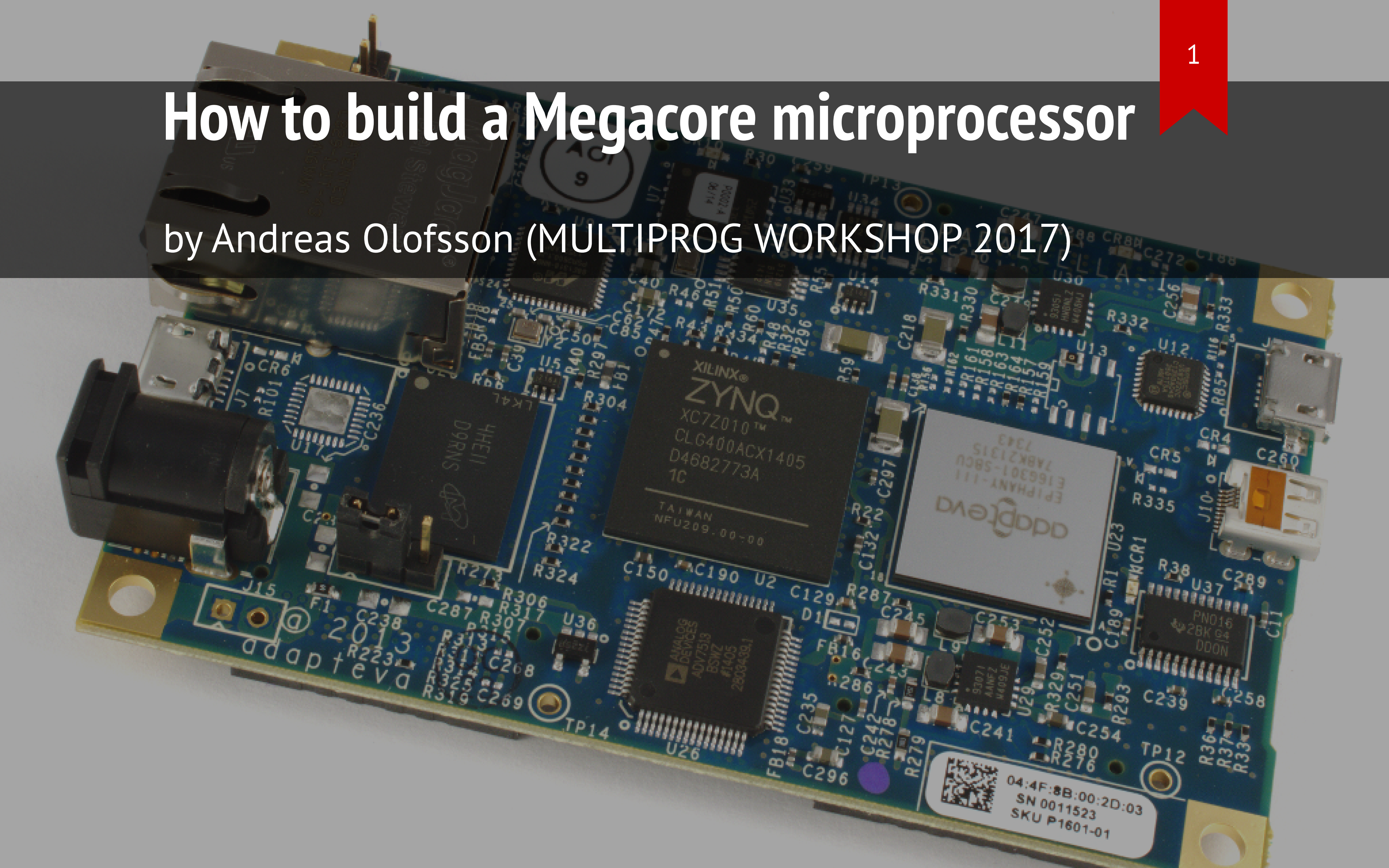
How to build a Megacore microprocessor

by Andreas Olofsson (MULTIPROG WORKSHOP 2017)



