Massively Parallel Architectures

Colin Egan, Jason McGuiness and Michael Hicks
(Compiler Technology and
Computer Architecture Research Group)

University of Hertfordshire

Presentation Structure

- The memory wall/von Neumann bottleneck
- Processing In Memory (PIM)
- Cellular Architectures
- Cyclops/DIMES
- Gilgamesh
- Shamrock
- picoChip
- Questions? (ask as we go along and We'll also leave time for questions at then end of this presentation)

The memory wall

- Today's processors are 10 times faster than processors of only 5 years ago.
- But today's processors do not perform at $^{1}/_{10}^{th}$ of the time of processors from 5 years ago.
- CPU speeds double approximately every eighteen months, while main memory speeds double only about every ten years.
  - This causes a bottle-neck in the memory subsystem, which impacts on overall system performance.
  - The "memory wall" or the "Von Neumann bottleneck".

The memory wall

- In today's processors RAM roughly holds 1% of the data stored on a hard disk drive.
- And hard disk drives supply data nearly a thousand times slower than a processor can use it.
- This means that for many data requests, a processor is idle while it waits waiting for data to be found and transferred.

The memory wall

- IBM’s vice president Mark Dean has said:
  - "*What's needed today is not so much the ability to process each piece of data a great deal, it's the ability to swiftly sort through a huge amount of data.*"

Overcoming the memory wall

- Mark Dean again:
  - "*The key idea is that instead of focusing on processor micro-architecture and structure, as in the past, we optimize the memory system's latency and throughput—how fast we can access, search and move data …*"

- Leads to the concept of Processing In Memory (PIM).

Processing in memory

- The idea of PIM is to overcome the bottleneck between the processor and main memory by combining a processor and memory on a single chip.

Processing in memory

- In a PIM architecture:
  - the CPUs are much closer electrically to the memory arrays containing instructions and data,
  - the number of bits available from each access can be literally orders of magnitude greater than can be transferred in a single clock cycle from today's conventional memory chip to today's conventional (and separate) CPU chip or cache system.

Processing in memory

- The benefits of a PIM architecture are:
  - reduced memory latency,
  - increases memory bandwidth,
  - simplifies the memory hierarchy,
  - provides multi-processor scaling capabilities
    - Cellular architectures,
  - avoids the Von Neumann bottleneck.

Processing in memory

- This means that:
  - much of the expensive memory hierarchy can be dispensed with,
  - CPU cores can be replaced with simpler designs,
  - less power is used by PIM,
  - less silicon space is used by PIM.

Processing in memory

- Reduced latency:
  - In PIM architectures accesses from the processor to memory:
    - do not have to go through multiple memory levels (caches),
    - do not have to travel along a printed circuit line,
    - do not have to be reconverted back down to logic levels,
    - do not have to be re-synchronised with a local clock.
  - Currently there is about an order of magnitude reduction in latency
    - thereby reducing the impact of the memory wall problem.

Processing in memory

- Power:
  - by simplifying the architectural design of PIM leads to a less complex architecture and reduces the amount of silicon space consumed,
  - the greater the architectural design complexity the greater the amount of power consumed and the greater the amount of silicon space consumed,
  - therefore PIM reduces power consumption and reduces the amount of silicon space consumed when compared with a typical complex architecture.

Processing in memory

- But …
  - processor speed is reduced,
  - and the amount of available memory is reduced.
- However, PIM is easily scaled:
  - multiple PIM chips connected together forming a network of PIM cells,
  - such scaled architectures are called Cellular architectures.

Cellular architectures

- Cellular architectures consist of a high number of cells (PIM units):
  - with tens of thousands up to one million processors,
  - each cell (PIM) is small enough to achieve extremely large-scale parallel operations,
  - to minimise communication time between cells, each cell is only connected to its neighbours.

Cellular architectures

- Cellular architectures are fault tolerant:
  - with so many cells, it is inevitable that some processors will fail,
  - cellular architecture simply re-route instructions and data around failed cells .

- Cellular architectures are ranked highly as today's Supercomputers.

Cellular architectures

- Cellular architectures are threaded:
  - each thread unit is independent of all other thread units,
  - each thread unit serves as a single in-order issue processor,
  - each thread unit shares computationally expensive hardware such as floating-point units,
  - there can be a large number of thread units (1,000s if not 100,000s of thousands) – therefore they are massively parallel architectures.

Cellular architectures

- Cellular architectures have irregular memory access:
  - some memory is very close to the thread units and is extremely fast,
  - some is off-chip and slow.
- Cellular architectures, therefore, use caches and have a memory hierarchy.

Cellular architectures

- In Cellular architectures multiple thread units perform memory accesses independently.
- This means that the memory subsystem of Cellular architectures do in fact require some form of memory access model that permits memory accesses to be effectively served.

Cellular architectures

- Uses of Cellular architectures:
  - games machines (simple Cellular architecture),
  - bioinformatics (protein folding),
  - imaging,
    - satellite,
    - medical,
    - etcetera.
  - research,
  - etcetera.

