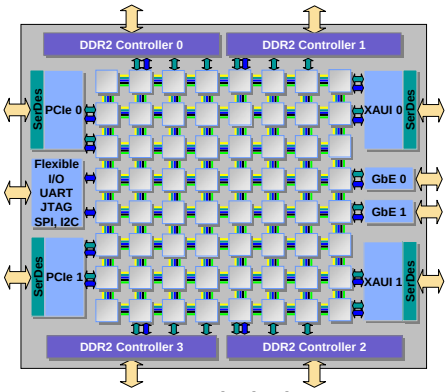
Tile Processor

Case Study of Contemporary Multicore

Fall 2010
Agarwal
6.173

Tile Processor (TILEPro64)

Performance	TILE*Pro*64
# of cores	64
On-chip cache (MB)	5.6
Cache coherency	Yes w/ DDC
Operations (16/32-bit BOPS)	221/166
On chip bandwidth (Terabit/s)	38
Clock speed (MHz)	700, 866
Power	
Typical power -7 device (W)	17-21
I/O and Memory	
Ethernet bandwidth	2 XAUI, 2GbE
PCIe interfaces	2 x 4-lanes
DDR2 bandwidth (peak Gbps)	200



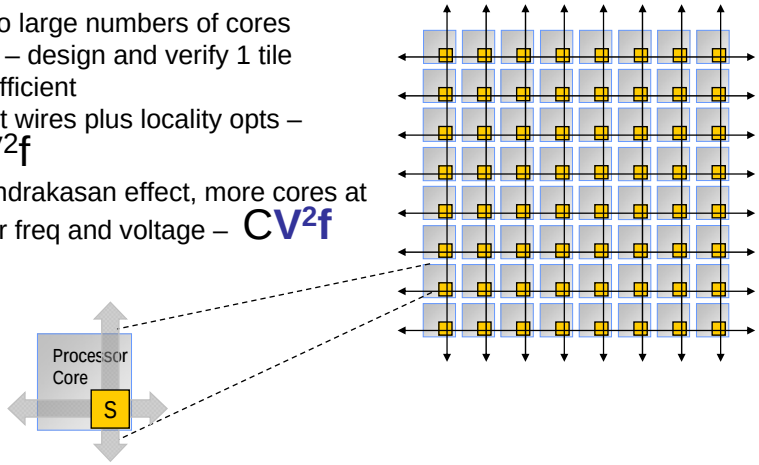
TILE*Pro*64 Block Diagram

TILE64 in 2007
TILEPro64 and TILEPro36 in 2008
TILEGx36 and TILEGx100 in 2011

Tilera systems

Tiled Multicore Concept

- Scales to large numbers of cores
- Modular – design and verify 1 tile
- Power efficient
  - Short wires plus locality opts – CV^2f
  - Chandrakasan effect, more cores at lower freq and voltage – CV^2f

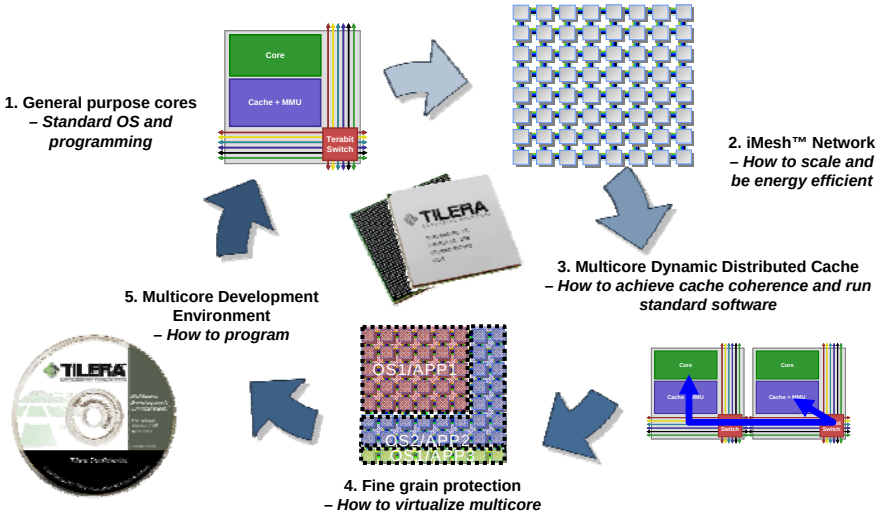


Core + Switch = Tile

Remember the 3 P's?

