Smalltalk-80 : hardware and software

Bernard Pottier

[1]LESTER - Architecture & Système
CNRS FRE 2734 - Université de Bretagne Occidentale
pottier@univ-brest.fr

ESUG'05 16/8/05

Outline

Group presentation

Status:

- Architectures and Systems (AS) is a research group at Université de Bretagne Occidentale, Brest, France.
- AS participates in the LESTER, a laboratory distributed on two universities at Lorient and Brest,
- LESTER received a recognition from french CNRS for its research activities on software methods for integrated systems: synthesis, power consumption, reconfigurable devices, algorithms for communications.

AS is a small group (at most 10 persons) working in computer science, sometime in relation with people working in electronics (academy and industry).

Previous projects

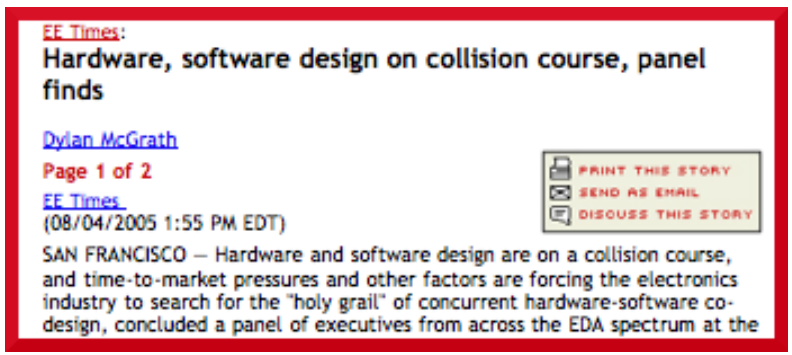
Summary of my researches at UBO:

- 80-86 : virtual machines and support for programming languages (Lisp, Smalltalk, . . .)
- 87-89 : applications of concurrent architectures (Transputers, data-flow languages, mapping signal processing applications),
- 90-95 : parallel and reconfigurable architectures, dedicated language designs and synthesis techniques. Design of a parallel reconfigurable machine (ArMen),
- 96-2005 : tools for reconfigurable circuit ans systems (Madeo workbench).

In this last period, most of the work was achieved using Smalltalk-80 (Visualworks).

Motivations : software problems in the silicon industry

EETimes Aout 2005 :



EE Times:

Hardware, software design on collision course, panel finds

Dylan McGrath

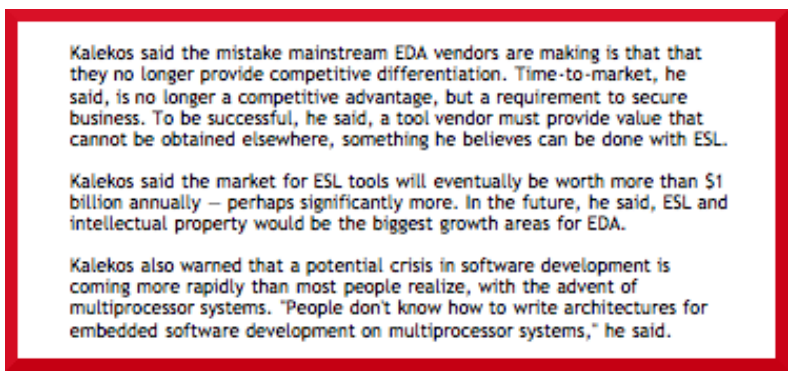
Page 1 of 2

EE Times

(08/04/2005 1:55 PM EDT)

PRINT THIS STORY
SEND AS EMAIL
DISCUSS THIS STORY

SAN FRANCISCO — Hardware and software design are on a collision course, and time-to-market pressures and other factors are forcing the electronics industry to search for the "holy grail" of concurrent hardware-software co-design, concluded a panel of executives from across the EDA spectrum at the



Kalekos said the mistake mainstream EDA vendors are making is that that they no longer provide competitive differentiation. Time-to-market, he said, is no longer a competitive advantage, but a requirement to secure business. To be successful, he said, a tool vendor must provide value that cannot be obtained elsewhere, something he believes can be done with ESL.

Kalekos said the market for ESL tools will eventually be worth more than $1 billion annually — perhaps significantly more. In the future, he said, ESL and intellectual property would be the biggest growth areas for EDA.

Kalekos also warned that a potential crisis in software development is coming more rapidly than most people realize, with the advent of multiprocessor systems. "People don't know how to write architectures for embedded software development on multiprocessor systems," he said.

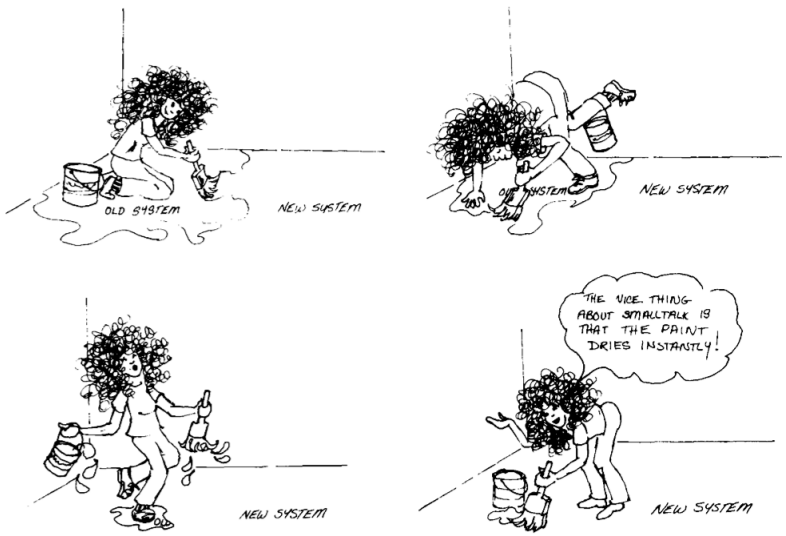
Pressure on integrated application developments:

- Quick developments are mandatory,
- Part of the software is increasing,
- The frontier between software and hardware is less and less clear,
- Tools are inaccurate,
- Multiprocessing is a key point.