Cellular architectures

- Examples:
  - Bluegene Project
    - Cyclops (IBM)
      - DIMES
  - Gilgamesh (NASA)
  - Shamrock (Notre Dame)

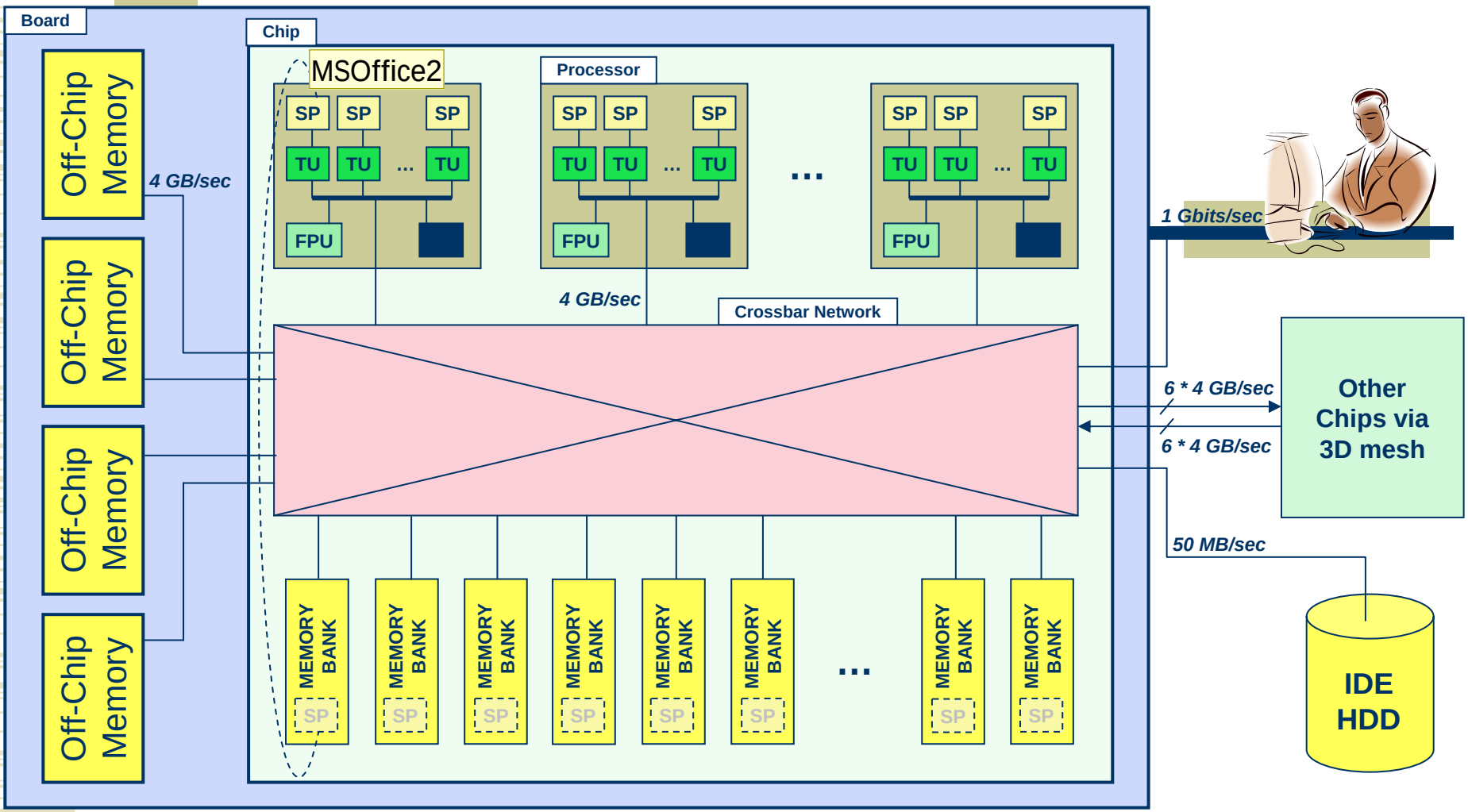
Cyclops

- Developed by IBM at the Tom Watson Research Center.
- Also called Bluegene/C in comparison with the later version of Bluegene/L.

Cyclops

- The idea of Cyclops is to provide around one million processors:
  - Where each processor can perform a billion operations per second,
  - Which means that Cyclops will be capable of one petaflop of computations per second (a thousand trillion calculations per second).

Logical View of the Cyclops64 Chip Architecture



Processors : 80 / chip
Thread Units (TU) : 2 / processor
Floating-Point Unit (FPU) : 1 / processor
Shared Registers (SR) : 0 / processor
Scratch-Pad Memory (SP) : *n* KB / TU

On-chip Memory (SRAM) : 160 x 28KB = 4480 KB
Off-chip Memory (DDR2) : 4 x 256MB = 1GB
Hard Drive (IDE) : 1 x 120GB = 120 GB

SP is located in the memory banks and is accessed directly by the TU and via the crossbar network by the other TUs.

MSOffice2 NOTE ABOUT SCRATCH-PAD MEMORIES:

There is one memory bank(MB) per TU on the chip. One part of the MB is reserved for the SP of the corresponding TU. The size of the SP is customizable.

The TU can access its SP directly without having to use the crossbar network with a latency of 3/2 for ld/st. The other TUs can access the SP of another TU by accessing the corresponding MB through the network with a latency of 22/11 for ld/st. Even TUs from the same processor must use the network to access the SPs of each other.

The memory bank can support only one access per cycle. Therefore if a TU accesses its SP while another TU accesses the same SP through the network, some arbitration will occur and one of the two accesses will be delayed.

, 03/10/2003

DIMES

- Delaware Interactive Multiprocessor Emulation System (DIMES):
  - under development by Prof Guang Gao's research group - the Computer Architecture and Parallel Systems Laboratory (CAPSL) at the University of Delaware, Newark, DE. USA.

