Self-Optimising and Self-Verifying Design: a Vision

Wayne Luk

Department of Computing
Imperial College

Stamatis Vassiliadis Symposium

28 September 2007

Motivation

- 2005 *International Technology Roadmap for Semiconductors:* overall design challenges
  - cost-driven design optimisation
  - verification and test
  - re-use
- approach to address all 3 challenges?
  - key elements
  - challenges
  - summary

Approach: key elements

- optimise and verify: hardware + software
  - meet requirements efficiently and demonstrably
- self-optimising and self-verifying design (SOSV)
  - preserve property in design composition
- self?
  - aware of context
  - capable of planning
  - effective external control
- 2 stages
  - pre-deployment: building design, compile time
  - post-deployment: operational, run time

Pre- and post-deployment

	Pre-deployment	Post-deployment
focus	designer productivity	design efficiency
aim	optimize/verify initial post-deployment design	optimize according to situation
context	design tool environment, often static	operation environment, often dynamic
acquire context	from parameters affecting tool performance	from data input e.g. sensors
planning	plan post-deployment optimise/verify	plan to meet post-deployment goals
external control	frequent	infrequent

Re-use

- high-level generic design
  - requirements + context: multiple designs
  - optimise: options + parameters + abstraction levels
- facilitate design composition
  - preserve self-optimising and self-verifying
  - modularity of building blocks + interfaces
- platform-based evolution
  - re-use un-verified design: risky
  - automate re-verification after changes
  - platform for re-use: from *auto* to *self*, helpfully

