MANY-CORE ARCHITECTURES AND CONCURRENCY IN DISTRIBUTED AND EMBEDDED SYSTEMS

20th April 2012

EPSRC Workshop Report

Christina Turner
ICT Portfolio Manager
Executive Summary

The EPSRC ICT Theme announced the Many-core Architectures and Concurrency in Distributed and Embedded Systems (MACDES) priority in July 2011. The aim of this workshop was to bring together researchers from across the ICT portfolio to identify the challenges which lie underneath this priority and consider what the strategy for its delivery should be over the 2011-2015 Delivery Plan.

The EPSRC organised workshop followed another that was organised by members of the research community. This preceding workshop brought together researchers from across the ICT landscape to hear presentations on work relevant to many-core and parallel architectures. It provided an opportunity for participants to gain an understanding of research beyond their own area and carry this forward into the EPSRC organised workshop (many participants attended both this meeting and the workshop), as well as creating connections between researchers who hadn’t previously interacted to any significant degree.

The EPSRC organised workshop itself began with a session for early career researchers, setting the context of the day into the wider strategic changes at EPSRC. This main content of the workshop was a series of information gathering exercises to identify challenges from each delegate’s own research field, followed by considering problems which held common ground between different fields. The outputs of these sessions were collated to identify several topics and in the final facilitated session the delegates were asked to comment on these highlighting 1) how they would describe this issue to their community 2) dependencies and connections which were important to consider and 3) specific delivery mechanisms which could be needed in order to deliver advances, beyond EPSRC’s standard mechanisms.

Overall, it was thought that the workshop had enabled researchers from across the ICT community to discuss the same broad technical challenge and explore areas of common ground. Collocation of the workshop with the community organised workshop was particularly beneficial in enabling this to occur.

A number of challenges were highlighted during the day but the over-riding observation was that there is a clear need for researchers from across the system stack levels to work together in order to achieve significant results. This is particularly true for system-wide issues such as energy, performance, optimisation and reliability/resilience. Transparency between layers (or exposure of one layer to another) will require researchers from different areas to answer key questions such as ‘what is the appropriate abstraction level for multiple fields to work at?’, and to develop tools to make these cross-stack issues easier to handle.
Contents

Executive Summary ............................................................................................................... i
Contents ................................................................................................................................ii

Background ........................................................................................................................... 1
  Objectives of the workshop ............................................................................................ 1
  Delegate Selection ......................................................................................................... 1
  Agenda ............................................................................................................................ 2

Workshop Outputs ................................................................................................................ 3
  Many-core cross- ICT priority in the context of wider strategic change ....................... 3
  Manycore and Parallel Architectures, Programming Models, and Verification Challenges – Overview of 19th March workshop ........................................................................ 3
  Session 1 - Identifying the challenges ....................................................................... 3
  Session 2 - Common ground ....................................................................................... 4
  Session 3 - Fleshing out the MACDES priority ......................................................... 5
    Programming Abstractions ...................................................................................... 5
    Communication and Data Movement ...................................................................... 6
    Energy Efficient Computation ............................................................................. 6
    Adaptive Run-time Optimisation ........................................................................... 7
    Models, Reasoning and Methodologies ................................................................. 7
    Living with Failure and Exposing Unreliability ...................................................... 8
    Sharing Knowledge and Expertise ....................................................................... 8
    The MACDES Priority and Call ......................................................................... 8

Conclusions ......................................................................................................................... 9

Appendix A – Delegate List ............................................................................................. 10
Appendix B – Outputs from Session 1 and 2 ................................................................ 11
Appendix C – Outputs from Session 3 .......................................................................... 17
  Programming Abstractions ...................................................................................... 17
  Communication and Data Movement ...................................................................... 18
  Energy Efficient Computation ............................................................................. 18
  Adaptive Run-time Optimisation .......................................................................... 20
  Models, Reasoning and Methodologies ................................................................. 21
  Living with Failure and Exposing Unreliability ...................................................... 21
  Sharing Knowledge and Expertise ....................................................................... 22
Background

In 2011 the ICT Theme of EPSRC announced that the broad research challenge of ‘Many-core Architectures and Concurrency in Distributed and Embedded Systems’ would form one of the priorities for the Theme over the 2011-2015 Delivery Plan http://www.epsrc.ac.uk/ourportfolio/themes/ict/introduction/Pages/manycore.aspx. As with the other cross-ICT priorities, it is expected that this priority will draw on, and require contributions from, a number of research areas in the ICT portfolio.

In order to achieve advances aligned to this priority, the ICT Theme needs to engage with the research community. EPSRC wishes to involve researchers in the development of strategy, discuss the direction in which research contributing to this priority should take, and explore the connections between research communities which need to be formed or strengthened.

