The Mitrion Virtual Processor

Using FPGAs in HPC

Stefan Möhl, Co-founder, Chief Strategy Officer,

Mitrionics

mitrionics™

Why FPGAs in HPC?

- Compared to fixed silicon at the same process technology (65nm):
  - ~10 times slower clock frequency
  - ~50-100 times larger area used per gate
- Compared to CPUs
  - 10-100 times faster
  - At 20-25 Watts!
  - Performance gap is increasing

mitrionics™

The HPC FPGA Eco-System

mitrionics™

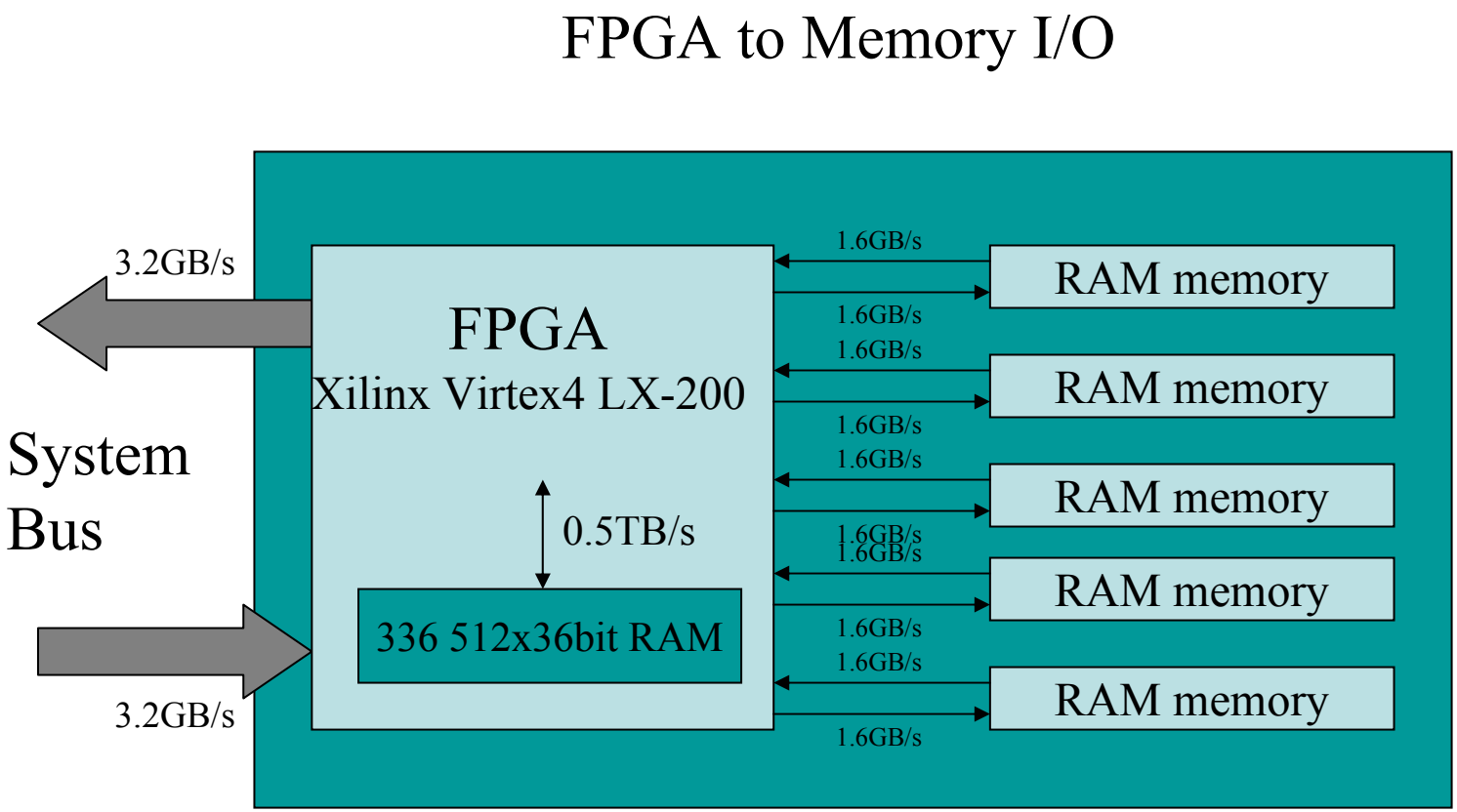
FPGAs in HPC vs Embedded

- Embedded is about building devices
  - Adapt the machine to the implementation
  - Hardware design

