Compiler and System Techniques for SoC distributed accelerators

J. Cambonie, S. Guérin, R. Keryell, L. Lagadec
B.Pottier, O. Sentieys, B. Weber, S. Yazdani

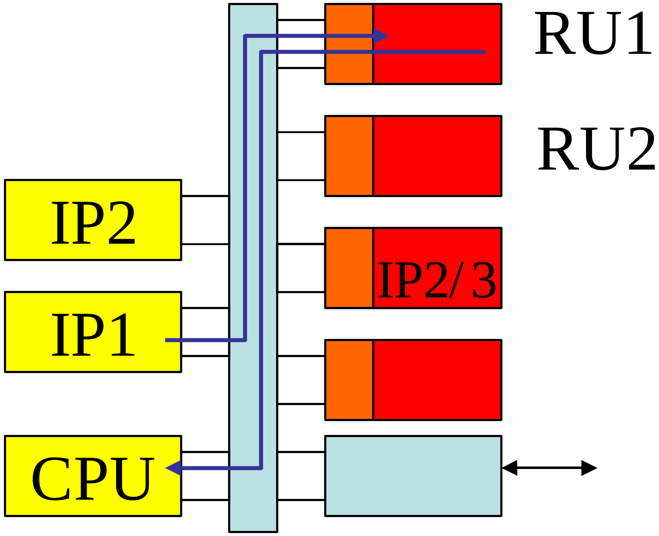
Speaker : Bernard Pottier

Framework

- An organization of reconfigurable units (RU) in a System on Chips is
  - Modular and Scalable
  - Flexible, and easy to compose

We are studying:

- Balanced coarse/fine grain RU architectures
- Local 'low level' tools/compilers
- System behaviour
- HL compiler support (GP intensive computations ?)



(ref. SCORE project, UCB)

Author affiliations

Architecture

Low level tools

- STMicroelectronics CR&D Berkeley and AST Grenoble
  - J.Cambonie (+ V.George)
- UBO
  - L. Lagadec
  - B. Pottier
  - S.Yazdani

Architecture synthesis

- ENST-Bretagne
  - S. Guérin
  - R. Keryell
  - B. Weber
- IRISA/R2D2
  - O.Sentieys

