The BondMachine project

Mirko Mariotti [1,2] Giulio Bianchini [1] Loriano Storchi [3,2] Giacomo Surace [2]
Daniele Spiga [2]

[1]Dipartimento di Fisica e Geologia, Universitá degli Studi di Perugia

[2]INFN sezione di Perugia

[3]Dipartimento di Farmacia, Universitá degli Studi G. D'Annunzio

Outline

Demo sessions

Some topic will have a demo session.

The code will be available at:

`http://bondmachine.fisica.unipg.it`

Requirements:

- Linux Workstation
- Vivado
- Zedboard

Introduction

Current challenges in computing

- Von Neumann Bottleneck:
  New computational problems show that current architectural models has to be improved or changed to address future payloads.

- Energy Efficient computation:
  Not wasting "resources" (silicon, time, energy, instructions).
  Using the right resource for the specific case

- Edge/Fog/Cloud Computing:
  Making the computation where it make sense
  Avoiding the transfer of unnecessary data
  Creating consistent interfaces for distributed systems

Current challenges in computing

- Von Neumann Bottleneck:
  New computational problems show that current architectural models has to be improved or changed to address future payloads.

- Energy Efficient computation:
  Not wasting "resources" (silicon, time, energy, instructions).
  Using the right resource for the specific case

- Edge/Fog/Cloud Computing:
  Making the computation where it make sense
  Avoiding the transfer of unnecessary data
  Creating consistent interfaces for distributed systems

Current challenges in computing

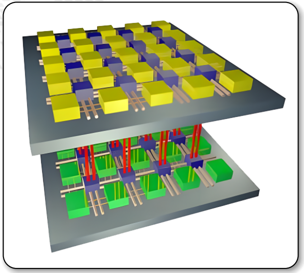
- Von Neumann Bottleneck:
  New computational problems show that current architectural models has to be improved or changed to address future payloads.

- Energy Efficient computation:
  Not wasting "resources" (silicon, time, energy, instructions).
  Using the right resource for the specific case

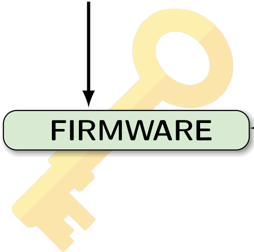
- Edge/Fog/Cloud Computing:
  Making the computation where it make sense
  Avoiding the transfer of unnecessary data
  Creating consistent interfaces for distributed systems

FPGA

A field programmable gate array (FPGA) is an integrated circuit whose logic is re-programmable.



- Parallel computing
- Highly specialized
- Energy efficient



- Array of programmable logic blocks
- Logic blocks configurable to perform complex functions
- The configuration is specified with the hardware description language

FPGA

Use in computing

The use of FPGA in computing is growing due several reasons:

- can potentially deliver great performance via massive parallelism
- can address payloads which are not performing well on uniprocessors (Neural Networks, Deep Learning)
- can handle efficiently non-standard data types

FPGA

Use in computing

The use of FPGA in computing is growing due several reasons:

- can potentially deliver great performance via massive parallelism
- can address payloads which are not performing well on uniprocessors (Neural Networks, Deep Learning)
- can handle efficiently non-standard data types

FPGA

Use in computing

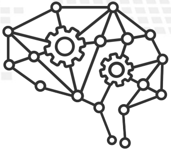
The use of FPGA in computing is growing due several reasons:

- can potentially deliver great performance via massive parallelism
- can address payloads which are not performing well on uniprocessors (Neural Networks, Deep Learning)
- can handle efficiently non-standard data types

Integration of neural networks on FPGA

FPGAs are playing an increasingly important role in the industry sampling and data processing.

AMD
XILINX



Deep Learning

intel
ALTERA

In the industrial field

- Intelligent vision;
- Financial services;
- Scientific simulations;
- Life science and medical data analysis;

In the scientific field

- Real time deep learning in particle physics;
- Hardware trigger of LHC experiments;
- And many others ...

FPGA

Challenges in computing

On the other hand the adoption on FPGA poses several challenges:

- Porting of legacy code is usually hard.
- Interoperability with standard applications is problematic.

FPGA

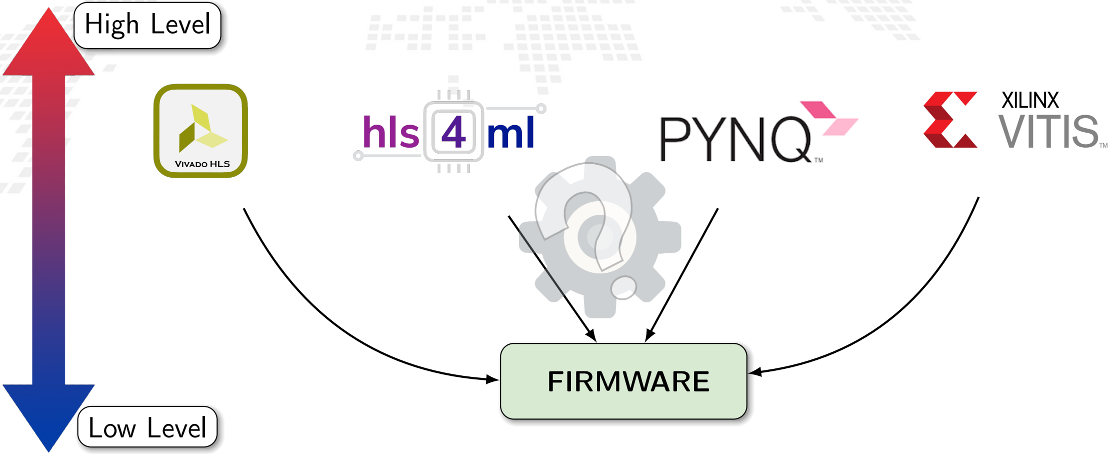
Challenges in computing

On the other hand the adoption on FPGA poses several challenges:

- Porting of legacy code is usually hard.
- Interoperability with standard applications is problematic.

Firmware generation

Many projects have the goal of abstracting the firmware generation and use process.



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 payloads scheduling has to be addressed.

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 payloads 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 payloads scheduling has to be addressed.

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 payloads 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 payloads 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 payloads 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 payloads scheduling has to be addressed.

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 payloads 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 payloads 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 payloads scheduling has to be addressed.

The BondMachine

First 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

First 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

First 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.

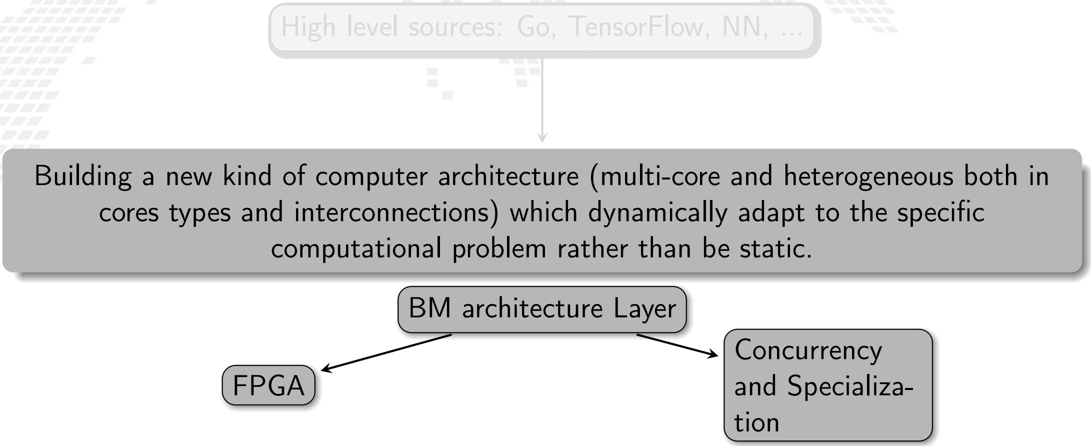
BM architecture Layer

FPGA

Concurrency and Specialization

The BondMachine

First idea



The BondMachine

First 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

Layer, Abstractions and Interfaces

A Computing system is a matter of abstraction and interfaces. A lower layer exposes its functionalities (via interfaces) to the above layer hiding (abstraction) its inner details.

The quality of a computing system is determined by how abstractions are simple and how interfaces are clean.

Layers, Abstractions and Interfaces

An example

Programming language

User mode

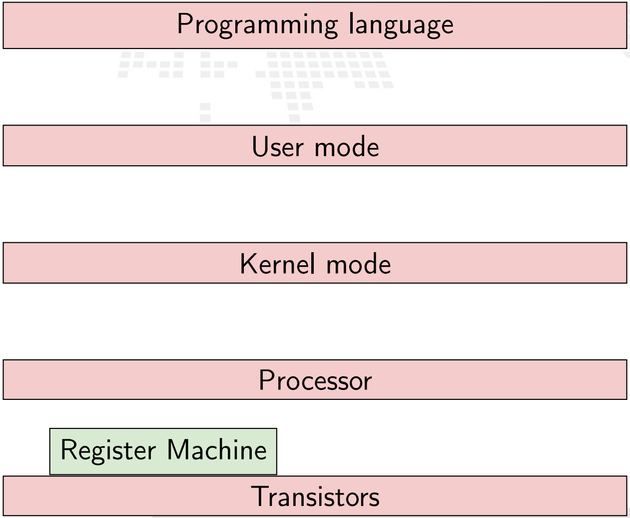
Kernel mode

Processor

Transistors

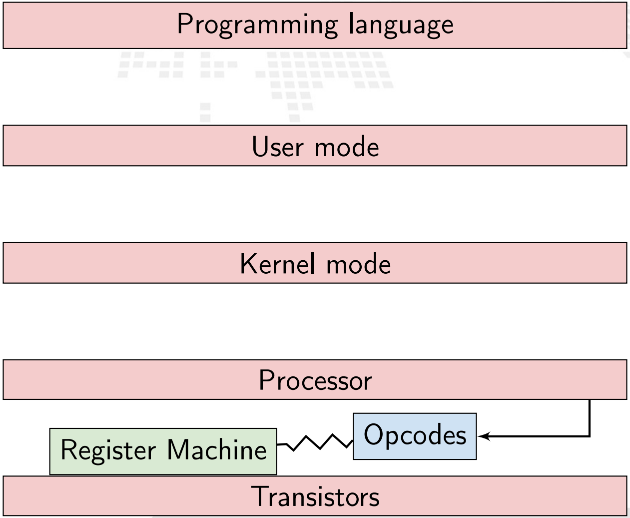
Layers, Abstractions and Interfaces

An example



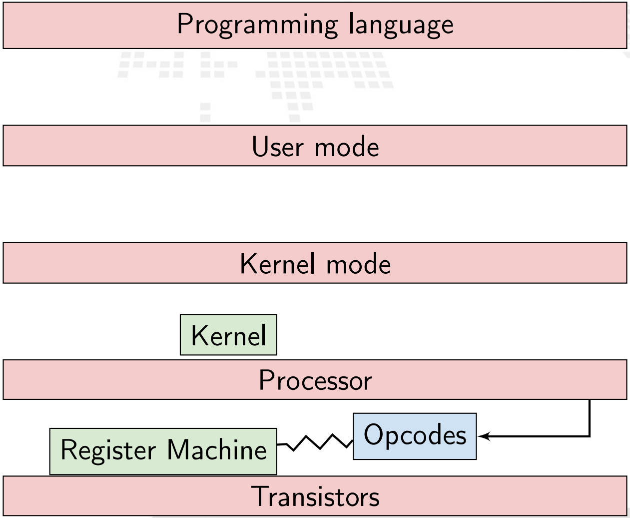
Layers, Abstractions and Interfaces

An example



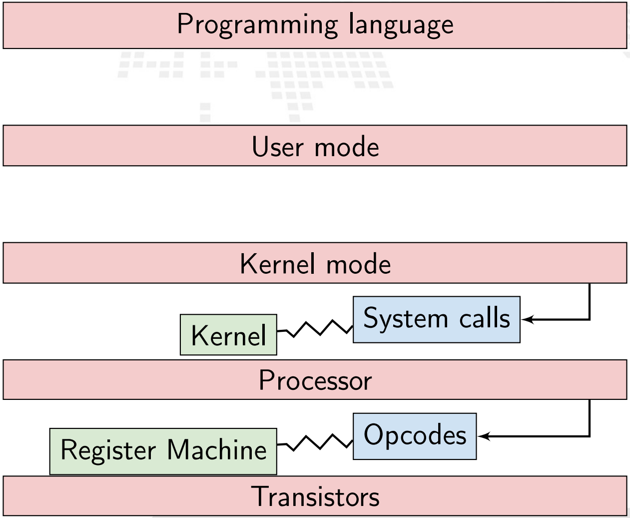
Layers, Abstractions and Interfaces

An example



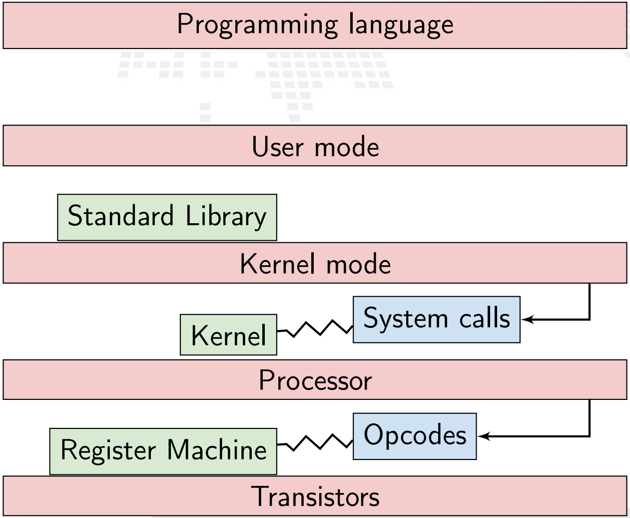
Layers, Abstractions and Interfaces

An example



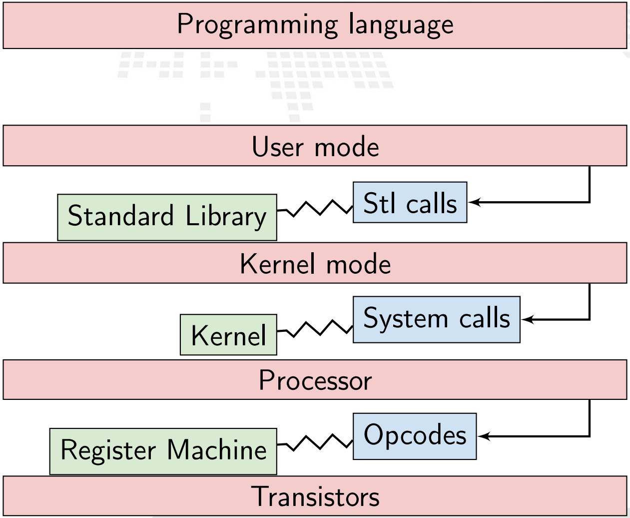
Layers, Abstractions and Interfaces

An example



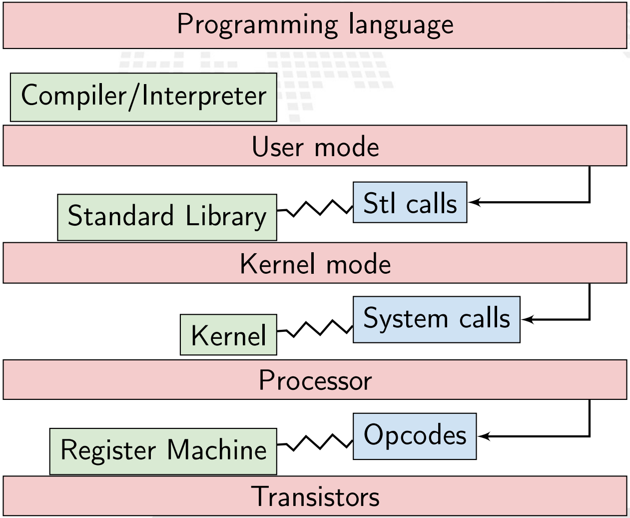
Layers, Abstractions and Interfaces

An example



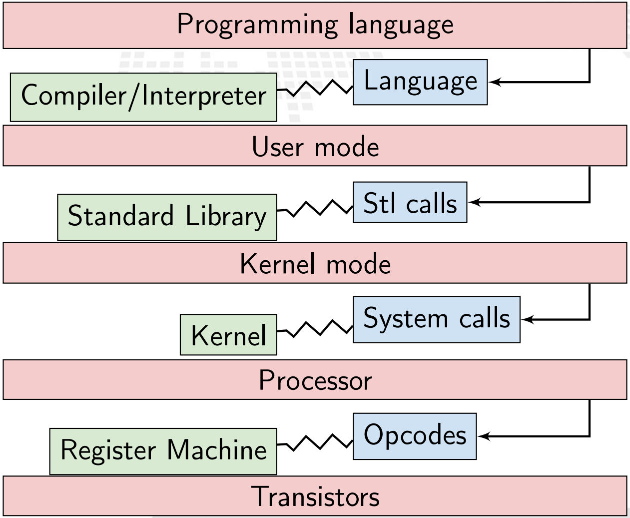
Layers, Abstractions and Interfaces

An example



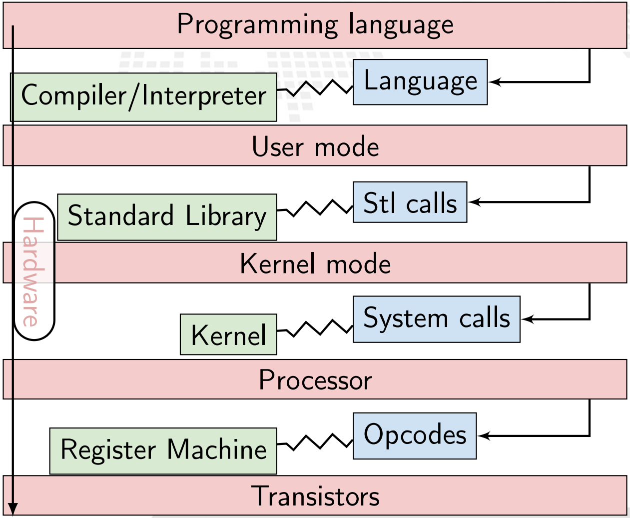
Layers, Abstractions and Interfaces

An example



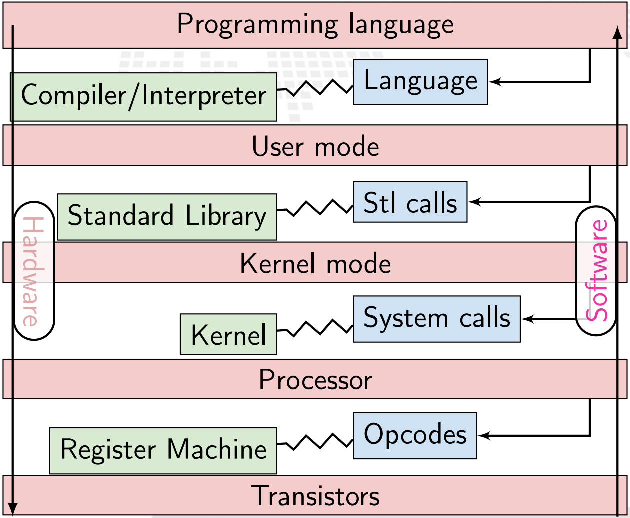
Layers, Abstractions and Interfaces

An example



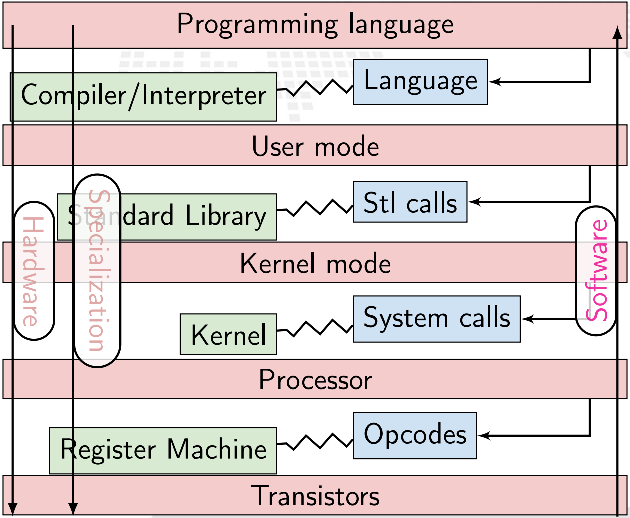
Layers, Abstractions and Interfaces

An example



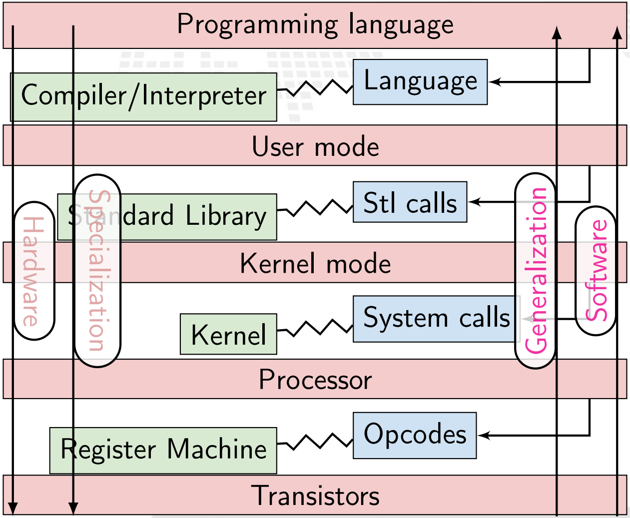
Layers, Abstractions and Interfaces

An example



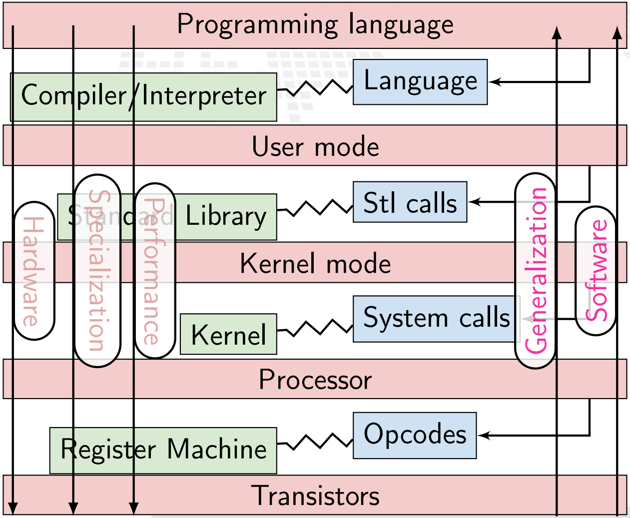
Layers, Abstractions and Interfaces

An example



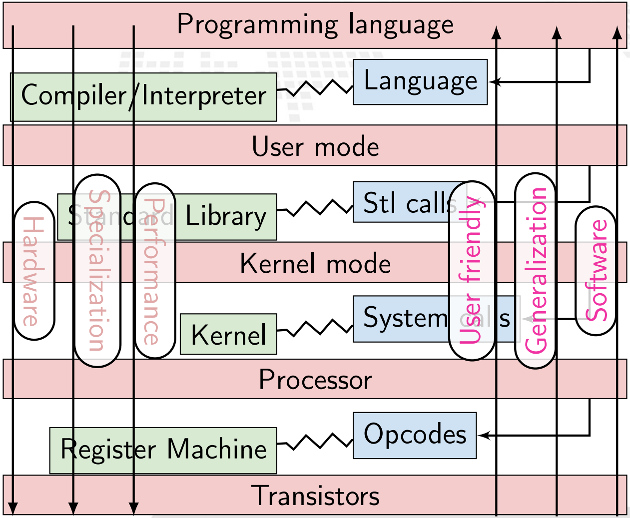
Layers, Abstractions and Interfaces

An example



Layers, Abstractions and Interfaces

An example



Layers, Abstractions and Interfaces

The second idea

Rethinking the stack

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.

The BondMachine project

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).

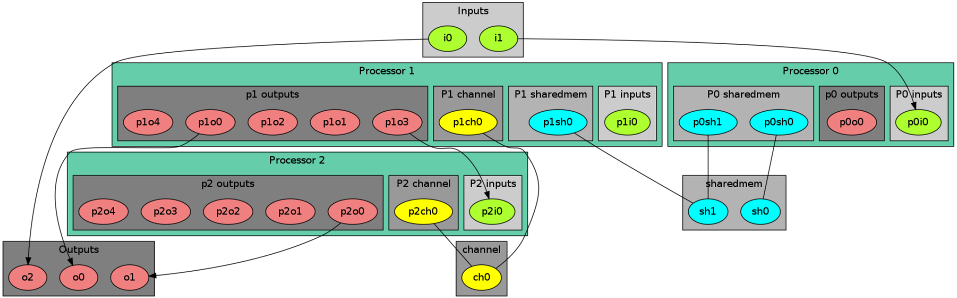
Introducing the BondMachine (BM)

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.
- Three 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.
- Three possible operating modes.

General purpose registers

2^R registers: r0,r1,r2,r3 ... $r2^R$

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.
- Three possible operating modes.

I/O specialized registers

N input registers: i0,i1 ... iN
M output registers: o0,o1 ... oM

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.
- Three possible operating modes.

Full set of possible opcodes

adc,add,addf,addi,and,chc,chw,cil,cilc,cir,cirn,clc,clr,cpy,cset,dec,div,divf,dpc,expf,hit
hlt,i2r,i2rw,incc,inc,j,jc,je,jgt0f,jlt,jlte,jr,jz,lfsr82,lfsr162r,m2r,mod,mulc,mult,multf
nand,nop,nor,not,or,r2m,r2o,r2owa,r2owaa,r2s,r2v,r2vri,ro2r,ro2rri,rsc,rset,sic,s2r,saj,sbc
sub,wrd,wwr,xnor,xor

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.
- Three possible operating modes.

RAM and ROM

- 2^L RAM memory cells.
- 2^O ROM memory cells.

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.
- Three possible operating modes.

Operating modes

- Full Harvard mode.
- Full Von Neuman mode.
- Hybrid mode.

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.

Channel

The Channel SO is an hardware implementation of the CSP (communicating sequential processes) channel.

It is a model for inter-core communication and synchronization via message passing.

CPs use channels via 4 opcodes

- *wrd*: Want Read.
- *wwr*: Want Write.
- *chc*: Channel Check.
- *chw*: Channel Wait.

Channel

The Channel SO is an hardware implementation of the CSP (communicating sequential processes) channel.

It is a model for inter-core communication and synchronization via message passing.

CPs use channels via 4 opcodes

- *wrd*: Want Read.
- *wwr*: Want Write.
- *chc*: Channel Check.
- *chw*: Channel Wait.

Channel

The Channel SO is an hardware implementation of the CSP (communicating sequential processes) channel.

It is a model for inter-core communication and synchronization via message passing.

CPs use channels via 4 opcodes

- *wrd*: Want Read.
- *wwr*: Want Write.
- *chc*: Channel Check.
- *chw*: Channel Wait.

Shared Memory

The Shared Memory SO is a RAM block accessible from more than one CP.

Different Shared Memories can be used by different CP and not necessarily by all of them.

CPs use shared memories via 2 opcodes

- *s2r*: Shared memory read.
- *r2s*: Shared memory write.

Shared Memory

The Shared Memory SO is a RAM block accessible from more than one CP.

Different Shared Memories can be used by different CP and not necessarily by all of them.

CPs use shared memories via 2 opcodes

- *s2r*: Shared memory read.
- *r2s*: Shared memory write.

Shared Memory

The Shared Memory SO is a RAM block accessible from more than one CP.

Different Shared Memories can be used by different CP and not necessarily by all of them.

CPs use shared memories via 2 opcodes

- *s2r*: Shared memory read.
- *r2s*: Shared memory write.

Barrier

The Barrier SO is used to make CPs act synchronously.

When a CP hits a barrier, the execution stop until all the CPs that share the same barrier hit it.

CPs use barriers via 1 opcode

- *hit*: Hit the barrier.

Barrier

The Barrier SO is used to make CPs act synchronously.

When a CP hits a barrier, the execution stop until all the CPs that share the same barrier hit it.

CPs use barriers via 1 opcode

- *hit*: Hit the barrier.

Barrier

The Barrier SO is used to make CPs act synchronously.

When a CP hits a barrier, the execution stop until all the CPs that share the same barrier hit it.

CPs use barriers via 1 opcode

- *hit*: Hit the barrier.

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, arbitrarily interconnected and sharing non computing elements.

Handle BM computer architectures

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 Hardware Description Language Code (HDL)

Processor Builder

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

BondMachine Builder

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

Simulation Framework

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

Handle BM computer architectures

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 Hardware Description Language Code (HDL)

Processor Builder

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

BondMachine Builder

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

Simulation Framework

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

Handle BM computer architectures

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 Hardware Description Language Code (HDL)

Processor Builder

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

BondMachine Builder

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

Simulation Framework

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

Handle BM computer architectures

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 Hardware Description Language Code (HDL)

Processor Builder

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

BondMachine Builder

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

Simulation Framework

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

Processor Builder

Procbuilder is the CP manipulation tool.

- CP Creation
- CP Load/Save
- CP Assembler/Disassembler
- CP HDL

Examples

(32 bit registers counter machine)

procbuilder -register-size 32 -opcodes clr,cpy,dec,inc,je,jz

(Input and Output registers)

procbuilder -inputs 3 -outputs 2 ...

Processor Builder

Procbuilder is the CP manipulation tool.

- CP Creation
- CP Load/Save
- CP Assembler/Disassembler
- CP HDL

Examples

(Loading a CP)

procbuilder -load-machine conproc.json ...

(Saving a CP)

procbuilder -save-machine conproc.json ...

Processor Builder

Procbuilder is the CP manipulation tool.

- CP Creation
- CP Load/Save
- CP Assembler/Disassembler
- CP HDL

Examples

(Assembiling)

procbuilder -input-assembly program.asm ...

(Disassembling)

procbuilder -show-program-disassembled ...

Processor Builder

Procbuilder is the CP manipulation tool.

- CP Creation
- CP Load/Save
- CP Assembler/Disassembler
- CP HDL

Examples

(Create the CP RTL code in Verilog)

procbuilder -create-verilog ...

(Create testbench)

procbuilder -create-verilog-testbench test.v ...

Procbuilder demo

Demo N.1

It will be shown how:

- To create a simple processor
- To assemble and disassemble code for it
- To produce its HDL code

BondMachine Builder

Bondmachine is the tool that compose CP and SO to form BondMachines.

- BM CP insert and remove
- BM SO insert and remove
- BM Inputs and Outputs
- BM Bonding Processors and/or IO
- BM Visualizing or HDL

Examples

(Add a processor)

bondmachine -add-domains proc.json ... ; ... -add-processor 0

(Remove a processor)

bondmachine -bondmachine-file bmach.json -del-processor n

BondMachine Builder

Bondmachine is the tool that compose CP and SO to form BondMachines.

- BM CP insert and remove
- BM SO insert and remove
- BM Inputs and Outputs
- BM Bonding Processors and/or IO
- BM Visualizing or HDL

Examples

(Add a Shared Object)

bondmachine -add-shared-objects specs ...

(Connect an SO to a processor)

bondmachine -connect-processor-shared-object ...

BondMachine Builder

Bondmachine is the tool that compose CP and SO to form BondMachines.

- BM CP insert and remove
- BM SO insert and remove
- BM Inputs and Outputs
- BM Bonding Processors and/or IO
- BM Visualizing or HDL

Examples

(Adding inputs or outputs)

bondmachine -add-inputs ... ; bondmachine -add-outputs ...

(Removing inputs or outputs)

bondmachine -del-input ... ; bondmachine -del-output ...

BondMachine Builder

Bondmachine is the tool that compose CP and SO to form BondMachines.

- BM CP insert and remove
- BM SO insert and remove
- BM Inputs and Outputs
- BM Bonding Processors and/or IO
- BM Visualizing or HDL

Examples

(Bonding processor)

bondmachine -add-bond p0i2,p1o4 ...

(Bonding IO)

bondmachine -add-bond i2,p0i6 ...

BondMachine Builder

Bondmachine is the tool that compose CP and SO to form BondMachines.

- BM CP insert and remove
- BM SO insert and remove
- BM Inputs and Outputs
- BM Bonding Processors and/or IO
- BM Visualizing or HDL

Examples

(Visualizing)

bondmachine -emit-dot ...

(Create RTL code)

bondmachine -create-verilog ...

BondMachine demo

Demo N.2

It will be shown how:

- To create a single-core BondMachine
- To attach an external output
- To produce its HDL code

Toolchains

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

Bondgo Toolchain main targets

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 hdl creates the HDL files of the BM
make show displays a graphical representation of the BM
make simulate [simbatch] start a simulation [batch simulation]
make bitstream [design_bitstream] create the firwware [accelerator firmware]
make program flash the device into the destination target

BondMachine demo

Demo N.3

It will be shown how:

- To explore the toolchain
- To flash the board with the code from the previous example

BondMachine demo

Demo N.4

It will be shown how:

- To build a BondMachine with a processor and a shared object
- To flash the board

BondMachine demo

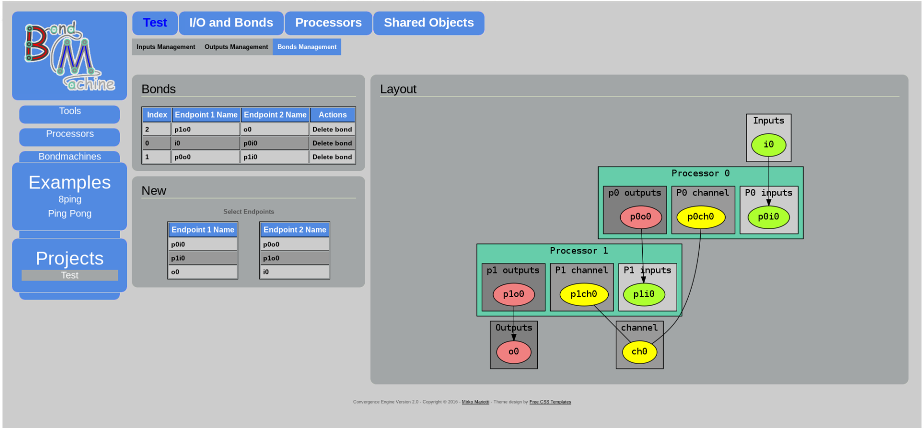
Demo N.5

It will be shown how:

- To build a dual-core BondMachine
- To connect cores
- To flash the board

BondMachine web front-end

Operations on BondMachines can also be performed via an under development 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 one has 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

Graphical simulation in action

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 one has 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

Graphical simulation in action

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 one has 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

Graphical simulation in action

Simulation demo

Demo N.6

It will be shown how:

- To show the simulation capabilities of the framework

Emulation

The simulation facility is not necessarily used for debug purposes, it can be used also to run payloads without having a real FPGA.

The same engine that simulate BondMachines can be used as emulator.

Through the emulator BondMachines can be used on Linux workstations.