DIMES

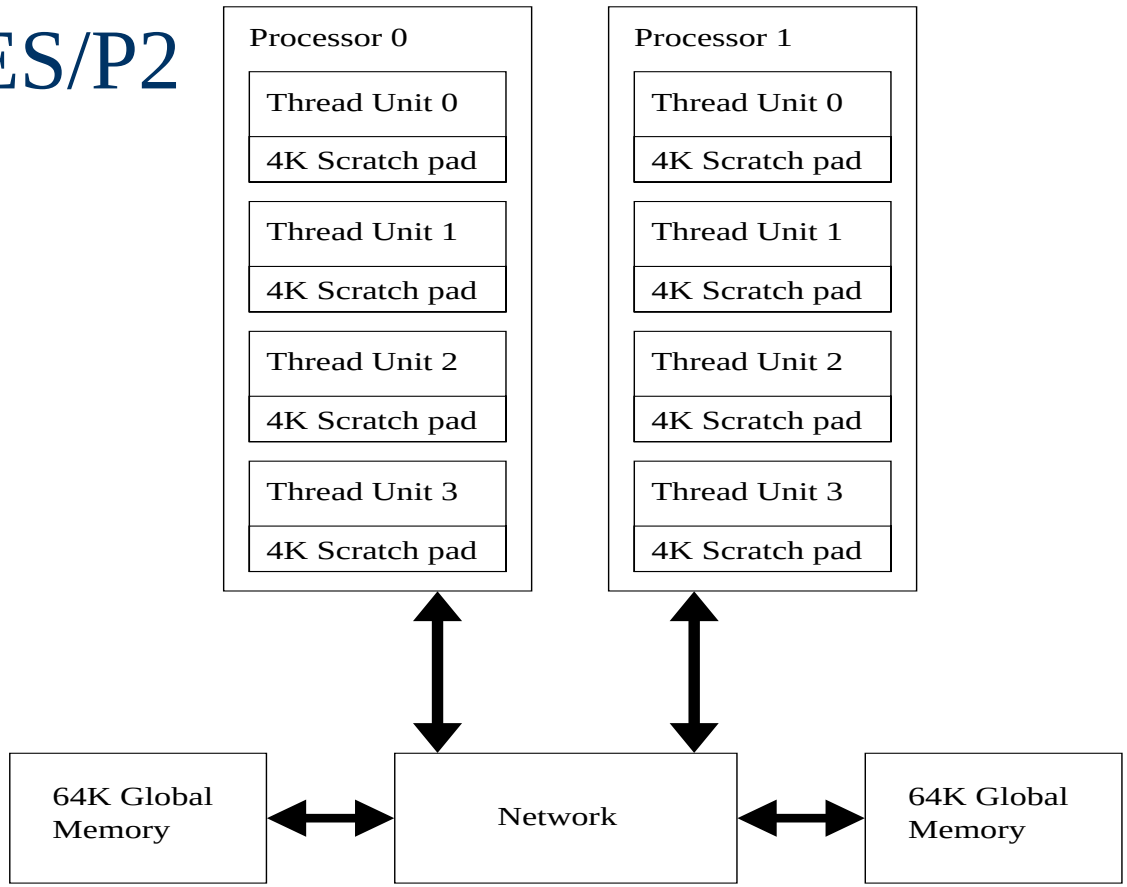
- DIMES:
  - is the first hardware implementation of a Cellular architecture,
  - is a simplified ‘cut-down’ version of Cyclops,
  - is hardware validation tool for Cellular architectures,
  - emulates Cellular architectures, in particular Cyclops, cycle-by-cycle,
  - is implemented on at least one FPGA,
  - has been evaluated by Jason McGuiness.

DIMES

- The DIMES implementation that Jason evaluated:
  - supports a P-thread programming model,
  - is a dual processor where each processor has four thread units,
  - has 4K of scratch-pad (local) memory per thread unit,
  - has two banks of 64K global shared memory,
  - has different memory models:
    - scratch pad memory obeys the program consistency model for all of the eight thread units,
    - global memory obeys the sequential consistency model for all of the eight thread units,
  - is called DIMES/P2.

DIMES

- DIMES/P2



Jason's Work with DIMES

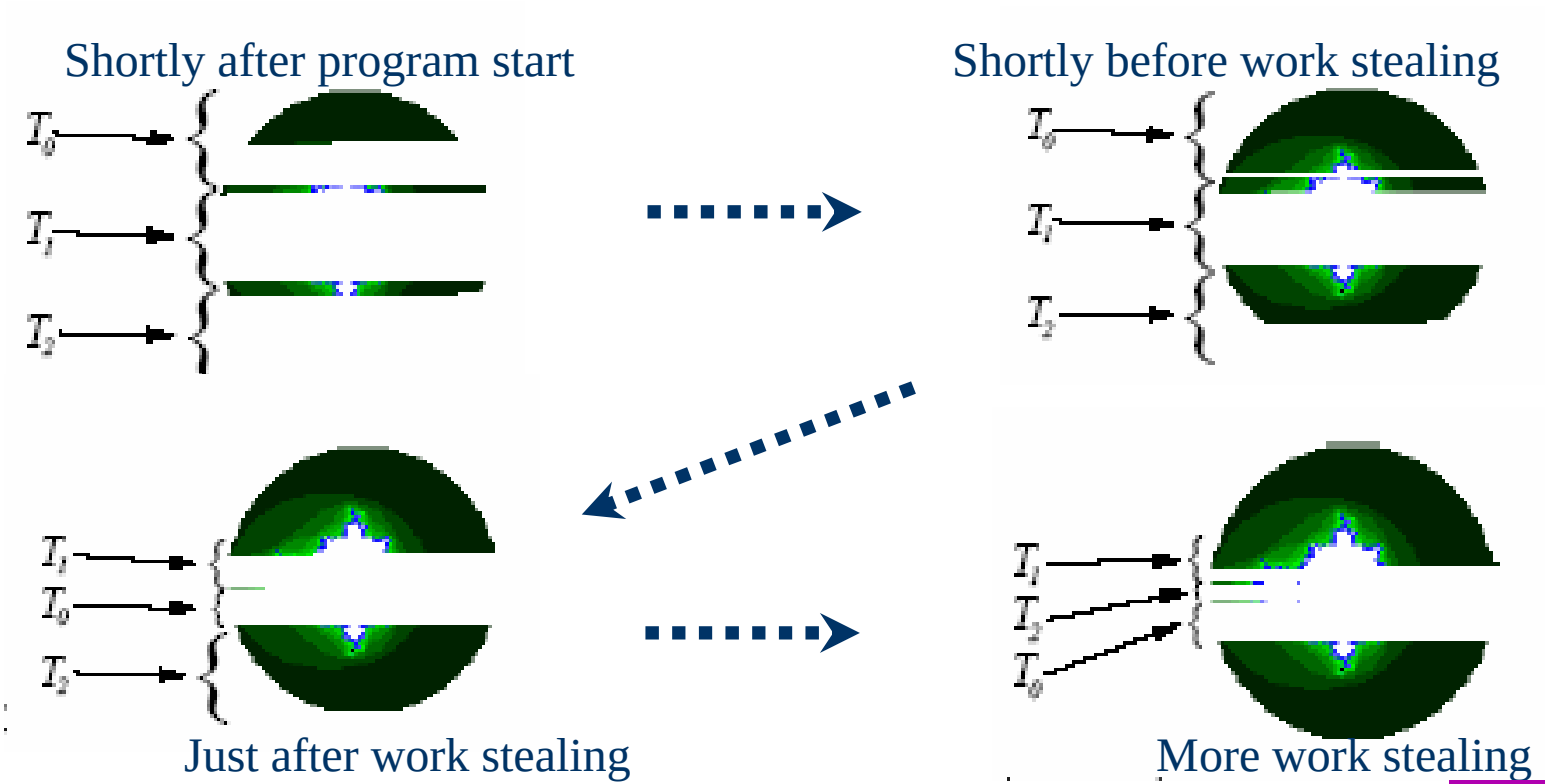
- Jason's concerns were:
  - how to manage a potentially large number of threads?
  - how to exploit parallelism from the input source code in these threads.
  - how to manage memory consistency?