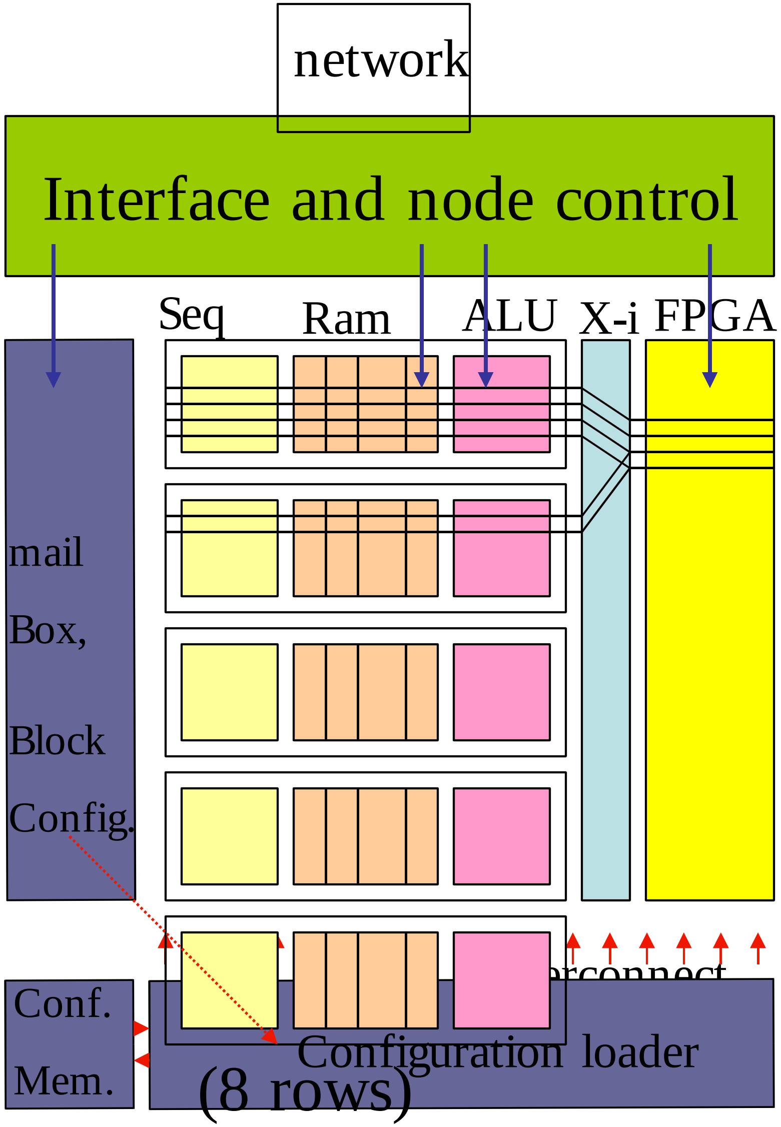
Summary

1. Architecture
2. System interface
3. Physical tools
4. Compiler interface

I. Architecture

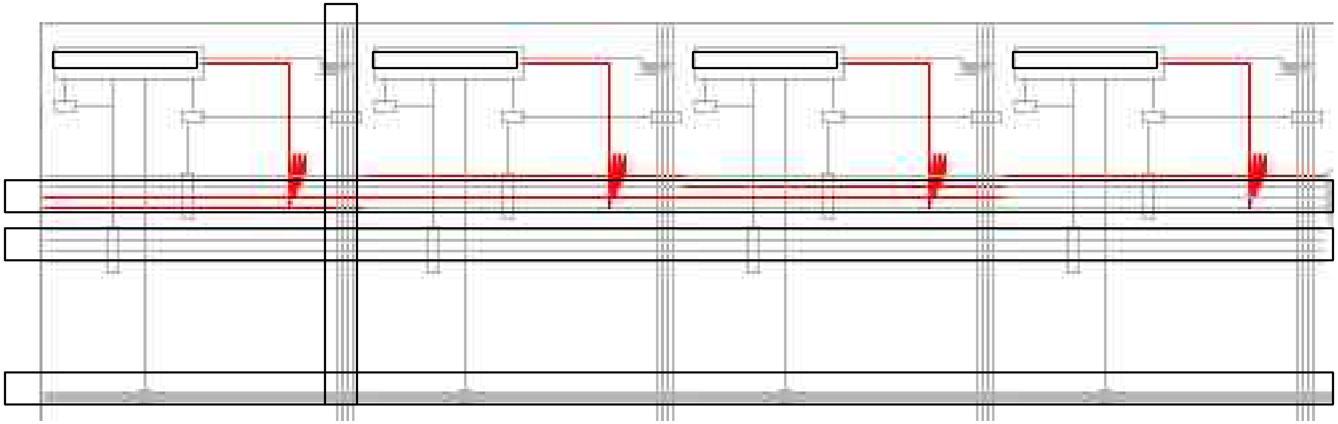
Overview of a Mixed Grain Architecture

- Tile organization
  - Ram banks
  - Address sequencers
  - Arithmetic array
- Fine grain eFPGA
- Configuration support
- Interface and steering logic



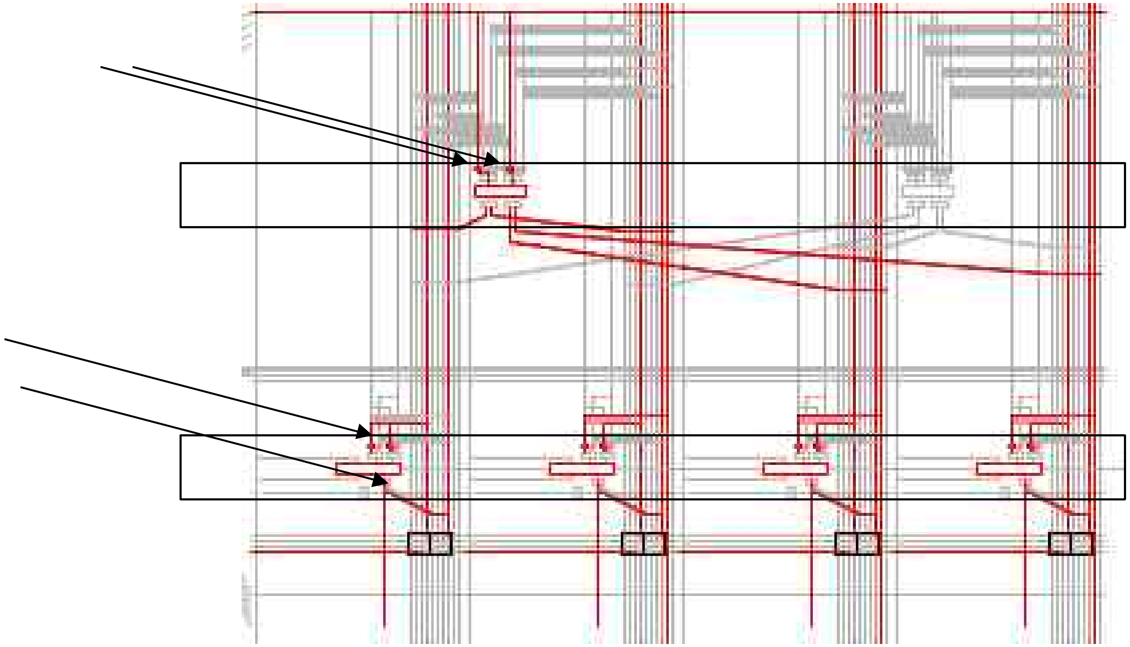
Architecture details:RAM

- Ram per tile
  - 4 x 256 bytes dual ported to CPU address space
  - Configurable connection into the tiles:
    - 2 horizontal and vertical data bus
    - 1 address bus
    - 1 control bus
  - Direct external R/W access from network transactions



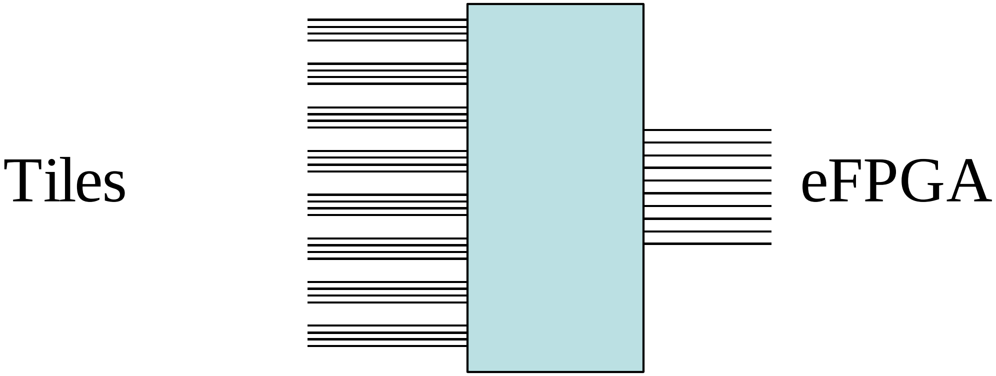
Architecture details: arithmetic

- Arithmetic / tile
  - 4 x 8 Bits cascadable ALU slices
  - 2 x 16 Bits multipliers
  - registered



