Outline

• TILE-Gx100 Processor Overview
• mPIPE Network Interface Architecture
• MiCA Accelerator Offload Architecture
The TILE-Gx100™ Processor:
System-on-a-Chip with 100 64-bit cores

**Tiles**
- 100 64-bit Processor Cores
- 1.0GHz - 1.5GHz
- 32 MBytes total cache
- 200 Tbps iMesh BW

**DDR3 RAM**
- >475 Gb/sec DDR3 BW

**I/O**
- 100 Gbps packet I/O
  - 2 ports 50Gb Interlaken
  - 8 ports 10Gb XAUI / double XAUI
  - 32 ports 1 GbE (SGMII)
- 96 Gbps PCIe I/O
- mPIPE™
  - Wire speed packet engine
  - 120 Mpps throughput

**Acceleration**
- MiCA™ Engines:
  - 40 Gbps crypto (IPsec, AES, small pkt.)
  - 20 Gbps compress & decompress

---

The TILE-Gx100™ Processor: System-on-a-Chip with 100 64-bit cores

Tiles
- 100 64-bit Processor Cores
- 1.0GHz - 1.5GHz
- 32 MBytes total cache
- 200 Tbps iMesh BW

DDR3 RAM
- >475 Gb/sec DDR3 BW

I/O
- 100 Gbps packet I/O
  - 2 ports 50Gb Interlaken
  - 8 ports 10Gb XAUI / double XAUI
  - 32 ports 1 GbE (SGMII)
- 96 Gbps PCIe I/O
- mPIPE™
  - Wire speed packet engine
  - 120 Mpps throughput

Acceleration
- MiCA™ Engines:
  - 40 Gbps crypto (IPsec, AES, small pkt.)
  - 20 Gbps compress & decompress
Coherence and I/O Architecture

• **Globally shared physical address space**
  – 1 TByte addressable DDR3 memory
  – Full Hardware Cache Coherence
  – Standard shared memory programming
  – I/O Reads/writes delivered directly to cache

• **Virtualization and User I/O**
  – TLBs in I/O interfaces
  – Hardware resource partitioning/provisioning
  – Protection between multiple OSes and applications
  – I/O data delivery into application VA space
  – Application SW may choose to send descriptors to on-chip accelerators

• **Accelerators**
  – Provide required performance for complex operations (crypto, compression)
  – Maintain virtualization and partitioning
mPIPE™: Network Interface Engine

**mPIPE™**
(multicore Programmable Intelligent Packet Engine)

- Wire-speed performance
- Programmable operation
- Packet data and descriptors fed directly to user
- Virtualization of hardware resources

---

**mPIPE**

**Packet Parsing & Classification**
- C-programmable engine . . . . . . Over 120 Mpps
- Efficient header parser & protocol de-capsulation

**Load Balancer**
- Flexible modes: i.e. Flow affinity, round robin, load-leveling
- Direct-to-Tile distribution, cache coherent

**Packet Buffer Manager**
- Configurable modes, buffer sizes, queues, notification rings
- Low-overhead buffer release, egress gather DMA
mPIPE Software Interface

- Tiles configure I/O devices via loads and stores to MMIO pages
- mPIPE moves data & descriptors via cache-coherent reads and writes
- Interrupts delivered to Tile
- mPIPE descriptors may be fed to MiCA via application software
- Virtualization through TLBs  
  - Resource allocation  
  - Protection
Ingress Packet Flow

- Packets delivered to mPIPE
- Headers sent to classifier
- C-Programmable classifier:
  - Parses headers
  - Selects flow
  - Assigns Buffer Pool
- Load balancer chooses target OS/application/worker
- DMA engine writes packet data to distributed cache
- Packet descriptor and tail pointer written to ring buffer in worker Tile’s cache
- Worker polls tail pointer or gets interrupt (e.g. Linux NAPI)

Diagram:

- I/O Ports
- MAC Distribution
- Packet Buffer
- Packet Classification
- Buffer Manager
- Load Balancer
- DMA
- Worker Notification
- Packet Data
- Notification & Descriptor Data
C-programmable RISC processor
Generates packet descriptor used for load balancing, buffer management & DMA control
L2-4 header parsing & validation (L3/L4 alignment)
Parallel processors used for classification (linear scaling)
Software view is a single high speed processor
Headers complete out-of-order
  - Reorder queue re-establishes packet order

