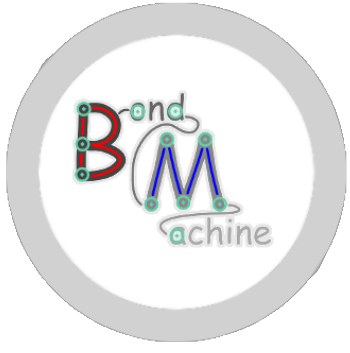
The BondMachine Toolkit

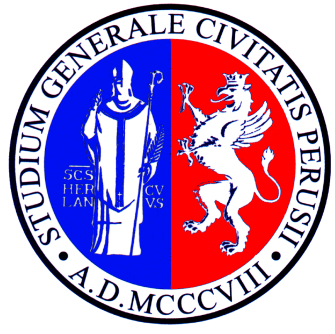
Enabling Machine Learning on FPGA

Mirko Mariotti

Department of Physics and Geology - University of Perugia
INFN Perugia

International Symposium on Grids & Clouds 2019 (ISGC 2019)
April 5, 2019
Academia Sinica - Taipei (TW)





Introduction

The BondMachine Toolkit: Enabling Machine Learning on FPGA

In this presentation i will talk about:

- Technological background of the project.
- The BondMachine Project: architecture and tools.
- BondMachine for Machine Learning.
- Building accelerators and their use on the Cloud.
- Conclusion.

FPGA

What is it ?

- A field-programmable gate array (FPGA) is an integrated circuit whose logic is re-programmable. It's used to build reconfigurable digital circuits.
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together".

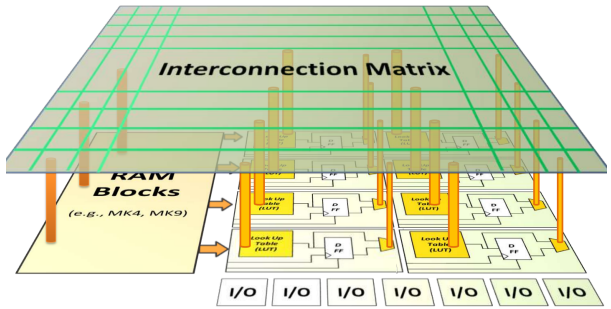
- Logic blocks can be configured to perform complex combinational functions.

- The FPGA configuration is generally specified using a hardware description language (HDL).

FPGA

What is it ?

- A field-programmable gate array (FPGA) is an integrated circuit whose logic is re-programmable. It's used to build reconfigurable digital circuits.
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together".

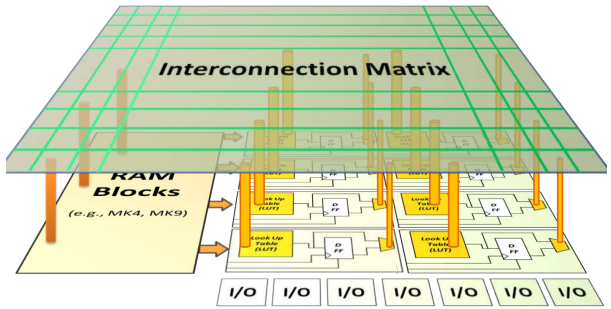


- Logic blocks can be configured to perform complex combinational functions.
- The FPGA configuration is generally specified using a hardware description language (HDL).

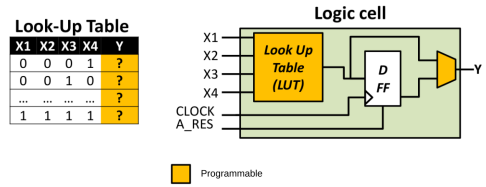
FPGA

What is it ?

- A field-programmable gate array (FPGA) is an integrated circuit whose logic is re-programmable. It's used to build reconfigurable digital circuits.
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together".



- Logic blocks can be configured to perform complex combinational functions.

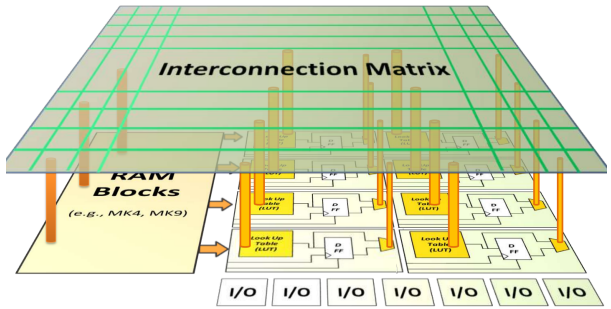


- The FPGA configuration is generally specified using a hardware description language (HDL).

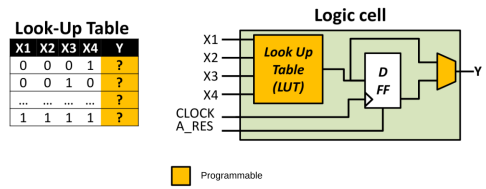
FPGA

What is it ?

- A field-programmable gate array (FPGA) is an integrated circuit whose logic is re-programmable. It's used to build reconfigurable digital circuits.
- FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together".



- Logic blocks can be configured to perform complex combinational functions.



- The FPGA configuration is generally specified using a hardware description language (HDL).

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

Computer Architectures

Multi-core and Heterogeneous

Today's computer architecture are:

- Multi-core,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number of cores.
  - Parallelism has to be addressed.
- Heterogeneous, different types of processing units.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - The units data transfer has to be addressed.
  - The scheduling has to be addressed.

The BondMachine

The idea

High level sources: Go, TensorFlow, NN

Building a new kind of computer architecture (multi-core and heterogeneous both in cores types and interconnections) which dynamically adapt to the specific computational problem rather than be static.

BM architecture Layer

FPGA

Concurrency and Specialization

The BondMachine

The idea

High level sources: Go, TensorFlow, NN

Building a new kind of computer architecture (multi-core and heterogeneous both in cores types and interconnections) which dynamically adapt to the specific computational problem rather than be static.

BM architecture Layer

FPGA

Concurrency and Specialization

The BondMachine

The idea

High level sources: Go, TensorFlow, NN

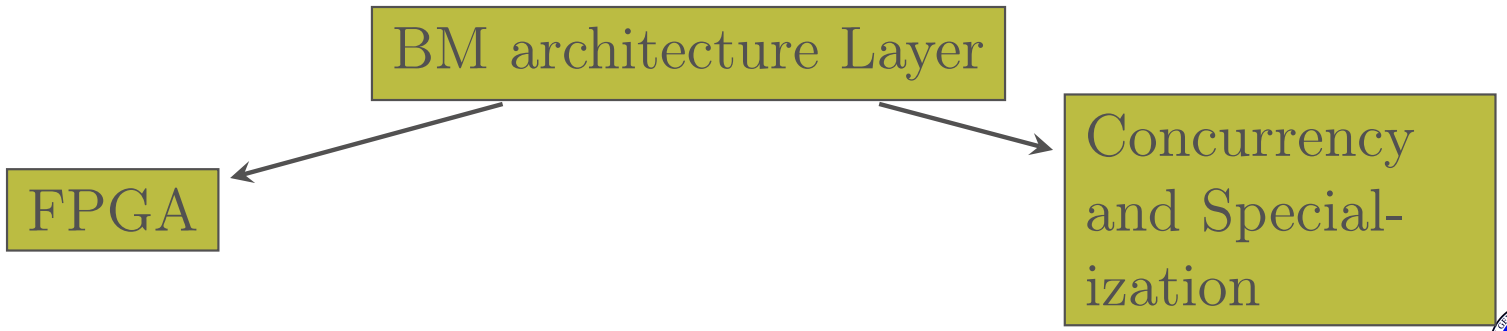
Building a new kind of computer architecture (multi-core and heterogeneous both in cores types and interconnections) which dynamically adapt to the specific computational problem rather than be static.

The BondMachine

The idea

High level sources: Go, TensorFlow, NN

Building a new kind of computer architecture (multi-core and heterogeneous both in cores types and interconnections) which dynamically adapt to the specific computational problem rather than be static.



The BondMachine

The idea

High level sources: Go, TensorFlow, NN

Building a new kind of computer architecture (multi-core and heterogeneous both in cores types and interconnections) which dynamically adapt to the specific computational problem rather than be static.

BM architecture Layer

FPGA

Concurrency and Specialization

Introducing the BondMachine (BM)

The BondMachine is a software ecosystem for the dynamic generation of computer architectures that:

- Are composed by many, possibly hundreds, computing cores.
- Have very small cores and not necessarily of the same type (different ISA and ABI).
- Have a not fixed way of interconnecting cores.
- May have some elements shared among cores (for example channels and shared memories).

Introducing the BondMachine (BM)

The BondMachine is a software ecosystem for the dynamic generation of computer architectures that:

- Are composed by many, possibly hundreds, computing cores.
- Have very small cores and not necessarily of the same type (different ISA and ABI).
- Have a not fixed way of interconnecting cores.
- May have some elements shared among cores (for example channels and shared memories).

Introducing the BondMachine (BM)

The BondMachine is a software ecosystem for the dynamic generation of computer architectures that:

- Are composed by many, possibly hundreds, computing cores.
- Have very small cores and not necessarily of the same type (different ISA and ABI).
- Have a not fixed way of interconnecting cores.
- May have some elements shared among cores (for example channels and shared memories).

Introducing the BondMachine (BM)

The BondMachine is a software ecosystem for the dynamic generation of computer architectures that:

- Are composed by many, possibly hundreds, computing cores.
- Have very small cores and not necessarily of the same type (different ISA and ABI).
- Have a not fixed way of interconnecting cores.
- May have some elements shared among cores (for example channels and shared memories).

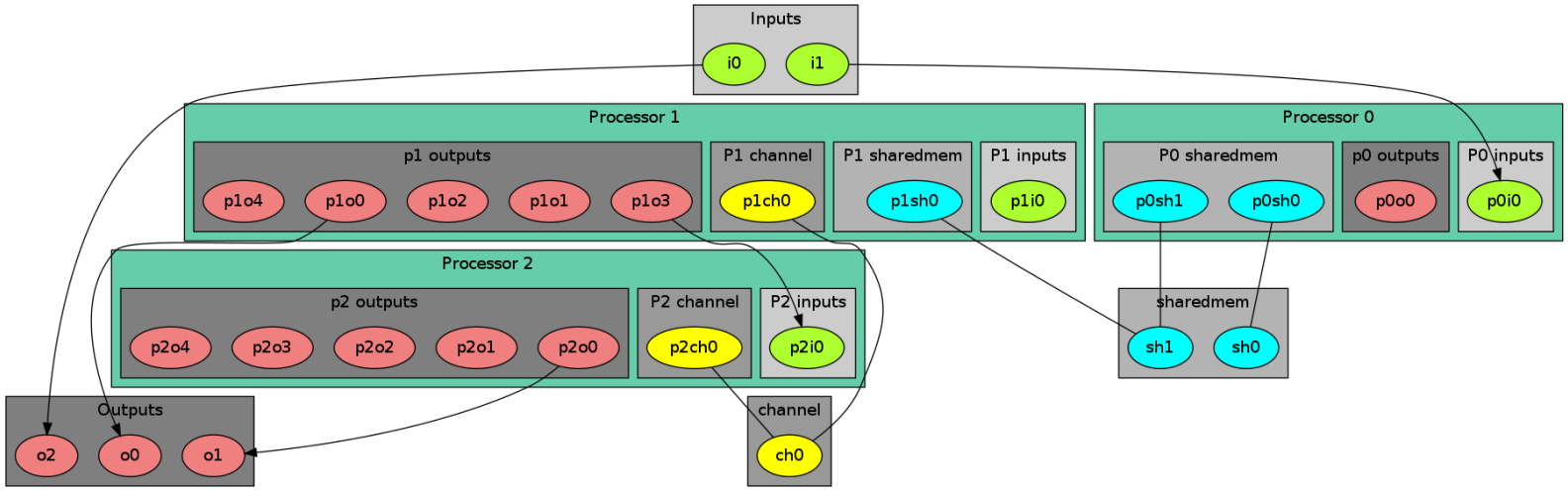
Introducing the BondMachine (BM)

The BondMachine is a software ecosystem for the dynamic generation of computer architectures that:

- Are composed by many, possibly hundreds, computing cores.
- Have very small cores and not necessarily of the same type (different ISA and ABI).
- Have a not fixed way of interconnecting cores.
- May have some elements shared among cores (for example channels and shared memories).

The BondMachine

An example



Connecting Processor (CP)

The computational unit of the BM

The atomic computational unit of a BM is the "connecting processor" (CP) and has:

- Some general purpose registers of size Rsize.
- Some I/O dedicated registers of size Rsize.
- A set of implemented opcodes chosen among many available.
- Dedicated ROM and RAM.
- There possible operating modes.

Connecting Processor (CP)

The computational unit of the BM

The atomic computational unit of a BM is the "connecting processor" (CP) and has:

- Some general purpose registers of size Rsize.
- Some I/O dedicated registers of size Rsize.
- A set of implemented opcodes chosen among many available.
- Dedicated ROM and RAM.
- There possible operating modes.

Connecting Processor (CP)

The computational unit of the BM

The atomic computational unit of a BM is the "connecting processor" (CP) and has:

- Some general purpose registers of size Rsize.
- Some I/O dedicated registers of size Rsize.
- A set of implemented opcodes chosen among many available.
- Dedicated ROM and RAM.
- There possible operating modes.

Connecting Processor (CP)

The computational unit of the BM

The atomic computational unit of a BM is the "connecting processor" (CP) and has:

- Some general purpose registers of size Rsize.
- Some I/O dedicated registers of size Rsize.
- A set of implemented opcodes chosen among many available.
- Dedicated ROM and RAM.
- There possible operating modes.

Connecting Processor (CP)

The computational unit of the BM

The atomic computational unit of a BM is the "connecting processor" (CP) and has:

- Some general purpose registers of size Rsize.
- Some I/O dedicated registers of size Rsize.
- A set of implemented opcodes chosen among many available.
- Dedicated ROM and RAM.
- There possible operating modes.

Connecting Processor (CP)

The computational unit of the BM

The atomic computational unit of a BM is the "connecting processor" (CP) and has:

- Some general purpose registers of size Rsize.
- Some I/O dedicated registers of size Rsize.
- A set of implemented opcodes chosen among many available.
- Dedicated ROM and RAM.
- There possible operating modes.

Shared Objects (SO)

The non-computational element of the BM

Alongside CPs, BondMachines include non-computing units called "Shared Objects" (SO).

Examples of their purposes are:

- Data storage (Memories).
- Message passing.
- CP synchronization.

A single SO can be shared among different CPs. To use it CPs have special instructions (opcodes) oriented to the specific SO.

Four kind of SO have been developed so far: the Channel, the Shared Memory, the Barrier and a Pseudo Random Numbers Generator.

Shared Objects (SO)

The non-computational element of the BM

Alongside CPs, BondMachines include non-computing units called "Shared Objects" (SO).

Examples of their purposes are:

- Data storage (Memories).
- Message passing.
- CP synchronization.

A single SO can be shared among different CPs. To use it CPs have special instructions (opcodes) oriented to the specific SO.

Four kind of SO have been developed so far: the Channel, the Shared Memory, the Barrier and a Pseudo Random Numbers Generator.

Shared Objects (SO)

The non-computational element of the BM

Alongside CPs, BondMachines include non-computing units called "Shared Objects" (SO).

Examples of their purposes are:

- Data storage (Memories).
- Message passing.
- CP synchronization.

A single SO can be shared among different CPs. To use it CPs have special instructions (opcodes) oriented to the specific SO.

Four kind of SO have been developed so far: the Channel, the Shared Memory, the Barrier and a Pseudo Random Numbers Generator.

Shared Objects (SO)

The non-computational element of the BM

Alongside CPs, BondMachines include non-computing units called "Shared Objects" (SO).

Examples of their purposes are:

- Data storage (Memories).
- Message passing.
- CP synchronization.

A single SO can be shared among different CPs. To use it CPs have special instructions (opcodes) oriented to the specific SO.

Four kind of SO have been developed so far: the Channel, the Shared Memory, the Barrier and a Pseudo Random Numbers Generator.

Shared Objects (SO)

The non-computational element of the BM

Alongside CPs, BondMachines include non-computing units called "Shared Objects" (SO).

Examples of their purposes are:

- Data storage (Memories).
- Message passing.
- CP synchronization.

A single SO can be shared among different CPs. To use it CPs have special instructions (opcodes) oriented to the specific SO.

Four kind of SO have been developed so far: the Channel, the Shared Memory, the Barrier and a Pseudo Random Numbers Generator.

Shared Objects (SO)

The non-computational element of the BM

Alongside CPs, BondMachines include non-computing units called "Shared Objects" (SO).

Examples of their purposes are:

- Data storage (Memories).
- Message passing.
- CP synchronization.

A single SO can be shared among different CPs. To use it CPs have special instructions (opcodes) oriented to the specific SO.

Four kind of SO have been developed so far: the Channel, the Shared Memory, the Barrier and a Pseudo Random Numbers Generator.

Handle the BM computer architecture

The BM computer architecture is managed by a set of tools to:

- build a specify architecture
- modify a pre-existing architecture
- simulate or emulate the behavior
- Generate the Register Tranfer Code (RTL)

Processor Builder

Selects the single processor, assembles and disassembles, saves on disk as JSON, creates the RTL code of a CP

BondMachine Builder

Connects CPs and SOs together in custom topologies, loads and saves on disk as JSON, create BM's RTL code

Simulation Framework

Simulates the behaviour, emulates a BM on a standard Linux workstation

Handle the BM computer architecture

The BM computer architecture is managed by a set of tools to:

- build a specify architecture
- modify a pre-existing architecture
- simulate or emulate the behavior
- Generate the Register Tranfer Code (RTL)

Processor Builder

Selects the single processor, assembles and disassembles, saves on disk as JSON, creates the RTL code of a CP

BondMachine Builder

Connects CPs and SOs together in custom topologies, loads and saves on disk as JSON, create BM's RTL code

Simulation Framework

Simulates the behaviour, emulates a BM on a standard Linux workstation

Handle the BM computer architecture

The BM computer architecture is managed by a set of tools to:

- build a specify architecture
- modify a pre-existing architecture
- simulate or emulate the behavior
- Generate the Register Tranfer Code (RTL)

Processor Builder

Selects the single processor, assembles and disassembles, saves on disk as JSON, creates the RTL code of a CP

BondMachine Builder

Connects CPs and SOs together in custom topologies, loads and saves on disk as JSON, create BM's RTL code

Simulation Framework

Simulates the behaviour, emulates a BM on a standard Linux workstation

Handle the BM computer architecture

The BM computer architecture is managed by a set of tools to:

- build a specify architecture
- modify a pre-existing architecture
- simulate or emulate the behavior
- Generate the Register Tranfer Code (RTL)

Processor Builder

Selects the single processor, assembles and disassembles, saves on disk as JSON, creates the RTL code of a CP

BondMachine Builder

Connects CPs and SOs together in custom topologies, loads and saves on disk as JSON, create BM's RTL code

Simulation Framework

Simulates the behaviour, emulates a BM on a standard Linux workstation

Use the BM computer architecture

Mapping specific computational problems to BMs

Symbond

Map symbolic mathematical expressions to BM

Boolbond

Map boolean systems to BM

Matrixwork

Basic matrix computation

Evolutive BM

Evolutionary computing to BM

Neuralbond

Map neural networks to BM

tf2bm & nnef2bm

Map computational graphs to BM

Use the BM computer architecture

Mapping specific computational problems to BMs

Symbond

Map symbolic mathematical expressions to BM

Boolbond

Map boolean systems to BM

Matrixwork

Basic matrix computation

Evolutive BM

Evolutionary computing to BM

Neuralbond

Map neural networks to BM

tf2bm & nnef2bm

Map computational graphs to BM

Use the BM computer architecture

Mapping specific computational problems to BMs

Symbond

Map symbolic mathematical expressions to BM

Boolbond

Map boolean systems to BM

Matrixwork

Basic matrix computation

Evolutive BM

Evolutionary computing to BM

Neuralbond

Map neural networks to BM

tf2bm & nnef2bm

Map computational graphs to BM

Use the BM computer architecture

Mapping specific computational problems to BMs

Symbond

Map symbolic mathematical expressions to BM

Boolbond

Map boolean systems to BM

Matrixwork

Basic matrix computation

Evolutive BM

Evolutionary computing to BM

Neuralbond

Map neural networks to BM

tf2bm & nnef2bm

Map computational graphs to BM

Use the BM computer architecture

Mapping specific computational problems to BMs

Symbond

Map symbolic mathematical expressions to BM

Boolbond

Map boolean systems to BM

Matrixwork

Basic matrix computation

Evolutive BM

Evolutionary computing to BM

Neuralbond

Map neural networks to BM

tf2bm & nnef2bm

Map computational graphs to BM

Use the BM computer architecture

Mapping specific computational problems to BMs

Symbond

Map symbolic mathematical expressions to BM

Boolbond

Map boolean systems to BM

Matrixwork

Basic matrix computation

Evolutive BM

Evolutionary computing to BM

Neuralbond

Map neural networks to BM

tf2bm & nnef2bm

Map computational graphs to BM

Use the BM computer architecture

Mapping specific computational problems to BMs

Symbond

Map symbolic mathematical expressions to BM

Boolbond

Map boolean systems to BM

Matrixwork

Basic matrix computation

Evolutive BM

Evolutionary computing to BM

Neuralbond

Map neural networks to BM

tf2bm & nnef2bm

Map computational graphs to BM

Bondgo

The major innovation of the BondMachine Project is its compiler.