Molding the BondMachine

As stated before BondMachines are not general purpose architectures, and to be effective 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.
- *basm*: The BondMachine Assembler.
- A set of API to create BondMachine to fit a specific computational problems.
- An Evolutionary Computation framework to "grow" BondMachines according some fitness function via simulation.
- A set of tools to use BondMachine in Machine Learning.

Molding the BondMachine

As stated before BondMachines are not general purpose architectures, and to be effective 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.
- *basm*: The BondMachine Assembler.
- A set of API to create BondMachine to fit a specific computational problems.
- An Evolutionary Computation framework to “grow” BondMachines according some fitness function via simulation.
- A set of tools to use BondMachine in Machine Learning.

Molding the BondMachine

As stated before BondMachines are not general purpose architectures, and to be effective 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.
- *basm*: The BondMachine Assembler.
- A set of API to create BondMachine to fit a specific computational problems.
- An Evolutionary Computation framework to "grow" BondMachines according some fitness function via simulation.
- A set of tools to use BondMachine in Machine Learning.

Molding the BondMachine

As stated before BondMachines are not general purpose architectures, and to be effective 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.
- *basm*: The BondMachine Assembler.
- A set of API to create BondMachine to fit a specific computational problems.
- An Evolutionary Computation framework to "grow" BondMachines according some fitness function via simulation.
- A set of tools to use BondMachine in Machine Learning.

Molding the BondMachine

As stated before BondMachines are not general purpose architectures, and to be effective 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.
- *basm*: The BondMachine Assembler.
- A set of API to create BondMachine to fit a specific computational problems.
- An Evolutionary Computation framework to “grow” BondMachines according some fitness function via simulation.
- A set of tools to use BondMachine in Machine Learning.

Molding the BondMachine

As stated before BondMachines are not general purpose architectures, and to be effective 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.
- *basm*: The BondMachine Assembler.
- A set of API to create BondMachine to fit a specific computational problems.
- An Evolutionary Computation framework to "grow" BondMachines according some fitness function via simulation.
- A set of tools to use BondMachine in Machine Learning.

Molding the BondMachine

As stated before BondMachines are not general purpose architectures, and to be effective 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.
- *basm*: The BondMachine Assembler.
- A set of API to create BondMachine to fit a specific computational problems.
- An Evolutionary Computation framework to “grow” BondMachines according some fitness function via simulation.
- A set of tools to use BondMachine in Machine Learning.

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

Basm

The BondMachine assembler

Bondgo

The architecture compiler

ML tools

Map computational graphs to BM

more about these tools

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

Basm

The BondMachine assembler

Bondgo

The architecture compiler

ML tools

Map computational graphs to BM

more about these tools

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

ML tools

Map computational graphs to BM

Basm

The BondMachine assembler

Bondgo

The architecture compiler

more about these tools

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

Basm

The BondMachine assembler

Bondgo

The architecture compiler

ML tools

Map computational graphs to BM

more about these tools

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

Basm

The BondMachine assembler

Bondgo

The architecture compiler

ML tools

Map computational graphs to BM

more about these tools

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

Basm

The BondMachine assembler

Bondgo

The architecture compiler

ML tools

Map computational graphs to BM

more about these tools

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

Basm
The BondMachine assembler

Bondgo
The architecture compiler

ML tools
Map computational graphs to BM

more about these tools

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

This is the standard flow when building computer programs

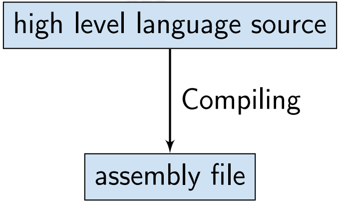
Bondgo

This is the standard flow when building computer programs

high level language source

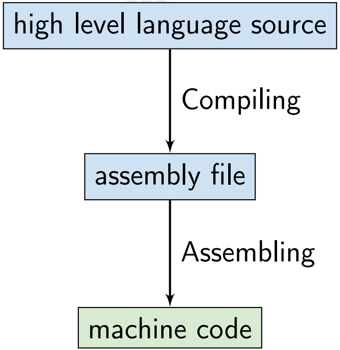
Bondgo

This is the standard flow when building computer programs



Bondgo

This is the standard flow when building computer programs



Bondgo

Bondgo does something different from standard compilers ...

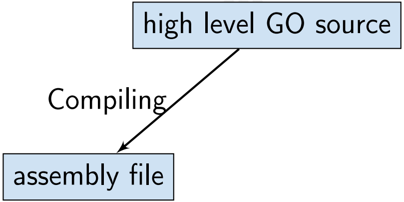
Bondgo

Bondgo does something different from standard compilers ...

high level GO source

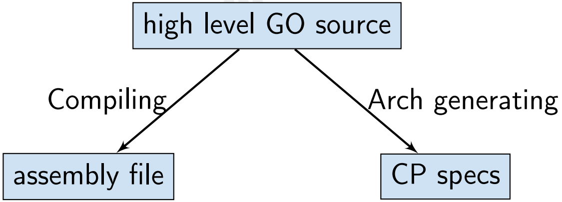
Bondgo

Bondgo does something different from standard compilers ...



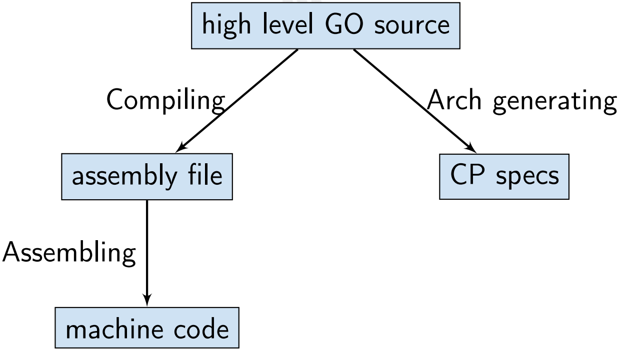
Bondgo

Bondgo does something different from standard compilers ...



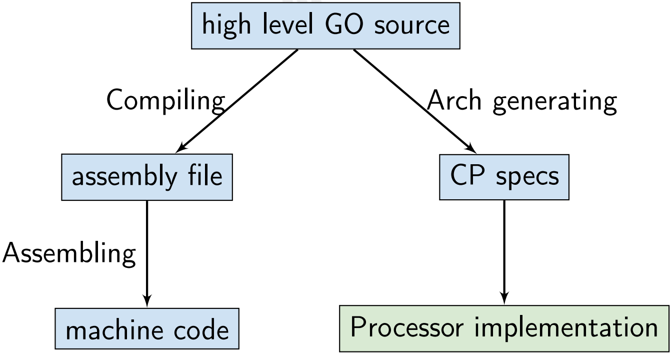
Bondgo

Bondgo does something different from standard compilers ...



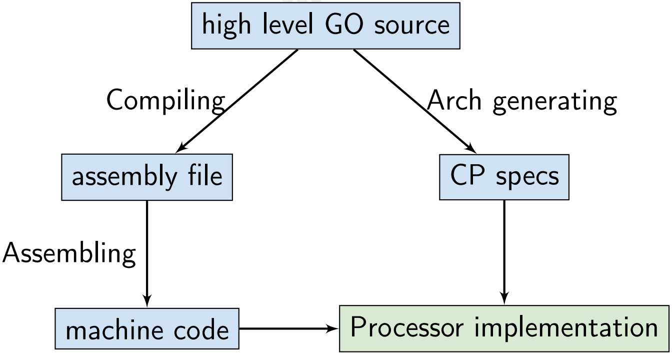
Bondgo

Bondgo does something different from standard compilers ...



Bondgo

Bondgo does something different from standard compilers ...



Bondgo workflow example

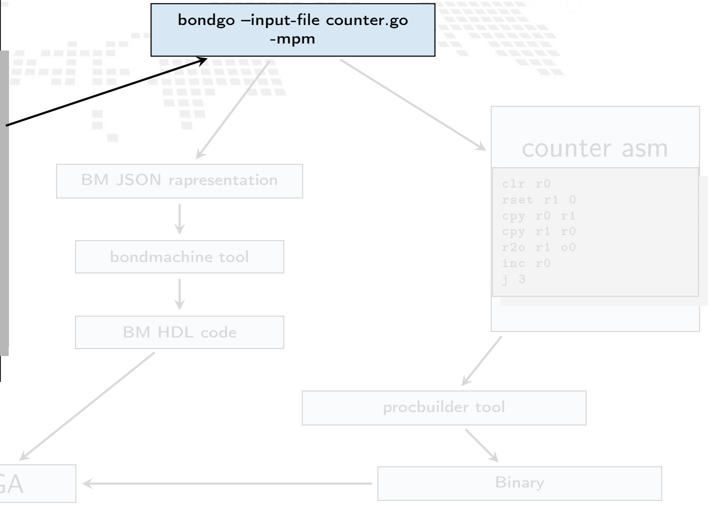
counter go source

```
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 workflow example

counter go source

```go
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 workflow example

counter go source

```go
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

counter asm

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

BM JSON rapresentation

bondmachine tool

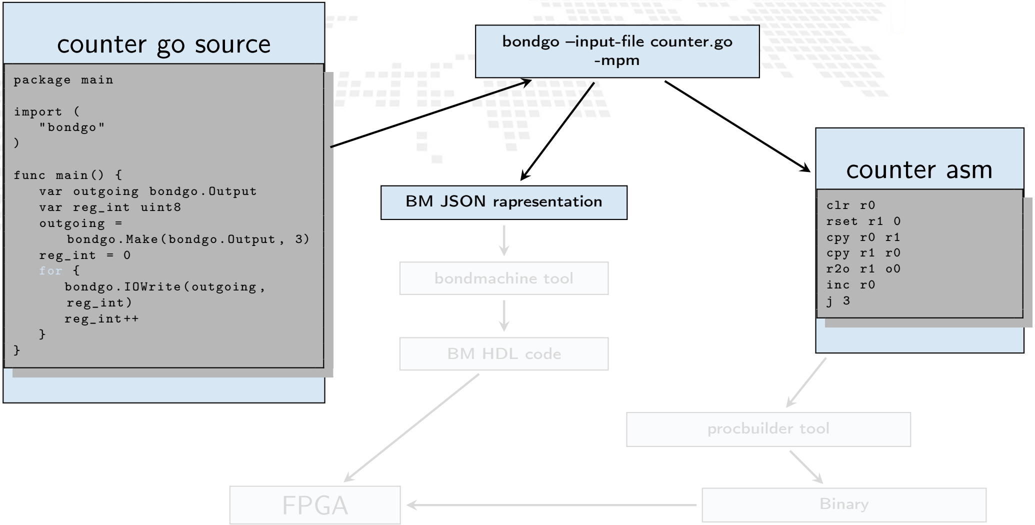
BM HDL code

procbuilder tool

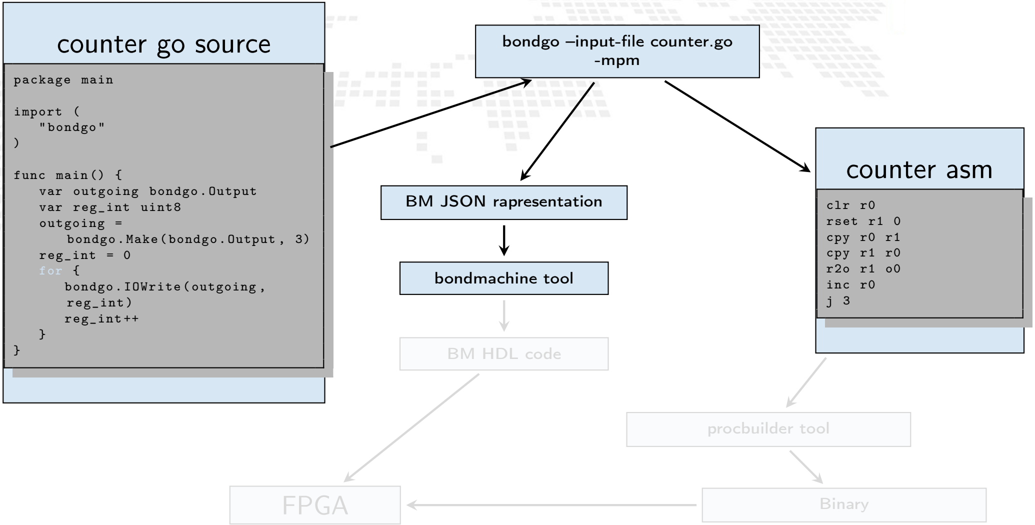
Binary

FPGA

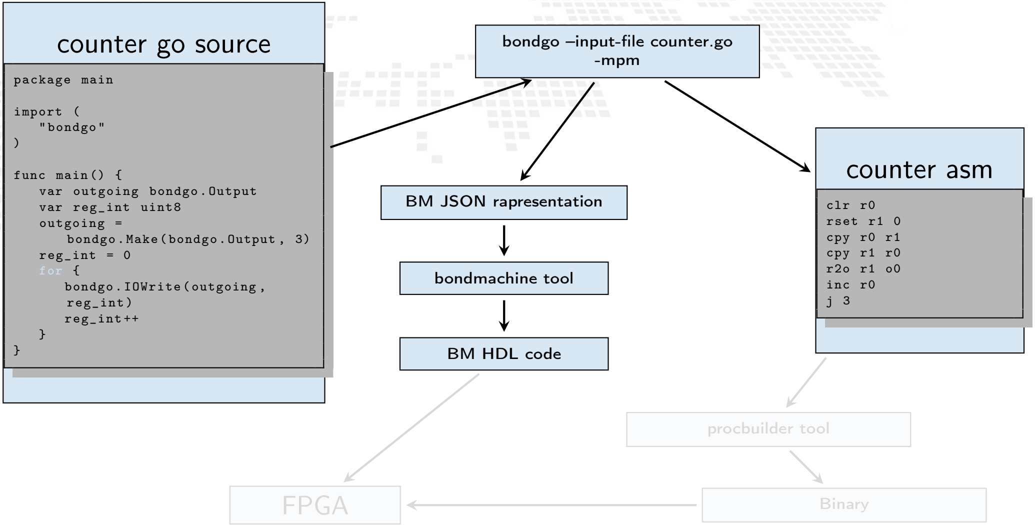
Bondgo workflow example



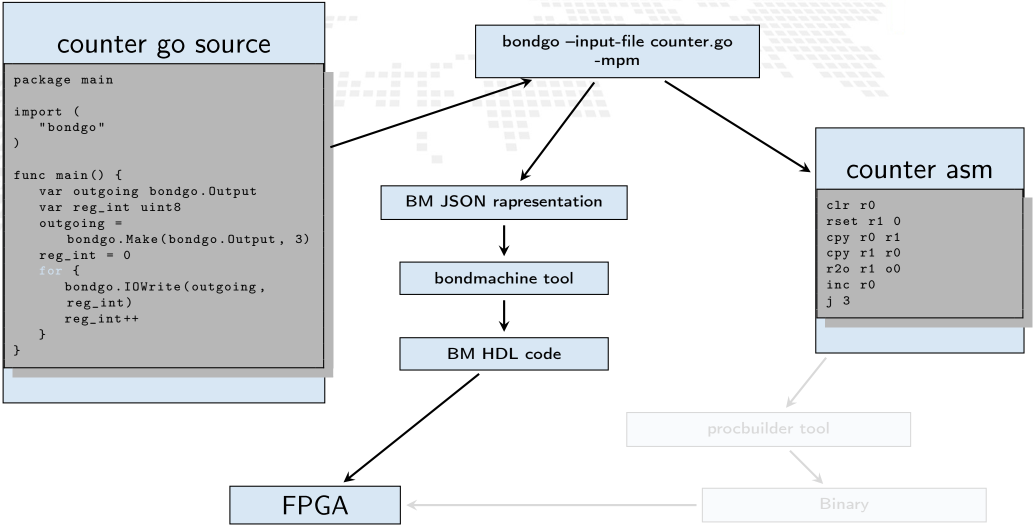
Bondgo workflow example



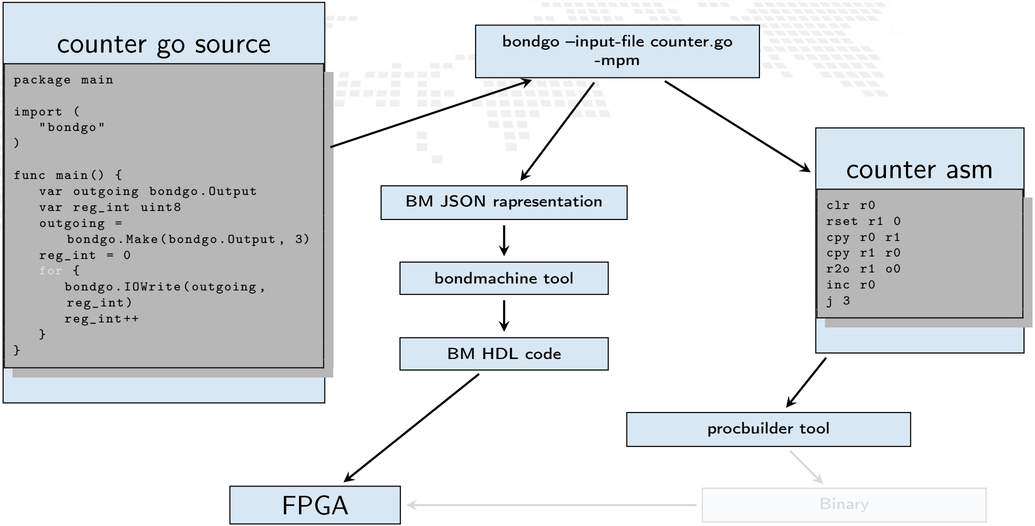
Bondgo workflow example



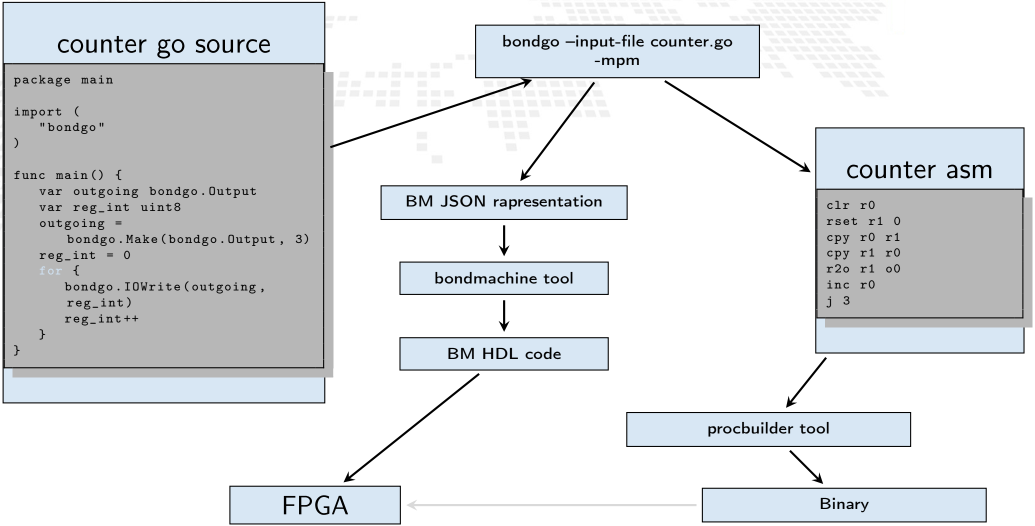
Bondgo workflow example



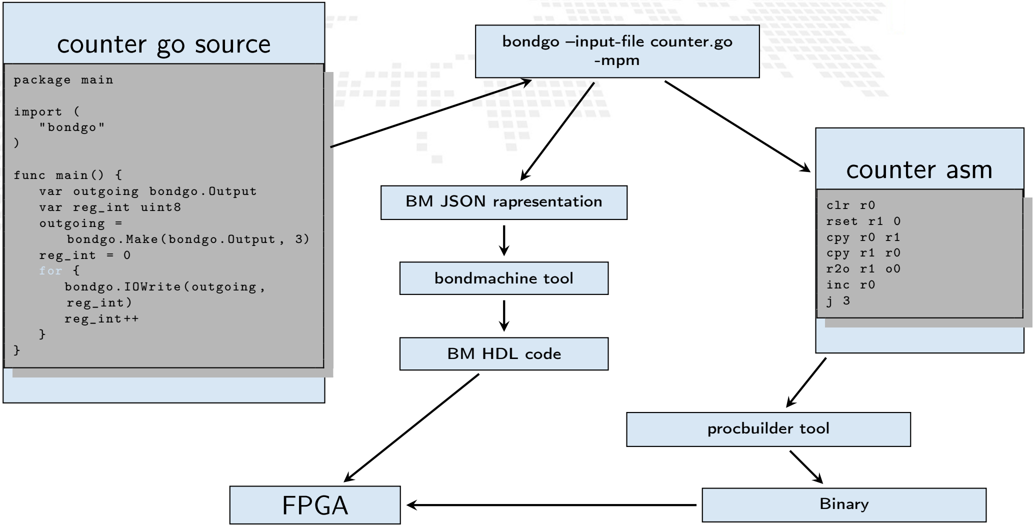
Bondgo workflow example



Bondgo workflow example



Bondgo workflow example



Bondgo

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

Bondgo demo

Demo N.7

It will be shown how:

- To create a BondMachine from a Go source file
- To build the architecture
- To build the program
- To create the firmware and flash it to the board

Bondgo

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

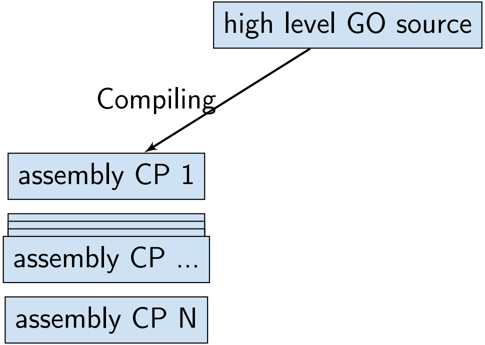
Bondgo

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

high level GO source

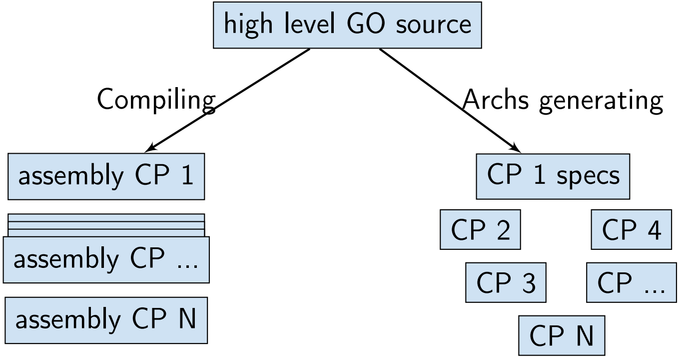
Bondgo

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



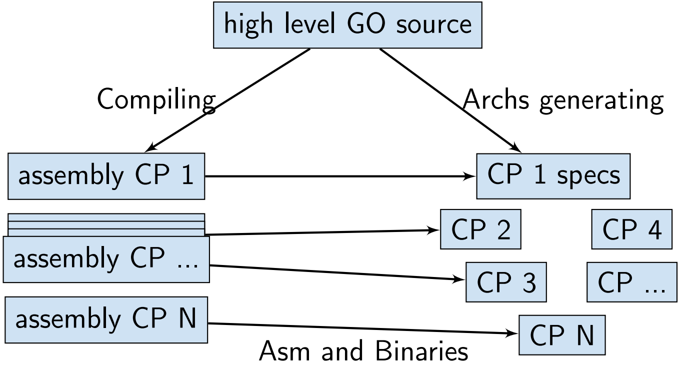
Bondgo

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



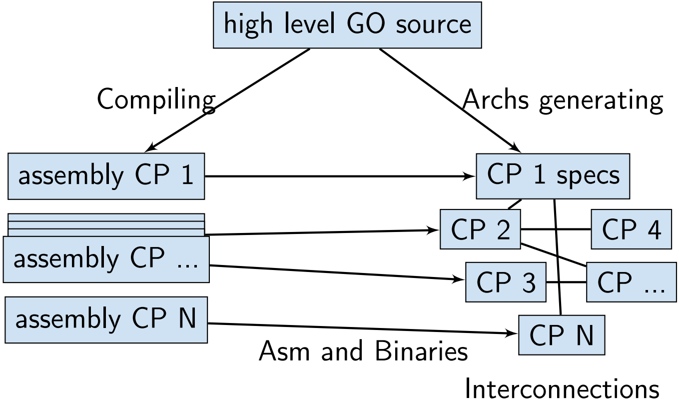
Bondgo

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



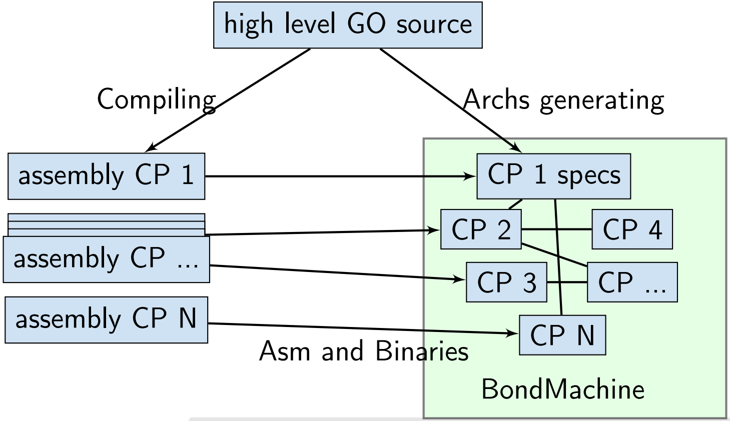
Bondgo

... 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

multi-core counter

```
package main

import (
   "bondgo"
)

func pong() {
   var in0 bondgo.Input
   var out0 bondgo.Output
   in0 = bondgo.Make(bondgo.Input, 3)
   out0 = bondgo.Make(bondgo.Output, 5)
   for {
      bondgo.IOWrite(out0, bondgo.IORead(in0)+1)
   }
}

func main() {
   var in0 bondgo.Input
   var out0 bondgo.Output
   in0 = bondgo.Make(bondgo.Input, 5)
   out0 = bondgo.Make(bondgo.Output, 3)
device_0:
   go pong()
   for {
      bondgo.IOWrite(out0, bondgo.IORead(in0))
   }
}
```

Bondgo

A multi-core example

Compiling the code with the bondgo compiler:

bondgo -input-file ds.go -mpm

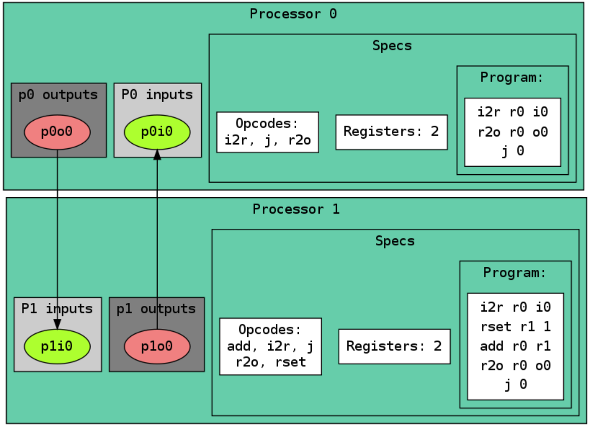
The toolchain perform the following steps:

- Map the two goroutines to two hardware cores.
- Creates two types of core, each one optimized to execute the assigned goroutine.
- Creates the two binaries.
- Connected the two core as inferred from the source code, using special IO registers.

The result is a multicore BondMachine:

Bondgo

A multi-core example



Compiling Architectures

One of the most important result

The architecture creation is a part of the compilation process.

Simulation demo

Demo N.8

It will be shown how:

- To use bondgo to create a chain of interconnected processors
- To flash the firmware to the board

Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

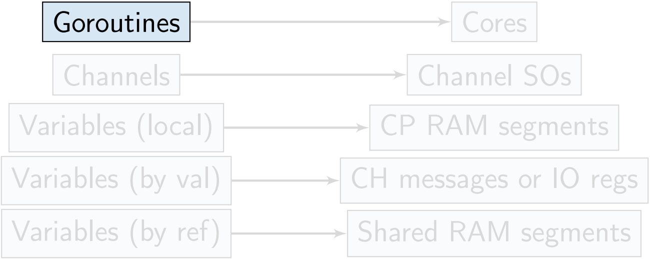
High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

Bondgo

Go in hardware

Bondgo implements a sort of “Go in hardware”.

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

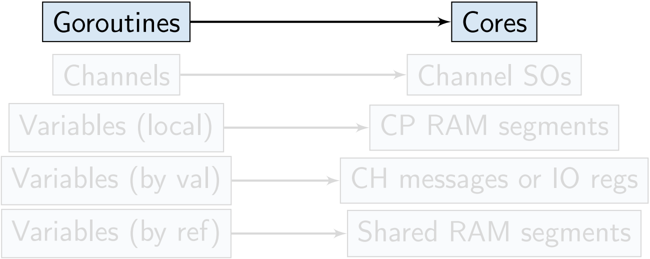


Bondgo

Go in hardware

Bondgo implements a sort of “Go in hardware”.

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

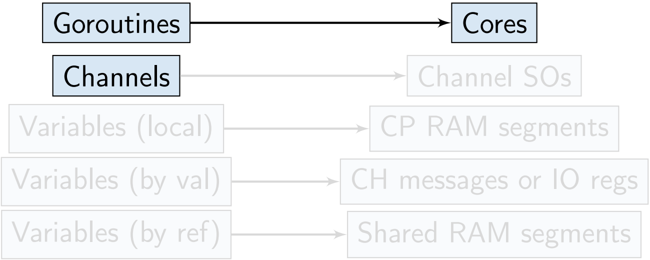


Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

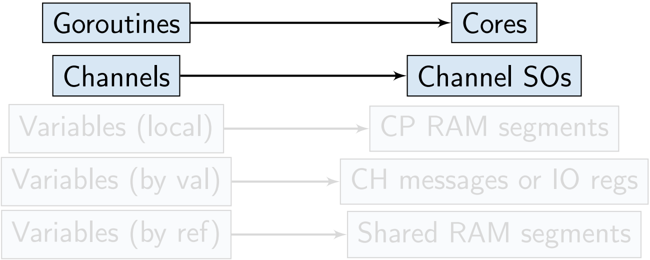


Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

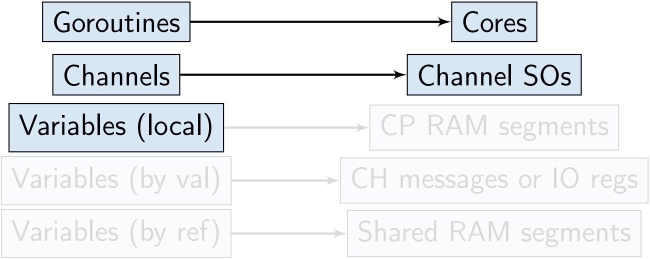


Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

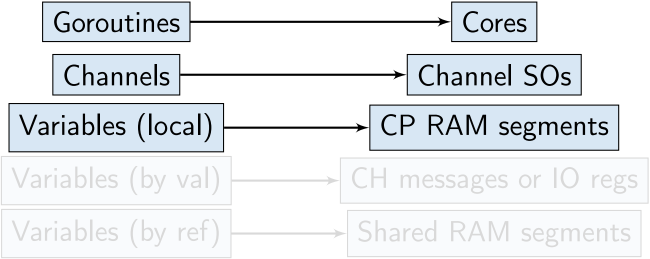


Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

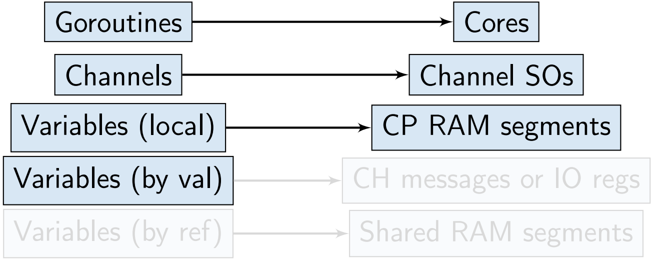


Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

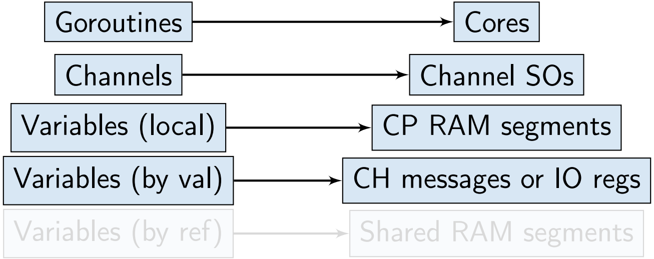


Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

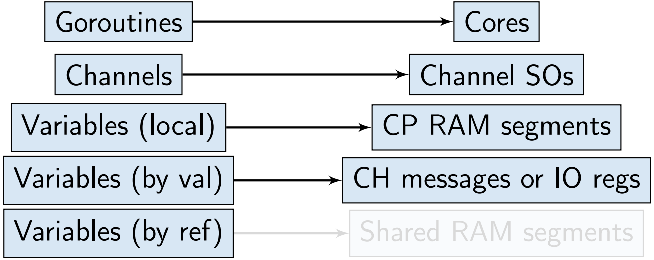


Bondgo

Go in hardware

Bondgo implements a sort of "Go in hardware".

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.

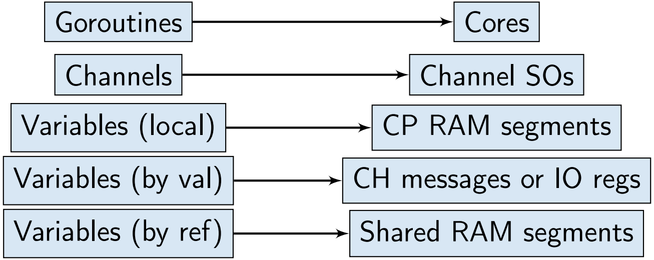


Bondgo

Go in hardware

Bondgo implements a sort of “Go in hardware”.

High level Go source code is directly mapped to interconnected processors without Operating Systems or runtimes.



Go in hardware

Second idea on the BondMachine

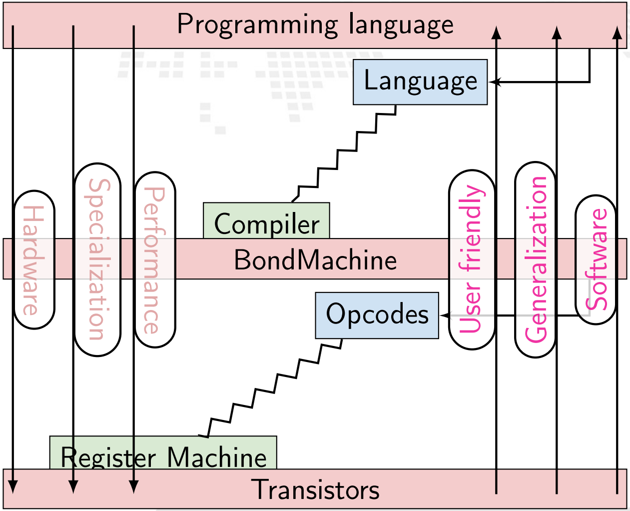
The idea was:
Build a computing system with a decreased number of layers resulting in a lower HW/SW gap.

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

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.

Basm

The BondMachine assembler *Basm* is the compiler complementary tools.

The BondMachine "fluid" nature gives the assembler some unique features:

- Support for template based assembly code
- Combining and rewriting fragments of assembly code
- Building hardware from assembly
- Software/Hardware rearrange capabilities

Basm

The BondMachine assembler *Basm* is the compiler complementary tools.

The BondMachine "fluid" nature gives the assembler some unique features:

- Support for template based assembly code
- Combining and rewriting fragments of assembly code
- Building hardware from assembly
- Software/Hardware rearrange capabilities

Basm

The BondMachine assembler *Basm* is the compiler complementary tools.

The BondMachine "fluid" nature gives the assembler some unique features:

- Support for template based assembly code
- Combining and rewriting fragments of assembly code
- Building hardware from assembly
- Software/Hardware rearrange capabilities

Basm

The BondMachine assembler *Basm* is the compiler complementary tools.

The BondMachine "fluid" nature gives the assembler some unique features:

- Support for template based assembly code
- Combining and rewriting fragments of assembly code
- Building hardware from assembly
- Software/Hardware rearrange capabilities

Basm

The BondMachine assembler *Basm* is the compiler complementary tools.

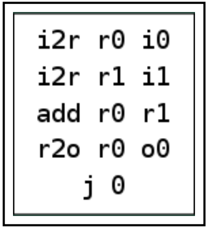
The BondMachine "fluid" nature gives the assembler some unique features:

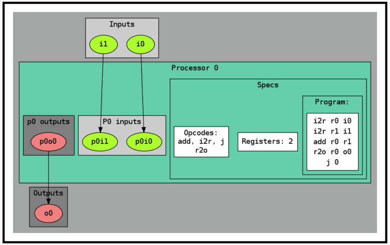
- Support for template based assembly code
- Combining and rewriting fragments of assembly code
- Building hardware from assembly
- Software/Hardware rearrange capabilities

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.





These are Building Blocks for complex BondMachines.

Builders API

With these Building Blocks

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

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to perform matrices operations.

more about these tools

Builders API

With these Building Blocks

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

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to perform matrices operations.

more about these tools

Builders API

With these Building Blocks

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

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to perform matrices operations.

more about these tools

Builders API

With these Building Blocks

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

- Symbond, to handle mathematical expression.
- Boolbond, to map boolean expression.
- Matrixwork, to perform matrices operations.

more about these tools

Builders API

Symbond

A mathematical expression, or a system can be converted to a BondMachine:

sum(var(x),const(2))

Boolbond