Motivations to use Smalltalk at low level

- Smalltalk-80 has powerful development environments and is ideal to build generic solutions,
- It is easy to extend the language because it is not bound to a small number of types (int, char..).
- An interest is appearing for evolutions on Smalltalk allowing to take benefits of the emerging technologies.

Frustrations and dreams with current virtual machines



Long term desirable evolutions:

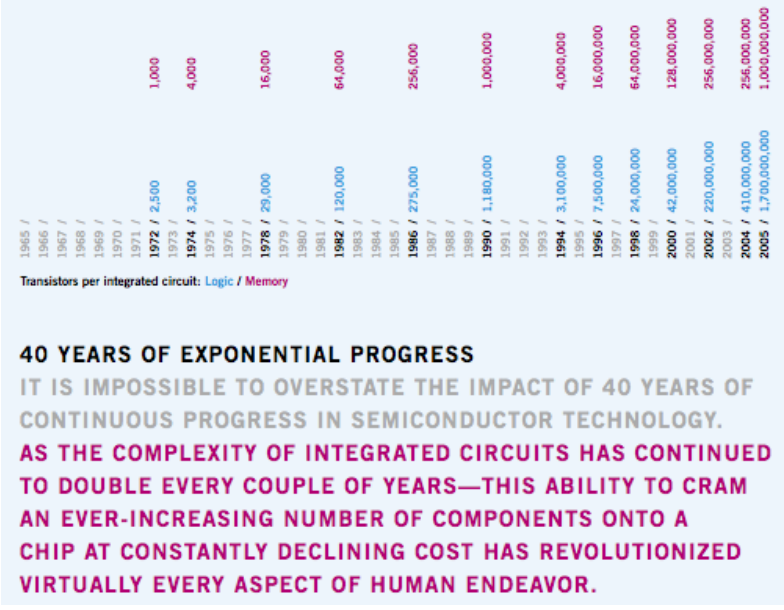
- Numeric processing with new hardware primitives, (arbitrary floating and fixed points, Galois fields, etc . . .),
- Automatic synthesis of these circuits from behavioural code,
- Concurrent execution for Collections (pipelined or data parallel), and Stream computations,

Outline

THE ELECTRONIC INDUSTRY ROADMAP

Technology progress: a scientific, economic and social fact

Exponential increase in transistors,



(from the SIA Report 2005)

Effect of Moore's law, $\times 10000$ transistors increase in 25 years.

1. 1982: logic = 120K, ram = 64K
2. 2005: logic = 1.7G , ram = 1G

Increase expected to continue until 2020.

In the meantime, nanotechnologies are preparing cheaper, smaller, more flexible integration solutions with a variety of applications.

Industry roadmaps

The semiconductor industry prepare roadmaps, listing potential difficulties, preparing objectives, research, and use of the technologies. This is the underlying activity behind Moore's law.
A number of reports are available on this roadmap:

- http://www.sia-online.org/
- http://public.itrs.net

The future is :

- massive parallelism
- software synthesis, or hard/soft synthesis

..rather than hardware developments.

Known Challenges through 2009

Silicon complexity : devices and interconnects	(SP) Technology and library characterization (SP) eDRAM, eFPGA
System complexity : number of states, design, diversity	(SP) verifying systems with exploding number of states (SP) scalable algorithms
Design Productivity	(S) Integrating 3rd party components (SP) Design tool interoperability
Time-to-market	(S) Support for platform based design (SP) Exploiting Parallel Processing

Variety of hardwares

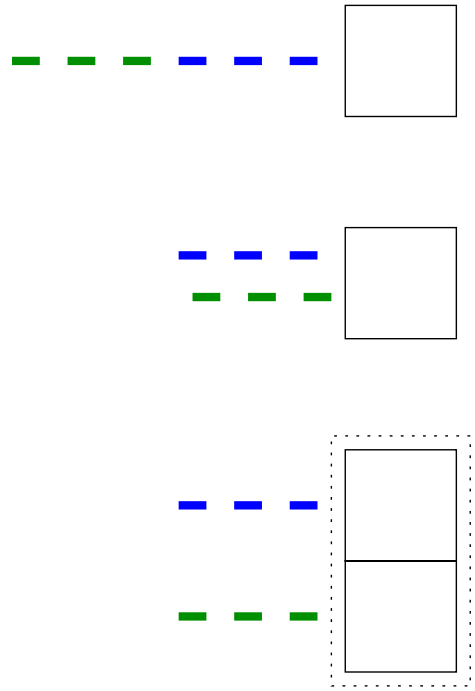
As Low Scale Integration (LSI) has disappeared, VLSI is also on the edge to be replaced by new technologies. Currently, the hardware available for application developments is:

1. $\downarrow$ microprocessors: VLSI, does not scale.
2. $\uparrow$ system on chips: VLSI-ULSI, scale, but is specific.
3. $\uparrow$ reconfigurable circuits: VLSI-ULSI, scale, general purpose.

Processors evolve toward multi-processing

General purpose processors have internal instruction level parallelism and evolve toward task level parallelism:

1. time-shared multi-tasking: several tasks are sequenced following time slices or race conditions on one processor,
2. simultaneaous multi-threading (SMT): several tasks run concurrently sharing processor resources,
3. multi-cores: several processors are implemented in the same circuit and act in real concurrency.

