Reconfigurable Computing in the Multi-Core Era

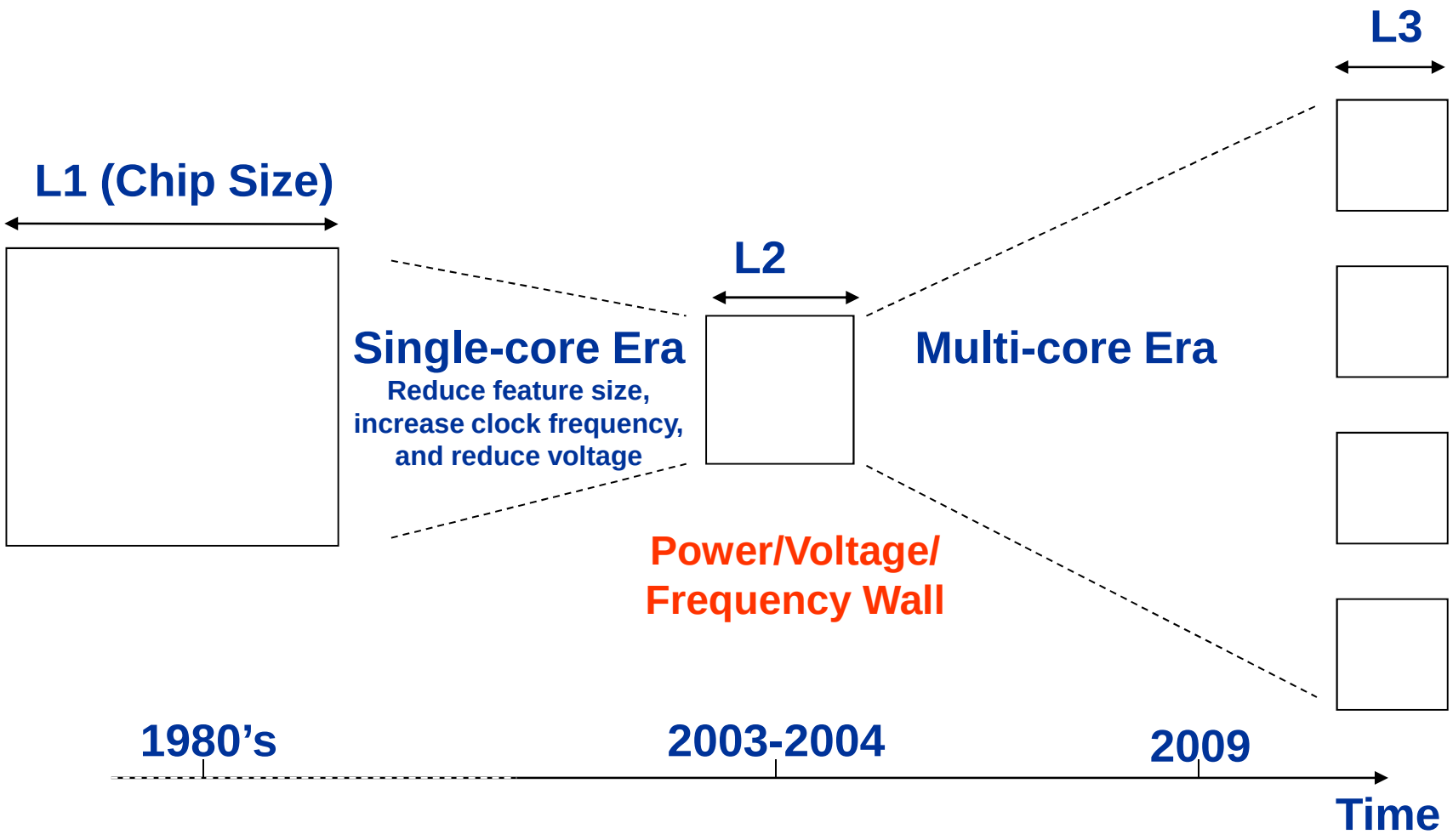
Khaled Benkrid, PhD, CEng, MBA

International Workshop on Highly Efficient Accelerators and Reconfigurable Technologies (HEART 2010)