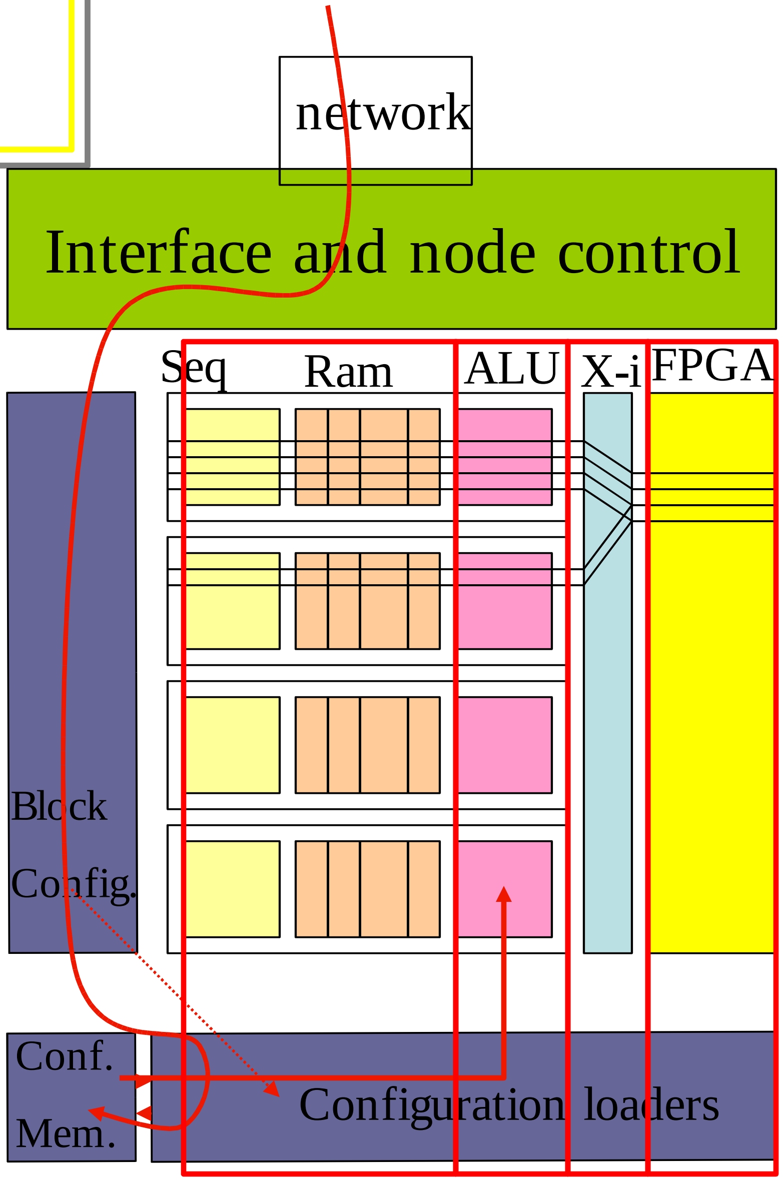
Architecture details:eFPGA

- Embedded FPGA (M2000)
  - Equivalent of 15 K Xilinx gates per unit, x 2
- Reconfigurable interconnect
  - Tile address, control and data bus, one side (#120x8)
  - eFPGA 350 signals, other side



