BondMachine

A Mouldable Computer Architecture

Mirko Mariotti

Department of Physics and Geology - University of Perugia
and INFN Perugia
Workshop INFN CCR 2017
Laboratori Nazionali del Gran Sasso, 22-26 Maggio 2017

May 23, 2017

Introduction

The BondMachine: a comprehensive approach to computing.

In this presentation i will talk about:

- Ideas that bring to the BondMachine.
- What is it.
- Developed software tools.
- Hardware implementation.
- Uses.

Computer Architectures
Base for the first idea

Today's computer architecture are:

- Multicore, Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore, Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

Base for the first idea

Today's computer architecture are:

- Multicore,Two or more independent actual processing units execute multiple instructions at the same time.
  - The power is given by the number.
  - Parallel algorithms.
- Heterogeneous, processing units of different type.
  - Cell, GPU, Parallela, TPU.
  - The power is given by the specialization.
  - Hard to make units communicate.
  - Hard to program.
  - Hard to schedule.

Computer Architectures

First idea

Having a multi-core architecture completely heterogeneous both in cores types and interconnections.

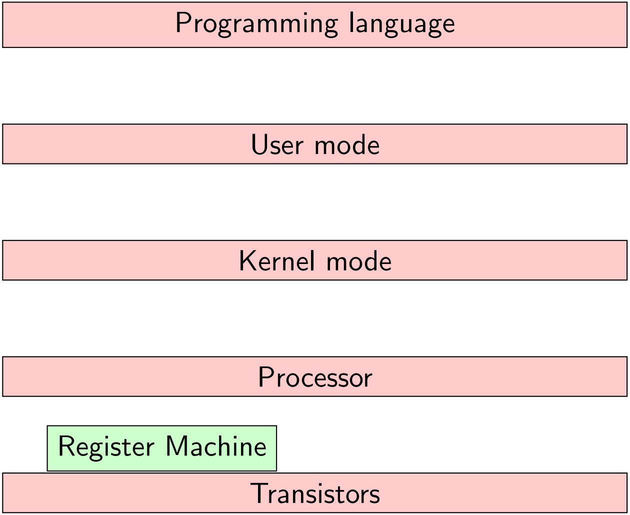
Layers, Abstractions and Interfaces

base for the second idea

Programming language
User mode
Kernel mode
Processor
Transistors

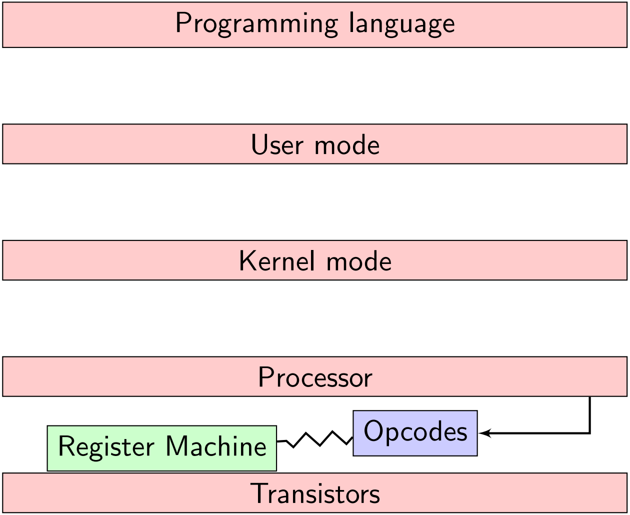
Layers, Abstractions and Interfaces

base for the second idea



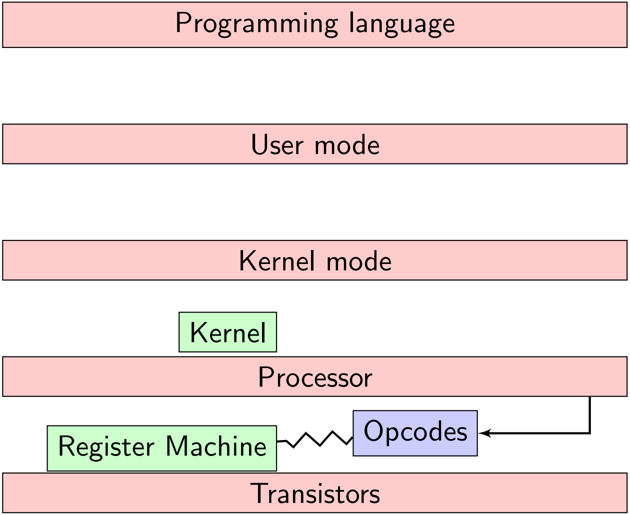
Layers, Abstractions and Interfaces

base for the second idea



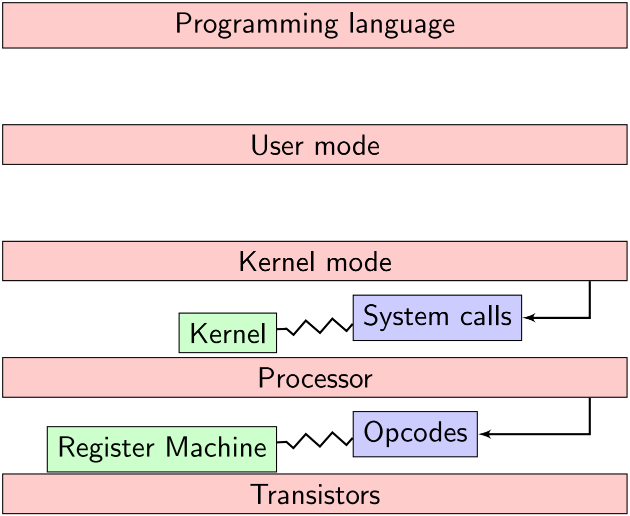
Layers, Abstractions and Interfaces

base for the second idea



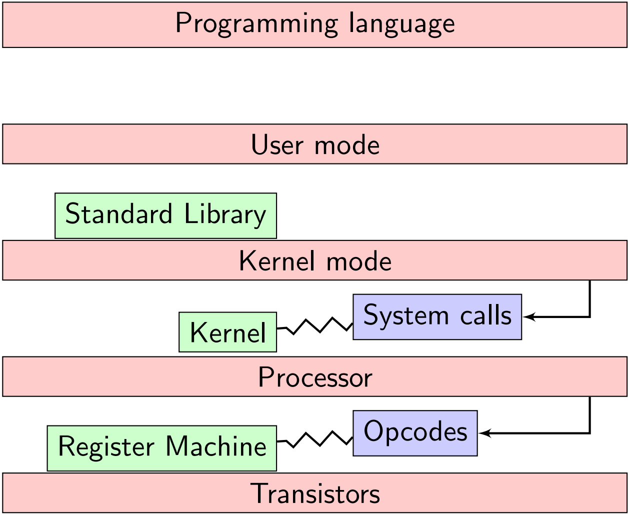
Layers, Abstractions and Interfaces

base for the second idea



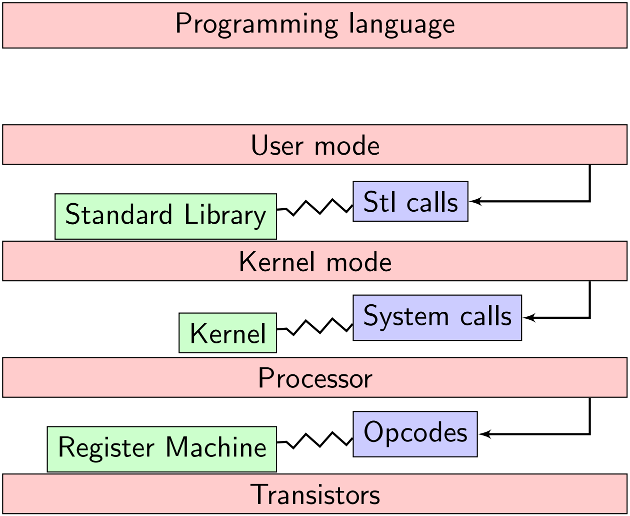
Layers, Abstractions and Interfaces

base for the second idea



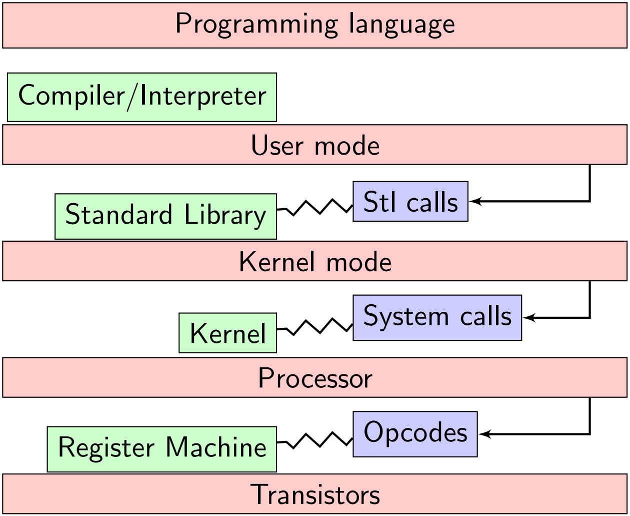
Layers, Abstractions and Interfaces

base for the second idea



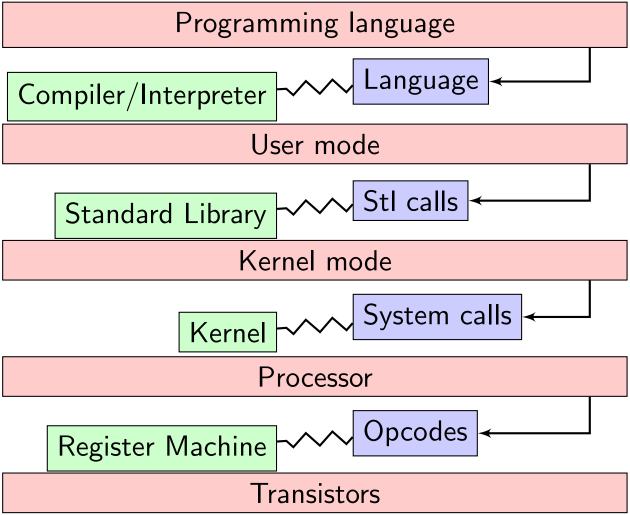
Layers, Abstractions and Interfaces

base for the second idea



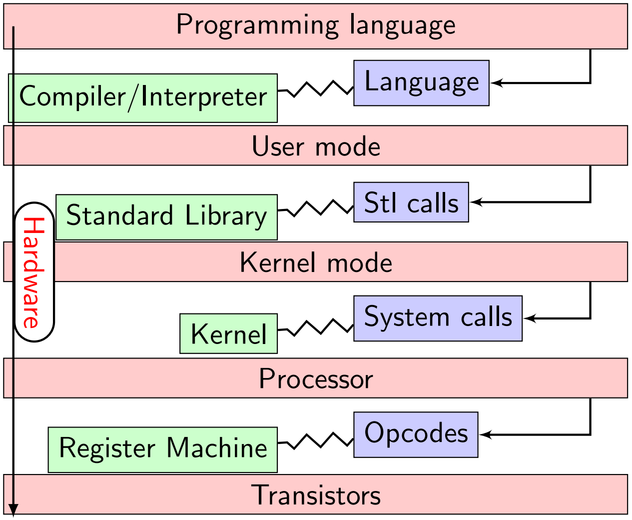