```
symbond -expression "sum(var(x),const(2))" -save-bondmachine bondmachine.json
```

Resulting in:

Builders API

Symbond

A mathematical expression, or a system can be converted to a BondMachine:

sum(var(x),const(2))

Boolbond

symbond -expression "sum(var(x),const(2))" -save-bondmachine bondmachine.json

Resulting in:

Builders API

Symbond

A mathematical expression, or a system can be converted to a BondMachine:

sum(var(x),const(2))

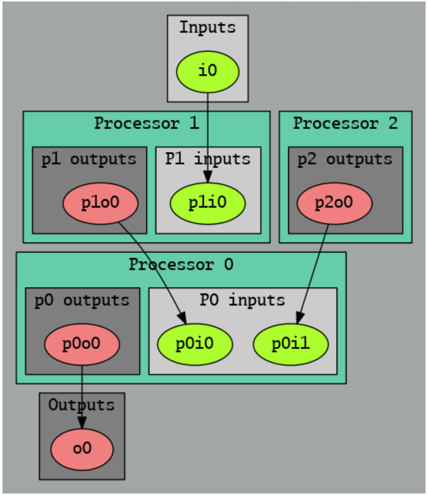
Boolbond

```
symbond -expression "sum(var(x),const(2))" -save-bondmachine bondmachine.json
```

Resulting in:

Builders API

Symbond



Builders API

Boolbond

A system of boolean equations, input and output variables are expressed as in the example file:

```
var(z)=or(var(x),not(var(y)))
var(t)=or(and(var(x),var(y)),var(z))
var(l)=and(xor(var(x),var(y)),var(t))
i:var(x)
i:var(y)
o:var(z)
o:var(t)
o:var(l)
```

Boolbond

boolbond -system-file expression.txt -save-bondmachine bondmachine.json

Resulting in:

Builders API

Boolbond

A system of boolean equations, input and output variables are expressed as in the example file:

```
var(z)=or(var(x),not(var(y)))
var(t)=or(and(var(x),var(y)),var(z))
var(l)=and(xor(var(x),var(y)),var(t))
i:var(x)
i:var(y)
o:var(z)
o:var(t)
o:var(l)
```

Boolbond

boolbond -system-file expression.txt -save-bondmachine bondmachine.json

Resulting in:

Builders API

Boolbond

A system of boolean equations, input and output variables are expressed as in the example file:

```
var(z)=or(var(x),not(var(y)))
var(t)=or(and(var(x),var(y)),var(z))
var(l)=and(xor(var(x),var(y)),var(t))
i:var(x)
i:var(y)
o:var(z)
o:var(t)
o:var(l)
```

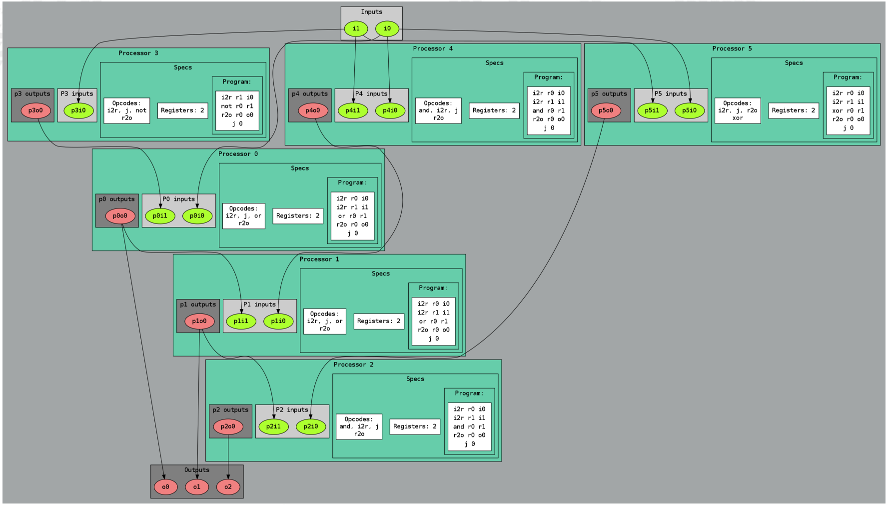
Boolbond

boolbond -system-file expression.txt -save-bondmachine bondmachine.json

Resulting in:

Builders API

Boolbond



Boolbond demo

Demo N.9

It will be shown how:

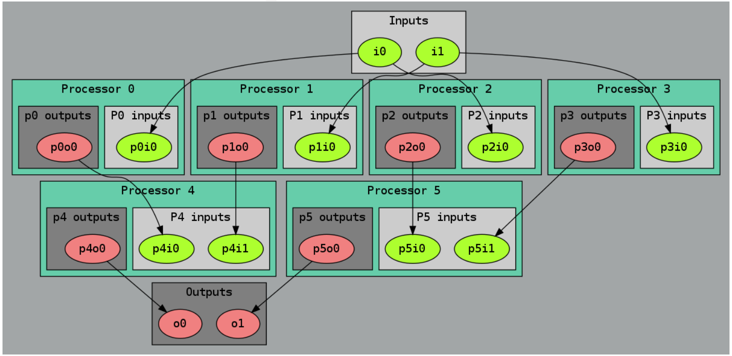
- To create complex multi-cores from boolean expressions

Builders API

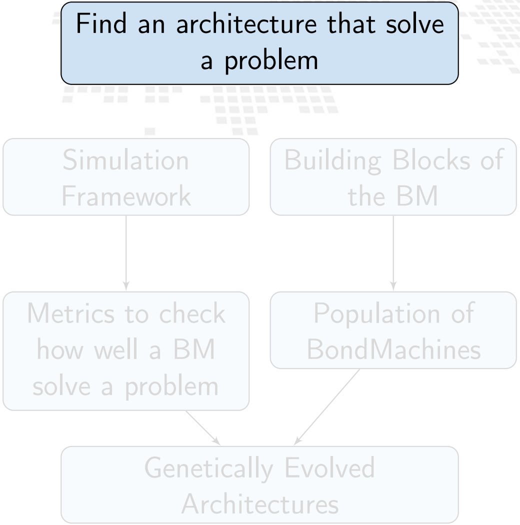
Matrixwork

Matrix multiplication

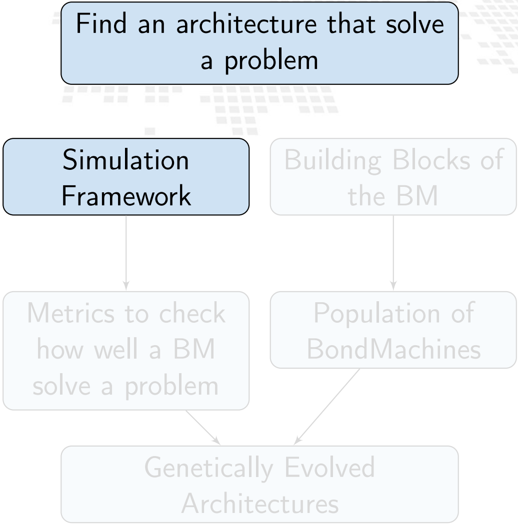
if mymachine, ok := matrixwork.Build_M(n, t); ok == nil ...



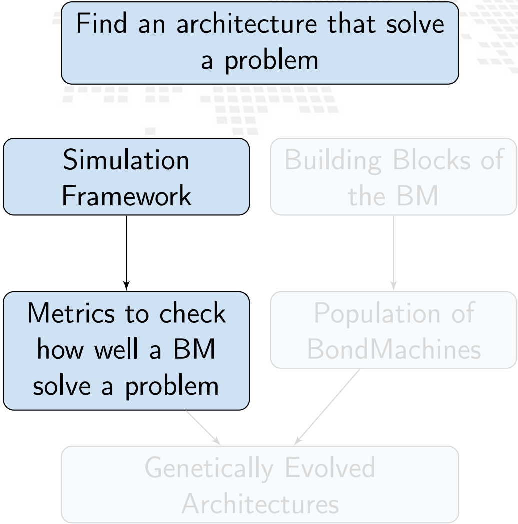
Evolutionary BondMachine



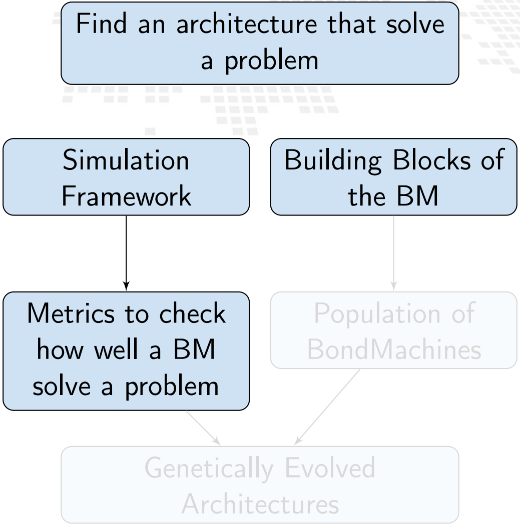
Evolutionary BondMachine



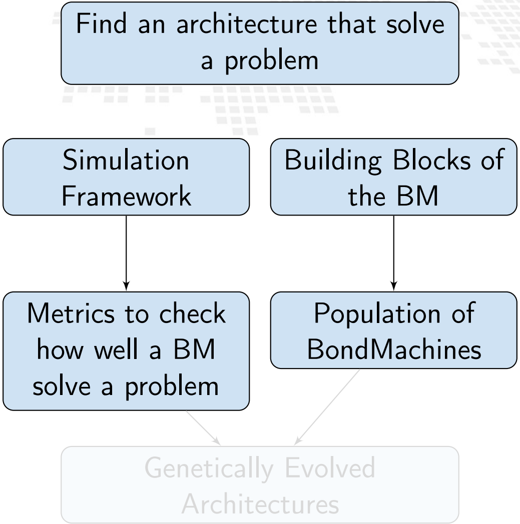
Evolutionary BondMachine



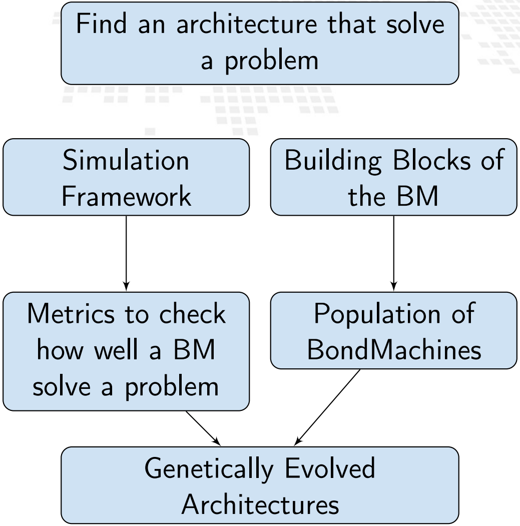
Evolutionary BondMachine



Evolutionary BondMachine



Evolutionary BondMachine



Clustering

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.

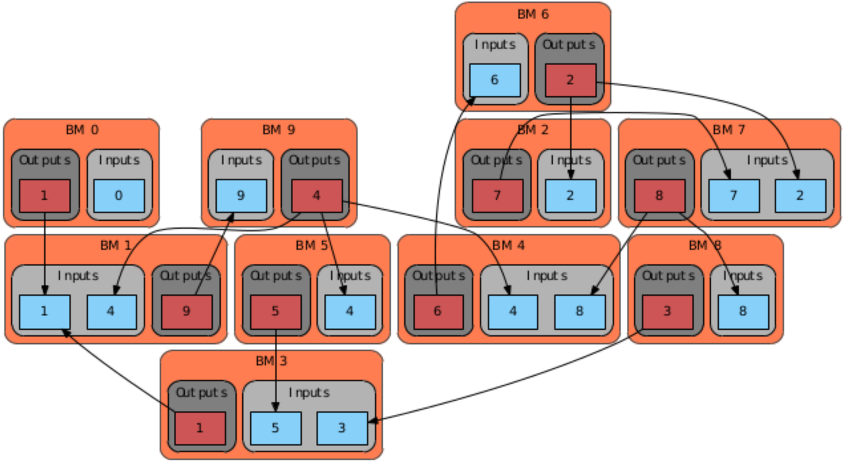
BondMachine Clustering

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



BondMachine Clustering

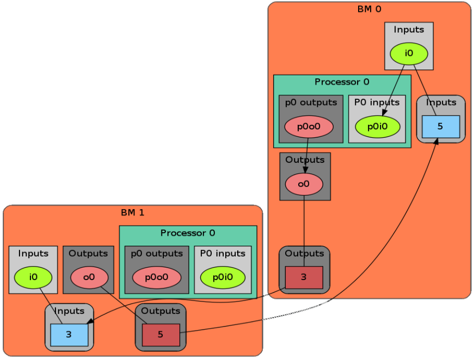
A distributed example

distributed counter

```
package main

import (
   "bondgo"
)

func pong() {
   var in0 bondgo.Input
   var out0 bondgo.Output
   in0 = bondgo.Make(bondgo.Input, 3)
   out0 = bondgo.Make(bondgo.Output, 5)
   for {
      bondgo.IOWrite(out0, bondgo.IORead(in0)+1)
   }
}

func main() {
   var in0 bondgo.Input
   var out0 bondgo.Output
   in0 = bondgo.Make(bondgo.Input, 5)
   out0 = bondgo.Make(bondgo.Output, 3)
device_1:
   go pong()
   for {
      bondgo.IOWrite(out0, bondgo.IORead(in0))
   }
}
```

BondMachine Clustering

A distributed example



BondMachine Clustering

A distributed example

See it working:

`https://youtube.com/embed/g9xYHK0zca4`

A general result

Parts of the system can be redeployed among different devices without changing the system behavior (only the performances).

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.

Accelerators

Specs

FPGA

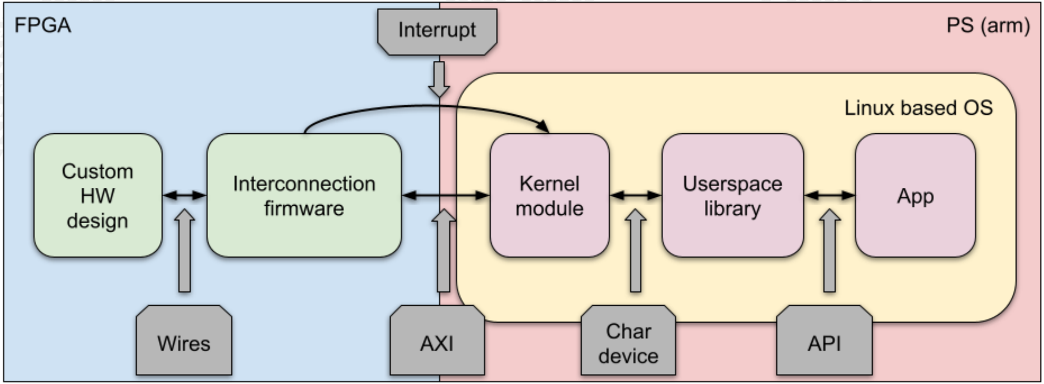
- Digilent Zedboard
- Soc: Zynq XC7Z020-CLG484-1
- 512 MB DDR3
- Vivado 2020.2

Workstations

- Dell Precision Tower 3620
- Intel(R) Xeon(R) CPU E3-1270 v5 @ 3.60GHz
- 16GB Ram
- Golang 1.18.1

- Intel(R) CPU I5-8500 v5 @ 3GHz
- 16GB Ram
- GCC with -O0

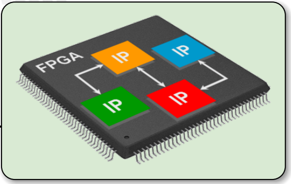
The whole system overview

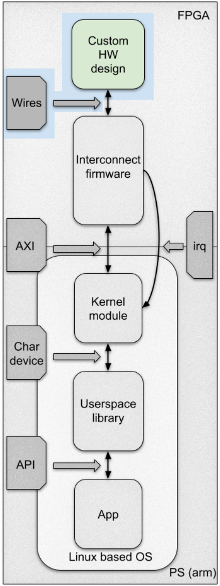


The Accelerator IP

Hardware Description Language

```
module Mat_mult(A,B,Res);
    input [31:0] A;
    input [31:0] B;
    output [31:0] Res;
    //internal variables
    reg [31:0] Res;
    reg [7:0] A1 [0:1][0:1];
    reg [7:0] B1 [0:1][0:1];
    reg [7:0] Res1 [0:1][0:1];
    integer i,j,k;

    always@ (A or B)
    begin
        {A1[0][0],A1[0][1],A1[1][0],A1[1][1]} = A;
        {B1[0][0],B1[0][1],B1[1][0],B1[1][1]} = B;
        i = 0;
        j = 0;
        k = 0;
        {Res1[0][0],Res1[0][1],Res1[1][0],Res1[1][1]} = 32'd0;
        for(i=0;i < 2;i=i+1)
            for(j=0;j < 2;j=j+1)
                for(k=0;k < 2;k=k+1)
                    Res1[i][j] = Res1[i][j] + (A1[i][k] * B1[k][j]);
        Res = {Res1[0][0],Res1[0][1],Res1[1][0],Res1[1][1]};
    end
endmodule
```



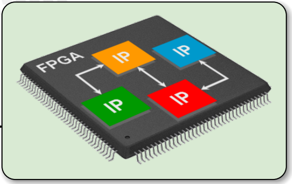


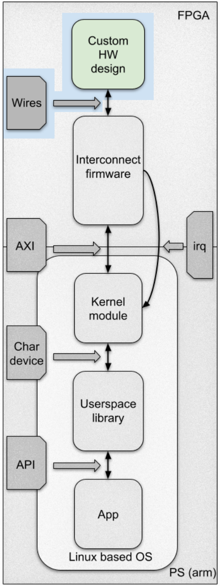
The Accelerator IP

Hardware Description Language

High Level Synthesis

```
template <typename T, int DIM>
void mmult_hw(T A[DIM][DIM], T B[DIM][DIM], T C[DIM][DIM])
{
    // matrix multiplication of a A*B matrix
    L1:for (int ia = 0; ia < DIM; ++ia)
    {
        L2:for (int ib = 0; ib < DIM; ++ib)
        {
            T sum = 0;
            L3:for (int id = 0; id < DIM; ++id)
            {
                sum += A[ia][id] * B[id][ib];
            }
            C[ia][ib] = sum;
        }
    }
}
```



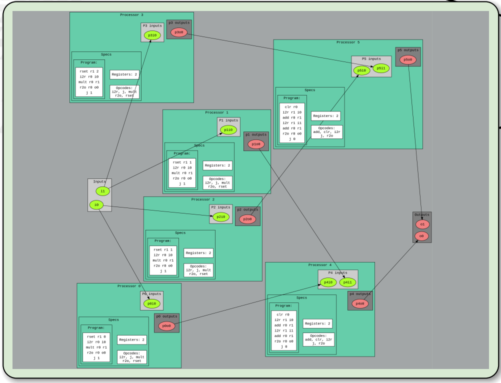


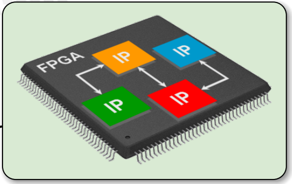
The Accelerator IP

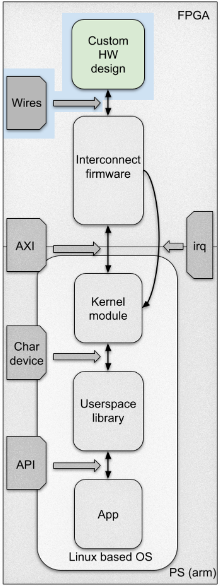
Hardware Description Language

High Level Synthesis

BondMachine





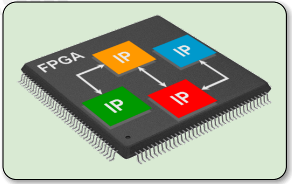


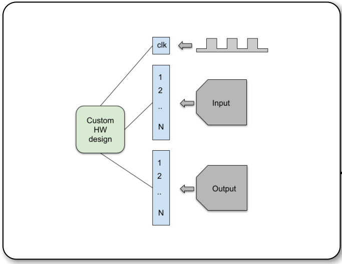
The Accelerator IP

Hardware Description Language

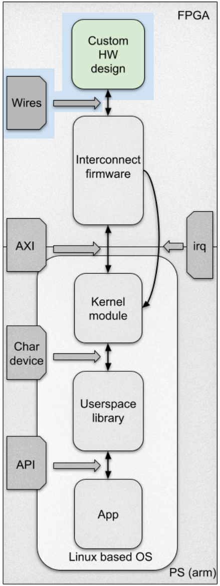
High Level Synthesis

BondMachine



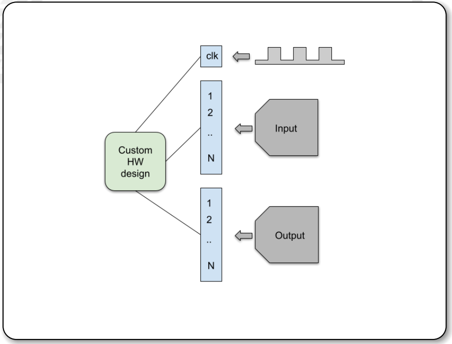


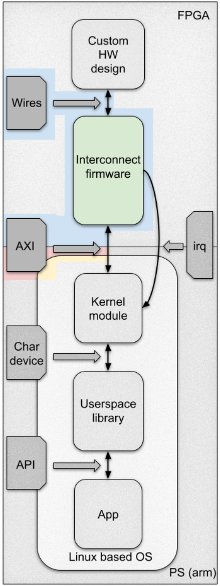
Wires:
- a clock signal,
- an input bus,
- an output bus for the result



Interconnection firmware

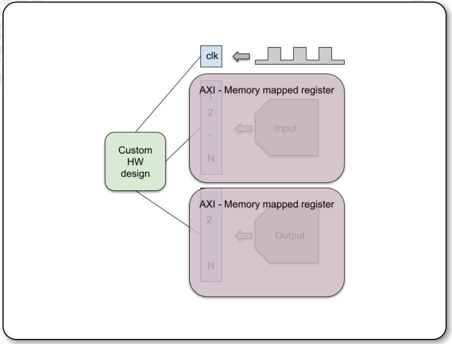
The input and output buses are the endpoints that we would like to have on the linux system.

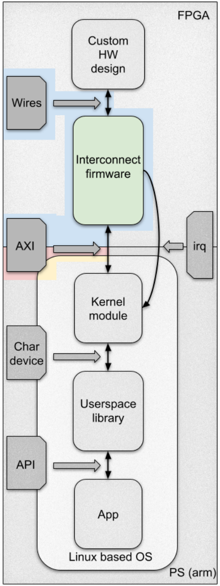




Interconnection firmware

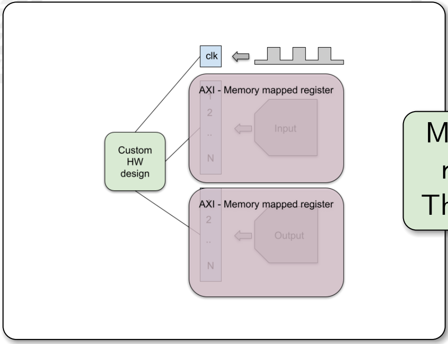
The input and output buses are the endpoints that we would like to have on the linux system.





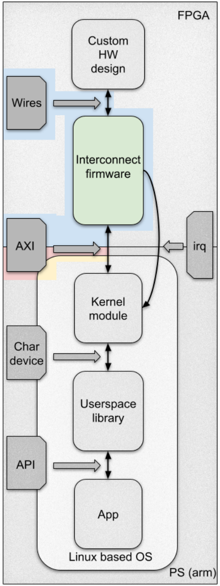
Interconnection firmware

The input and output buses are the endpoints that we would like to have on the linux system.



Memory mapped registers using The AXI protocol

```
wire [31:0] states;
wire [31:0] changes;
wire [31:0] DVDR_PS2PL;
wire [31:0] DVDR_PL2PS;
wire [31:0] port_o0;
wire [31:0] port_o1;
wire [31:0] port_o2;
wire [31:0] port_i0;
wire [31:0] port_i1;
wire [31:0] port_i2;

bondmachine_main bondmachine_inst(
    .clk(S_AXI_ACLK),
    .btnC(btnC),
    .A_DVDR_PS2PL(DVDR_PS2PL),
    .A_DVDR_PL2PS(DVDR_PL2PS),
    .A_changes(changes),
    .A_states(states),
    .A_port_o0(port_o0),
    .A_port_o1(port_o1),
    .A_port_o2(port_o2),
    .A_port_i0(port_i0),
    .A_port_i1(port_i1),
    .A_port_i2(port_i2),
    .interrupt(interrupt)
);
assign port_i0 = slv_reg0[31:0];
assign port_i1 = slv_reg1[31:0];
assign port_i2 = slv_reg2[31:0];
assign DVDR_PS2PL = slv_reg3[31:0];
assign states = slv_reg4[31:0];

always @( posedge S_AXI_ACLK )
begin
    slv_reg5 <= port_o0[31:0];
    slv_reg6 <= port_o1[31:0];
    slv_reg7 <= port_o2[31:0];
    sv_reg8 <= DVDR_PL2PS[31:0];
    slv_reg9 <= changes[31:0];
end
```

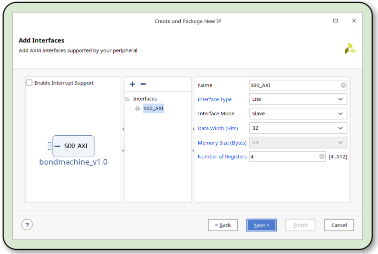


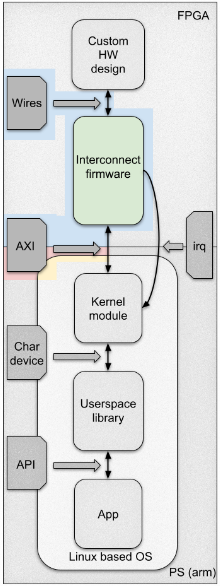
The Advanced eXtensible Interface Protocol

AXI is a communication bus protocol defined by ARM as part of the Advanced Microcontroller Bus Architecture (AMBA) standard.

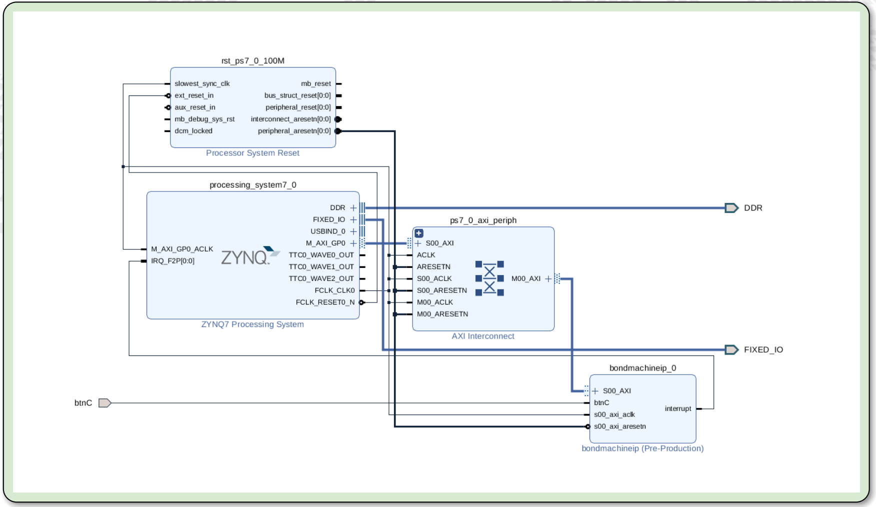
There are 3 types of AXI Interfaces:

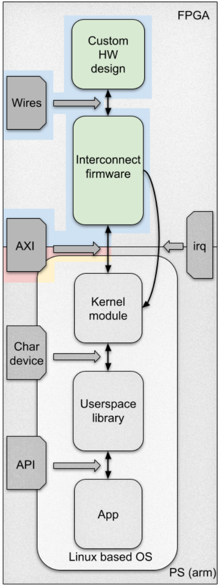
- AXI Full: for high-performance memory-mapped requirements.
- AXI Lite: for low-throughput memory-mapped communication.
- AXI Stream: for high-speed streaming data.





Block Design





Linux

Now that we have a custom accelerated hardware, we need a Linux distro to run on it.

Common Features

Complete system build from source
Allow choice of kernel and bootloader
Support for modifying packages with patches or custom configuration files
Can build cross-toolchains for development
Convenient support for read-only root filesystems
Support offline builds
The build configuration files integrate well with SCM tools

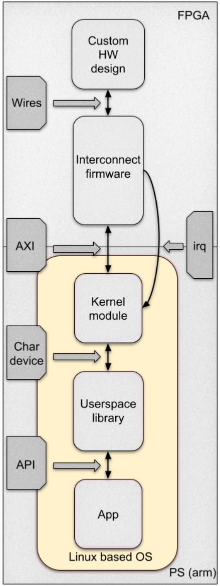
Yocto

Convenient sharing of build configuration among similar projects (meta-layers)
Larger community (Linux Foundation project)
Can build a toolchain that runs on the target
A package management system

Buildroot

Simple Makefile approach, easier to understand how the build system works
Reduced resource requirements on the build machine
Very easy to customize the final root filesystem (overlays)

Credits: https://jumpnowtek.com/linux/Choosing-an-embedded-linux-build-system.html



Ingredients to build the distro

Kernel config

Linux kernel configuration

Boot Loader config

U-boot Boot loader configuration

Device tree

Data structure describing hardware components

Used by: Kernel Boot loader

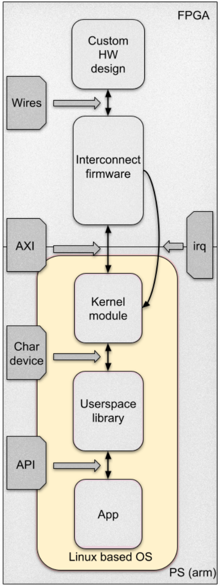
Created by: Vivado + GCC

Bitstream (firmware)

File describing Cells and routing of the FPGA

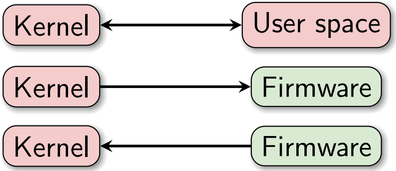
Used by Boot loader to programm FPGA

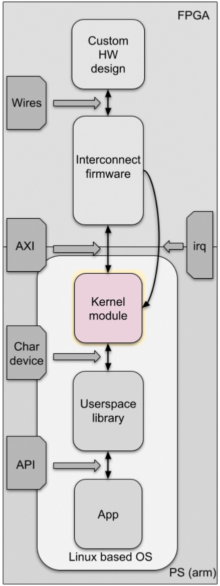
Created by: Vivado



kernel module

- The accelerator endpoints are exposed via AXI memory-mapped as memory location of the arm processor running Linux.
- To properly use the accelerator from user space, the kernel has to handle the accelerator endpoints and make them available to user space.
- We developed a kernel module for our accelerators. It manages 3 data flows:





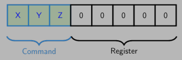
Kernel from and to user space: char device

The communication are through the standard read and write system call on a kernel generated char device

A language has been implemented for the desired operations

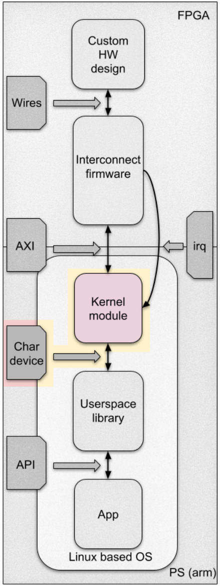
```
static ssize_t bm_read(struct file *filp, char __user *buf, size_t len, loff_t *off)
{
        struct work_data *writework;

        wait_event_interruptible(wait_queue_bm, wait_queue_flag != 0);

        switch (wait_queue_flag)
        {
        case 1:
                switch (bmacc_state)
                {
                case stateHSRECV:
                        if (copy_to_user(buf, &hsmask, 1))
                        {
                                pr_err("Data Read : Err!\n");
                        }
                        //pr_info("Sent MASK");
                        bmacc_state = stateMASKSENT;
                        wait_queue_flag = 0;
                        return 1;
                        brealk
// ....
```