Outline

1. Introduction: the semantic gap between applications and hardware
2. Parallel computer technologies
   - Multi-core general purpose processors (GPPs)
   - Application-specific Processors
   - Field Programmable Gate Arrays (FPGAs)
   - Fixed ASICs
3. Comparative Studies: FPGAs vs. GPUs vs. IBM Cell vs. GPPs
4. Reconfigurable computing, is it finally the time?
5. Heterogeneous computing, is it the way forward?
6. Conclusions

Introduction 1/2



Introduction 2/2

- A **semantic gap** is opening between applications (traditionally written in sequential code) and hardware (now essentially parallel)
- Different approaches to parallel hardware programming are being followed stretching from:
  - Attempting to parallelise sequential code automatically

    to
  - Programming/designing from pure parallel code
- This gap is, ironically perhaps, opening a window of opportunity for new parallel technologies in mainstream computing

Outline

1. Introduction: the semantic gap between applications and hardware
2. Parallel computer technologies
   - Multi-core general purpose processors (GPPs)
   - Application-specific Processors
   - Field Programmable Gate Arrays (FPGAs)
   - Fixed ASICs
3. Comparative Studies: FPGAs vs. GPUs vs. IBM Cell vs. GPPs
4. Reconfigurable computing, is it finally the time?
5. Heterogeneous computing, is it the way forward?
6. Conclusions

Parallel computer technologies 1/2

Technology	Performance / Cost	Time to market	Time to change code functionality	Power Consumption
GPPs	Low-Medium	Very Short	Very Short	High
Application-specific processors e.g. DSPs, GPUs	Medium	Medium	Medium	Medium-High
FPGAs	Medium-High	Long	Long	Low-Medium
Fixed ASICs	Very High (for high volumes)	Very Long	Impossible	Low

Speed Performance

Flexibility

Parallel computer technologies 2/2

- Different technologies offer different advantages
- On the performance scale, fixed ASICs offer the ultimate speed and power consumption, whereas GPPs offer the ultimate flexibility
- FPGAs and application-specific processors e.g. DSPs and GPUs, occupy the middle-ground
- FPGAs have ASIC-like performance and are now leading the process technology
  - They suffer from their low-level programming model though
- GPUs can offer much higher performance than GPPs at a low cost
  - They suffer from a relatively longer development time (compared to GPPs) and a high power consumption

Outline

1. Introduction: the semantic gap between applications and hardware
2. Parallel computer technologies
   - Multi-core general purpose processors (GPPs)
   - Application-specific Processors
   - Field Programmable Gate Arrays (FPGAs)
   - Fixed ASICs
3. Comparative Studies: FPGAs vs. GPUs vs. IBM Cell vs. GPPs
4. Reconfigurable computing, is it finally the time?
5. Heterogeneous computing, is it the way forward?
6. Conclusions

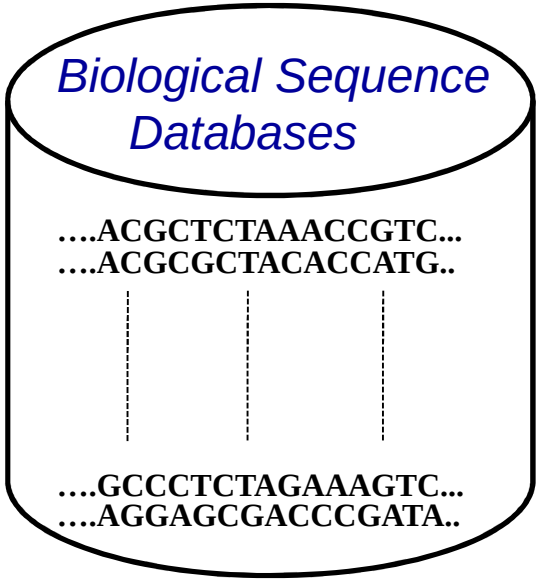
Comparative Study 1: Smith-Waterman Algorithm