Layers, Abstractions and Interfaces

base for the second idea



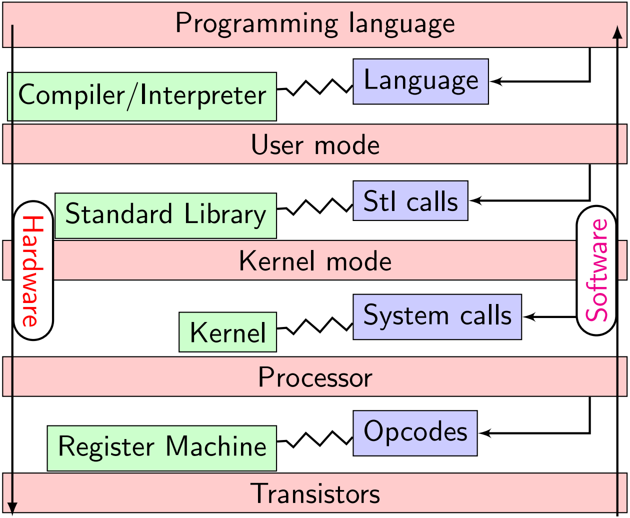
Layers, Abstractions and Interfaces

base for the second idea



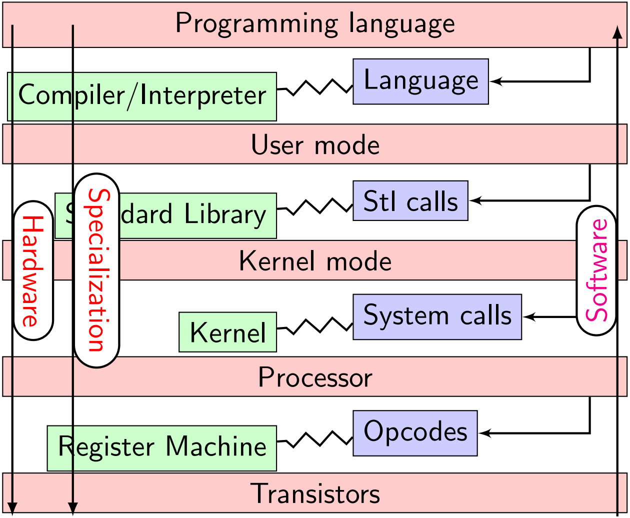
Layers, Abstractions and Interfaces

base for the second idea



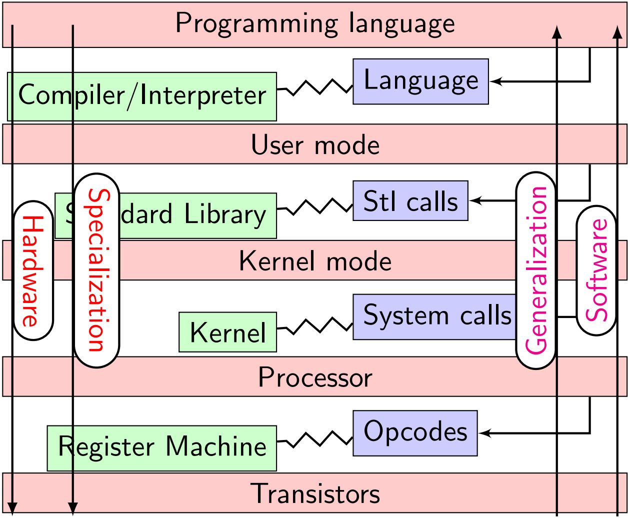
Layers, Abstractions and Interfaces

base for the second idea



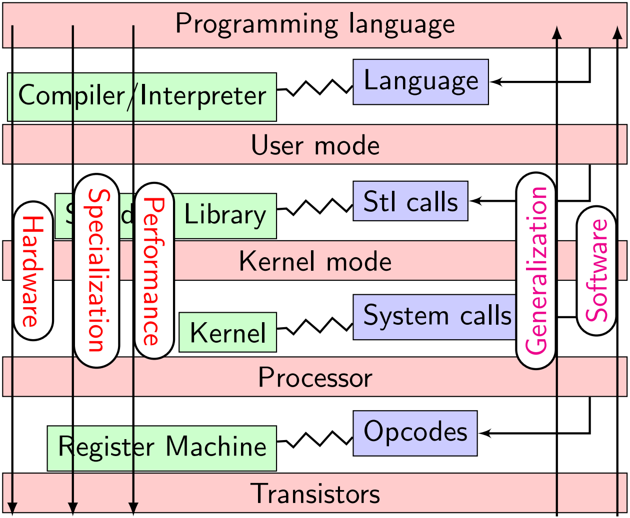
Layers, Abstractions and Interfaces

base for the second idea



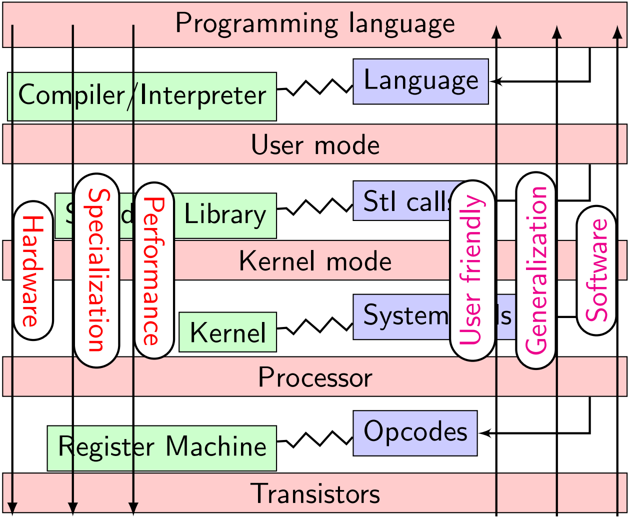
Layers, Abstractions and Interfaces

base for the second idea



Layers, Abstractions and Interfaces

base for the second idea



Layers, Abstractions and Interfaces
Second idea

Build a computing system with a decreased number of layers resulting in a minor gap between HW and SW but keeping an user friendly way of programming it.

Introducing the BondMachine (BM)

Inspired from both the ideas we create a new computer architecture that:

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

Introducing the BondMachine (BM)

Inspired from both the ideas we create a new computer architecture that:

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

Introducing the BondMachine (BM)

Inspired from both the ideas we create a new computer architecture that:

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

Introducing the BondMachine (BM)

Inspired from both the ideas we create a new computer architecture that:

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

Introducing the BondMachine (BM)

Inspired from both the ideas we create a new computer architecture that:

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

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)

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.

Three kind of SO have been developed so far: the Channel, the Shared Memory and the Barrier.

Shared Objects (SO)

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.

Three kind of SO have been developed so far: the Channel, the Shared Memory and the Barrier.

Shared Objects (SO)

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.

Three kind of SO have been developed so far: the Channel, the Shared Memory and the Barrier.