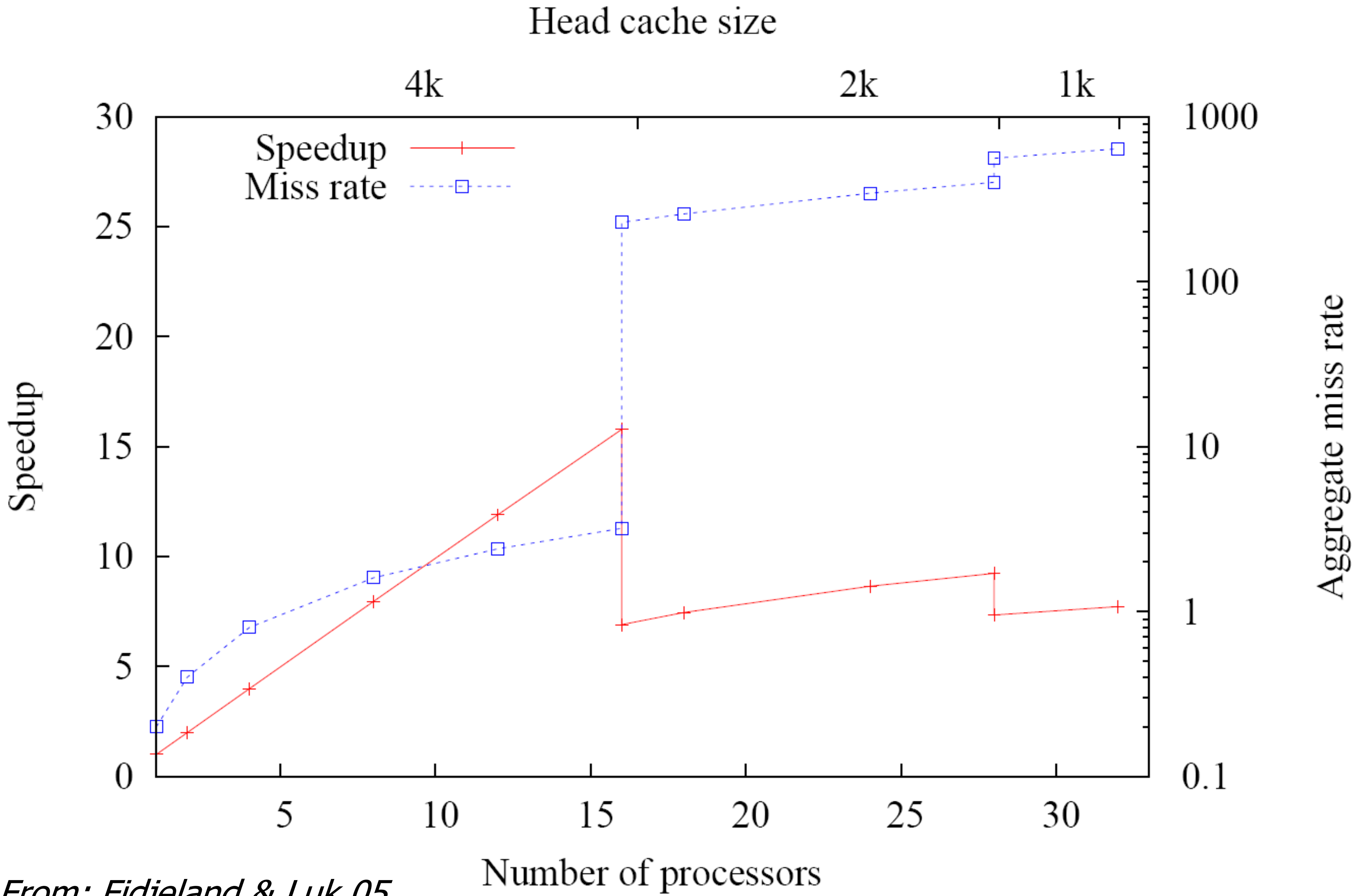
Pre-deployment: overview

- available computing resources
  - components + pre-context: locate and tune tools
  - current context: optimise resource + error recovery
- designer
  - adapt components: requirements + post-context
  - choose control: automation of search strategies
  - decide: re-use or re-invent
- challenges
  - productive interaction: designer + tools
  - avoid combinatorial explosion
  - maximise re-use: incremental design

Pre-deployment: example of choices

- circuit technology: eg ASIC or FPGA
- input/output: options
- memory: hierarchy + options
- interconnect: e.g. bus, switch, network-on-chip
- granularity: configurable unit, custom instruction
- synchronisation: e.g. clock domains, self-timed
- parallelism: processors, hardware/software
- data representation optimisation
- post-deployment optimisation/verification

Example: number of processors



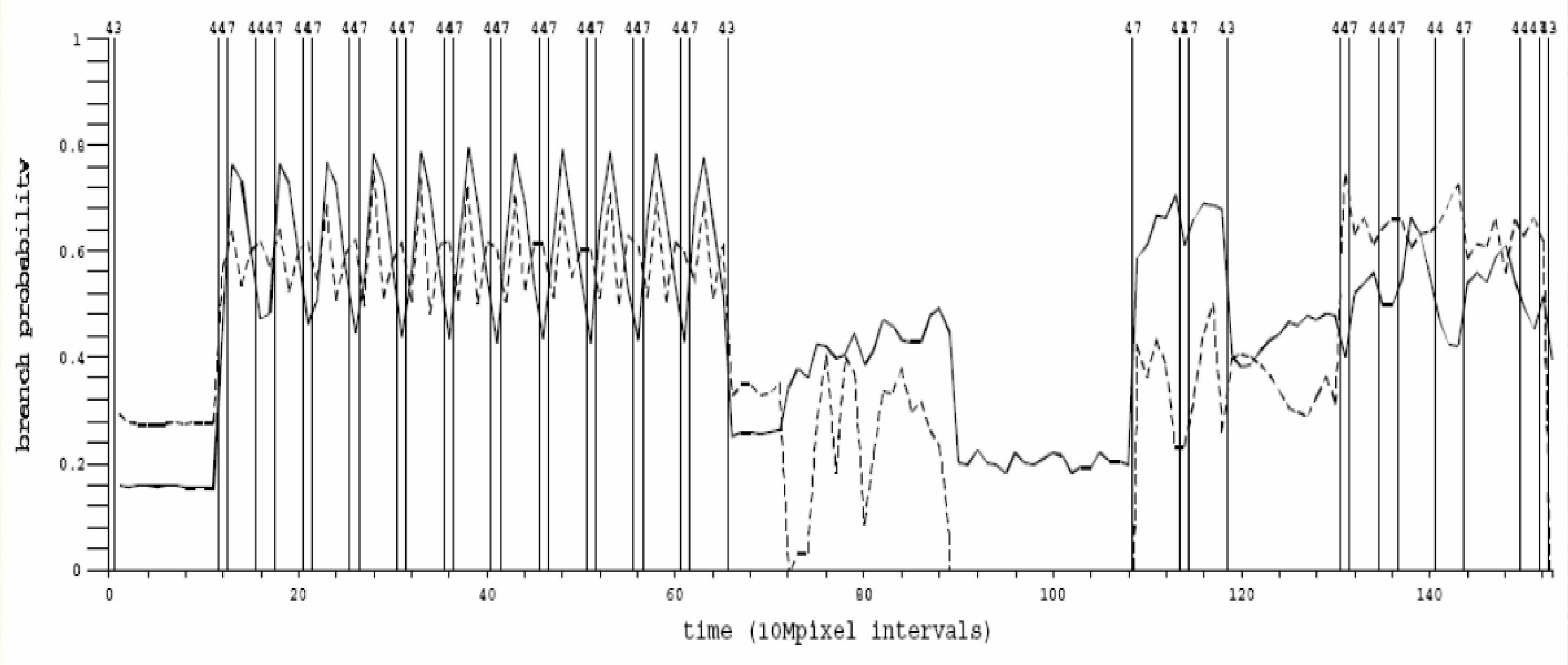
From: Fidjeland & Luk 05

Post-deployment: situation-specific

- optimization and verification opportunities
  - design upgrade
  - run-time conditions, e.g. noise, process variation
  - program phase optimisation
- optimisation and verification process
  - light-weight: on-site, e.g. proof-carrying code
  - heavy-weight: remotely, verified by signature
- run-time system
  - deals with exceptions
  - error diagnosis facilities

Example: program phase optimisation

- program phase: working set remains constant
- reconfigure to speed up frequent branches