Packet Classification

Classifier Processor Engines

Header Distribution

Descriptor Reorder
Programmable Classification Engine

- Optimized for parsing
  - 16-bit, single issue, in-order
  - 3-stage fully bypassed pipeline
  - Checksum and hash ops
  - Fast “switch” statements
- Operand source directly from header
- ALU destination directly to descriptor
- Lookup Table
- Dynamically reconfigurable
- Performance for 20 engines exceeds 120 Mpps using full-featured classification program
  - Program allowed at least 260 cycles/packet

From header distribution

256 Header Bytes

Instruction Memory  Processor Core

64 Descriptor Bytes

To reorder queue
Outline

• TILE-Gx100 Processor Overview
• mPIPE Network Interface Architecture
• MiCA Accelerator Offload Architecture
Tradeoffs with Accelerator Architectures

- **Two basic accelerator types:**
  - Off-Core Acceleration (OCA)
    - Cores can block waiting for accelerator
      - Large / small packet queuing
      - High & variable latency
  - In-core acceleration (ICA)
    - Low performance on high b/w flows
    - Heavy core CPU utilization
    - Imperfect tradeoff between silicon area and maximum performance
    - Scaling is bound to core count
    - Poor match for “manycore”

What is the right programming model for hardware acceleration?
• OCA with multi-client support
• Scales to meet system requirement
  – Devices in the TILE-Gx family scale compute and acceleration independently
• Low overhead, high single-stream performance
• Direct access from user level
  – Remote procedure call style programming model
• Provide different types of acceleration, e.g.
  – Encryption/Decryption
  – Compression/Decompression
  – memcpy
  – Others (TBD) in future generations
• Single architecture for multiple design points
MiCA Contexts

- Tiles access services through “Contexts”
  - Service request parameters written via Memory-Mapped I/O
  - Simple and low overhead
  - Prevents blocking between cores
- Protection through TLB
  - Context allocation (provision bandwidth and QoS)
  - Context isolation
- User can initiate request without syscall
MiCA Commands

- Operation parameters include:
  - Source data
    (in virtual memory)
  - Destination buffer
    (in virtual memory)
  - Operation size
    (number of bytes)
  - Operation type
    (e.g. encrypt)
  - Meta-data, as needed
    (e.g. encryption keys, etc.)

- Parameters are written to Context’s registers
MiCA Scheduler

- Scheduler assigns requesting context(s) to accelerator HW
  - MiCA contains multiple types and number of accelerators
  - Scheduler assigns accelerator based on requested service (e.g. compress vs. decompress)
  - Scheduler supports multiple priority levels
MiCA DMA

- DMA accesses cache coherent memory to read source data, writes results to destination buffers
- Data can be directly read from/written to Tile cache
- VA to PA mapping in MiCA match Tile TLB
- Scatter/Gather support
MiCA API

- C API is used to interface to Mica
  
  `gxio_mica_start_op(&cb, src_data_addr, dst_data_addr, extra_data_addr, opcode_oplen);`

- Notification of completion can be done by polling or interrupt

- If waiting for interrupt, SW can:
  - Nap (low power state)
  - Perform other tasks

- Interrupts delivered directly to user level process without OS intervention
Manycore requires new thinking in I/O and accelerator architecture

mPIPE network packet interface allows programmable classification that scales to 100Gbps

Off-Core Accelerator (OCA) architecture adopted for Tile processors

OCA provides high single-stream performance and is scalable
  - 40 Gbps crypto, 20 Gbps compression
Backup Material

- Processor Core Overview
- Software Overview
General Purpose 64-bit Cores

- 64b RISC processor architecture
  - 64b datapath, 64b address architecture
  - 40 PA bits = 1 TByte addressable DDR3 memory

- Powerful general-purpose core
  - DSP & SIMD extensions
  - Specialized instructions for multimedia & networking
  - Atomic operations for synchronization & signaling
  - 8-bit, 16-bit, complex, fixed-point SIMD/DSP modes

- Power/Performance Optimized Core

- 3-issue in-order VLIW
- 5-stage pipeline
- 64-entry register file
- Extensive Clock Gating and low power ‘NAP’ operation
- 5 on-chip networks: memory, tile-to-tile, coherence, user, system
Software and Tools

Multicore Development Environment (Tools)

Standards-based tools

Standard programming
- GNU ANSI C/C++
- Java
- Erlang, TBB
- PHP, Perl, Python

Integrated tools
- GCC V4.4 compiler
- Gdb, gprof
- perf event, oprofile
- Eclipse IDE
- Chip Simulator

Standard application stack

Application layer
- Standard C/C++ libs
- Open JDK

Operating System
- SMP Linux 2.6.36
- Zero Overhead Linux (ZOL)
- Bare metal

Hypervisor layer
- Virtualization (KVM)
- Hardware abstraction
- I/O device drivers