The workshop outcomes will feed into the ICT Theme strategy for this priority, at least over the current Delivery Plan period. In particular, the outputs are expected to influence a call, due to be announced in early June 2012, which will contribute to this particular cross-ICT priority being addressed.

Objectives of the workshop

- To explore common interest areas between the ICT research communities which contribute to this cross-ICT priority;
- To help inform EPSRC of the direction this priority should take over the Delivery Plan;
- To discuss what actions can be taken by EPSRC and/or the community to achieve advances in this priority;
- To encourage the formation of partnerships between members of these communities which lead to tangible research work and outputs that align to the cross-ICT priority MACDES: Many-core Architectures and Concurrency in Distributed and Embedded Systems.

Delegate Selection

Following an open call for expressions of interest, issued on the EPSRC website in December 2011, we received around 80 applications to attend the workshop. We were able to invite 42 delegates to the event, held on the 20th March 2012 in Birmingham. Selection was based on the application form which asked for a brief biography, information about research expertise and a statement detailing their interest in the event. The ability to think beyond a particular research expertise was an important criterion; in addition to this, delegates were chosen to cover a range of research areas, career stages and institutions.

The ICT Theme particularly wanted to encourage involvement of early career researchers in the development of the priority’s strategy. Of the 42 delegates, 16 were early career researchers.
Agenda

The workshop was held on the 20th March 2012 at the Hilton Metropole, Birmingham

0900   Arrival and registration for early career researchers

0930   Many-core cross- ICT priority in the context of wider strategic change  
        Dr Liam Blackwell, ICT Theme Leader

1000   Arrival and registration

1030   Welcome  
        Dr Liam Blackwell, ICT Theme Leader

1040   Manycore and Parallel Architectures, Programming Models, and Verification  
        Challenges – Overview of 19th March workshop  
        Dr George Constantinides, Imperial College London

1100   Identifying the challenges

1215   Lunch

1315   Common ground

1430   Break

1500   Fleshing out the MACDES priority

1545   The MACDES priority and call

1615   Close
Workshop Outputs

**Many-core cross- ICT priority in the context of wider strategic change**

The EPSRC ICT Theme wishes to encourage Early Career Researchers (ECRs) to contribute to strategy development. Engagement with the community over the last year has led the ICT Theme to believe that researchers at this career stage are less aware of the changes at EPSRC, and the reasons for them. There is also a perception that a number of ECRs believe EPSRC only wishes to engage and seek advice from established researchers.

In order to partially address these issues this workshop included a session tailored for ECRs. This introductory talk was given by Dr Liam Blackwell (ICT Theme Leader) before the main workshop began. The presentation provided an overview of the recent changes at EPSRC and in particular the approach taken by the ICT Theme. He explained the context in which the MACDES cross-ICT priority sits and how researchers could contribute during the workshop to help inform EPSRC policy.

**Manycore and Parallel Architectures, Programming Models, and Verification Challenges – Overview of 19th March workshop**

Dr George Constantinides (Imperial College London) gave an overview of the workshop held on the previous day, titled *Manycore and Parallel Architectures, Programming Models, and Verification Challenges*, and co-located with the EPSRC workshop.

This workshop consisted of ten invited talks with speakers drawn from computer architecture, electronics design and a range of computer science. Three of the speakers were industrialists; Ian Phillips from ARM, Steve Perry from Altera and Theo Drane from Imagination Technologies.

It was observed that the hardware-orientated speakers were very focussed on the energy/performance trade-off while software-orientated speakers were focussed on threads, concurrency and proof. One of the key points from this day was the need for the software community to understand the concerns of the hardware community and visa versa; with the community working together at appropriately defined levels of abstraction.

Details of the speakers and some of the presentations are available at [http://www.parallel-challenges.net/](http://www.parallel-challenges.net/)

**Session 1 - Identifying the challenges**

To begin developing a strategy for the MACDES priority over the Delivery Plan we first need to consider what the key problems are that the ICT community faces and how it can contribute to the advancement of technology in this area. Delegates were asked to comment on what technical challenges existed, or would do in the future, from their own area of expertise.

Trade-offs were a common theme in this session, in particular, the cost benefit to be considered when thinking about energy, performance and reliability. Other trade-offs were
highlighted such as adaptive/evolvable systems which have defined bounds but can cope when going beyond them (resilience/fault tolerance), optimisation vs. safety, computation vs. communication, and dynamic vs. static analysis. Other key issues were those associated with heterogeneous architectures, whether memory should be shared or not, how non-deterministic systems can be programmed, how programmability can be improved, and what abstractions levels should/could the community work at when working together.

Session 2 - Common ground

Taking forward the outputs from the previous session, the next objective was to expose the areas of cross-over in the ICT portfolio relating to the MACDES priority. Delegates moved positions and were asked to consider the information now available to them. The delegates were asked whether the same or similar problems existed in their area of expertise and how their area was helping to tackle it (or could in the future). Delegates also identified existing tools, techniques or methodologies in their area which could contribute to solutions. Finally, the connections between fields were explored in terms of which needed to be strengthened or established to enable all the required experts to contribute to finding solutions for the problems identified.

