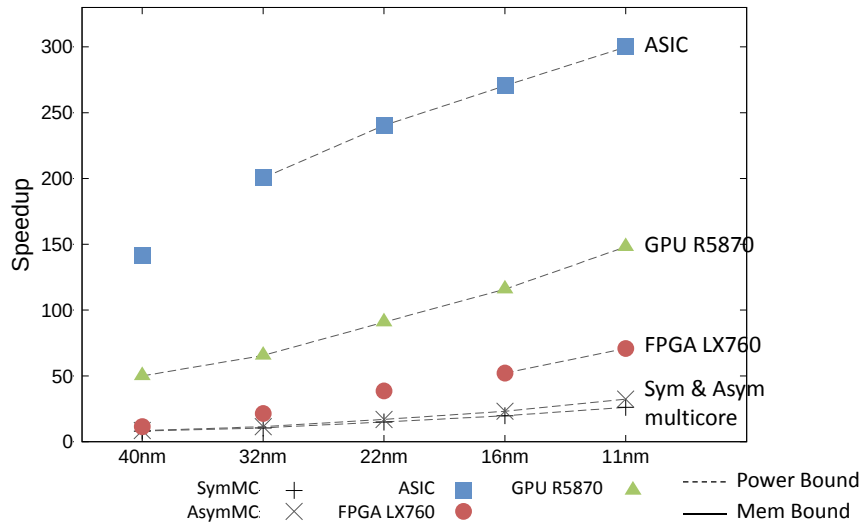


- 2011 parameters reflect high-end systems today; future parameters extrapolated from ITRS 2009
- $432mm^2$ populated by an optimally sized Fast Core and U-cores of choice

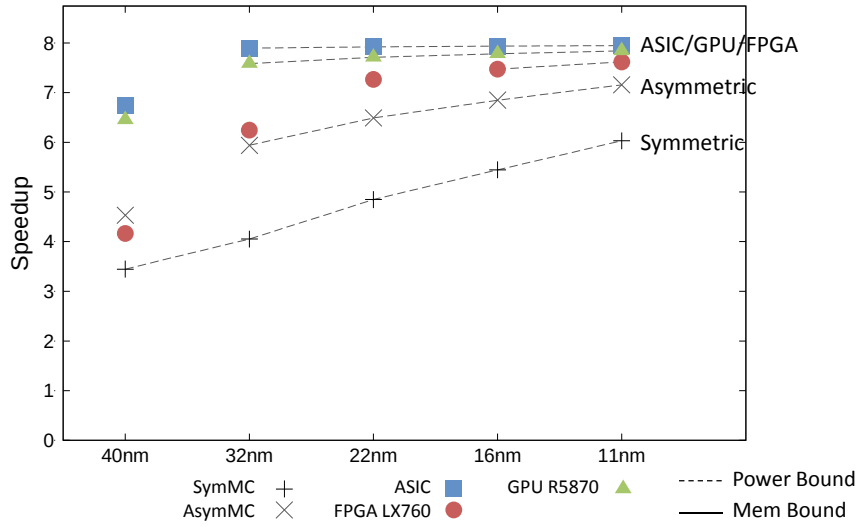
Single-Prec. MMMult (f=90%)



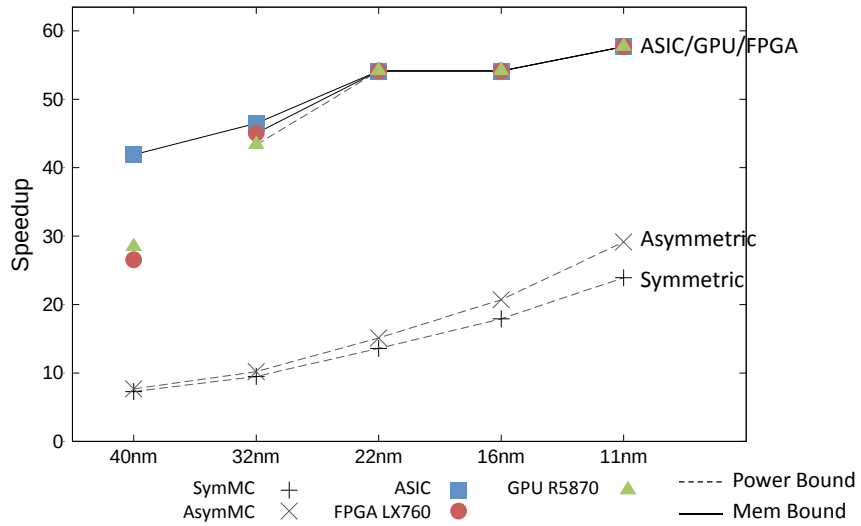
Single-Prec. MMMult (f=99%)



Single-Prec. MMMult (f=50%)