pairwise Sequence Alignment e.g. DNA

Query Sequence

….ACCCTTTTGGGG

Find the closest matching sequence

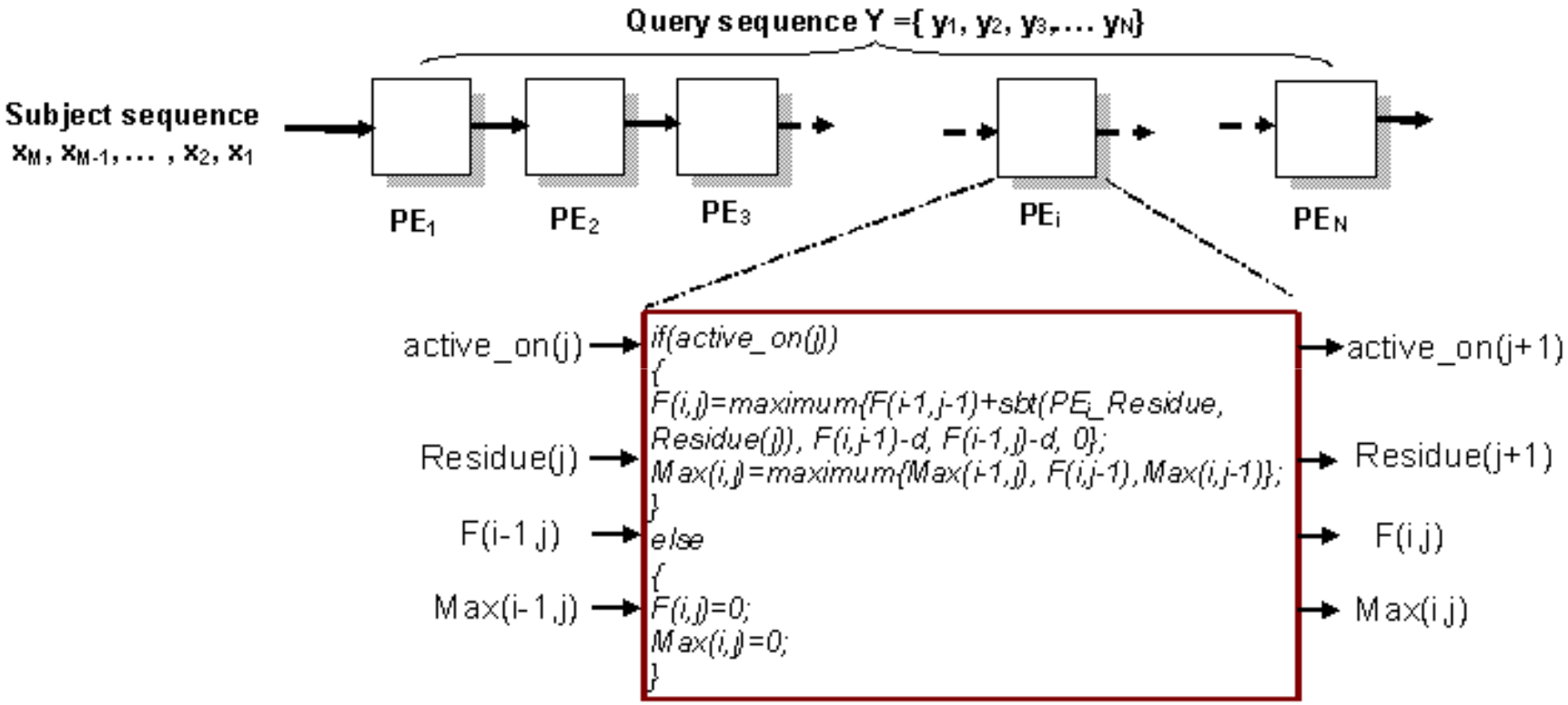


		H	E	A	G	A	W	G	H	E	E
	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	**5**	0	0	0	0	0
W	0	0	0	0	2	0	**20**	**12**	4	0	0
H	0	10	2	0	0	0	12	18	**22**	14	6
E	0	2	16	8	0	0	4	10	18	**28**	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