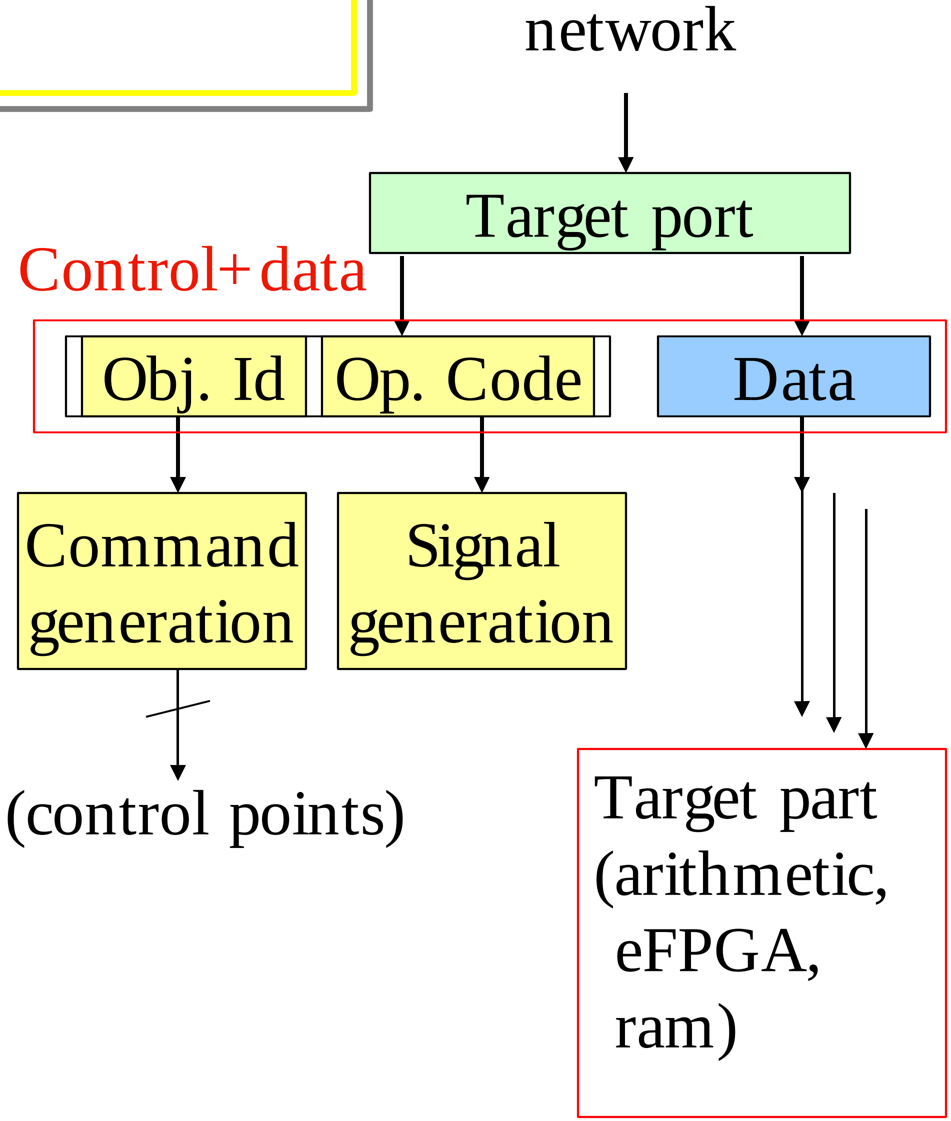
Architecture details: Configuration

- Configuration memory
  - 1Mbits Dram
- 2 stage configuration using a configuration manager
  1. Write specified adresses into the configuration memory
  2. Load configuration bit ranges to RAM, arithmetic or eFPGA … or save configurations



Mailbox support (input)

- Target port
  - Demultiplex to internal devices
  - Propagate data to devices
  - Generate signals from configurable tables

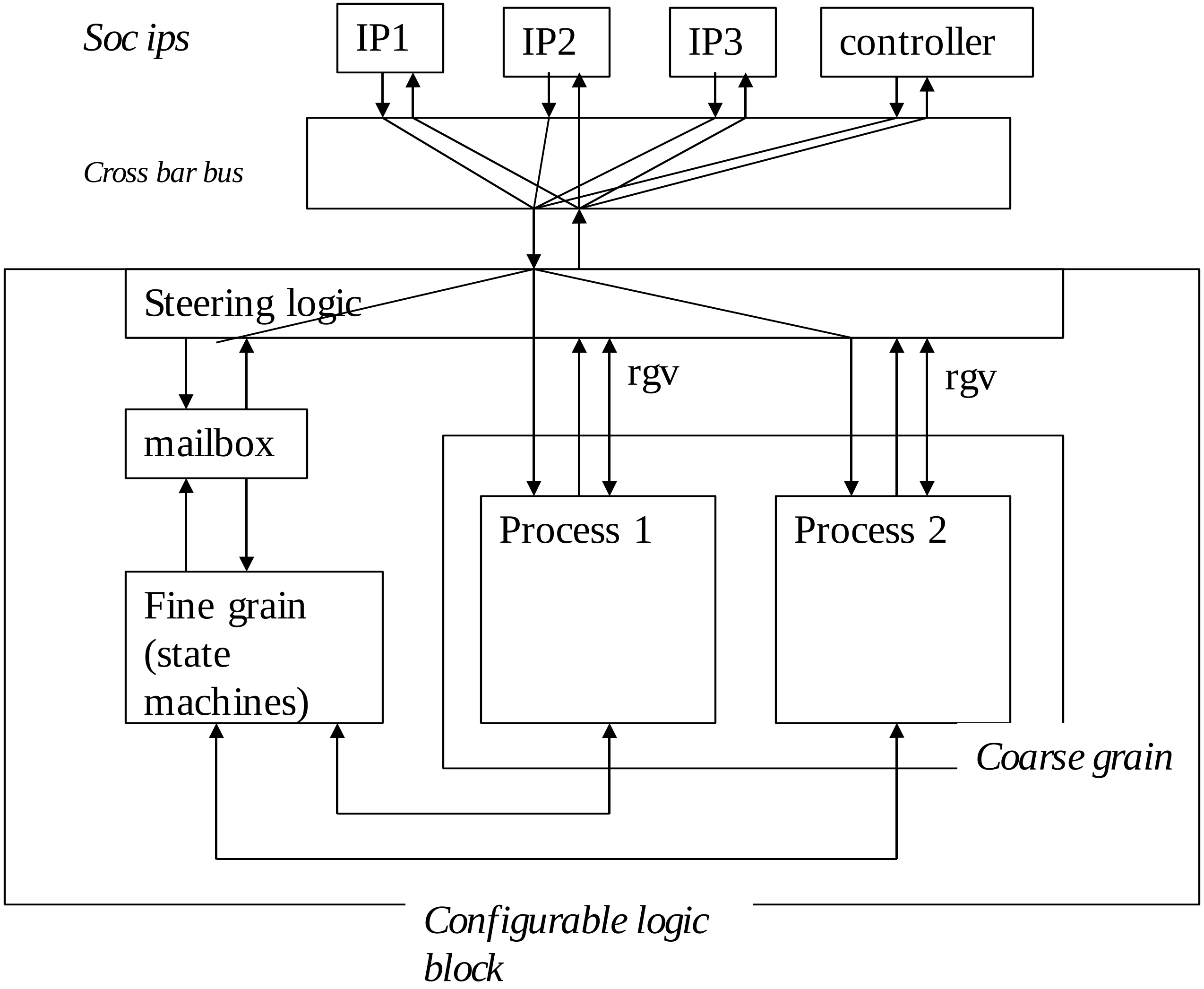


Mailbox support (output)