Disclaimers

“ *This presentation summarizes work done by Adapteva from 2008-2016. Statements and opinions are my own and do not represent my current employer.*

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressly or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Talk outline

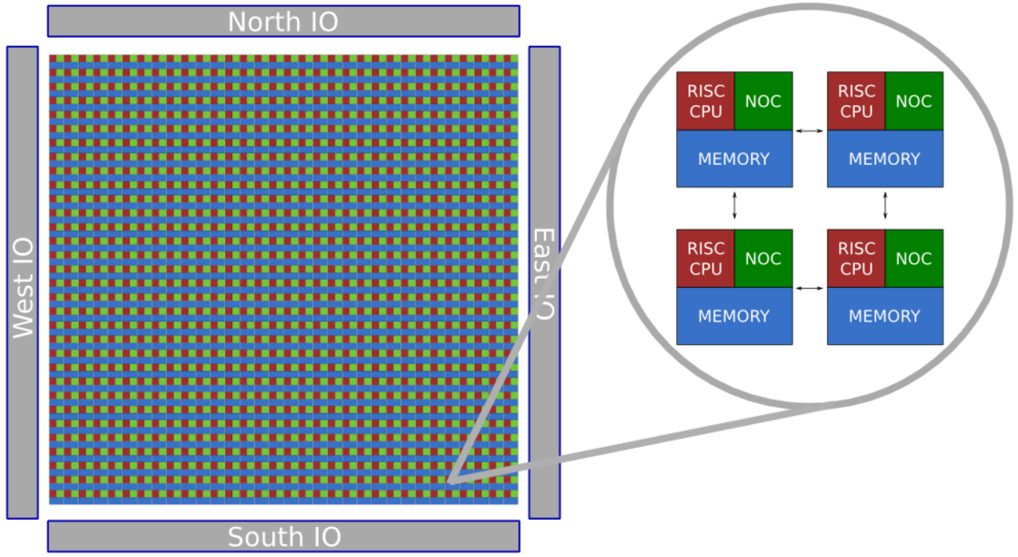
1. Normalize
2. Minimize
3. Maximize
4. Miniaturize
5. Optimize

Challenge Definition

“ *How do we fit one million CPU cores in this package?*

Epiphany-V Chip

- 1024 64bit cores
- 16nm, 117mm^2
- 4.5B transistors
- 64 MB SRAM
- 1024 GPIO signals
- One full-time designer
- Extended ISA for deep learning, comms, crypto
- Dies are back, silicon bringup starting March 2017



Epiphany Intro

- An array of RISC processors
- Shared distributed memory
- Explicit x/y/z memory addressing
- No hardware caching
- Multiple mesh connected NOCs
- Transparent off-chip NOC scaling

Epiphany Programming Models

- Bare Metal
  - C/C++, interrupts, memcpy(), 1 thread/core
- Dataflow/Stream/Message Passing
  - MPI, CAL, BSP, ePython
- Accelerator
  - OpenCL
  - OpenMP
  - OpenSHMEM

Does it Work? Yes!

Program	Cores	Value	Author	Model
OFDM (256)	8	2958 cycles	Ericsson	BOS
128x128 Matmul	16	12 GFLOPS	Ross et al (ARL)	MPI
Sobel	16	8.7 GFLOPS	Ross et al (ARL)	MPI
N-Body	16	8.28 GFLOPS	Ross et al (ARL)	MPI
FFT	16	2.5 GFLOPS	Ross et al (ARL)	MPI

Over 100 more publications at: parallella.org/publications

Area Breakdown

Function	Value (mm^2)	Share of Total Die Area
SRAM	62.4	53.3%
Register File	15.1	12.9%
FPU	11.8	10.1%
NOC	12.1	10.3%
IO Logic / Pads	10.4	8.9%
"Other" Core Stuff	5.77	5.0%

E5 Performance (@500 MHz)

- Compute:
  - FLOAT: 1,024 DP GFLOPS, SPF 2,048 SP GFLOPS
  - FIXED: 4,096 GOPS
- Memory/IO:
  - 16 TB/s local memory bandwidth
  - 1.5 TB/s bisection NOC bandwidth
  - <100ns on-chip communication latency*
  - 512 gbps IO bandwidth (at 250MHz IO clock)