- HPC is about using the machine
  - Find the implementation that suits the machine
  - Software programming

mitrionics™

A Typical FPGA Module (SGI RC100)



mitrionics™

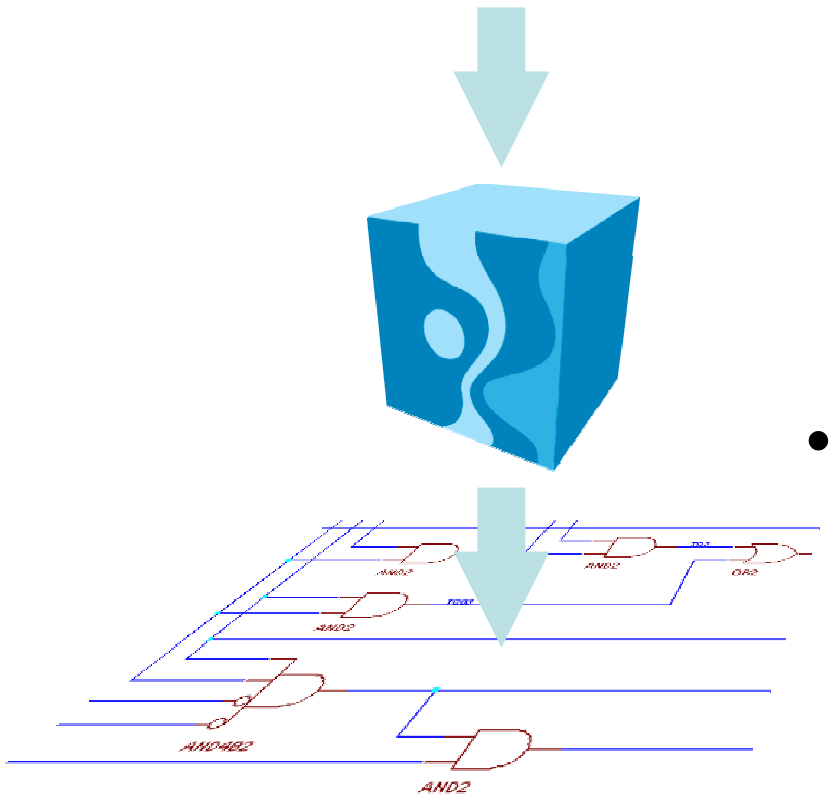
The Mitrion Platform

1) The Mitrion Virtual Processor
   - A configurable processor design for a fine-grain massively parallel, soft-core processor
   - 10-30 times faster than traditional CPUs at 20-25 Watts
   - Executes a program in an FPGA
2) The Mitrion-C programming language
   - An intrinsically parallel C-family language
3) The Mitrion Software Development Kit
   - Compiler
   - Debugger/Simulator
   - Processor configurator