- Initiator port, inversely:
  - Monitor signal transitions
  - Multiplex from internal devices
  - Generate network messages from configurable tables

II. System interface

System Integration



Receiving messages

1. One word at a time
   - Name a target, or group of targets using object-ids
   - Send one data word to RAM, eFPGA, arithmetic bank
2. Packet based
   - A start/stop mark enable local automata to process messages of unknown size

Sending messages

- Transform low level event into high level messages
  - Monitor internal event from fine grain control automata
  - Generate messages automatically based on tables
  - Act as a hardware to software interface

Process Organization

At least two options:

- Direct mapping of a computation graph flow,
- Overlapped input/output and computations, with possibility of on-the-fly reconfiguration

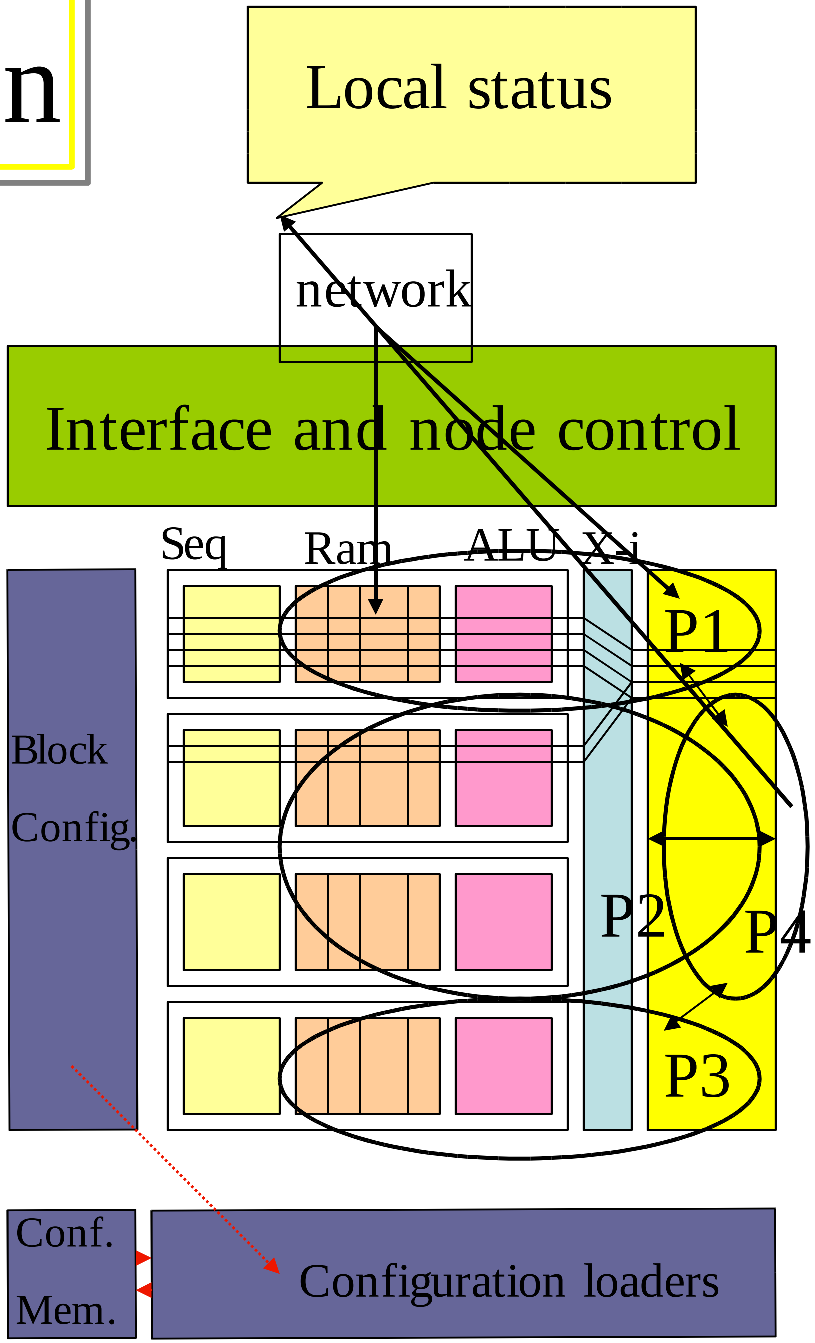
Distributed or centralized application control