```
static ssize_t bm_write(struct file *filp, const char __user *buf, size_t len, loff_t *off)
{
        struct work_data *writework;

        if (copy_from_user(write_buffer, buf, len))
        {
                pr_err("data write error\n");
        }
        else
        {
                for (i = 0; i < len; i++)
                {
                        switch (bmacc_state)
                        {
                        case stateWAIT:
                                switch (write_buffer[i] & cmdMASK)
                                {
                                case cmdHANDSH:
                                        //pr_info("HandShake");
                                        hs = write_buffer[i];
                                        bmacc_state = stateHSRECV;
                                        wait_queue_flag = 0;
                                        writework = kmalloc(sizeof(struct work_data), GFP_KERNEL);
                                        INIT_WORK(&writework->work, work_handler);
                                        writework->cs = 1;
                                        queue_work(wq, &writework->work);
                                        break;
```



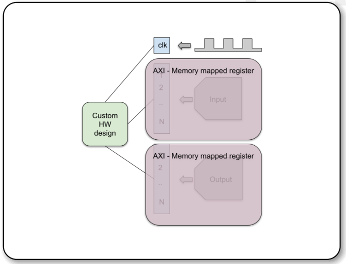
cmdNEWVAL	000	cmdDVALIDH	001
cmdDVALIDL	010	cmdDRECVH	011
cmdDRECVL	100	cmdHANDSH	101
cmdKEEP	110	cmdMASK	111

BMRRP: BondMachine Register Replica Protocol



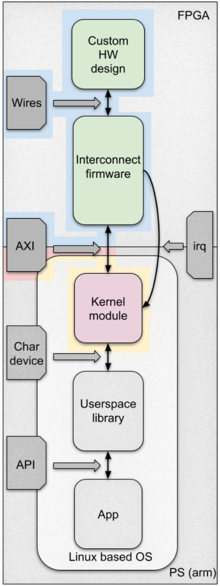
Kernel to firmware

Once the kernel has correctly decoded the data from the char device, it can directly write on AXI registers.



AXI registers are directly written by the kernel

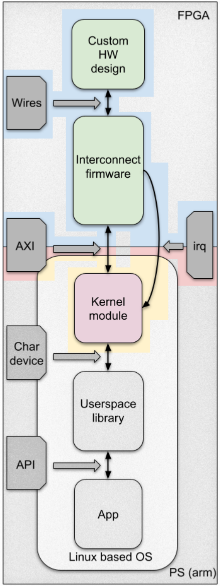
AXI guarantees consistency and transfer to the firmware input ports. Moreover the data flow from kernel cannot saturate the PL part.



Firmware to kernel: IRQ

Different story is the data flow from the FPGA to the PS part. Data can easily flow so fast to saturate and make the PS part completely unusable.

The firmware collect all the changes to send and fill in a list using a dedicated AXI register

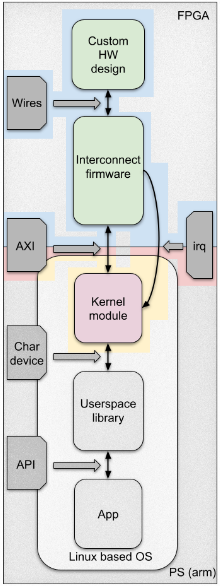


Firmware to kernel: IRQ

Different story is the data flow from the FPGA to the PS part. Data can easily flow so fast to saturate and make the PS part completely unusable.

The firmware collect all the changes to send and fill in a list using a dedicated AXI register

Stop accepting new changes from the IP



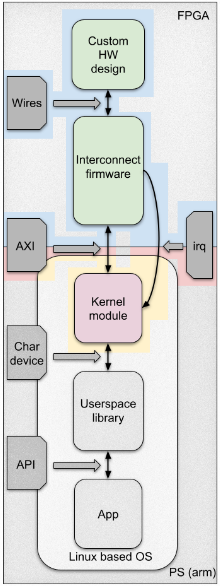
Firmware to kernel: IRQ

Different story is the data flow from the FPGA to the PS part. Data can easily flow so fast to saturate and make the PS part completely unusable.

The firmware collect all the changes to send and fill in a list using a dedicated AXI register

Send an interrupt request to the kernel

Stop accepting new changes from the IP



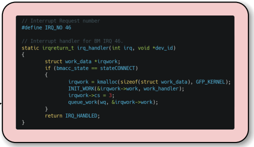
Firmware to kernel: IRQ

Different story is the data flow from the FPGA to the PS part. Data can easily flow so fast to saturate and make the PS part completely unusable.

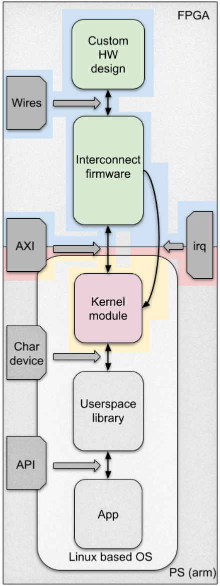
The firmware collect all the changes to send and fill in a list using a dedicated AXI register

Send an interrupt request to the kernel

Stop accepting new changes from the IP



```c
// Interrupt Request number
#define IRQ_NO 46

// Interrupt handler for BM IRQ 46.
static irqreturn_t irq_handler(int irq, void *dev_id)
{
        struct work_data *irqwork;
        if (bmacc_state == stateCONNECT)
        {
                irqwork = kmalloc(sizeof(struct work_data), GFP_KERNEL);
                INIT_WORK(&irqwork->work, work_handler);
                irqwork->cs = 3;
                queue_work(wq, &irqwork->work);
        }
        return IRQ_HANDLED;
}
```



Firmware to kernel: IRQ

Different story is the data flow from the FPGA to the PS part. Data can easily flow so fast to saturate and make the PS part completely unusable.

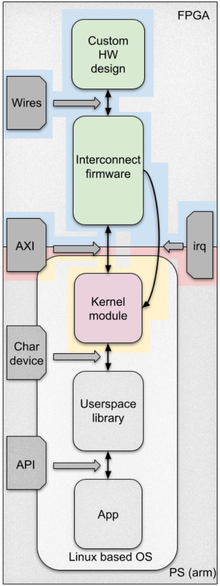
The firmware collect all the changes to send and fill in a list using a dedicated AXI register

Stop accepting new changes from the IP

Send an interrupt request to the kernel

```c
// Interrupt Request number
#define IRQ_NO 46

// Interrupt handler for BM IRQ 46.
static irqreturn_t irq_handler(int irq, void *dev_id)
{
        struct work_data *irqwork;
        if (bmacc_state == stateCONNECT)
        {
                irqwork = kmalloc(sizeof(struct work_data), GFP_KERNEL);
                INIT_WORK(&irqwork->work, work_handler);
                irqwork->cs = 3;
                queue_work(wq, &irqwork->work);
        }
        return IRQ_HANDLED;
}
```

The kernel get the IRQ, read the list of changes and send each of they through the char dev



Firmware to kernel: IRQ

Different story is the data flow from the FPGA to the PS part. Data can easily flow so fast to saturate and make the PS part completely unusable.

The firmware collect all the changes to send and fill in a list using a dedicated AXI register

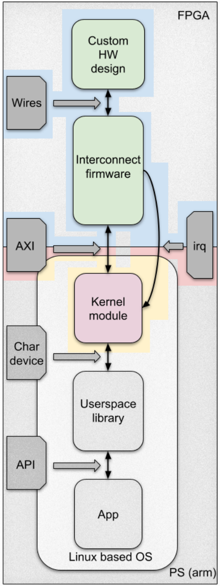
Send an interrupt request to the kernel

```c
// Interrupt Request number
#define IRQ_NO 46

// Interrupt handler for BM IRQ 46.
static irqreturn_t irq_handler(int irq, void *dev_id)
{
        struct work_data *irqwork;
        if (bmacc_state == stateCONNECT)
        {
                irqwork = kmalloc(sizeof(struct work_data), GFP_KERNEL);
                INIT_WORK(&irqwork->work, work_handler);
                irqwork->cs = 3;
                queue_work(wq, &irqwork->work);
        }
        return IRQ_HANDLED;
}
```

The kernel get the IRQ, read the list of changes and send each of they through the char dev

The kernel notify the firmware when done

Stop accepting new changes from the IP



Library

The char device created by the kernel is opened by the BMAPI user space library that implements the BMMRP.

/dev/bm

BMAPI Library

(*BMAPI) BMr2owa

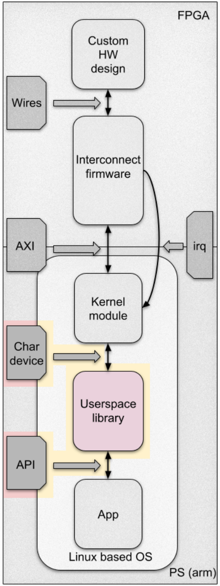
(*BMAPI) BMr2ow

(*BMAPI) BMr2o

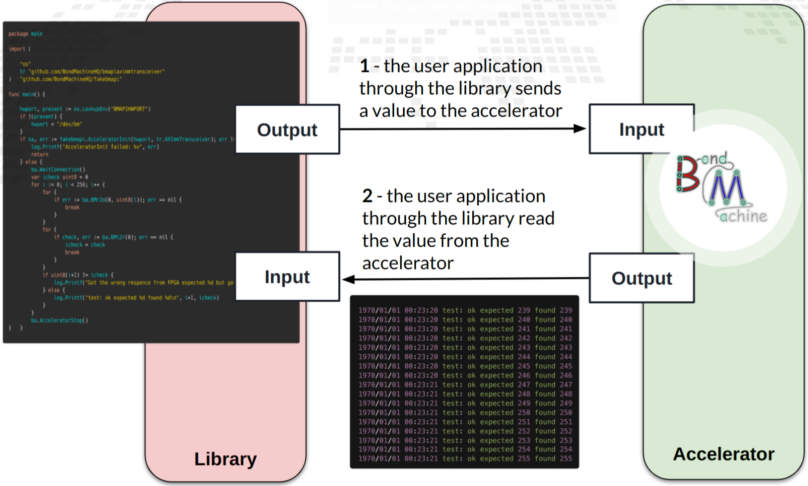
(*BMAPI) BMi2rw

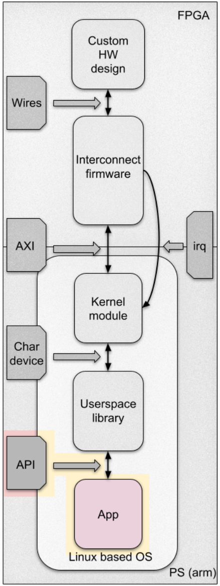
(*BMAPI) BMi2r

The library functions can be used by the application

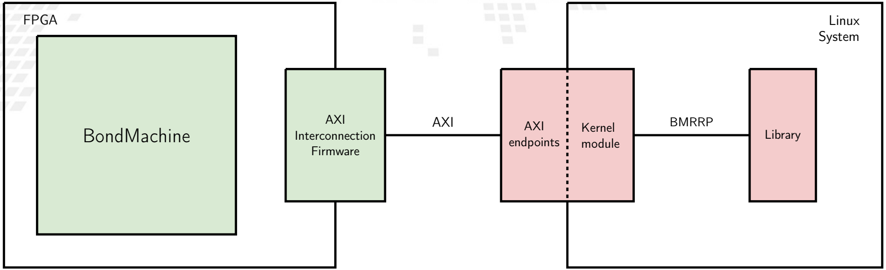


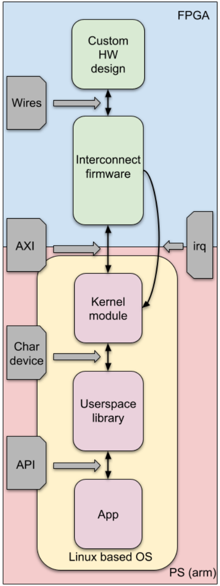
Accelerated application: an example





Accelerated Application





An example

- Definition of an example
- Check of the correctness of the accelerator results
- Benchmark of the execution

Squared Matrix-vector multiplication

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \left[c_i\right]_{i=1}^{n} = \left[\sum_{k=1}^{n} a_{ik} b_k\right]_{i=1}^{n}$$

Squared Matrix-vector multiplication

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \left[c_i\right]_{i=1}^{n} = \left[\sum_{k=1}^{n} a_{ik} b_k\right]_{i=1}^{n}$$

```
"A": [
      [6,5],
      [1,2]
],
"B": [
      [3,1,1],
      [6,7,2],
      [7,1,4]
],
"C": [
      [6,3,7,1],
      [1,6,4,2],
      [3,2,1,7],
      [5,3,1,7]
],
```

Squared Matrix-vector multiplication

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = [c_i]_{i=1}^{n} = \left[\sum_{k=1}^{n} a_{ik} b_k\right]_{i=1}^{n}$$

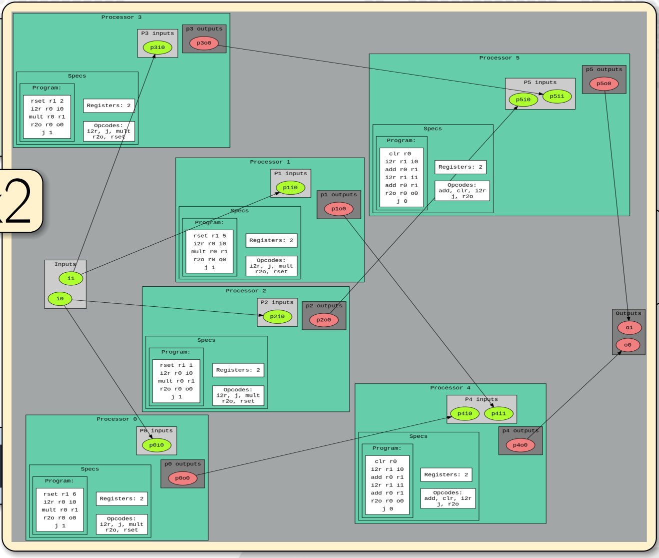
```
"A": [
      [6,5],
      [1,2]
],
"B": [
      [3,1,1],
      [6,7,2],
      [7,1,4]
],
"C": [
      [6,3,7,1],
      [1,6,4,2],
      [3,2,1,7],
      [5,3,1,7]
],
```

```
matrixwork -constants constants.json -constant-matrix A -numerical-type uint8 ...
```

Squared Matrix-vector multiplication

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{n1} & a_{n2} \end{bmatrix}$$

2x2




```
matrixwork
```

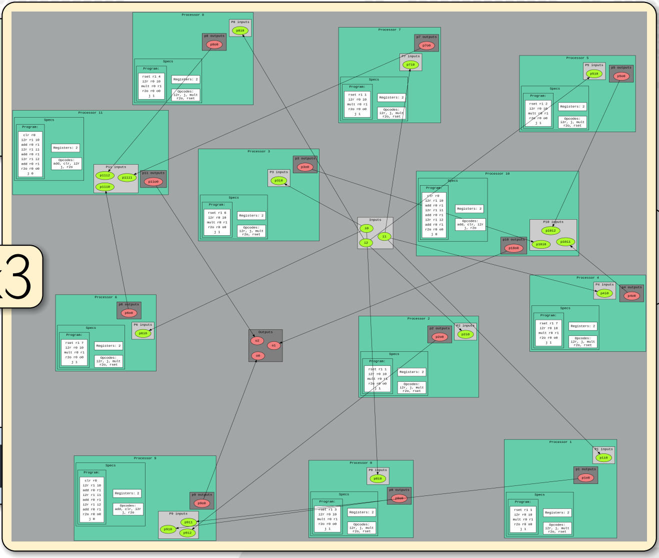
```
"A": [
      [6,5],
      [1,2]
],
"B": [
      [3,1,1],
      [6,7,2],
      [7,1,4]
],
"C": [
      [6,3,7,1],
      [1,6,4,2],
      [3,2,1,7],
      [5,3,1,7]
],
```

Squared Matrix-vector multiplication

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{n1} & a_{n2} \end{bmatrix}$$

3x3

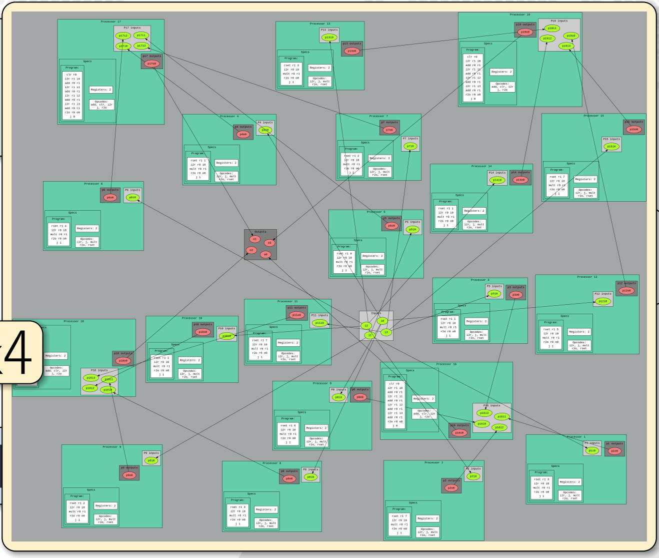
`matrixwork`




```
"A": [
      [6,5],
      [1,2]
],
"B": [
      [3,1,1],
      [6,7,2],
      [7,1,4]
],
"C": [
      [6,3,7,1],
      [1,6,4,2],
      [3,2,1,7],
      [5,3,1,7]
],
```

Squared Matrix-vector multiplication

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{n1} & a_{n2} \end{bmatrix}$$



4x4

matrixwork

```
"A": [
      [6,5],
      [1,2]
],
"B": [
      [3,1,1],
      [6,7,2],
      [7,1,4]
],
"C": [
      [6,3,7,1],
      [1,6,4,2],
      [3,2,1,7],
      [5,3,1,7]
],
```

Correctness and module debug

To verify the correct computation of the accelerator:

- a tool to monitor the AXI memory

```
# ./monitor -g 0x43c00000 -n 8
i0:     00000000 (0x43c00003) 00000000 (0x43c00002) 00000000 (0x43c00001) 11111010 (0x43c00000)
i1:     00000000 (0x43c00007) 00000000 (0x43c00006) 00000000 (0x43c00005) 00000000 (0x43c00004)
i2:     00000000 (0x43c0000b) 00000000 (0x43c0000a) 00000000 (0x43c00009) 00000000 (0x43c00008)
i3:     00000000 (0x43c0000f) 00000000 (0x43c0000e) 00000000 (0x43c0000d) 00000000 (0x43c0000c)
i4:     00000000 (0x43c00013) 00000000 (0x43c00012) 00000000 (0x43c00011) 00000000 (0x43c00010)
i5:     00000000 (0x43c00017) 00000000 (0x43c00016) 00000000 (0x43c00015) 00000000 (0x43c00014)
i6:     00000000 (0x43c0001b) 00000000 (0x43c0001a) 00000000 (0x43c00019) 00000000 (0x43c00018)
i7:     00000000 (0x43c0001f) 00000000 (0x43c0001e) 00000000 (0x43c0001d) 00000000 (0x43c0001c)
PS2PL:  00000000 (0x43c00023) 00000000 (0x43c00022) 00000000 (0x43c00021) 00000000 (0x43c00020)
STATES: 00000000 (0x43c00027) 00000000 (0x43c00026) 00000000 (0x43c00025) 00000000 (0x43c00024)
o0:     00000000 (0x43c0002b) 00000000 (0x43c0002a) 00000000 (0x43c00029) 11011100 (0x43c00028)
o1:     00000000 (0x43c0002f) 00000000 (0x43c0002e) 00000000 (0x43c0002d) 11101110 (0x43c0002c)
o2:     00000000 (0x43c00033) 00000000 (0x43c00032) 00000000 (0x43c00031) 11011100 (0x43c00030)
o3:     00000000 (0x43c00037) 00000000 (0x43c00036) 00000000 (0x43c00035) 11101000 (0x43c00034)
o4:     00000000 (0x43c0003b) 00000000 (0x43c0003a) 00000000 (0x43c00039) 11011100 (0x43c00038)
o5:     00000000 (0x43c0003f) 00000000 (0x43c0003e) 00000000 (0x43c0003d) 11100010 (0x43c0003c)
o6:     00000000 (0x43c00043) 00000000 (0x43c00042) 00000000 (0x43c00041) 11110100 (0x43c00040)
o7:     00000000 (0x43c00047) 00000000 (0x43c00046) 00000000 (0x43c00045) 11011100 (0x43c00044)
bench:  00000000 (0x43c0004b) 00000000 (0x43c0004a) 00000000 (0x43c00049) 00011101 (0x43c00048)
PL2PS:  00000000 (0x43c0004f) 11111111 (0x43c0004e) 10000000 (0x43c0004d) 00000000 (0x43c0004c)
CHANGE: 00000000 (0x43c00053) 11111111 (0x43c00052) 11111111 (0x43c00051) 11000000 (0x43c00050)
```

- write directly to AXI memory mapped input addresses (through devmem)
- check the AXI memory mapped output addresses

Correctness and module debug

To verify the correct computation of the accelerator:

- a tool to monitor the AXI memory
- write directly to AXI memory mapped input addresses (through devmem)
- check the AXI memory mapped output addresses

```
# ./monitor -g 0x43c00000 -n 8
i0:     00000000 (0x43c00003) 00000000 (0x43c00002) 00000000 (0x43c00001) 11111010 (0x43c00000)
i1:     00000000 (0x43c00007) 00000000 (0x43c00006) 00000000 (0x43c00005) 00000000 (0x43c00004)
i2:     00000000 (0x43c0000b) 00000000 (0x43c0000a) 00000000 (0x43c00009) 00000000 (0x43c00008)
i3:     00000000 (0x43c0000f) 00000000 (0x43c0000e) 00000000 (0x43c0000d) 00000000 (0x43c0000c)
i4:     00000000 (0x43c00013) 00000000 (0x43c00012) 00000000 (0x43c00011) 00000000 (0x43c00010)
i5:     00000000 (0x43c00017) 00000000 (0x43c00016) 00000000 (0x43c00015) 00000000 (0x43c00014)
i6:     00000000 (0x43c0001b) 00000000 (0x43c0001a) 00000000 (0x43c00019) 00000000 (0x43c00018)
i7:     00000000 (0x43c0001f) 00000000 (0x43c0001e) 00000000 (0x43c0001d) 00000000 (0x43c0001c)
PS2PL:  00000000 (0x43c00023) 00000000 (0x43c00022) 00000000 (0x43c00021) 00000000 (0x43c00020)
STATES: 00000000 (0x43c00027) 00000000 (0x43c00026) 00000000 (0x43c00025) 00000000 (0x43c00024)
o0:     00000000 (0x43c0002b) 00000000 (0x43c0002a) 00000000 (0x43c00029) 11011100 (0x43c00028)
o1:     00000000 (0x43c0002f) 00000000 (0x43c0002e) 00000000 (0x43c0002d) 11101110 (0x43c0002c)
o2:     00000000 (0x43c00033) 00000000 (0x43c00032) 00000000 (0x43c00031) 11011100 (0x43c00030)
o3:     00000000 (0x43c00037) 00000000 (0x43c00036) 00000000 (0x43c00035) 11101000 (0x43c00034)
o4:     00000000 (0x43c0003b) 00000000 (0x43c0003a) 00000000 (0x43c00039) 11011100 (0x43c00038)
o5:     00000000 (0x43c0003f) 00000000 (0x43c0003e) 00000000 (0x43c0003d) 11100010 (0x43c0003c)
o6:     00000000 (0x43c00043) 00000000 (0x43c00042) 00000000 (0x43c00041) 11110100 (0x43c00040)
o7:     00000000 (0x43c00047) 00000000 (0x43c00046) 00000000 (0x43c00045) 11011100 (0x43c00044)
bench:  00000000 (0x43c0004b) 00000000 (0x43c0004a) 00000000 (0x43c00049) 00011101 (0x43c00048)
PL2PS:  00000000 (0x43c0004f) 11111111 (0x43c0004e) 10000000 (0x43c0004d) 00000000 (0x43c0004c)
CHANGE: 00000000 (0x43c00053) 11111111 (0x43c00052) 11111111 (0x43c00051) 11000000 (0x43c00050)
```

```
devmem 0x43c00000 b 1
```

Correctness and module debug

To verify the correct computation of the accelerator:

- a tool to monitor the AXI memory

```
# ./monitor -g 0x43c00000 -n 8
i0:     00000000 (0x43c00003) 00000000 (0x43c00002) 00000000 (0x43c00001) 11111010 (0x43c00000)
i1:     00000000 (0x43c00007) 00000000 (0x43c00006) 00000000 (0x43c00005) 00000000 (0x43c00004)
i2:     00000000 (0x43c0000b) 00000000 (0x43c0000a) 00000000 (0x43c00009) 00000000 (0x43c00008)
i3:     00000000 (0x43c0000f) 00000000 (0x43c0000e) 00000000 (0x43c0000d) 00000000 (0x43c0000c)
i4:     00000000 (0x43c00013) 00000000 (0x43c00012) 00000000 (0x43c00011) 00000000 (0x43c00010)
i5:     00000000 (0x43c00017) 00000000 (0x43c00016) 00000000 (0x43c00015) 00000000 (0x43c00014)
i6:     00000000 (0x43c0001b) 00000000 (0x43c0001a) 00000000 (0x43c00019) 00000000 (0x43c00018)
i7:     00000000 (0x43c0001f) 00000000 (0x43c0001e) 00000000 (0x43c0001d) 00000000 (0x43c0001c)
PS2PL:  00000000 (0x43c00023) 00000000 (0x43c00022) 00000000 (0x43c00021) 00000000 (0x43c00020)
STATES: 00000000 (0x43c00027) 00000000 (0x43c00026) 00000000 (0x43c00025) 00000000 (0x43c00024)
o0:     00000000 (0x43c0002b) 00000000 (0x43c0002a) 00000000 (0x43c00029) 11011100 (0x43c00028)
o1:     00000000 (0x43c0002f) 00000000 (0x43c0002e) 00000000 (0x43c0002d) 11101110 (0x43c0002c)
o2:     00000000 (0x43c00033) 00000000 (0x43c00032) 00000000 (0x43c00031) 11011100 (0x43c00030)
o3:     00000000 (0x43c00037) 00000000 (0x43c00036) 00000000 (0x43c00035) 11101000 (0x43c00034)
o4:     00000000 (0x43c0003b) 00000000 (0x43c0003a) 00000000 (0x43c00039) 11011100 (0x43c00038)
o5:     00000000 (0x43c0003f) 00000000 (0x43c0003e) 00000000 (0x43c0003d) 11100010 (0x43c0003c)
o6:     00000000 (0x43c00043) 00000000 (0x43c00042) 00000000 (0x43c00041) 11110100 (0x43c00040)
o7:     00000000 (0x43c00047) 00000000 (0x43c00046) 00000000 (0x43c00045) 11011100 (0x43c00044)
bench:  00000000 (0x43c0004b) 00000000 (0x43c0004a) 00000000 (0x43c00049) 00011101 (0x43c00048)
PL2PS:  00000000 (0x43c0004f) 11111111 (0x43c0004e) 10000000 (0x43c0004d) 00000000 (0x43c0004c)
CHANGE: 00000000 (0x43c00053) 11111111 (0x43c00052) 11111111 (0x43c00051) 11000000 (0x43c00050)
```

- write directly to AXI memory mapped input addresses (through devmem)

```
devmem 0x43c00000 b 1
```

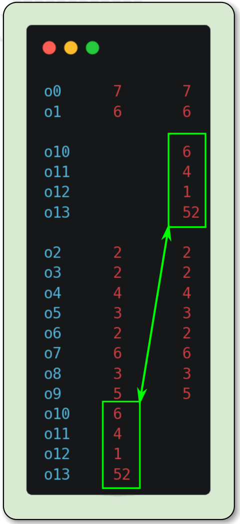
- check the AXI memory mapped output addresses

An example of error

```
#  ./monitor -g 0x43c00000 -n 13
i0:     00000000 (0x43c00003) 00000000 (0x43c00002) 00000000 (0x43c00001) 00000001 (0x43c00000)
i1:     00000000 (0x43c00007) 00000000 (0x43c00006) 00000000 (0x43c00005) 00000000 (0x43c00004)
i2:     00000000 (0x43c0000b) 00000000 (0x43c0000a) 00000000 (0x43c00009) 00000000 (0x43c00008)
i3:     00000000 (0x43c0000f) 00000000 (0x43c0000e) 00000000 (0x43c0000d) 00000000 (0x43c0000c)
i4:     00000000 (0x43c00013) 00000000 (0x43c00012) 00000000 (0x43c00011) 00000000 (0x43c00010)
i5:     00000000 (0x43c00017) 00000000 (0x43c00016) 00000000 (0x43c00015) 00000000 (0x43c00014)
i6:     00000000 (0x43c0001b) 00000000 (0x43c0001a) 00000000 (0x43c00019) 00000000 (0x43c00018)
i7:     00000000 (0x43c0001f) 00000000 (0x43c0001e) 00000000 (0x43c0001d) 00000000 (0x43c0001c)
i8:     00000000 (0x43c00023) 00000000 (0x43c00022) 00000000 (0x43c00021) 00000000 (0x43c00020)
i9:     00000000 (0x43c00027) 00000000 (0x43c00026) 00000000 (0x43c00025) 00000000 (0x43c00024)
i10:    00000000 (0x43c0002b) 00000000 (0x43c0002a) 00000000 (0x43c00029) 00000000 (0x43c00028)
i11:    00000000 (0x43c0002f) 00000000 (0x43c0002e) 00000000 (0x43c0002d) 00000000 (0x43c0002c)
i12:    00000000 (0x43c00033) 00000000 (0x43c00032) 00000000 (0x43c00031) 00000000 (0x43c00030)
PS2PL:  00000000 (0x43c00037) 00000000 (0x43c00036) 00000000 (0x43c00035) 00000000 (0x43c00034)
STATES: 00000000 (0x43c0003b) 00000000 (0x43c0003a) 00000000 (0x43c00039) 00000000 (0x43c00038)
o0:     00000000 (0x43c0003f) 00000000 (0x43c0003e) 00000000 (0x43c0003d) 00000111 (0x43c0003c)
o1:     00000000 (0x43c00043) 00000000 (0x43c00042) 00000000 (0x43c00041) 00000110 (0x43c00040)
o2:     00000000 (0x43c00047) 00000000 (0x43c00046) 00000000 (0x43c00045) 00000110 (0x43c00044)
o3:     00000000 (0x43c0004b) 00000000 (0x43c0004a) 00000000 (0x43c00049) 00000100 (0x43c00048)
o4:     00000000 (0x43c0004f) 00000000 (0x43c0004e) 00000000 (0x43c0004d) 00000001 (0x43c0004c)
o5:     00000000 (0x43c00053) 00000000 (0x43c00052) 00000000 (0x43c00051) 00110100 (0x43c00050)
o6:     00000000 (0x43c00057) 00000000 (0x43c00056) 00000000 (0x43c00055) 00000010 (0x43c00054)
o7:     00000000 (0x43c0005b) 00000000 (0x43c0005a) 00000000 (0x43c00059) 00000010 (0x43c00058)
o8:     00000000 (0x43c0005f) 00000000 (0x43c0005e) 00000000 (0x43c0005d) 00000100 (0x43c0005c)
o9:     00000000 (0x43c00063) 00000000 (0x43c00062) 00000000 (0x43c00061) 00000011 (0x43c00060)
o10:    00000000 (0x43c00067) 00000000 (0x43c00066) 00000000 (0x43c00065) 00000010 (0x43c00064)
o11:    00000000 (0x43c0006b) 00000000 (0x43c0006a) 00000000 (0x43c00069) 00000110 (0x43c00068)
o12:    00000000 (0x43c0006f) 00000000 (0x43c0006e) 00000000 (0x43c0006d) 00000011 (0x43c0006c)
o13 bcm 00000000 (0x43c00073) 00000000 (0x43c00072) 00000000 (0x43c00071) 00000101 (0x43c00070)
PL2PS:  00000000 (0x43c00077) 00000111 (0x43c00076) 11111111 (0x43c00075) 11100000 (0x43c00074)
CHANGE: 00000000 (0x43c0007b) 00000111 (0x43c0007a) 11111111 (0x43c00079) 11111111 (0x43c00078)
```

An example of error

```
#  ./monitor -g 0x43c00000 -n 13
i0:     00000000 (0x43c00003) 00000000 (0x43c00002) 00000000 (0x43c00001) 00000001 (0x43c00000)
i1:     00000000 (0x43c00007) 00000000 (0x43c00006) 00000000 (0x43c00005) 00000000 (0x43c00004)
i2:     00000000 (0x43c0000b) 00000000 (0x43c0000a) 00000000 (0x43c00009) 00000000 (0x43c00008)
i3:     00000000 (0x43c0000f) 00000000 (0x43c0000e) 00000000 (0x43c0000d) 00000000 (0x43c0000c)
i4:     00000000 (0x43c00013) 00000000 (0x43c00012) 00000000 (0x43c00011) 00000000 (0x43c00010)
i5:     00000000 (0x43c00017) 00000000 (0x43c00016) 00000000 (0x43c00015) 00000000 (0x43c00014)
i6:     00000000 (0x43c0001b) 00000000 (0x43c0001a) 00000000 (0x43c00019) 00000000 (0x43c00018)
i7:     00000000 (0x43c0001f) 00000000 (0x43c0001e) 00000000 (0x43c0001d) 00000000 (0x43c0001c)
i8:     00000000 (0x43c00023) 00000000 (0x43c00022) 00000000 (0x43c00021) 00000000 (0x43c00020)
i9:     00000000 (0x43c00027) 00000000 (0x43c00026) 00000000 (0x43c00025) 00000000 (0x43c00024)
i10:    00000000 (0x43c0002b) 00000000 (0x43c0002a) 00000000 (0x43c00029) 00000000 (0x43c00028)
i11:    00000000 (0x43c0002f) 00000000 (0x43c0002e) 00000000 (0x43c0002d) 00000000 (0x43c0002c)
i12:    00000000 (0x43c00033) 00000000 (0x43c00032) 00000000 (0x43c00031) 00000000 (0x43c00030)
PS2PL:  00000000 (0x43c00037) 00000000 (0x43c00036) 00000000 (0x43c00035) 00000000 (0x43c00034)
STATES: 00000000 (0x43c0003b) 00000000 (0x43c0003a) 00000000 (0x43c00039) 00000000 (0x43c00038)
o0:     00000000 (0x43c0003f) 00000000 (0x43c0003e) 00000000 (0x43c0003d) 00000111 (0x43c0003c)
o1:     00000000 (0x43c00043) 00000000 (0x43c00042) 00000000 (0x43c00041) 00000110 (0x43c00040)
o2:     00000000 (0x43c00047) 00000000 (0x43c00046) 00000000 (0x43c00045) 00000110 (0x43c00044)
o3:     00000000 (0x43c0004b) 00000000 (0x43c0004a) 00000000 (0x43c00049) 00000100 (0x43c00048)
o4:     00000000 (0x43c0004f) 00000000 (0x43c0004e) 00000000 (0x43c0004d) 00000001 (0x43c0004c)
o5:     00000000 (0x43c00053) 00000000 (0x43c00052) 00000000 (0x43c00051) 00110100 (0x43c00050)
o6:     00000000 (0x43c00057) 00000000 (0x43c00056) 00000000 (0x43c00055) 00000010 (0x43c00054)
o7:     00000000 (0x43c0005b) 00000000 (0x43c0005a) 00000000 (0x43c00059) 00000010 (0x43c00058)
o8:     00000000 (0x43c0005f) 00000000 (0x43c0005e) 00000000 (0x43c0005d) 00000100 (0x43c0005c)
o9:     00000000 (0x43c00063) 00000000 (0x43c00062) 00000000 (0x43c00061) 00000011 (0x43c00060)
o10:    00000000 (0x43c00067) 00000000 (0x43c00066) 00000000 (0x43c00065) 00000010 (0x43c00064)
o11:    00000000 (0x43c0006b) 00000000 (0x43c0006a) 00000000 (0x43c00069) 00000110 (0x43c00068)
o12:    00000000 (0x43c0006f) 00000000 (0x43c0006e) 00000000 (0x43c0006d) 00000011 (0x43c0006c)
o13 bcm 00000000 (0x43c00073) 00000000 (0x43c00072) 00000000 (0x43c00071) 00000101 (0x43c00070)
PL2PS:  00000000 (0x43c00077) 00000111 (0x43c00076) 11111111 (0x43c00075) 11100000 (0x43c00074)
CHANGE: 00000000 (0x43c0007b) 00000111 (0x43c0007a) 11111111 (0x43c00079) 11111111 (0x43c00078)
```



Benchmark: caveats

This is a preliminary work.

We trust some tools:

- Vivado reports
- perf

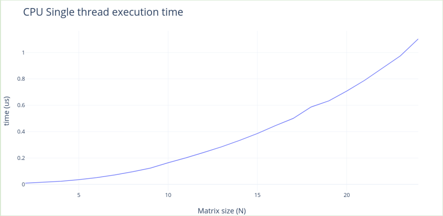
The FPGA benchmarks do not include the PS part overhead (the comparisons are not really fair)

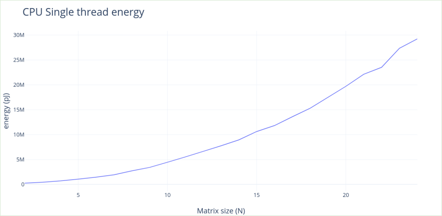
Benchmark: the CPU (Golang)