mitrionics™

The Mitrion Virtual Processor

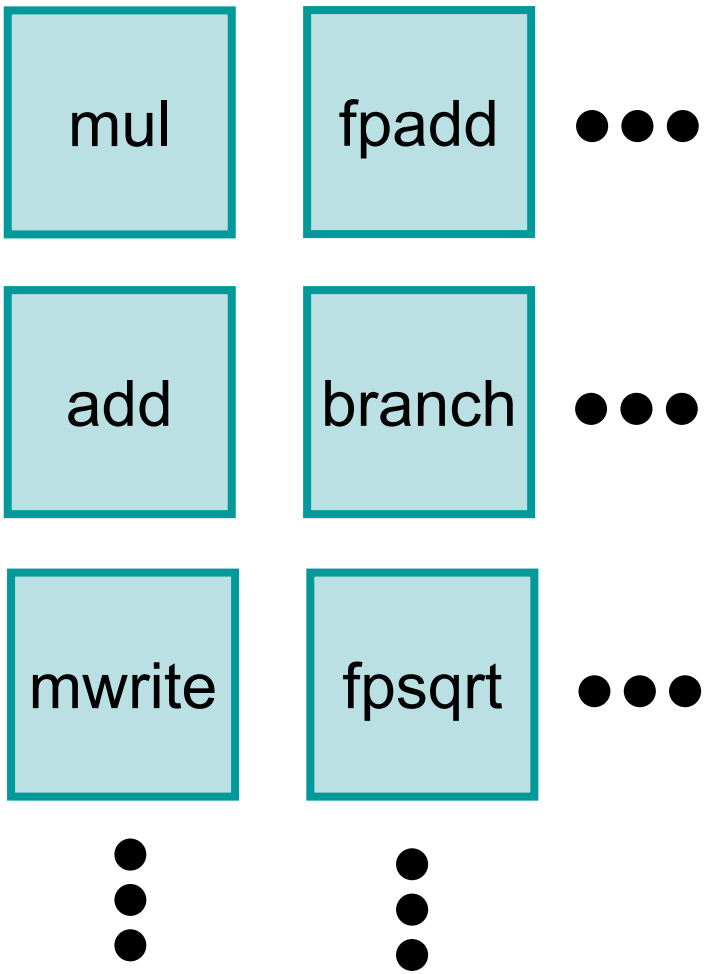
```
int:48<30> main()
{
  int:48 prev  = 1;
  int:48 fib   = 1;

  int:48<30> fibonnacci = for(i in <1..30>)
  {
    fib  = fib+prev;
    prev = fib;
  } <>fib;

} fibonnacci;
```



- Fine-grain parallel
  - Instruction level parallel
- A configurable processor design
  - The processing elements suitable for the application
  - Processing elements adapted to bit-width
- Allows software programming of FPGAs
  - You program the MVP, rather than design HW

mitrionics™

The Mitrion Virtual Processor



A set of pre-designed and verified hardware tiles

- Roughly 60 different kinds of tiles
- Arithmetic-logic and I/O tiles
- Control flow tiles