Bondgo is the name chosen for the compiler developed for the BondMachine.

The compiler source language is Go as the name suggest.

Bondgo

Bondgo does something different from standard compilers ...

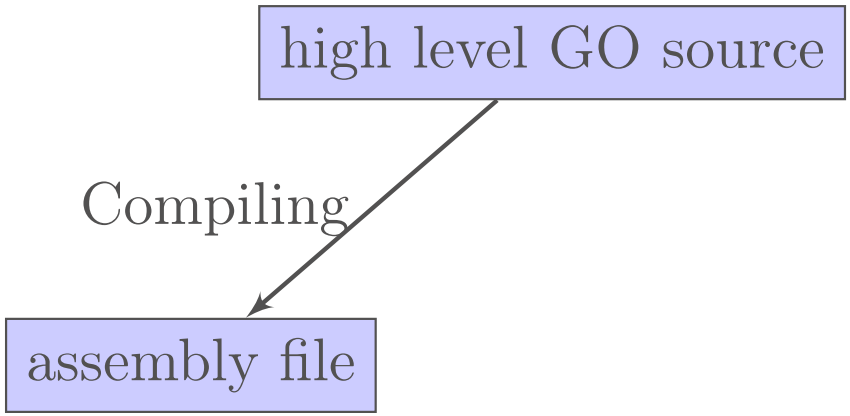
Bondgo

Bondgo does something different from standard compilers ...

high level GO source

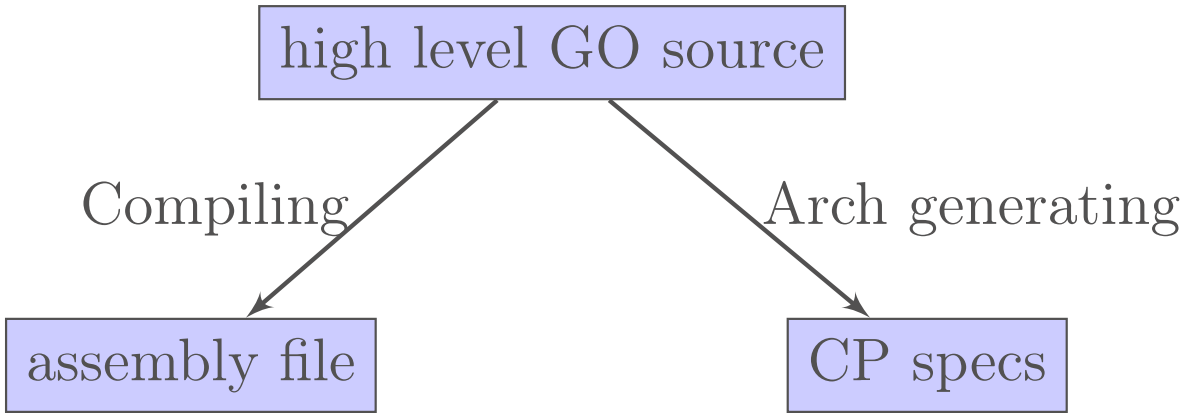
Bondgo

Bondgo does something different from standard compilers ...



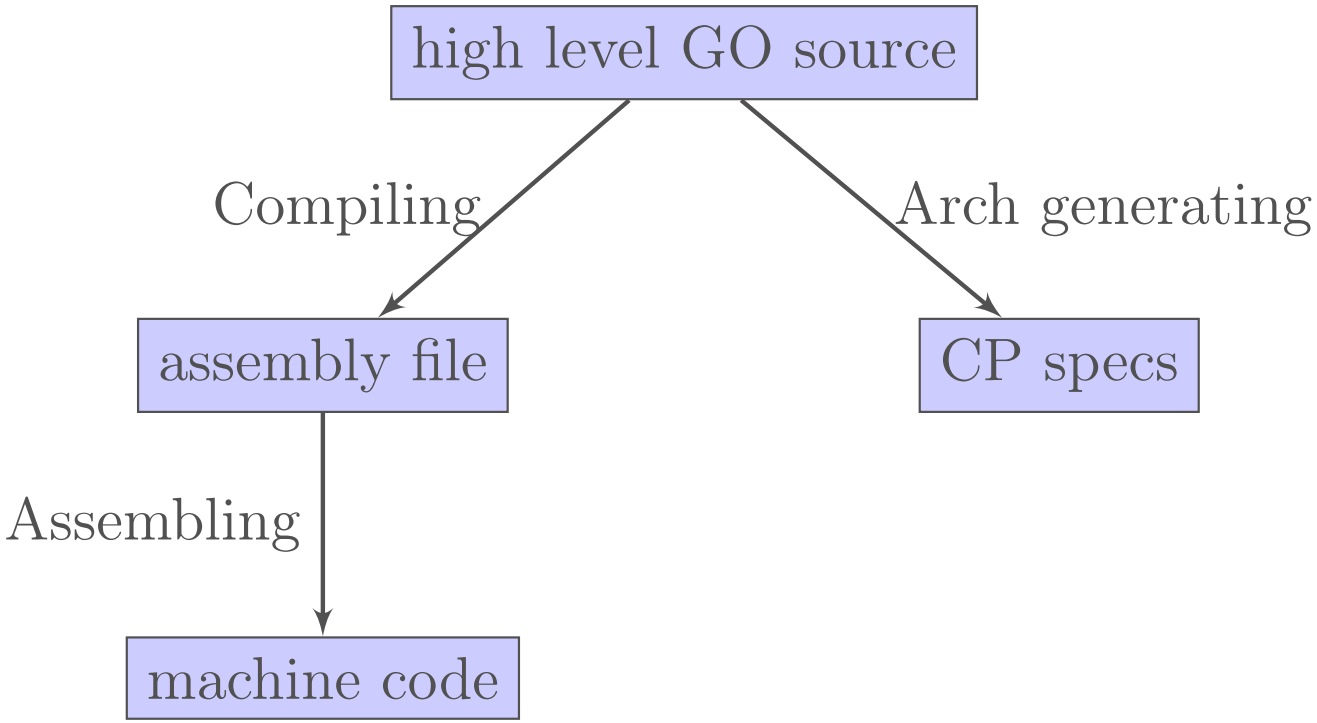
Bondgo

Bondgo does something different from standard compilers ...



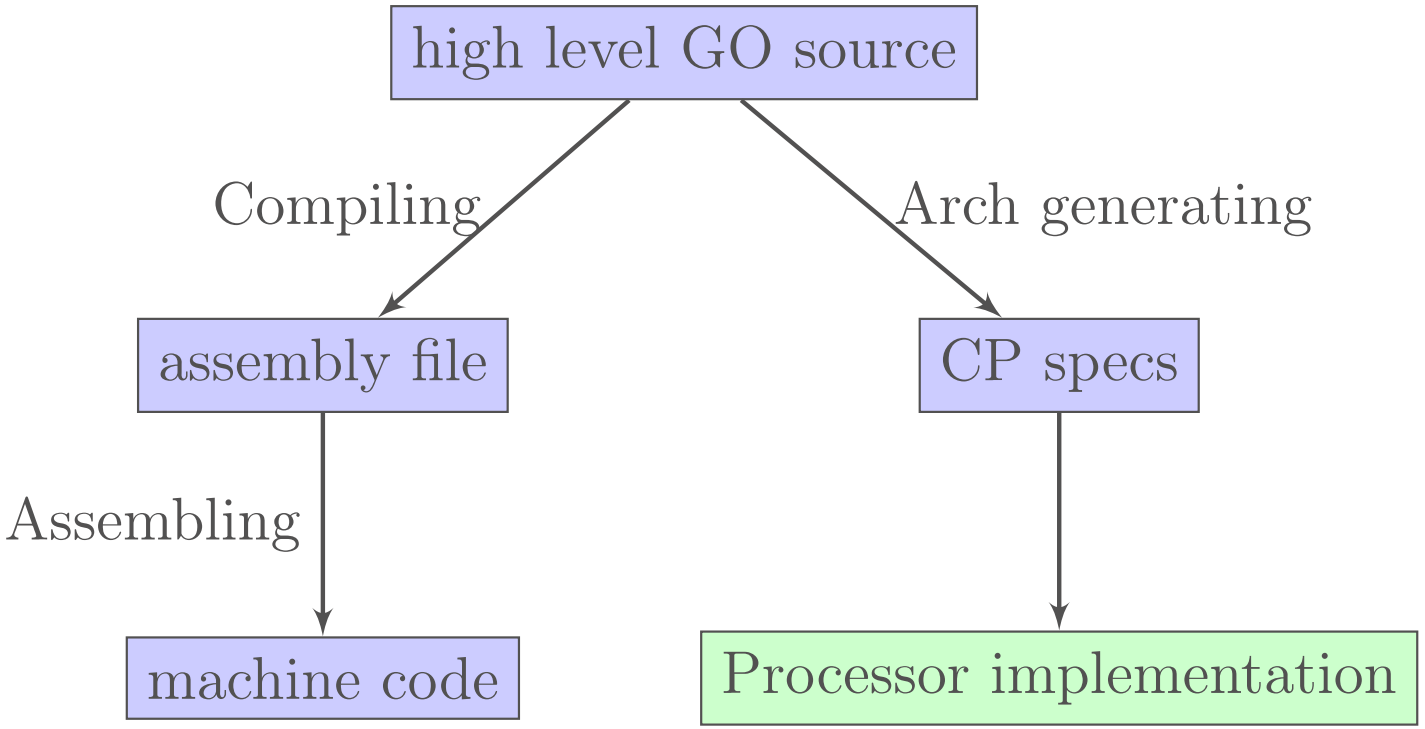
Bondgo

Bondgo does something different from standard compilers ...



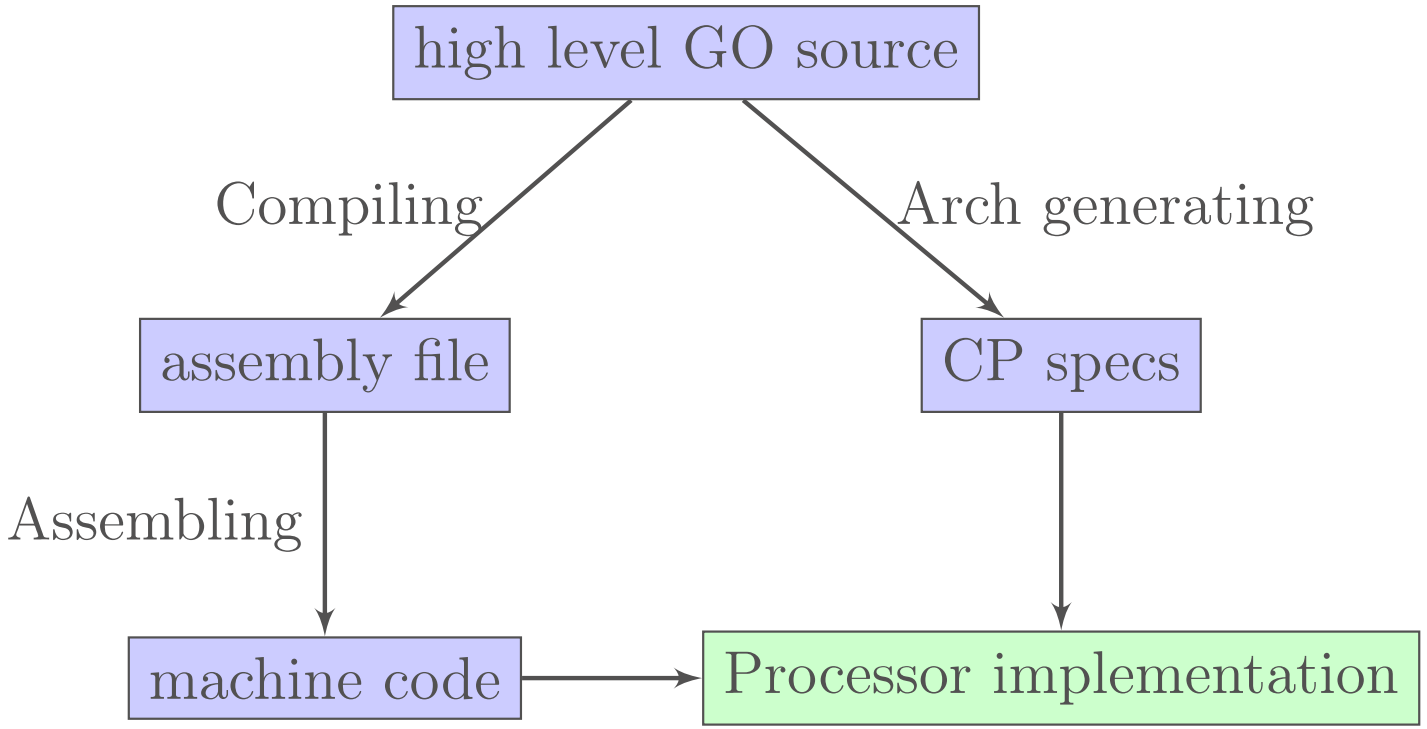
Bondgo

Bondgo does something different from standard compilers ...



Bondgo

Bondgo does something different from standard compilers ...



Bondgo

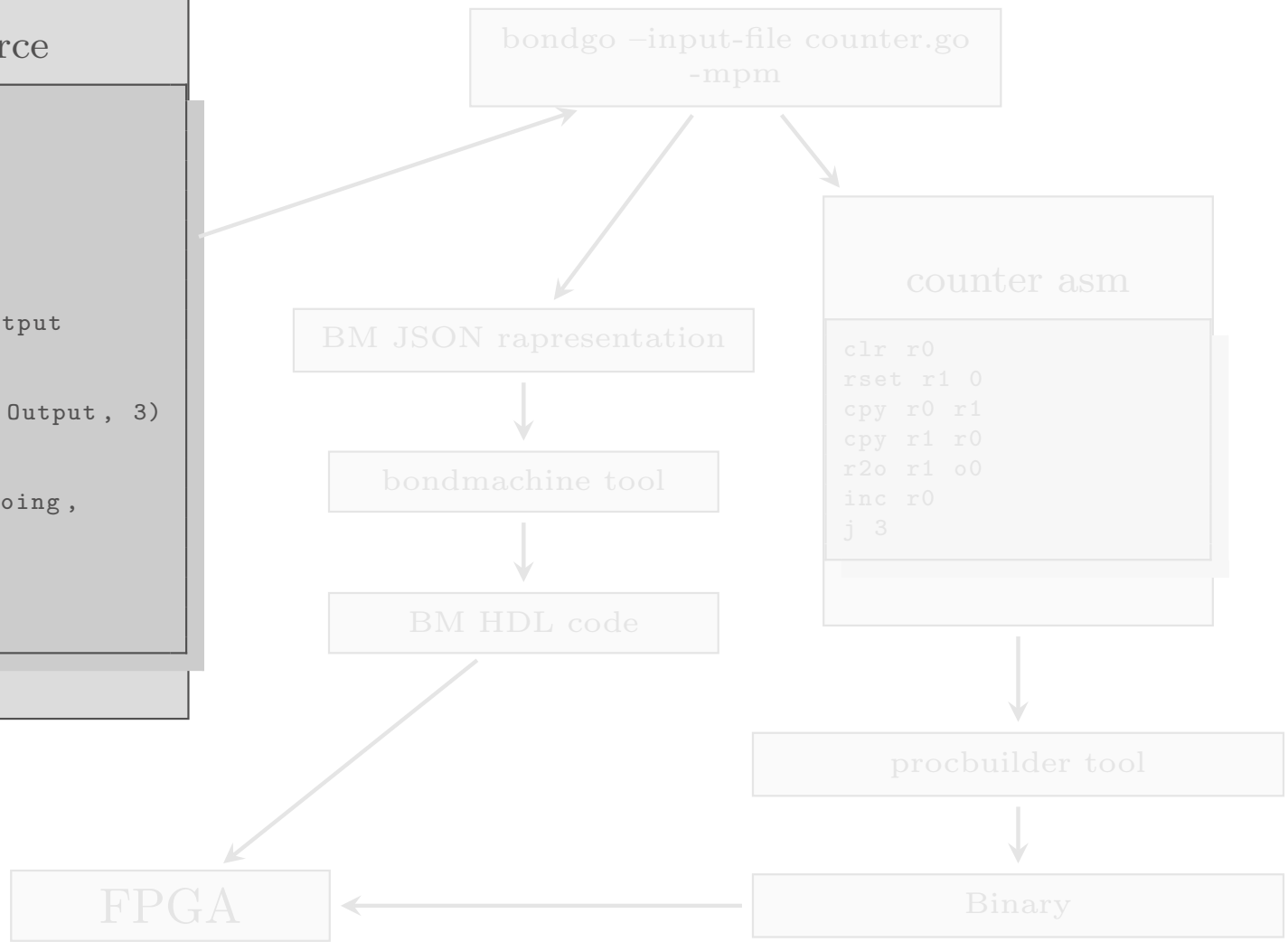
A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```