In order to bridge the areas of expertise and different terminologies during this session Dr George Constantinides (Imperial College London), Prof Peter O’Hearn (University College London), Prof Steve Furber (The University of Manchester) and Prof Roger Woods (Queen’s University Belfast) acted as ‘Translators’. They also assisted the EPSRC ICT Theme in collating the gathered information for the final facilitated session.

The following comments in this section aim to provide an overview of the final outputs from sessions 1 and 2. Full details can be found in Appendix B.

The session exposed the need for many of the issues facing the community to be considered at all levels of the system. It was recognised that considerations such as resilience, reliability, energy and performance need to be optimised across the stack but this requires exposure of one layer to another or transparency between them, consideration of the appropriate abstraction levels and tools to enable this. Taking energy as an example, delegates considered how hardware can expose energy to the software and how energy could be treated like a resource - to be reasoned about, formalised and expressed within programming languages. At even higher levels in the stack, delegates questioned whether energy could be managed through operating system policies or through the operating system working directly with the hardware to adjust the number of active cores dynamically; in a holistic power management system.

Continuing to take a whole system view, it was commented upon that specialist systems with Domain Specific Languages (DSLs), specialist software and hardware are more efficient than general-purpose systems but it was questioned how we could take advantage of this while accounting for the increased complexity associated with them.

Programming of many-core systems was also seen as challenging, with new tools (and possibly new languages) considered necessary. One particular difficulty expressed by delegates is that programming occurs at several layers in the stack and the needs of the programmer at each are different. It was thought that there is a need for programming
languages and models which have ease of expression, some going as far as to suggest that text instruction should be replaced entirely by graphical representations. However, it was important to also consider how legacy and code refactoring would be taken into account given the reluctance of users, and the costs involved, to ‘start from scratch’.

Session 3 - Fleshing out the MACDES priority

Following discussions with the four Translators, seven topics were identified from sessions 1 and 2. These topics were:

- Programming Abstractions
- Communication and Data Movement
- Energy Efficient Computation
- Adaptive Run-time Optimisation
- Models, Reasoning and Methodologies
- Living with Failure and Exposing Unreliability
- Sharing Knowledge and Expertise

In order for the ICT Theme to form a strategy for the MACDES priority it is important to consider any dependencies which exist, the people who need to be involved and the best delivery mechanisms. In addition to these, delegates were asked to comment on how they would describe the topics to the ICT community.

The following subsections provide a summary of the delegate’s comments posed as questions for the community to consider. A full list of the outputs from this session can be found in Appendix C.

Programming Abstractions

- Abstraction Levels
  - How can we marry the need for various levels of abstractions with a need to bridge these levels (particularly with issues such as energy and performance)?
  - Researchers work at different levels of abstraction. Can we provide some integration across stack levels while still allowing transparency when needed?
- Programming parallel and/or distributed systems?
  - Graph representation rather than text instruction?
  - Purely functional languages?
  - Representing problem rather than implementation spaces?
  - Can programmes be generated from Domain Specific Languages?
  - How do you program non-deterministic systems?
  - How could energy efficiency be supported within programming?
  - How do you represent communications in a computation model?
    - How could reliability, accuracy or quality of service be represented to support computation?
- Legacy
  - How can existing technology be taken into account?
  - How do we deal with legacy?
  - How would refactoring be enabled?
**Communication and Data Movement**

- Communication vs. Computation
  - What are the required trade-offs between data movement and local computation and the affect of this?
  - Can locality be managed implicitly rather than explicitly?
  - How can this be modelled?
    - How would we deal with incomplete data?
- Integration
  - How would this be mapped or compiled into hardware?
  - Can programming languages explicitly capture data movement?
    - Can data movement code be auto-synthesised?
  - How can this theme be considered part of the architecture while meeting functional/non-functional constraints?
  - Can software include clear indications of communication primitives and their link to performance?
  - How can data movement be determined as correct?
- Dynamic systems
  - Should traffic management be dynamic or static, or a mixture? What would be the trade-off implications?
  - How can deadlock, heat, hotspots for example be dealt with?
- Scalability and Energy Efficiency
  - How would you ensure approaches are scalable and/or energy efficient?
  - Could communication channels be pruned to manage resource effectively?