Best Local Alignment: **AW GHE**
AW - HE

$$F(i,j) = \max \begin{cases} 0 \\ F(i-1,j-1) + S(x_i, y_j), & \text{where } x_i \text{ aligned to } y_j \\ F(i-1,j) - d, & \text{where } x_i \text{ aligned to a gap} \\ F(i,j-1) - d, & \text{where } y_j \text{ aligned to a gap} \end{cases}$$

Comparative Study 1: Smith-Waterman Reconfigurable Hardware Skeleton



*Reduces time complexity from O(m*n) to O(m+n)*

K. Benkrid et al., 'A Highly Parameterised and Efficient FPGA-Based Skeleton for Pairwise Biological Sequence Alignment', IEEE TVLSI Journal, Vol. 17, Issue 4, pp. 561-570, April 2009

Comparative Study 1: Smith-Waterman Skeleton Implementation

- Xilinx Virtex 4 LX160 -11 vs. NVIDIA GeForce 8800GTX GPU vs. IBM's Cell BE processor vs. 3.4 GHz Pentium 4 Prescott processor
- Design and implementations performed by four PhD students with equal experience on each respective platform
- Comparative criteria:
  - Speed performance
  - Energy Consumption
  - Development Time
  - Cost of development
  - Performance per $
  - Performance per Watt

Comparative Study 1: Smith-Waterman Skeleton Implementation

Speed performance comparison for query sequence of length 256

Platform	GCUPS*	Speed-Up
FPGA	19.4	228:1
GPU	1.2	14:1
Cell BE	3.84	45:1
GPP	0.085	1:1

** Giga Cell Updates Per Second*

Comparative Study 1: Smith-Waterman Skeleton Implementation

Development times

Platform	Development time in Days
FPGA	300
GPU	45
Cell BE	90
GPP	1

Comparative Study 1: Smith-Waterman Skeleton Implementation

Cost of development

Platform	Purchase Cost ($)	Development Cost ($)	Overall Cost ($)	Normalised Overall Cost
FPGA	10,000	48,000	58,000	50
GPU	1450	7,200	8,650	8
Cell BE	8,000	14,400	22,400	19
GPP	1000	160	1160	1

Comparative Study 1: Smith-Waterman Skeleton Implementation

Performance per $

Platform	Performance (MCUPS*) per $ spent	Normalised Performance per $ spent
FPGA	0.34	4.6
GPU	0.14	1.9
Cell BE	0.17	2.3
GPP	0.07	1

** Mega Cell Updates Per Second*

Comparative Study 1: Smith-Waterman Skeleton Implementation

Power and energy consumption

Platform	Power (Watt)	Energy (Joule)	Normalised Energy Consumption
FPGA (clocked at 80MHz)	39	73	0.0017
GPU	56	1682	0.04
Cell BE	140	1317	0.03
GPP	100	42400	1

Comparative Study 1: Smith-Waterman Skeleton Implementation

Performance per Watt

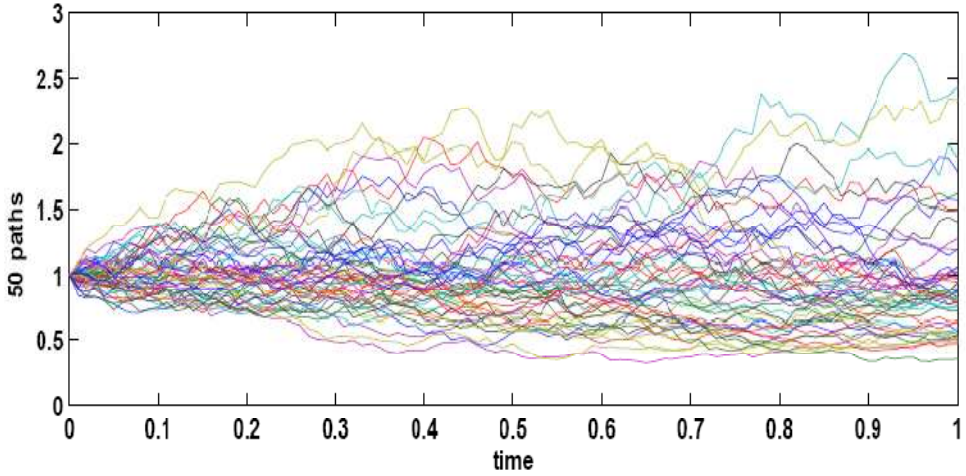
Platform	Performance (MCUPS) per Watt	Normalised Performance per Watt
FPGA	508	584
GPU	22	25
Cell BE	27	31
GPP	0.87	1