Jason's Work with DIMES

- Jason's tested his concerns by using an "*embarrassingly parallel program which generated Mandelbrot sets.*"

- Jason's approach was to distribute the work-load between threads and he also implemented a work-stealing algorithm to balance loads between threads:
  - when a thread completed its '*work-load*', rather than remain idle that thread would '*steal-work*' from another '*busy*' thread,
  - this meant that he maximised parallelism and improved thread performance and hence overall program execution time.

Jason's Work with DIMES



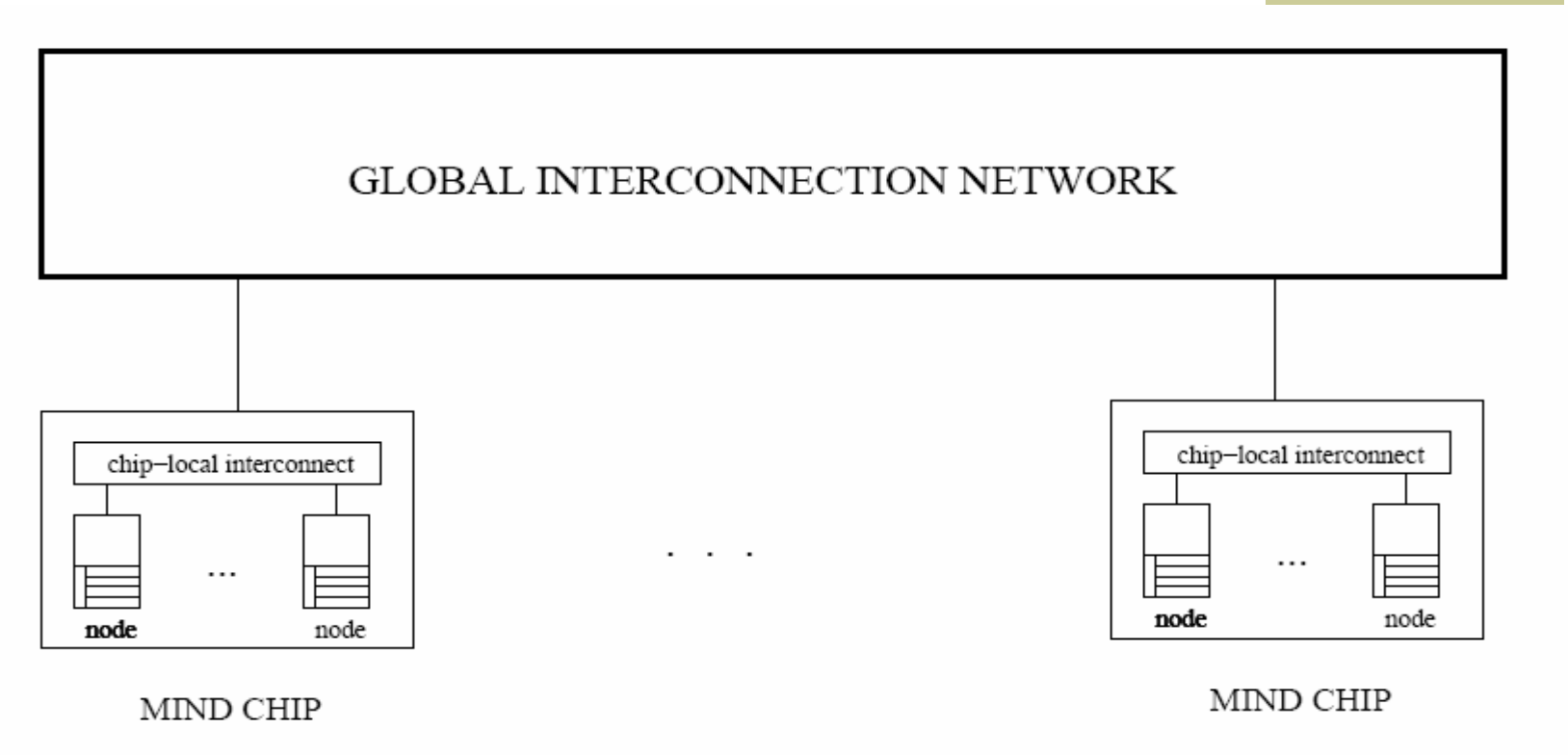
Gilgamesh

- Is being developed by NASA.
- Billions of Logic Gate Assemblies through MESH interconnect.
- Is a PIM-based massively parallel architecture.