```go
func matrixtest(n int, iter int64) float32 {
//...
    start := time.Now()

    for k := 0; int64(k) < iter; k++ {
        for i := 0; i < n; i++ {
            output[i] = uint8(0)
        }

        for i := 0; i < n; i++ {
            for j := 0; j < n; j++ {
                output[i] += input[j] * matrix[i+j*n]
            }
        }
    }
    return float32(time.Since(start).Microseconds()) / float32(iter)
}
func main() {
    for i := 2; i <= 32; i++ {
        fmt.Println(i, ",", matrixtest(i, 100000000))
    }
}
```

- Time measures: built-in golang facilities
- Energy measures: perf
- Intel(R) Xeon(R) CPU E3-1270 v5 @ 3.60GHz
- Go 1.18.2

	N	single op time (us)	single op energy (pJ)	Energy eff
1	2	0.00940269	259200	3.85802E-06
2	3	0.01631968	454200	2.20167E-06
3	4	0.02399984	722200	1.38466E-06
4	5	0.03632906	1070400	9.3423E-07
5	6	0.05176583	1471400	6.79625E-07
6	7	0.07183811	1935800	5.16582E-07
7	8	0.09597738	2737000	3.65364E-07
8	9	0.12337612	3429200	2.91613E-07
9	10	0.16450378	4465900	2.23919E-07
10	11	0.20173332	5530300	1.80822E-07
11	12	0.24265632	6643300	1.50528E-07
12	13	0.28554472	7762600	1.28823E-07
13	14	0.33400026	8954000	1.11682E-07
14	15	0.3861176	10633500	9.40424E-08
15	16	0.44550508	11832200	8.45151E-08
16	17	0.5004054	13604700	7.3504E-08
17	18	0.5867083	15324500	6.5255E-08
18	19	0.63376665	17524400	5.70633E-08
19	20	0.708154	19716100	5.072E-08
20	21	0.7893206	22133800	4.51798E-08
21	22	0.8830365	23525100	4.25078E-08
22	23	0.97467226	27346700	3.65675E-08
23	24	1.1031791	29235100	3.42055E-08

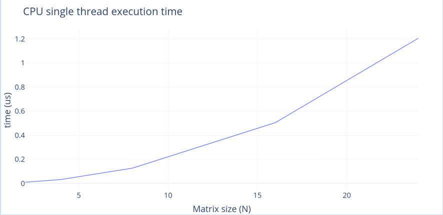


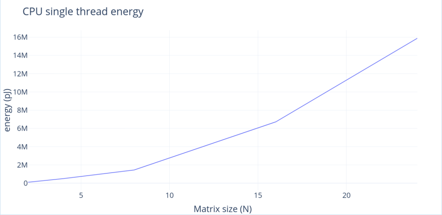


Benchmark: the CPU (C)

- Time measures: time
- Energy measures: perf
- Intel(R) CPU I5-8500 v5 @ 3GHz
- gcc with -O0

	n	single op energy (pJ)	single op time (us)	energy eff
1	2	100000	0.01	0.000008333333333
2	4	500000	0.033	0.000002702702703
3	8	1450000	0.127	0.0000005524861878
4	16	6720000	0.505	0.0000001326259947
5	24	15880000	1.205	0.00000006854009596



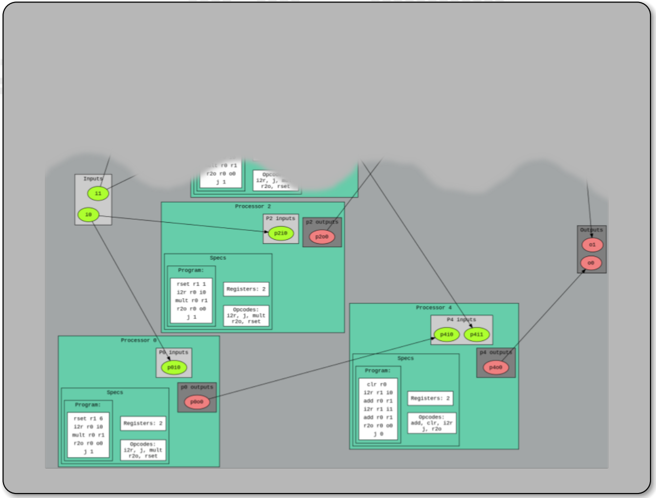


Benchmark: the FPGA

Benchmark an IP is not an easy task.

Fortunately we have a custom design and an FPGA.

We can put the benchmarks tool inside the accelerator.

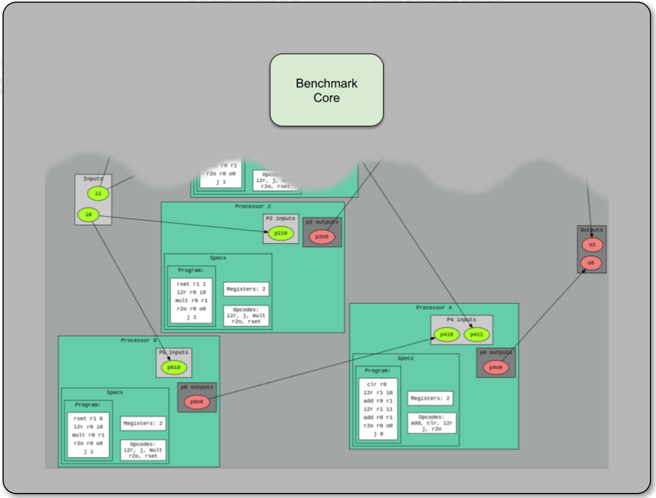


Benchmark: the FPGA

Benchmark an IP is not an easy task.

Fortunately we have a custom design and an FPGA.

We can put the benchmarks tool inside the accelerator.

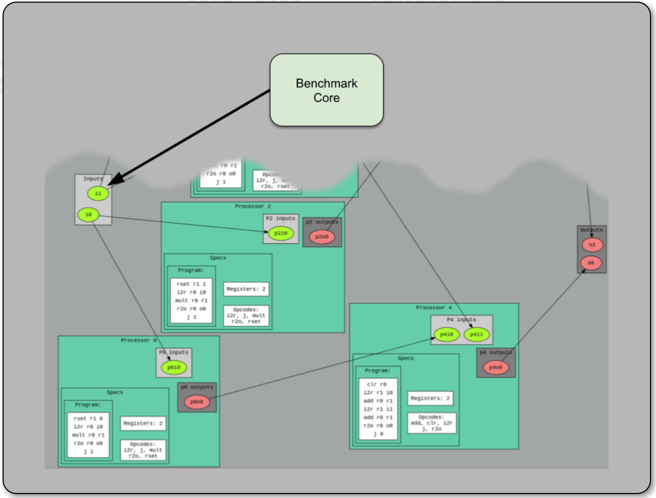


Benchmark: the FPGA

Benchmark an IP is not an easy task.

Fortunately we have a custom design and an FPGA.

We can put the benchmarks tool inside the accelerator.

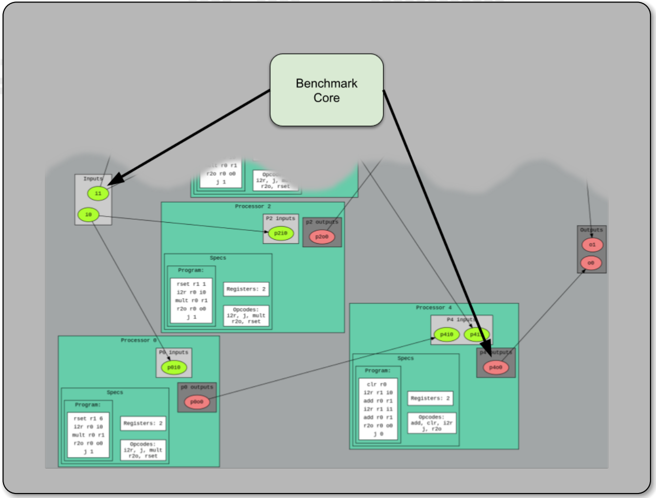


Benchmark: the FPGA

Benchmark an IP is not an easy task.

Fortunately we have a custom design and an FPGA.

We can put the benchmarks tool inside the accelerator.

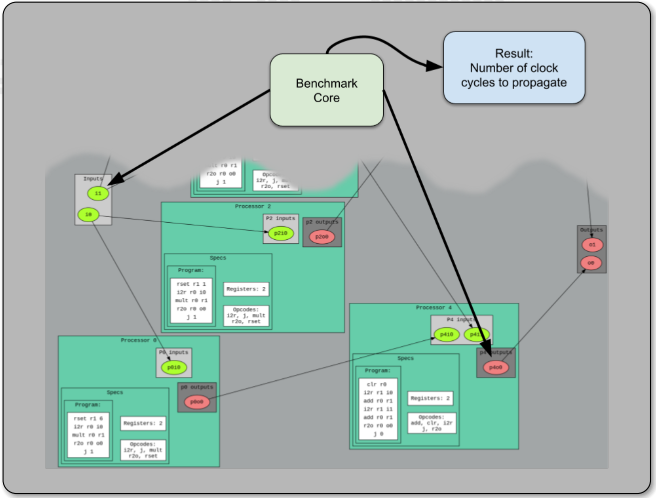


Benchmark: the FPGA

Benchmark an IP is not an easy task.

Fortunately we have a custom design and an FPGA.

We can put the benchmarks tool inside the accelerator.

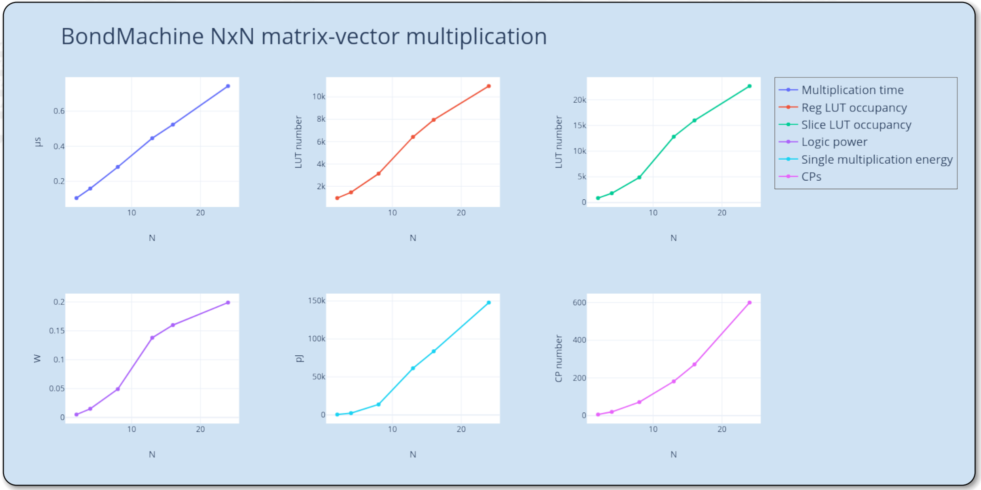


Benchmark core clock cycles distributions

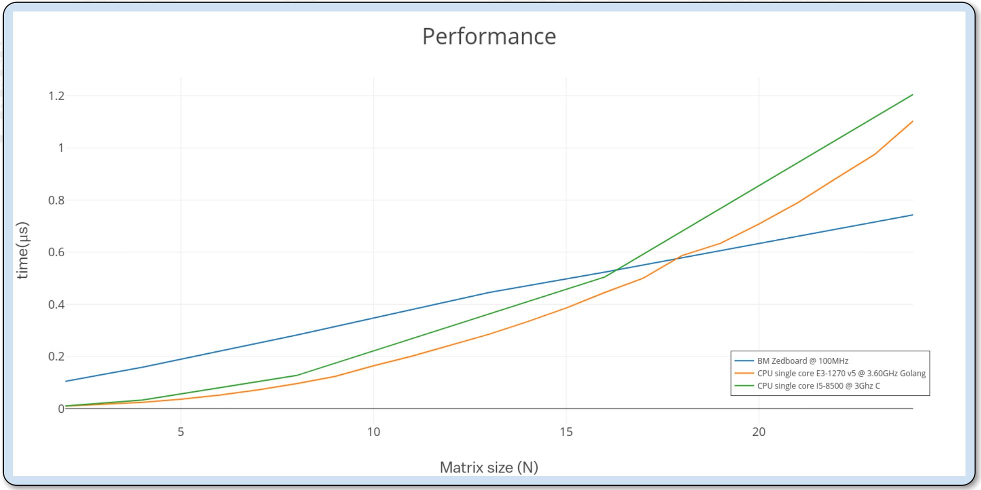
FPGA benchmark summary

	N	single op time (us)	Register LUTs	Slice LUTs	Power	single op energy (pJ)	CPs
1	2	0.1044	947	875	0.005	522	6
2	4	0.1587	1457	1813	0.015	2380.5	20
3	8	0.2819	3131	4897	0.049	13813.1	72
4	13	0.4456	6422	12819	0.138	61492.8	182
5	16	0.5234	7950	15979	0.160	83744	272
6	24	0.7432	10974	22669	0.199	147896.8	600

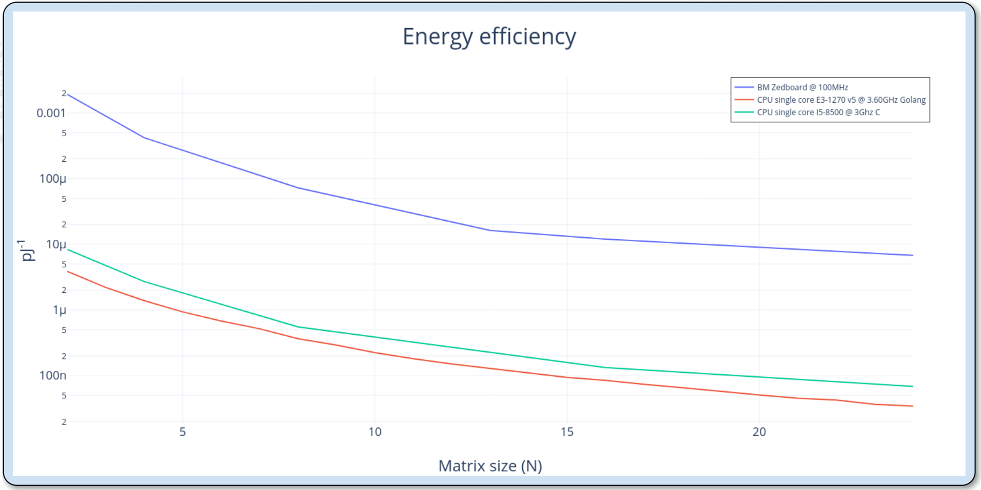
Benchmark core



Comparisons: Performace



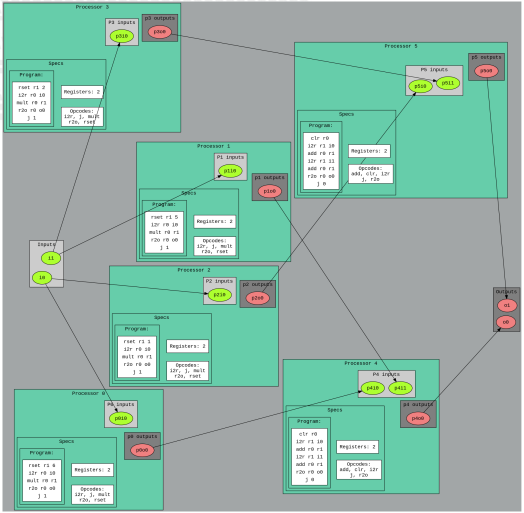
Comparisons: Energy



Misc

BondMachine recap

- The BondMachine is a software ecosystem for the dynamical generation (from several HL types of origin) of computer architectures that can be synthesized of FPGA and
- used as standalone devices,
- as clustered devices,
- and as firmware for computing accelerators.



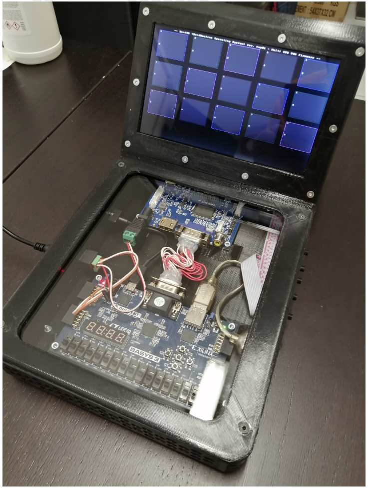
BondMachine recap

- The BondMachine is a software ecosystem for the dynamical generation (from several HL types of origin) of computer architectures that can be synthesized of FPGA and

- used as standalone devices,

- as clustered devices,

- and as firmware for computing accelerators.



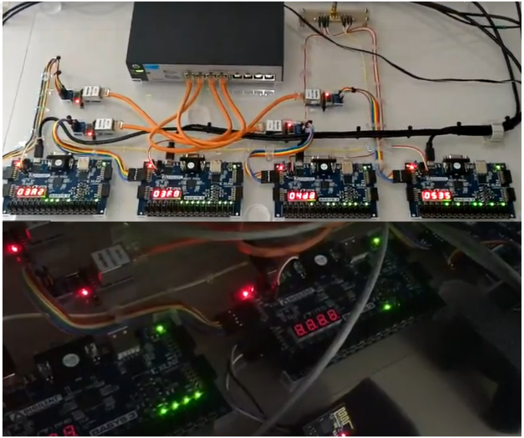
BondMachine recap

- The BondMachine is a software ecosystem for the dynamical generation (from several HL types of origin) of computer architectures that can be synthesized of FPGA and

- used as standalone devices,

- as clustered devices,

- and as firmware for computing accelerators.

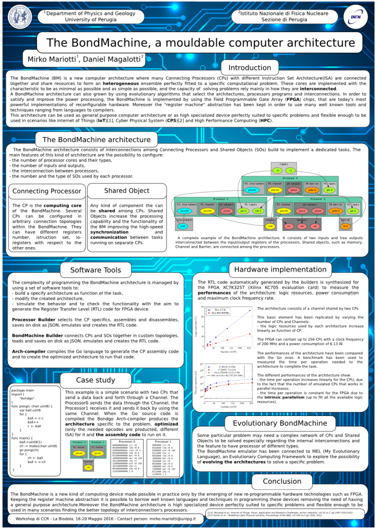


BondMachine recap

- The BondMachine is a software ecosystem for the dynamical generation (from several HL types of origin) of computer architectures that can be synthesized of FPGA and
- used as standalone devices,
- as clustered devices,
- and as firmware for computing accelerators.

Project timeline

- CCR 2015 First ideas, 2016 Poster, 2017 Talk
- InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA
- Invited lectures at: "Advanced Workshop on Modern FPGA Based Technology for Scientific Computing", ICTP 2019
- Invited lectures at: "NiPS Summer School 2019 – Architectures and Algorithms for Energy-Efficient IoT and HPC Applications"
- Golab 2018 talk and ISGC 2019 PoS
- Article published on Parallel Computing, Elsevier 2022
- PON PHD program



Project timeline

- CCR 2015 First ideas, 2016 Poster, 2017 Talk
- InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA
- Invited lectures at: "Advanced Workshop on Modern FPGA Based Technology for Scientific Computing", ICTP 2019
- Invited lectures at: "NiPS Summer School 2019 – Architectures and Algorithms for Energy-Efficient IoT and HPC Applications"
- Golab 2018 talk and ISGC 2019 PoS
- Article published on Parallel Computing, Elsevier 2022
- PON PHD program

Project timeline

- CCR 2015 First ideas, 2016 Poster, 2017 Talk
- InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA
- Invited lectures at: "Advanced Workshop on Modern FPGA Based Technology for Scientific Computing", ICTP 2019
- Invited lectures at: "NiPS Summer School 2019 – Architectures and Algorithms for Energy-Efficient IoT and HPC Applications"
- Golab 2018 talk and ISGC 2019 PoS
- Article published on Parallel Computing, Elsevier 2022
- PON PHD program

Project timeline

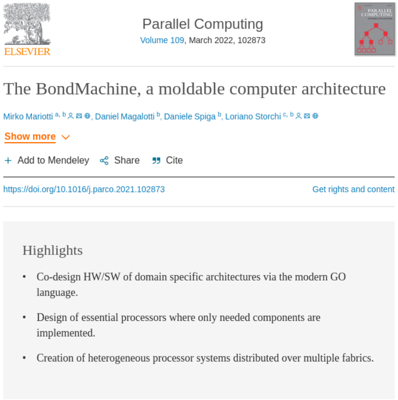
- CCR 2015 First ideas, 2016 Poster, 2017 Talk
- InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA
- Invited lectures at: "Advanced Workshop on Modern FPGA Based Technology for Scientific Computing", ICTP 2019
- Invited lectures at: "NiPS Summer School 2019 – Architectures and Algorithms for Energy-Efficient IoT and HPC Applications"
- Golab 2018 talk and ISGC 2019 PoS
- Article published on Parallel Computing, Elsevier 2022
- PON PHD program

Project timeline

- CCR 2015 First ideas, 2016 Poster, 2017 Talk
- InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA
- Invited lectures at: "Advanced Workshop on Modern FPGA Based Technology for Scientific Computing", ICTP 2019
- Invited lectures at: "NiPS Summer School 2019 – Architectures and Algorithms for Energy-Efficient IoT and HPC Applications"
- Golab 2018 talk and ISGC 2019 PoS
- Article published on Parallel Computing, Elsevier 2022
- PON PHD program

Project timeline

- CCR 2015 First ideas, 2016 Poster, 2017 Talk
- InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA
- Invited lectures at: "Advanced Workshop on Modern FPGA Based Technology for Scientific Computing", ICTP 2019
- Invited lectures at: "NiPS Summer School 2019 – Architectures and Algorithms for Energy-Efficient IoT and HPC Applications"
- Golab 2018 talk and ISGC 2019 PoS
- Article published on Parallel Computing, Elsevier 2022
- PON PHD program


ELSEVIER

Parallel Computing
Volume 109, March 2022, 102873

The BondMachine, a moldable computer architecture

Mirko Mariotti [a, b], Daniel Magalotti [b], Daniele Spiga [b], Loriano Storchi [c, b]

Show more

+ Add to Mendeley Share Cite

https://doi.org/10.1016/j.parco.2021.102873 Get rights and content

Highlights

- Co-design HW/SW of domain specific architectures via the modern GO language.
- Design of essential processors where only needed components are implemented.
- Creation of heterogeneous processor systems distributed over multiple fabrics.

Project timeline

- CCR 2015 First ideas, 2016 Poster, 2017 Talk
- InnovateFPGA 2018 Iron Award, Grand Final at Intel Campus (CA) USA
- Invited lectures at: "Advanced Workshop on Modern FPGA Based Technology for Scientific Computing", ICTP 2019
- Invited lectures at: "NiPS Summer School 2019 – Architectures and Algorithms for Energy-Efficient IoT and HPC Applications"
- Golab 2018 talk and ISGC 2019 PoS
- Article published on Parallel Computing, Elsevier 2022
- PON PHD program

Fabrics

The HDL code for the BondMachine has been tested on these devices/system:

- Digilent Basys3 - Xilinx Artix-7 - Vivado
- Kintex7 Evaluation Board - Vivado
- Digilent Zedboard - Xilinx Zynq 7020 - Vivado
- ZC702 - Xilinx Zynq 7020 - Vivado
- ebaz4205 - Xilinx Zynq 7020 - Vivado
- Linux - Iverilog
- ice40lp1k icefun icebreaker icesugarnano - Lattice - Icestorm
- Terasic De10nano - Intel Cyclone V - Quartus
- Arrow Max1000 - Intel Max10 - Quartus

Within the project other firmware have been written or tested:

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

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

Machine Learning

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

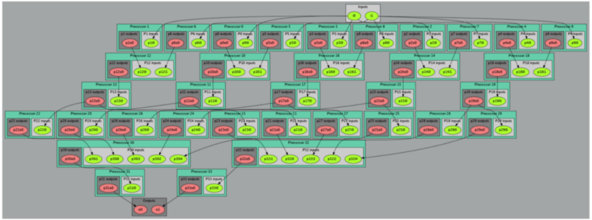
Machine Learning with BondMachine

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()
        }
    }
}
```

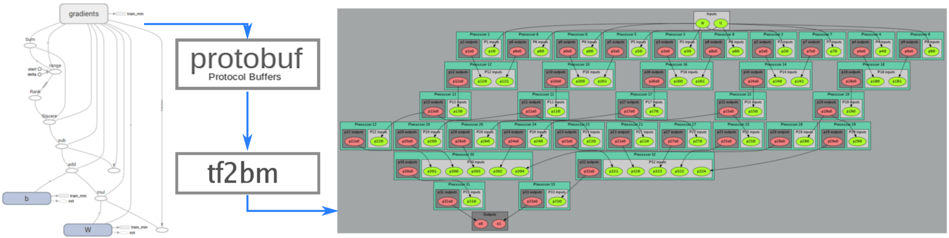


TensorFlow™ to Bondmachine

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

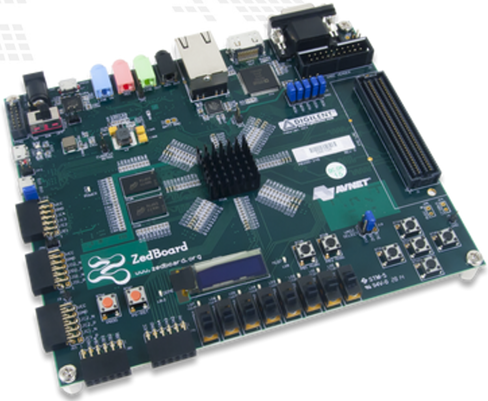
This approch has several advandages over the previous:

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

Specs

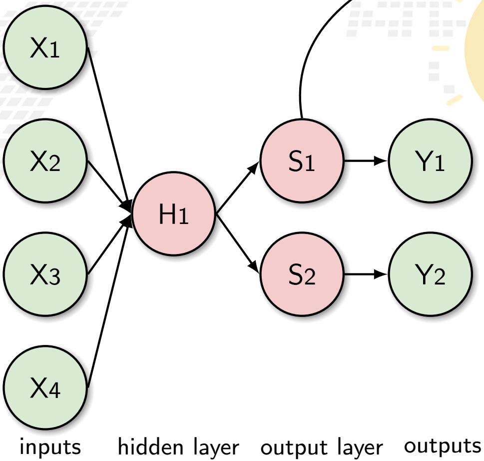
FPGA

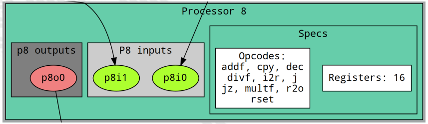
- Digilent Zedboard
- Soc: Zynq XC7Z020-CLG484-1
- 512 MB DDR3
- Vivado 2020.2
- 100MHz
- PYNQ 2.6 (custom build)



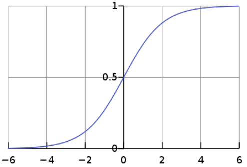
BM inference: A first tentative idea

A neuron of a neural network
can be seen as Connecting Processor of BM





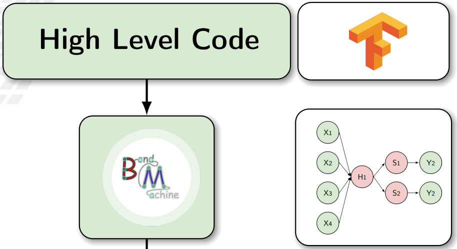
$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$




```
%section softmax .romtext iomode:sync
        entry _start    ; Entry point
_start:
 mov r8, 0f0.0
{{range $y := intRange "0" .Params.inputs}}
{{printf "i2r r1,i%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
        addf    r4, r5
        mov     r6, r2
        divf    r6, r3

        addf    r0, r6

        dec     r7
        jz      r7,exit{{printf "%d" $y}}
        j       loop{{printf "%d" $y}}
exit{{printf "%d" $y}}:
{{$z := atoi $.Params.pos}}
{{if eq $y $z}}
 mov r9, r0
%endsection
```

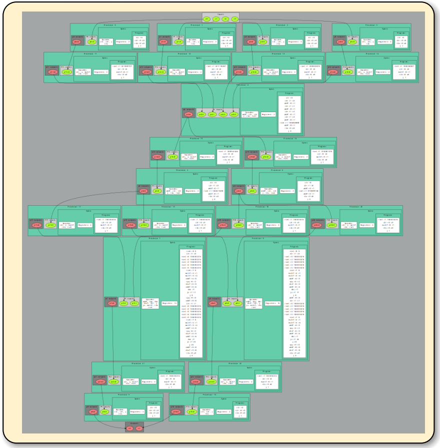
From idea to implementation

Starting from High Level Code, a NN model trained with **TensorFlow** and exported in a standard interpreted by **neuralbond** that converts nodes and weights of the network into a set of heterogeneous processors.




```
neuralbond -net-file banknote.json -neuron-lib-path neurons -save-basm working_dir/bondmachine.basm -config-file
neuralbondconfig.json ; basm  -o working_dir/bondmachine.json working_dir/bondmachine.basm neurons/*.basm
```

Firmware



A first test

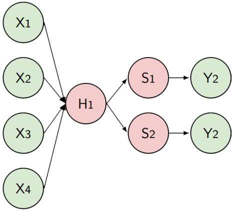
Dataset info:

- **Dataset name**: Banknote Authentication
- **Description**: Dataset on the distinction between genuine and counterfeit banknotes. The data was extracted from images taken from genuine and fake banknote-like samples.
- **N. features**: 4
- **Classification**: binary
- **Samples**: 1097

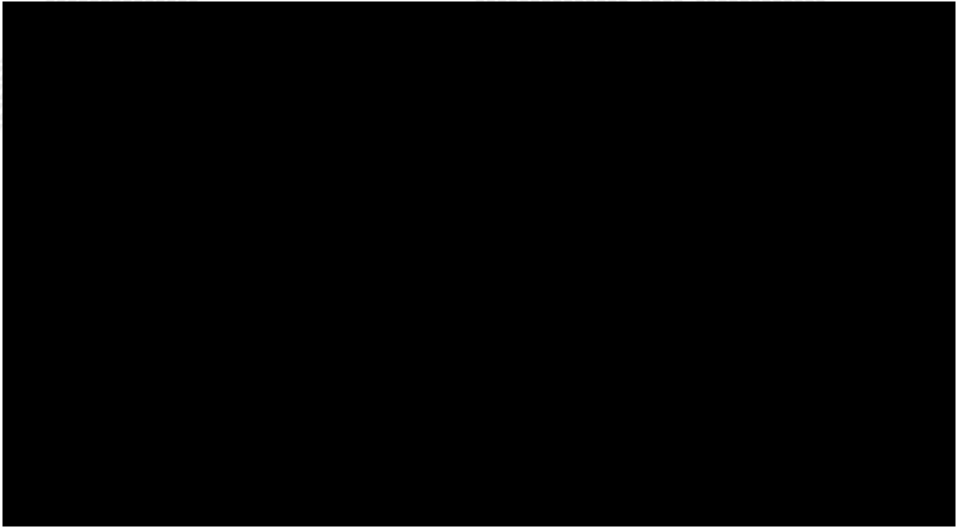
Neural network info:

- **Class**: Multilayer perceptron fully connected
- **Layers**:
  1. An hidden layer with 1 **linear** neuron
  2. One output layer with 2 **softmax** neurons

Graphic representation:



Demo - Train the model



Demo - Train the model

```
[ Command > mkdir Example
```

Demo - Train the model

```
[ Command > mkdir Example
[ Command > cd Example
```

Demo - Train the model

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > git clone https://github.com/BondMachineHQ/ml-zedboard.git
cd ml-zedboard
```

Demo - Train the model

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > git clone https://github.com/BondMachineHQ/ml-zedboard.git
cd ml-zedboard
[ Output  >
Cloning into 'ml-zedboard'...
remote: Enumerating objects: 103, done.
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 103 (delta 38), reused 95 (delta 30), pack-reused 0
Receiving objects: 100% (103/103), 2.70 MiB | 6.33 MiB/s, done.
Resolving deltas: 100% (38/38), done.
```

Demo - Train the model

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > git clone https://github.com/BondMachineHQ/ml-zedboard.git
cd ml-zedboard
[ Output  >
Cloning into 'ml-zedboard'...
remote: Enumerating objects: 103, done.
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 103 (delta 38), reused 95 (delta 30), pack-reused 0
Receiving objects: 100% (103/103), 2.70 MiB | 6.33 MiB/s, done.
Resolving deltas: 100% (38/38), done.
[ Command > ls -al
```

Demo - Train the model