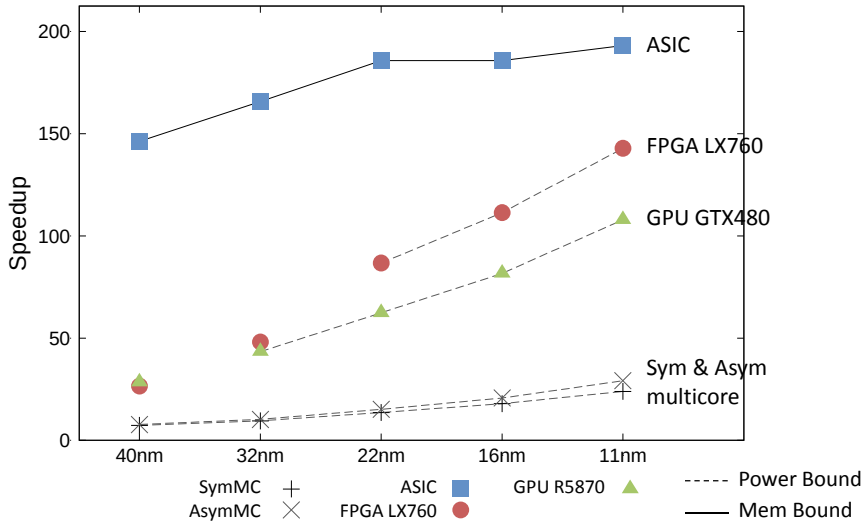


Single-Prec. FFT-1024 (f=99%)



FFT-1024 (f=99%)

if hypothetical 1TB/sec bandwidth



Performance Scaling Summary

- Perf-per-Watt and op-per-Joule matter
  - need to look beyond programmable processors
  - GPUs and FPGAs are viable programmable candidates
- GPU/FPGA/custom logic help performance *only if*
  - significant fraction amenable to acceleration, *and*
  - adequate bandwidth to sustain acceleration

3D-stacked memory could be very helpful!

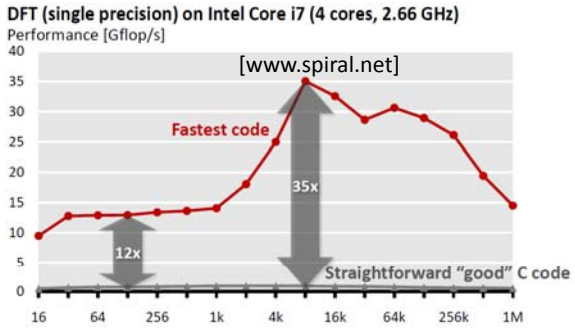
- Without adequate bandwidth, GPU/FPGA catches up with custom logic in achievable performance but remains programmable and flexible
- Custom logic best for maximizing performance under tight energy/power constraints

A quick plug for using FPGAs in computing

check out the CARL 2010 Workshop

Programming FPGA is Hard?

- So is programming CPUs and GPUs at the level of optimization we are talking about; and they are not "portable"

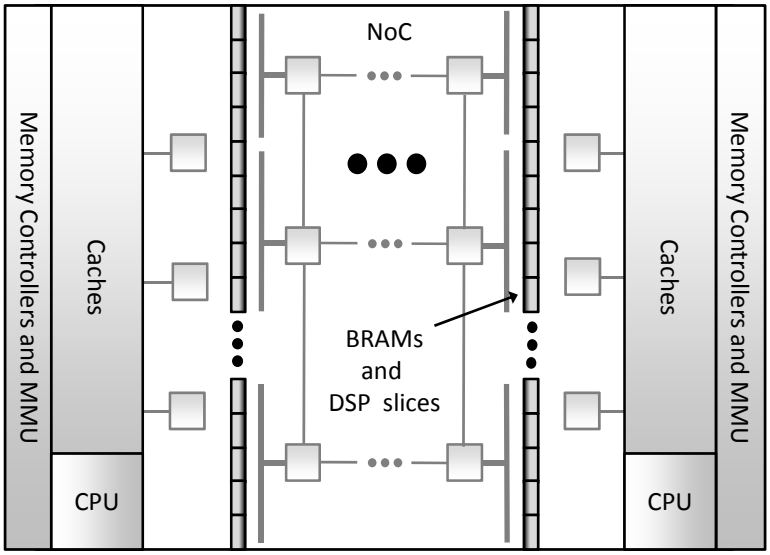


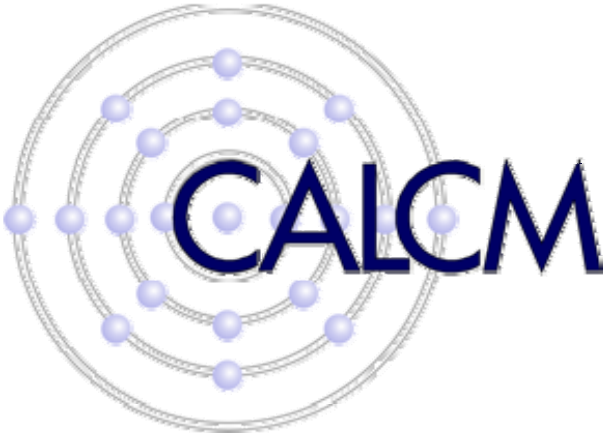
- Whatever the platform, these kernels will be developed by a few and used by many
- Efficiency has to be #1; ease of programming #2

So what is the problem?

- Building HW kernels for FPGA fabric is relatively easy (at least for some of us)
- The real pain is in connecting to external system, in particular off-chip memory
- FPGAs have no native memory architecture
  - soft implementations of memory controller and memory hierarchy leave performance (capacity, bandwidth and latency) on the table
  - why can't I have virtual memory?
- FPGA fabric has no memory abstraction

CoRAM: FPGA Architecture for Computing





Computer Architecture Lab (CALCM)
Carnegie Mellon University
http://www.ece.cmu.edu/CALCM