Bondgo

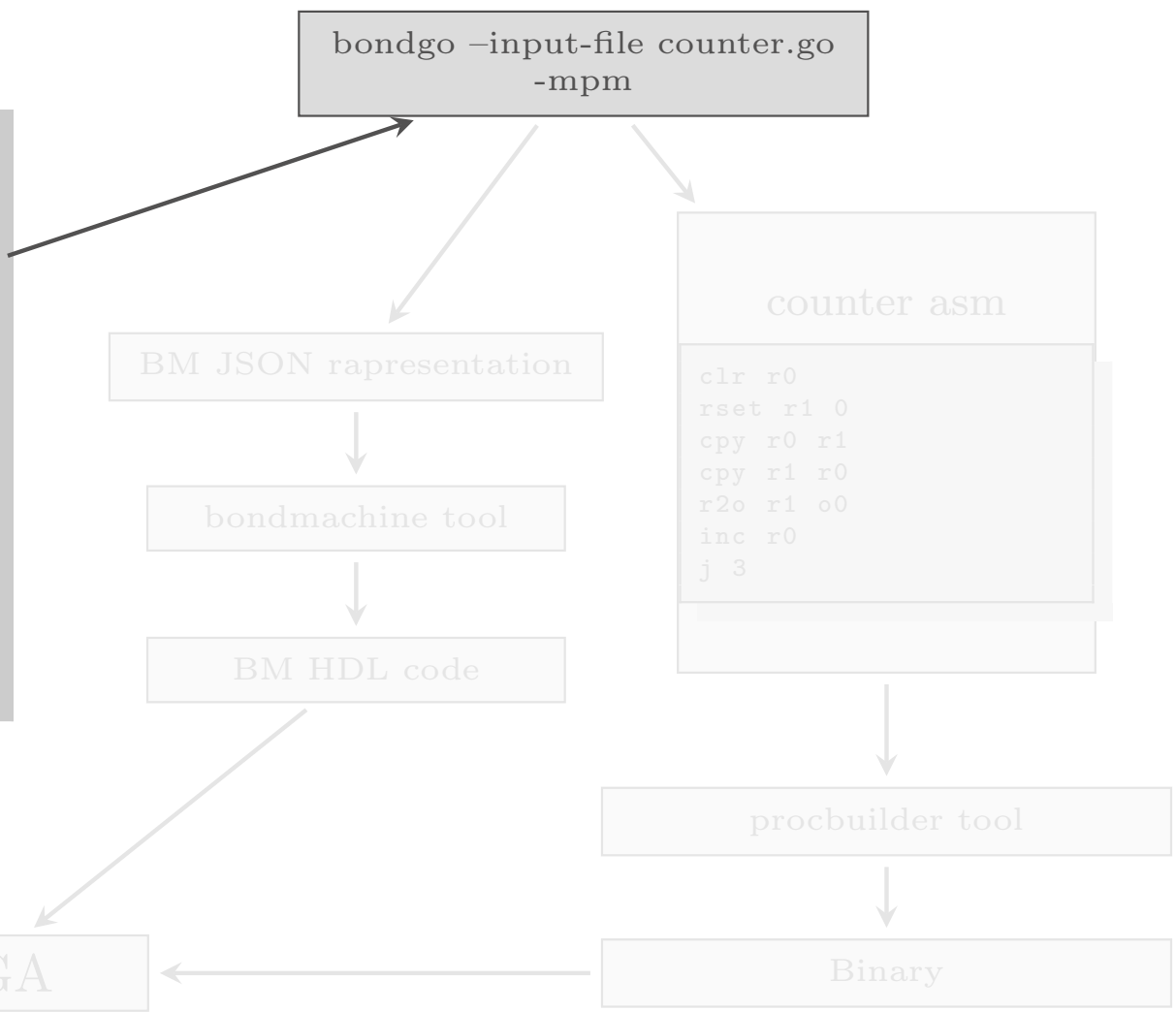
A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```



Bondgo

A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```

bondgo –input-file counter.go -mpm

BM JSON rapresentation

bondmachine tool

BM HDL code

counter asm

```
clr r0
rset r1 0
cpy r0 r1
cpy r1 r0
r2o r1 o0
inc r0
j 3
```

procbuilder tool

Binary

FPGA

Bondgo

A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```

bondgo –input-file counter.go -mpm

BM JSON rapresentation

bondmachine tool

BM HDL code

FPGA

counter asm

```
clr r0
rset r1 0
cpy r0 r1
cpy r1 r0
r2o r1 o0
inc r0
j 3
```

procbuilder tool

Binary

Bondgo

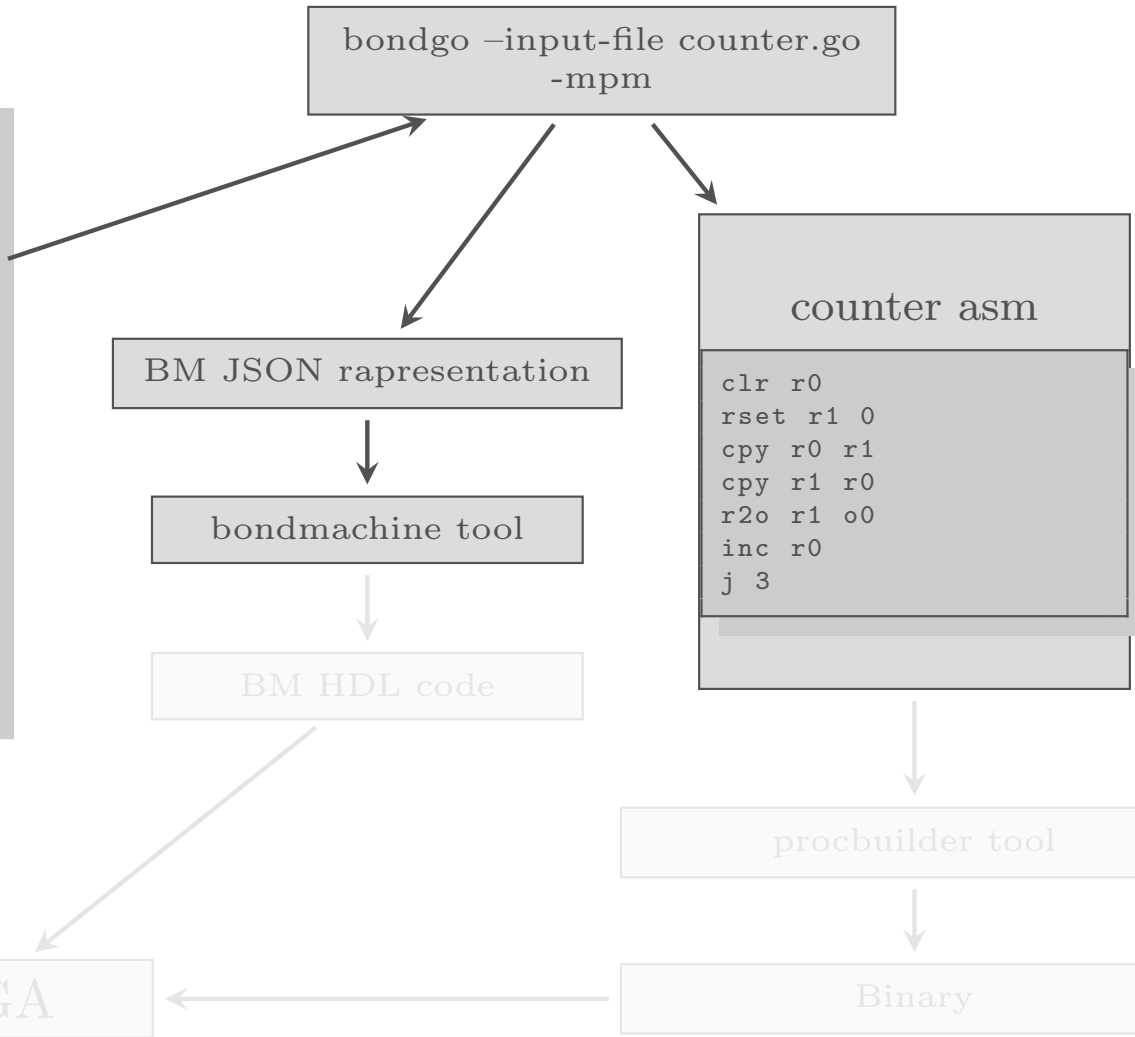
A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```



Bondgo

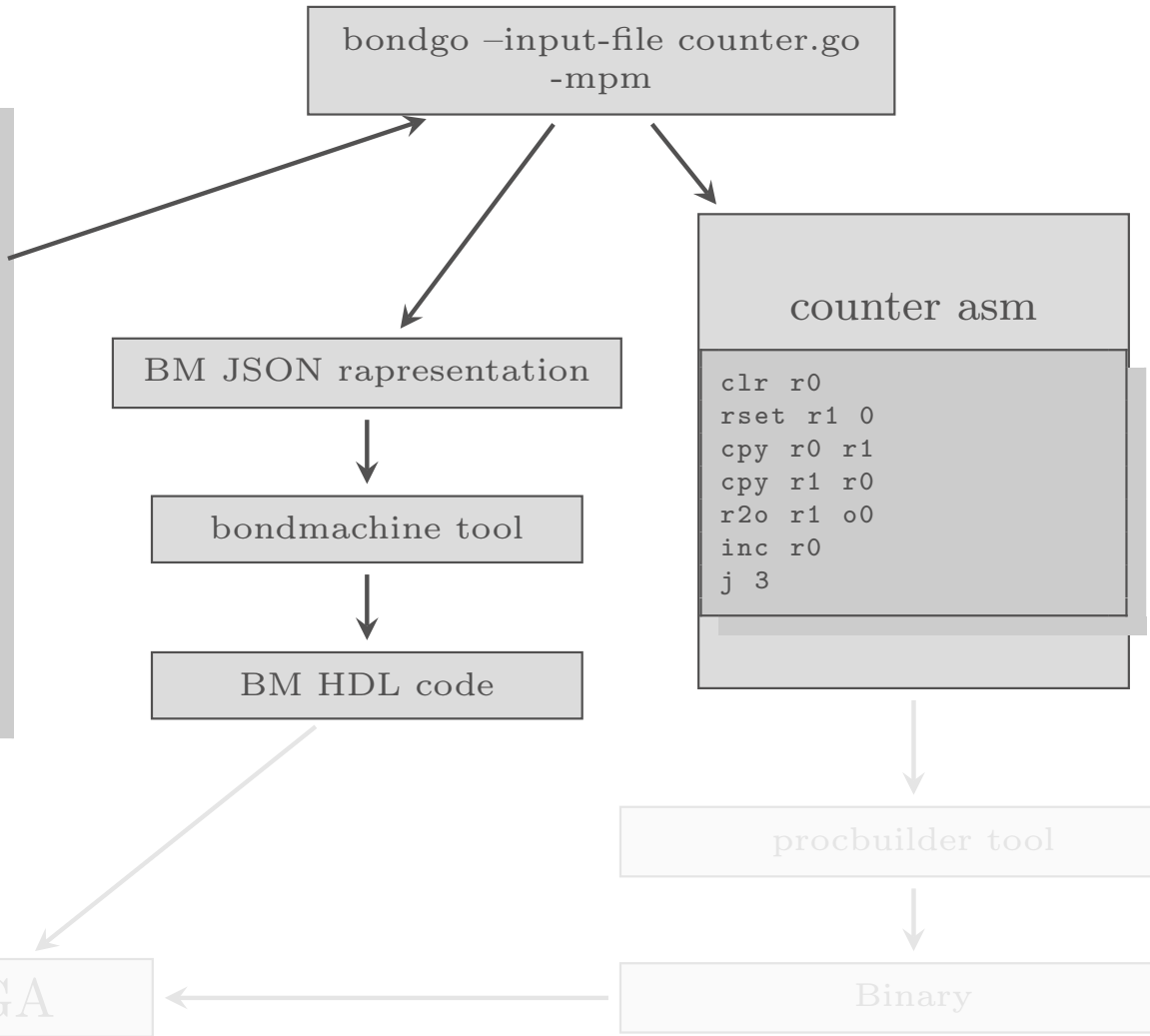
A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```



Bondgo

A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```

bondgo –input-file counter.go -mpm

BM JSON rapresentation

bondmachine tool

BM HDL code

FPGA

counter asm

```
clr r0
rset r1 0
cpy r0 r1
cpy r1 r0
r2o r1 o0
inc r0
j 3
```

procbuilder tool

Binary

Bondgo

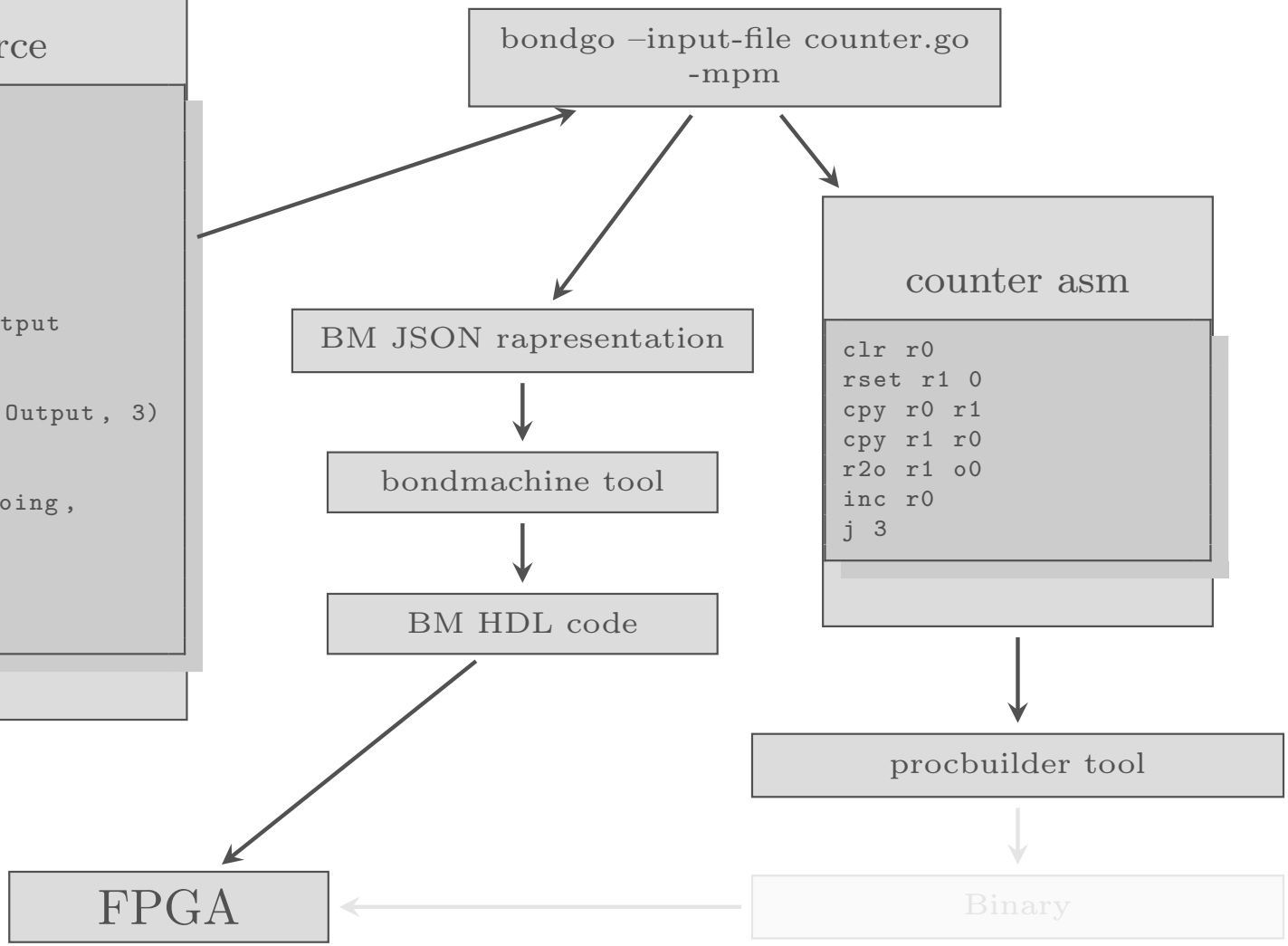
A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```



Bondgo

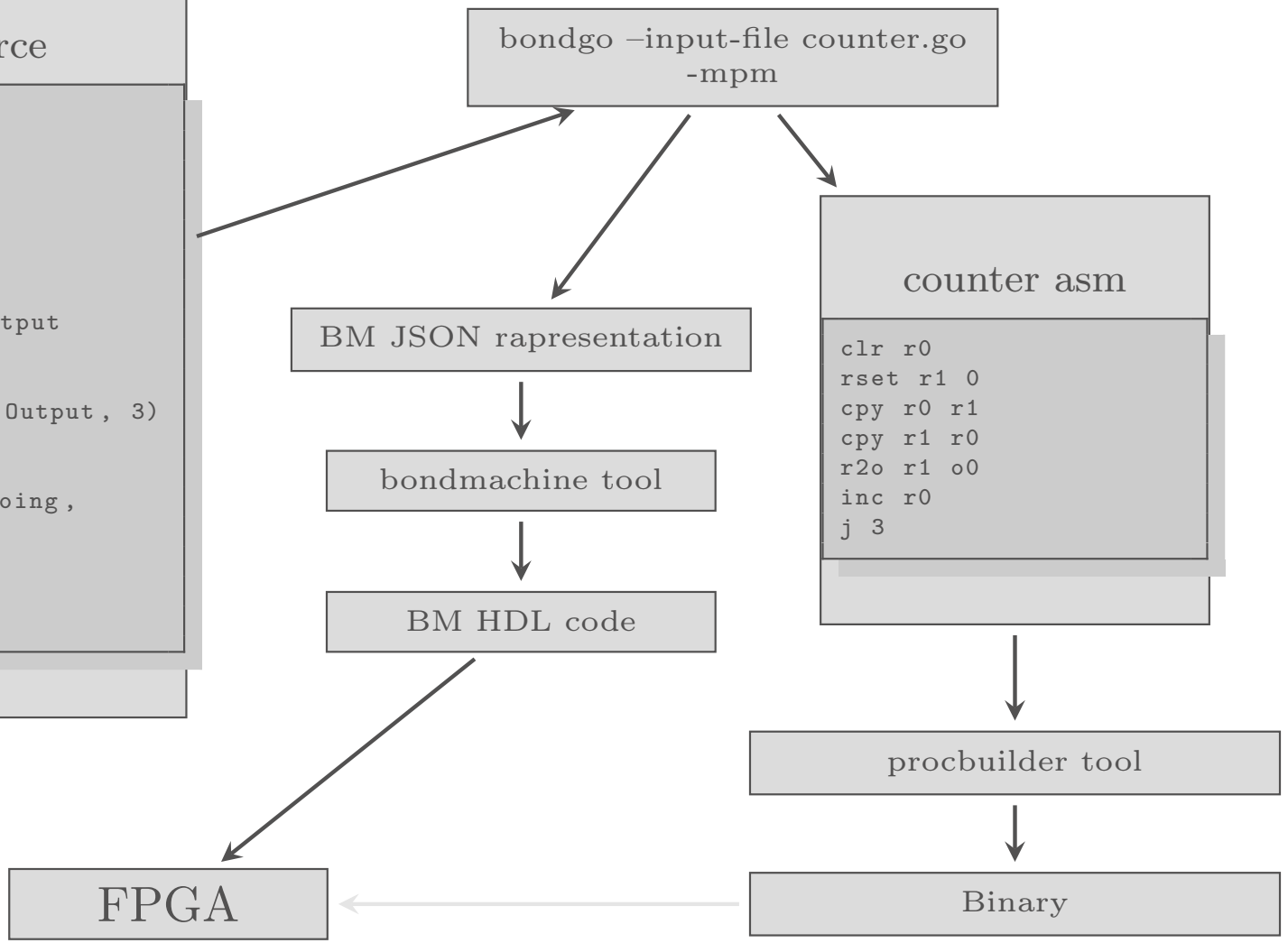
A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```



Bondgo

A first example

counter go source

```
package main

import (
   "bondgo"
)

func main() {
   var outgoing bondgo.Output
   var reg_int uint8
   outgoing =
      bondgo.Make(bondgo.Output, 3)
   reg_int = 0
   for {
      bondgo.IOWrite(outgoing,
      reg_int)
      reg_int++
   }
}
```

bondgo –input-file counter.go -mpm

BM JSON rapresentation

bondmachine tool

BM HDL code

counter asm

```
clr r0
rset r1 0
cpy r0 r1
cpy r1 r0
r2o r1 o0
inc r0
j 3
```

procbuilder tool

Binary

FPGA

Bondgo

... bondgo may not only create the binaries, but also the CP architecture, and ...

Bondgo

... it can do even much more interesting things when compiling concurrent programs.