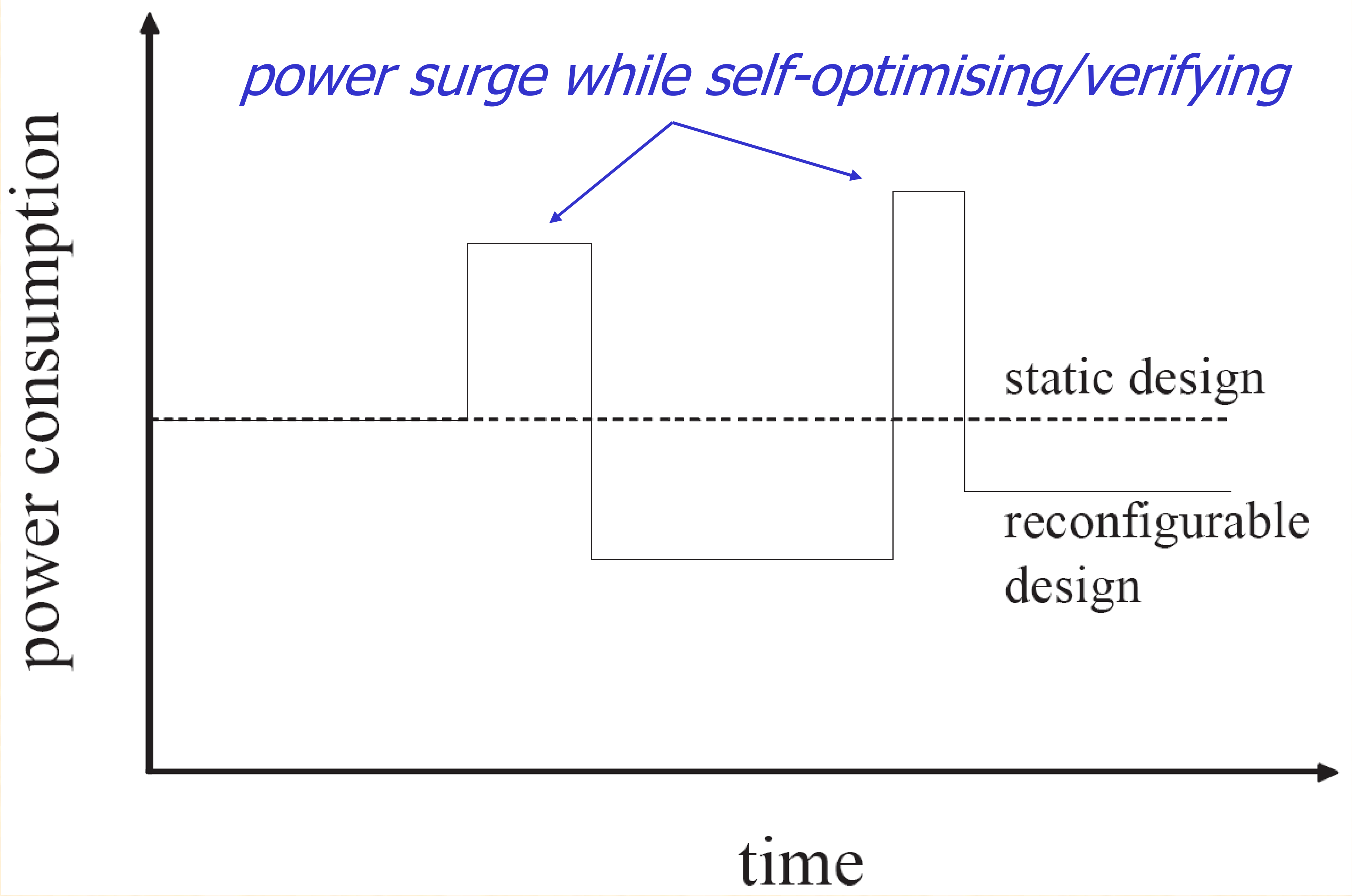


From: Styles and Luk 05

Autonomous systems

- control strategy
  - make decisions to optimise itself
  - model of world: planning and action
  - understand trade-offs: e.g. reconfigure or not
- event-driven just-in-time reconfiguration
  - component meta-data description
  - assemble + tune partially-optimised components
  - hide reconfiguration latency
- other possibilities
  - machine learning
  - self-organising feature map

Example: dynamic power optimisation



Challenges: theory + practice for:

- productive automate: evolutionary vs disruptive
- SOSV design: specify + analyse requirements
- composable description: design + context
- multi-level capture: domain-specific constraints
- open standard: design, optim/verify programs

Summary

- self * (optimising+verifying) = trusted re-use
  - unify: autonomic, self-test, dynamic optim., RTR
  - better design + more productive
- self-optimising self-verifying design platform
  - FPGA-based systems: large + small
  - autonomous system-on-chip + network of ASOCs
  - applications: ubiquitous, dependable, secure, robust
- new generation of designers
  - building blocks + tools: made smarter
  - specify, analyse, adapt: requirements + search