**Energy Efficient Computation**

- The software/hardware gap
  - Can energy be managed from a high abstraction level?
  - How could an operating system support energy efficiency policies?
  - Why does software keep hardware busy?
  - How can energy be a transparent parameter over the stack?
- Tools
  - How can we model energy?
  - How can energy be exposed to programmers?
  - How do you reason about energy?
  - Could you have language representations?
  - How can we estimate energy costs and/or characterise future technology?
  - How can people better understand the affect of architecture choices?
  - How do you design energy efficient algorithms or arithmetic systems?
- Complexity
  - Specialist systems are more energy efficient than general purpose. What can we learn from this without increasing complexity?
  - Can/should computational complexity be re-evaluated with respect to energy efficiency?
  - What can we learn from biology particularly regarding slow biological computation and scalability?
- Can energy efficiency be delivered through adaptive run-time optimisation?
Adaptive Run-time Optimisation

- What would an optimised state look like?
  - How can this be defined?
  - What would be the bounds?
  - What would be the appropriate trade-offs between energy, performance, reliability and cost? Could these change dynamically?
  - How can non-functional properties be integrated into an optimised state too?

- Tools
  - What do we need to know and what can be known as prior offline knowledge?
  - Performance models
  - Statistical models
  - Static vs. dynamic analysis
    - When can static be used?
  - How can you model the problem?

- Across the stack
  - What information will need to pass across layers?
  - How can the whole stack integrate/co-operate to give overall optimisation
  - How can users be exposed to other layers in a transparent, effective way?
  - Can we learn from biology?
    - How do we cope with increasing complexity at each level?
      - Can software and hardware auto-tune?
      - How can layers adapt and co-operate under dynamic, heterogeneous regimes? (software, architecture and/or hardware changes)
  - How can efficient, effective communication across layers, or message optimisation across distributed systems, be achieved.
  - How do you ensure adaptation remains within bounds or the system can cope going beyond these bounds
    - Links to/depends on the ‘Living with Failure and Exposing Unreliability’ theme

Models, Reasoning and Methodologies

- Programming languages
  - What tools and models would make programming easier?
- Is there a future for threads?
  - Could process-based or message-based models be used instead?
- How can we reason about energy, performance, reliability?
- How can weak memory models be verified?
- How can an evolvable system be formalised?
- Can engineering methodologies and formal reasoning be integrated into tools and languages?
- What tools would allow system development, at all levels, to be simplified?
- How can the impact of software and operating systems on performance and reliability be assessed?
• Can design and programming consider several complementary failure states from the outset?
• Can a system degrade ‘gracefully’ rather than crash?
• How might reducing precision/performance impact on energy?
• Can failures be captured at a system level (verify/validate intent non-formally)
• How do we take account of finite reliability without jeopardising scalability?
• Can we develop stochastic specifications for both hardware and software synthesis?
• How can a system cope with incomplete or noisy data?
• How would you dynamically update this type of system?
• How can fault tolerant algorithms be developed and employed?

Sharing Knowledge and Expertise

• What can be learned from current work such as
  o Functional programming community (Haskell, Erlang, F#, Scala…);
  o Process community (π-calculus, bigraphs…)?
• What can be learnt from biology, particularly neuroscience?
• Collaboration is essential for multi-level solutions. What are the appropriate levels of abstraction that allow sharing?
• How can cross-pollination occur between the HPC and embedded communities?

The MACDES Priority and Call

Dr Christina Turner (EPSRC) gave an overview of how the information from the workshop would be used. The development of a strategy for the MACDES cross-ICT priority will be formed by identifying sub-priorities and considering what can be delivered through standard mechanisms and what needs specific activities such as a dedicated call. She went on to explained that a call relating to this priority was planned for early June 2012 and that the outputs of the workshop, and subsequent engagement with the ICT Strategic Advisory Team (SAT), would inform the content of this call.
Conclusions

The MACDES workshop aimed to engage researchers from across the ICT portfolio to explore common interest areas to help inform EPSRC about what direction this priority should take over the Delivery Plan. Specifically delegates were asked to identify what the key problems are for this priority and consider what activities both EPSRC and the community should undertake in order to deliver against them.

The workshop brought together a wide selection of people covering areas from across the community including microelectronic designers, computer architects, theoretical computer scientists and communication network researchers. Feedback the Theme received during and following the workshop suggests the delegates found it useful and a good opportunity to network with researchers in different areas around the same broad technical subject.

The information gathered from the workshop shows a clear need for researchers at the different levels in the system stack to work together around a number of problem areas. Whether it is to determine an appropriate abstraction level to work at for certain problems, to develop tools which require transparency between layers, or whether it is to begin answering questions such as ‘what does it mean for a system to be in an optimised state’? Energy, performance, reliability and resilience are just some of the issues highlighted. All of these need to be considered as whole system, integrated issues which require thought and cross-working by application programmers (at the top of the stack) right down to hardware architects.