P1: input DMA

P2: processing

P3: output DMA

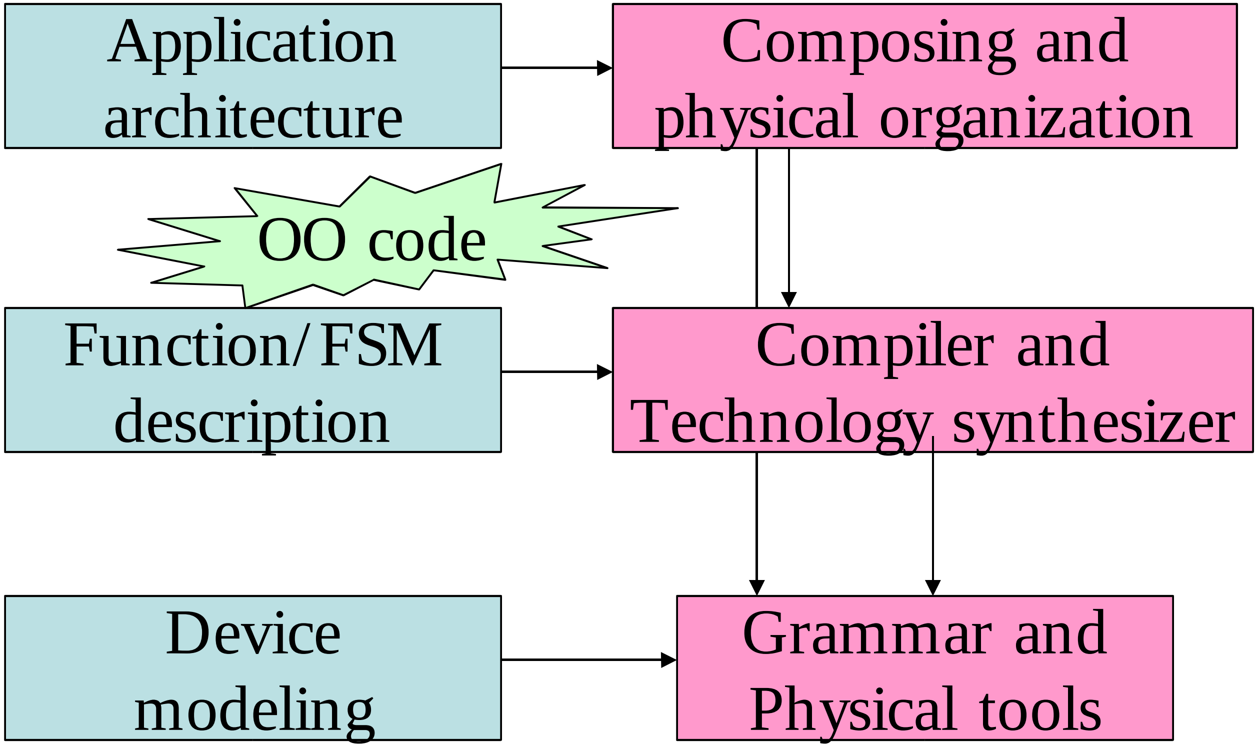
P4: pipeline task control+signaling



III. Physical mapping (process based)

(for LUT based synthesis and physical tools,
refer to our previous Samos2/3 papers)

Fine grain tools



Application architecture

Composing and physical organization

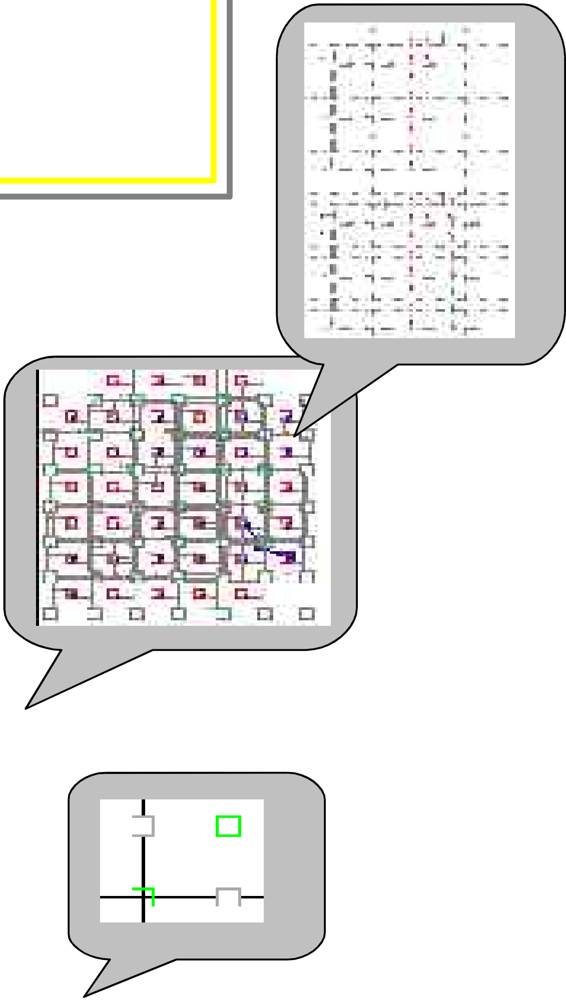
OO code

Function/FSM description

Compiler and Technology synthesizer

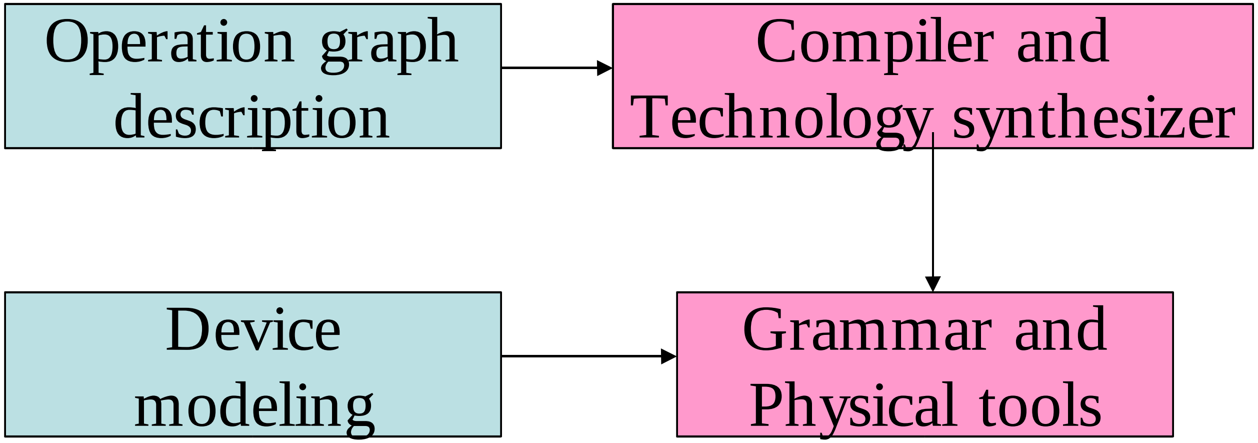
Device modeling

Grammar and Physical tools



Medium grain tools

- Homogeneous with fine grain tools
  - Architecture description
  - Place and route approach (channels and signals)
- Connectivity with eFPGA using X interface