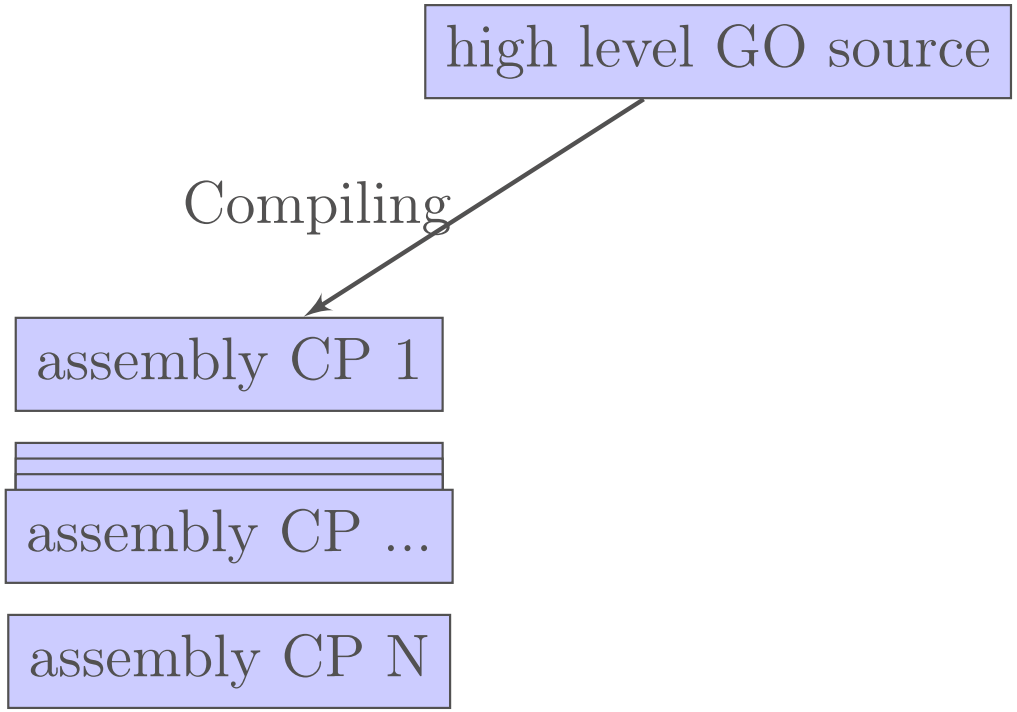
Bondgo

... it can do even much more interesting things when compiling concurrent programs.

high level GO source

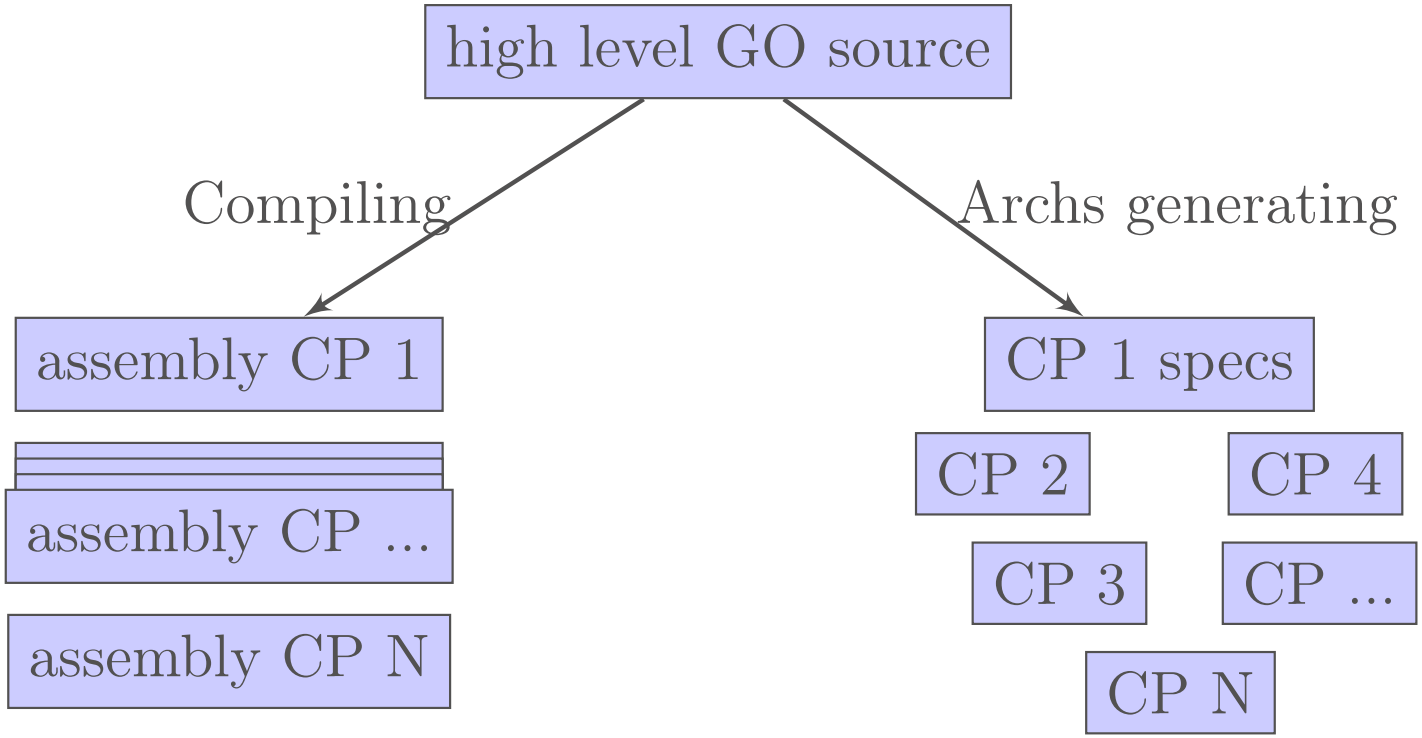
Bondgo

... it can do even much more interesting things when compiling concurrent programs.



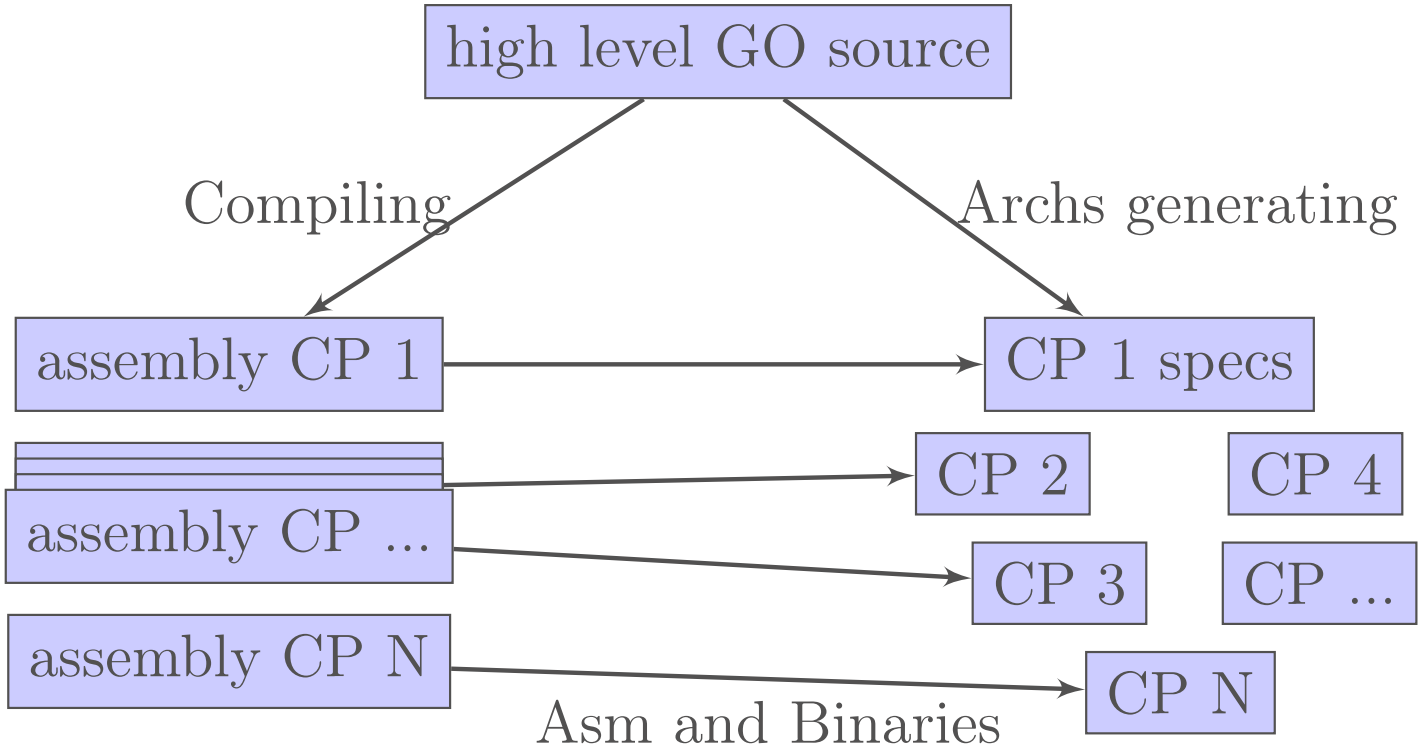
Bondgo

... it can do even much more interesting things when compiling concurrent programs.



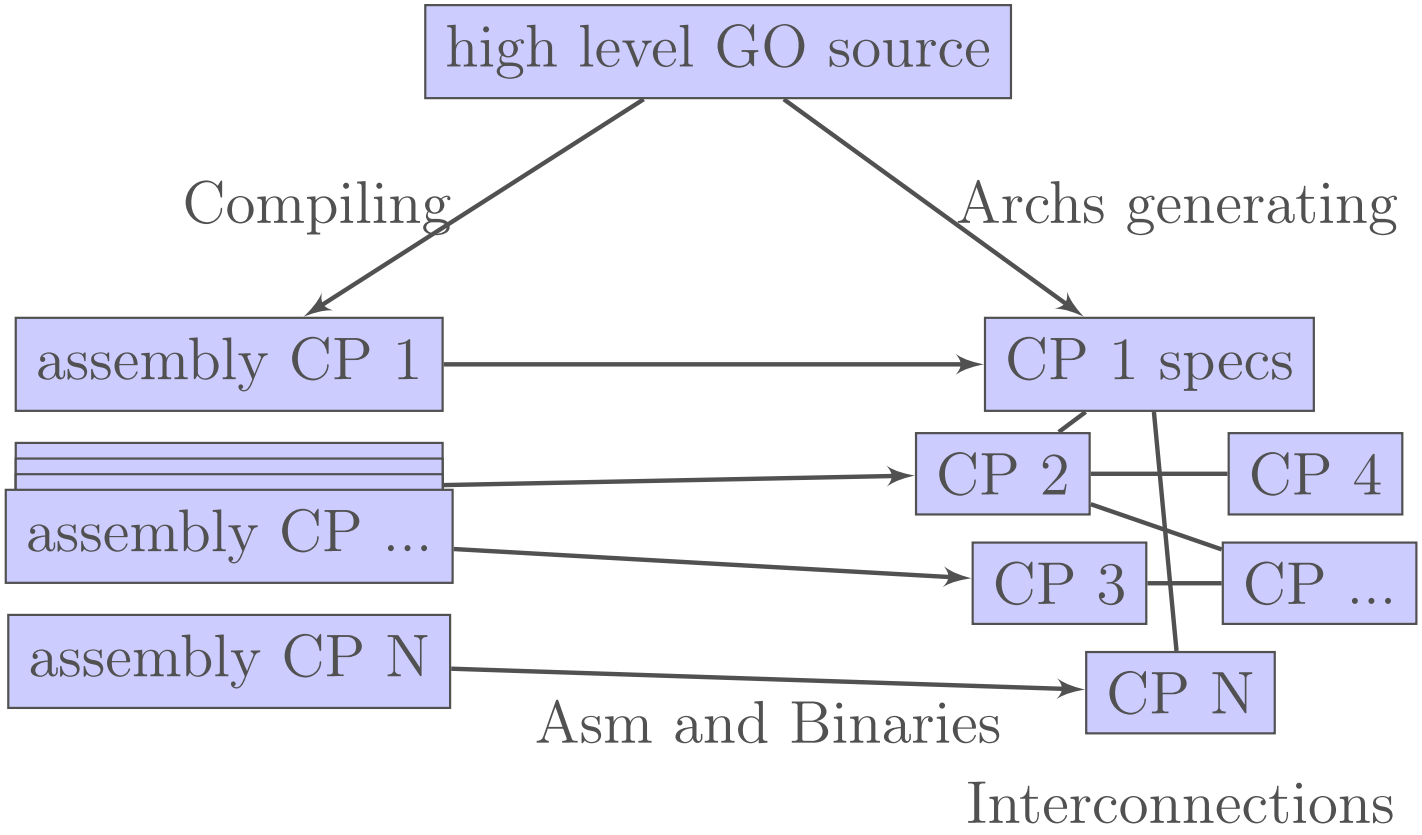
Bondgo

... it can do even much more interesting things when compiling concurrent programs.



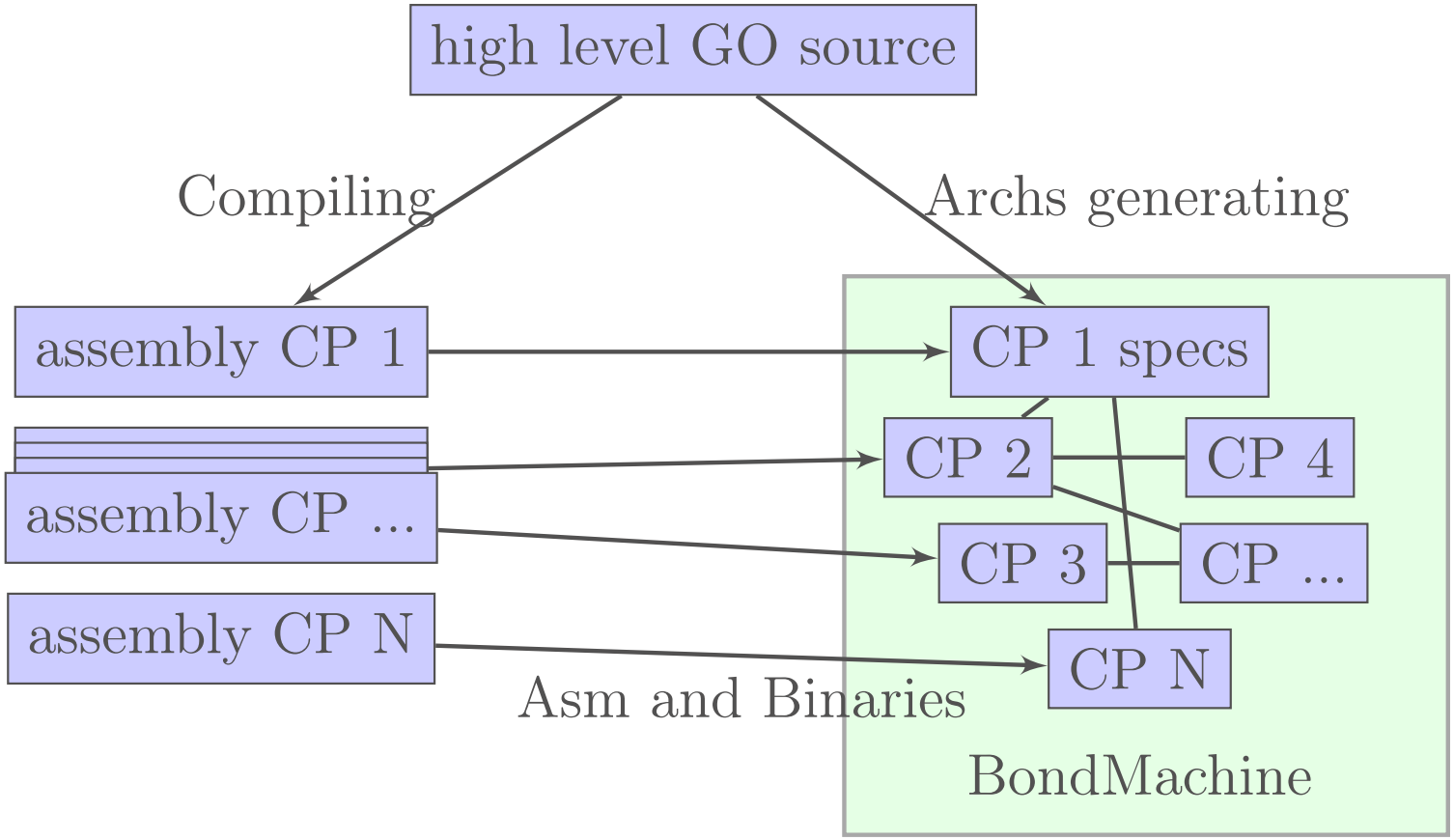
Bondgo

... it can do even much more interesting things when compiling concurrent programs.



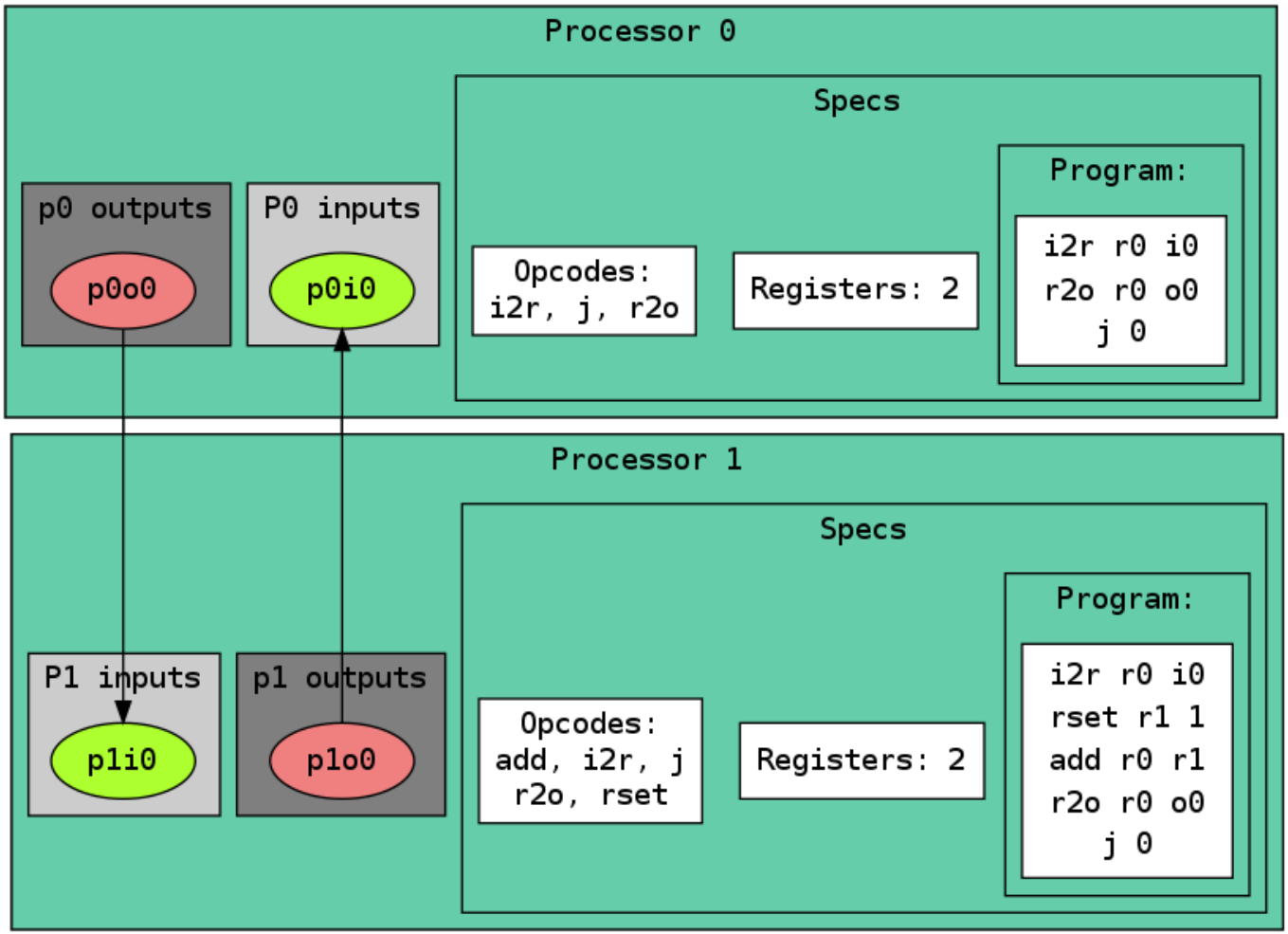
Bondgo

... it can do even much more interesting things when compiling concurrent programs.