The outputs of this workshop will inform the ICT Theme on actions it should take working with the research community to address the MACDES priority. The ICT Theme will engage with the ICT Strategic Advisory Team on this matter (including defining the scope of the call planned for June 2012) and also encourages researchers to consider the workshop outputs when developing their ideas for research proposals.
## Appendix A – Delegate List

<table>
<thead>
<tr>
<th>First name</th>
<th>Surname</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ali</td>
<td>Ahmadinia</td>
<td>Glasgow Caledonian University</td>
</tr>
<tr>
<td>Bashir M.</td>
<td>Al-Hashimi</td>
<td>University of Southampton</td>
</tr>
<tr>
<td>Frederick R.M.</td>
<td>Barnes</td>
<td>University of Kent</td>
</tr>
<tr>
<td>Jose Oswaldo</td>
<td>Cadenas</td>
<td>University of Reading</td>
</tr>
<tr>
<td>Murray</td>
<td>Cole</td>
<td>University of Edinburgh</td>
</tr>
<tr>
<td>George</td>
<td>Constantinides</td>
<td>Imperial College London</td>
</tr>
<tr>
<td>Kristof</td>
<td>Cools</td>
<td>University of Nottingham</td>
</tr>
<tr>
<td>Alastair</td>
<td>Donaldson</td>
<td>Imperial College London</td>
</tr>
<tr>
<td>Kerstin</td>
<td>Eder</td>
<td>University of Bristol</td>
</tr>
<tr>
<td>Bernd</td>
<td>Fischer</td>
<td>University of Southampton</td>
</tr>
<tr>
<td>Steve</td>
<td>Furber</td>
<td>University of Manchester</td>
</tr>
<tr>
<td>Dan R</td>
<td>Ghica</td>
<td>University of Birmingham</td>
</tr>
<tr>
<td>Mike</td>
<td>Giles</td>
<td>University of Oxford</td>
</tr>
<tr>
<td>Horacio</td>
<td>González-Vélez</td>
<td>Robert Gordon University</td>
</tr>
<tr>
<td>Kevin</td>
<td>Hammond</td>
<td>University of St Andrews</td>
</tr>
<tr>
<td>Cliff</td>
<td>Jones</td>
<td>Newcastle University</td>
</tr>
<tr>
<td>Daniel</td>
<td>Kroening</td>
<td>University of Oxford</td>
</tr>
<tr>
<td>Peter</td>
<td>Lee</td>
<td>University of Kent</td>
</tr>
<tr>
<td>Shujun</td>
<td>Li</td>
<td>University of Surrey</td>
</tr>
<tr>
<td>Mikel</td>
<td>Lujan</td>
<td>University of Manchester</td>
</tr>
<tr>
<td>Terrence</td>
<td>Mak</td>
<td>Newcastle University</td>
</tr>
<tr>
<td>Klaus</td>
<td>McDonald-Maier</td>
<td>University of Essex</td>
</tr>
<tr>
<td>Stephen</td>
<td>McGough</td>
<td>Newcastle University</td>
</tr>
<tr>
<td>Simon</td>
<td>McIntosh-Smith</td>
<td>University of Bristol</td>
</tr>
<tr>
<td>Robert</td>
<td>Mullins</td>
<td>University of Cambridge</td>
</tr>
<tr>
<td>Vijay</td>
<td>Nagarajan</td>
<td>University of Edinburgh</td>
</tr>
<tr>
<td>Peter</td>
<td>Newman</td>
<td>University of Edinburgh</td>
</tr>
<tr>
<td>Dimitrios S.</td>
<td>Nikolopoulos</td>
<td>Queen's University of Belfast</td>
</tr>
<tr>
<td>Peter</td>
<td>O'Hearn</td>
<td>Queen Mary, University of London</td>
</tr>
<tr>
<td>Ian</td>
<td>Phillips</td>
<td>ARM Ltd</td>
</tr>
<tr>
<td>Julian</td>
<td>Rathke</td>
<td>University of Southampton</td>
</tr>
<tr>
<td>Peter</td>
<td>Sewell</td>
<td>University of Cambridge</td>
</tr>
<tr>
<td>Jeremy</td>
<td>Singer</td>
<td>University of Glasgow</td>
</tr>
<tr>
<td>Gianluca</td>
<td>Tempesti</td>
<td>University of York</td>
</tr>
<tr>
<td>Phil</td>
<td>Trinder</td>
<td>Heriot Watt University</td>
</tr>
<tr>
<td>Wim</td>
<td>Vanderbauwhede</td>
<td>University of Glasgow</td>
</tr>
<tr>
<td>Hans</td>
<td>Vandierendonck</td>
<td>Queen's University Belfast</td>
</tr>
<tr>
<td>Philip</td>
<td>Wadler</td>
<td>University of Edinburgh</td>
</tr>
<tr>
<td>Roger</td>
<td>Woods</td>
<td>Queen's University Belfast</td>
</tr>
<tr>
<td>Nobuko</td>
<td>Yoshida</td>
<td>Imperial College London</td>
</tr>
<tr>
<td>Mark</td>
<td>Zwolinski</td>
<td>University of Southampton</td>
</tr>
<tr>
<td>Matthew</td>
<td>Ball</td>
<td>EPSRC</td>
</tr>
<tr>
<td>Liam</td>
<td>Blackwell</td>
<td>EPSRC</td>
</tr>
<tr>
<td>Katie</td>
<td>Blaney</td>
<td>EPSRC</td>
</tr>
<tr>
<td>Diane</td>
<td>Howard</td>
<td>EPSRC</td>
</tr>
<tr>
<td>Kathryn</td>
<td>Rosier</td>
<td>EPSRC</td>
</tr>
<tr>
<td>Christina</td>
<td>Turner</td>
<td>EPSRC</td>
</tr>
</tbody>
</table>
Appendix B – Outputs from Session 1 and 2