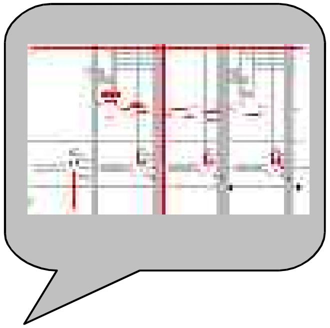


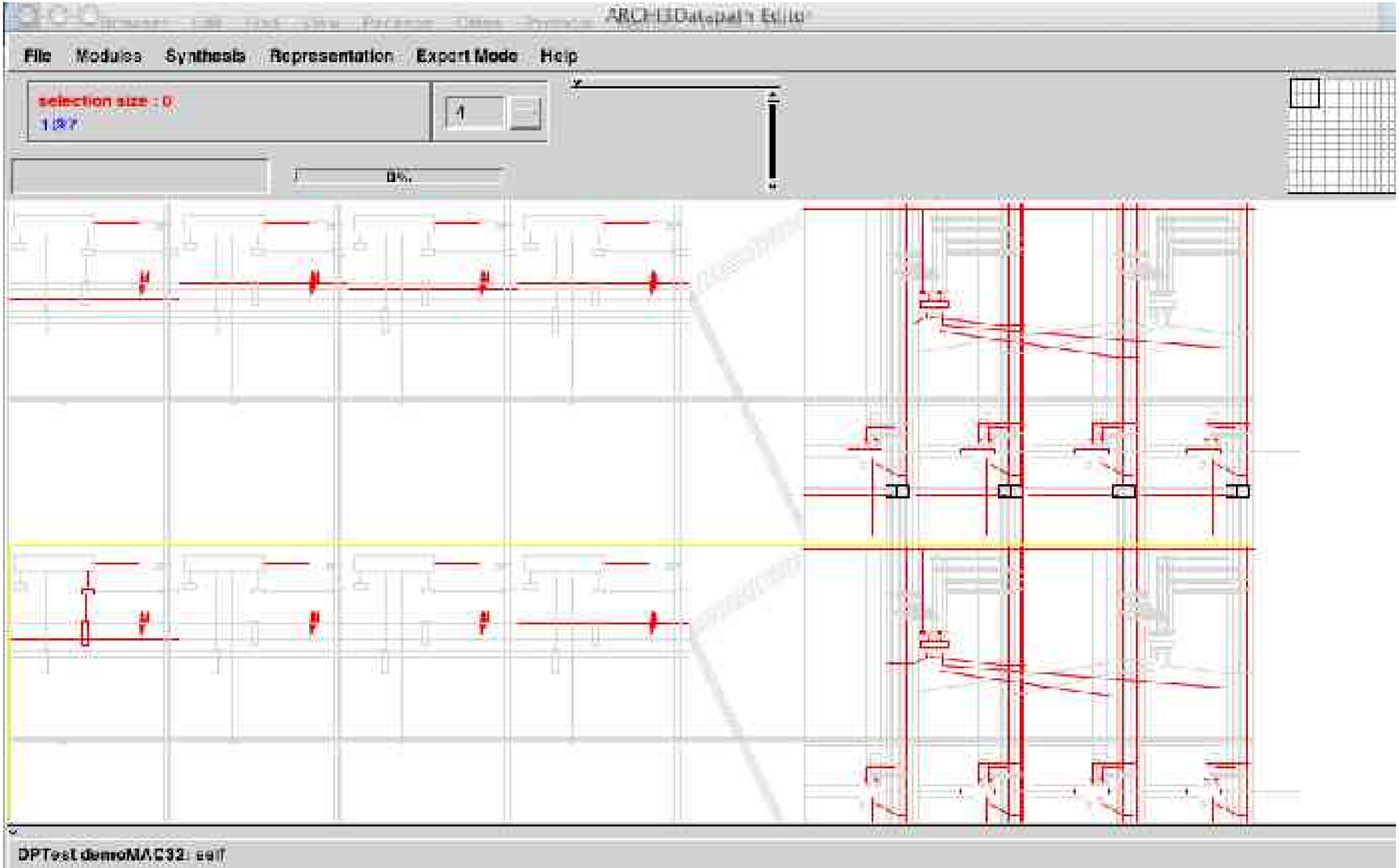
Operation graph description

Compiler and Technology synthesizer

Device modeling

Grammar and Physical tools





Low level architecture description, macro programming

IV. Compiler interface

Motivation and techniques

- Need tools for :
  - rapid prototyping
  - Input close to algorithmic programming
  - Trying to keep expressiveness for the designer
- Generating code suitable for the Madeo back-end programming model
- Multi-threaded control program
- Heterogeneous hardware accelerators, programmable or not.

As an example: PIPS tools

- Source-to-source code restructurer (École des Mines de Paris since 1988)
- Linear algebra framework: many things represented as integer polyhedra
- Precise semantics analysis: predicates on integer variables, various data dependence graph,...
- Many different analysis and transformation phases to compile, optimize, restructure,...
- Cope with interprocedurality and big programs.