Bondgo

A multi-core example



Compiling Architectures

One of the most important result

The architecture creation is a part of the compilation process.

Machine Learning with BondMachine

Architectures with multiple interconnected processors like the ones produced by the BondMachine Toolkit are a perfect fit for Neural Networks and Computational Graphs.

Several ways to map this structures to BondMachine has been developed:

- A native Neural Network library
- A Tensorflow to BondMachine translator
- An NNEF based BondMachine composer

Machine Learning with BondMachine

Architectures with multiple interconnected processors like the ones produced by the BondMachine Toolkit are a perfect fit for Neural Networks and Computational Graphs.

Several ways to map this structures to BondMachine has been developed:

- A native Neural Network library
- A Tensorflow to BondMachine translator
- An NNEF based BondMachine composer

Machine Learning with BondMachine

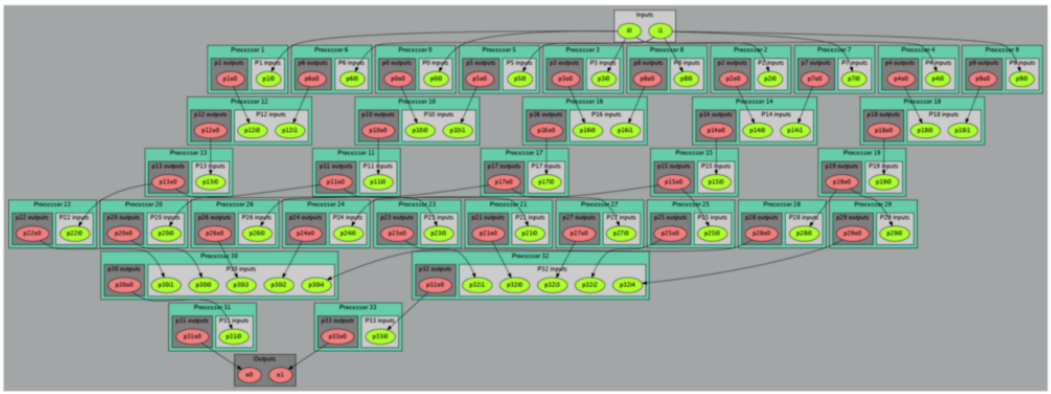
Native Neural Network library

The tool neuralbond allow the creation of BM-based neural chips from an API go interface.

- Neurons are converted to BondMachine connecting processors.
- Tensors are mapped to CP connections.

```
layers := []int{2, 5, 2}
weights := make([]neuralbond.Weight, 0)

if *save_bondmachine != "" {
   if mymachine, ok :=
      neuralbond.Build_MLP(layers, weights); ok
      == nil {
      if _, err := os.Stat(*save_bondmachine);
      os.IsNotExist(err) {
         f, err := os.Create(*save_bondmachine)
         check(err)
         defer f.Close()
      }
   }
}
```

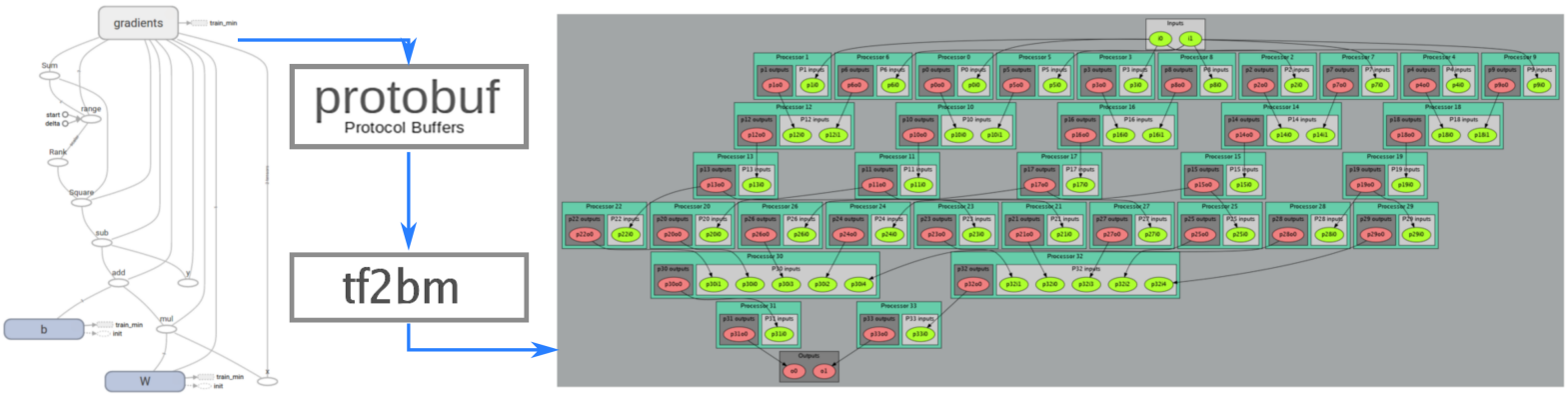


TensorFlow™ to Bondmachine

tf2bm

TensorFlow™ is an open source software library for numerical computation using data flow graphs.

Graphs can be converted to BondMachines with the tf2bm tool.



Machine Learning with BondMachine

NNEF Composer

Neural Network Exchange Format (NNEF) is a standard from Khronos Group to enable the easy transfer of trained networks among frameworks, inference engines and devices

The NNEF BM tool approach is to descent NNEF models and build BondMachine multi-core accordingly

This approch has several advandages over the previous:

- It is not limited to a single framework
- NNEF is a textual file, so no complex operations are needed to read models

BondMachine Clustering

So far we saw:

- An user friendly approach to create processors (single core).
- Optimizing a single device to support intricate computational work-flows (multi-cores) over an heterogeneous layer.

Interconnected BondMachines

What if we could extend the this layer to multiple interconnected devices ?

BondMachine Clustering

So far we saw:

- An user friendly approach to create processors (single core).
- Optimizing a single device to support intricate computational work-flows (multi-cores) over an heterogeneous layer.

Interconnected BondMachines

What if we could extend the this layer to multiple interconnected devices ?

BondMachine Clustering

So far we saw:

- An user friendly approach to create processors (single core).
- Optimizing a single device to support intricate computational work-flows (multi-cores) over an heterogeneous layer.

Interconnected BondMachines

What if we could extend the this layer to multiple interconnected devices ?

BondMachine Clustering

So far we saw:

- An user friendly approach to create processors (single core).
- Optimizing a single device to support intricate computational work-flows (multi-cores) over an heterogeneous layer.

Interconnected BondMachines

What if we could extend the this layer to multiple interconnected devices ?

BondMachine Clustering

The same logic existing among CP have been extended among different BondMachines organized in clusters.

Protocols, one ethernet called etherbond and one using UDP called udpbond have been created for the purpose.

FPGA based BondMachines, standard Linux Workstations, Emulated BondMachines might join a cluster an contribute to a single distributed computational problem.

BondMachine Clustering

The same logic existing among CP have been extended among different BondMachines organized in clusters.

Protocols, one ethernet called etherbond and one using UDP called udpbond have been created for the purpose.

FPGA based BondMachines, standard Linux Workstations, Emulated BondMachines might join a cluster an contribute to a single distributed computational problem.

BondMachine Clustering

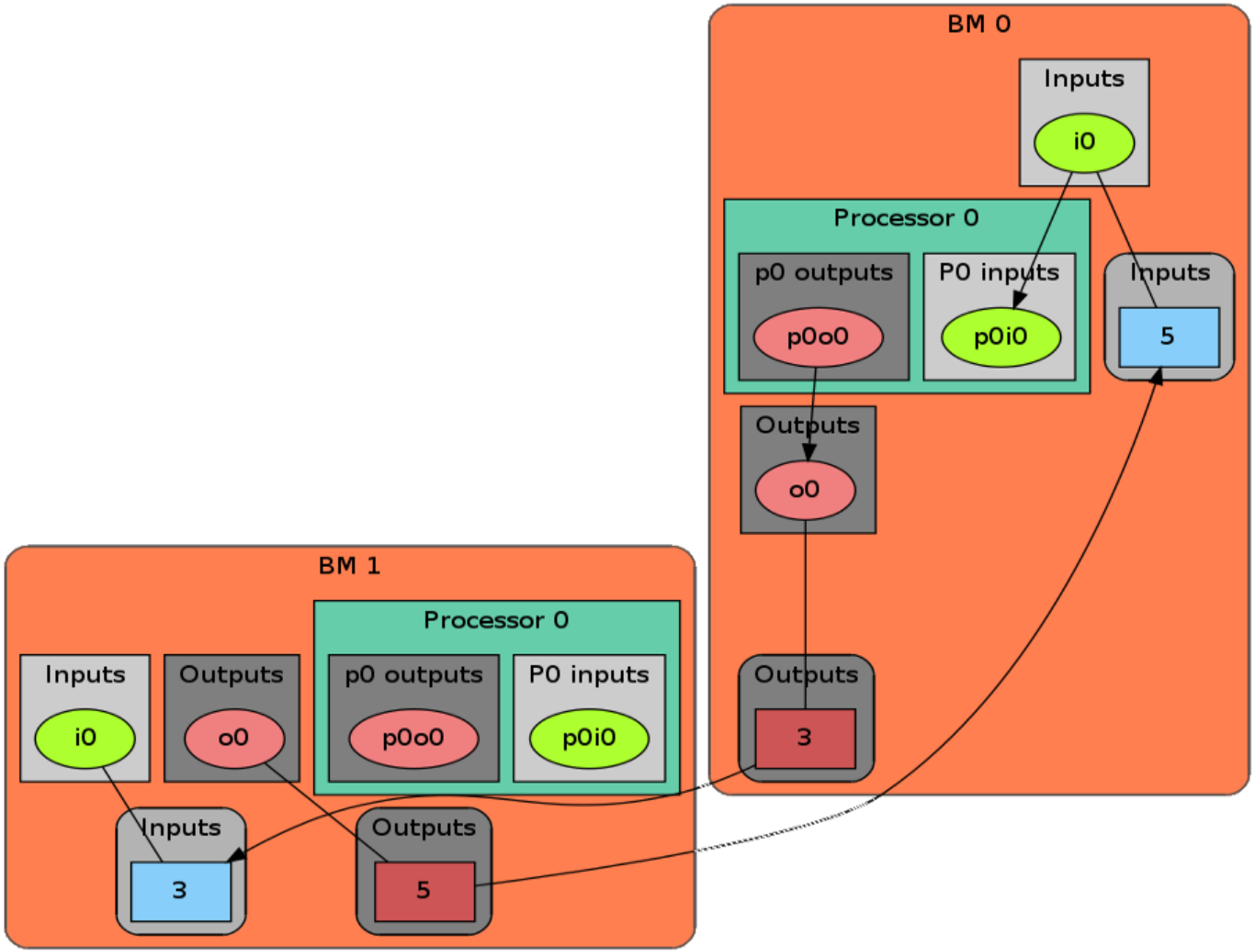
The same logic existing among CP have been extended among different BondMachines organized in clusters.

Protocols, one ethernet called etherbond and one using UDP called udpbond have been created for the purpose.

FPGA based BondMachines, standard Linux Workstations, Emulated BondMachines might join a cluster an contribute to a single distributed computational problem.

BondMachine Clustering

A distributed example



BondMachine Clustering

Results

Results

- User can deploy an entire HW/SW cluster starting from code written in a high level description (Go, NNEF, etc)
- Workstation with emulated BondMachines, workstation with etherbond drivers, standalone BondMachines (FPGA) may join these clusters.

BondMachine Clustering

Results

Results

- User can deploy an entire HW/SW cluster starting from code written in a high level description (Go, NNEF, etc)
- Workstation with emulated BondMachines, workstation with etherbond drivers, standalone BondMachines (FPGA) may join these clusters.

Use cases

Two use cases in Physics experiments are currently being developed:

- Real time pulse shape analysis in neutron detectors
  - bringing the intelligence to the edge
- Test beam for space experiments (DAMPE, HERD)
  - increasing testbed operations efficiency

Computing Accelerator

Our effort is now in enabling the possibility of building computing accelerators to be used from within standard (Linux) applications.

Use cases

Two use cases in Physics experiments are currently being developed:

- Real time pulse shape analysis in neutron detectors
  - bringing the intelligence to the edge
- Test beam for space experiments (DAMPE, HERD)
  - increasing testbed operations efficiency

Computing Accelerator

Our effort is now in enabling the possibility of building computing accelerators to be used from within standard (Linux) applications.

Accelerators

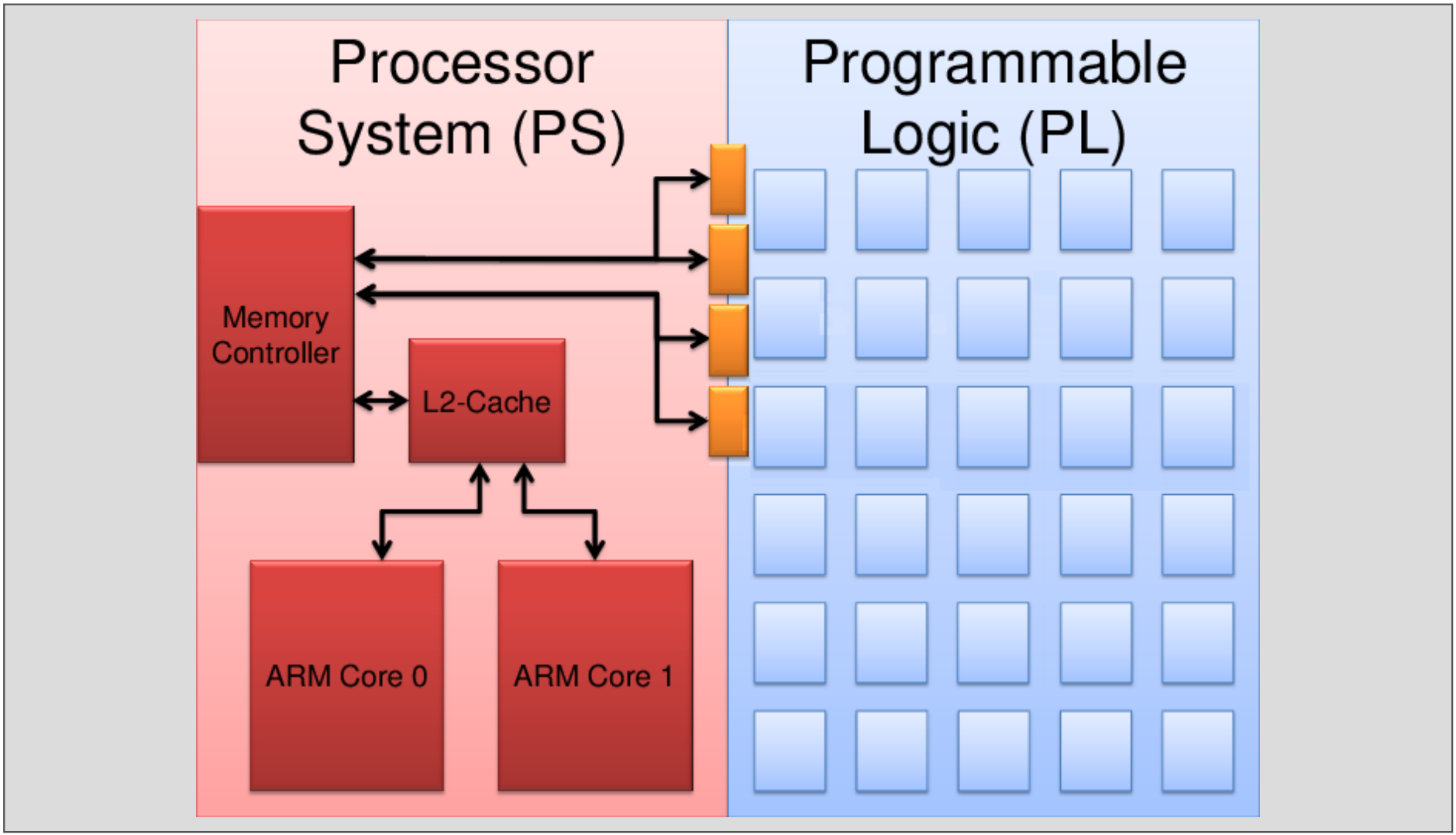
Types

We are currently working to enable the use the BM as accelerator in two directions:

- Using standard processor/FPGA hybrid chips
  - Zynq, Cyclone V
- Using PCI-express FPGA evaluation boards
  - Kintek 7 Evaluation board