Comparative Study 2: Quasi-Monte-Carlo-based Financial Option Pricing

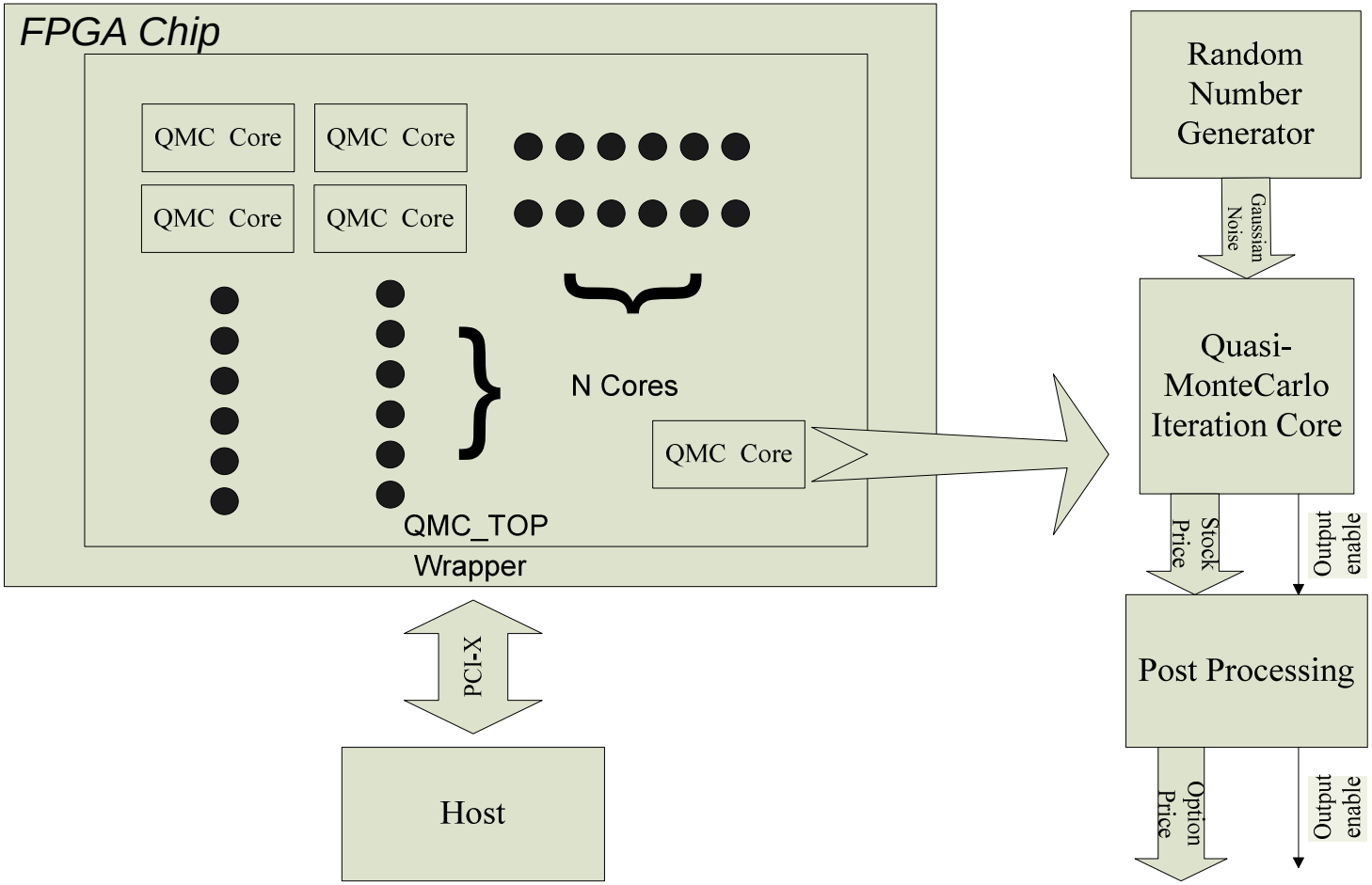
- Quasi Monte-Carlo simulation of European options
  - Simulation of stochastic processes
  - Random sampling using Sobol numbers