- Performance challenge
  - How to scale with number of cores – mesh network
  - Distributed scheme for cache coherence
  - User-level network access
- Power efficiency challenge
  - Distributed architecture
  - Mesh network
  - Local caches
- Programming challenge
  - Cache coherence
  - General purpose, full featured cores
  - Fine-grain protection

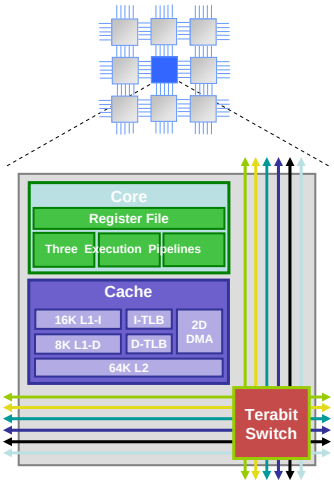
Key Ideas



1 – What's in a Core

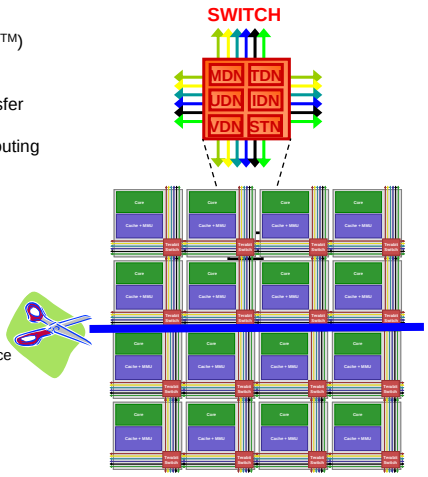
- Processor
  - Each core is a complete computer
  - 3-way VLIW CPU
  - Designed for low power – 200mW per core
  - SIMD instructions: 32, 16, and 8-bit ops
  - Instructions for video (e.g., SAD) and networking
  - Protection and interrupts
  - Single core performance roughly the same as a modern MIPS or ARM core
- Memory
  - L1 cache: 8KB I, 8KB D, 1 cycle latency
  - L2 cache: 64KB unified, 7 cycle latency
  - 32-bit virtual address space per process
  - 64-bit physical address space
  - Instruction and data TLBs
  - Cache integrated 2D DMA engine
- Switch in each tile
  - ISA allows direct processor access to networks

- **Runs SMP Linux**
- **Runs off-the-shelf open-source C/C++, pthreads programs**