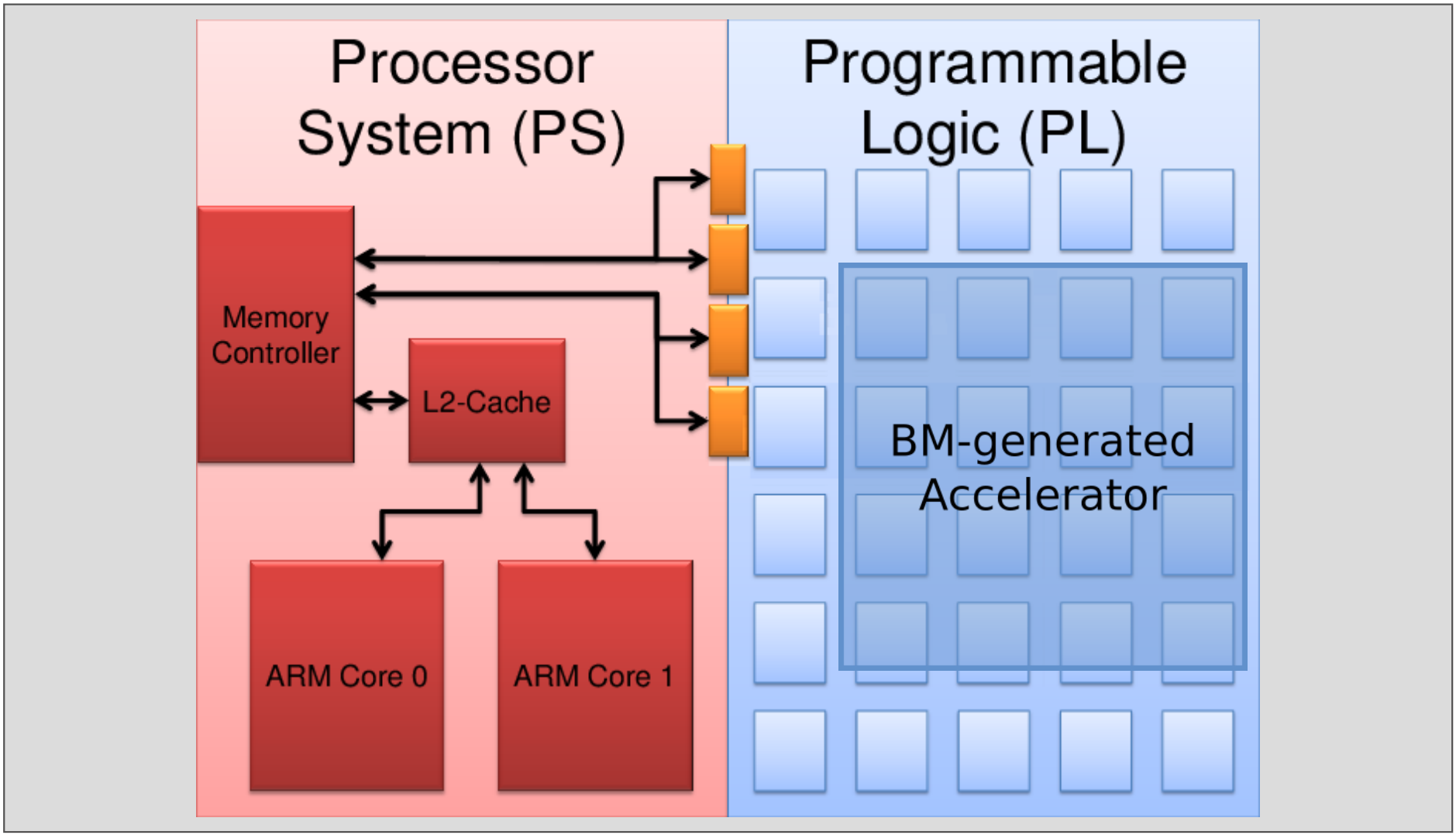
Accelerators

Hybrid chips



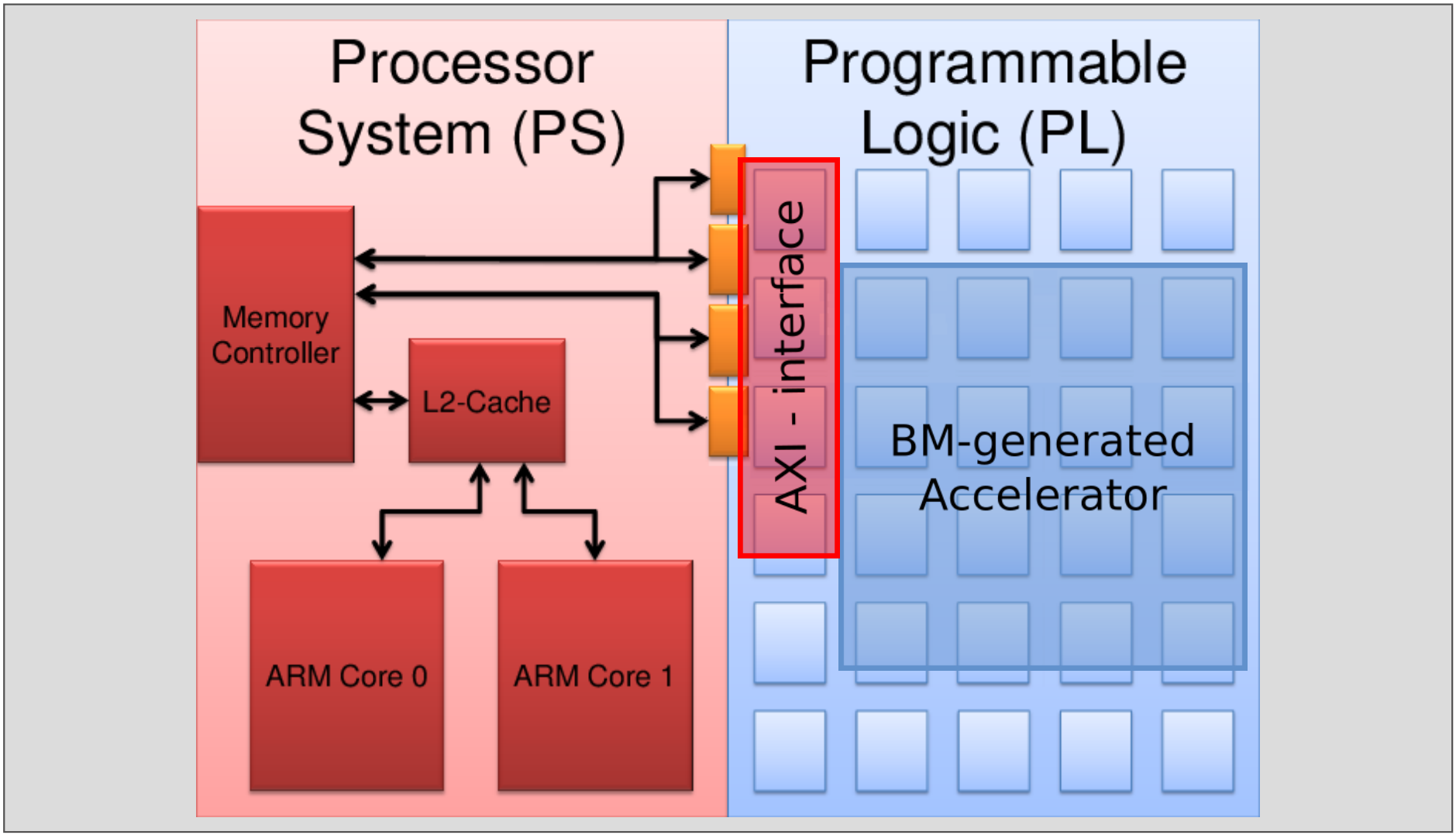
Accelerators

Hybrid chips



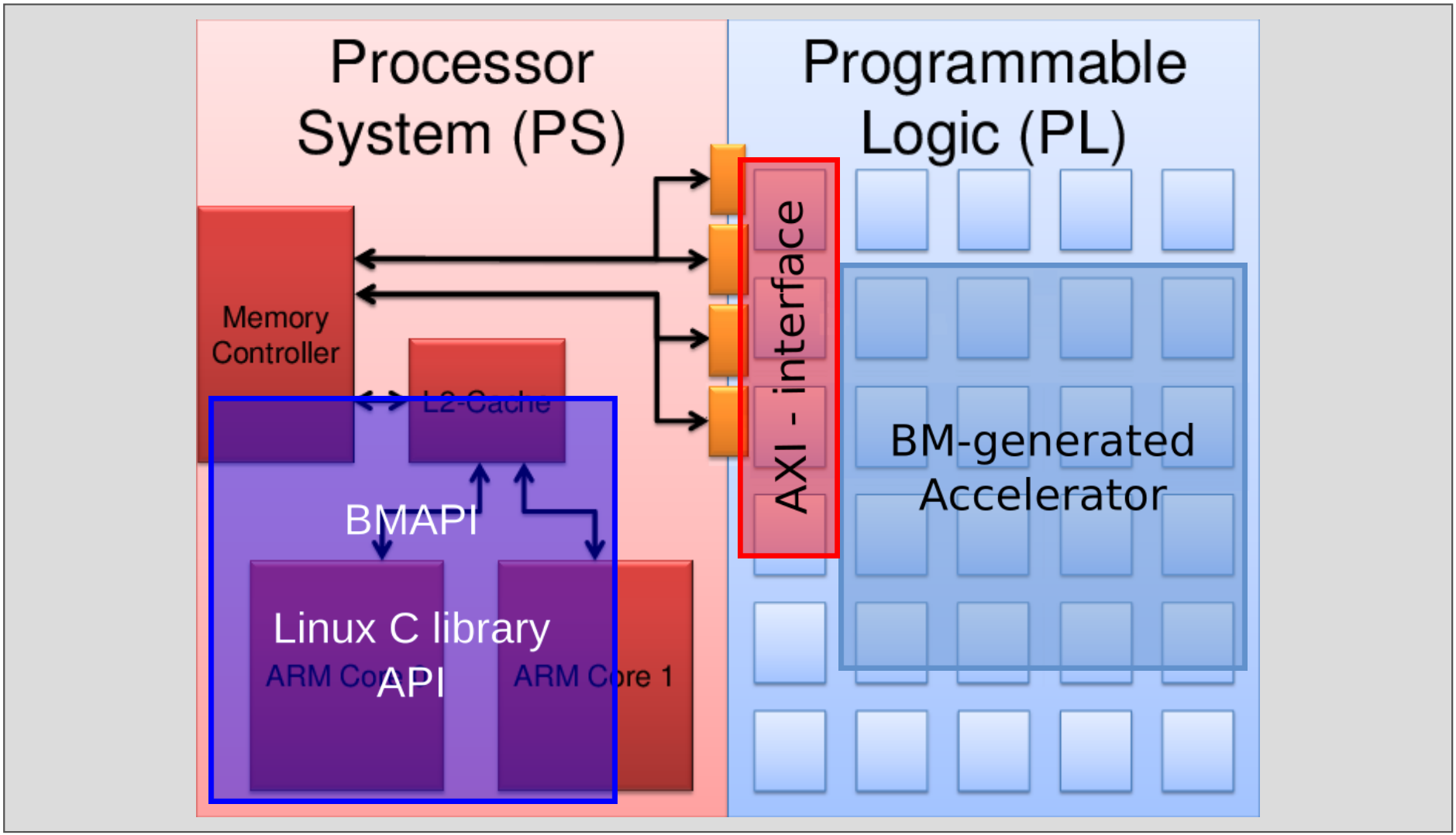
Accelerators

Hybrid chips



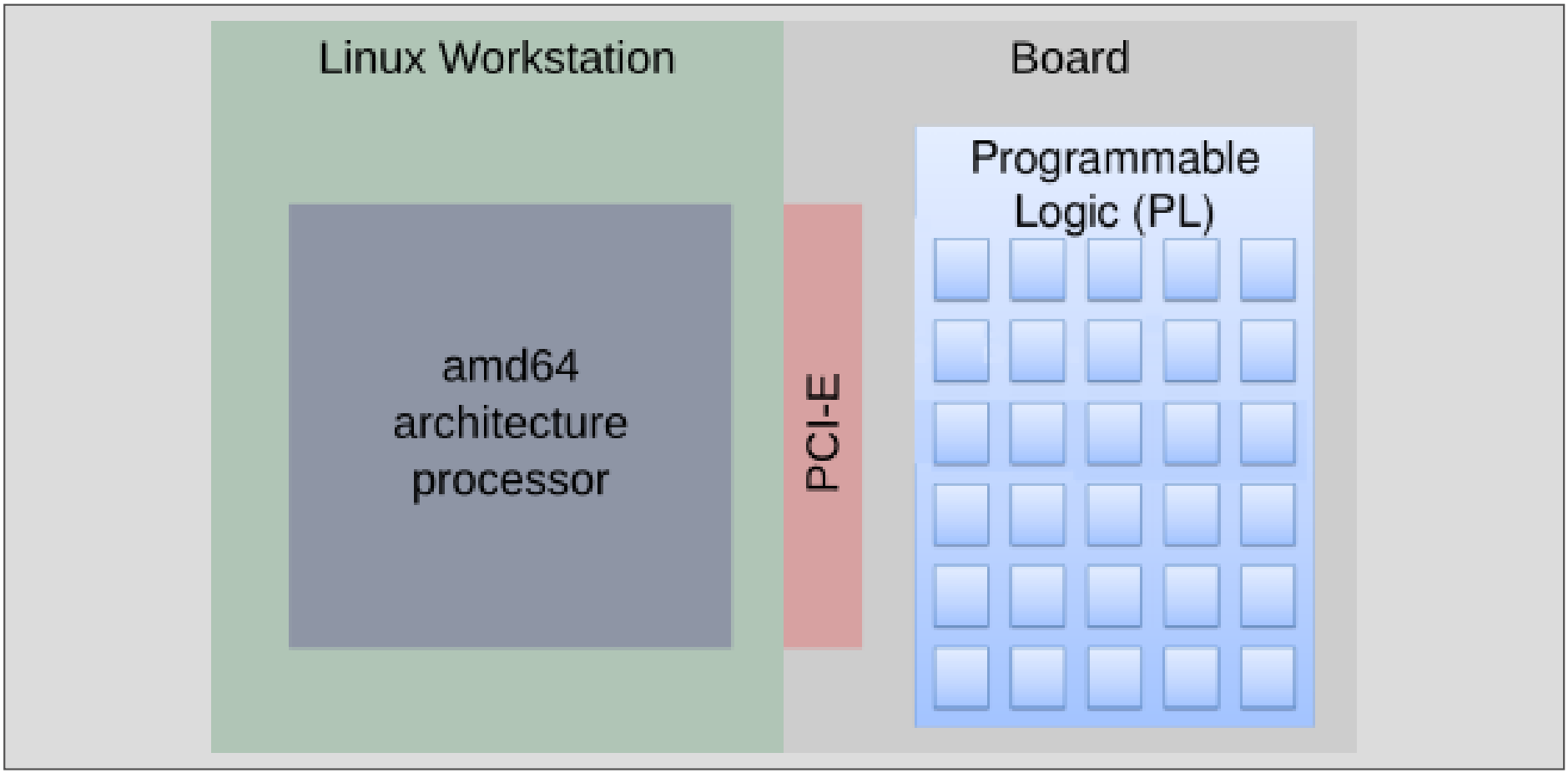
Accelerators

Hybrid chips



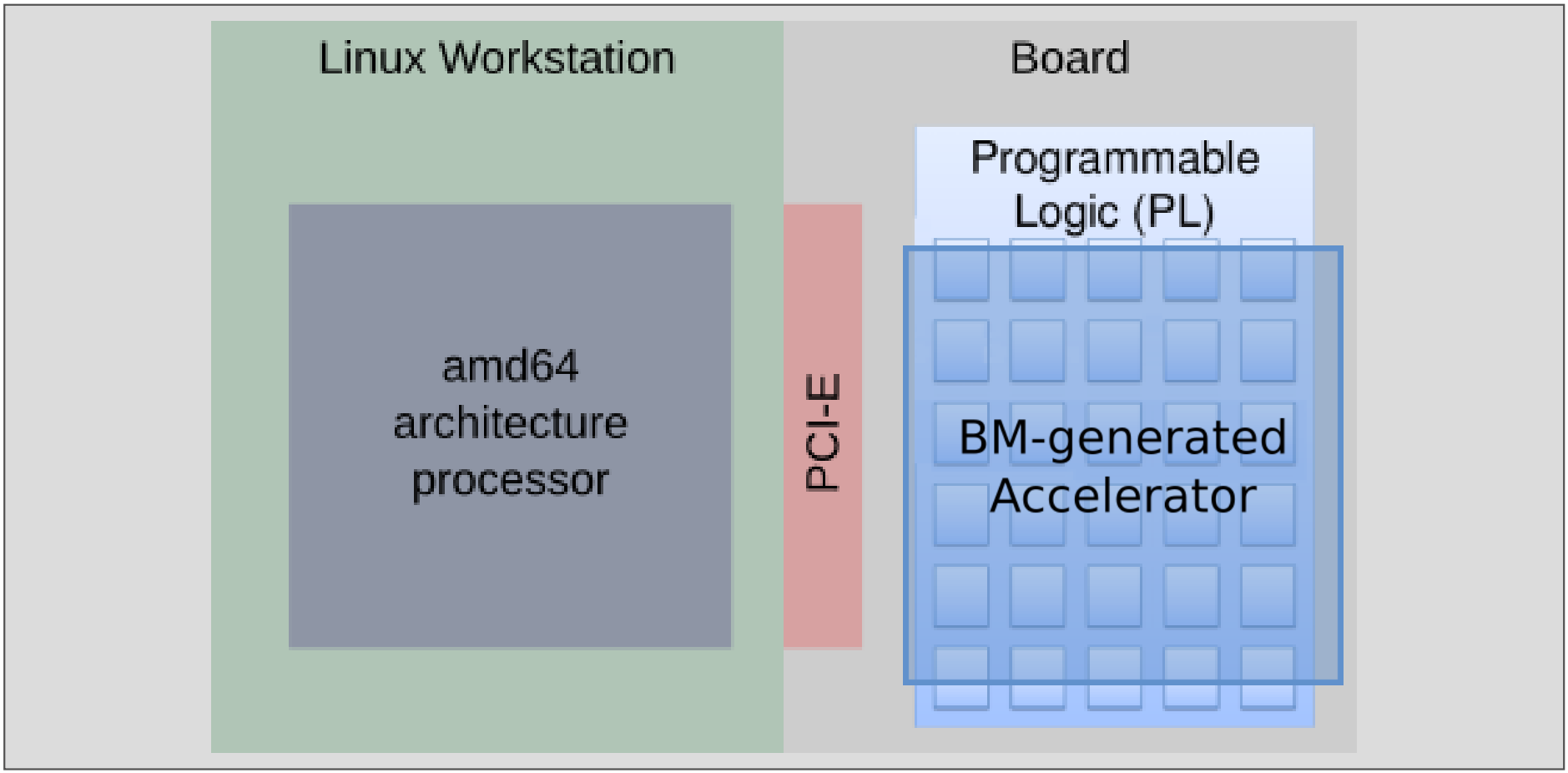
Accelerators

PCI-express boards



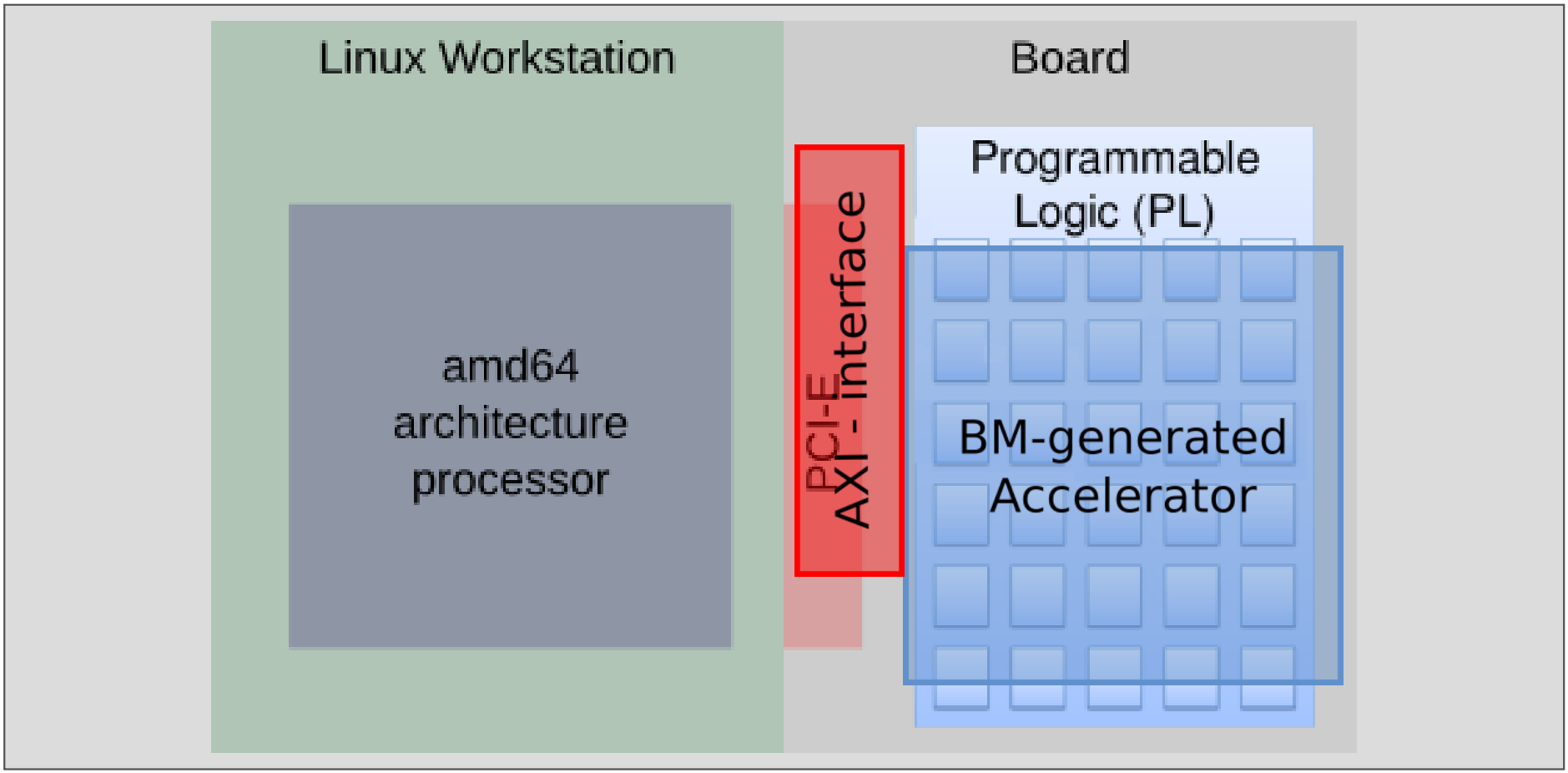
Accelerators

PCI-express boards



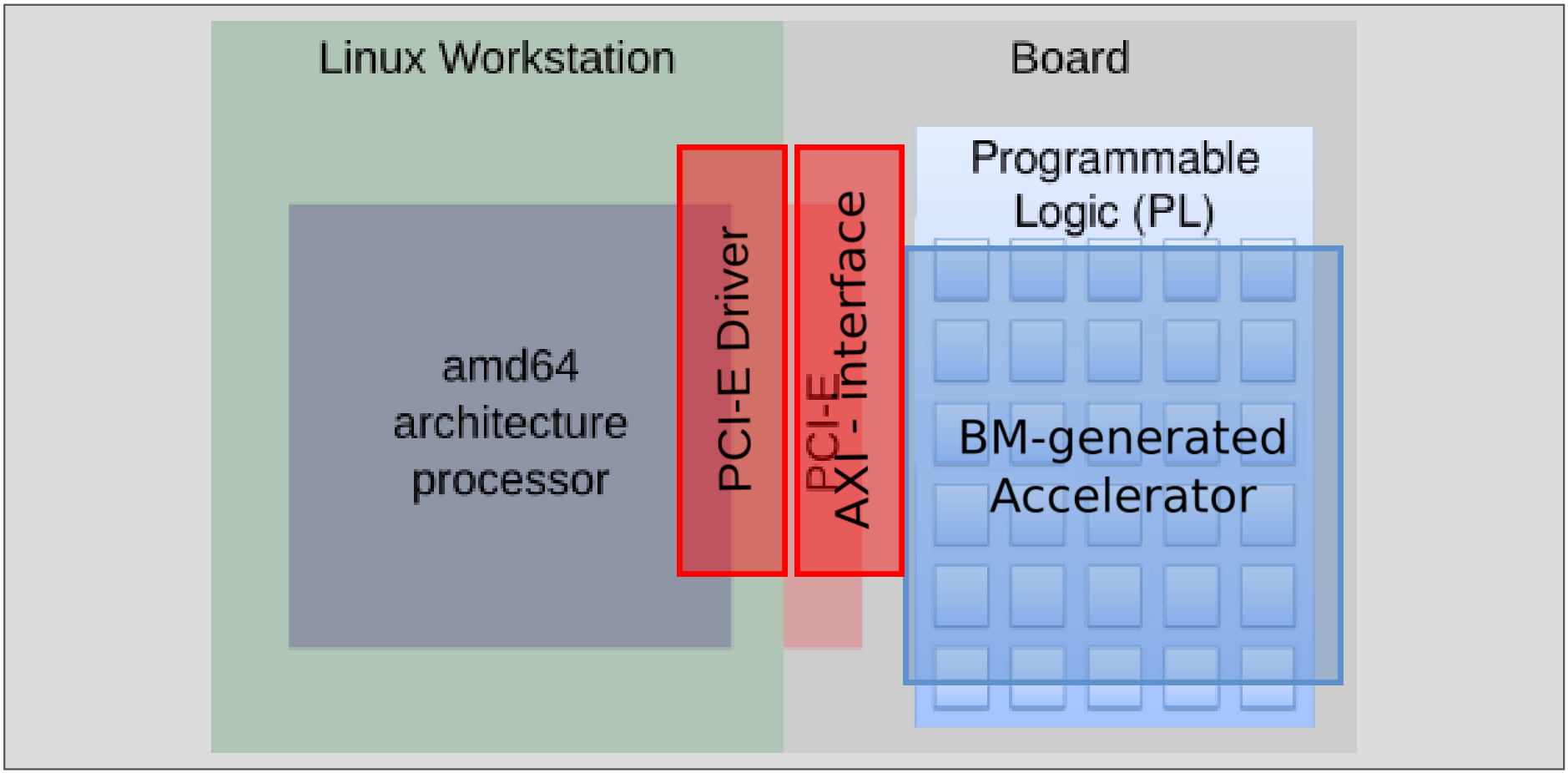
Accelerators

PCI-express boards



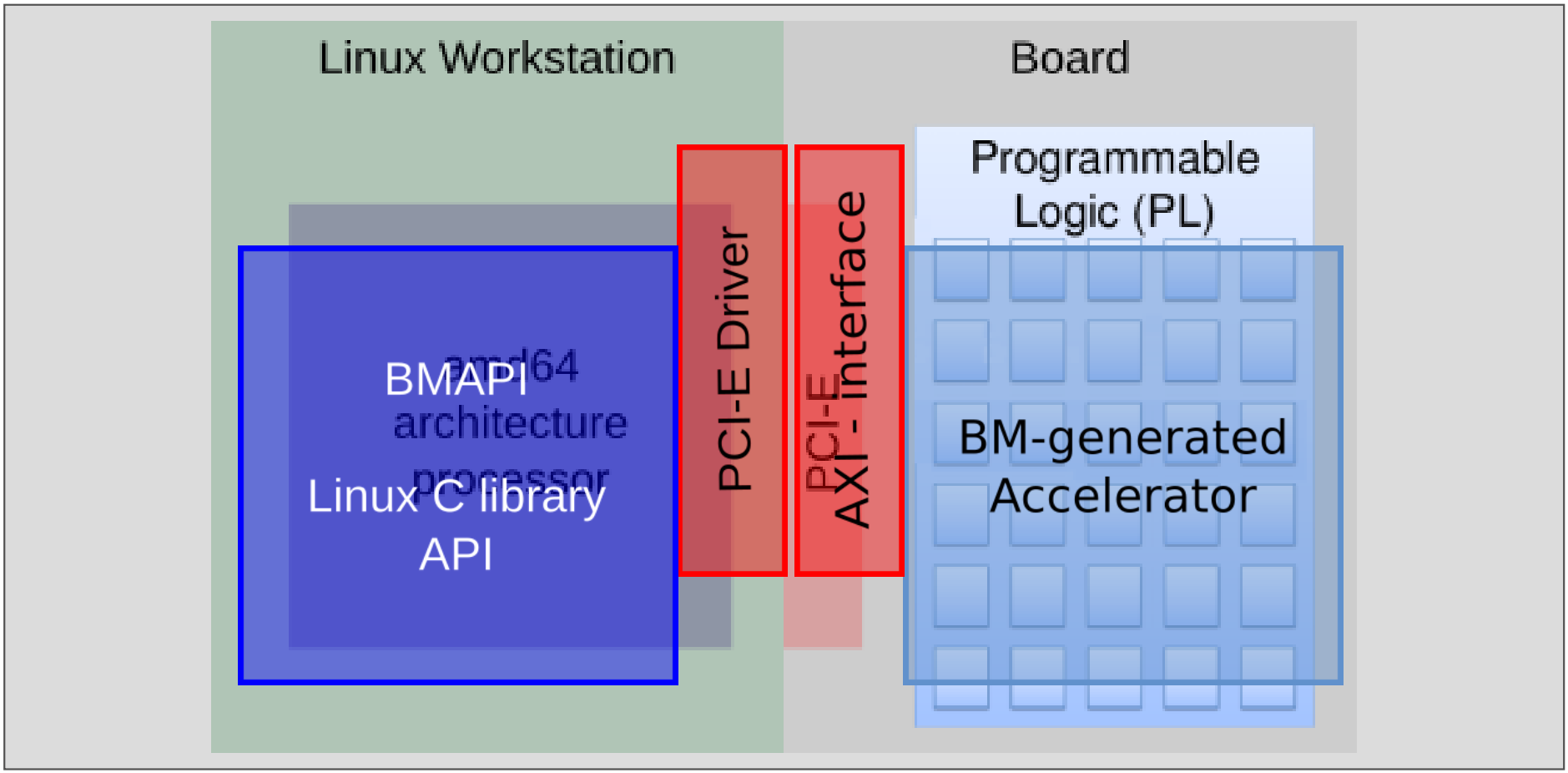
Accelerators

PCI-express boards



Accelerators

PCI-express boards

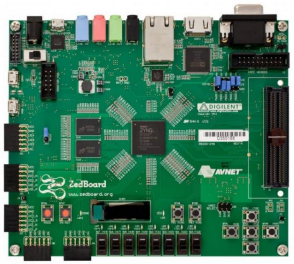


Accelerators

Hardware

Hybrid chips

Digilent Zedboard

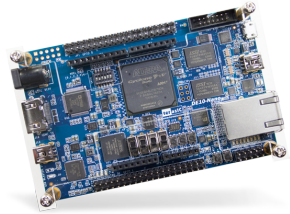


Zynq-7000 SoC XC7Z020
512 MB DDR3
Up to 667 MHz

Xilinx ZC702

Zynq-7000 SoC XC7Z020
1GB DDR3
85k cells - 220 DSP slices

Terasic DE10Nano



Intel Cyclone V
1GB DDR3 SDRAM
110K LEs

PCI-Express board

Xilinx KC705

Kintex-7 FPGAs
1GB DDR3 SODIM
326k cells - 840 DSP slices

