Automatic Instruction-Set Specialisation

Paolo Ienne

(based on work with **L. Pozzi, K. Atasu, P. Biswas, N. Dutt,** and **J. Großschädl**)

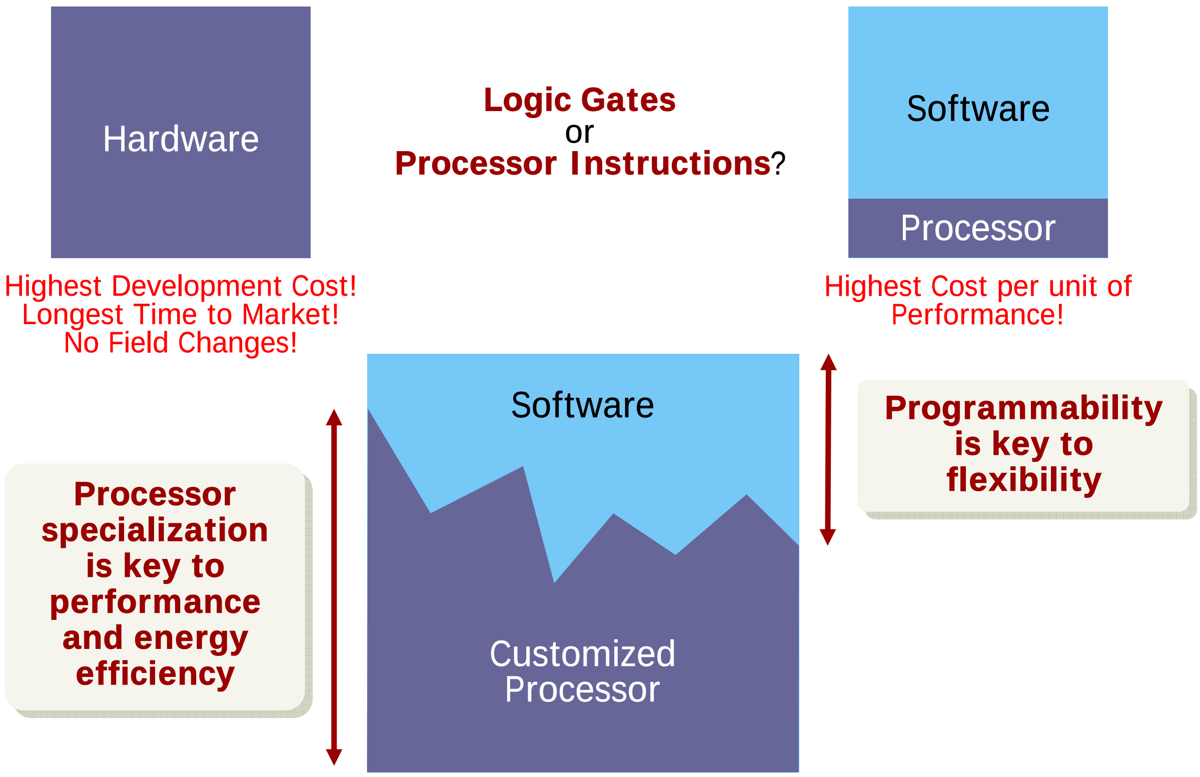
lap csda

Processor Architecture Laboratory (LAP)
& Centre for Advanced Digital Systems (CSDA)

EPFL
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Ecole Polytechnique Fédérale de Lausanne (EPFL)

Customizable Processors



Highest Development Cost!
Longest Time to Market!
No Field Changes!

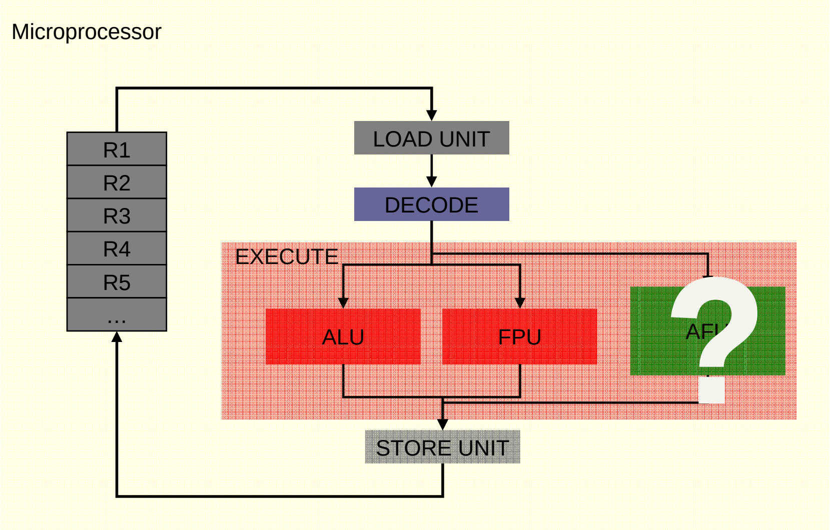
Highest Cost per unit of Performance!