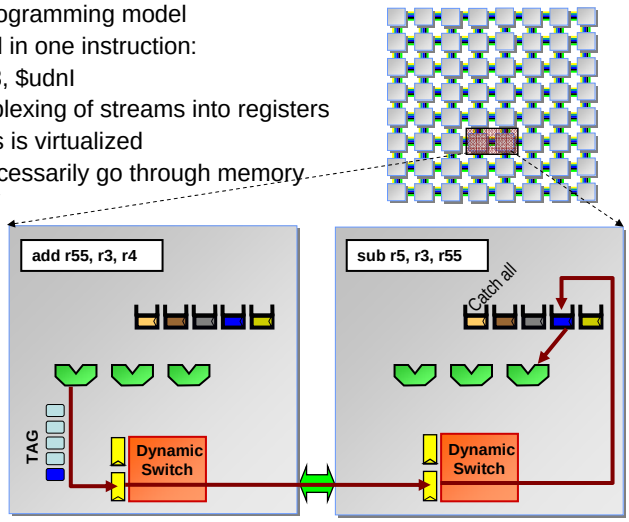
2- On-Chip Networks

- Distributed resources
  - 2D Mesh peer-to-peer tile networks (named iMesh™)
  - 6 independent networks
  - Each with 32-bit channels, full duplex
  - Tile-to-memory, tile-to-tile, and tile-to-IO data transfer
  - Packet switched, wormhole routed, point-to-point
  - Near-neighbour flow control, dimension-ordered routing
- Performance and energy efficiency
  - One cycle hop latency
  - 2 Tbps bisection bandwidth
  - 32 Tbps interconnect bandwidth
  - Low power
- 6 independent networks
  - Five dynamic
    - IDN – System and I/O
    - MDN – Cache misses, DMA, other memory
    - TDN, VDN – Tile to tile memory access and coherence
    - UDN
  - One static, scalar operand network
    - STN – User-level streaming and scalar transfer
- **for scalability and power efficiency**

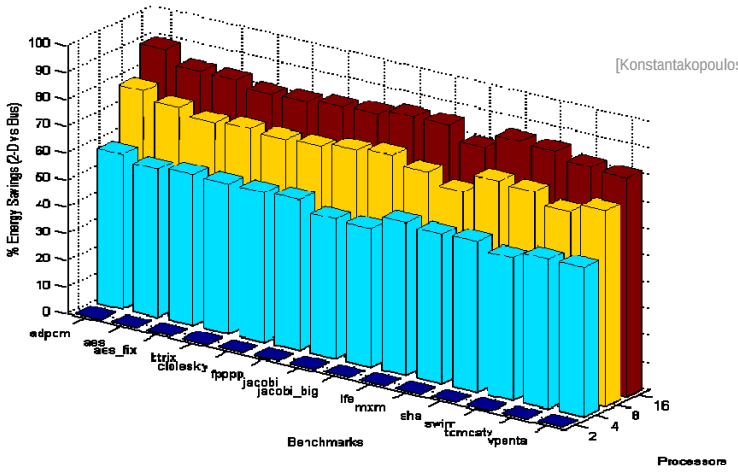


Direct User Access to Interconnect

- Enables stream programming model
- Compute and send in one instruction:
  - add $udnO, $r3, $udnI
- Automatic demultiplexing of streams into registers
- Number of streams is virtualized
- Streams do not necessarily go through memory for power efficiency



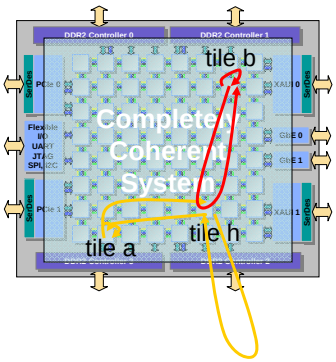
Mesh Power Efficiency



80% power savings over buses

3 – Coherent On-Chip Cache System

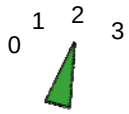
- **Distributed cache**
  - Each tile has local L1 and L2 cache
  - Aggregate of L2 serves as a globally shared L3
- **Dynamic Distributed Cache (DDC™)**
  - Hardware based cache coherence
  - Hardware tracks sharers, invalidates stale copies
  - One or multiple coherency domains
  - Dedicated networks to manage coherency
- **Coherent direct-to-cache I/O**
  - Header/packet delivered directly to tile caches
  - Cache coherent delivery
  - Significant DRAM bandwidth and latency reduction



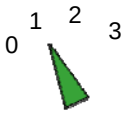
4 – Configurable Fine Grain Protection (CFP)