Processor Comparison Table

Chip	Company	Nodes	FLOPS	Area	Trans.	Power	Process
P100	Nvidia	56	4.7T	610	15.3B	250W	16FF+
KNL	Intel	72	3.6T	683	7.1B	245W	14nm
Broadwell	Intel	24	1.3T	456	7.2B	165W	14nm
Epiphany-V	Adapteva	1024	1.0T	117	4.5B	10W	16FF+

Adapteva's budget was $1M. How much did Nvidia/Intel spend?

Compute Density Comparison (DPF)

Chip	GFLOPS/mm^2	GFLOPS/W	W/mm^2
KNL	5.27	14.69	0.35
P100	7.7	18.8	0.40
Broadwell	2.85	7.88	0.36
Epiphany-V	17.55	100	0.17

Preliminary! Kind of amazing that MIMD beats SIMD...

Myths debunked

- "It costs $300M to design a 16nm SOC/ASIC"
- "CPUs are inefficient because (decode, regs, cache etc)"
- "FPGAs are more efficient than CPUs because of ..."
- "GPUs are more efficient than CPUs because of ..."

> *"Optimization principles and application performance evaluation using fairy dust"*

Minimize Tile Size

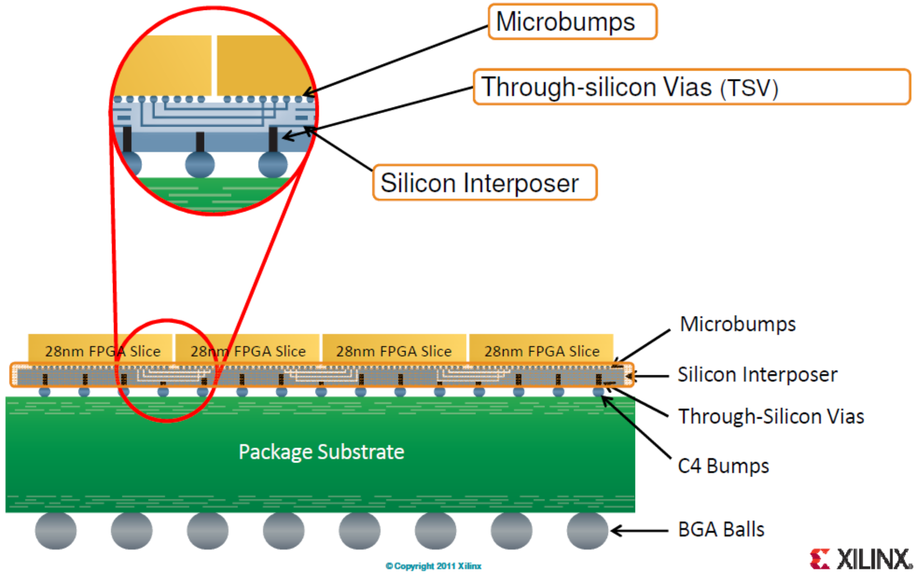
- Parametrized verilog, 4hrs to get through 1st synthesis
- Dual issue 16-bit integer RISC core
- Write only single mesh network
- Remove nice to have stuff
- 0.5KB memory/core, synthesized memories
- Majority of area register file and SRAM
- 10x reduction in area
- Total area is 0.0089 mm^2 / core (<50 red blood cells)

Configuration Example

```
`define CFG_NM        1    // number of meshes
`define CFG_AW        16   // Size of global address space
`define CFG_R         16   // Number of GP registers
`define CFG_MD        128  // Memory depth
`define CFG_MW        16   // Memory width
`define CFG_MB        2    // Number of memory banks
`define CFG_DC        0    // Number of DMA channels (<8)
`define CFG_IW        4    // Number of interrupts (<32)
`define CFG_HL        0    // use hardware loops
`define CFG_IC        0    // use interrupt controller
`define CFG_TI        0    // use timer
```