Shared Objects (SO)

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.

Three kind of SO have been developed so far: the Channel, the Shared Memory and the Barrier.

Shared Objects (SO)

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.

Three kind of SO have been developed so far: the Channel, the Shared Memory and the Barrier.

Shared Objects (SO)

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

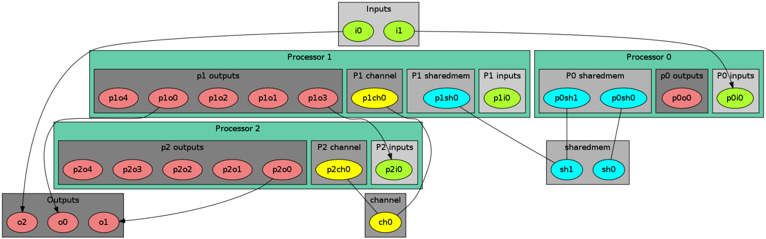
Examples of their purposes are:

- 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.

Three kind of SO have been developed so far: the Channel, the Shared Memory and the Barrier.

The BondMachine



Multicore and Heterogeneous

First idea on the BondMachine

The idea was:

Having a multi-core architecture completely heterogeneous both in cores types and interconnections.

The BondMachine may have many cores, eventually all different one another, arbitrarily interconnected and sharing non computing elements.

Tools

The complexity of programming the BondMachine architecture is managed by using a set of software tools to:

- build a specify architecture as function of the task,
- modify the created architecture,
- simulate the behavior and to check the functionality with the aim to generate the Register Transfer Level (RTL) code.

Processor Builder selects the CP specifics, assembles and disassembles, saves on disk as JSON, emulates and creates the RTL code.

BondMachine Builder connects CPs and SOs together in custom topologies, loads and saves on disk as JSON, emulates and creates the RTL code.

Tools

The complexity of programming the BondMachine architecture is managed by using a set of software tools to:

- build a specify architecture as function of the task,
- modify the created architecture,
- simulate the behavior and to check the functionality with the aim to generate the Register Transfer Level (RTL) code.

Processor Builder selects the CP specifics, assembles and disassembles, saves on disk as JSON, emulates and creates the RTL code.

BondMachine Builder connects CPs and SOs together in custom topologies, loads and saves on disk as JSON, emulates and creates the RTL code.