Full Stack Linux with Hypervisor

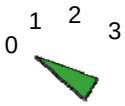
TLB Access



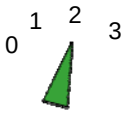
DMA Engine



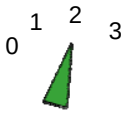
"User" Network



I/O Network



Timers



Key:
0 – User Code
1 – OS
2 – Hypervisor
3 – Hypervisor Debugger

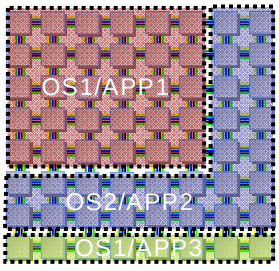
Multicore Hardwall™ Technology for Virtualization and Protection

The virtualization and protection challenge

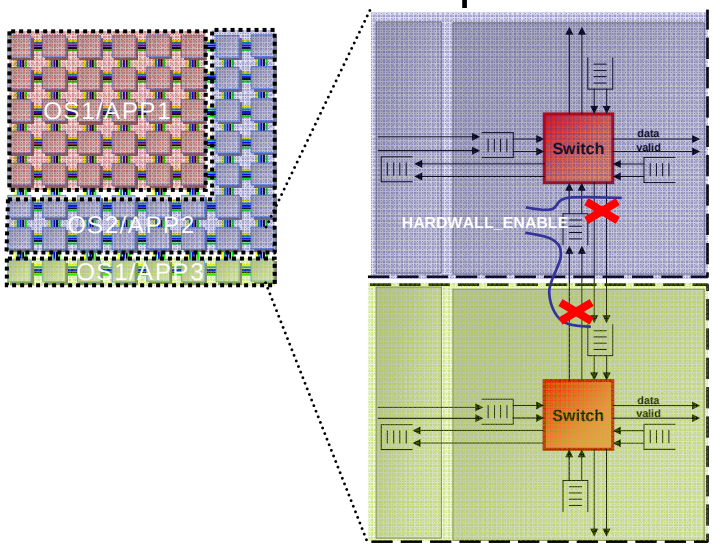
- Multicores need to run multiple OS's and applications in cloud environments
- OS's must be protected from each other
- I/O and other shared resources must be virtualized

Multicore Hardwall technology

- Protects applications and OS by prohibiting unwanted interactions
- Configurable to include one or many tiles in a protected area
- Supported by Tilera hypervisor running on all the tiles



Multicore Hardwall Implementation



5 – Standard Tools and Software

Multicore Development Environment

Standards-based tools

Standard programming
- SMP Linux 2.6.26
- ANSI C/C++
- pthreads

Integrated tools
- SGI compiler
- Standard gdb gprof
- Eclipse IDE

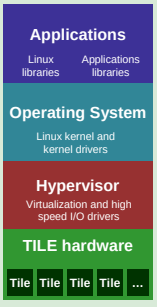
Innovative tools
- Multicore debug
- Multicore profile

Standard application stack

Application layer
- Open source apps
- Standard C/C++ libs

Operating System layer
- 64-way SMP Linux
- Zero Overhead Linux
- Bare metal environment

Hypervisor layer
- Virtualizes hardware
- I/O devices drivers
- Load balancer

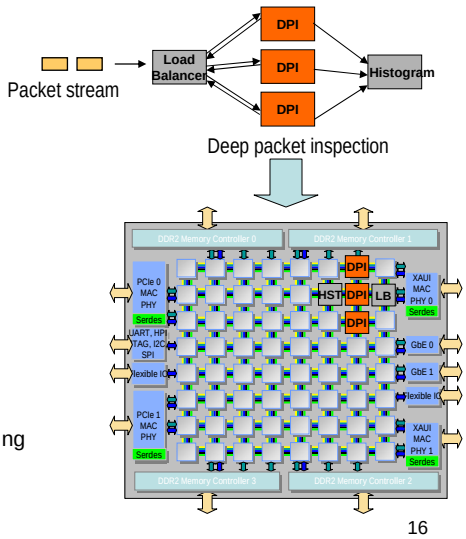


Multiple Software Environments to Meet Diverse Needs of Embedded and Cloud Systems