```
remote: Enumerating objects: 103, done.
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 103 (delta 38), reused 95 (delta 30), pack-reused 0
Receiving objects: 100% (103/103), 2.70 MiB | 6.33 MiB/s, done.
Resolving deltas: 100% (38/38), done.
[ Command > ls -al
[ Output  >
total 69
drwx------ 8 mirko users    18 Nov  3 23:20 .
drwx------ 3 mirko users     3 Nov  3 23:20 ..
drwx------ 7 mirko users    12 Nov  3 23:20 .git
-rw------- 1 mirko users  9548 Nov  3 23:20 README.md
-rwx------ 1 mirko users    25 Nov  3 23:20 activate_environment.sh
-rw------- 1 mirko users  5818 Nov  3 23:20 analyze.py
-rw------- 1 mirko users  7515 Nov  3 23:20 analyze_output.py
-rw------- 1 mirko users 18799 Nov  3 23:20 bmtrain.py
drwx------ 2 mirko users    11 Nov  3 23:20 images
-rw------- 1 mirko users  2229 Nov  3 23:20 main.py
drwx------ 2 mirko users     3 Nov  3 23:20 notebooks
drwx------ 3 mirko users     3 Nov  3 23:20 outputs
drwx------ 4 mirko users     4 Nov  3 23:20 reports
-rw------- 1 mirko users    69 Nov  3 23:20 requirements.txt
drwx------ 2 mirko users     4 Nov  3 23:20 resources
-rwx------ 1 mirko users    43 Nov  3 23:20 setup_enviroment.sh
-rw------- 1 mirko users   519 Nov  3 23:20 specifics.json
-rw------- 1 mirko users   559 Nov  3 23:20 utils.txt
```

Demo - Train the model

```
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 103 (delta 38), reused 95 (delta 30), pack-reused 0
Receiving objects: 100% (103/103), 2.70 MiB | 6.33 MiB/s, done.
Resolving deltas: 100% (38/38), done.
[ Command > ls -al
[ Output  >
total 69
drwx------ 8 mirko users    18 Nov  3 23:20 .
drwx------ 3 mirko users     3 Nov  3 23:20 ..
drwx------ 7 mirko users    12 Nov  3 23:20 .git
-rw------- 1 mirko users  9548 Nov  3 23:20 README.md
-rwx------ 1 mirko users    25 Nov  3 23:20 activate_environment.sh
-rw------- 1 mirko users  5818 Nov  3 23:20 analyze.py
-rw------- 1 mirko users  7515 Nov  3 23:20 analyze_output.py
-rw------- 1 mirko users 18799 Nov  3 23:20 bmtrain.py
drwx------ 2 mirko users    11 Nov  3 23:20 images
-rw------- 1 mirko users  2229 Nov  3 23:20 main.py
drwx------ 2 mirko users     3 Nov  3 23:20 notebooks
drwx------ 3 mirko users     3 Nov  3 23:20 outputs
drwx------ 4 mirko users     4 Nov  3 23:20 reports
-rw------- 1 mirko users    69 Nov  3 23:20 requirements.txt
drwx------ 2 mirko users     4 Nov  3 23:20 resources
-rwx------ 1 mirko users    43 Nov  3 23:20 setup_enviroment.sh
-rw------- 1 mirko users   519 Nov  3 23:20 specifics.json
-rw------- 1 mirko users   559 Nov  3 23:20 utils.txt
[ Command > conda create --name ml-zedboard -y python==3.8.0
```

Demo - Train the model

```
  libstdcxx-ng       pkgs/main/linux-64::libstdcxx-ng-11.2.0-h1234567_1 None
  ncurses            pkgs/main/linux-64::ncurses-6.3-h5eee18b_3 None
  openssl            pkgs/main/linux-64::openssl-1.1.1q-h7f8727e_0 None
  pip                pkgs/main/linux-64::pip-22.2.2-py38h06a4308_0 None
  python             pkgs/main/linux-64::python-3.8.0-h0371630_2 None
  readline           pkgs/main/linux-64::readline-7.0-h7b6447c_5 None
  setuptools         pkgs/main/linux-64::setuptools-65.5.0-py38h06a4308_0 None
  sqlite             pkgs/main/linux-64::sqlite-3.33.0-h62c20be_0 None
  tk                 pkgs/main/linux-64::tk-8.6.12-h1ccaba5_0 None
  wheel              pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0 None
  xz                 pkgs/main/linux-64::xz-5.2.6-h5eee18b_0 None
  zlib               pkgs/main/linux-64::zlib-1.2.13-h5eee18b_0 None


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ml-zedboard
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Retrieving notices: ...working... done
```

Demo - Train the model

```
  ncurses            pkgs/main/linux-64::ncurses-6.3-h5eee18b_3 None
  openssl            pkgs/main/linux-64::openssl-1.1.1q-h7f8727e_0 None
  pip                pkgs/main/linux-64::pip-22.2.2-py38h06a4308_0 None
  python             pkgs/main/linux-64::python-3.8.0-h0371630_2 None
  readline           pkgs/main/linux-64::readline-7.0-h7b6447c_5 None
  setuptools         pkgs/main/linux-64::setuptools-65.5.0-py38h06a4308_0 None
  sqlite             pkgs/main/linux-64::sqlite-3.33.0-h62c20be_0 None
  tk                 pkgs/main/linux-64::tk-8.6.12-h1ccaba5_0 None
  wheel              pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0 None
  xz                 pkgs/main/linux-64::xz-5.2.6-h5eee18b_0 None
  zlib               pkgs/main/linux-64::zlib-1.2.13-h5eee18b_0 None


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ml-zedboard
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Retrieving notices: ...working... done
[ Command > conda activate ml-zedboard
```

Demo - Train the model

```
  ncurses            pkgs/main/linux-64::ncurses-6.3-h5eee18b_3 None
  openssl            pkgs/main/linux-64::openssl-1.1.1q-h7f8727e_0 None
  pip                pkgs/main/linux-64::pip-22.2.2-py38h06a4308_0 None
  python             pkgs/main/linux-64::python-3.8.0-h0371630_2 None
  readline           pkgs/main/linux-64::readline-7.0-h7b6447c_5 None
  setuptools         pkgs/main/linux-64::setuptools-65.5.0-py38h06a4308_0 None
  sqlite             pkgs/main/linux-64::sqlite-3.33.0-h62c20be_0 None
  tk                 pkgs/main/linux-64::tk-8.6.12-h1ccaba5_0 None
  wheel              pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0 None
  xz                 pkgs/main/linux-64::xz-5.2.6-h5eee18b_0 None
  zlib               pkgs/main/linux-64::zlib-1.2.13-h5eee18b_0 None


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ml-zedboard
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Retrieving notices: ...working... done
[ Command > conda activate ml-zedboard
```

Demo - Train the model

```
  openssl            pkgs/main/linux-64::openssl-1.1.1q-h7f8727e_0 None
  pip                pkgs/main/linux-64::pip-22.2.2-py38h06a4308_0 None
  python             pkgs/main/linux-64::python-3.8.0-h0371630_2 None
  readline           pkgs/main/linux-64::readline-7.0-h7b6447c_5 None
  setuptools         pkgs/main/linux-64::setuptools-65.5.0-py38h06a4308_0 None
  sqlite             pkgs/main/linux-64::sqlite-3.33.0-h62c20be_0 None
  tk                 pkgs/main/linux-64::tk-8.6.12-h1ccaba5_0 None
  wheel              pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0 None
  xz                 pkgs/main/linux-64::xz-5.2.6-h5eee18b_0 None
  zlib               pkgs/main/linux-64::zlib-1.2.13-h5eee18b_0 None


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ml-zedboard
#
# To deactivate an active environment, use
#
#     $ conda deactivate

Retrieving notices: ...working... done
[ Command > conda activate ml-zedboard
[ Command > pip3 install -r requirements.txt
```

Demo - Train the model

```
Collecting MarkupSafe>=2.1.1
  Using cached MarkupSafe-2.1.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Collecting zipp>=0.5
  Using cached zipp-3.10.0-py3-none-any.whl (6.2 kB)
Collecting pyasn1<0.5.0,>=0.4.6
  Using cached pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
Collecting oauthlib>=3.0.0
  Using cached oauthlib-3.2.2-py3-none-any.whl (151 kB)
Installing collected packages: tensorboard-plugin-wit, pytz, pyasn1, libclang, keras, flatbuffers, z
ipp, xlrd, wrapt, urllib3, typing-extensions, threadpoolctl, termcolor, tensorflow-io-gcs-filesystem
, tensorflow-estimator, tensorboard-data-server, six, rsa, pyyaml, pyparsing, pyasn1-modules, protob
uf, pillow, oauthlib, numpy, networkx, MarkupSafe, kiwisolver, joblib, idna, gast, fonttools, cycler
, charset-normalizer, cachetools, absl-py, werkzeug, scipy, requests, python-dateutil, pydot, packag
ing, opt-einsum, onnx, keras-preprocessing, importlib-metadata, h5py, grpcio, google-pasta, google-a
uth, contourpy, astunparse, scikit-learn, requests-oauthlib, pandas, matplotlib, markdown, hls4ml, s
klearn, google-auth-oauthlib, tensorboard, tensorflow
Successfully installed MarkupSafe-2.1.1 absl-py-1.3.0 astunparse-1.6.3 cachetools-5.2.0 charset-norm
alizer-2.1.1 contourpy-1.0.6 cycler-0.11.0 flatbuffers-22.10.26 fonttools-4.38.0 gast-0.4.0 google-a
uth-2.14.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.50.0 h5py-3.7.0 hls4ml-0.6.0 idna-
3.4 importlib-metadata-5.0.0 joblib-1.2.0 keras-2.10.0 keras-preprocessing-1.1.2 kiwisolver-1.4.4 li
bclang-14.0.6 markdown-3.4.1 matplotlib-3.6.2 networkx-2.8.8 numpy-1.23.4 oauthlib-3.2.2 onnx-1.12.0
 opt-einsum-3.3.0 packaging-21.3 pandas-1.5.1 pillow-9.3.0 protobuf-3.19.6 pyasn1-0.4.8 pyasn1-modul
es-0.2.8 pydot-1.4.2 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.6 pyyaml-6.0 requests-2.28.1 re
quests-oauthlib-1.3.1 rsa-4.9 scikit-learn-1.1.3 scipy-1.9.3 six-1.16.0 sklearn-0.0 tensorboard-2.10
.1 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.10.0 tensorflow-estimator
-2.10.0 tensorflow-io-gcs-filesystem-0.27.0 termcolor-2.1.0 threadpoolctl-3.1.0 typing-extensions-4.
4.0 urllib3-1.26.12 werkzeug-2.2.2 wrapt-1.14.1 xlrd-2.0.1 zipp-3.10.0
```

Demo - Train the model

```
  Using cached MarkupSafe-2.1.1-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (25 kB)
Collecting zipp>=0.5
  Using cached zipp-3.10.0-py3-none-any.whl (6.2 kB)
Collecting pyasn1<0.5.0,>=0.4.6
  Using cached pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
Collecting oauthlib>=3.0.0
  Using cached oauthlib-3.2.2-py3-none-any.whl (151 kB)
Installing collected packages: tensorboard-plugin-wit, pytz, pyasn1, libclang, keras, flatbuffers, z
ipp, xlrd, wrapt, urllib3, typing-extensions, threadpoolctl, termcolor, tensorflow-io-gcs-filesystem
, tensorflow-estimator, tensorboard-data-server, six, rsa, pyyaml, pyparsing, pyasn1-modules, protob
uf, pillow, oauthlib, numpy, networkx, MarkupSafe, kiwisolver, joblib, idna, gast, fonttools, cycler
, charset-normalizer, cachetools, absl-py, werkzeug, scipy, requests, python-dateutil, pydot, packag
ing, opt-einsum, onnx, keras-preprocessing, importlib-metadata, h5py, grpcio, google-pasta, google-a
uth, contourpy, astunparse, scikit-learn, requests-oauthlib, pandas, matplotlib, markdown, hls4ml, s
klearn, google-auth-oauthlib, tensorboard, tensorflow
Successfully installed MarkupSafe-2.1.1 absl-py-1.3.0 astunparse-1.6.3 cachetools-5.2.0 charset-norm
alizer-2.1.1 contourpy-1.0.6 cycler-0.11.0 flatbuffers-22.10.26 fonttools-4.38.0 gast-0.4.0 google-a
uth-2.14.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.50.0 h5py-3.7.0 hls4ml-0.6.0 idna-
3.4 importlib-metadata-5.0.0 joblib-1.2.0 keras-2.10.0 keras-preprocessing-1.1.2 kiwisolver-1.4.4 li
bclang-14.0.6 markdown-3.4.1 matplotlib-3.6.2 networkx-2.8.8 numpy-1.23.4 oauthlib-3.2.2 onnx-1.12.0
 opt-einsum-3.3.0 packaging-21.3 pandas-1.5.1 pillow-9.3.0 protobuf-3.19.6 pyasn1-0.4.8 pyasn1-modul
es-0.2.8 pydot-1.4.2 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.6 pyyaml-6.0 requests-2.28.1 re
quests-oauthlib-1.3.1 rsa-4.9 scikit-learn-1.1.3 scipy-1.9.3 six-1.16.0 sklearn-0.0 tensorboard-2.10
.1 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.10.0 tensorflow-estimator
-2.10.0 tensorflow-io-gcs-filesystem-0.27.0 termcolor-2.1.0 threadpoolctl-3.1.0 typing-extensions-4.
4.0 urllib3-1.26.12 werkzeug-2.2.2 wrapt-1.14.1 xlrd-2.0.1 zipp-3.10.0
[ Command > ls -al main.py bmtrain.py banknote-authentication*
```

Demo - Train the model

```
  Using cached pyasn1-0.4.8-py2.py3-none-any.whl (77 kB)
Collecting oauthlib>=3.0.0
  Using cached oauthlib-3.2.2-py3-none-any.whl (151 kB)
Installing collected packages: tensorboard-plugin-wit, pytz, pyasn1, libclang, keras, flatbuffers, z
ipp, xlrd, wrapt, urllib3, typing-extensions, threadpoolctl, termcolor, tensorflow-io-gcs-filesystem
, tensorflow-estimator, tensorboard-data-server, six, rsa, pyyaml, pyparsing, pyasn1-modules, protob
uf, pillow, oauthlib, numpy, networkx, MarkupSafe, kiwisolver, joblib, idna, gast, fonttools, cycler
, charset-normalizer, cachetools, absl-py, werkzeug, scipy, requests, python-dateutil, pydot, packag
ing, opt-einsum, onnx, keras-preprocessing, importlib-metadata, h5py, grpcio, google-pasta, google-a
uth, contourpy, astunparse, scikit-learn, requests-oauthlib, pandas, matplotlib, markdown, hls4ml, s
klearn, google-auth-oauthlib, tensorboard, tensorflow
Successfully installed MarkupSafe-2.1.1 absl-py-1.3.0 astunparse-1.6.3 cachetools-5.2.0 charset-norm
alizer-2.1.1 contourpy-1.0.6 cycler-0.11.0 flatbuffers-22.10.26 fonttools-4.38.0 gast-0.4.0 google-a
uth-2.14.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.50.0 h5py-3.7.0 hls4ml-0.6.0 idna-
3.4 importlib-metadata-5.0.0 joblib-1.2.0 keras-2.10.0 keras-preprocessing-1.1.2 kiwisolver-1.4.4 li
bclang-14.0.6 markdown-3.4.1 matplotlib-3.6.2 networkx-2.8.8 numpy-1.23.4 oauthlib-3.2.2 onnx-1.12.0
 opt-einsum-3.3.0 packaging-21.3 pandas-1.5.1 pillow-9.3.0 protobuf-3.19.6 pyasn1-0.4.8 pyasn1-modul
es-0.2.8 pydot-1.4.2 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.6 pyyaml-6.0 requests-2.28.1 re
quests-oauthlib-1.3.1 rsa-4.9 scikit-learn-1.1.3 scipy-1.9.3 six-1.16.0 sklearn-0.0 tensorboard-2.10
.1 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.10.0 tensorflow-estimator
-2.10.0 tensorflow-io-gcs-filesystem-0.27.0 termcolor-2.1.0 threadpoolctl-3.1.0 typing-extensions-4.
4.0 urllib3-1.26.12 werkzeug-2.2.2 wrapt-1.14.1 xlrd-2.0.1 zipp-3.10.0
[ Command > ls -al main.py bmtrain.py banknote-authentication*
[ Output  >
ls: cannot access 'banknote-authentication*': No such file or directory
-rw------- 1 mirko users 18799 Nov  3 23:20  bmtrain.py
-rw------- 1 mirko users  2229 Nov  3 23:20  main.py
```

Demo - Train the model

```
Installing collected packages: tensorboard-plugin-wit, pytz, pyasn1, libclang, keras, flatbuffers, z
ipp, xlrd, wrapt, urllib3, typing-extensions, threadpoolctl, termcolor, tensorflow-io-gcs-filesystem
, tensorflow-estimator, tensorboard-data-server, six, rsa, pyyaml, pyparsing, pyasn1-modules, protob
uf, pillow, oauthlib, numpy, networkx, MarkupSafe, kiwisolver, joblib, idna, gast, fonttools, cycler
, charset-normalizer, cachetools, absl-py, werkzeug, scipy, requests, python-dateutil, pydot, packag
ing, opt-einsum, onnx, keras-preprocessing, importlib-metadata, h5py, grpcio, google-pasta, google-a
uth, contourpy, astunparse, scikit-learn, requests-oauthlib, pandas, matplotlib, markdown, hls4ml, s
klearn, google-auth-oauthlib, tensorboard, tensorflow
Successfully installed MarkupSafe-2.1.1 absl-py-1.3.0 astunparse-1.6.3 cachetools-5.2.0 charset-norm
alizer-2.1.1 contourpy-1.0.6 cycler-0.11.0 flatbuffers-22.10.26 fonttools-4.38.0 gast-0.4.0 google-a
uth-2.14.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.50.0 h5py-3.7.0 hls4ml-0.6.0 idna-
3.4 importlib-metadata-5.0.0 joblib-1.2.0 keras-2.10.0 keras-preprocessing-1.1.2 kiwisolver-1.4.4 li
bclang-14.0.6 markdown-3.4.1 matplotlib-3.6.2 networkx-2.8.8 numpy-1.23.4 oauthlib-3.2.2 onnx-1.12.0
 opt-einsum-3.3.0 packaging-21.3 pandas-1.5.1 pillow-9.3.0 protobuf-3.19.6 pyasn1-0.4.8 pyasn1-modul
es-0.2.8 pydot-1.4.2 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.6 pyyaml-6.0 requests-2.28.1 re
quests-oauthlib-1.3.1 rsa-4.9 scikit-learn-1.1.3 scipy-1.9.3 six-1.16.0 sklearn-0.0 tensorboard-2.10
.1 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.1 tensorflow-2.10.0 tensorflow-estimator
-2.10.0 tensorflow-io-gcs-filesystem-0.27.0 termcolor-2.1.0 threadpoolctl-3.1.0 typing-extensions-4.
4.0 urllib3-1.26.12 werkzeug-2.2.2 wrapt-1.14.1 xlrd-2.0.1 zipp-3.10.0
[ Command > ls -al main.py bmtrain.py banknote-authentication*
[ Output  >
ls: cannot access 'banknote-authentication*': No such file or directory
-rw------- 1 mirko users 18799 Nov  3 23:20  bmtrain.py
-rw------- 1 mirko users  2229 Nov  3 23:20  main.py
[ Command >
export PYTHONPATH=/tmp/tmptj5_gk0p/Example/ml-zedboard
python-inspect -m bmtrain -o build_model 2> /dev/null | pygmentize -l python | head -n 20
```

Demo - Train the model

```
[ Command >
export PYTHONPATH=/tmp/tmptj5_gk0p/Example/ml-zedboard
python-inspect -m bmtrain -o build_model 2> /dev/null | pygmentize -l python | head -n 20
[ Output  >
    def build_model(self):
        if self.nn_model_type == "MLP":

            self.model = Sequential()

            self.parse_network_specifics()

            if self.network_spec == None:
                for i in range(0, 24, 3):
                    self.model.add(Dense(i, input_shape=(self.X_train_val.shape[1],)))
                for i in reversed(range(0, 24, 3)):
                    self.model.add(Dense(i, input_shape=(self.X_train_val.shape[1],)))
                opt = Adam(lr=0.0001)
            else:
                arch = self.network_spec["network"]["arch"]
                for i in range(0, len(arch)):
                    layer_name = self.network_spec["network"]["arch"][i]["layer_name"]
                    activation_function = self.network_spec["network"]["arch"][i]["activation_functi
on"]
                    neurons = self.network_spec["network"]["arch"][i]["neurons"]
                    if i == 0:
Exception ignored in: <_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>
BrokenPipeError: [Errno 32] Broken pipe
```

Demo - Train the model

```
    def build_model(self):
        if self.nn_model_type == "MLP":

            self.model = Sequential()

            self.parse_network_specifics()

            if self.network_spec == None:
                for i in range(0, 24, 3):
                    self.model.add(Dense(i, input_shape=(self.X_train_val.shape[1],)))
                for i in reversed(range(0, 24, 3)):
                    self.model.add(Dense(i, input_shape=(self.X_train_val.shape[1],)))
                opt = Adam(lr=0.0001)
            else:
                arch = self.network_spec["network"]["arch"]
                for i in range(0, len(arch)):
                    layer_name = self.network_spec["network"]["arch"][i]["layer_name"]
                    activation_function = self.network_spec["network"]["arch"][i]["activation_functi
on"]
                    neurons = self.network_spec["network"]["arch"][i]["neurons"]
                    if i == 0:
Exception ignored in: <_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>
BrokenPipeError: [Errno 32] Broken pipe
[ Command >
export PYTHONPATH=/tmp/tmptj5_gk0p/Example/ml-zedboard
python-inspect -m bmtrain -o dump_json_for_bondmachine 2> /dev/null | pygmentize -l python | head -n
 20
```

Demo - Train the model

```
[ Command >
export PYTHONPATH=/tmp/tmptj5_gk0p/Example/ml-zedboard
python-inspect -m bmtrain -o dump_json_for_bondmachine 2> /dev/null | pygmentize -l python | head -n
 20
[ Output  >
    def dump_json_for_bondmachine(self):
        import json

        layers = self.model.layers
        weights = self.model.weights

        to_dump = {}

        weights = []
        nodes = []

        # save weigths
        for i in range(0 , len(layers)):

            layer_weights = layers[i].get_weights()

            for m in range(0, len(layer_weights)):
                for w in range(0, len(layer_weights[m])):
                    try:
                        for v in range(0, len(layer_weights[m][w])):
Exception ignored in: <_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>
BrokenPipeError: [Errno 32] Broken pipe
```

Demo - Train the model

```
export PYTHONPATH=/tmp/tmptj5_gk0p/Example/ml-zedboard
python-inspect -m bmtrain -o dump_json_for_bondmachine 2> /dev/null | pygmentize -l python | head -n
 20
[ Output  >
    def dump_json_for_bondmachine(self):
        import json

        layers = self.model.layers
        weights = self.model.weights

        to_dump = {}

        weights = []
        nodes = []

        # save weigths
        for i in range(0 , len(layers)):

            layer_weights = layers[i].get_weights()

            for m in range(0, len(layer_weights)):
                for w in range(0, len(layer_weights[m])):
                    try:
                        for v in range(0, len(layer_weights[m][w])):
Exception ignored in: <_io.TextIOWrapper name='<stdout>' mode='w' encoding='utf-8'>
BrokenPipeError: [Errno 32] Broken pipe
[ Command > python3 main.py --dataset banknote-authentication -m MLP
```

Demo - Train the model

```
822/822 [==============================] - 0s 37us/sample - loss: 0.4611 - acc: 0.9599 - val_loss: 0
.4695 - val_acc: 0.9636
 *** dump model
 # INFO: Training finished, saved model path: models/banknote-authentication_KERAS_model.h5
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 5

 dense_1 (Dense)             (None, 2)                 4

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
None
/tools/Conda/envs/ml-zedboard/lib/python3.8/site-packages/keras/engine/training_v1.py:2356: UserWarn
ing: `Model.state_updates` will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
Sofware predicions have been exported in CSV (path is: datasets/banknote-authentication_swprediction
.csv)
 # INFO: Accuracy is 0.9454545454545454
Model has been exported in JSON for Bondmachine (path is: models/banknote-authentication/modelBM.jso
n)
```

Demo - Train the model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 5

 dense_1 (Dense)             (None, 2)                 4

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
None
/tools/Conda/envs/ml-zedboard/lib/python3.8/site-packages/keras/engine/training_v1.py:2356: UserWarn
ing: `Model.state_updates` will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
Sofware predicions have been exported in CSV (path is: datasets/banknote-authentication_swprediction
.csv)
 # INFO: Accuracy is 0.9454545454545454
Model has been exported in JSON for Bondmachine (path is: models/banknote-authentication/modelBM.jso
n)
[ Command >
cp models/banknote-authentication/modelBM.json /tmp/modelBM.json
cp datasets/banknote-authentication_swprediction.csv /tmp/sw.csv
cp datasets/banknote-authentication_sample.csv /tmp/sample.csv
```

Demo - Train the model

```
______________________________________________________________
 Layer (type)                Output Shape              Param #
==============================================================
 dense (Dense)               (None, 1)                 5

 dense_1 (Dense)             (None, 2)                 4

==============================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
______________________________________________________________
None
/tools/Conda/envs/ml-zedboard/lib/python3.8/site-packages/keras/engine/training_v1.py:2356: UserWarn
ing: `Model.state_updates` will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
Sofware predicions have been exported in CSV (path is: datasets/banknote-authentication_swprediction
.csv)
 # INFO: Accuracy is 0.9454545454545454
Model has been exported in JSON for Bondmachine (path is: models/banknote-authentication/modelBM.jso
n)
[ Command >
cp models/banknote-authentication/modelBM.json /tmp/modelBM.json
cp datasets/banknote-authentication_swprediction.csv /tmp/sw.csv
cp datasets/banknote-authentication_sample.csv /tmp/sample.csv
[ Output  >
```

Demo - Train the model

```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 5

 dense_1 (Dense)             (None, 2)                 4

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
None
/tools/Conda/envs/ml-zedboard/lib/python3.8/site-packages/keras/engine/training_v1.py:2356: UserWarn
ing: `Model.state_updates` will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
Sofware predicions have been exported in CSV (path is: datasets/banknote-authentication_swprediction
.csv)
 # INFO: Accuracy is 0.9454545454545454
Model has been exported in JSON for Bondmachine (path is: models/banknote-authentication/modelBM.jso
n)
[ Command >
cp models/banknote-authentication/modelBM.json /tmp/modelBM.json
cp datasets/banknote-authentication_swprediction.csv /tmp/sw.csv
cp datasets/banknote-authentication_sample.csv /tmp/sample.csv
[ Output  >
[ Command > conda deactivate ; conda env remove --name ml-zedboard
```

Demo - Train the model

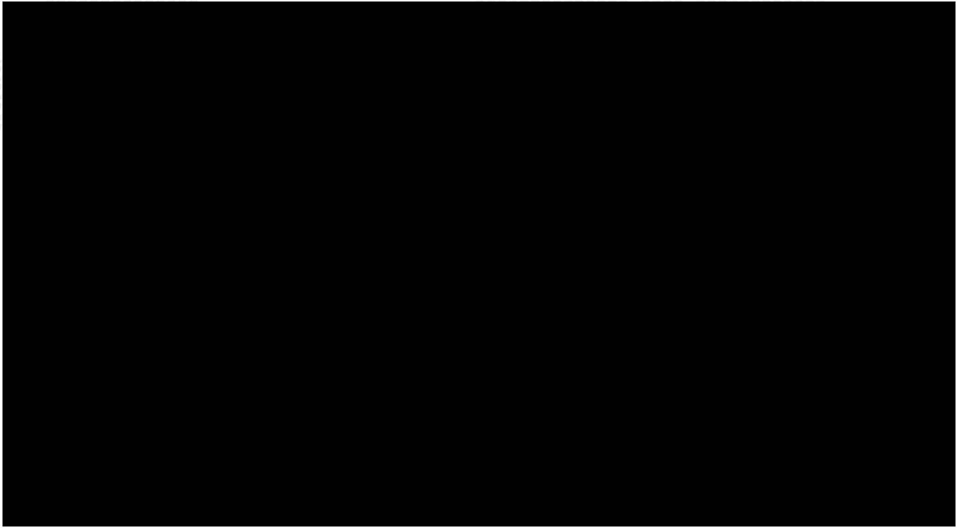
```
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 1)                 5

 dense_1 (Dense)             (None, 2)                 4

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
None
/tools/Conda/envs/ml-zedboard/lib/python3.8/site-packages/keras/engine/training_v1.py:2356: UserWarn
ing: `Model.state_updates` will be removed in a future version. This property should not be used in
TensorFlow 2.0, as `updates` are applied automatically.
  updates=self.state_updates,
Sofware predicions have been exported in CSV (path is: datasets/banknote-authentication_swprediction
.csv)
 # INFO: Accuracy is 0.9454545454545454
Model has been exported in JSON for Bondmachine (path is: models/banknote-authentication/modelBM.jso
n)
[ Command >
cp models/banknote-authentication/modelBM.json /tmp/modelBM.json
cp datasets/banknote-authentication_swprediction.csv /tmp/sw.csv
cp datasets/banknote-authentication_sample.csv /tmp/sample.csv
[ Output  >
[ Command > conda deactivate ; conda env remove --name ml-zedboard
```

Project creation

The outcome of this first part of the demo are three files:

- sample.csv is a test dataset that will be used to feed the inferences of both: the BM hardware and the BM simulation
- sw.csv is the software predictions over that dataset and will be used to check the BM inference probabilities and predictions
- modelBM.json is the trained network that will use as BM source in the next demo

DEMO - BondMachine creation



DEMO - BondMachine creation

DEMO - BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
```

DEMO - BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

DEMO - BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

DEMO - BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command >
cd proj_mlinfn
ls -al
```

DEMO - BondMachine creation

```
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command >
cd proj_mlinfn
ls -al
[ Output  >
total 10
drwx------ 3 mirko users    20 Nov  2 22:21 .
drwx------ 3 mirko users    17 Nov  2 22:21 ..
-rw------- 1 mirko users 44179 Nov  2 22:21 Makefile
-rw------- 1 mirko users   397 Nov  2 22:21 authorized_keys
-rw------- 1 mirko users  1962 Nov  2 22:21 banknote.json
-rw------- 1 mirko users   150 Nov  2 22:21 bmapi.json
-rw------- 1 mirko users   242 Nov  2 22:21 bmapi.mk
-rw------- 1 mirko users  1351 Nov  2 22:21 bminfo.json
-rw------- 1 mirko users   130 Nov  2 22:21 buildroot.mk
-rw------- 1 mirko users   129 Nov  2 22:21 crosscompile.mk
-rwx------ 1 mirko users  3613 Nov  2 22:21 deploy_jupyter_board.py
-rw------- 1 mirko users   495 Nov  2 22:21 local.mk
-rw------- 1 mirko users   145 Nov  2 22:21 neuralbondconfig.json
drwx------ 2 mirko users    20 Nov  2 22:21 neurons
-rw------- 1 mirko users   145 Nov  2 22:21 simbatch.mk
-rwx------ 1 mirko users  4059 Nov  2 22:21 simbatch.py
-rw------- 1 mirko users 24057 Nov  2 22:21 simbatch_input.csv
-rw------- 1 mirko users  1100 Nov  2 22:21 sumapp.go
-rw------- 1 mirko users 21319 Nov  2 22:21 zedboard.xdc
-rw------- 1 mirko users    53 Nov  2 22:21 zedboard_maps.json
```

DEMO - BondMachine creation

```
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command >
cd proj_mlinfn
ls -al
[ Output  >
total 10
drwx------ 3 mirko users    20 Nov  2 22:21 .
drwx------ 3 mirko users    17 Nov  2 22:21 ..
-rw------- 1 mirko users 44179 Nov  2 22:21 Makefile
-rw------- 1 mirko users   397 Nov  2 22:21 authorized_keys
-rw------- 1 mirko users  1962 Nov  2 22:21 banknote.json
-rw------- 1 mirko users   150 Nov  2 22:21 bmapi.json
-rw------- 1 mirko users   242 Nov  2 22:21 bmapi.mk
-rw------- 1 mirko users  1351 Nov  2 22:21 bminfo.json
-rw------- 1 mirko users   130 Nov  2 22:21 buildroot.mk
-rw------- 1 mirko users   129 Nov  2 22:21 crosscompile.mk
-rwx------ 1 mirko users  3613 Nov  2 22:21 deploy_jupyter_board.py
-rw------- 1 mirko users   495 Nov  2 22:21 local.mk
-rw------- 1 mirko users   145 Nov  2 22:21 neuralbondconfig.json
drwx------ 2 mirko users    20 Nov  2 22:21 neurons
-rw------- 1 mirko users   145 Nov  2 22:21 simbatch.mk
-rwx------ 1 mirko users  4059 Nov  2 22:21 simbatch.py
-rw------- 1 mirko users 24057 Nov  2 22:21 simbatch_input.csv
-rw------- 1 mirko users  1100 Nov  2 22:21 sumapp.go
-rw------- 1 mirko users 21319 Nov  2 22:21 zedboard.xdc
-rw------- 1 mirko users    53 Nov  2 22:21 zedboard_maps.json
[ Command > cat local.mk
```

DEMO - BondMachine creation

```
-rw------- 1 mirko users   145 Nov  2 22:21 simbatch.mk
-rwx------ 1 mirko users  4059 Nov  2 22:21 simbatch.py
-rw------- 1 mirko users 24057 Nov  2 22:21 simbatch_input.csv
-rw------- 1 mirko users  1100 Nov  2 22:21 sumapp.go
-rw------- 1 mirko users 21319 Nov  2 22:21 zedboard.xdc
-rw------- 1 mirko users    53 Nov  2 22:21 zedboard_maps.json
[ Command > cat local.mk
[ Output  >
WORKING_DIR=working_dir
CURRENT_DIR=$(shell pwd)
SOURCE_NEURALBOND=banknote.json
NEURALBOND_LIBRARY=neurons
NEURALBOND_ARGS=-config-file neuralbondconfig.json -operating-mode fragment
BMINFO=bminfo.json
BOARD=zedboard
MAPFILE=zedboard_maps.json
SHOWARGS=-dot-detail 5
SHOWRENDERER=dot
VERILOG_OPTIONS=-comment-verilog
#BASM_ARGS=-d
BENCHCORE=i0,p0o0
#HDL_REGRESSION=bondmachine.sv
#BM_REGRESSION=bondmachine.json
include bmapi.mk
include crosscompile.mk
include buildroot.mk
include simbatch.mk
```

DEMO - BondMachine creation

```
-rwx------ 1 mirko users  4059 Nov  2 22:21 simbatch.py
-rw------- 1 mirko users 24057 Nov  2 22:21 simbatch_input.csv
-rw------- 1 mirko users  1100 Nov  2 22:21 sumapp.go
-rw------- 1 mirko users 21319 Nov  2 22:21 zedboard.xdc
-rw------- 1 mirko users    53 Nov  2 22:21 zedboard_maps.json
[ Command > cat local.mk
[ Output  >
WORKING_DIR=working_dir
CURRENT_DIR=$(shell pwd)
SOURCE_NEURALBOND=banknote.json
NEURALBOND_LIBRARY=neurons
NEURALBOND_ARGS=-config-file neuralbondconfig.json -operating-mode fragment
BMINFO=bminfo.json
BOARD=zedboard
MAPFILE=zedboard_maps.json
SHOWARGS=-dot-detail 5
SHOWRENDERER=dot
VERILOG_OPTIONS=-comment-verilog
#BASM_ARGS=-d
BENCHCORE=i0,p0o0
#HDL_REGRESSION=bondmachine.sv
#BM_REGRESSION=bondmachine.json
include bmapi.mk
include crosscompile.mk
include buildroot.mk
include simbatch.mk
[ Command > ls neurons
```