Tools

The complexity of programming the BondMachine architecture is managed by using a set of software tools to:

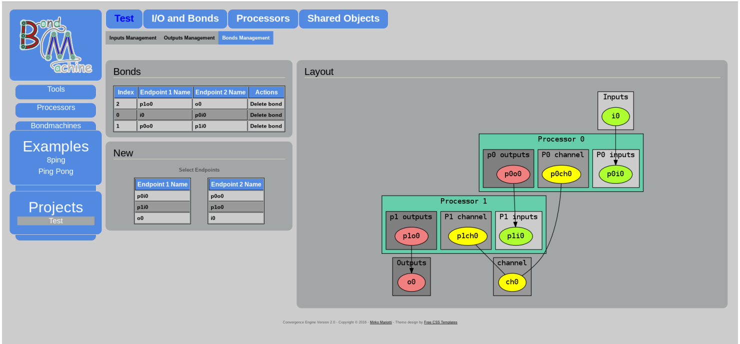
- build a specify architecture as function of the task,
- modify the created architecture,
- simulate the behavior and to check the functionality with the aim to generate the Register Transfer Level (RTL) code.

Processor Builder selects the CP specifics, assembles and disassembles, saves on disk as JSON, emulates and creates the RTL code.

BondMachine Builder connects CPs and SOs together in custom topologies, loads and saves on disk as JSON, emulates and creates the RTL code.

BondMachine web front-end

Operations on BondMachines can also be performed via a developed web framework



Simulation

An important feature of the tools is the possibility of simulating BondMachine behavior.

An event input file describes how BondMachines elements has to change during the simulation timespan and which are to be be reported.

The simulator can produce results in the form of:

- Activity log of the BM internal.
- Graphical representation of the simulation.
- Report file with quantitative data. Useful to construct metrics

Simulation

An important feature of the tools is the possibility of simulating BondMachine behavior.

An event input file describes how BondMachines elements has to change during the simulation timespan and which are to be be reported.

The simulator can produce results in the form of:

- Activity log of the BM internal.
- Graphical representation of the simulation.
- Report file with quantitative data. Useful to construct metrics

Simulation

An important feature of the tools is the possibility of simulating BondMachine behavior.

An event input file describes how BondMachines elements has to change during the simulation timespan and which are to be be reported.

The simulator can produce results in the form of:

- Activity log of the BM internal.
- Graphical representation of the simulation.
- Report file with quantitative data. Useful to construct metrics

Simulation

examples

Activity log example:

```
[discovery]> /home/mirko/Projects/conproc/tests/asm2sim % bondmachine -register-size 8 -bondmachine-file  asmtest05.json  -sim -sim-i
Loading simbox rule: config:show_pc
Loading simbox rule: config:show_ticks
Loading simbox rule: config:show_instruction
Loading simbox rule: config:show_disasm
Loading simbox rule: config:show_proc_io_pre
Loading simbox rule: config:show_proc_io_post
Loading simbox rule: config:show_proc_regs_pre
Loading simbox rule: config:show_proc_regs_post
Loading simbox rule: config:show_io_post
Loading simbox rule: config:show_io_pre
Loading simbox rule: absolute:1:set:i0:2
Absolute tick:0
        Pre-compute IO: i0: 00000000 o0: 00000000
        Proc: 0
                PC: 0
                Instr: 00000
                Disasm: i2r r0 i0
                Pre-compute IO: i0: 00000000 o0: 00000000
                Pre-compute Regs: r0: 00000000 r1: 00000000
                Post-compute IO: i0: 00000000 o0: 00000000
                Post-compute Regs: r0: 00000000 r1: 00000000
        Post-compute IO: i0: 00000000 o0: 00000000
Absolute tick:1
        Pre-compute IO: i0: 00000010 o0: 00000000
        Proc: 0
                PC: 1
                Instr: 00000
                Disasm: i2r r0 i0
                Pre-compute IO: i0: 00000010 o0: 00000000
                Pre-compute Regs: r0: 00000000 r1: 00000000
                Post-compute IO: i0: 00000010 o0: 00000000
                Post-compute Regs: r0: 00000010 r1: 00000000
        Post-compute IO: i0: 00000010 o0: 00000000
Absolute tick:2
```

A graphical example:

https://youtube.com/embed/Cc1Qzioh2Ng

Moulding the BondMachine

As said BondMachines are not general purpose architectures, and to be useful have to be shaped according the specific problem.

Several methods (apart from writing in assembly and building a BondMachine from scratch) have been developed to do that:

- *bondgo*: A new type of compiler that create not only the CPs assembly but also the architecture itself.
- A set of API to create BondMachine to suit specific computational problems.
- An Evolutionary Computation framework to “grow” BondMachines according some fitness function via simulation.
- *tf2bm*: A TensorFlow to BondMachine translator.

Moulding the BondMachine

As said BondMachines are not general purpose architectures, and to be useful have to be shaped according the specific problem.

Several methods (apart from writing in assembly and building a BondMachine from scratch) have been developed to do that:

- *bondgo*: A new type of compiler that create not only the CPs assembly but also the architecture itself.
- A set of API to create BondMachine to suit specific computational problems.
- An Evolutionary Computation framework to "grow" BondMachines according some fitness function via simulation.
- *tf2bm*: A TensorFlow to BondMachine translator.

Moulding the BondMachine

As said BondMachines are not general purpose architectures, and to be useful have to be shaped according the specific problem.

Several methods (apart from writing in assembly and building a BondMachine from scratch) have been developed to do that:

- *bondgo*: A new type of compiler that create not only the CPs assembly but also the architecture itself.
- A set of API to create BondMachine to suit specific computational problems.
- An Evolutionary Computation framework to "grow" BondMachines according some fitness function via simulation.
- *tf2bm*: A TensorFlow to BondMachine translator.

Moulding the BondMachine

As said BondMachines are not general purpose architectures, and to be useful have to be shaped according the specific problem.

Several methods (apart from writing in assembly and building a BondMachine from scratch) have been developed to do that:

- *bondgo*: A new type of compiler that create not only the CPs assembly but also the architecture itself.
- A set of API to create BondMachine to suit specific computational problems.
- An Evolutionary Computation framework to "grow" BondMachines according some fitness function via simulation.
- *tf2bm*: A TensorFlow to BondMachine translator.

Moulding the BondMachine

As said BondMachines are not general purpose architectures, and to be useful have to be shaped according the specific problem.

Several methods (apart from writing in assembly and building a BondMachine from scratch) have been developed to do that:

- *bondgo*: A new type of compiler that create not only the CPs assembly but also the architecture itself.
- A set of API to create BondMachine to suit specific computational problems.
- An Evolutionary Computation framework to “grow” BondMachines according some fitness function via simulation.
- *tf2bm*: A TensorFlow to BondMachine translator.

Moulding the BondMachine

As said BondMachines are not general purpose architectures, and to be useful have to be shaped according the specific problem.

