Porting EvtGen to the Intel MIC Architecture

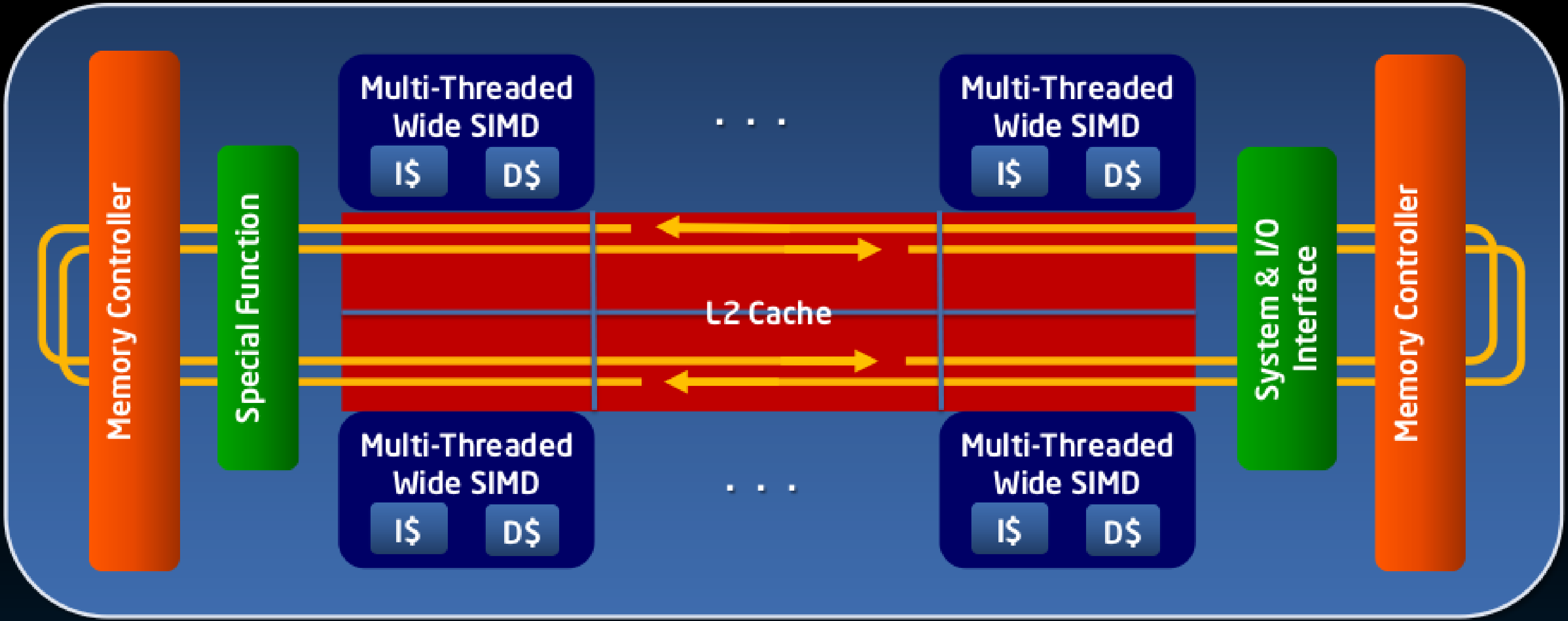
Francesco Giacomini – INFN-CNAF

3rd SuperB Collaboration Meeting
Laboratori Nazionali di Frascati – 19-23 March 2012

Goal

- **M**any-**I**ntegrated **C**ore Architecture
- Play with the MIC
  - Contribution to the INFN COKA project (**CO**mputation on **K**nigths **A**rchitecture)
- Target is part of EvtGen
  - EvtBtoXsgammaKagan::computeHadronicMass()
  - Complement work done by S. Longo
    - Parallelization and Legacy code: a preliminary work on EvtGen
    - Possibility to compare results in the future
- Longer-term goal is to integrate the possibility to offload computation to an accelerator (such as a MIC or a GPU) directly in the Framework

Intel® MIC Architecture – Knights Family



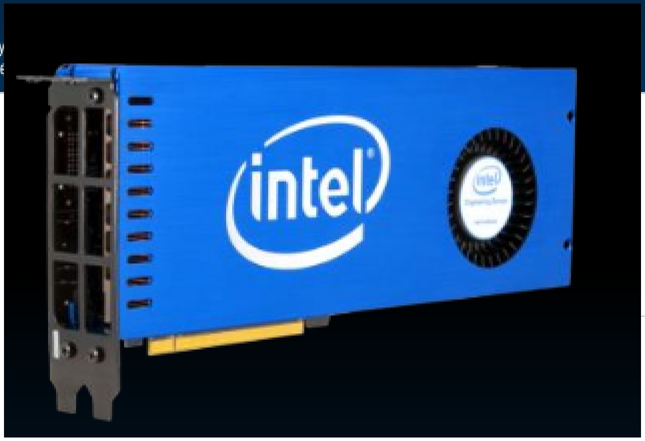
Multiple IA cores
- In-order, short pipeline
- Multi-thread support