Programmability is key to flexibility

Processor specialization is key to performance and energy efficiency

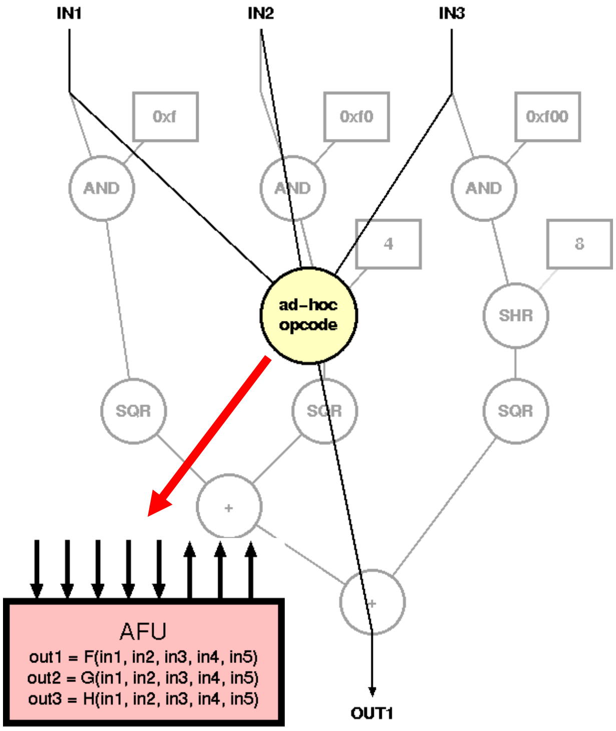
Instruction Set Extensions

- A “safe” technique for customization



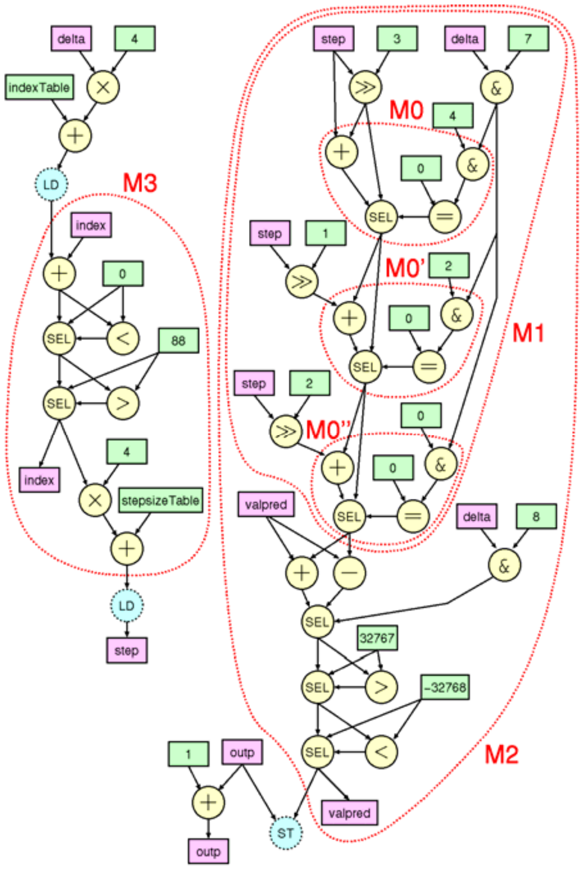
- Available in many commercial processors (from MIPS, STM, IFX, Tensilica, ARC, Xilinx, Altera,…)

Instruction Set Extensions (ISEs)


