# Brains for Robots

David May

Bristol University and XMOS

# Robotics

Distributed sensing and control

Thousands of interfaces

Real time

Stream and Event processing

A lot of *natural* parallelism

# 1984 - Inmos

Transputer

Single chip computer

Designed for multiprocessing

Interprocessor communication links

Concurrent programming language

# 2007 - Xmos

Multi-threaded processors

Multicore chips

Real time - deterministic - processing

Interprocessor communication links and switches

Event driven processing

(Another) concurrent programming language

# Today - what next?

We can build chips with hundreds of processors

We can build computers with millions of processors

We can support parallel programming in hardware

We can build digital systems in software

... but our programming is still based on sequential computing

# Robotics and Embedded Intelligence

Architecture and components for scalable, real-time, concurrent processing

Think in terms of *Processes and Communication Patterns*, not *Algorithms and Data Structures*

Scalable interconnect to support *any* communication pattern (eg Clos networks, not 2-D meshes)

We can think of a *universal* processor as an infinite collection of small processors, instead of a small collection of infinite processors

# Parallel Program Patterns

Parallel Random Access Machines (PRAMs/BSP)

Data Parallelism / Process Arrays

Directed Dataflow Graphs

Task Farms and Server Farms

Event handlers

... any combination of the above

# Communication

... is not an “overhead”

Most programs spend a lot of instructions moving data around

The communication pattern is an important part of the processing

“The majority of actual instruction tables will consist almost entirely of the initiation of subsidiary operations and transfers of material”

Alan Turing, 1945

Design the interconnect first; then design the processors to keep it busy

# Communication Patterns

Patterns are often known, especially in embedded processing

With *non-blocking* networks, compilers can often allocate processors and network routes

Communication timing can be deterministic - potential of *real time* concurrent processing

Multiple routes per processor can be implemented using time-division multiplexing

For unknown patterns, use randomisation, hashing, combining ...

# Programming for Robotics

We need to express processes and communication patterns easily and clearly

We want to be able to use different patterns in combination

We may need some new data types - for example to program with probabilities

We need to program events and interactivity - there will be lots of it

We want to be able to optimise concurrent programs

# Processor Architecture

Send and receive messages concurrently as fast as the interconnect can handle them

Eliminate caches - memory operations should take one CPU cycle

Low latency on very short messages; multi-threading for latency hiding

Time-deterministic execution

Appropriate data-types: for example, probabilities

# Summary

Scalable, real-time, *universal* parallel architecture

Supports all patterns with low overhead - including large-memory sequential algorithms

Software topology independent of physical topology

Energy efficient - idle processors can be switched off

Responsive - waiting processors can respond immediately

# Commodity processing components

Ideally, we want to build processing like memory

Choose an economical chip size ($70mm^2$ for DRAM), $100mm^2$ for logic

... this will hold hundreds of processors

Stack the chips up in 3D using *through silicon vias*.

Connect them using silicon photonics

General purpose components with behaviour defined by software

# The Opportunity

Real-time high performance processing for Robotics and Embedded intelligence

Interaction: sensors, actuators and haptics

Vision, Language ...

High-performance control

Systems that *learn* replacing systems that have to be *programmed*

Rapid design of innovative consumer products

# Education

We have an opportunity to build systems of unprecedented capability: we need to educate a generation of *concurrent thinkers*

*Processes and Communication Patterns*, not *Algorithms and Data Structures*

They should learn about Sequence, Concurrency and Event-handling (interaction) *at the same time*

In schools, a good way to do this is through robotics: “I want my robot to dance and sing at the same time”

In Universities, we need to re-think the curriculum: today most systems involve concurrency but most courses don’t