Several methods (apart from writing in assembly and building a BondMachine from scratch) have been developed to do that:

- *bondgo*: A new type of compiler that create not only the CPs assembly but also the architecture itself.
- A set of API to create BondMachine to suit specific computational problems.
- An Evolutionary Computation framework to “grow” BondMachines according some fitness function via simulation.
- *tf2bm*: A TensorFlow to BondMachine translator.

Bondgo

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

The compiler source language is Go as the name suggest.

Bondgo

This is the standard flow when building computer programs

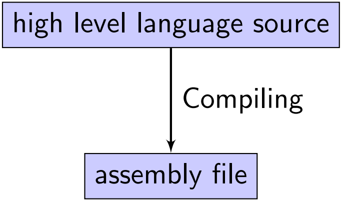
Bondgo

This is the standard flow when building computer programs

high level language source

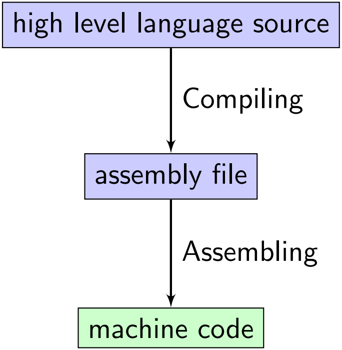
Bondgo

This is the standard flow when building computer programs



Bondgo

This is the standard flow when building computer programs



Bondgo

The standard flow in bondgo

bondgo loop example

```
package main

import ()

func main() {
   var reg_aa uint8
   var reg_ab uint8
   for reg_aa = 10; reg_aa > 0; reg_aa-- {
      reg_ab = reg_aa
      break
   }
}
```

bondgo loop example in asm

```
clr aa
clr ab
rset ac 10
cpy aa ac
cpy ac aa
jz ac 11
cpy ac aa
cpy ab ac
j 11
dec aa
j 4
```

Bondgo

bondgo may also do something different when compiling a single threaded program ...

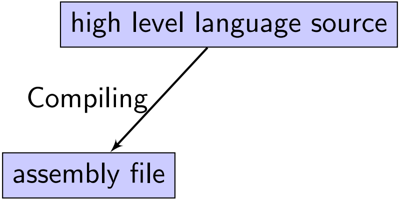
Bondgo

bondgo may also do something different when compiling a single threaded program ...

high level language source

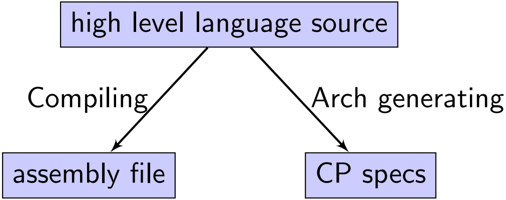
Bondgo

bondgo may also do something different when compiling a single threaded program ...



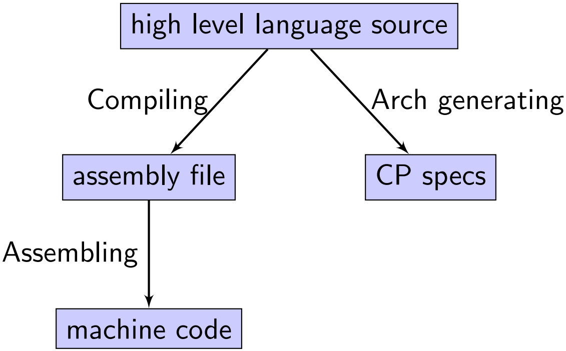
Bondgo

bondgo may also do something different when compiling a single threaded program ...



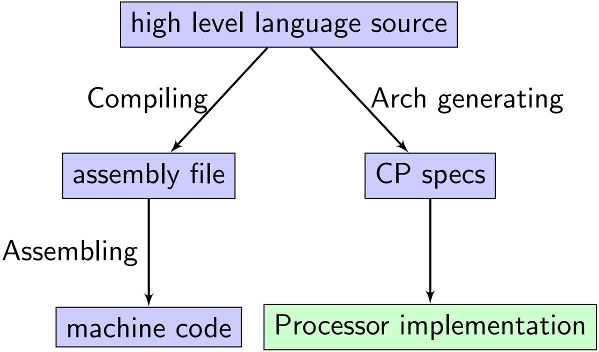
Bondgo

bondgo may also do something different when compiling a single threaded program ...



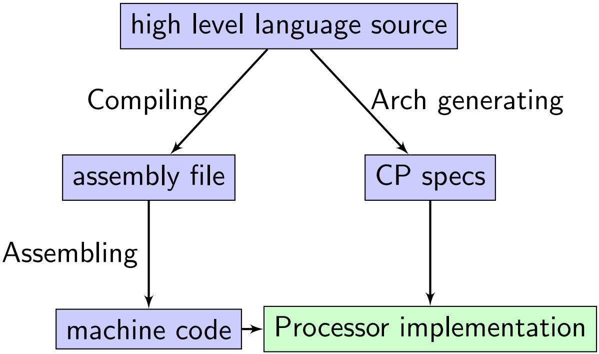
Bondgo

bondgo may also do something different when compiling a single threaded program ...



Bondgo

bondgo may also do something different when compiling a single threaded program ...



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.