IN1
IN2
IN3
0xf
0xf0
0xf00
AND
AND
AND
4
8
ad-hoc opcode
SHR
SQR
SQR
SQR
+
+
OUT1
AFU
out1 = F(in1, in2, in3, in4, in5)
out2 = G(in1, in2, in3, in4, in5)
out3 = H(in1, in2, in3, in4, in5)

- Collapse a subset of the Direct Acyclic Graph nodes into a single **Application-Specific Functional Unit** (AFU)
  - Exploit cheaply the parallelism within the basic block
  - Simplify operations with constant operands
  - Optimise sequences of instructions (logic, arithmetic, etc.)
  - Exploit limited precision

The Basic Problem

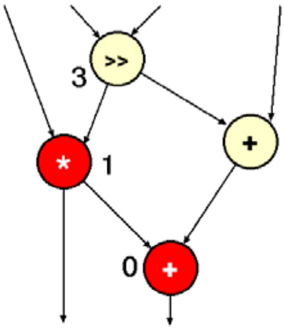


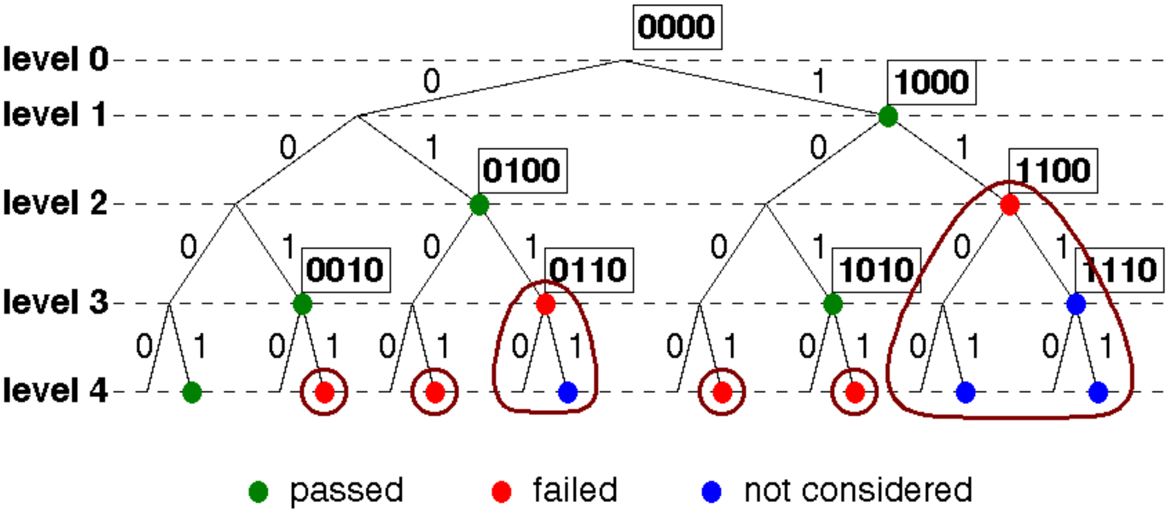
- **Goal**: Find subgraphs
  - having a user defined maximum number of inputs and outputs,
  - including disconnected components, and
  - that maximize the overall speedup