Gilgamesh System Architecture

- Is a collection of MIND chips that are interconnected by a system network.
  - MIND: Memory Intelligence Network Device.
- Each MIND chip contains:
  - Multiple banks of D-RAM storage,
  - Processing logic,
  - External I/O interfaces,
  - Inter-chip communication channels.

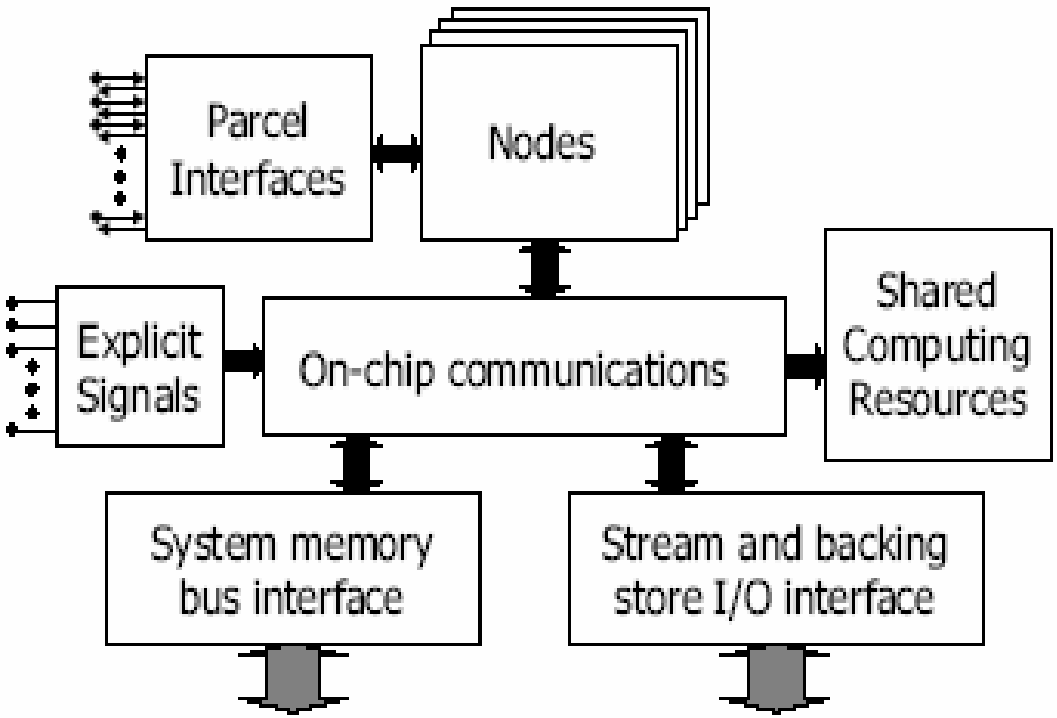
Gilgamesh System Architecture



MIND Chip

- Each MIND chip is multithreaded.
- MIND chips communication:
  - via parcels (special classes of active messages that support message driven computation),
  - a parcel can be used to perform conventional operations as well as invoking methods on a remote chip.

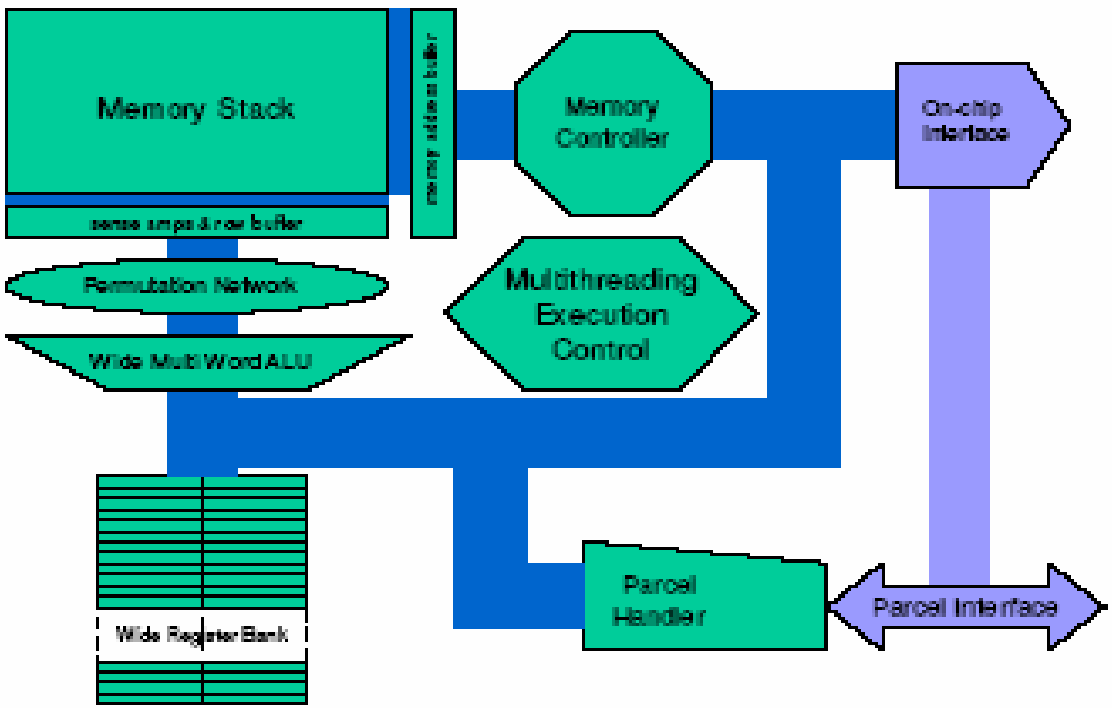
MIND Chip Architecture



MIND Chip Architecture

- Nodes provide storage, information processing and execution control.
- Local wide registers serve as:
  - thread state, instruction-cache, vector memory row buffer.
- A wide ALU performs multiple operations on separate field simultaneously.
- A system memory bus interface connects MIND to workstation and server motherboard.
- A data streaming interface deals with rapid high data bandwidth movement.
- On-chip communication interconnects allow sharing of pipelined FPUs between nodes.