$$S_t = S_0(1+((\mu-\frac{\sigma^2}{2})\delta t+\sigma\varepsilon\sqrt{\delta t}))$$

S_t : stock price at time t
μ : expected rate of return
σ : volatility of the stock price
ε : random variable with a normal distribution



Comparative Study 2: Generic Hardware Skeleton for QMC Simulation



X. Tian and K. Benkrid, "High Performance Quasi-Monte Carlo Financial Simulation: FPGA vs. GPP vs. GPU", to appear In ACM Transaction on Reconfigurable Technology and Systems, 2010.

Comparative Study 2: QMC Simulation Skeleton Implementation

- Xilinx Virtex4 VFX100-10 vs. NVIDIA GeForce 8800GTX GPU vs. vs. 2.8GHz Intel Xeon Processor
- Design and implementations performed by three PhD students with equal experience on each respective platform
- Comparative criteria:
  - Speed performance
  - Energy Consumption
  - Development Time
  - Cost of development
  - Performance per $
  - Performance per Watt

Comparative Study 2:
QMC Simulation Skeleton Implementation

Speed performance price for a single option pricing, using 524,288 simulation paths

Platform	Speed (ms)	Performance (Paths per Sec)	Normalised Speed-Up
FPGA	7.9	66,618,551	545:1
GPU	86	6,110,583	50:1
GPP	4291	122,180	1:1

Comparative Study 2:
QMC Simulation Skeleton Implementation

Development times

Platform	Development time in Days
FPGA	60
GPU	3
GPP	1

Comparative Study 2: QMC Simulation Skeleton Implementation

Cost of development

Platform	Purchase Cost ($)	Development Cost ($)	Overall Cost ($)	Normalised Overall Cost
FPGA	10,000	9600	19,600	17:1
GPU	1350	480	1,830	1.6:1
GPP	1000	160	1160	1:1

Comparative Study 2:
QMC Simulation Skeleton Implementation

Performance per $

Platform	Performance (Paths/sec) per $ spent	Normalised Performance per $ spent
FPGA	3399	32:1
GPU	3339	32:1
GPP	105	1:1

Comparative Study 2:
QMC Simulation Skeleton Implementation

Power and energy consumption

Platform	Power (Watt)	Energy (Joule)	Normalised Energy Consumption
FPGA (clocked at 75MHz)	20	0.16	0.001:1
GPU	95	8.5	0.05:1
GPP	170	172	1:1

Comparative Study 2: QMC Simulation Skeleton Implementation

Performance per Watt

Platform	Paths per Second Per Watt	Normalised Performance per Watt
FPGA	3,330,928	1090:1
GPU	64,322	21:1
GPP	3,055	1:1

Outline

1. Introduction: the semantic gap between applications and hardware
2. Parallel computer technologies
   - Multi-core general purpose processors (GPPs)
   - Application-specific Processors
   - Field Programmable Gate Arrays (FPGAs)
   - Fixed ASICs
3. Comparative Studies: FPGAs vs. GPUs vs. IBM Cell vs. GPPs
4. Reconfigurable computing, is it finally the time?
5. Heterogeneous computing, is it the way forward?
6. Conclusions

Reconfigurable computing, is it finally the time?