DEMO - BondMachine creation

```
[ Command > cat local.mk
[ Output  >
WORKING_DIR=working_dir
CURRENT_DIR=$(shell pwd)
SOURCE_NEURALBOND=banknote.json
NEURALBOND_LIBRARY=neurons
NEURALBOND_ARGS=-config-file neuralbondconfig.json -operating-mode fragment
BMINFO=bminfo.json
BOARD=zedboard
MAPFILE=zedboard_maps.json
SHOWARGS=-dot-detail 5
SHOWRENDERER=dot
VERILOG_OPTIONS=-comment-verilog
#BASM_ARGS=-d
BENCHCORE=i0,p0o0
#HDL_REGRESSION=bondmachine.sv
#BM_REGRESSION=bondmachine.json
include bmapi.mk
include crosscompile.mk
include buildroot.mk
include simbatch.mk
[ Command > ls neurons
[ Output  >
frag-linear.basm     frag-terminal.basm   rom-linear.basm      rom-terminal.basm   terminal.nb
frag-relu.basm       frag-weight.basm     rom-relu.basm        rom-weight.basm     weight.nb
frag-softmax.basm    linear.nb            rom-softmax.basm     softmax.nb
frag-summation.basm  relu.nb              rom-summation.basm   summation.nb
```

DEMO - BondMachine creation

```
[ Output  >
WORKING_DIR=working_dir
CURRENT_DIR=$(shell pwd)
SOURCE_NEURALBOND=banknote.json
NEURALBOND_LIBRARY=neurons
NEURALBOND_ARGS=-config-file neuralbondconfig.json -operating-mode fragment
BMINFO=bminfo.json
BOARD=zedboard
MAPFILE=zedboard_maps.json
SHOWARGS=-dot-detail 5
SHOWRENDERER=dot
VERILOG_OPTIONS=-comment-verilog
#BASM_ARGS=-d
BENCHCORE=i0,p0o0
#HDL_REGRESSION=bondmachine.sv
#BM_REGRESSION=bondmachine.json
include bmapi.mk
include crosscompile.mk
include buildroot.mk
include simbatch.mk
[ Command > ls neurons
[ Output  >
frag-linear.basm     frag-terminal.basm  rom-linear.basm     rom-terminal.basm  terminal.nb
frag-relu.basm       frag-weight.basm    rom-relu.basm       rom-weight.basm    weight.nb
frag-softmax.basm    linear.nb           rom-softmax.basm    softmax.nb
frag-summation.basm  relu.nb             rom-summation.basm  summation.nb
[ Command > cat neurons/frag-softmax.basm | head -n 15
```

DEMO - BondMachine creation

```
include crosscompile.mk
include buildroot.mk
include simbatch.mk
[ Command > ls neurons
[ Output  >
frag-linear.basm     frag-terminal.basm  rom-linear.basm     rom-terminal.basm  terminal.nb
frag-relu.basm       frag-weight.basm    rom-relu.basm       rom-weight.basm    weight.nb
frag-softmax.basm    linear.nb           rom-softmax.basm    softmax.nb
frag-summation.basm  relu.nb             rom-summation.basm  summation.nb
[ Command > cat neurons/frag-softmax.basm | head -n 15
[ Output  >
%fragment softmax iomode:sync template:true resout:r9
%meta literal resin {{ with $last := adds "10" .Params.inputs }}{{range $y := intRange "10" $last }}
{{printf "r%d:" $y}}{{end}}{{end}}
        mov     r8, 0f0.0
{{ with $last := adds "10" .Params.inputs }}
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
```

DEMO - BondMachine creation

```
include buildroot.mk
include simbatch.mk
[ Command > ls neurons
[ Output  >
frag-linear.basm     frag-terminal.basm  rom-linear.basm     rom-terminal.basm  terminal.nb
frag-relu.basm       frag-weight.basm    rom-relu.basm       rom-weight.basm    weight.nb
frag-softmax.basm    linear.nb           rom-softmax.basm    softmax.nb
frag-summation.basm  relu.nb             rom-summation.basm  summation.nb
[ Command > cat neurons/frag-softmax.basm | head -n 15
[ Output  >
%fragment softmax iomode:sync template:true resout:r9
%meta literal resin {{ with $last := adds "10" .Params.inputs }}{{range $y := intRange "10" $last }}
{{printf "r%d:" $y}}{{end}}{{end}}
        mov     r8, 0f0.0
{{ with $last := adds "10" .Params.inputs }}
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
[ Command > cp /tmp/modelBM.json banknote.json
```

DEMO - BondMachine creation

```
include buildroot.mk
include simbatch.mk
[ Command > ls neurons
[ Output  >
frag-linear.basm     frag-terminal.basm  rom-linear.basm     rom-terminal.basm  terminal.nb
frag-relu.basm       frag-weight.basm    rom-relu.basm       rom-weight.basm    weight.nb
frag-softmax.basm    linear.nb           rom-softmax.basm    softmax.nb
frag-summation.basm  relu.nb             rom-summation.basm  summation.nb
[ Command > cat neurons/frag-softmax.basm | head -n 15
[ Output  >
%fragment softmax iomode:sync template:true resout:r9
%meta literal resin {{ with $last := adds "10" .Params.inputs }}{{range $y := intRange "10" $last }}
{{printf "r%d:" $y}}{{end}}{{end}}
        mov     r8, 0f0.0
{{ with $last := adds "10" .Params.inputs }}
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
[ Command > cp /tmp/modelBM.json banknote.json
```

DEMO - BondMachine creation

```
include simbatch.mk
[ Command > ls neurons
[ Output  >
frag-linear.basm     frag-terminal.basm  rom-linear.basm     rom-terminal.basm  terminal.nb
frag-relu.basm       frag-weight.basm    rom-relu.basm       rom-weight.basm    weight.nb
frag-softmax.basm    linear.nb           rom-softmax.basm    softmax.nb
frag-summation.basm  relu.nb             rom-summation.basm  summation.nb
[ Command > cat neurons/frag-softmax.basm | head -n 15
[ Output  >
%fragment softmax iomode:sync template:true resout:r9
%meta literal resin {{ with $last := adds "10" .Params.inputs }}{{range $y := intRange "10" $last }}
{{printf "r%d:" $y}}{{end}}{{end}}
        mov     r8, 0f0.0
{{ with $last := adds "10" .Params.inputs }}
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
[ Command > cp /tmp/modelBM.json banknote.json
[ Command > make bondmachine
```

DEMO - BondMachine creation

```
%meta literal resin {{ with $last := adds "10" .Params.inputs }}{{range $y := intRange "10" $last }}
{{printf "r%d:" $y}}{{end}}{{end}}
        mov     r8, 0f0.0
{{ with $last := adds "10" .Params.inputs }}
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
[ Command > cp /tmp/modelBM.json banknote.json
[ Command > make bondmachine
[Project: proj_mlinfn] - [Working directory creation begin] - [Target: working_dir]
mkdir -p working_dir
[Project: proj_mlinfn] - [Working directory creation end]

[Project: proj_mlinfn] - [BondMachine generation begin] - [Target: working_dir/bondmachine_target]
neuralbond -net-file banknote.json -neuron-lib-path neurons -save-basm working_dir/bondmachine.basm
-config-file neuralbondconfig.json -operating-mode fragment -bminfo-file bminfo.json ; basm  -bminfo
-file bminfo.json -o working_dir/bondmachine.json working_dir/bondmachine.basm neurons/*.basm
[Project: proj_mlinfn] - [BondMachine generation end]
```

DEMO - BondMachine creation

```
{{printf "r%d:" $y}}{{end}}{{end}}
        mov     r8, 0f0.0
{{ with $last := adds "10" .Params.inputs }}
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
[ Command > cp /tmp/modelBM.json banknote.json
[ Command > make bondmachine
[Project: proj_mlinfn] - [Working directory creation begin] - [Target: working_dir]
mkdir -p working_dir
[Project: proj_mlinfn] - [Working directory creation end]

[Project: proj_mlinfn] - [BondMachine generation begin] - [Target: working_dir/bondmachine_target]
neuralbond -net-file banknote.json -neuron-lib-path neurons -save-basm working_dir/bondmachine.basm
-config-file neuralbondconfig.json -operating-mode fragment -bminfo-file bminfo.json ; basm  -bminfo
-file bminfo.json -o working_dir/bondmachine.json working_dir/bondmachine.basm neurons/*.basm
[Project: proj_mlinfn] - [BondMachine generation end]

[ Command > ls working_dir
```

DEMO - BondMachine creation

```
{{ with $last := adds "10" .Params.inputs }}
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
[ Command > cp /tmp/modelBM.json banknote.json
[ Command > make bondmachine
[Project: proj_mlinfn] - [Working directory creation begin] - [Target: working_dir]
mkdir -p working_dir
[Project: proj_mlinfn] - [Working directory creation end]

[Project: proj_mlinfn] - [BondMachine generation begin] - [Target: working_dir/bondmachine_target]
neuralbond -net-file banknote.json -neuron-lib-path neurons -save-basm working_dir/bondmachine.basm
-config-file neuralbondconfig.json -operating-mode fragment -bminfo-file bminfo.json ; basm  -bminfo
-file bminfo.json -o working_dir/bondmachine.json working_dir/bondmachine.basm neurons/*.basm
[Project: proj_mlinfn] - [BondMachine generation end]

[ Command > ls working_dir
[ Output  >
bondmachine.basm  bondmachine.json  bondmachine_target
```

DEMO - BondMachine creation

```
{{range $y := intRange "10" $last}}
{{printf "mov r1,r%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
[ Command > cp /tmp/modelBM.json banknote.json
[ Command > make bondmachine
[Project: proj_mlinfn] - [Working directory creation begin] - [Target: working_dir]
mkdir -p working_dir
[Project: proj_mlinfn] - [Working directory creation end]

[Project: proj_mlinfn] - [BondMachine generation begin] - [Target: working_dir/bondmachine_target]
neuralbond -net-file banknote.json -neuron-lib-path neurons -save-basm working_dir/bondmachine.basm
-config-file neuralbondconfig.json -operating-mode fragment -bminfo-file bminfo.json ; basm  -bminfo
-file bminfo.json -o working_dir/bondmachine.json working_dir/bondmachine.basm neurons/*.basm
[Project: proj_mlinfn] - [BondMachine generation end]

[ Command > ls working_dir
[ Output  >
bondmachine.basm  bondmachine.json  bondmachine_target
[ Command > cat working_dir/bondmachine.basm
```

DEMO - BondMachine creation

```
%meta filinkatt downweightfi_3_1__4_1 fi:weightfi_3_1__4_1, type:input, index:0
%meta filinkatt downweightfi_3_1__4_1 fi:node_3_1, type:output, index:0
%meta filinkatt upweightfi_3_1__4_1 fi:node_4_1, type:input, index:0
%meta filinkatt upweightfi_3_1__4_1 fi:weightfi_3_1__4_1, type:output, index:0
%meta cpdef node_4_0 fragcollapse:node_4_0
%meta cpdef node_1_0 fragcollapse:node_1_0
%meta cpdef node_2_0 fragcollapse:node_2_0
%meta cpdef weightfi_0_3__1_0 fragcollapse:weightfi_0_3__1_0
%meta cpdef weightfi_2_0__3_0 fragcollapse:weightfi_2_0__3_0
%meta cpdef weightfi_0_0__1_0 fragcollapse:weightfi_0_0__1_0
%meta cpdef weightfi_2_1__3_0 fragcollapse:weightfi_2_1__3_0
%meta cpdef node_4_1 fragcollapse:node_4_1
%meta cpdef weightfi_2_1__3_1 fragcollapse:weightfi_2_1__3_1
%meta cpdef node_3_0 fragcollapse:node_3_0
%meta cpdef weightfi_3_0__4_0 fragcollapse:weightfi_3_0__4_0
%meta cpdef node_0_2 fragcollapse:node_0_2
%meta cpdef weightfi_0_2__1_0 fragcollapse:weightfi_0_2__1_0
%meta cpdef node_0_1 fragcollapse:node_0_1
%meta cpdef weightfi_1_0__2_0 fragcollapse:weightfi_1_0__2_0
%meta cpdef weightfi_1_0__2_1 fragcollapse:weightfi_1_0__2_1
%meta cpdef weightfi_2_0__3_1 fragcollapse:weightfi_2_0__3_1
%meta cpdef node_0_0 fragcollapse:node_0_0
%meta cpdef weightfi_0_1__1_0 fragcollapse:weightfi_0_1__1_0
%meta cpdef node_0_3 fragcollapse:node_0_3
%meta cpdef node_3_1 fragcollapse:node_3_1
%meta cpdef weightfi_3_1__4_1 fragcollapse:weightfi_3_1__4_1
%meta cpdef node_2_1 fragcollapse:node_2_1
```

DEMO - BondMachine creation

```
%meta filinkatt downweightfi_3_1__4_1 fi:node_3_1, type:output, index:0
%meta filinkatt upweightfi_3_1__4_1 fi:node_4_1, type:input, index:0
%meta filinkatt upweightfi_3_1__4_1 fi:weightfi_3_1__4_1, type:output, index:0
%meta cpdef node_4_0 fragcollapse:node_4_0
%meta cpdef node_1_0 fragcollapse:node_1_0
%meta cpdef node_2_0 fragcollapse:node_2_0
%meta cpdef weightfi_0_3__1_0 fragcollapse:weightfi_0_3__1_0
%meta cpdef weightfi_2_0__3_0 fragcollapse:weightfi_2_0__3_0
%meta cpdef weightfi_0_0__1_0 fragcollapse:weightfi_0_0__1_0
%meta cpdef weightfi_2_1__3_0 fragcollapse:weightfi_2_1__3_0
%meta cpdef node_4_1 fragcollapse:node_4_1
%meta cpdef weightfi_2_1__3_1 fragcollapse:weightfi_2_1__3_1
%meta cpdef node_3_0 fragcollapse:node_3_0
%meta cpdef weightfi_3_0__4_0 fragcollapse:weightfi_3_0__4_0
%meta cpdef node_0_2 fragcollapse:node_0_2
%meta cpdef weightfi_0_2__1_0 fragcollapse:weightfi_0_2__1_0
%meta cpdef node_0_1 fragcollapse:node_0_1
%meta cpdef weightfi_1_0__2_0 fragcollapse:weightfi_1_0__2_0
%meta cpdef weightfi_1_0__2_1 fragcollapse:weightfi_1_0__2_1
%meta cpdef weightfi_2_0__3_1 fragcollapse:weightfi_2_0__3_1
%meta cpdef node_0_0 fragcollapse:node_0_0
%meta cpdef weightfi_0_1__1_0 fragcollapse:weightfi_0_1__1_0
%meta cpdef node_0_3 fragcollapse:node_0_3
%meta cpdef node_3_1 fragcollapse:node_3_1
%meta cpdef weightfi_3_1__4_1 fragcollapse:weightfi_3_1__4_1
%meta cpdef node_2_1 fragcollapse:node_2_1
[ Command > make hdl
```

DEMO - BondMachine creation

```
%meta filinkatt downweightfi_3_1__4_1 fi:node_3_1, type:output, index:0
%meta filinkatt upweightfi_3_1__4_1 fi:node_4_1, type:input, index:0
%meta filinkatt upweightfi_3_1__4_1 fi:weightfi_3_1__4_1, type:output, index:0
%meta cpdef node_4_0 fragcollapse:node_4_0
%meta cpdef node_1_0 fragcollapse:node_1_0
%meta cpdef node_2_0 fragcollapse:node_2_0
%meta cpdef weightfi_0_3__1_0 fragcollapse:weightfi_0_3__1_0
%meta cpdef weightfi_2_0__3_0 fragcollapse:weightfi_2_0__3_0
%meta cpdef weightfi_0_0__1_0 fragcollapse:weightfi_0_0__1_0
%meta cpdef weightfi_2_1__3_0 fragcollapse:weightfi_2_1__3_0
%meta cpdef node_4_1 fragcollapse:node_4_1
%meta cpdef weightfi_2_1__3_1 fragcollapse:weightfi_2_1__3_1
%meta cpdef node_3_0 fragcollapse:node_3_0
%meta cpdef weightfi_3_0__4_0 fragcollapse:weightfi_3_0__4_0
%meta cpdef node_0_2 fragcollapse:node_0_2
%meta cpdef weightfi_0_2__1_0 fragcollapse:weightfi_0_2__1_0
%meta cpdef node_0_1 fragcollapse:node_0_1
%meta cpdef weightfi_1_0__2_0 fragcollapse:weightfi_1_0__2_0
%meta cpdef weightfi_1_0__2_1 fragcollapse:weightfi_1_0__2_1
%meta cpdef weightfi_2_0__3_1 fragcollapse:weightfi_2_0__3_1
%meta cpdef node_0_0 fragcollapse:node_0_0
%meta cpdef weightfi_0_1__1_0 fragcollapse:weightfi_0_1__1_0
%meta cpdef node_0_3 fragcollapse:node_0_3
%meta cpdef node_3_1 fragcollapse:node_3_1
%meta cpdef weightfi_3_1__4_1 fragcollapse:weightfi_3_1__4_1
%meta cpdef node_2_1 fragcollapse:node_2_1
[ Command > make hdl
```

DEMO - BondMachine creation

```
%meta filinkatt upweightfi_3_1__4_1 fi:node_4_1, type:input, index:0
%meta filinkatt upweightfi_3_1__4_1 fi:weightfi_3_1__4_1, type:output, index:0
%meta cpdef node_4_0 fragcollapse:node_4_0
%meta cpdef node_1_0 fragcollapse:node_1_0
%meta cpdef node_2_0 fragcollapse:node_2_0
%meta cpdef weightfi_0_3__1_0 fragcollapse:weightfi_0_3__1_0
%meta cpdef weightfi_2_0__3_0 fragcollapse:weightfi_2_0__3_0
%meta cpdef weightfi_0_0__1_0 fragcollapse:weightfi_0_0__1_0
%meta cpdef weightfi_2_1__3_0 fragcollapse:weightfi_2_1__3_0
%meta cpdef node_4_1 fragcollapse:node_4_1
%meta cpdef weightfi_2_1__3_1 fragcollapse:weightfi_2_1__3_1
%meta cpdef node_3_0 fragcollapse:node_3_0
%meta cpdef weightfi_3_0__4_0 fragcollapse:weightfi_3_0__4_0
%meta cpdef node_0_2 fragcollapse:node_0_2
%meta cpdef weightfi_0_2__1_0 fragcollapse:weightfi_0_2__1_0
%meta cpdef node_0_1 fragcollapse:node_0_1
%meta cpdef weightfi_1_0__2_0 fragcollapse:weightfi_1_0__2_0
%meta cpdef weightfi_1_0__2_1 fragcollapse:weightfi_1_0__2_1
%meta cpdef weightfi_2_0__3_1 fragcollapse:weightfi_2_0__3_1
%meta cpdef node_0_0 fragcollapse:node_0_0
%meta cpdef weightfi_0_1__1_0 fragcollapse:weightfi_0_1__1_0
%meta cpdef node_0_3 fragcollapse:node_0_3
%meta cpdef node_3_1 fragcollapse:node_3_1
%meta cpdef weightfi_3_1__4_1 fragcollapse:weightfi_3_1__4_1
%meta cpdef node_2_1 fragcollapse:node_2_1
[ Command > make hdl
[ Command > make show
```

DEMO - BondMachine creation

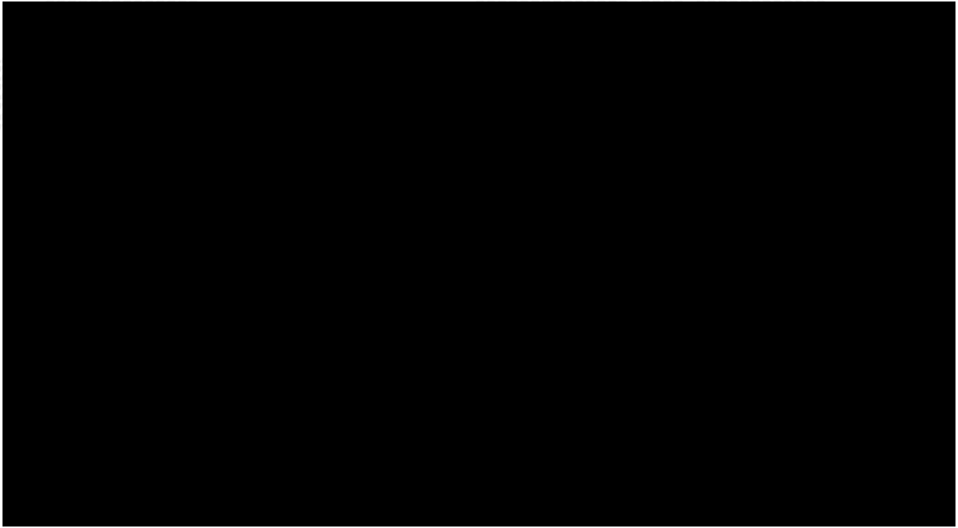
```
%meta cpdef weightfi_2_0__3_0 fragcollapse:weightfi_2_0__3_0
%meta cpdef weightfi_0_0__1_0 fragcollapse:weightfi_0_0__1_0
%meta cpdef weightfi_2_1__3_0 fragcollapse:weightfi_2_1__3_0
%meta cpdef node_4_1 fragcollapse:node_4_1
%meta cpdef weightfi_2_1__3_1 fragcollapse:weightfi_2_1__3_1
%meta cpdef node_3_0 fragcollapse:node_3_0
%meta cpdef weightfi_3_0__4_0 fragcollapse:weightfi_3_0__4_0
%meta cpdef node_0_2 fragcollapse:node_0_2
%meta cpdef weightfi_0_2__1_0 fragcollapse:weightfi_0_2__1_0
%meta cpdef node_0_1 fragcollapse:node_0_1
%meta cpdef weightfi_1_0__2_0 fragcollapse:weightfi_1_0__2_0
%meta cpdef weightfi_1_0__2_1 fragcollapse:weightfi_1_0__2_1
%meta cpdef weightfi_2_0__3_1 fragcollapse:weightfi_2_0__3_1
%meta cpdef node_0_0 fragcollapse:node_0_0
%meta cpdef weightfi_0_1__1_0 fragcollapse:weightfi_0_1__1_0
%meta cpdef node_0_3 fragcollapse:node_0_3
%meta cpdef node_3_1 fragcollapse:node_3_1
%meta cpdef weightfi_3_1__4_1 fragcollapse:weightfi_3_1__4_1
%meta cpdef node_2_1 fragcollapse:node_2_1
[ Command > make hdl
[ Command > make show
[ Output  >
[Project: proj_mlinfn] - [BondMachine diagram show begin] - [Target: show]
bondmachine -bondmachine-file working_dir/bondmachine.json -emit-dot -dot-detail 5 -bminfo-file bmin
fo.json | dot -Txlib
[Project: proj_mlinfn] - [BondMachine diagram show end]
```

BondMachine creation

The outcome of this second part of the demo are:

- bondmachine.json, a representation of the generated abstract machine
- Al the HDL files needed to build the firmware for the given board

Demo - BondMachine simulation



Demo - BondMachine simulation

```
[ Command > mkdir Example
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
[ Command > cp /tmp/sim.csv .
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
[ Command > cp /tmp/sim.csv .
```

Demo - BondMachine simulation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
[ Command > cp /tmp/sim.csv .
[ Command > make simbatch
```

Demo - BondMachine simulation

```
1.316250120615847
Running simulation with inputs: 0f-1.0601420171169955,0f0.3471542056645857,0f-0.4248275125188447,0f-
0.04608508181009227
Running simulation with inputs: 0f-0.15228760297525445,0f-0.2821256600040472,0f-0.3931947744117846,0
f0.5712245546772439
Running simulation with inputs: 0f1.052405089774165,0f0.7521166535304541,0f-0.7981025904661143,0f0.3
9395848746270123
Running simulation with inputs: 0f-0.324656804509,0f-0.15459195394199834,0f-0.7235721768066175,0f0.2
947911065500622
Running simulation with inputs: 0f1.1202121241061977,0f-0.05885513674190243,0f-0.03632330979904628,0
f1.4916348403989907
Running simulation with inputs: 0f-1.5832950470099985,0f0.18817820405272936,0f-0.14704366178162517,0
f0.2538807872456312
Running simulation with inputs: 0f-0.2817404268789742,0f-1.2433487678699013,0f0.7539298193063557,0f1
.3088205957275203
Running simulation with inputs: 0f-0.9459019049271576,0f-0.32230699215865727,0f0.3011633249327807,0f
0.8005548150124344
Running simulation with inputs: 0f0.3527853059363061,0f-0.19267696420599642,0f-0.8155007117971548,0f
1.0596199512474536
Running simulation with inputs: 0f1.0634254876935534,0f-1.0567651440981527,0f0.4696066746895383,0f0.
6412610081648472
Running simulation with inputs: 0f-0.24940573706008093,0f1.0770592106221761,0f-1.0709577426405883,0f
-0.6442337339483407
Running simulation with inputs: 0f0.8249426728456427,0f1.5484150348383172,0f-1.1686087366365605,0f-1
.5435897047849427
[Project: proj_mlinfn] - [BondMachine simbatch end]
```

Demo - BondMachine simulation

```
Running simulation with inputs: 0f-1.0601420171169955,0f0.3471542056645857,0f-0.4248275125188447,0f-
0.046085081810092 27
Running simulation with inputs: 0f-0.15228760297525445,0f-0.2821256600040472,0f-0.3931947744117846,0
f0.5712245546772439
Running simulation with inputs: 0f1.052405089774165,0f0.7521166535304541,0f-0.7981025904661143,0f0.3
9395848746270123
Running simulation with inputs: 0f-0.324656804509,0f-0.15459195394199834,0f-0.7235721768066175,0f0.2
947911065500622
Running simulation with inputs: 0f1.1202121241061977,0f-0.05885513674190243,0f-0.03632330979904628,0
f1.4916348403989907
Running simulation with inputs: 0f-1.5832950470099985,0f0.18817820405272936,0f-0.14704366178162517,0
f0.2538807872456312
Running simulation with inputs: 0f-0.2817404268789742,0f-1.2433487678699013,0f0.7539298193063557,0f1
.3088205957275203
Running simulation with inputs: 0f-0.9459019049271576,0f-0.32230699215865727,0f0.3011633249327807,0f
0.8005548150124344
Running simulation with inputs: 0f0.3527853059363061,0f-0.19267696420599642,0f-0.8155007117971548,0f
1.0596199512474536
Running simulation with inputs: 0f1.0634254876935534,0f-1.0567651440981527,0f0.4696066746895383,0f0.
6412610081648472
Running simulation with inputs: 0f-0.24940573706008093,0f1.0770592106221761,0f-1.0709577426405883,0f
-0.6442337339483407
Running simulation with inputs: 0f0.8249426728456427,0f1.5484150348383172,0f-1.1686087366365605,0f-1
.5435897047849427
[Project: proj_mlinfn] - [BondMachine simbatch end]

[ Command > cat working_dir/simbatch_output.csv
```

Demo - BondMachine simulation

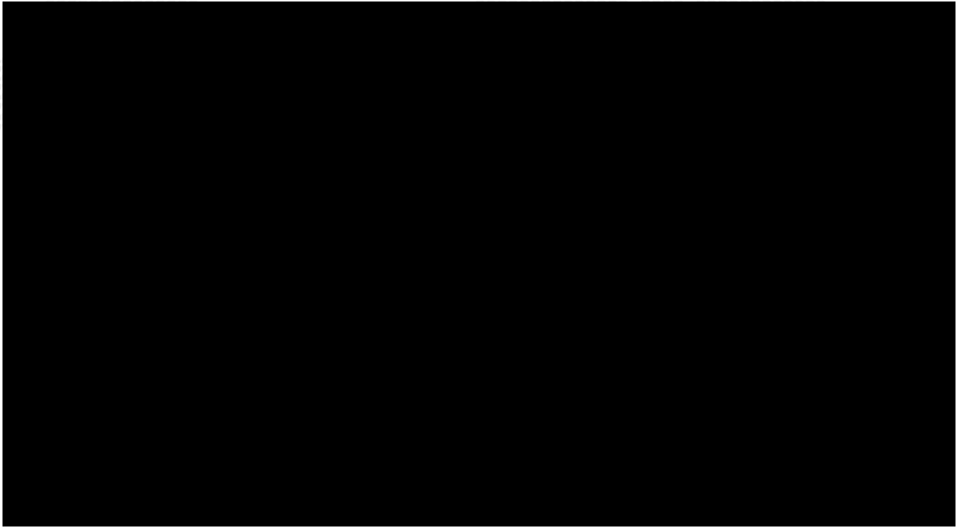
```
0.71279794,0.28720203,0
0.6313562,0.36864382,0
0.7589688,0.24103124,0
0.6479448,0.35205516,0
0.3601988,0.63980114,1
0.6425791,0.35742098,0
0.5682741,0.43172595,0
0.61973804,0.38026193,0
0.6914931,0.3085069,0
0.6783158,0.32168424,0
0.4921839,0.5078161,1
0.37793863,0.6220614,1
0.66365564,0.33634433,0
0.6749563,0.32504368,0
0.66059536,0.3394046,0
0.4266389,0.57336116,1
0.4380828,0.56191725,1
0.6834962,0.31650382,0
0.4042624,0.59573764,1
0.63697994,0.3630201,0
0.36208335,0.6379167,1
0.403224,0.59677607,1
0.40639094,0.5936091,1
0.4439535,0.5560465,1
0.593614,0.40638596,0
0.5749001,0.42509994,0
0.77141094,0.22858903,0
```

Simulation

The outcome of this third part of the demo is:

- simbatchoutput.csv, a simulated CSV files containing the output probabilities and the prediction

DEMO - BondMachine accelerator creation



DEMO - BondMachine accelerator creation

```
[ Command > mkdir Example
```

DEMO - BondMachine accelerator creation

```
[ Command > mkdir Example
[ Command > cd Example
```

DEMO - BondMachine accelerator creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

DEMO - BondMachine accelerator creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

DEMO - BondMachine accelerator creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
cat local.mk
```

DEMO - BondMachine accelerator creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
cat local.mk
[ Output  >
WORKING_DIR=working_dir
CURRENT_DIR=$(shell pwd)
SOURCE_NEURALBOND=banknote.json
NEURALBOND_LIBRARY=neurons
NEURALBOND_ARGS=-config-file neuralbondconfig.json -operating-mode fragment
BMINFO=bminfo.json
BOARD=zedboard
MAPFILE=zedboard_maps.json
SHOWARGS=-dot-detail 5
SHOWRENDERER=dot
VERILOG_OPTIONS=-comment-verilog
#BASM_ARGS=-d
BENCHCORE=i0,p0o0
#HDL_REGRESSION=bondmachine.sv
#BM_REGRESSION=bondmachine.json
include bmapi.mk
include crosscompile.mk
include buildroot.mk
include simbatch.mk
```

DEMO - BondMachine accelerator creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
cat local.mk
[ Output  >
WORKING_DIR=working_dir
CURRENT_DIR=$(shell pwd)
SOURCE_NEURALBOND=banknote.json
NEURALBOND_LIBRARY=neurons
NEURALBOND_ARGS=-config-file neuralbondconfig.json -operating-mode fragment
BMINFO=bminfo.json
BOARD=zedboard
MAPFILE=zedboard_maps.json
SHOWARGS=-dot-detail 5
SHOWRENDERER=dot
VERILOG_OPTIONS=-comment-verilog
#BASM_ARGS=-d
BENCHCORE=i0,p0o0
#HDL_REGRESSION=bondmachine.sv
#BM_REGRESSION=bondmachine.json
include bmapi.mk
include crosscompile.mk
include buildroot.mk
include simbatch.mk
[ Command > make accelerator
```

DEMO - BondMachine accelerator creation

```
INFO: [IP_Flow 19-3166] Bus Interface 'S00_AXI': References existing memory map 'S00_AXI'.
# set_property core_revision 4 [ipx::current_core]
# ipx::update_source_project_archive -component [ipx::current_core]
# ipx::create_xgui_files [ipx::current_core]
# ipx::update_checksums [ipx::current_core]
# ipx::save_core [ipx::current_core]
# ipx::move_temp_component_back -component [ipx::current_core]
# close_project -delete
# update_ip_catalog -rebuild -repo_path ${ip_directory}
INFO: [IP_Flow 19-725] Reloaded user IP repository 'ip_repo'
# close_project -delete
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:03:37 2022...
cp -a working_dir/bondmachine.sv working_dir/ip_repo/bondmachineip_1.0/hdl/bondmachine.sv
# Comments
bash -c "cd working_dir ; ./vivadoAXIcomment.sh"
# Insert the AXI code
bash -c "cd working_dir ; sed -i -e '/Add user logic here/r aux/axipatch.txt' ./ip_repo/bondmachinei
p_1.0/hdl/bondmachineip_v1_0_S00_AXI.v"
bash -c "cd working_dir ; sed -i -e '/Users to add ports here/r aux/designexternal.txt' ./ip_repo/bo
ndmachineip_1.0/hdl/bondmachineip_v1_0_S00_AXI.v"
bash -c "cd working_dir ; sed -i -e '/Users to add ports here/r aux/designexternal.txt' ./ip_repo/bo
ndmachineip_1.0/hdl/bondmachineip_v1_0.v"
bash -c "cd working_dir ; sed -i -e '/bondmachineip_v1_0_S00_AXI_inst/r aux/designexternalinst.txt'
./ip_repo/bondmachineip_1.0/hdl/bondmachineip_v1_0.v"
[Project: proj_mlinfn] - [Vivado toolchain - IP accelerator creation end]
```