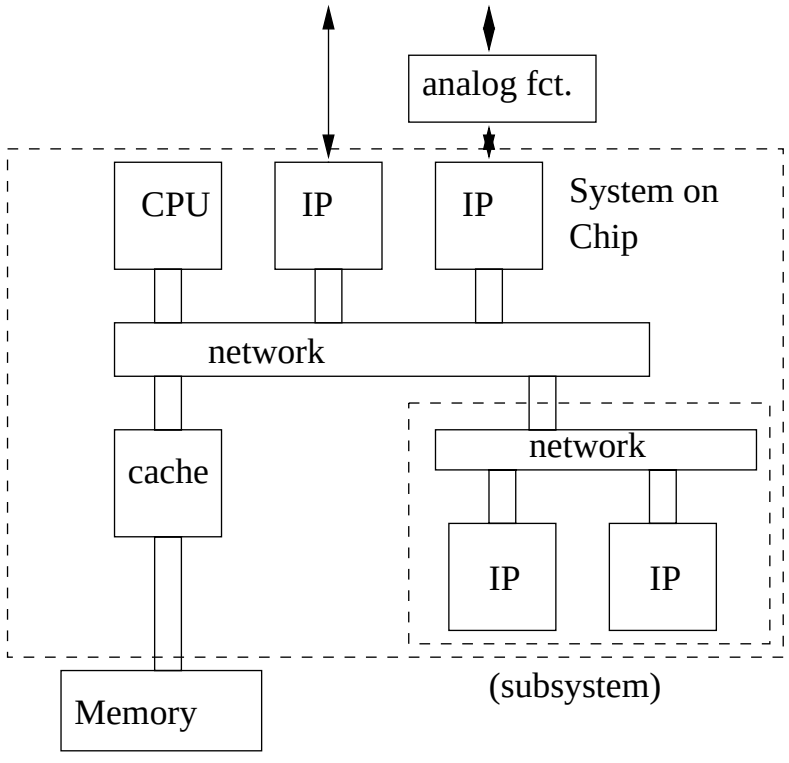


AMD and Intel are currently selling multi-core processors.

System-on-chips (SoC) are for the mass market

SoC are integrated systems optimized for an application.

1. Personal assistants, mobile phones, cameras, set-top-box are implemented at low cost due to the capability of the industry to produce applications on a small number of circuits.
2. These circuits assemble specific components 'IPs', general purpose processors.



Field Programmable Gate Arrays

FPGAs allow to build a dedicated system by characterizing an off-the-shelf component.

1. They are the shortest path from a need to an application because they only need 'software'.
2. They propose random logic integration, set of operators, multi-processors and high speed communications.

3 Xilinx FPGAs: maximum capacities. Configuration = 50MBits

	logic	dsp	system
logic cells	200K	55K	150K
RAM (bits)	6K	6K	10K
oper-ators	100	500	200
CPUs	0	0	2
serial ports	0	0	24

Gigascale will not have a tiny impact

The semi-conductor industry says:

1. Moore's law will continue its curve.
2. We are at a turn with a scale change in the architectures. Parallelism is a key point with several variants.

Gigascale will not have a tiny impact

The semi-conductor industry says:

1. Moore's law will continue its curve.
2. We are at a turn with a scale change in the architectures. Parallelism is a key point with several variants.

We would like to complement referencing two scientists.

From the academy: a Call to Arms

In 2002, David Patterson has published a position on present and future of computers. To summarize:

1. importance of computer for science and technologies,
2. obsession for performances, and ridiculous reliability for users,
3. serious problems with scaling: GB of memory, hundred of processors are the future.

The 'new manifesto' for research in computer science is:

1. more attention to human factors: interest of applications, security, privacy,
2. start *fresh researches* on new architectures, new software systems, new programming systems, new applications.

Computers is a pop culture

Another interesting opinion is given in the Alan Kay interview to ACM Queue (Dec, 2004)

1. computer systems reflect the understanding of the people (like TV),
2. comparison of the software building with pyramides brick tiling, build by brute force,
3. retrogression of systems due to personal computing,

On architecture support:

1. Recall the B5000, and pseudo-codes approach as used by Wirth, and refused by the industry.
2. Critics on current architecture efficiency.
3. Insists on the interest of late binding.

"Just as an aside, to give you an interesting benchmark on roughly the same system, roughly optimized the same way, a benchmark from 1979 at Xerox PARC runs only 50 times faster today."

What to do?

1. we must take into account the hypothesis of large amount of resources given from several sources,
2. most of these resources will be used for intensive computations: signal processing, multimedia, security, scientific,
3. it makes sense to define the whole translation and execution chain with transparency, using the same tools,
4. it is mandatory to pay attention to these topics, this does not forbid to improve the quality of the language and its support for general purpose programming.

To be demonstrated.

SYNTHESIS FROM SMALLTALK

Objectives

Madeo project was started a long term effort withe hope to *survive* in the evolving domain of reconfigurable architectures (FPGAs).
It came after ArMen that developped a parallel machine and ad-hoc compilers quickly lost due to the change in tools and architectures.
Several directions have been explored:

- technology modeling,
- portable physical tools,
- logic synthesis,
- structural design,
- system synthesis.

Architecture modeling

Reconfigurable architectures are charaterized at run time by connecting signal lines or buses, by programming switches, logic cells, operators, by filling memories.
These circuits are replication of patterns that can be organized hierarchically.
This is very similar to the organization of nodes in a program tree, or window components. The common interface allows to:

- compute costs,
- route signals or buses
- draw components in an editor window,
- ...

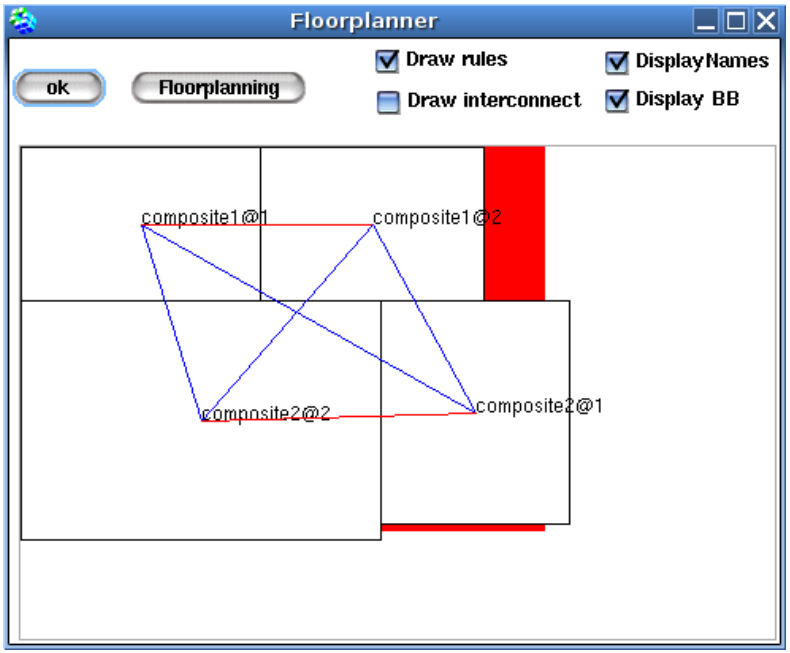
Physical synthesis

PLacing and routing,
Given an architecture, and given an abstract computation graph, this task consists of resource allocation, then route constitution, to obtain an implementation.