Pruning the Search Space

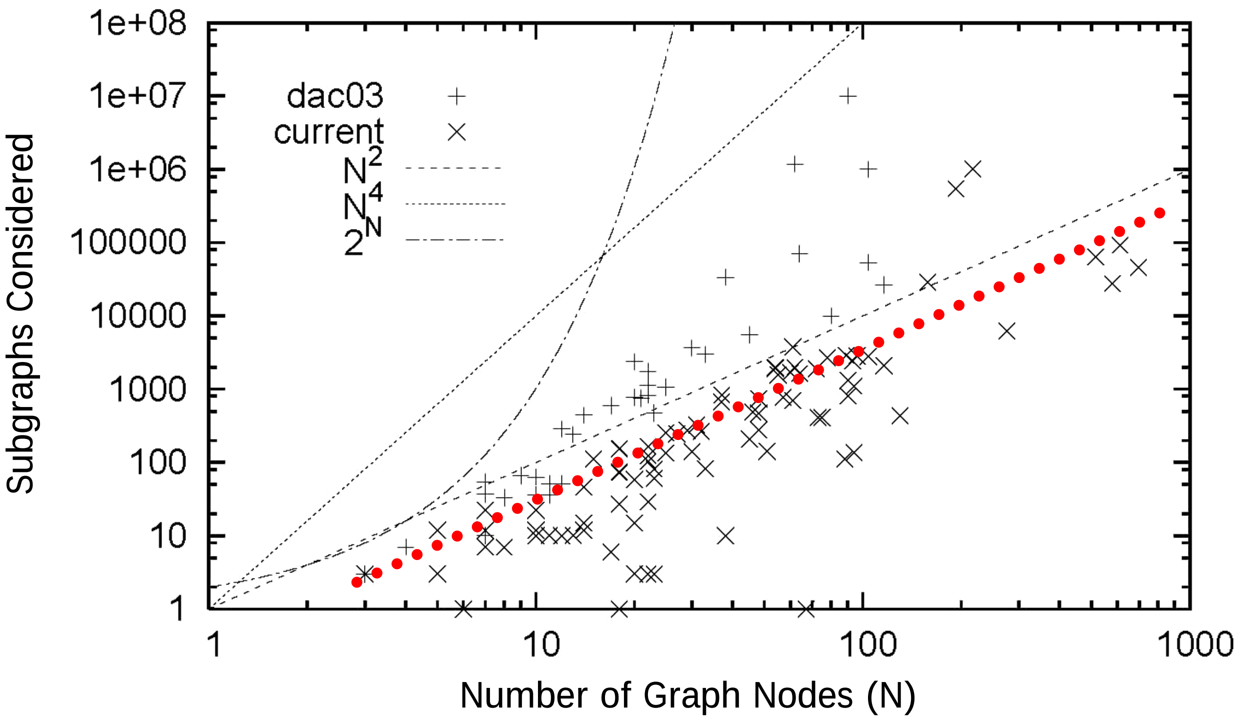
2^N possible solutions for an **N**-node graph (N is usually in the range 10-100 → **too many to explore** exhaustively)

Use the constraints of the problem to prune the search space





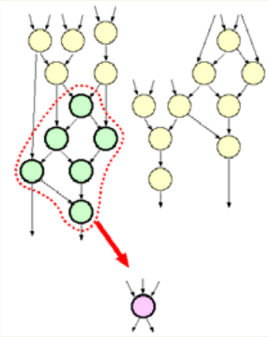
Algorithm Performance



Number of subgraphs considered using an output port constraint of two

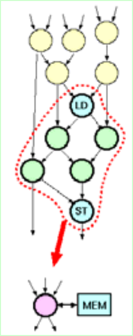
Many Related Problems

Automatic Identification of Instruction-Set Extensions



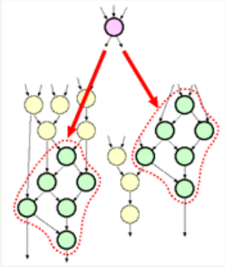
Atasu, Pozzi, Ienne (DAC 2003, BPA, and TCAD 2006)
Biswas, Pozzi, Ienne, Dutt, et al. (DATE 2005 and TVLSI)

Inclusion of Architecturally Visible Registers and Memory



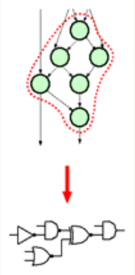
Biswas, Pozzi, Ienne, Dutt, et al. (DAC 2004)
Biswas, Pozzi, Ienne, Dutt, et al. (DATE 2006, BPA Nominee, and TCAD)

Symbolic Algebra for Instruction Selection



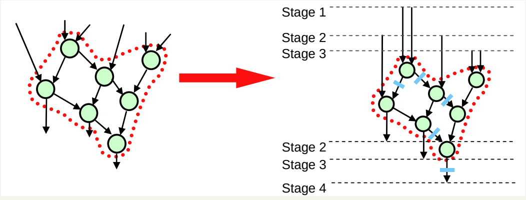
Peymandoust, Pozzi, Ienne, and De Micheli (ASAP 2003)