All tiles can connect to each other

- Tiles will function correctly regardless of configuration

Tile-set is Turing complete

- There is always a configuration of tiles that perform any program you write

Adaptable register and bus widths

- Tiles function correctly from 1-64 bits

mitrionics™

Mitrion Offers Portability And Scalability

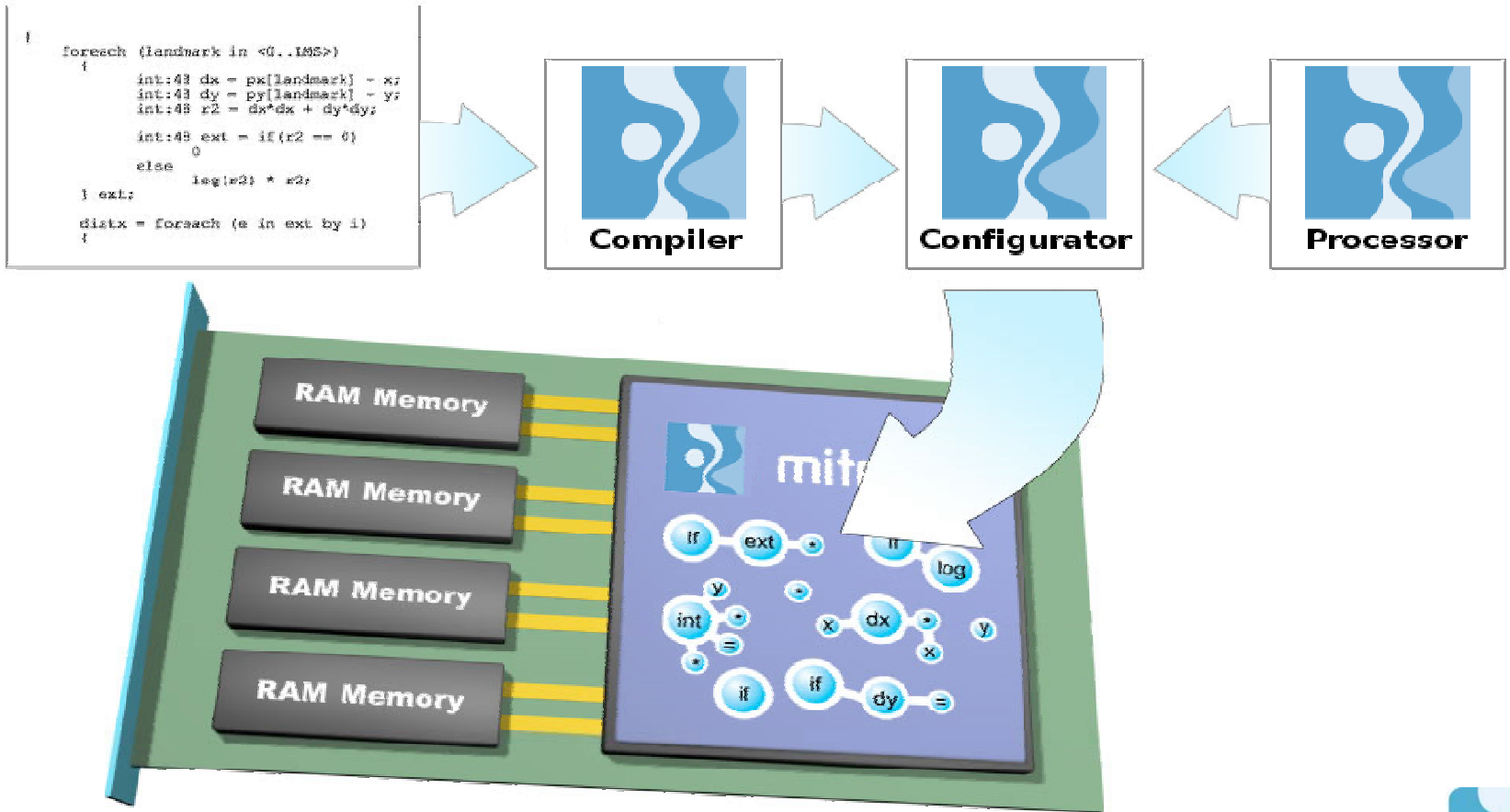
- You program the processor, you do not design a circuit for a specific FPGA platform

Just configure a processor for the new platform from your old source-code.

- Easy upgrades to the next generation of performance available.
- Exchange code with colleagues on other platforms

mitrionics™

The Mitrion Platform



mitrionics™

Programming the Mitrion Virtual Processor

Mitrion-C

- The MVP needs a fine-grain, fully parallel programming language
  - Parallel at the level of individual instructions
- A new C-family language - not ANSI-C!
  - Similar to C about as much as Java is similar to C
  - The code has to be re-written for parallelism anyway
- Allows a complete data-dependency graph to be created of the algorithm
  - Full representation of all parallelism in the algorithm

mitrionics™

Mitrion-C

- Implicitly parallel
  - Describes *data dependencies* rather than order-of-execution
  - Allows fine-grain parallelism to be described
- Syntactic support to make design-decisions regarding parallelism salient
  - Designed to *preserve* parallelism in the algorithm
  - Designed to *reveal* parallelism in the algorithm
- Defined behaviour
  - Parallelism is an integrated part of the language, not an optimization or a language extension

Preserving parallelism

- Data dependencies, not order of execution

```
x = a + b;
y = c + d;
Z = x * a;
```

Preserving Parallelism

- Non-strict execution

```
int:8 a; int:8 b; int:8 c;
x = f(a, b, c);

int:8<100> a; int:8<100> b;
(x, y) = f(a, b);
z = g(x, y);
```

mitrionics™

Revealing parallelism

- Loop syntax reveals loop dependencies

```
int:8 s = 0;
x = for(e in v)
{
      s = s + e;
} s;
```

```
x = foreach(e,f in u, v)
{
      s = e * f;
} s;
```

mitrionics™

Specifying parallelism

- Type system controls kind of parallelism

```
int:8<100> a;          // Pipelined execution

int:8[100] a;          // Unrolled execution

mem int:8[100] a;      // Random access
                          sequential
```

mitrionics™

Thank You!