Techniques
• map-reduce
• dynamic analysis
• session types
• rely guarantee
• resource logics/ separation - linear

Methodology
• transparency
• reasoning
• formalism
• calculi
• layers of abstraction
• QoS (beautiful)

DISTRIBUTION
Distributed Systems

Performance
Constraints (fast)

Reliability/Resilience  Energy
Resources (easy)

Algorithmic Complexity

Micro – Design/technology problems at device e.g. mobile phone
Macro – gauged at “true system” level e.g. within cloud

1) Dynamic decisions
   Run-time optimisation at device level driven by energy
   System decisions – communications driven
   Where/when to converge

2) Cross layer integration/optimisation
   Abstraction  Transparency
   Energy  Communication

3) Validation of key parameters
   What are the right techniques for structuring distributed systems?
   Map-reduce, session types, ...
   Ease of expression
Build Systems
System Design not individual silos end to end – physical consumer product
Build many-core systems (strengthen)

Abstractions/patterns/feedback
- Energy Efficiency Programmability
  - 100x needed
  - Non-functional constraints
  - System Design Space Exploitation
  - Energy as a function of precision and performance

3 key points for feedback
- System approach to Energy & Reliability
  - “safely sailing close to the wind”
  - System optimisation & verification/semantics
  - Two communities in dialogue
- Programmability = Productivity

Robustness
- Reliability
  - Soft errors (y rays)
  - Noise in package
  - Variability at low feature size/geometry below 80nm
  - SRAM, logic next

Task composition
- Complexity
  - Emergent behaviour

Evolutionary Architecture
- Components/boxes for refuse (Hume)
- Driven by cost, causes problems elsewhere in systems

Need to break sequentially
- “Think parallel”

What not How
- Work as team

Productivity
- Virtualisation JIT adaptation
  - Powerful Learnt Lessons

Right building blocks & strategy then detail
- How do you teach programmers to do this? (Teaching)

New logic families

Reuse of hardware and software components

Solutions that generate - software reuse
1) SHARING – MEMORY
   - Easier to programme
   - Data flow, Erlang – no sharing?
   - Distribution of data prevents sharing?
   - Global nature of memory is a difficult issue

   - Challenges
     - Models
     - Architecture
     - Languages and types

2) POWER & ENERGY
   - Programming language level support
   - Tools already available for embedded systems
   - Power constraints in the language
   - Balance energy vs. other constraints
   - Use distribution to meet these constraints
   - Compare against infrastructure developments (e.g. ?, relocation etc)
   - “energy transparency” over the system stack top (s/w) – down (h/w)

Key questions
1) What are the trade offs between computational models based on SHARING or SHARE NOTHING (message passing)?
2) How can we enable ENERGY TRANSPARENCY at all levels of the system?
3) What are the right abstractions to support DISTRIBUTION (especially in the presence of unreliability) and SCALABILITY
Full parallelisation without hell-on-earth: Many solutions in different application domains HPC, embedded,?

Everything needs to be able to adapt to changing situations

Heterogeneity

1) Appropriate high level programming
   1) Abstractions DSLs, code generation, ‘active’ libraries
      1) Mass market programmers
      2) Reflection on performance/energy
      3) Scope for flexible choice of algorithms

Abstract/compare information for apps programmer

2) Techniques for better many-core system softwares: Currently do this but need to expose higher

1) Better compilation and tools: Not one solution to fit all
   1) Optimisation/autotuning for energy/performance
   2) Performance portability
   3) Exploiting heterogeneity

Mathematically rigorous engineering
   e.g. session types, static/dynamic analysis
Memory models, cross checking of optimised and un-optimised code
Proven libraries?