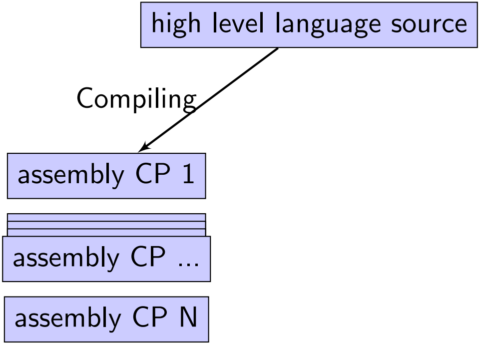
Bondgo

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

high level language source

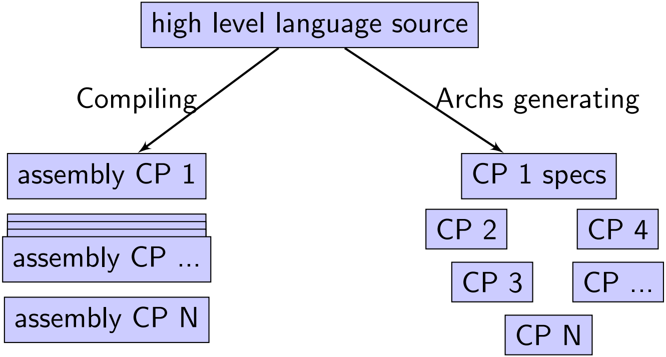
Bondgo

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



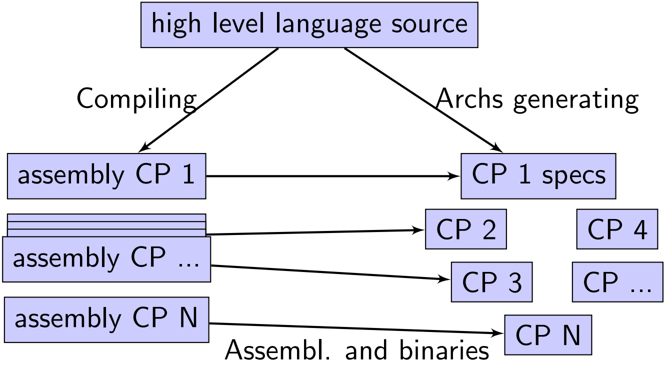
Bondgo

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



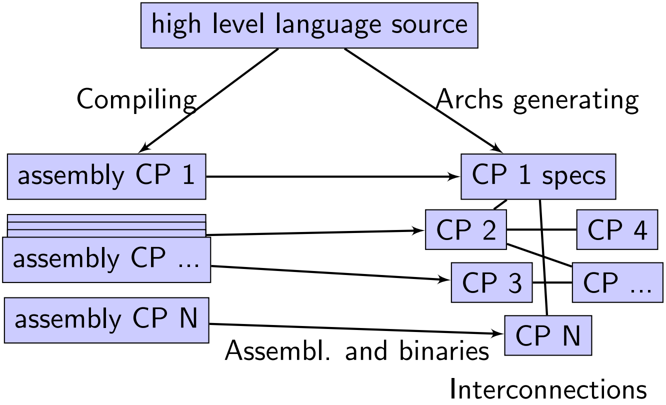
Bondgo

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



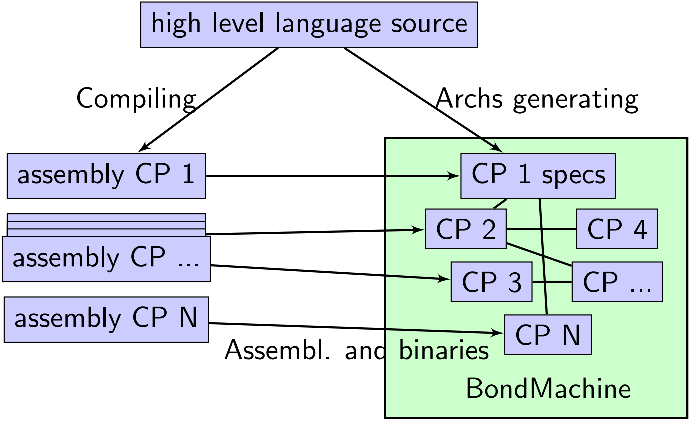
Bondgo

... 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

An example

bondgo stream processing example

```
package main

import (
   "bondgo"
)

func streamprocessor(a *[]uint8, b *[]uint8,
   c *[]uint8, gid uint8) {
   (*c)[gid] = (*a)[gid] + (*b)[gid]
}
func main() {
   a := make([]uint8, 256)
   b := make([]uint8, 256)
   c := make([]uint8, 256)

   // ... some a and b values fill

   for i := 0; i < 256; i++ {
      go streamprocessor(&a, &b, &c, uint8(i))
   }
}
```

The compilation of this example results in the creation of a 257 CPs where 256 are the stream processors executing the code in the function called *streamprocessor*, and one is the coordinating CP. Each stream processor is optimized and capable only to make additions since it is the only operation requested by the source code. The three slices created on the main function are passed by reference to the Goroutines then a shared RAM is created by the *Bondgo* compiler available to the generated CPs.

Compiling Architectures

One of the most important result

The architecture creation is a part of the compilation process.

Go in hardware

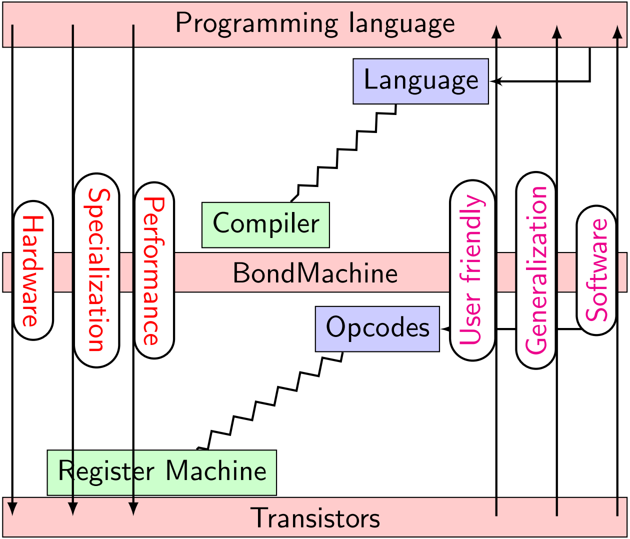
Second idea on the BondMachine

The idea was:
Build a computing system with a decreased number of layers resulting in a lower HW/SW gap.

This would raise the overall performances yet keeping an user friendly way of programming.

Between HW and SW there is only the processor abstraction, no Operating System nor runtimes. Despite that programming is done at high level.

Layers, Abstractions and Interfaces
and BondMachines



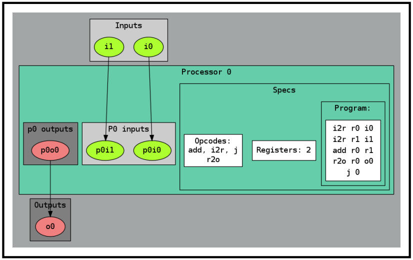
Bondgo

Abstract Assembly

The Assembly language for the BM has been kept as independent as possible from the particular CP.

Given a specific piece of assembly code Bondgo has the ability to compute the “minimum CP” that can execute that code.

```
i2r r0 i0
i2r r1 i1
add r0 r1
r2o r0 o0
j 0
```



These are Building Blocks for complex BondMachines.

Builders API

With these Building Blocks

Several libraries have to developed to map specific problems on BondMachines:

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to make matrices calculation.
- Neuralbond, to use neural networks.

Builders API

With these Building Blocks

Several libraries have to developed to map specific problems on BondMachines:

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to make matrices calculation.
- Neuralbond, to use neural networks.

Builders API

With these Building Blocks

Several libraries have to developed to map specific problems on BondMachines:

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to make matrices calculation.
- Neuralbond, to use neural networks.

Builders API

With these Building Blocks

Several libraries have to developed to map specific problems on BondMachines:

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to make matrices calculation.
- Neuralbond, to use neural networks.

