The golden triangle of Hybrid Computing: FPGAs, GPGPUs and Many-Core processors

Stefan Möhl, Co-founder, CSO, Mitrionics
The Mitrion Virtual Processor
The Mitrion Virtual Processor (MVP)

- Massively fine-grain MIMD parallel
  - Instruction level parallel
  - 10’s of thousands of PEs
- A configurable processor design
  - The processing elements suitable for the application
  - Processing elements adapted to bit-width
  - Ad-hoc fused instructions
- Allows software programming for FPGAs
  - You program the MVP rather than design HW
  - Programmed in inherently parallel programming language Mitrion-C
The Mitrion Platform

2009-07-28, by Stefan Möhl
The Mitrion Virtual Processor (MVP)

+ Massive parallelism gives 10-30 times the performance of traditional CPUs

+ High silicon utilization gives full performance at around 25 Watts

➢ Optimal for acceleration of data oriented programs

```c
int:48<30> main()
{
    int:48 prev = 1;
    int:48 fib = 1;
    int:48<30> fibonacci = for(i in <1..30>){
        fib = fib+prev;
        prev = fib;
        <>fib;
    } <>fib;
    fibonacci;
}
```
Hybrid Computers

FPGAs, GPGPUs and Many-Core
Hybrid a.k.a. Heterogeneous Computing

Many-core CPUs
Command & Control

FPGAs
App specific data types

GPGPUs
Floating Point
Why hybrid computing?

• Each compared to the two others:
  – CPUs are a magnitude faster on sequential code and control-flow dominated code
  – GPGPUs are a magnitude faster on floating-point
  – FPGAs are a magnitude faster on application specific data types
    » Any application specific data, like genomic data
    » Application specific internal data structures
    » Common but poorly supported data types, like Logarithmic number systems, Arbitrary precision math, Binary coded decimal, binary, integer, fix point, text, etc

• Large performance gains available if the strength of all platforms can be used in tight integration!
FPGAs are in it for the long run

• GPGPUs are getting a lot of attention
  – But that does not mean the market window for FPGAs has closed!

• FPGAs are old mature devices that will continue on Moore’s Law

• There will always be applications that have data that the more specialized devices don’t focus on

➢ FPGA acceleration will always be around!
The future of FPGA acceleration

In-socket Modules
In-socket modules

• The new generation of FPGA modules

• An FPGA mounted directly in a CPU socket of a standard multi-socket motherboard
  – Including SRAMs for local cache memory

• Gives direct access to system bus
In-socket modules

• In-socket accelerators are true peers to CPUs
  – Reside directly on system bus
  – Full, equal access to main system memory, I/O, etc

• No copying of data to accelerator card before processing and copying back afterwards
  – Accelerator can operate on data directly, where it is!
  – The difference between copying the pointer to an array and fully cloning the array
What Mitrionics Delivers

• Pre-configured Hybrid Computing Systems including full programming environment for application development and acceleration

• The Mitrion Platform: Enabling software for FPGAs
  – The Mitrion Virtual Processor (MVP) – Hardware IP for parallel processor, fully adapted for the application, runs in FPGA chip
  – The Mitrion SDK – including inherently parallel C family language Mitrion-C, compiler and debugger

• Training and development services available
  – See our website!
First time Sneak Preview here at SAAHPC:

Mitrionics is exploring Mitrion-C on Multi-core, Clusters and more...
What we are trying to do

• Allow Mitrion-C code to run efficiently across a Hybrid machine
  – FPGA
  – Multi-Core/Many-Core
  – Clusters
  – And more to follow (GPU, vector and others)

• Letting FPGA code out of the FPGA box
  – The FPGA acceleration market suffers from users afraid of locking their code into FPGA acceleration technology
Many core will require code re-writes
Code Re-Writes have happened many times...

- The starting point: Simple Sequential Computers
- Vector supercomputers
- Thinking Machines and MasPar
- SMPs/ccNUMA
- MPPs
- Clusters
- Grids
- FPGAs and GPGPUs
- Many-core processors
A new take on the problem

• Running sequential code in parallel is hard
• Running parallel code sequentially is easy
• Automatically parallelizing for the different paradigms was not possible
  – Code re-writes

• Automatically sequentializing for different parallel paradigms is probably comparatively easy
  – Less, or maybe even No Code Re-Writes
Some properties of Mitrion-C

• Fine-grain (instruction-level) MIMD parallel
  – The MVP requires fine-grain MIMD for performance
  – The extreme of parallelism
  – Currently scales to 10’s of thousands of PEs in the MVP, no scalability limit yet visible.

• Memory and bandwidth are taken seriously
  – FPGAs typically sit on data buses made for 1/10 the performance

• Though Imperative style syntax, semantics are purely Functional
  – Very amenable to compiler optimizations and theoretical treatment
Fine-Grain Parallelism: An example

- Non-strict execution

```c
int:8 a; int:8 b; int:8 c;
x = f(a, b, c);
// Non-strict allows f() to run as far as
// possible while a, b and/or c are still being
// computed

int:8<100> a; int:8<100> b;
(x, y) = f(a, b);
z = g(x, y);
// Non-strict allows f() and g() to run in
// parallel, while elements of x and y are being
// computed
```
Mitrion-C takes Memory, Locality and Bandwidth seriously

- Memory
  - No illusion of single address space (we haven’t had that since the introduction of caches 20 years ago)
  - Different memory spaces for different bandwidth and latency requirements
  - Careful control of memory usage (the MVP has limited access to fast memory)

- Bandwidth
  - Separate memory spaces implies separate data buses. This allows programming for bandwidth
  - Streaming data types that help conserve bandwidth
  - Highlighting of bandwidth consumption in debugger

- Parallelism allows latency to be hidden
Does it have to be a new language?

- Not completely, Mitrion-C is a C-family language
  - Syntax is as similar to C as Java is similar to C
  - But with radically different language semantics
- Learning a new language isn’t that hard
  - Most coders know 10+ different languages
- Learning parallel programming is very hard
  - Most parallel programmers are still learning
- A language should support the hard task
  - Do for Parallelism what Object-Oriented languages did for state
The approach is not going to be easy

- But at least it shows promise, and has not yet proven hopeless
- If anyone is interested in supporting us, please contact me!
Thank You!

Mitrionics AB
Ideon Science Park
SE-223 70 Lund
www.mitrionics.com

stefan@mitrionics.com