- Functions can deal with hardware hierarchy
- Supercomputer techniques in a SoC compiler

Compiler flow characteristics

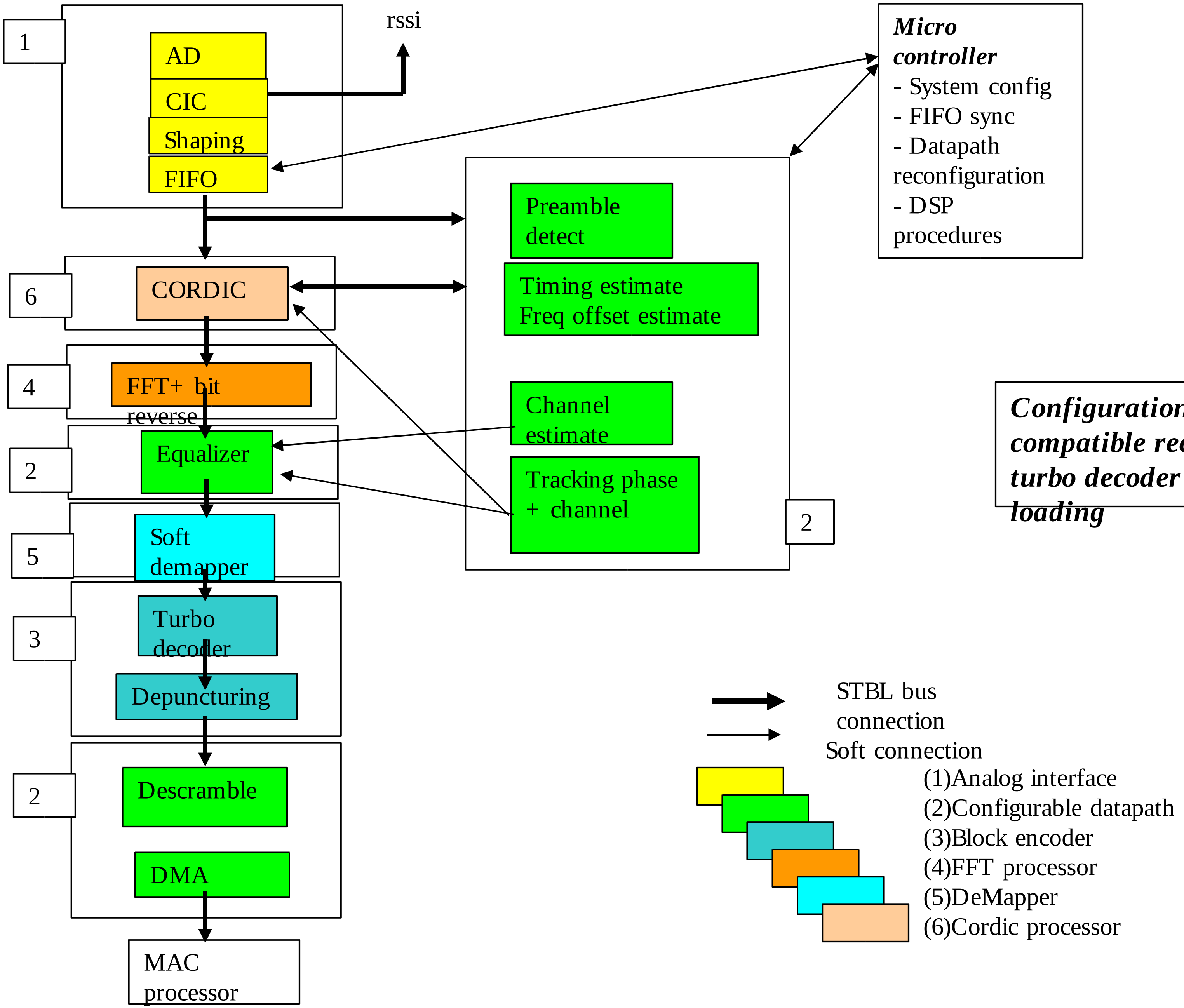
1. C or Fortran PIPS parser
2. Optimizations and transformations express needed parallelism
   - Fine grain parallelization for SIMD or vector like operators
   - Coarse grain parallelization to express task parallelism
3. Static complexity evaluator estimate the hardware cost
4. Hardware part is isolated in a new separate function
   - Control flow is replaced by finite state machines
   - Data-flow is computed according to PIPS polyhedral array region to generate bus movement
5. Prettyprint the source for Madeo hardware generation
6. C or Fortran generated for simulation

V. Work status & conclusion

- Architecture: being fabricated, (2 units = 14mm2, 0.13µ ST)
- Low level tools:
  - arithmetic + ram modeled,
  - Coarse grain P&R possible under programmer control
  - Fine grain : tools working, (but not for M2000)
- Compiler interfaces (PIPS & BSS)
  - Generates control automata suitable for low level tools
  - Operation graph automatic, P&R not implemented
- Attractive project, a number of issues!

Thanks.

Configuration for a 802.11a compatible receiver with turbo decoder and bit loading



1
AD
CIC
Shaping
FIFO
rssi

Micro controller
- System config
- FIFO sync
- Datapath reconfiguration
- DSP procedures

6
CORDIC

4
FFT+ bit reverse

2
Equalizer

5
Soft demapper

3
Turbo decoder
Depuncturing

2
Descramble
DMA

MAC processor

Preamble detect
Timing estimate
Freq offset estimate
Channel estimate
Tracking phase + channel
2

STBL bus connection
Soft connection

🟨 (1)Analog interface
🟩 (2)Configurable datapath
🟦 (3)Block encoder
🟧 (4)FFT processor
🟦 (5)DeMapper
🟫 (6)Cordic processor

Operation modes: control

- External process can:
  - Reset the unit
  - Write configuration memory
  - Load configurations to devices
  - Receive and emit messages, including synchronisation
    - To and from different threads