MIND Node Structure



MIND Node Structure

- Integrating memory blocks with processing logic and control mechanisms.
- The wide ALU is 256 bits wide, supporting multiple operations.
- The ALU does not provide explicit floating point operations.
- A Permutation Network rearranges bytes in a complete row.

MIND PIM Architecture

- High degree of memory bandwidth is achieved by:
  - accessing an entire memory row at a time (exploiting data parallelism),
  - partitioning the total memory into multiple independently accessible blocks (multiple memory banks).
- Therefore:
  - memory bandwidth is a square order over an off-chip memory architecture,
  - there is reduced latency.

Parcels

- A variable length of communication packet that contains sufficient information to perform a remote procedure invocation.
- It is targeted to a virtual address which may identify individual variables, blocks of data, structures, objects, or threads, I/O streams.
- Parcel fields mainly contain: destination address, parcel action specifier, parcel argument, continuation, and housekeeping.

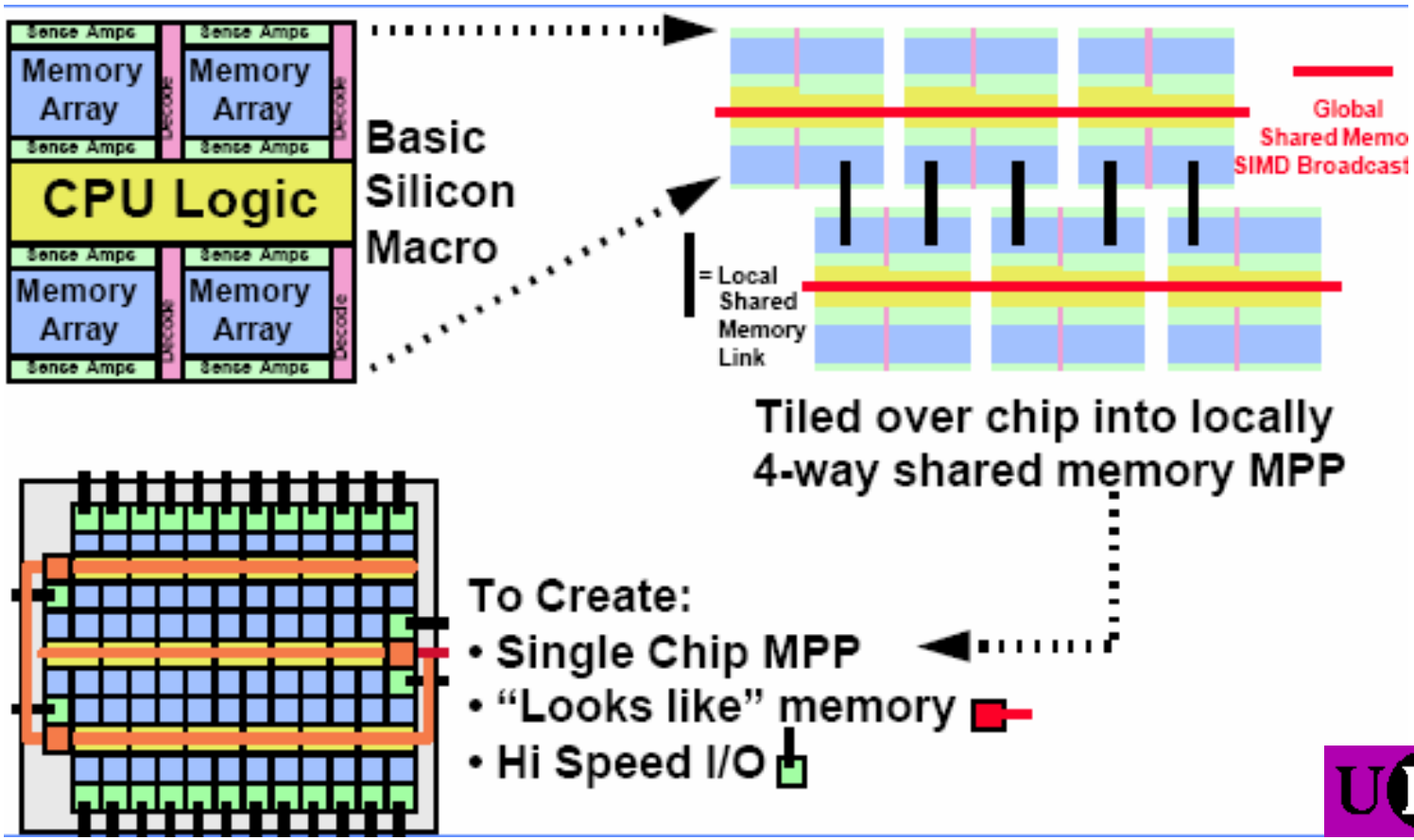
Multithreaded Execution

- Each thread resides in a MIND node wide register.
- The multithread controller determines:
  - when the thread is ready to perform next operation,
  - which resource will be required and allocated.
- Threads can be created, terminated/destroyed, suspended, and stored.