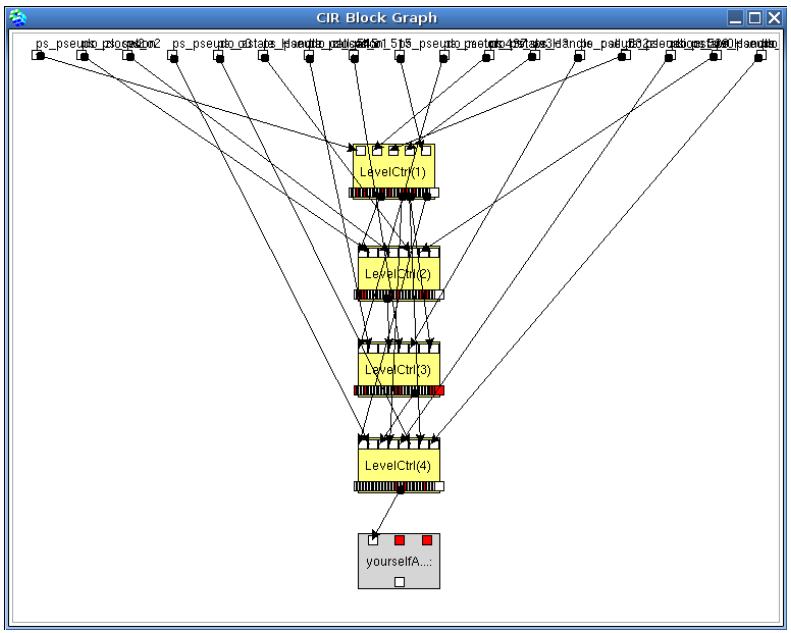
- Resource allocation is a selection of architecture components able to implement a primitive function node from the graph,
- Graph edges are implemented by path of routed components.

Physical synthesis

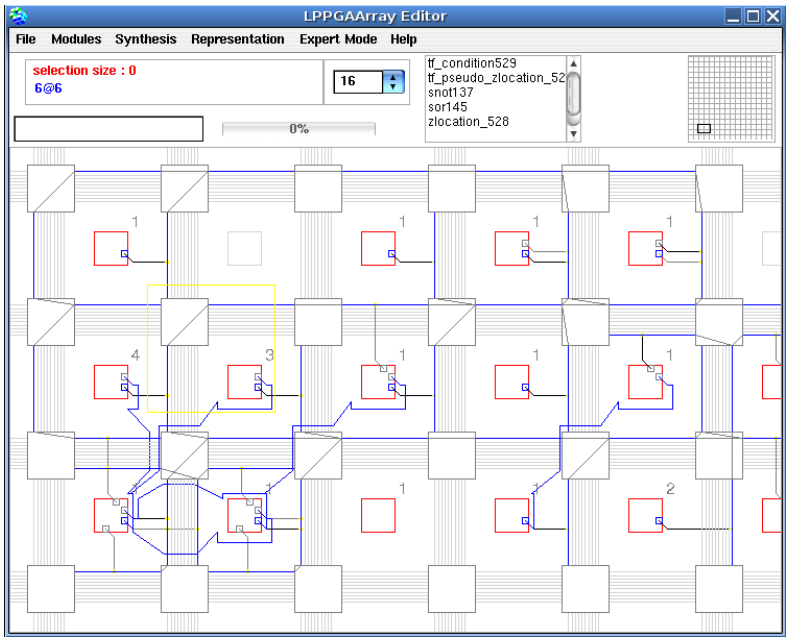
Floor-planning,
Graphes are hierarchies of graphes. For each non-primitive node, an opertion of floor-planning is achieved on the underlying components to find the best arrangement minimizing the number of resources.



Floor planning of a level in a hierachical graph,



Directed acyclic graph,



Equivalent graph placed and routed on an architecture.

Physical synthesis applicability

- Physical tools for logic based FPGAs are mature and has been applied on several experimental or commercial architectures, including a practical workbench on the Virtex1.
- Applicability to data-path and heterogeneous reconfigurable circuits has been partially investigated on a design from STMicroelectronics,
- On-going work on nanofabric modeling in a joint work with a team at UMass,
- Investigations on a massive parallel micro-programmable architecture model in a franco-german project,
- Integrated Project Morpheus will start this fall with the obective to develop a SoC approach with heterogeneous reconfigurable units...

Logic synthesis: an evolving grain

We need data-flow graphs for the physical layer.

- The grain of "*fabrics*" is evolving from gate arrays to tiles grouping operators, registers, memories and buses.
- The semiconductor roadmap predicts a move to mesh of small processors, and reconfigurability will evolve to allow local control,
- The term *Logic Synthesis* match the requirement to produce logic graphs, from higl level programs, for physical synthesis.
- It must be though as *Numeric Synthesis* when data-path are involved, and *Graph Mapping* when processor meshes will be addressed.

Algorithm+Data=Circuits

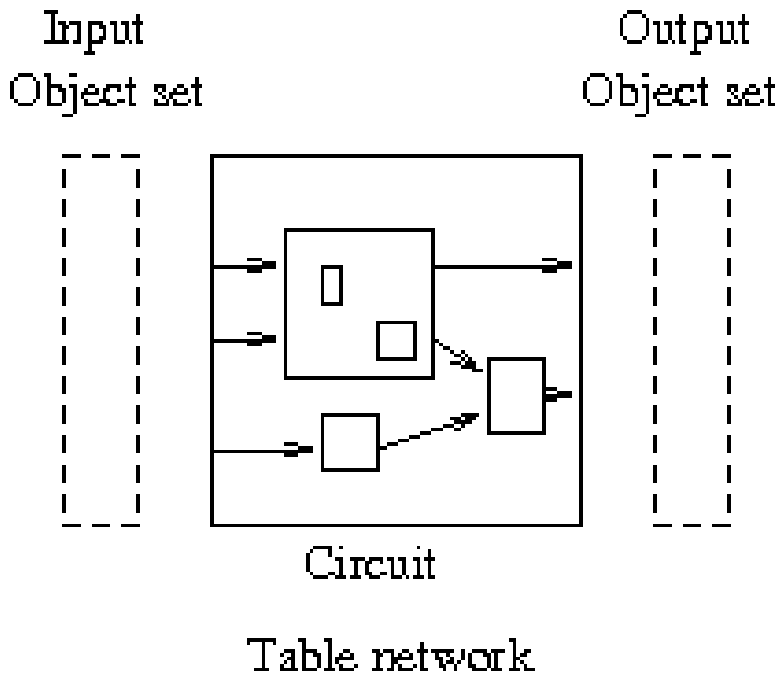
- Logic blocks are produced from Smalltalk methods *writen for this purpose*, but executable in the high level environment.
- As we are using a late-bound language, no type information is normally available.
- This need is filled by requiring a *compilation context* that holds the data that will be operated by the resulting circuit.
- The programming game is to avoid nodes of large complexity, to quickly converge to synthesizable nodes.
- The same code allows to synthesize from objects of different classes,
- Polymorphism survives in this technique,
- Fully automatic process,

