

New thought models for reconfigurability
and programmability

Dynamically Adaptable Behaviours (?)

Gordon Brebner
Xilinx Research Labs

What this talk contains

- Many questions
 - Some personal opinions
 - Some personal fragments
 - Few answers
-
- Aim to seed discussion during the day
 - ... and in the evening session too

Top-level thought model theme

- Might the next seminar be called

**“Dynamically Adaptable
Behaviours”**

instead of

**“Dynamically Reconfigurable
Architectures”?**

Reconfigurable, or programmable, or adaptable

- Question: “Is there a difference between a reconfigurable architecture and a programmable system?”
- Answer: The former can implement the latter - so “programmable” is maybe a more general and higher level term
- But “adaptable” is even higher level, because it refers to the system’s *behaviour*
 - It can adapt and/or it can be adapted

Behaviours or architectures?

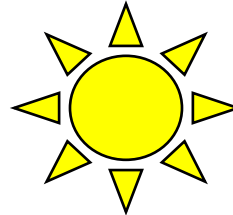
- Traditional focus has been on reconfigurable architectures (for programmable systems)
 - Then somehow map problem instances onto them
- It is more interesting – and is feasible - to bring reconfigurable architectures almost right up to the problem instances (in suitable domains)
- So the required instance-appropriate adaptability corresponds directly to a specially generated dynamically reconfigurable architecture

Idealised landscape

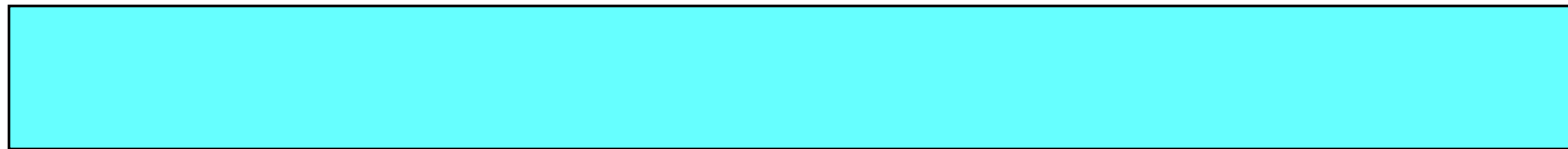
Systems in environments for people



Dynamically
Adaptable
Behaviours



Concurrency
Interconnectivity
Programmability



Raw technologies from nature

Our current landscape

Systems in environments for people



Hardware design



Software design



Dynamic reconfiguration black magic



FPGAs



...



Other programmables



Silicon, metal, etc.

Example of emphasis change

- Past decade or more has seen much work on run time “virtual hardware” support:
 - Linear or rectangular block placement
 - Some sort of routing between blocks
 - Mechanisms implemented on FPGAs
- Relative neglect of how to map real problem instances into this framework
- Should now work in the other direction

The FPGA: jailer or liberator?

- Buzzword for what the friendly FPGA can now offer:
 - *Hyper-programmability* (ASAP '04)
 - (Not to be confused with hyper-reconfigurability)
 - Program the architecture on the FPGA, then program it
- Current demonstrations:
 - At design time only – match architecture to static problem instance, within certain application domains
- Future work:
 - Post design time: system customization, configuration before use, adaptability when in use

Thought model sub-themes

- Is there a “third way” – neither conventional software nor conventional hardware nor a direct combination of both?
- Formal computability models for such a third way (in contrast to Turing machines, Boolean circuits, etc.)
- Theories of programming, and then languages and tools
- Application and environment specificity: planned irregularity
- Evolution, adaptability (self-programmability), fault tolerance, and other models from nature
- Safety and security: adaptability as a danger
- Future physical technologies and their potential for raw programmability
- Education, and encouraging non-legacy thinking

Background to the “third way”

- First Dagstuhl seminar on DRA, 1998
 - Subsequent paper in FPL '98
- The aim: stop talking about “hardware” and “software”
- Instead, trade off:
 - Algorithms versus architectures
 - Control flow versus data flow
 - Hardness versus softness

Is a “third way” madness?

- First way: traditional hardware design
- Second way: traditional software design
- Roadside is littered with skeletons of past attempts to do things differently: proposed languages, proposed methodologies, etc.
 - But C was new once, and not so long ago
- Now, we really have some different technologies to enable something different

Can a “third way” happen?