Arithmetic Optimisations



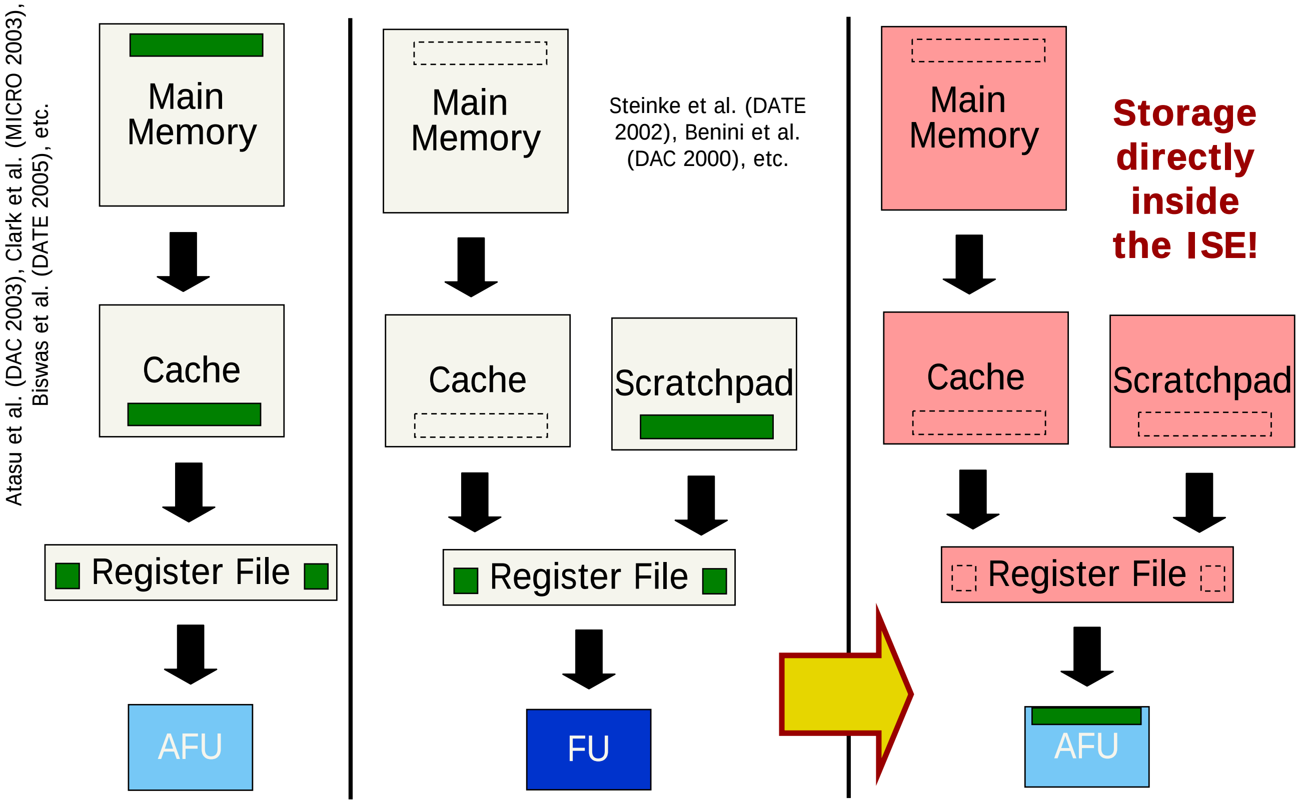
Verma and Ienne (ICCAD 2004)

Pipelining to Relax Port Constraints



Pozzi and Ienne (CASES 2005)