Compiler flow for logic synthesis:

1. **parser**: Smalltalk parser, produces a program graph,
2. **DAG**: Dependency analysis, removing temporaries, production of a Directed Acyclic Graph (DAG),
3. **Type inference**: the compiler context is propagated in the DAG:
   1. complex messsage sends are compiled recursively
   2. simple nodes are replaced by a table
4. **Optimization**: removing useless tables and values,
5. **Coding**: codes are attributed to objects,
6. **Logic production**: tables are split for adaptation to a target technology (SIS)

Relations with a software environment

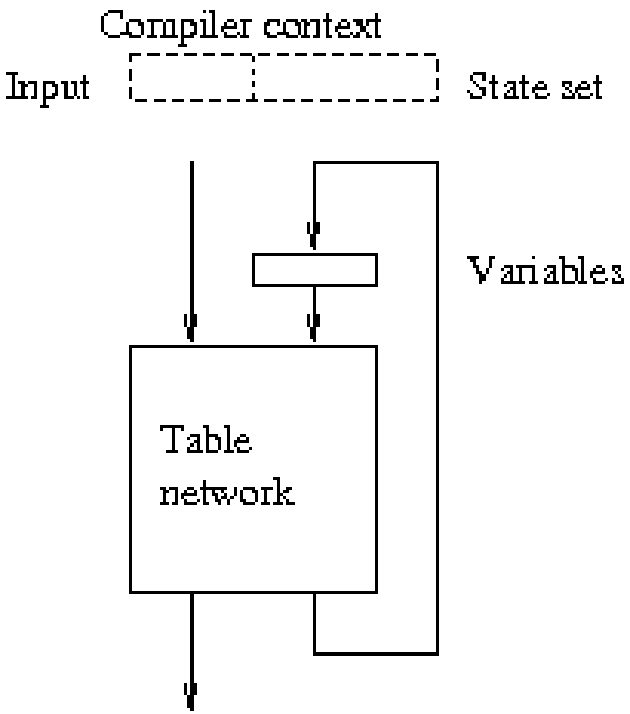
1. Combinational circuits produced at high level, applicative style of programmation,
2. Strong interest for operator production,
3. Very efficient optimization, proof of concept on critical applications (Galois fields and error correction)
4. 100% compatible with the needs of a high level software environment



If used from a virtual machine an adaptation is needed between object in memory and computing circuit (at least indexes).

Logic synthesis applicability

Finite state machines are synthesized by including the state values in the compiler context. The state set is progressively built by retrieving new values from an initial value.



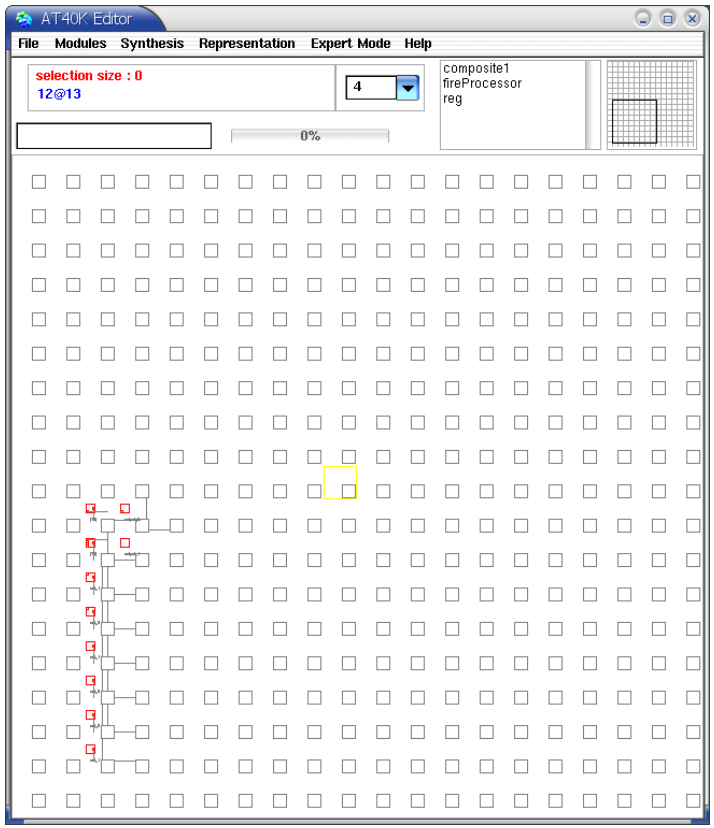
1. Limited for computation organization
2. Currently: only logic synthesis is working. Numeric synthesis for data path, micro machines are not implemented.
3. The type system allows to preserve numeric sets as intervals