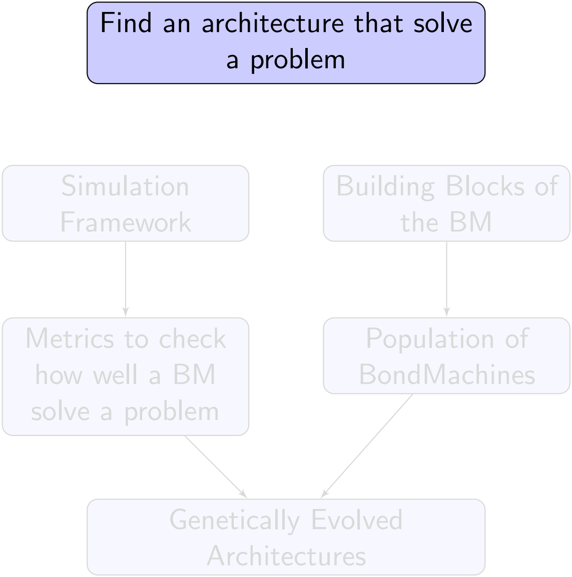
Builders API

With these Building Blocks

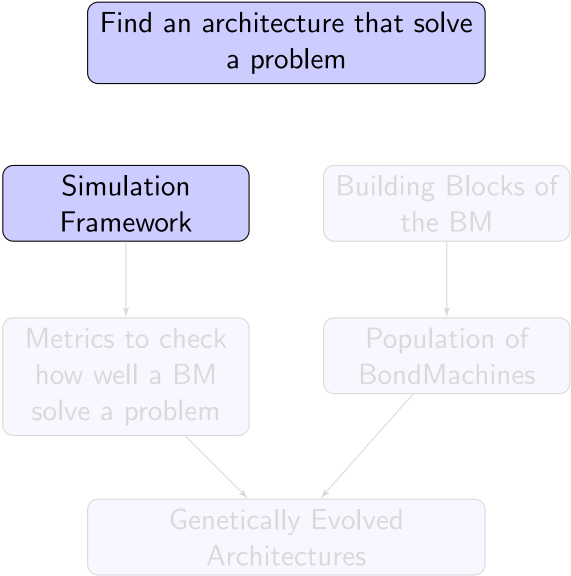
Several libraries have to developed to map specific problems on BondMachines:

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to make matrices calculation.
- Neuralbond, to use neural networks.

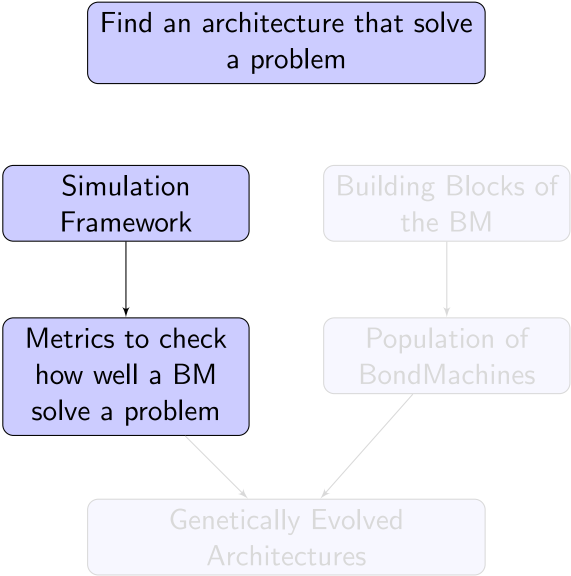
Evolutionary BondMachine



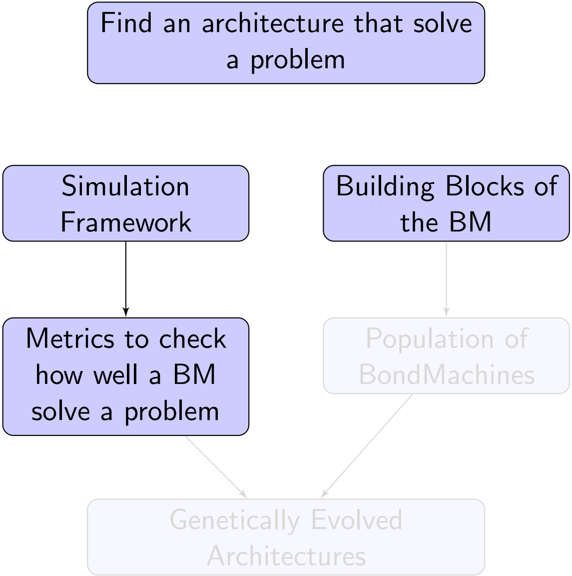
Evolutionary BondMachine



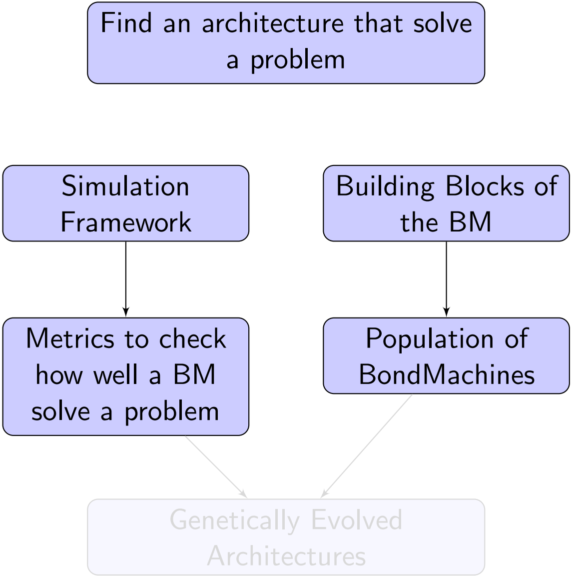
Evolutionary BondMachine



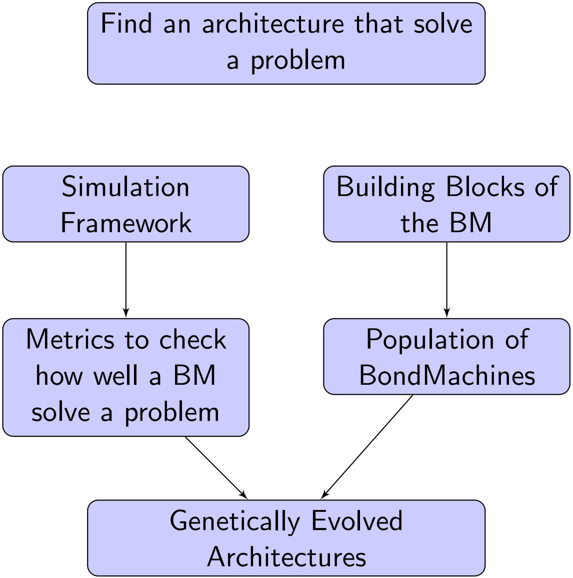
Evolutionary BondMachine



Evolutionary BondMachine



Evolutionary BondMachine

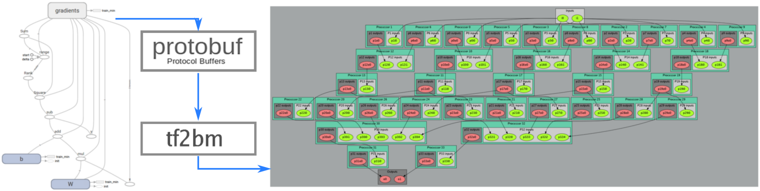


TensorFlow^{TM} to Bondmachine

tf2bm

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

Graphs can be converted to BondMachines with the tf2bm tool.