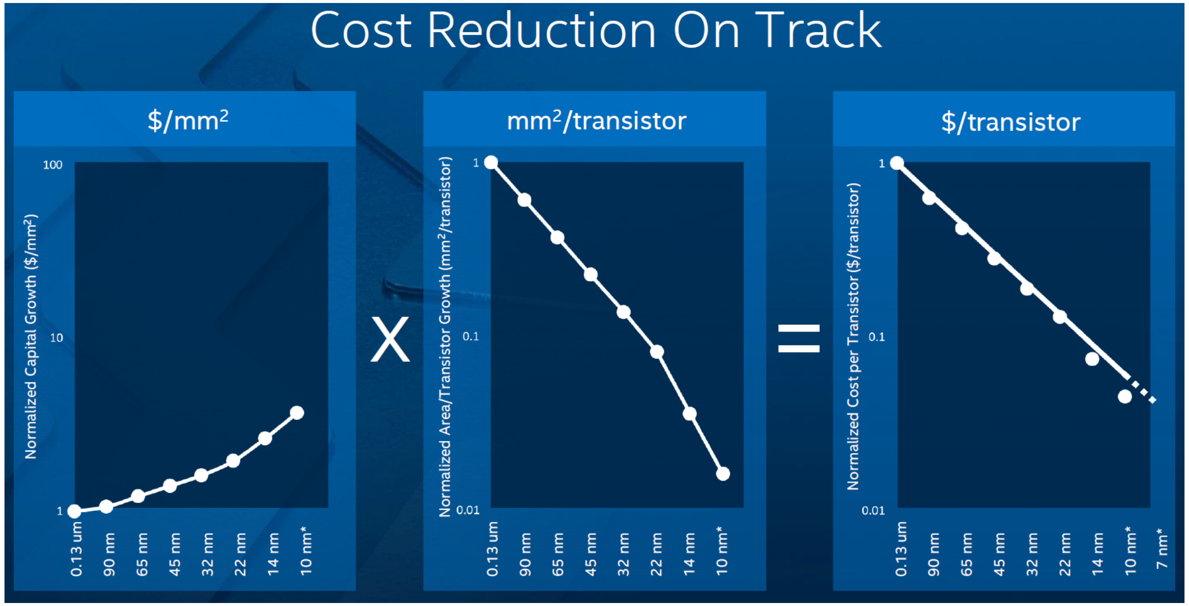
Maximize Chip Size

- Max die (625mm^2)
- Max package (42.5 x 42.5)
- Use 2.5D integration (yes!)
- Wafer scale, 3D (not yet..)



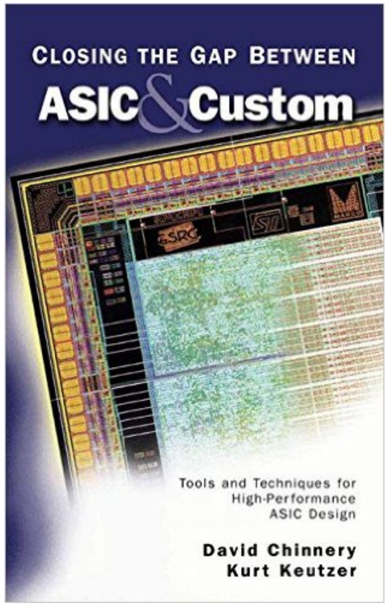
Shrink design to 7nm

- Let's trust Intel and TSMC numbers...
- Assume a 3X boost for 16nm->7nm migration



Custom logic design

- 90% of silicon area in register file/SRAM
- Replace standard cells with custom macros
- Need efficient register file/memory compiler
- Get a one time 2X boost



Million Core Arithmetic

Step	Factor	Cores
Epiphany-V	1	1,024
Minimize	11	11,000
Maximize	16	176,000
Miniaturize	3	528,000
Optimize	2	1,056,000

A million core RISC processor possible using standard practices

So what?

- Who cares?
- What are the applications? (good enough?)
- Need scalable algorithms? (the hard part)
- Need more programmers? (or replace by AI;-))
- Where are the tools? ("oh oh moment", synthesis)