Structural representations

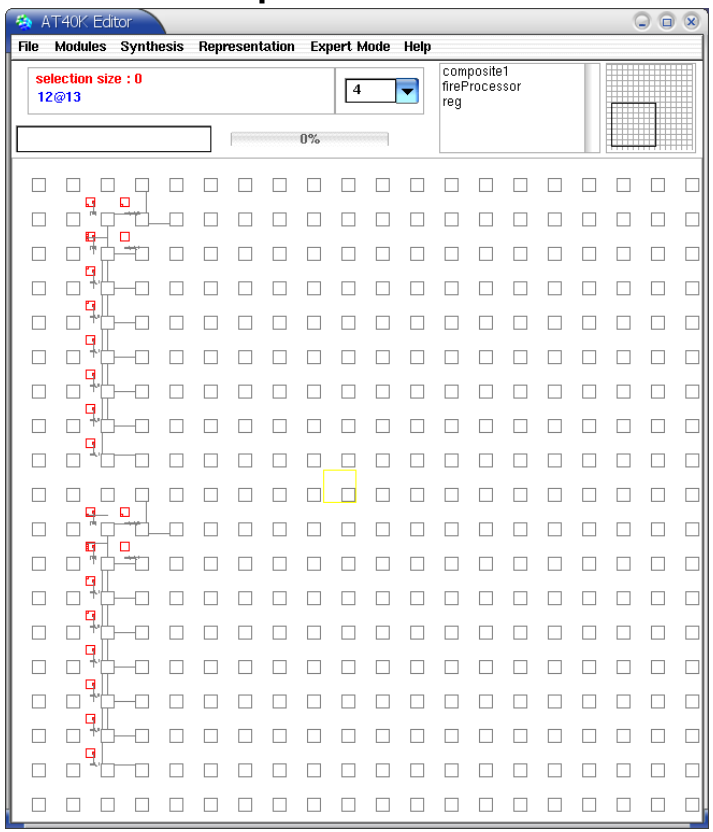
More complex circuits are produced by copying, replicating and binding components together on the top of logic synthesis and Physical synthesis.
These programs can be kept *“architecture independents”* due to the relative geometrical characteristics obtained from the synthesized modules.

Structural representations

Module component known by its geometry



is replicated after computing a correct position.

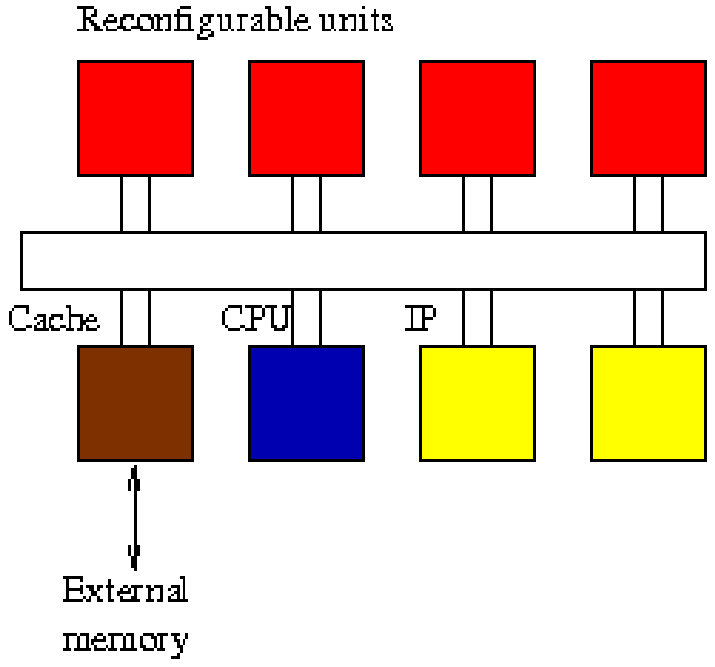


Use status

This software was started as an open alternative to industrial tools for architecture and compiler design.

- **System synthesis**: as a quick complement to the FPGA framework, a synthesizer has been built to allow the automatic production of code from *Smalltalk models*: C code, communication code, micro-code for a camera and FPGA. Demonstration on an embedded camera.
- **Language integration**: use of the framework as a back end for C compilers and the synchronous language Signal.
- **Architecture design**: most of the interest received is actually for modeling, quick tool generation and performance evaluations for new architectures.

- **Morpheus project**: will include a system on chip and reconfigurable units of different grain to be addressed in a similar way, and used under compiler control.
- **Next? dedicated support for virtual machines**: it is attractive and natural to investigate how we could build powerful support for 'virtual' machines using the reconfigurable units and .



Units can implement dedicated functions, reconfigurable on the fly. This is still general purpose HW, considered as a credible alternative to IP for flexibility and time-to-market reasons. It scales.

VIRTUAL MACHINES OF THE PAST

Virtual machines of the past

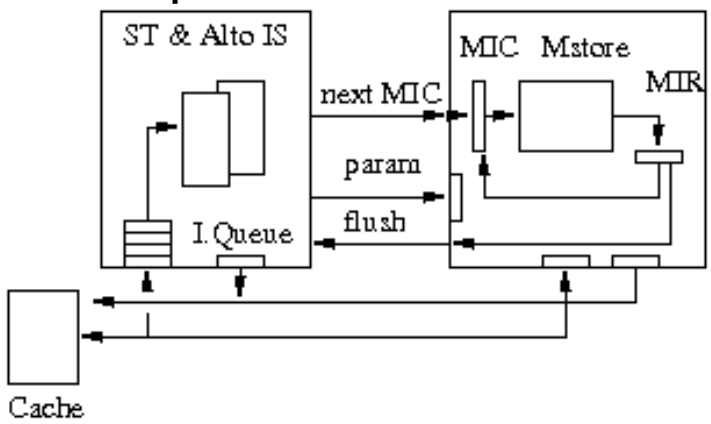
Past and present VM will not address new challenges due to the large change of scale. However they can be used to control larger processing elements coming from synthesis.
For VM, three reference points are :

1. *Micro-coded approach*, as used in the Dorado, provides flexibility in the definition of the instruction set and efficient micro-grain parallelism inside the processor.
   This model is a good match with current HW orientations.
2. *Software approach*, using a processor, we can interpret a pseudo-code and implement all the required functionalities.
   Characteristics: sequential specification and few use of specific HW support. (multicores, SMT ?)
3. *Specific processor tuned for performances*, as was Smalltalk-on-a-Risc (SOAR).
   Will not appear in SoC or FPGAs.

Dorado control path and organization

Pipeline for instruction decoding and execution : the Instruction Fetch Unit share accesses to the memory cache and produce adresses for the micro-machine (up to 6 bytes prepared in adavance).
Micro-machine: has a RAM microstore with 4K words of 36 bits. The IFU is restarted for jumps, message sends ,. . .