Accelerators

Cloud

FPGA accelerators can be used in the cloud:

- Several public cloud providers offers solution of VM connected to FPGAs (Amazon, Nimbix)
- FPGAs can be inserted in private clouds infrastructures

To be used a firmware has to be uploaded to the accelerated VM FPGA

The BondMachine toolkit can be used to build such firmware

Accelerators

Cloud

FPGA accelerators can be used in the cloud:

- Several public cloud providers offers solution of VM connected to FPGAs (Amazon, Nimbix)
- FPGAs can be inserted in private clouds infrastructures

To be used a firmware has to be uploaded to the accelerated VM FPGA

The BondMachine toolkit can be used to build such firmware

Accelerators

Cloud

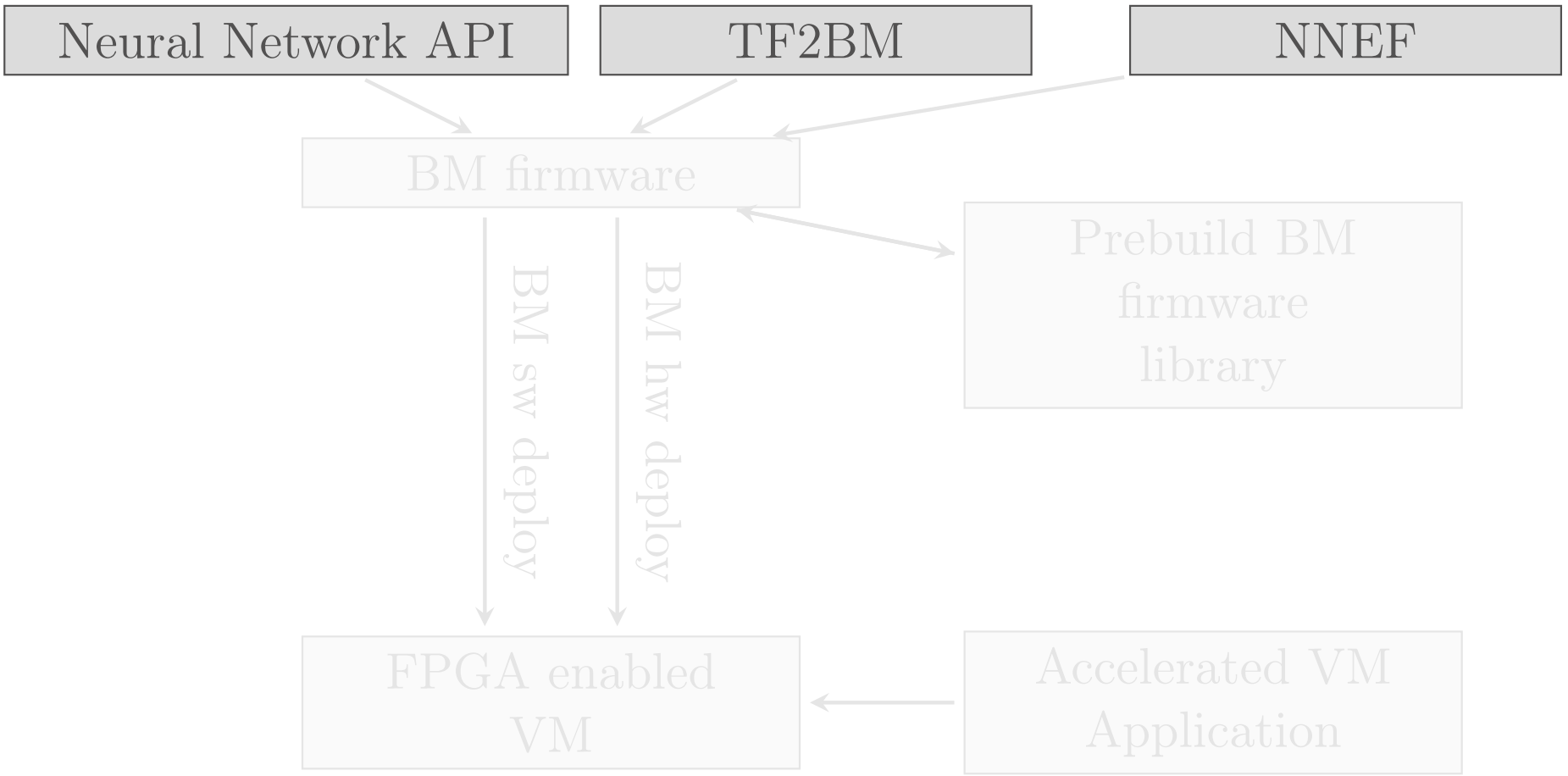
FPGA accelerators can be used in the cloud:

- Several public cloud providers offers solution of VM connected to FPGAs (Amazon, Nimbix)
- FPGAs can be inserted in private clouds infrastructures

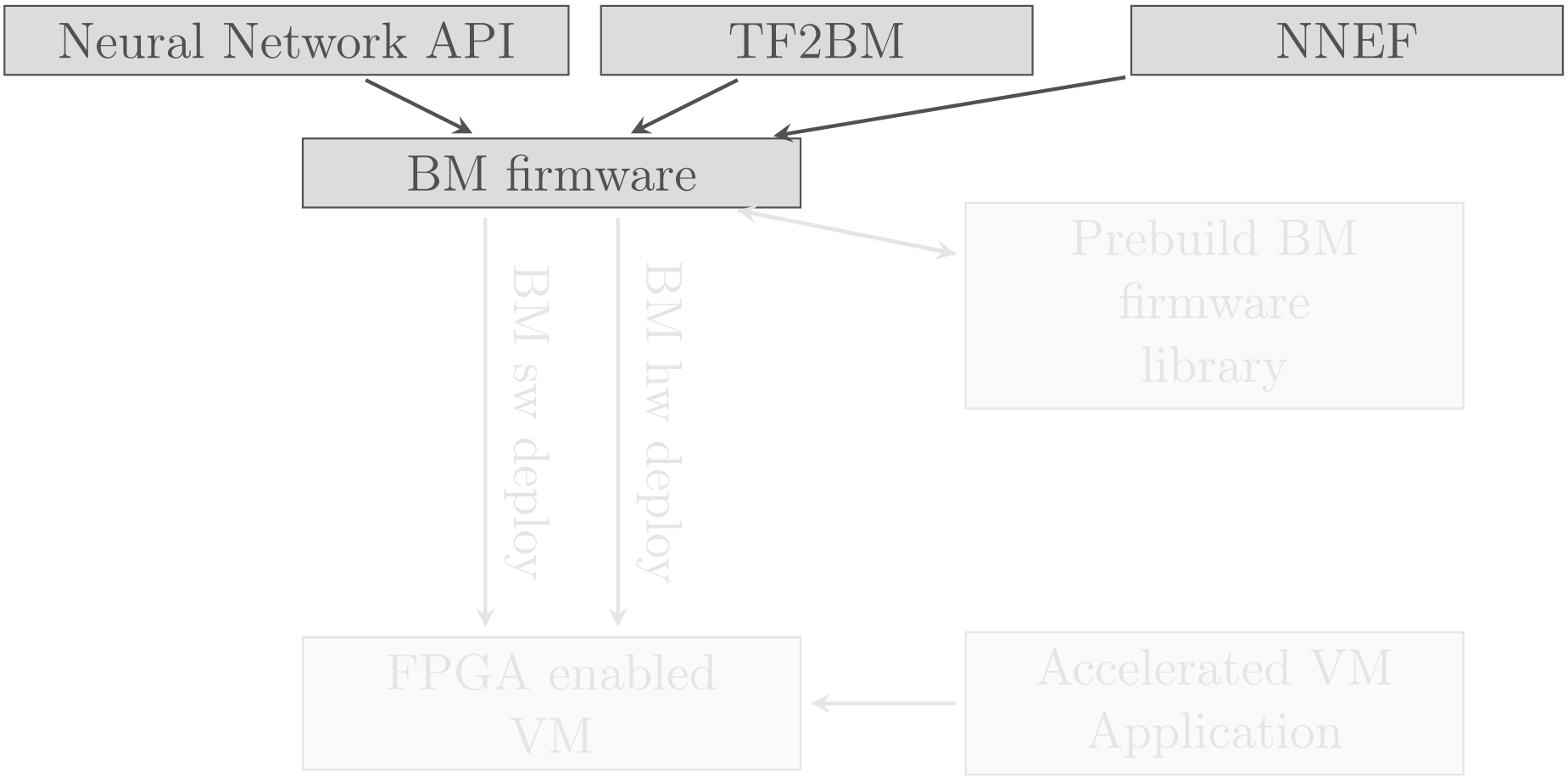
To be used a firmware has to be uploaded to the accelerated VM FPGA