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

stefan@mitrionics.com

mitrionics™

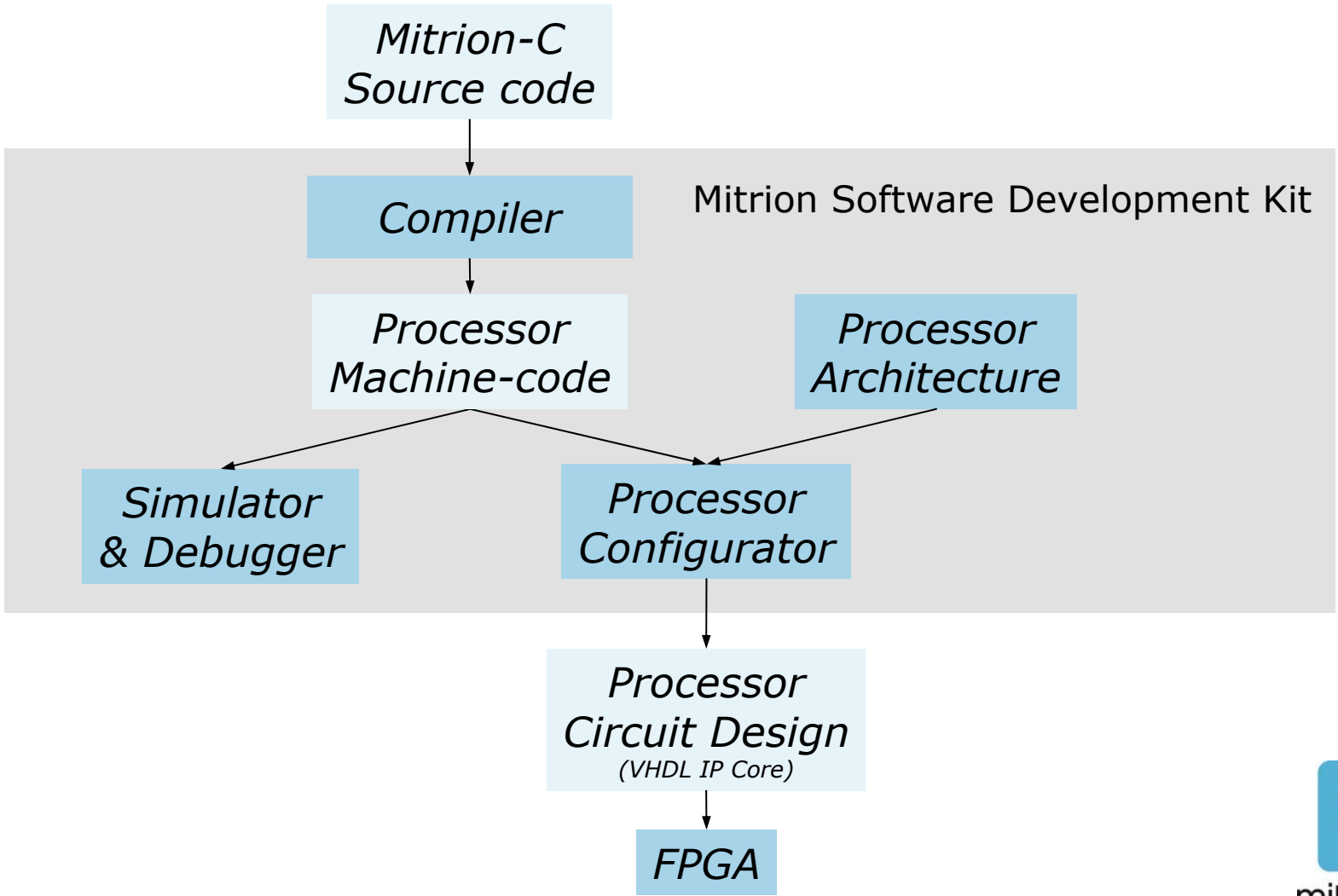
Applying the Mitrion Platform

1) Identify the time-critical portion of the program
2) Write it as code in Mitrion-C
3) Perform a function call to run your Mitrion-program

- Most of your code is unchanged

mitrionics™

Compiling A Mitrion-C Program



mitrionics™