Instruction set switching: Smalltalk was implemented with its own IS and micro-code, plus the Alto IS addressed from a B compiler. The IS is switched to execute some primitives.



Dorado processor characteristics

registers : 16 groups of 16 registers (3 were used)
stacks : 4 stacks of 64 registers (used only in the micro-code)
data-path : operated on 16 bits

pipeline stages : 6, including 2 in the IFU.
cycle time : 70ns
process switching: support of multitasking and I/Os in the micro-code
cache : 4K words access in 100ns, virtual memory not used.

A number of resources were not used, and the implementation relied mostly on the speed of the cache.
Cost: 100000$.

Code distribution

IFU + microcode :
interpreter
microcode :

- process switch and kernel operations
- most of the memory management: allocation, recursive freeing, garbage collectors, compactor loops,
- access to the objects,
- block context management,
- simple operations and bitblt.

macrocode (Alto) :

- more or less C
- large integers, floating point
- I/Os
- snapshot,
- other storage management.

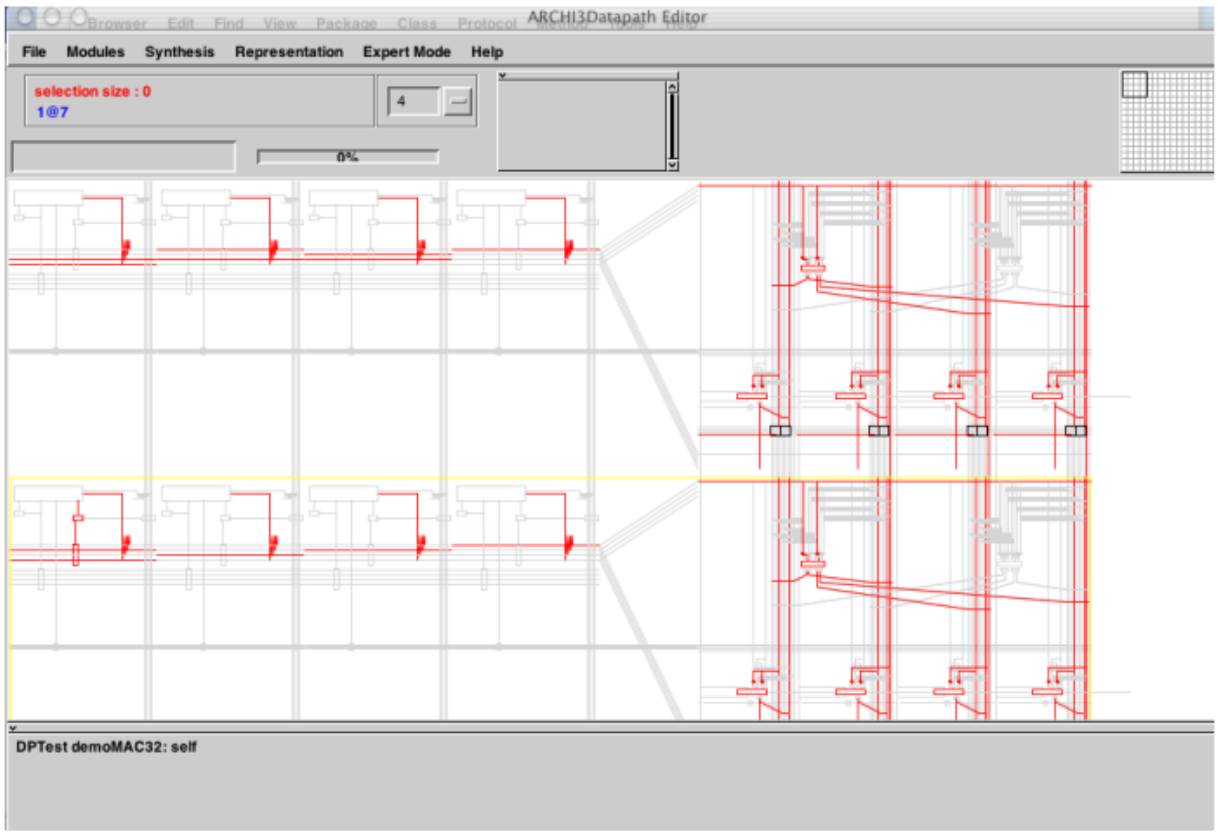
Micro-code interesting features

- **Compared with a software interpreter** : the time spent in the fetch, dispatch loop, process switch is removed by the IFU. Primitives call is reduced.
- **ILP**: micro-code allows 'by hand' instruction level parallelism that could be difficult to produce frm a compiler.
- **most of the code** is located in a surprisingly small micro-code memory.
- **cache architecture remains critical**.

EXAMPLE OF A MIXED-GRAIN RECONFIGURABLE DATA PATH

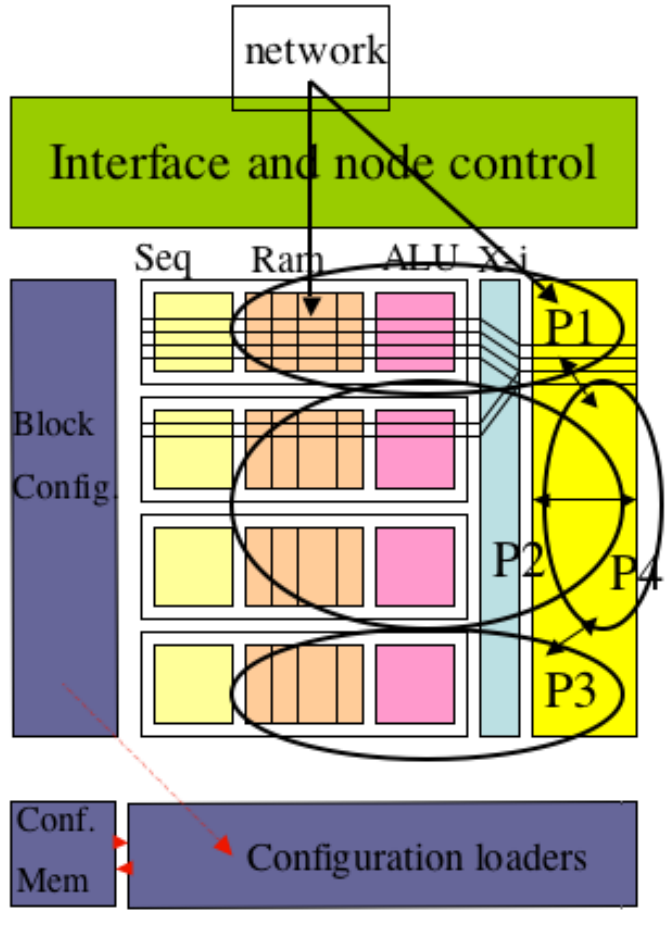
A data path slice

Data path slice showing 4 RAMs of 256 $\times$ 8 *bits* connected to an arithmetic part with 2 multipliers (16 bits), and 4 *8 bits ALU slices*.