The BondMachine toolkit can be used to build such firmware

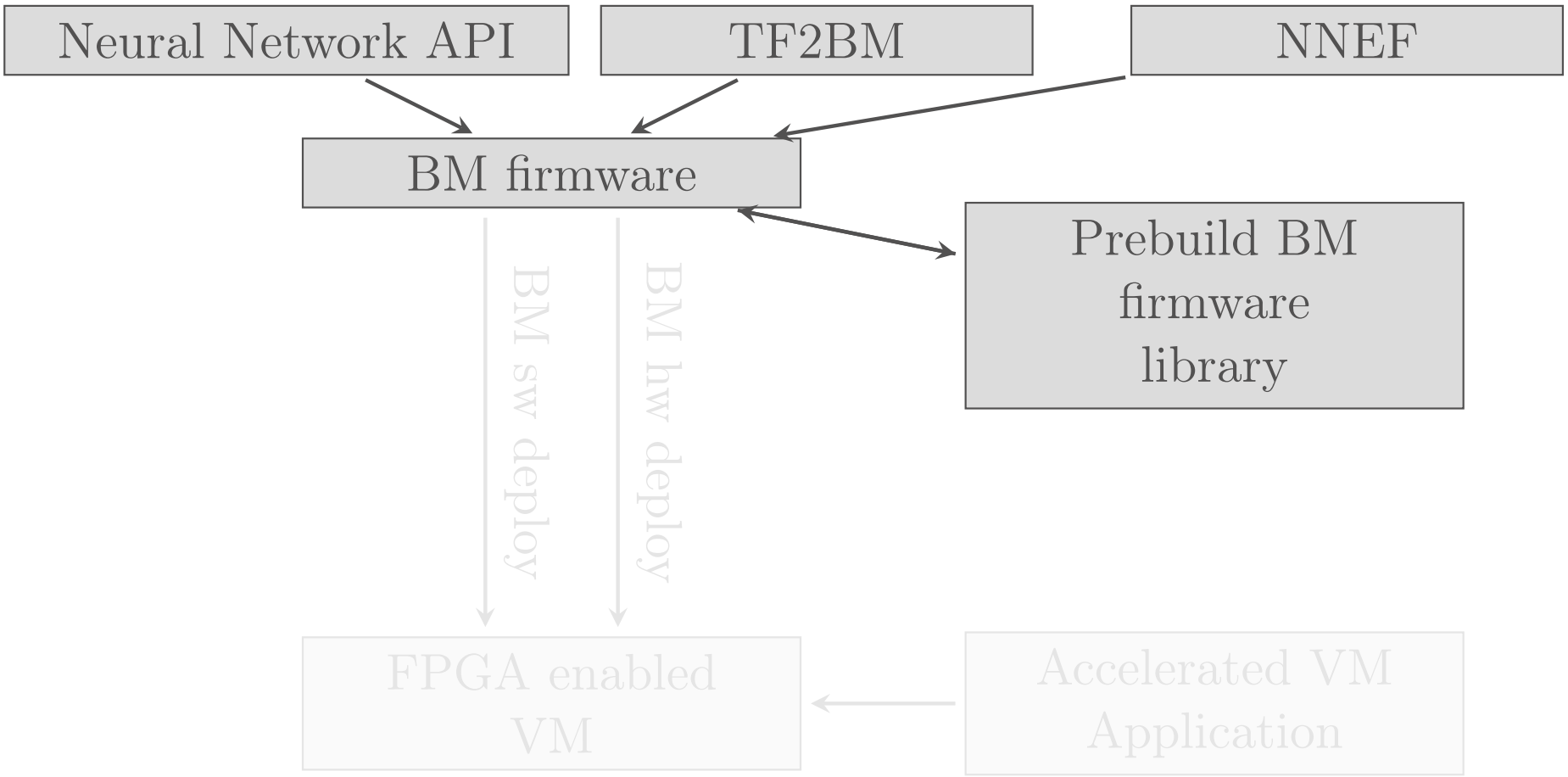
Accelerated ML in the Cloud



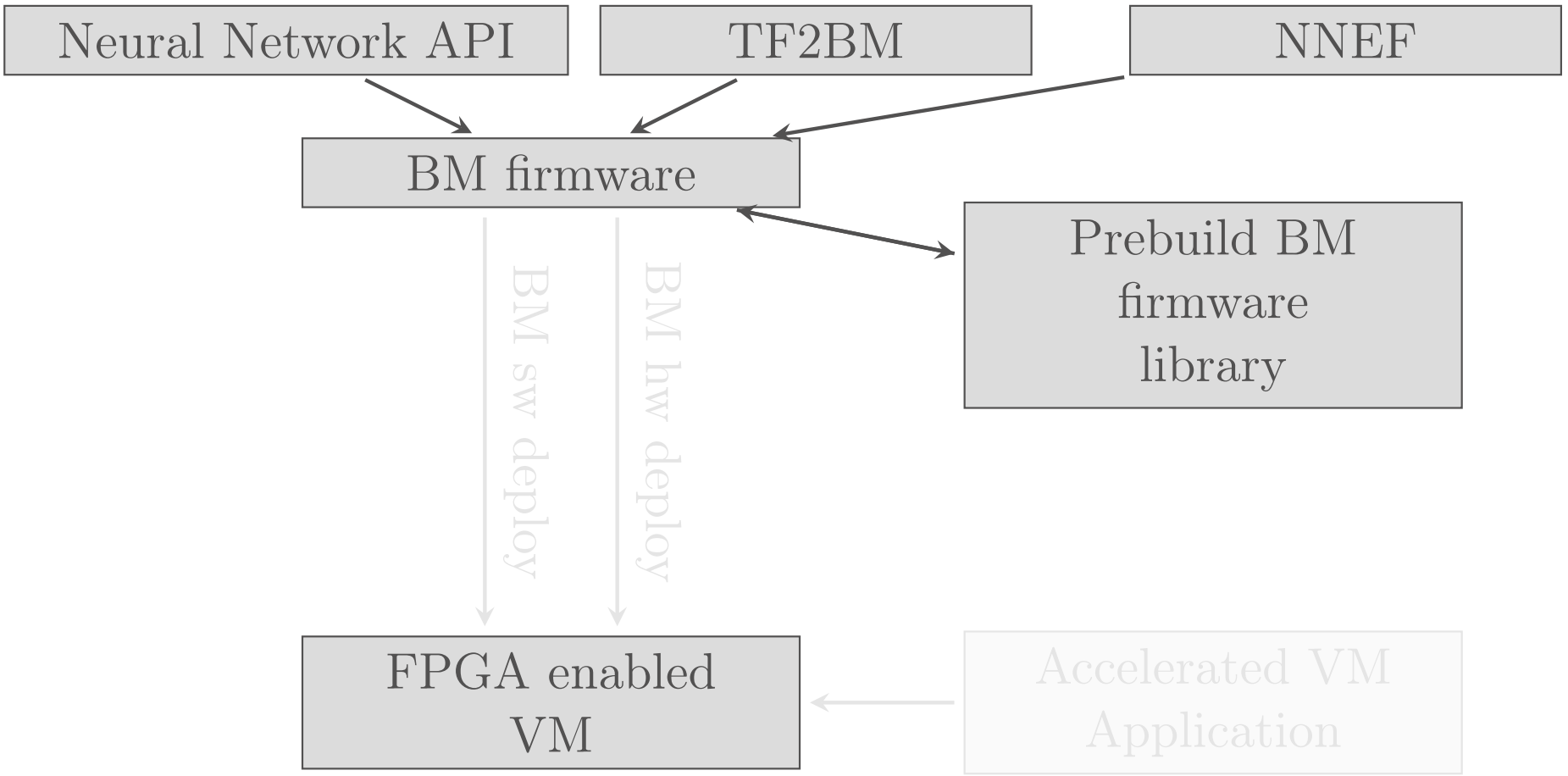
Accelerated ML in the Cloud



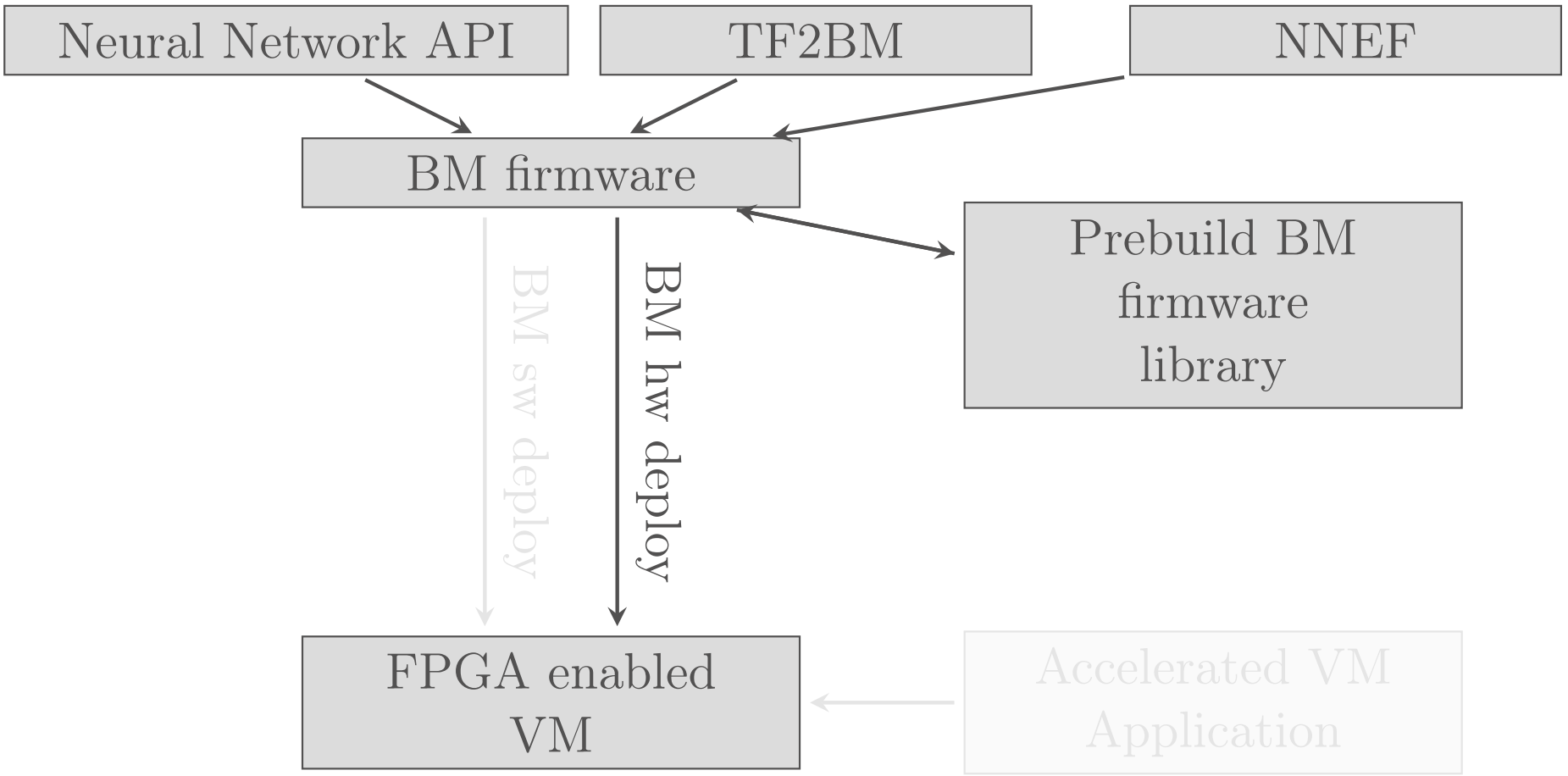
Accelerated ML in the Cloud



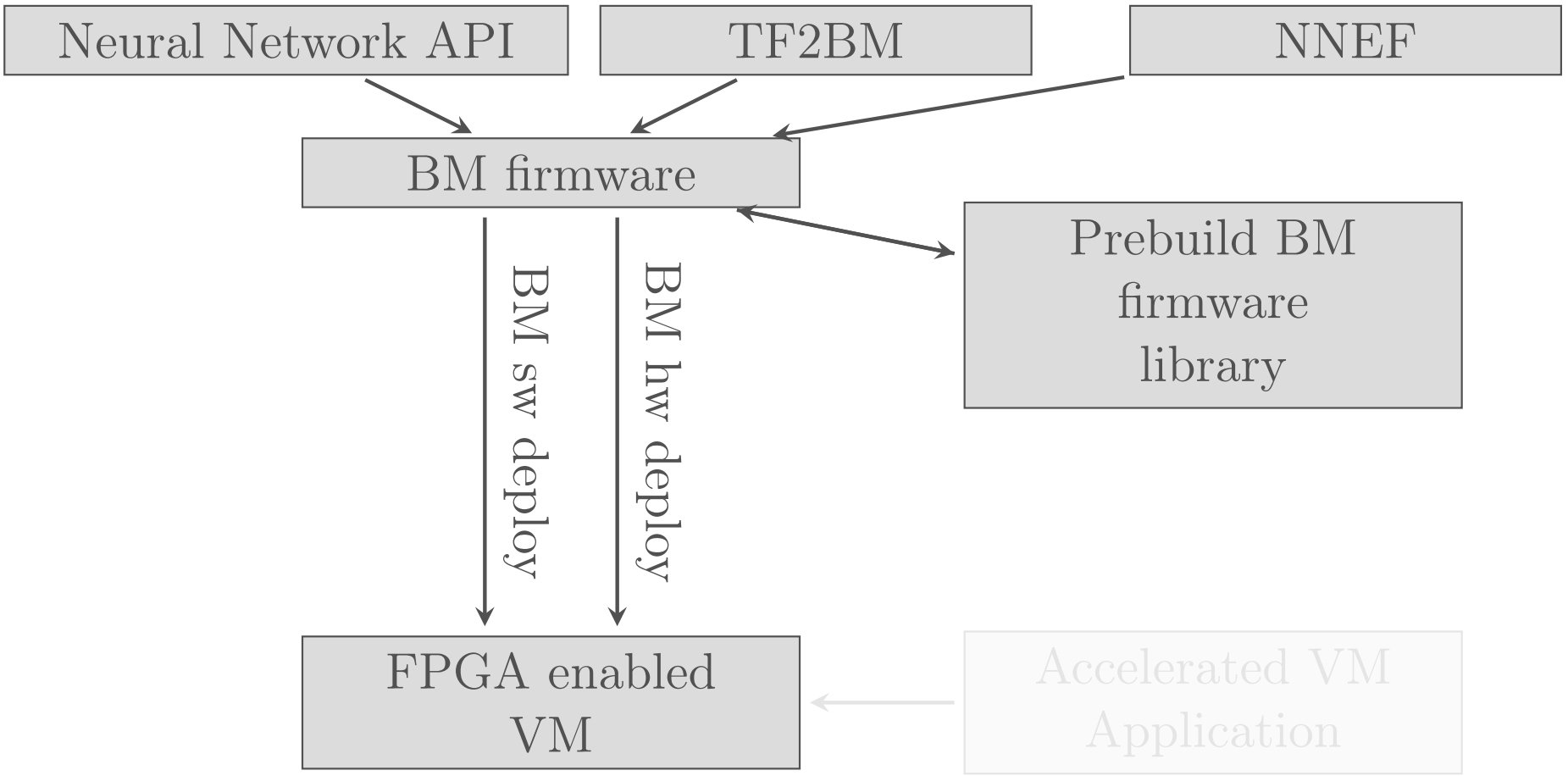
Accelerated ML in the Cloud



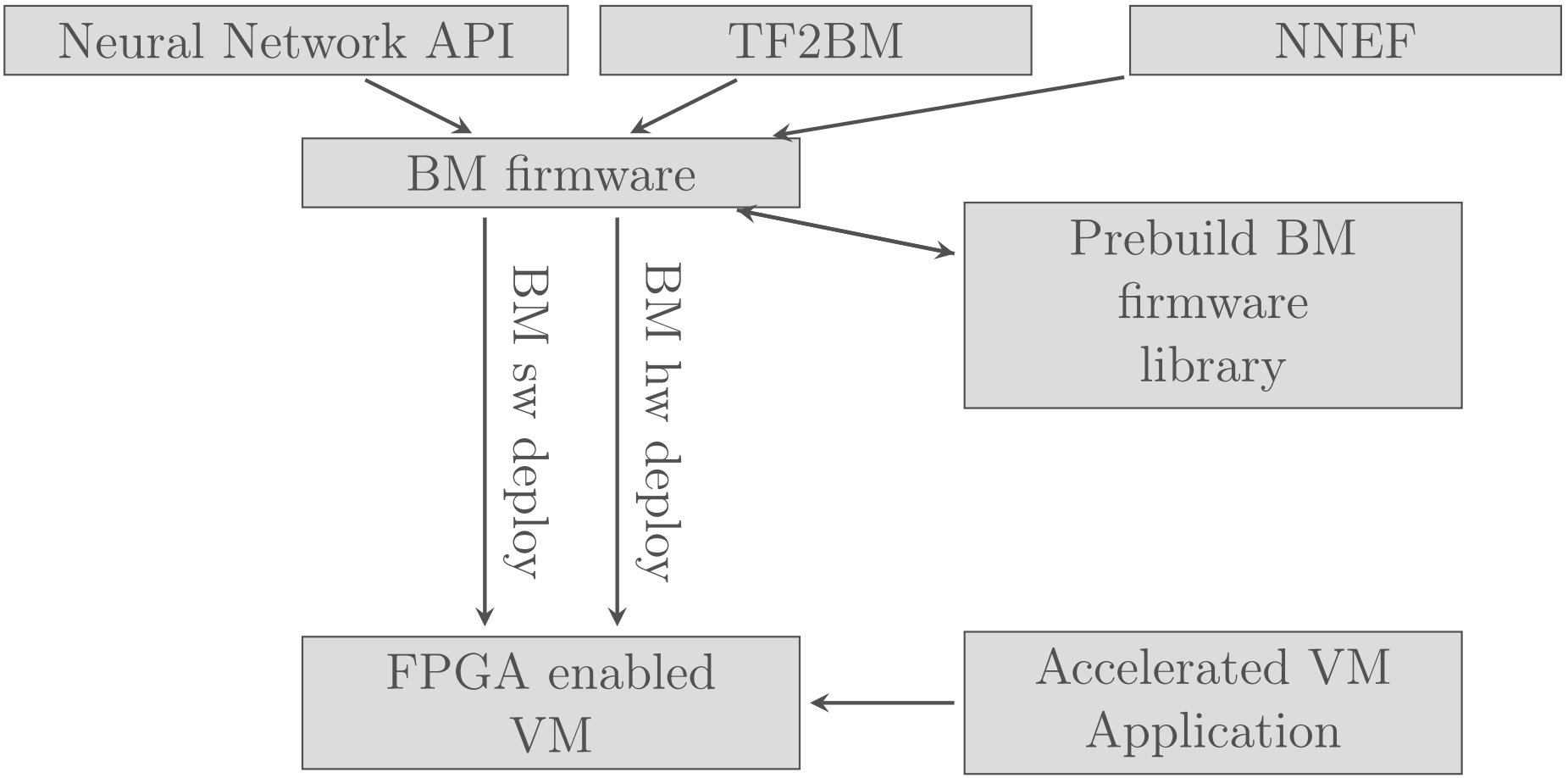
Accelerated ML in the Cloud



Accelerated ML in the Cloud

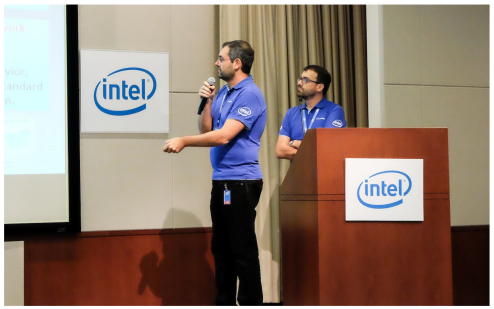


Accelerated ML in the Cloud



Project History

- May 2016 - First tests on the idea.
- October 2016 - Prototype at "Makerfaire 2016 Rome"
- Jul 2018 - InnovateFPGA EMEA Silver Award.
- Aug 2018 - Presented at Intel Campus, Santa Jose (CA) .
- Aug 2018 - InnovateFPGA Iron Award in the Grand Final.



Conclusions

The BondMachine is a new kind of computing device made possible in practice only by the emerging of new re-programmable hardware technologies such as FPGA.

The result of this process is the construction of a computer architecture that is not anymore a static constraint where computing occurs but its creation becomes a part of the computing process, gaining computing power and flexibility.

Over this abstraction is it possible to create a full computing Ecosystem, ranging from small interconnected IoT devices to Machine Learning accelerators.

Future work

The project is at the stage of a working prototype, so work has to be done in several areas:

- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking including new interconnection firmwares.
- Work on BondMachine as accelerators.

What would an OS for BondMachines look like ?

Future work

The project is at the stage of a working prototype, so work has to be done in several areas:

- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking including new interconnection firmwares.
- Work on BondMachine as accelerators.

What would an OS for BondMachines look like ?

Future work

The project is at the stage of a working prototype, so work has to be done in several areas:

- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking including new interconnection firmwares.
- Work on BondMachine as accelerators.

What would an OS for BondMachines look like ?

Future work

The project is at the stage of a working prototype, so work has to be done in several areas:

- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking including new interconnection firmwares.
- Work on BondMachine as accelerators.

What would an OS for BondMachines look like ?

Future work

The project is at the stage of a working prototype, so work has to be done in several areas:

- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking including new interconnection firmwares.
- Work on BondMachine as accelerators.

What would an OS for BondMachines look like ?

Future work

The project is at the stage of a working prototype, so work has to be done in several areas:

- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking including new interconnection firmwares.
- Work on BondMachine as accelerators.

What would an OS for BondMachines look like ?

If you have question/curiosity on the project:

Mirko Mariotti
mirko.mariotti@unipg.it
`http://bondmachine.fisica.unipg.it`