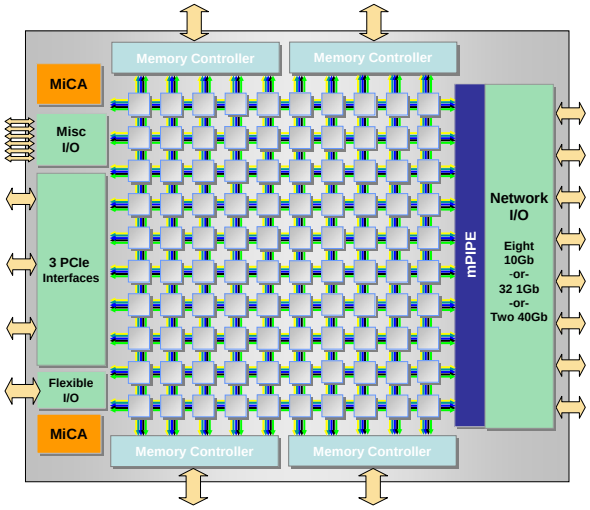
- Standard SMP Linux
  - Standard Linux environment with processes and threads
  - Ideal for applications and control plane code requiring operating system services
  - Open source applications work out of the box
- Standard SMP Linux with Zero Overhead Linux (ZOL)
  - Zero Linux overhead (Eliminates OS interrupts, timer ticks, etc..)
  - Transparent to programmer - no software change required
  - For high performance data-plane applications not requiring OS services
- Bare metal environment
  - Full control of the hardware on up to 64 tiles
  - No operating system or hypervisor layers
  - For embedded applications requiring fine grain control of memory, and IO
- Hybrid environment
  - Using 2 or all three of the above models
  - Each environment can be run on one or more Tiles
  - Ideal for customers aggregating data plane and control plane code on one chip

Parallel Programming using Standard Models

- 64-way SMP Linux
  - Single system image across all tiles
- Standard pthreads API
  - pthread_create()
  - Shared memory model by default
  - Synchronize using mutexes and locks
- Standard Linux processes
  - *fork(), exec()*
  - Separate *address space*
  - Share memory: mmap(), mspaces
  - Communicate: Pipes / local sockets
- Gentle slope programming optimizations using Linux extensions
  - Control memory location and distribution
  - Control thread scheduling and location
- New models and further optimizations using TMC library (Tile Multicore Components)

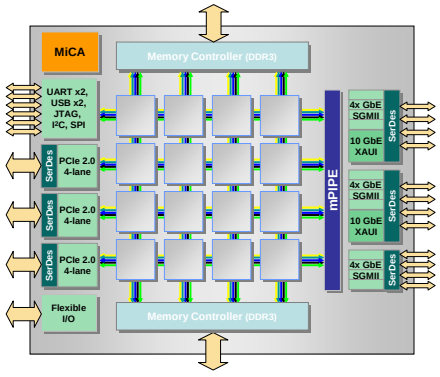


Scaling Up: TileGx100 in 2011



- 100 general-purpose cores
- Runs SMP Linux
- Standard programming
- 1.25GHz – 1.5GHz
- Full 64-bit processors
- 32 MBytes total cache
- 546 Gbps memory BW
- 200 Tbps iMesh BW
- 80-120 Gbps packet I/O
- 80 Gbps PCIe I/O
- Wire-speed packet engine
  - 120Mpps
- MiCA engines:
  - 40 Gbps crypto
  - 20 Gbps compress

Scaling Down: TILE*Gx*16™



- 16 Processor Cores
- 1.0 &1.25 GHz speeds
- Full 64-bit processors
- 5.2 MBytes total cache
- 200 Gbps memory BW
- 20 Tbps iMesh BW
- 24 Gbps total packet I/O
  - 2 ports 10GbE (XAUI)
  - 12 ports 1GbE (SGMII)
- 32 Gbps PCIe I/O
- Wire-speed packet engine
  - 30Mpps
- MiCA engine:
  - 10 Gbps crypto
  - 5 Gbps compress & 5 Gbps decompress
- Midrange 36 core part also announced