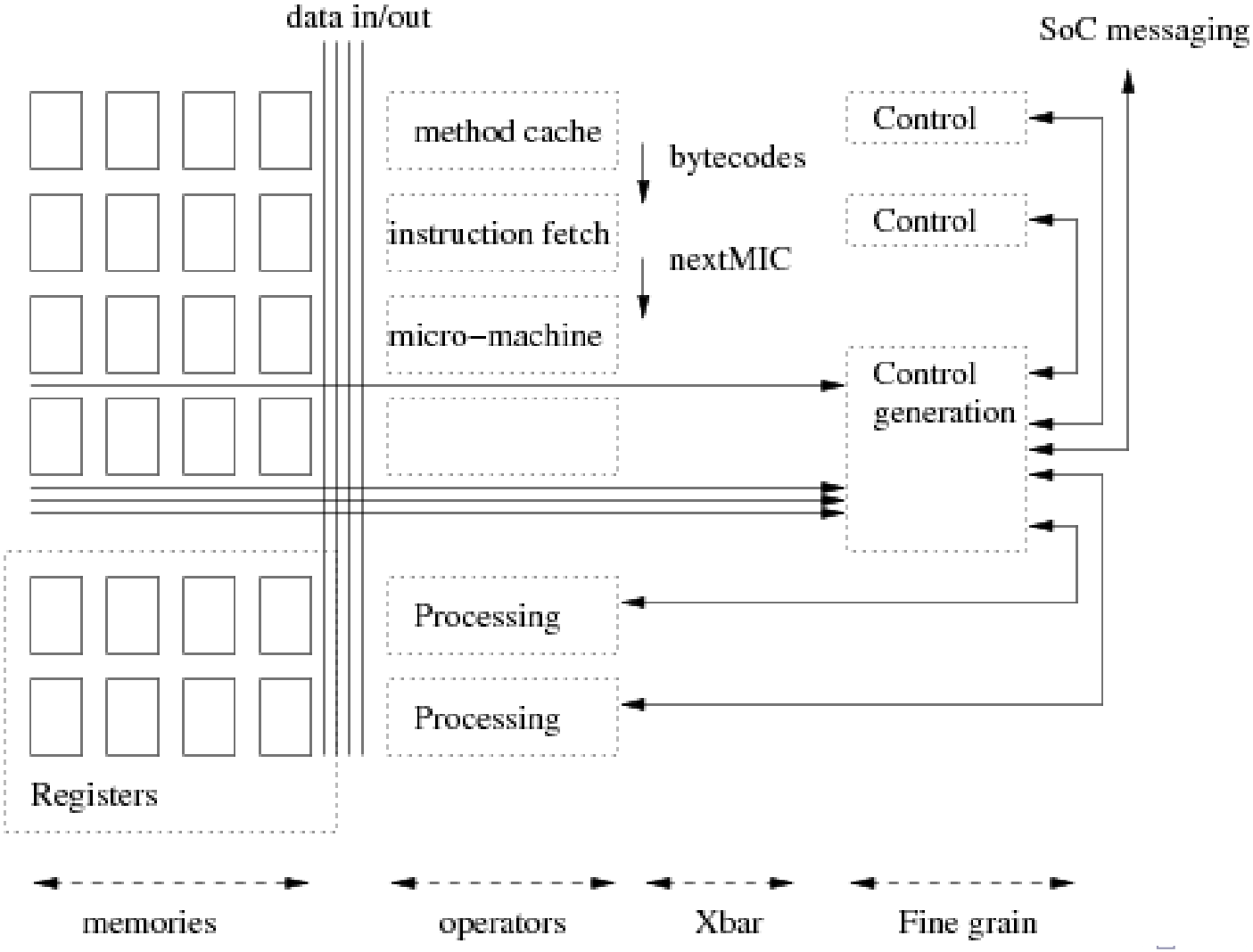
A reconfigurable unit

This unit has 8 similar slices organized as a data path, connected to a fine grain unit (M2000). A reconfigurable loader allows to program connectivities, fine grain from outside. Micro-code is emitted to/from the data-path. (J.Cambonie, V.George, STMicro)



Being a little bit speculative...

Mapping of a Dorado-like architecture on this ST Data-path is nearly possible!



CONCLUSION

As a conclusion

1. *feed back to the industry efforts*: given the future huge amount of hardware resources available, we can imagine methods and tools to produce applications quickly.
2. *it is best to build models and tools for future architectures* now*, rather than wait for the C compiler that will produce parallelism and ensure portability*,
3. *concurrency is the way to go, at the circuit level, distributed processing level, and machine level*,
4. *several mechanisms are lacking in current Smalltalk, especially for spatial parallelism description, time constraint, type systems, processor descriptions*.

Bibliography

Dorado (Lampson/Pier), (Deutsch), Soar (Patterson/Ungar), Self (Ungar/Chambers), Instruction path (DeBaer/Campenhout), Process networks(Lee), . . .

David Patterson.

A call to arms: A new manifesto for systems.

In *CRA Conference on "Grand Research Challenges" in Computer Science and Engineering*, Warrenton, Virginia, May 2002.

..we must broaden our research agenda beyond performance so that our legacy does not remain as fast as flaky... How should we design and program computers when the basic block is a processor rather than a transistor ?

Architecture of SOAR: Smalltalk on a RISC.

D. ungar and r. blau and p. foley and d. samples and d. patterson.

In *ISCA*, 1984.

(see also http://squeak.cs.uiuc.edu/)