- FPGA technology's main competitive advantage is on performance per watt grounds, but similar experiments for different applications need to be performed
- High performance computing applications where power consumption is often a bottleneck should hence benefit greatly from this technology
- For many applications, FPGAs are more competitive than alternative technologies on performance per $ ground
- For this to hold, a minimum of two-orders of magnitude speed up compared to GPPs and one order of magnitude compared to GPUs is needed

Reconfigurable computing, is it finally the time?

- GPUs are very competitive on performance per $ ground compared to FPGAs and GPPs. They are competitive compared to GPPs on performance per watt grounds
- FPGAs Achilles' Heel is in their long development time
  - Relatively low level HDLs (VHDL/Verilog) are still dominant
  - A large part of FPGA solution development is spent on learning specific FPGA board APIs and debugging in hardware (70% in our experiments!)
  - Unlike software, FPGAs do not currently offer forward/backward compatibility, not even within the same family!
  - FPGAs have a relatively low technology maturity and small user base compared to software

Reconfigurable computing, is it finally the time?

- Standard FPGA boards with standard languages and APIs can lower this hurdle drastically:
  - Standard FPGA boards' I/O
  - Standard High Level Languages (HLLs) for FPGA programming (C-to-gate)
  - Standard MPI-like support for FPGA-based process communication
- Unless standardisation efforts materialise, FPGAs will still struggle to get a foothold into more mainstream computing

Reconfigurable computing, is it finally the time?

- However, reconfigurable hardware has many advantages to bring to a heterogeneous computer platform, perhaps as a pre/co-processor:
  - Low latency and high bandwidth I/O
  - Reprogrammable custom hardware (high performance) and low power
- The recent announcement from Xilinx of a new FPGA architecture built around the ARM Cortex-A9 MPCore could be a template towards standardisation

Outline

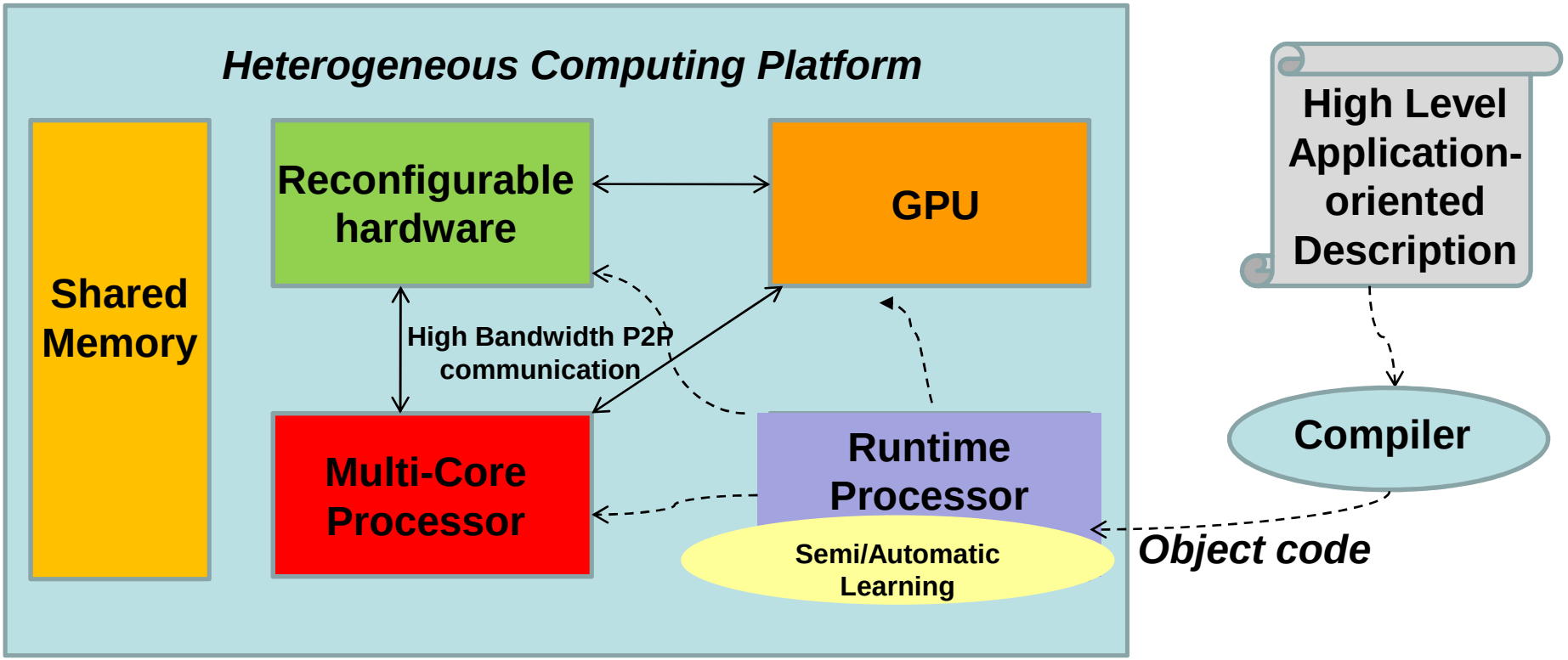
1. Introduction: the semantic gap between applications and hardware
2. Parallel computer technologies
   - Multi-core general purpose processors (GPPs)
   - Application-specific Processors
   - Field Programmable Gate Arrays (FPGAs)
   - Fixed ASICs
3. Comparative Studies: FPGAs vs. GPUs vs. IBM Cell vs. GPPs
4. Reconfigurable computing, is it finally the time?
5. Heterogeneous computing, is it the way forward?
6. Conclusions