2) Resilience to hardware and software failures
   1) Sensitivity analysis (to failure)
Real challenge – don’t currently expose to higher levels
Appendix C – Outputs from Session 3

Programming Abstractions

- Bridging high-level abstraction with energy/performance. How?
- Getting this right enables most of the other areas to progress
- Need popular adoption not niche
- Models representing reliability/accuracy/QoS and supporting composition
- Increased transparency across abstraction boundaries and a clear methodology for assessing the impact of SW and O/S on performance/reliability
- In [15] micro do {...}, or with 90% chance of success do {...}
- PGAS (Partitioned global address space) relevant to embedded multicore/many core?
- Must reflect legacy and evolitional architecture
- Think about alternatives to text instructions written sequentially – graph representations?
- Do we need new languages/libraries/abstractions of concurrency?
- Refactoring and abstractions and legacy code?
- Program generation from domain-specific languages – autotuning
- Models of computation that represent communication
- Session-based communication/message-passing co-ordination and abstraction
- Purely functional programming
- Patterns of parallelism (skeletons?)
- Map reduce
- Need to interact with architecture co-design
- How to programme non-deterministic (probabilistic, stochastic) systems?
- Models that represent problem spaces not (just) implementation spaces
- Task composition
  - But not integrated into application community at least not in engineering and science (not true)
- Language support for energy
- Expressing parallelism and communication independently from hardware although exposing it
- Lots of work in the
  - FP community (Erlang, Scala, F#, Haskell, O’C?, Rachet)
  - Process community (π-calc, bigraphs, join calc)
  - Some work in both (linear logic, session types)
- Needs levels of abstraction and higher levels of abstractions
- Capture the form of information passed down the layers? – What do we gain?
- Need application developers input on kind of abstractions they’d be happy to use (and why) – or maybe need to put effort into training/educating users
- We chiefly need locality-aware programming abstractions
Communication and Data Movement

- Comms and DM as a part of the architecture of a system who’s functionality is meeting non/functional constraints
- Software to include cleaner indication of communication primitives and their importance on performance reliability
- Allow massively parallel communication - locality of computation
- Concerns of dynamic communication issue deadlock, heat, hotspots etc
- Programming languages explicitly capturing data movement
- Mapping/compilation into hardware
- Communication infrastructure for scalable and energy efficient systems
- Dynamic and static traffic management
- Moving code to ? data (not a new idea)
- Automatic synthesis of data movement code
- Checking correctness of data movement
- How to prune comms channels? (e.g. for efficient resource usage)
- How to deal with incomplete data? (return of value prediction?)
- Implicit locality management – explicit can be tedious for programmers
- **PGAS** (Partitioned Global Address Space)
  - Dynamic and static communication management for distributed systems using dynamic monitors (ensure liveness and safety)
- Is cache coherence necessary? Can it scale to manycore?
- Modelling computation vs. communication
- Whether to recomputed locally or fetch?
- Novel h/w mechanism to move away from Von-Neumann paradigm of proc/memory dichotomy
- Re-evaluation of computational complexity in context of energy
- Compute in transit
- Abstractions and parallel language constructs which enable seamless data and communications
- Expose cost* to the programmer (*energy, performance etc.)
- Right balance between on-chip/off chip computation/storage to achieve energy efficiency
- Communication vs. computation (or even Te-computation)
- Modelling as part of the high-level abstraction

Energy Efficient Computation

- Is 64 bit FP necessary? Or just the easiest way to code?
- Revisit alternative energy efficient arithmetic systems
- Methods to achieve power/energy return of the order x100 over non-constrained
- Top-bottom run-time awareness of energy
- Top-bottom design-time awareness of energy
- To achieve x100 probably needs new logic family (sub-threshold, 1°C, etc) which may also need new methods/tools
- Energy efficient algorithms
• Very long-term agenda: general-purpose slow computing (bio-scale clock rates and efficiency)
  o In [15] micro do {...} – make it scalable for future technologies
• How much does ? cost to “idle”?
• Energy management from high level language representations
• Energy efficient non-traditional architectures e.g. FPGA
• Find architectures (h/w, s/w, performance measures) that solve real problems efficiently
• Re-evaluation of computational complexity in context of energy efficiency
• Overcoming hw/sw gap
• Energy transparency over the system layers
• Energy modelling
• ANALOGUE! (Carver Mead 1980)
• Operating system support for policies on energy efficiency
• Mapping algorithms to HW: how to do Kud? Efficiency wrt energy consumption
• Application specific HW is generally more power efficient than general-purpose cores
• Energy as a resource in reasoning and synthesis frameworks
• Expose energy to the language/programmer
• Energy efficiency/initialization improvement through learning and exploration in run-time
• Circuit techniques for improving energy - ? exposing unreliability
• Full energy modelling of HW
• Exascale HPC is only one aspect of this, but in that area it needs involvement of application people
• Permitting specialisation without increasing complexity
• Estimates of future energy costs
  o Prototyping systems
  o Characterising future technologies
• Energy modelling and abstraction for higher level
• Closing the gap between HW & SW. Why does SW keep HW busy?
• How do you design a system if energy consumption was the first concern?
• The impact of s/w code and o/s management need to be “energy aware” – both HW and SW need to be involved
• Energy as a resource that needs to be managed as well
• The implications of hw choices and architectures need to be fully understood
• Memory energy efficiency! Data transfers consume a lot of energy! Don’t focus just on processors
• Doctoral training centres
• Industrial collaborations
Adaptive Run-time Optimisation