- Need collective effort to demonstrate that there can be some very different approach to productive problem solving
 - Natural for people to use
 - Efficient for technology to implement
- This is a long-term project
 - Maybe for next generation of humans
 - Maybe for future generations of technologies

Underpinning: formal models

- Have boolean and algebraic circuits for the first way
- Have Turing machines and random access machines for the second way
- What might be considered for a third way?
 - The Reconfigurable Mesh (RMESH) model has featured at previous seminars on DRA
 - The Local Access Stored Circuit (LASC) model was introduced at FPL '04

LASC machine model

- Adjust sequential “computing in time” RASP model to parallel “computing in space” LASC model containing a collection of *nodes*
- A node:
 - represents a processing element
 - embraces a disjoint set of local memory addresses, one of which contains an instruction
 - can access any memory addresses as function inputs, but only local addresses as outputs

LASC machine state

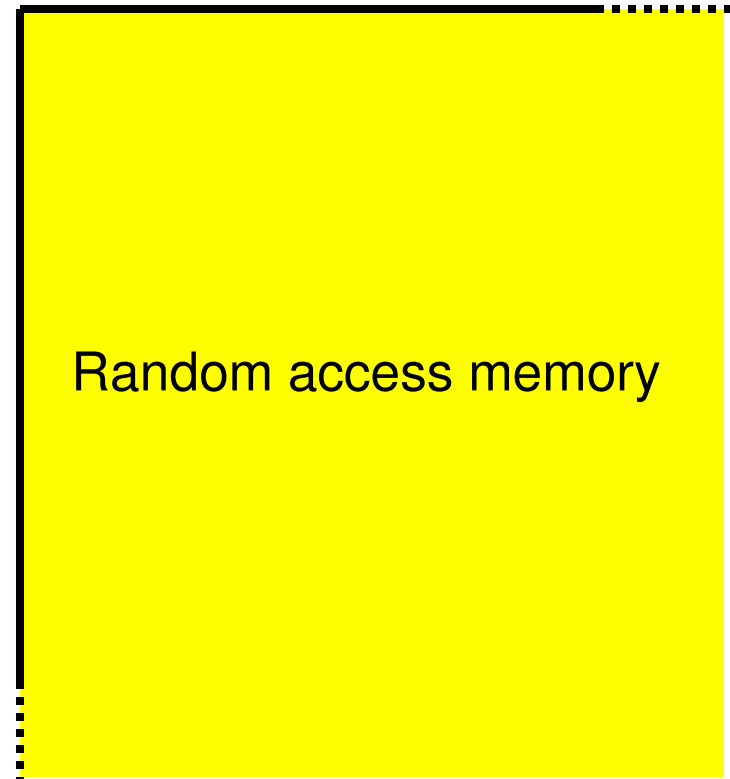
State:

Memory contents

**Instruction word
(one at each node):**
represents mapping
from current global
machine state to
new local node state

Words *(arbitrarily large)*

Addresses



(arbitrarily large)

LASC's only use, so far

- Used to prove:
 - Programmable logic has more computational power than fixed logic
 - Here, programmability refers to dynamic logic circuit changes during execution
- Possible beginnings for a computational model of dynamically adaptable behaviour
 - evolving set of interactions between behavioural (not architectural) elements

Theories of programming

- The current way: solutions are expressed as algorithms, which map to architectures
 - The programmability is in the architecture
- The proposed way:
 - Solutions are expressed in terms of the problem space, via human thought
 - Physical technology is harnessed to assemble flexible implementations of the solutions
 - Hyper-programmability may bridge the chasm

Doesn't look much like a theory ...

- Looked like it might be very special-case mapping on an ad hoc per-problem basis
- In fact, two compromises to incorporate:
 - Move from “general purpose” programming to broad domain-specific programming
 - Move from “general purpose” architectures to hyper-programmed domain-specific architectures
- Within a theory of hyper-programmability applicable to all domains of interest

Side comment on generalities

- “System on chip”:
 - Often means a scaled-down board architecture – processor, buses, i/o devices
 - This is far too specific – can do better
- “Network on chip”:
 - Often means a scaled-down packet switching network, or all-ways crossbar
 - This is far too general – can do better

Environment specificity

- Dynamically adaptive behaviours also need to be tailored to an application's entire environment:
 - What system features need to be adaptable?
 - How often do features need to be adapted?
 - What happens during adaptation?
- These need to be programmed naturally, just as software programs react to input events within a von Neumann straitjacket

Some evidence: networking

- Programming model: collection of (unordered) rules for packet processing, plus performance constraints
- Programmer just thinks in terms of packets, protocols, networks, etc.
 - i.e. domain behaviours
- Efficient mapping from programming model to hyper-programmed architecture framework

Some evidence: networking

- Hyper-programmed architecture is based on framework for efficient packet processing implemented on platform FPGA
- Processing elements:
 - Single/combined FSMs, microcoded datapath, processors
- Packet data distribution and storage:
 - Threads, pipelines, distributed memories

Nothing deep to say here about ...

- Adaptive and evolving systems, and gaining inspiration from nature
 - Important as source of ideas
 - Also important to model natural systems
- Safety and security issues
- Future physical technologies, and how they might be programmable
 - Don't just use them to model logic gates
 - Meanwhile, focus on FPGAs and silicon ...

Probably the most important thing

- Education at all levels
- Now: need to educate people on just where we are today
- More importantly, if we devise a “third way”, need to unravel legacy thinking and make people think afresh
 - Maybe it has to start at kindergarten, before the children get corrupted ...

The End

- No conclusion, just beginnings
- We have an opportunity to push further beyond the existing progress in dynamically reconfigurable architectures
- Substitute “deep adaptability” for current “shallow generality”
- However, it will be a battle against legacy
- But the technology is there, and the time is ripe, so this is a call to arms