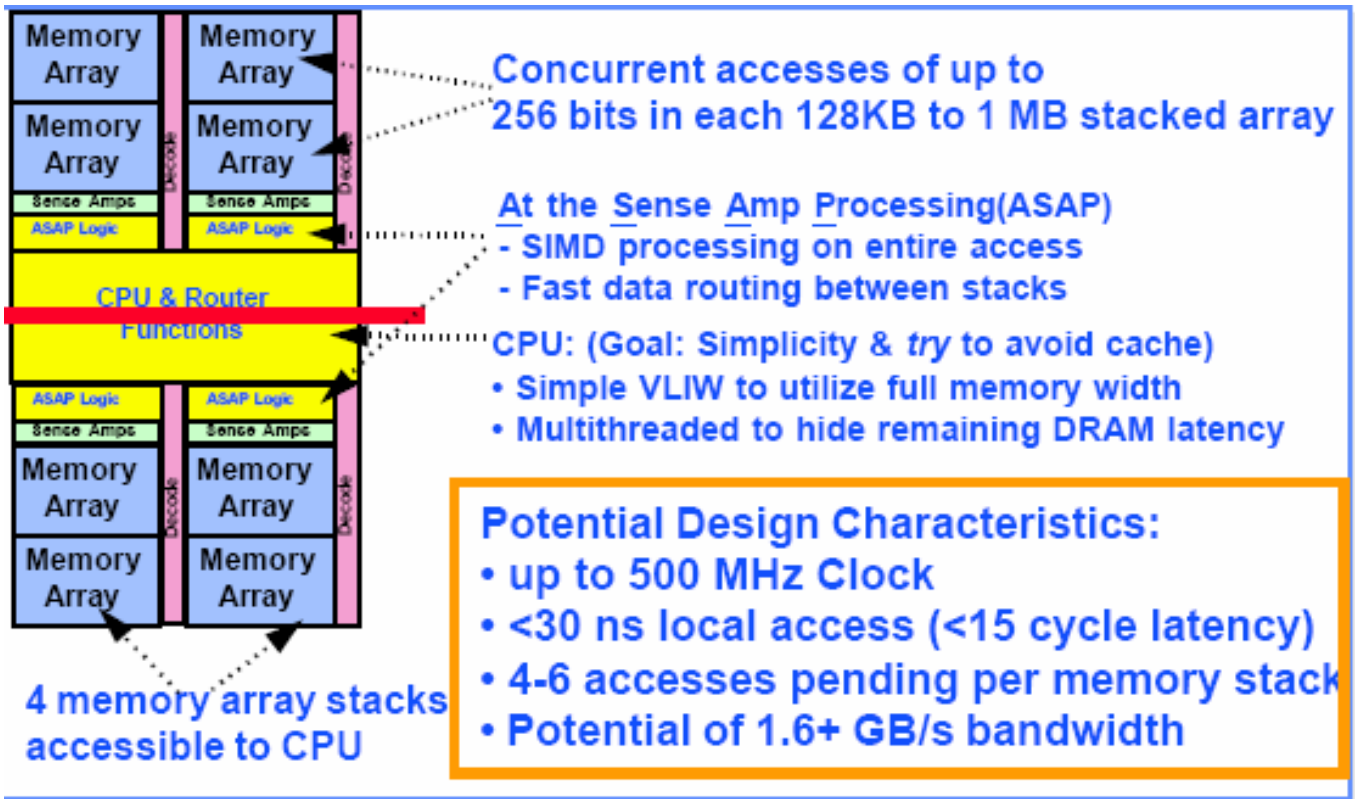
Shamrock Notre Dame

- A PIM architecture developed at the University of Notre Dame.

Shamrock Architecture Overview



Node Structure



Node Structure

- A node consists of logic for a CPU, four arrays of memory and data routing:
  - two separately addressable memories on each face of the CPU.
- Each memory array is made up of multiple sub-units:
  - allows all bits read from a memory in one cycle to be available at the same time to the CPU logic.

Shamrock Chip

- Nodes are arranged in rows, staggering so that the two memory arrays on one side of one node impinge directly on the memory arrays of two different nodes in the next row:
  - each CPU has a true shared memory interface with four other CPUs.