- Ultra-lightweight (millions of threads) threads (pico-threads/filaments)
- World and workload constantly changing – systems have to cope and adapt
- Better performance models
- High-level abstraction to support adaptive run-time optimisation
- Adapt existing static analysis e.g. timing analysis
- Learn from biology at all layers of runtime stack
- Static and dynamic analysis trade-offs – providing hardware “hooks” and tools for analysis
- What, not how
  - Program – what (specification)
  - Runtime – how (implementation)
- Need smart balance
  - Offline prior knowledge
  - On-line new circumstance
  - Cost benefit
- Tuneable architecture to enable run-time adaptation
- Run-time efficient optimisation architectures
- Better statistical models
- Smart run-time management to monitor energy and reliability as well as “performance”
- Real-time adaptation to heterogeneous resources
- Messaging optimisation for HPC and large-scale distributed systems
- How to make all (h/w and s/w) layers adapt towards some optimum? (need cost functions too)
  - Use control theory or game theory
- Continuous optimisation as runtime (both h/w and s/w)
- Increasingly important as hardware gets more complex. Needs involvement from application people
- Need new ML techniques to deal with own problems
- Autotuning software and hardware
- It is impractical to statistically define optimal run-time configurations in complex embedded systems. Run-time optimisation allows optima to be sought despite change of system context
- Notation systems to capture non-functional criteria that may subsequently be used for optimisation and a framework to capture specific criteria values
- Depends on a ‘what if fail’ framework to allow for out of bounds
- To be done before/with this - Need to work with abstraction designers to inform/define the optimisation space
- Find what information to pass between layers to make adaptation efficient. Think hard on what information to expose to or receive from the user
- Adaptation all the way down to the h/w – how to manage h/w dynamics?
Models, Reasoning and Methodologies

- Even professional, experienced programmers find multi-core challenging few genuine experts
- To allow formalism of the architecture of evolutorial systems
- Feedback directed (auto-) parallelisation; programmer in the loop
- Need new tools and models
- Need to go beyond “concurrency”
- Separation logic/ linear logic/ session types
- Most/all parallel programs are buggy – types for parallelism
- Engineering methodologies and formal reasoning into languages and tools?
- Make them simple
- Verification for weak memory models
- Reasoning about energy, performance and reliability etc.
- Connect all the theory to what is done/will/may be done in HW/system design
- Process calculi
- Functional languages
- Data flow programming and related models of computation
- Graphical programming languages
- Process-based/message-passing based modes
- Type-checking and design by contracts
- The end for threads?

Living with Failure and Exposing Unreliability

- Dying with failure or reducing unreliability
  - [Reducing unreliability] this is what we most need to do in all of CS
- Is this just a hardware issue? (Do we have to deal with failure in software?)
  - No, we have to eliminate it – never possible
- Observe – people aren’t perfect either
  - Hardware and software are never perfect, how do we cope with this
- Graceful degradation – never complete crash but lose precision/performance
- Consider different complementary failure states: non-operational, degraded…at design and programming
- System should be able to deal with continuous approximation based on noisy and incomplete data
- Failure is an inevitability due to imperfection in the design and implementation of products. As systems get more complex this will occur more regularly (transient and permanent). How do we make systems that work despite (a number of) these failures
- Capture and verification of intent at the system level (this is not formalism)
- Implementations of qualitatively functional systems need a change of methodology throughout the life-cycle.
- Hardware and software synthesis from stochastic specifications
- Taking into account finite reliability without jeopardising scalability
- Provide (tuneable) sweet spots between reliability and energy/performance costs
- Appropriate asynchronous exceptions based on communications
• Dynamic updates
• Balancing performance, energy and reliability
• Fault tolerant algorithm design
• Erlang community successful in this area
• The impact of graceful degradation and variable precision and reliable performance needs to be considered and its impact on “energy” evaluated.

Sharing Knowledge and Expertise

• Learning from biology (neuroscience?) – but not just copying biology
• Cross pollination between embedded computing and HPC – network? Produce tutorials?
• Requirements from different applications?
• Not objects-first, concurrent processes first!
• Semantics/verification people talking to hardware architects
• Language and hardware architects
• Languages for FPGA
• HW and SW experts exchange knowledge
• Community networking to define appropriate abstractions to enable sharing/encoding of information
• Where is the industrial input – problems? Money!
• Abstraction
• Explicit reusable models
  o Weak memory
  o Optimisations
• HW/SW co-design is a collaborative process
• Collaboration essential for the multilevel solutions (HW, OP. Sys, Apps) required