16-wide vector units (512b)
- Extended instruction set
Fully coherent caches

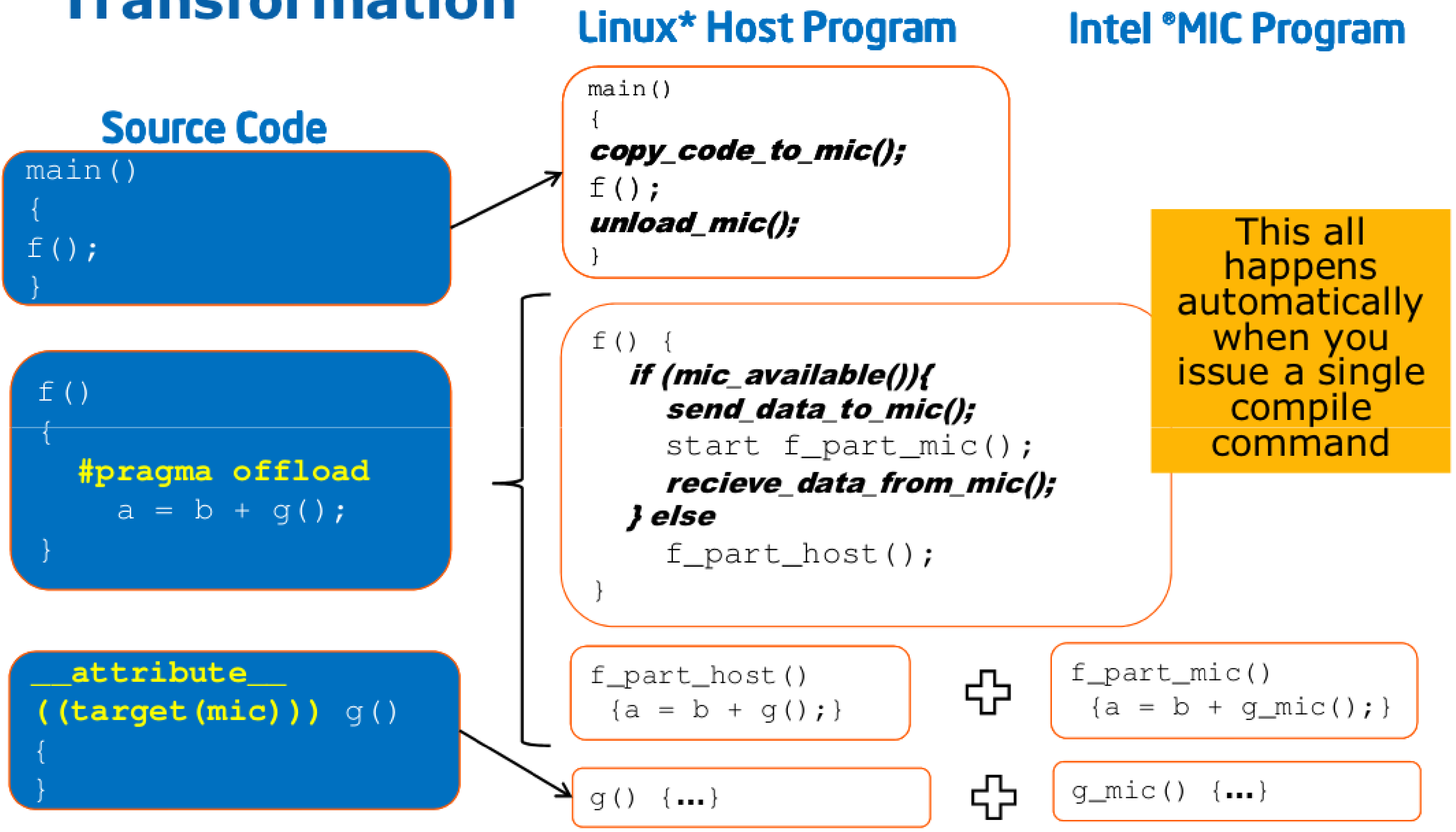
1024-bit ring bus
GDDR5 memory
- Supports virtual memory

Standard IA Shared Memory Programming

For illustration only.
Future options subject to change without notice.



Heterogeneous Compiler – Conceptual Transformation



Source Code

```
main()
{
f();
}
```

```
f()
{
  #pragma offload
     a = b + g();
}
```

```
__attribute__
((target(mic))) g()
{
}
```

Linux* Host Program

```
main()
{
copy_code_to_mic();
f();
unload_mic();
}
```

```
f() {
   if (mic_available()){
      send_data_to_mic();
      start f_part_mic();
      recieve_data_from_mic();
   } else
      f_part_host();
}
```

```
f_part_host()
 {a = b + g();}
```

```
g() {…}
```

Intel ®MIC Program

```
f_part_mic()
 {a = b + g_mic();}
```

```
g_mic() {…}
```

This all happens automatically when you issue a single compile command

Current status

- The code has been heavily refactored to make it more parallel-friendly
  - Value-based, possibly const
    - Use only stack-based objects/variables, i.e. no pointers
  - A lot less sharing between loop iterations
- Just add a pragma in front of the main loop to enable parallelism with OpenMP

```
  //Calculating the Branching Fractions
#pragma omp parallel for
  for (i=0 ; i < int(_nIntervalmH + 1.0); i++) {
    // …
  }
```

Next steps

- MIC-specific modifications to the code
  - Not clear yet how intrusive the modifications will be

```
  //Calculating the Branching Fractions
#pragma offload target(mic)
#pragma omp parallel for
  for (i=0 ; i < int(_nIntervalmH + 1.0); i++) {
    // …
  }
```

- Access to a MIC and to the Intel compiler to test the changes and make measurements
- Further code restructuring to replace runtime-polymorphism (i.e. inheritence) with static-polymorphism (i.e. templates)