Heterogeneous computing, is it the way forward?

- Increased choice in computer platforms means that heterogeneity is increasingly a practical and economical solution to many modern application needs
- The trend towards more consumer choice and increasingly different and various needs also favours heterogeneous computer platforms
- Many issues need to be addressed including:
  - Which Architecture?
  - Which Programming model?
  - Which Compiler?
  - Which Runtime System?

Heterogeneous computing, is it the way forward?

Possible Heterogeneous Platform



Heterogeneous computing, is it the way forward?

- Heterogeneity makes programming more difficult, but the potential gains can justify the extra effort
- A multi-language programming flow is perhaps the answer
  - With standard interfaces between languages that allow for dynamic exchange of data and remote procedure invocation
- Heterogeneity facilitates performance and power scalability because it increases design space options for system developers
- Efficacious and efficient compiler and run-time system support for such systems is critical!

Outline

1. Introduction: the semantic gap between applications and hardware
2. Parallel computer technologies
   - Multi-core general purpose processors (GPPs)
   - Application-specific Processors
   - Field Programmable Gate Arrays (FPGAs)
   - Fixed ASICs
3. Comparative Studies: FPGAs vs. GPUs vs. IBM Cell vs. GPPs
4. Reconfigurable computing, is it finally the time?
5. Heterogeneous computing, is it the way forward?
6. Conclusions

Conclusions

- The lack of standards (APIs, boards, communication interfaces) is holding reconfigurable hardware technology back
- Reconfigurable hardware is likely to be part of a heterogeneous computing mix in the future because of its unique position in the mix
  - Low latency and high bandwidth I/Os, custom hardware performance, low power + re-programmability
- Recent processor-centric architectures e.g. Xilinx ARM Cortex-A9 MPCore based platform, are a good sign
  - Ride the processor standards curve
- The custom computing community should encourage standards, open source IPs and platforms

Acknowledgement

- PhD students
  - Mr Xiang Tian
  - Ms Ying Liu
  - Mr Server Kasap

- Research Collaborators
  - Dr. Tsuyoshi Hamada, Nagasaki University
  - Dr. Ali Akoglu, University of Arizona

Thank You For Your Attention!

Questions?