Hardware implementation
FPGA

The RTL code for the BondMachine is written in Verilog and System Verilog, and has been tested on these devices/system:

- Digilent Basys3 - Xilinx Artix-7 - Vivado.
- Kintex7 Evaluation Board - Vivado.
- Digilent Zedboard - Xilinx Zynq 7020 - Vivado.
- Linux - Iverilog.

Within the project other firmwares have been written or tested:

- Microchip ENC28J60 Ethernet interface controller.
- Microchip ENC424J600 10/100 Base-T Ethernet interface controller.
- ESP8266 Wi-Fi chip.

Toolchains

A set of toolchains allow the build and the direct deploy to a target device of BondMachines.

Bondgo Toolchain example

A file local.mk contains references to the source code as well all the build necessities.

make bondmachine creates the JSON representation of the BM and assemble its code.

make show displays a graphical representation of the BM.

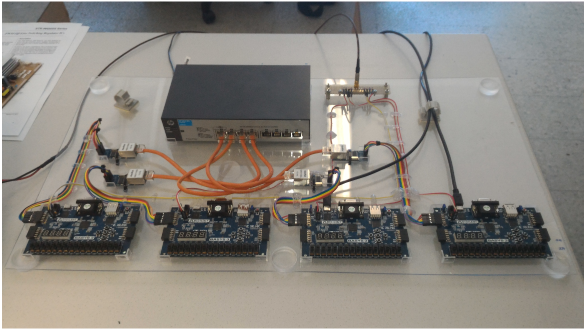
make simulate start a simulation.

make videosim create a simulation video.

make flash the device into the destination target.

The Prototype

The project has been selected for the participation at MakerFaire 2016 Rome (The Europen Edition) and a prototype has been assembled and presented.



First run:
https://youtube.com/embed/hukTrGxTb7A

The BondMachine computational Ecosystem

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

A protocol over ethernet called *etherbond* have been created for the purpose.

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

Parts of the system can be redeployed without changing the problem behavior (only the performances).

The BondMachine computational Ecosystem

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

A protocol over ethernet called *etherbond* have been created for the purpose.

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

Parts of the system can be redeployed without changing the problem behavior (only the performances).

The BondMachine computational Ecosystem

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

A protocol over ethernet called *etherbond* have been created for the purpose.

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

Parts of the system can be redeployed without changing the problem behavior (only the performances).

The BondMachine computational Ecosystem

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

A protocol over ethernet called *etherbond* have been created for the purpose.

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

Parts of the system can be redeployed without changing the problem behavior (only the performances).

Uses in Physics

Two possible uses in Physics experiments are currently being explored:

- Real time pulse shape analysis in neutron detectors.
- Test beam for space experiments (DAMPE, HERD)

Real time pulse shape analysis in neutron detectors

The operation of the new generation of high-intensity neutron sources like SNS, JSNS and European Spallation Source (ESS, Lund, Sweden), now under construction, are introducing a new demand for neutron detection capabilities.

These demands yield to the need for new data collection procedures and new technology based on solid state Si devices.

We are trying to use BondMachines to make the real time shape analysis in this kind of detecting devices.

Courtesy of Prof. F.Sacchetti

Test beam for space experiments (DAMPE, HERD)

Trigger logic for test beams

In test beams, the DAQ system relies on the trigger system for data tacking (sensor signal digitization) during

- Calibration (random trigger or "off-spill" trigger)
- On spill data taking

Minimum elements used for trigger system:

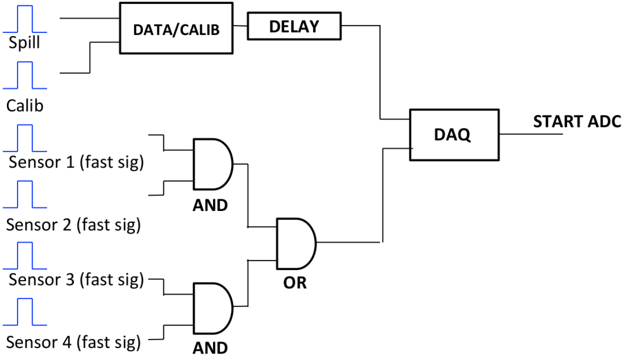
- Clock, pulser
- Logic gates (AND, OR,…)
- Delays

Trigger system implemented using NIM crates and DAQ machines

Courtesy of V.Vagelli and M.Duranti

Test beam for space experiments (DAMPE, HERD)

Trigger logic for test beams



Courtesy of V.Vagelli and M.Duranti

Test beam for space experiments (DAMPE, HERD)

Trigger logic for test beams

We are trying to explore the possibility of using BondMachine to handle efficiently this kind of operations.

Possible other uses

The BondMachine could be used in several types of real world applications, some of them being:

- Workstation FPGA accelerators for any kind of intensive computation.
- IoT and CyberPhysical systems.
- Computer Science educational applications.

Possible other uses

The BondMachine could be used in several types of real world applications, some of them being:

- Workstation FPGA accelerators for any kind of intensive computation.
- IoT and CyberPhysical systems.
- Computer Science educational applications.

Possible other uses

The BondMachine could be used in several types of real world applications, some of them being:

- Workstation FPGA accelerators for any kind of intensive computation.
- IoT and CyberPhysical systems.
- Computer Science educational applications.

Possible other uses

The BondMachine could be used in several types of real world applications, some of them being:

- Workstation FPGA accelerators for any kind of intensive computation.
- IoT and CyberPhysical systems.
- Computer Science educational applications.

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.

Keeping the register machine abstraction it is possible to borrow well known languages and techniques in programming these devices removing the need of having a general purpose architecture.

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.

Future work

- The project is a prototype.
- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking with the support for wifi.
- Work on BondMachine as accelerators.
- What would an OS for BondMachines look like ?

Future work

- The project is a prototype.
- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking with the support for wifi.
- Work on BondMachine as accelerators.
- What would an OS for BondMachines look like ?

Future work

- The project is a prototype.
- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking with the support for wifi.
- Work on BondMachine as accelerators.
- What would an OS for BondMachines look like ?

Future work

- The project is a prototype.
- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking with the support for wifi.
- Work on BondMachine as accelerators.
- What would an OS for BondMachines look like ?

Future work

- The project is a prototype.
- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking with the support for wifi.
- Work on BondMachine as accelerators.
- What would an OS for BondMachines look like ?

Future work

- The project is a prototype.
- Include new processor shared objects and currently unsupported opcodes.
- Extend the compiler to include more data structures.
- Improve the networking with the support for wifi.
- Work on BondMachine as accelerators.
- What would an OS for BondMachines look like ?

Mirko Mariotti
Department of Physics and Geology, University of Perugia and INFN PG
mirko.mariotti@unipg.it
http://bondmachine.fisica.unipg.it