DEMO - BondMachine accelerator creation

```
# set_property core_revision 4 [ipx::current_core]
# ipx::update_source_project_archive -component [ipx::current_core]
# ipx::create_xgui_files [ipx::current_core]
# ipx::update_checksums [ipx::current_core]
# ipx::save_core [ipx::current_core]
# ipx::move_temp_component_back -component [ipx::current_core]
# close_project -delete
# update_ip_catalog -rebuild -repo_path ${ip_directory}
INFO: [IP_Flow 19-725] Reloaded user IP repository 'ip_repo'
# close_project -delete
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:03:37 2022...
cp -a working_dir/bondmachine.sv working_dir/ip_repo/bondmachineip_1.0/hdl/bondmachine.sv
# Comments
bash -c "cd working_dir ; ./vivadoAXIcomment.sh"
# Insert the AXI code
bash -c "cd working_dir ; sed -i -e '/Add user logic here/r aux/axipatch.txt' ./ip_repo/bondmachinei
p_1.0/hdl/bondmachineip_v1_0_S00_AXI.v"
bash -c "cd working_dir ; sed -i -e '/Users to add ports here/r aux/designexternal.txt' ./ip_repo/bo
ndmachineip_1.0/hdl/bondmachineip_v1_0_S00_AXI.v"
bash -c "cd working_dir ; sed -i -e '/Users to add ports here/r aux/designexternal.txt' ./ip_repo/bo
ndmachineip_1.0/hdl/bondmachineip_v1_0.v"
bash -c "cd working_dir ; sed -i -e '/bondmachineip_v1_0_S00_AXI_inst/r aux/designexternalinst.txt'
./ip_repo/bondmachineip_1.0/hdl/bondmachineip_v1_0.v"
[Project: proj_mlinfn] - [Vivado toolchain - IP accelerator creation end]

[ Command > make design
```

DEMO - BondMachine accelerator creation

```
# make_wrapper -files [get_files ${project_dir}/${project_name}.srcs/sources_1/bd/bm_design/bm_desig
n.bd] -top
INFO: [BD 41-1662] The design 'bm_design.bd' is already validated. Therefore parameter propagation w
ill not be re-run.
Wrote  : </tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerator.srcs/sources_
1/bd/bm_design/bm_design.bd>
VHDL Output written to : /tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerato
r.srcs/sources_1/bd/bm_design/synth/bm_design.v
VHDL Output written to : /tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerato
r.srcs/sources_1/bd/bm_design/sim/bm_design.v
VHDL Output written to : /tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerato
r.srcs/sources_1/bd/bm_design/hdl/bm_design_wrapper.v
# update_compile_order -fileset sources_1
CRITICAL WARNING: [filemgmt 20-730] Could not find a top module in the fileset sources_1.
Resolution: With the gui up, review the source files in the Sources window. Use Add Sources to add a
ny needed sources. If the files are disabled, enable them. You can also select the file and choose S
et Used In from the pop-up menu. Review if they are being used at the proper points of the flow.
# add_files -norecurse -scan_for_includes ${project_dir}/${project_name}.srcs/sources_1/bd/bm_design
/hdl/bm_design_wrapper.v
# update_compile_order -fileset sources_1
# add_files -fileset constrs_1 -norecurse zedboard.xdc
# update_compile_order -fileset sources_1
# close_project
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:04:33 2022...
[Project: proj_mlinfn] - [Vivado toolchain - design creation end]
```

DEMO - BondMachine accelerator creation

```
n.bd] -top
INFO: [BD 41-1662] The design 'bm_design.bd' is already validated. Therefore parameter propagation w
ill not be re-run.
Wrote  : </tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerator.srcs/sources_
1/bd/bm_design/bm_design.bd>
VHDL Output written to : /tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerato
r.srcs/sources_1/bd/bm_design/synth/bm_design.v
VHDL Output written to : /tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerato
r.srcs/sources_1/bd/bm_design/sim/bm_design.v
VHDL Output written to : /tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerator/bmaccelerato
r.srcs/sources_1/bd/bm_design/hdl/bm_design_wrapper.v
# update_compile_order -fileset sources_1
CRITICAL WARNING: [filemgmt 20-730] Could not find a top module in the fileset sources_1.
Resolution: With the gui up, review the source files in the Sources window. Use Add Sources to add a
ny needed sources. If the files are disabled, enable them. You can also select the file and choose S
et Used In from the pop-up menu. Review if they are being used at the proper points of the flow.
# add_files -norecurse -scan_for_includes ${project_dir}/${project_name}.srcs/sources_1/bd/bm_design
/hdl/bm_design_wrapper.v
# update_compile_order -fileset sources_1
# add_files -fileset constrs_1 -norecurse zedboard.xdc
# update_compile_order -fileset sources_1
# close_project
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:04:33 2022...
[Project: proj_mlinfn] - [Vivado toolchain - design creation end]

[ Command > make design_synthesis
```

DEMO - BondMachine accelerator creation

```
INFO: [Project 1-571] Translating synthesized netlist
Netlist sorting complete. Time (s): cpu = 00:00:00.01 ; elapsed = 00:00:00.01 . Memory (MB): peak =
2133.133 ; gain = 0.000 ; free physical = 3826 ; free virtual = 4432
INFO: [Project 1-570] Preparing netlist for logic optimization
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 2140.0
62 ; gain = 0.000 ; free physical = 3774 ; free virtual = 4381
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

INFO: [Common 17-83] Releasing license: Synthesis
36 Infos, 26 Warnings, 0 Critical Warnings and 0 Errors encountered.
synth_design completed successfully
synth_design: Time (s): cpu = 00:00:49 ; elapsed = 00:00:51 . Memory (MB): peak = 2140.062 ; gain =
55.961 ; free physical = 3905 ; free virtual = 4512
INFO: [Common 17-1381] The checkpoint '/tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerato
r/bmaccelerator.runs/synth_1/bm_design_wrapper.dcp' has been generated.
INFO: [runtcl-4] Executing : report_utilization -file bm_design_wrapper_utilization_synth.rpt -pb bm
_design_wrapper_utilization_synth.pb
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:17:23 2022...
[Wed Nov  2 23:17:33 2022] synth_1 finished
wait_on_run: Time (s): cpu = 00:15:18 ; elapsed = 00:12:01 . Memory (MB): peak = 2258.160 ; gain = 0
.000 ; free physical = 4618 ; free virtual = 5223
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:17:34 2022...
[Project: proj_mlinfn] - [Vivado toolchain - design synthesis end]
```

DEMO - BondMachine accelerator creation

```
Netlist sorting complete. Time (s): cpu = 00:00:00.01 ; elapsed = 00:00:00.01 . Memory (MB): peak =
2133.133 ; gain = 0.000 ; free physical = 3826 ; free virtual = 4432
INFO: [Project 1-570] Preparing netlist for logic optimization
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 2140.0
62 ; gain = 0.000 ; free physical = 3774 ; free virtual = 4381
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

INFO: [Common 17-83] Releasing license: Synthesis
36 Infos, 26 Warnings, 0 Critical Warnings and 0 Errors encountered.
synth_design completed successfully
synth_design: Time (s): cpu = 00:00:49 ; elapsed = 00:00:51 . Memory (MB): peak = 2140.062 ; gain =
55.961 ; free physical = 3905 ; free virtual = 4512
INFO: [Common 17-1381] The checkpoint '/tmp/tmpof_4uxc5/Example/proj_mlinfn/working_dir/bmaccelerato
r/bmaccelerator.runs/synth_1/bm_design_wrapper.dcp' has been generated.
INFO: [runtcl-4] Executing : report_utilization -file bm_design_wrapper_utilization_synth.rpt -pb bm
_design_wrapper_utilization_synth.pb
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:17:23 2022...
[Wed Nov  2 23:17:33 2022] synth_1 finished
wait_on_run: Time (s): cpu = 00:15:18 ; elapsed = 00:12:01 . Memory (MB): peak = 2258.160 ; gain = 0
.000 ; free physical = 4618 ; free virtual = 5223
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:17:34 2022...
[Project: proj_mlinfn] - [Vivado toolchain - design synthesis end]

[ Command > make design_implementation
```

DEMO - BondMachine accelerator creation

```
report_power completed successfully
report_power: Time (s): cpu = 00:00:27 ; elapsed = 00:00:15 . Memory (MB): peak = 3082.508 ; gain =
26.992 ; free physical = 3048 ; free virtual = 3694
INFO: [runtcl-4] Executing : report_route_status -file bm_design_wrapper_route_status.rpt -pb bm_des
ign_wrapper_route_status.pb
INFO: [runtcl-4] Executing : report_timing_summary -max_paths 10 -file bm_design_wrapper_timing_summ
ary_routed.rpt -pb bm_design_wrapper_timing_summary_routed.pb -rpx bm_design_wrapper_timing_summary_
routed.rpx -warn_on_violation
INFO: [Timing 38-91] UpdateTimingParams: Speed grade: -1, Delay Type: min_max.
INFO: [Timing 38-191] Multithreading enabled for timing update using a maximum of 4 CPUs
INFO: [runtcl-4] Executing : report_incremental_reuse -file bm_design_wrapper_incremental_reuse_rout
ed.rpt
INFO: [Vivado_Tcl 4-1062] Incremental flow is disabled. No incremental reuse Info to report.
INFO: [runtcl-4] Executing : report_clock_utilization -file bm_design_wrapper_clock_utilization_rout
ed.rpt
INFO: [runtcl-4] Executing : report_bus_skew -warn_on_violation -file bm_design_wrapper_bus_skew_rou
ted.rpt -pb bm_design_wrapper_bus_skew_routed.pb -rpx bm_design_wrapper_bus_skew_routed.rpx
INFO: [Timing 38-91] UpdateTimingParams: Speed grade: -1, Delay Type: min_max.
INFO: [Timing 38-191] Multithreading enabled for timing update using a maximum of 4 CPUs
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:23:42 2022...
[Wed Nov  2 23:23:58 2022] impl_1 finished
wait_on_run: Time (s): cpu = 00:00:02 ; elapsed = 00:05:46 . Memory (MB): peak = 2202.250 ; gain = 0
.000 ; free physical = 4625 ; free virtual = 5273
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:23:58 2022...
[Project: proj_mlinfn] - [Vivado toolchain - design implementation end]
```

DEMO - BondMachine accelerator creation

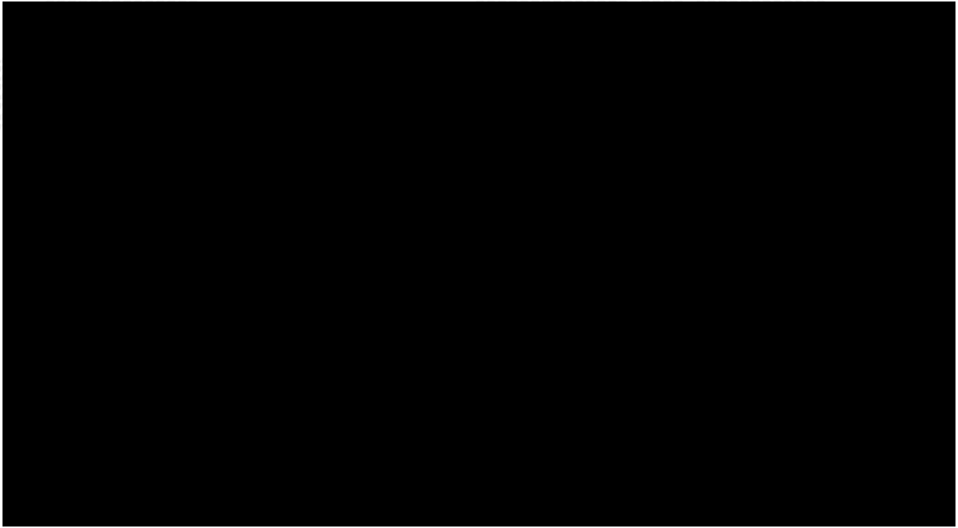
```
report_power: Time (s): cpu = 00:00:27 ; elapsed = 00:00:15 . Memory (MB): peak = 3082.508 ; gain =
26.992 ; free physical = 3048 ; free virtual = 3694
INFO: [runtcl-4] Executing : report_route_status -file bm_design_wrapper_route_status.rpt -pb bm_des
ign_wrapper_route_status.pb
INFO: [runtcl-4] Executing : report_timing_summary -max_paths 10 -file bm_design_wrapper_timing_summ
ary_routed.rpt -pb bm_design_wrapper_timing_summary_routed.pb -rpx bm_design_wrapper_timing_summary_
routed.rpx -warn_on_violation
INFO: [Timing 38-91] UpdateTimingParams: Speed grade: -1, Delay Type: min_max.
INFO: [Timing 38-191] Multithreading enabled for timing update using a maximum of 4 CPUs
INFO: [runtcl-4] Executing : report_incremental_reuse -file bm_design_wrapper_incremental_reuse_rout
ed.rpt
INFO: [Vivado_Tcl 4-1062] Incremental flow is disabled. No incremental reuse Info to report.
INFO: [runtcl-4] Executing : report_clock_utilization -file bm_design_wrapper_clock_utilization_rout
ed.rpt
INFO: [runtcl-4] Executing : report_bus_skew -warn_on_violation -file bm_design_wrapper_bus_skew_rou
ted.rpt -pb bm_design_wrapper_bus_skew_routed.pb -rpx bm_design_wrapper_bus_skew_routed.rpx
INFO: [Timing 38-91] UpdateTimingParams: Speed grade: -1, Delay Type: min_max.
INFO: [Timing 38-191] Multithreading enabled for timing update using a maximum of 4 CPUs
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:23:42 2022...
[Wed Nov  2 23:23:58 2022] impl_1 finished
wait_on_run: Time (s): cpu = 00:00:02 ; elapsed = 00:05:46 . Memory (MB): peak = 2202.250 ; gain = 0
.000 ; free physical = 4625 ; free virtual = 5273
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:23:58 2022...
[Project: proj_mlinfn] - [Vivado toolchain - design implementation end]

[ Command > make design_bitstream
```

DEMO - BondMachine accelerator creation

```
the MREG and PREG registers to be used.  If the DSP48 was instantiated in the design, it is suggeste
d to set both the MREG and PREG attributes to 1 when performing multiply functions.
INFO: [Vivado 12-3199] DRC finished with 0 Errors, 84 Warnings
INFO: [Vivado 12-3200] Please refer to the DRC report (report_drc) for more information.
INFO: [Designutils 20-2272] Running write_bitstream with 4 threads.
Loading data files...
Loading site data...
Loading route data...
Processing options...
Creating bitmap...
Creating bitstream...
Writing bitstream ./bm_design_wrapper.bit...
Writing bitstream ./bm_design_wrapper.bin...
INFO: [Vivado 12-1842] Bitgen Completed Successfully.
INFO: [Common 17-83] Releasing license: Implementation
22 Infos, 84 Warnings, 0 Critical Warnings and 0 Errors encountered.
write_bitstream completed successfully
write_bitstream: Time (s): cpu = 00:00:58 ; elapsed = 00:00:44 . Memory (MB): peak = 2909.914 ; gain
 = 498.211 ; free physical = 3484 ; free virtual = 4141
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:26:13 2022...
[Wed Nov  2 23:26:14 2022] impl_1 finished
wait_on_run: Time (s): cpu = 00:01:47 ; elapsed = 00:01:38 . Memory (MB): peak = 2186.242 ; gain = 0
.000 ; free physical = 4694 ; free virtual = 5344
# exit
INFO: [Common 17-206] Exiting Vivado at Wed Nov  2 23:26:14 2022...
[Project: proj_mlinfn] - [Vivado toolchain - design bitstream end]
```

Accelerator creation

DEMO - Standalone BondMachine creation



DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
[ Command > cat zedboard_maps.json
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
[ Command > cat zedboard_maps.json
[ Output  >
{
"Assoc" : {
        "clk" : "clk",
        "reset" : "btnC"
        }
}
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
[ Command > cat zedboard_maps.json
[ Output  >
{
"Assoc" : {
        "clk" : "clk",
        "reset" : "btnC"
        }
}
[ Command >
make project
make synthesis
make implementation
make bitstream
```

DEMO - Standalone BondMachine creation

```
[ Command > mkdir Example
[ Command > cd Example
[ Command > bmhelper create --project_name mlinfn --board zedboard --project_type neural_network --n
_inputs 4 --n_outputs 3 --source_neuralbond banknote.json
[ Command > cd proj_mlinfn
[ Command > cat zedboard_maps.json
[ Output  >
{
"Assoc" : {
        "clk" : "clk",
        "reset" : "btnC"
        }
}
[ Command >
make project
make synthesis
make implementation
make bitstream
```

Notebook on the board - predictions and correctness

```
In [138]: from pynq import Overlay
          from pynq import MMIO
          import os
          import numpy as np
          import struct
          import time

In [149]: # SETTINGS
          project_name  = "proj_ebaz4205_neuralbond_expanded"
          firmware_name = project_name+".bit"
          n_input       = 4
          n_output      = 2
          benchcore     = True

In [153]: # LOAD OVERLAY
          overlay = Overlay(os.getcwd()+"/"+firmware_name)

In [154]: # GET MEMORY ADDRESS OF IP
          bm_starting_address = (overlay.ip_dict["bondmachineip_0"]["phys_addr"])
          print(" Starting memory address of Bondmachine IP is (in dec): ", bm_starting_address)
          print(" Starting memory address of Bondmachine IP is (in hex): ", hex(bm_starting_address))

           Starting memory address of Bondmachine IP is (in dec):  1136656384
           Starting memory address of Bondmachine IP is (in hex):  0x43c00000

In [155]: # GET THE OBJECT NECESSARY TO INTERACT WITH ML IP
          spi0 = MMIO(bm_starting_address, 128)

In [156]: # LOAD BANKNOTE TESTSET
          X_test = np.load('banknote-authentication_X_test.npy')
          Y_test = np.load('banknote-authentication_y_test.npy')
          # get the first 32 samples
          X_test = X_test[0:10]
          y_test = Y_test[0:10]
          print(" Example of first two input:  ", X_test[:1])
          print(" Example of first two output: ", Y_test[:1])

           Example of first two input:   [[ 0.39886742  0.76609776 -0.39003127 -0.58781728]]
           Example of first two output:  [[ 1.  0.]]

In [157]: # IN THIS CASE I WANT TO SEND ONLY THE FIRST X INPUT SAMPLE
          idx = 0
          results_to_dump = []
          for xSample in X_test:
              offset = 0
              for feature in list(xSample): #
                  binToSend = get_binary_from_float(feature)
                  decToSend = int(binToSend, 2)
                  spi0.write_mm(offset, decToSend) # WRITE THE FEATURE TO THE CORRESPONDING INPUT
                  offset = offset + 4 # 4 BYTE = 32 BIT
              time.sleep(0.5)
              out = np.asarray(read_output())
              #print(" #",idx," -> classification: ", np.argmax(out[0:2]))
              classification = np.argmax(out[0:2])
              if (benchcore == True):
                  results_to_dump.append([out[0], out[1], classification, out[2]])
              else:
                  results_to_dump.append([out[0], out[1], classification])
              idx = idx + 1
              #break

          print(results_to_dump)

          [[0.6895769834518432б, 0.3104230463504791З, 0, 10402.0], [0.57489913702011108, 0.42510083317756653, 0, 10522.0], [0.
          60581505298614502, 0.39418494701385498, 0, 8728.0], [0.78091907501220703, 0.21908088028430939, 0, 5440.0], [0.692104
          93564605713, 0.30789506435394287, 0, 14502.0], [0.63284581899642944, 0.36715418100357056, 0, 12267.0], [0.6657641530
          0369263, 0.33423587679862976, 0, 6191.0], [0.69290423393249512, 0.30709570646286011, 0, 10592.0], [0.637744188308715
          82, 0.36225578188896179, 0, 11427.0], [0.67555481195449829, 0.32444518804550171, 0, 15070.0]]

In [115]: import csv
          fields = ['probability_0', 'probability_1', 'classification', 'clock_cycles']

          with open(project_name+".csv", 'w') as f:
              write = csv.writer(f)
              write.writerow(fields)
              write.writerows(results_to_dump)
```

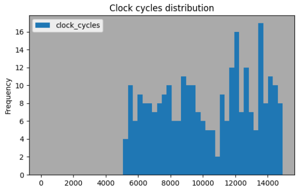
- Thanks to PYNQ we can easily load the bitstream and program the FPGA in real time.
- With their APIs we interact with the memory addresses of the BM IP to send data into the inputs and read the outputs (not using BM kernel module)
- Dump output results for future analysis

Open the notebook

Inference evaluation

Evaluation metrics used:

- **Inference speed**: time taken to predict a sample i.e. time between the arrival of the input and the change of the output measured with the **benchcore**;
- **Resource usage**: luts and registers in use;
- **Accuracy**: as the average percentage of error on probabilities.



- σ: 2875.94
- Mean: 10268.45
- Latency: 102.68 µs

Resource usage

resource	value	occupancy
regs	15122	28.42%
luts	11192	10.51%

Analysis notebook

Another notebook is used to compare runs from different accelerators.

Software

prob0	prob1	class
0.6895	0.3104	0
0.5748	0.4251	0
0.4009	0.5990	1

BondMachine

prob0	prob1	class
0.6895	0.3104	0
0.5748	0.4251	0
0.4009	0.5990	1

The output of the bm corresponds to the software output

Open the notebook

Optimizations

A first example of optimization

Remember the **softmax function**?

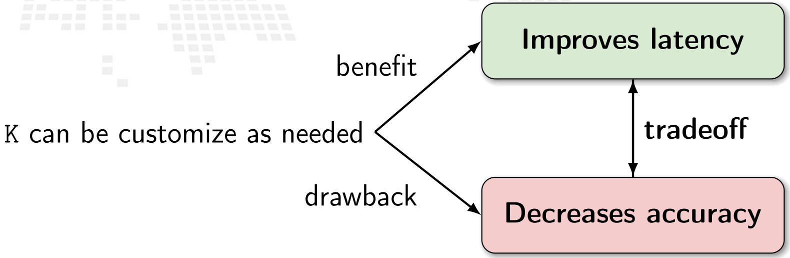
$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

$$e^x = \sum_{l=0}^{K} \frac{x^l}{l!}$$

```
%section softmax .romtext iomode:sync
        entry _start     ; Entry point
_start:
 mov r8, 0f0.0
{{range $y := intRange "0" .Params.inputs}}
{{printf "i2r r1,i%d\n" $y}}
        mov     r0, 0f1.0
        mov     r2, 0f1.0
        mov     r3, 0f1.0
        mov     r4, 0f1.0
        mov     r5, 0f1.0
        mov     r7, {{$.Params.expprec}}
loop{{printf "%d" $y}}:
        multf   r2, r1
        multf   r3, r4
        addf    r4, r5
        mov     r6, r2
        divf    r6, r3

        addf    r0, r6

        dec     r7
        jz      r7,exit{{printf "%d" $y}}
        j       loop{{printf "%d" $y}}
exit{{printf "%d" $y}}:
{{$z := atoi $.Params.pos}}
{{if eq $y $z}}
 mov r9, r0
%endsection
```

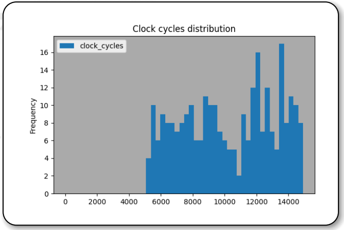
A first example of optimization

$$e^x = \sum_{l=0}^{K} \frac{x^l}{l!}$$



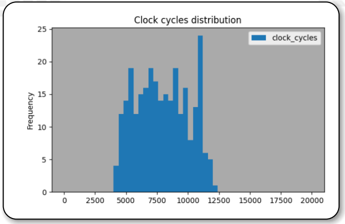
Results of optimization

Changing number of K of the exponential factors in the softmax function...



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 μs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

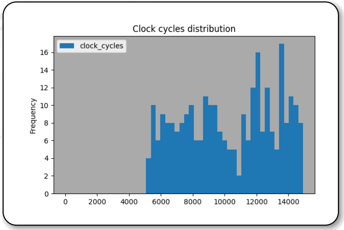


- K: 16
- σ: 2106.32
- Mean: 7946.16
- Latency: 79 μs
- Prediction: 100%

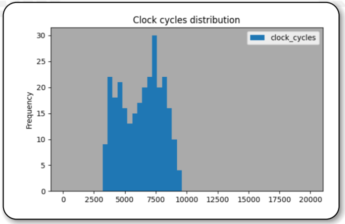
	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

Results of optimization

Changing number of K of the exponential factors in the softmax function...



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 µs
- Prediction: 100%



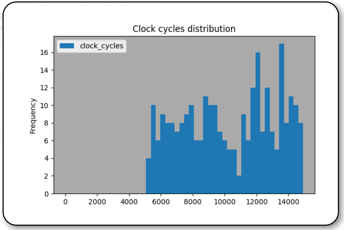
- K: 13
- σ: 1669.88
- Mean: 6312.26
- Latency: 63 µs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

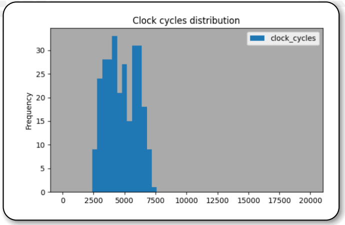
Results of optimization

Changing number of K of the exponential factors in the softmax function...



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 µs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

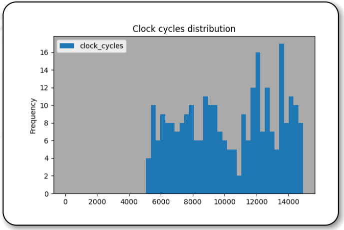


- K: 10
- σ: 1232.47
- Mean: 4766.75
- Latency: 47 µs
- Prediction: 100%

	mean	σ
prob0	1.6162E-07	1.1013E-07
prob1	1.6525E-07	1.1831E-07

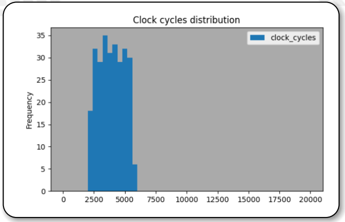
Results of optimization

Changing number of K of the exponential factors in the softmax function...



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 µs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

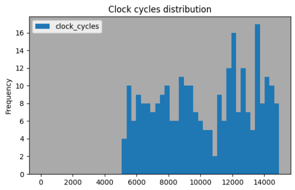


- K: 8
- σ: 1015.50
- Mean: 3913.66
- Latency: 39 µs
- Prediction: 100%

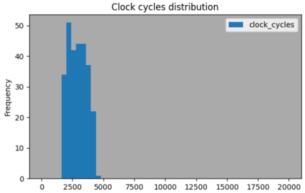
	mean	σ
prob0	6.5562E-05	1.7607E-05
prob1	6.6098E-05	1.7609E-05

Results of optimization

Changing number of K of the exponential factors in the softmax function...



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 µs
- Prediction: 100%



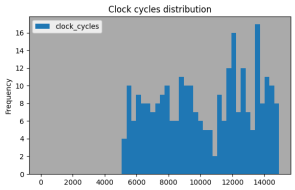
- K: 5
- σ: 740
- Mean: 2911
- Latency: 29 µs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

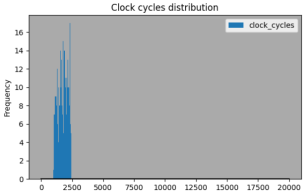
	mean	σ
prob0	3.1070E-05	7.5290E-05
prob1	3.1070E-05	7.5290E-05

Results of optimization

Changing number of K of the exponential factors in the softmax function…



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 μs
- Prediction: 100%



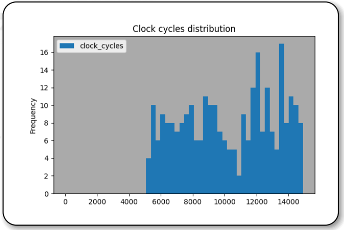
- K: 3
- σ: 394.10
- Mean: 1750.93
- Latency: 17 μs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

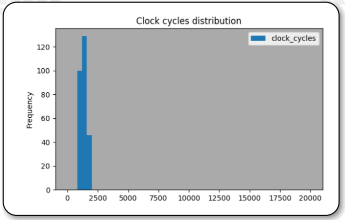
	mean	σ
prob0	0.0053	0.0090
prob1	0.0053	0.0090

Results of optimization

Changing number of K of the exponential factors in the softmax function...



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 µs
- Prediction: 100%



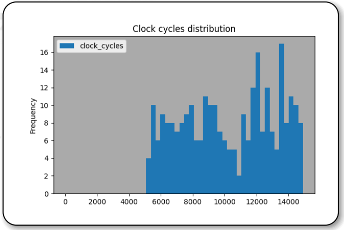
- K: 2
- σ: 268.69
- Mean: 1311.11
- Latency: 13.11 µs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

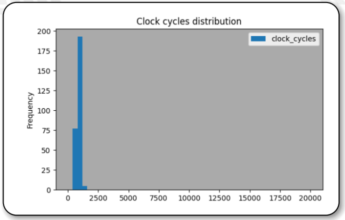
	mean	σ
prob0	0.0193	0.0232
prob1	0.0193	0.0232

Results of optimization

Changing number of K of the exponential factors in the softmax function...



- K: 20
- σ: 2875.94
- Mean: 10268.45
- Latency: 102 µs
- Prediction: 100%

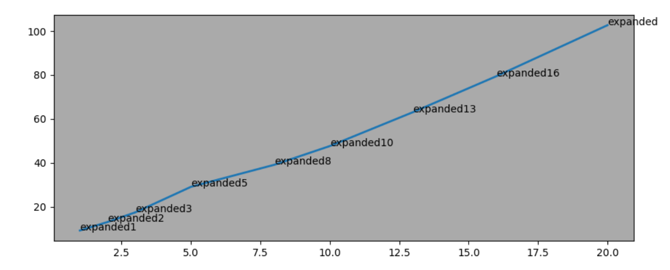


- K: 1
- σ: 173.25
- Mean: 923.71
- Latency: 9.23 µs
- Prediction: 100%

	mean	σ
prob0	1.6470E-07	1.2332E-07
prob1	1.6623E-07	1.2142E-07

	mean	σ
prob0	0.0990	0.1641
prob1	0.0990	0.1641

Results of optimization

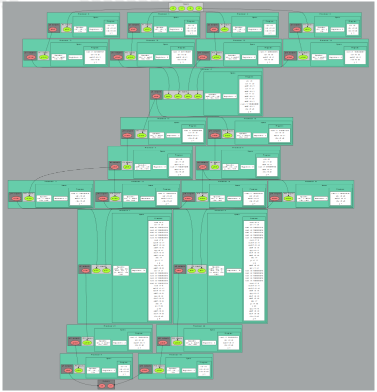


K	**Inference time**
1	9.23 µs
2	13.11 µs
3	17.50 µs
5	29.11 µs
8	39.13 µs
10	47.66 µs
13	63.12 µs
16	79.46 µs
20	102.68 µs

Reduced inference times by a factor of 10 ... only by decreasing the number of iterations.

Fragments composition

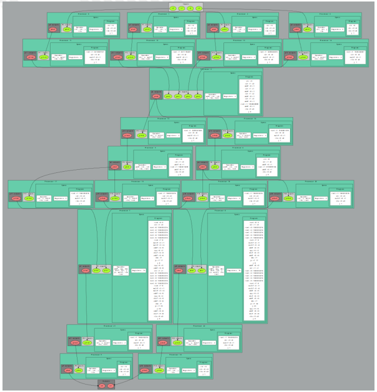
- The tools (neuralbond+basm) create a graph of relations among fragments of assembly
- Not necessarily a fragment has to be mapped to a single CP
- They can arbitrarily be rearranged into CPs
- The resulting firmwares are identical in term of the computing outcome, but differs in occupancy and latency.



Let see it live

Fragments composition

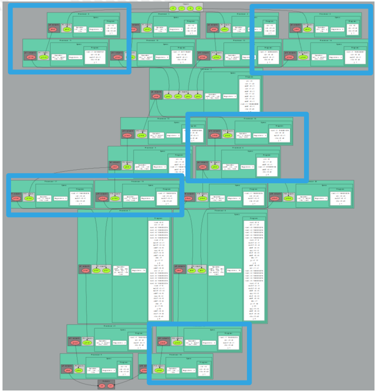
- The tools (neuralbond+basm) create a graph of relations among fragments of assembly
- Not necessarily a fragment has to be mapped to a single CP
- They can arbitrarily be rearranged into CPs
- The resulting firmwares are identical in term of the computing outcome, but differs in occupancy and latency.



Let see it live

Fragments composition

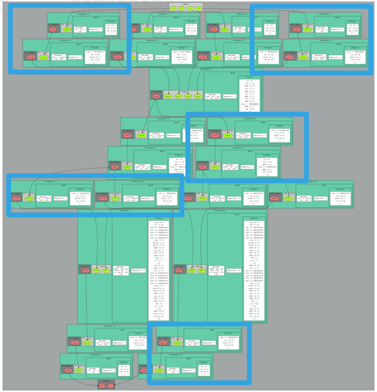
- The tools (neuralbond+basm) create a graph of relations among fragments of assembly
- Not necessarily a fragment has to be mapped to a single CP
- They can arbitrarily be rearranged into CPs
- The resulting firmwares are identical in term of the computing outcome, but differs in occupancy and latency.



Let see it live

Fragments composition

- The tools (neuralbond+basm) create a graph of relations among fragments of assembly
- Not necessarily a fragment has to be mapped to a single CP
- They can arbitrarily be rearranged into CPs
- The resulting firmwares are identical in term of the computing outcome, but differs in occupancy and latency.



Let see it live

Several ways for customization and optimization

The great control over of the architectures generated by the BondMachine gives several possible optimizations.

- Mixing hardware and software optimizations
- CP Pruning and/or collapsing
- Fabric independent
- HW instructions swapping
- Fine control over occupancy vs latency
- Fragment composition
- HW/SW Templates
- Software based functions

Conclusions and Future directions

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.

Ongoing

The project

- Move all the code to github
- Documentation
- First DAQ use case
- Complete the inclusion of Intel and Lattice FPGAs
- ML inference in a cloud workflow

Ongoing

Accelerators

- Different data types and operations, especially low and trans-precision
- Different boards support, especially data center accelerator
- Compare with GPUs
- Include some real power consumption measures

Ongoing

Machine Learning

With ML we are still at the beginning ...

- **Quantization**
- **More datasets**: test on other datasets with more features and multiclass classification
- **Neurons**: increase the library of neurons to support other activation functions
- **Evaluate results**: compare the results obtained with other technologies (CPU and GPU) in terms of inference speed and energy efficiency

Future work

- Include new processor shared objects and currently unsupported opcodes
- Extend the compiler to include more data structures
- Assembler improvements, fragments optimization and others
- Improve the networking including new kind of interconnection firmware

What would an OS for BondMachines look like ?

Future work

- Include new processor shared objects and currently unsupported opcodes
- Extend the compiler to include more data structures
- Assembler improvements, fragments optimization and others
- Improve the networking including new kind of interconnection firmware

What would an OS for BondMachines look like ?

Future work

- Include new processor shared objects and currently unsupported opcodes
- Extend the compiler to include more data structures
- Assembler improvements, fragments optimization and others
- Improve the networking including new kind of interconnection firmware

What would an OS for BondMachines look like ?

Future work

- Include new processor shared objects and currently unsupported opcodes
- Extend the compiler to include more data structures
- Assembler improvements, fragments optimization and others
- Improve the networking including new kind of interconnection firmware

What would an OS for BondMachines look like ?

Future work

- Include new processor shared objects and currently unsupported opcodes
- Extend the compiler to include more data structures
- Assembler improvements, fragments optimization and others
- Improve the networking including new kind of interconnection firmware

What would an OS for BondMachines look like ?

website: http://bondmachine.fisica.unipg.it
code: https://github.com/BondMachineHQ
parallel computing paper: link
contact email: mirko.mariotti@unipg.it