Shamrock Chip Architecture

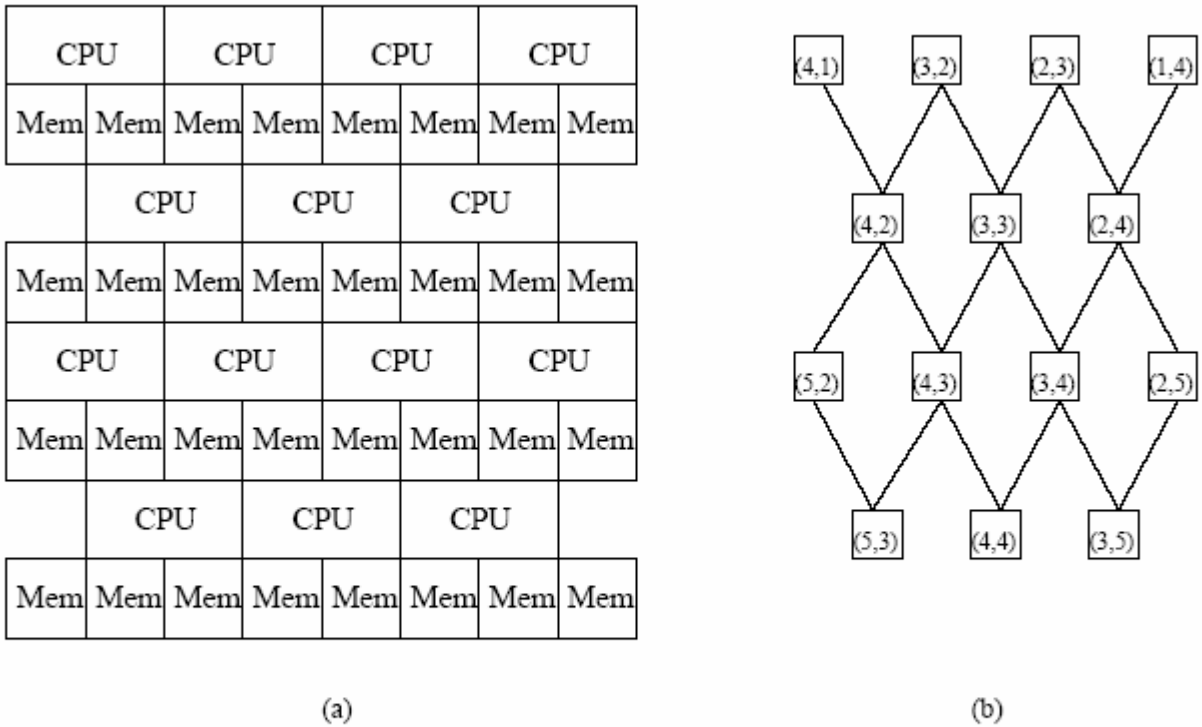


Figure 1: Part of a Shamrock PIM chip (a) and the corresponding "mesh-like" organization of the processing nodes (b)

Shamrock Chip

- Off-chip connections can:
  - connect to adjoining chips in the same tile topology,
  - allow a processor on the outside to view the chip as memory in the conventional sense.

picoChip

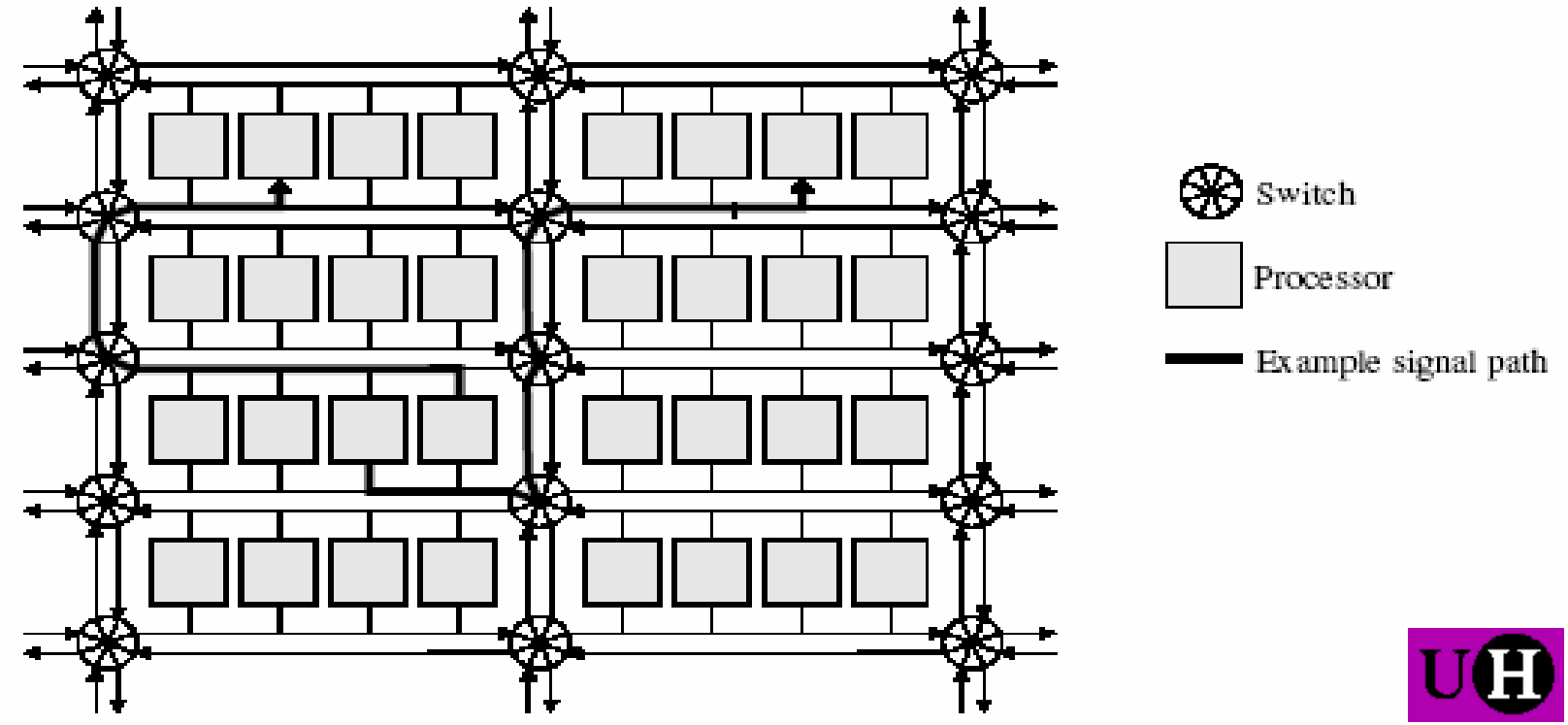
- picoChip are based in Bath
  - "*… is dedicated to providing fast, flexible wireless solutions for next generation telecommunications systems.*"

picoChip

- picoArray™
  - Is a tiled architecture,
  - 308 heterogeneous processors connected together,
  - The interconnects consist of bus switches joined by picoBus™,
  - Each processor is connected to the picoBus™ above and below it.

picoChip

- picoArray™



Summary

- In this talk we have justified the reasons for Processing In Memory (PIM) and therefore cellular architectures.
- We have briefly looked at four example architectures:
  - Cyclops,
  - Gilgamesh,
  - Shamrock,
  - picoChip.
- Jason has worked on DIMES, the first implementation of a (cut-down) version of a cellular architecture.

Massively Parallel Architectures

- Questions?

Colin Egan, Jason McGuiness and Michael Hicks
(Compiler Technology and
Computer Architecture Research Group)

University of Hertfordshire