Bringing Data Closer to the Consumer



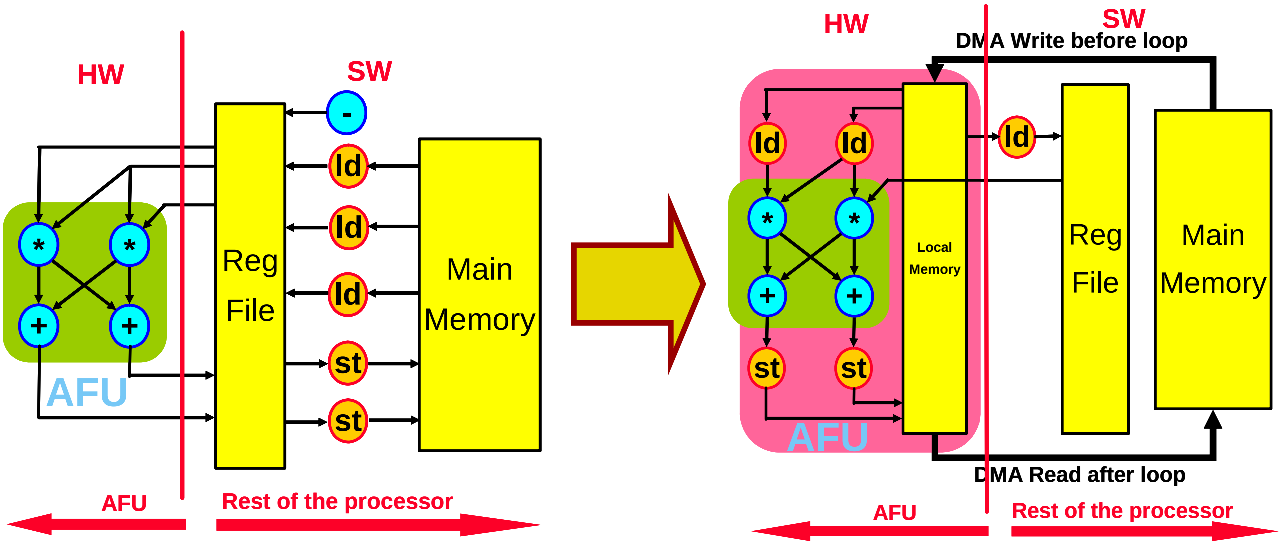
Atasu et al. (DAC 2003), Clark et al. (MICRO 2003), Biswas et al. (DATE 2005), etc.

Steinke et al. (DATE 2002), Benini et al. (DAC 2000), etc.

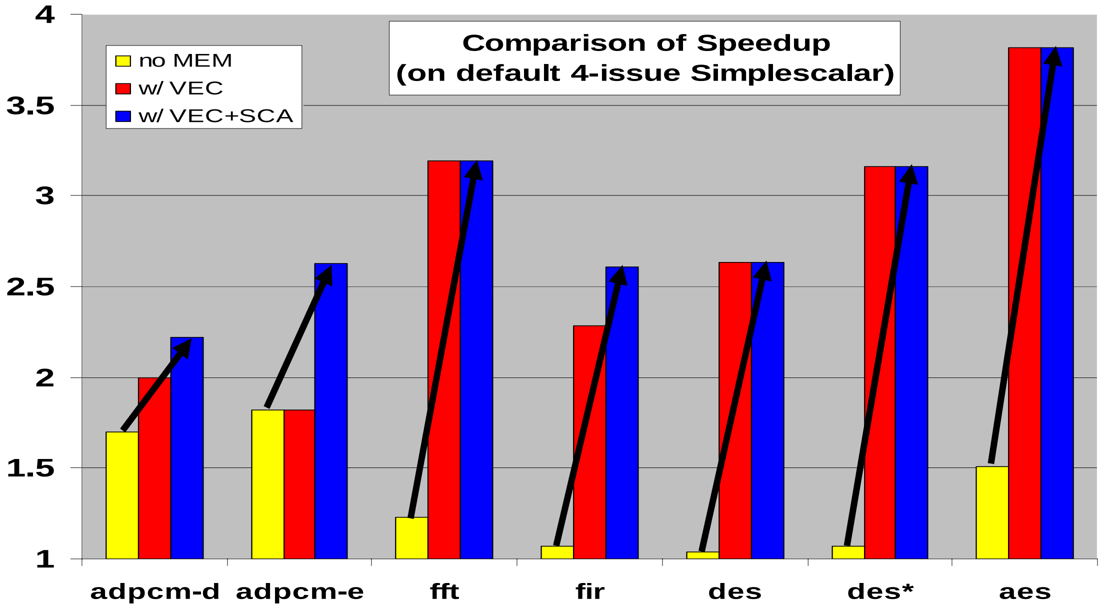
Storage directly inside the ISE!

Adding Local Memory to ISEs

- Include selected **LD/ST operations in subgraphs** for instruction set-extensions
- Decide which pieces of data are best **stored locally** in the AFU
- Preload/unload the AFU memories with **DMA transfers** placed in the most economical spots in the Control Flow Graph



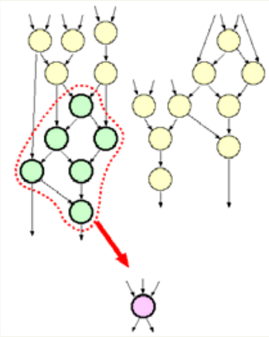
Advantage of Local Storage



Average speedup up from 1.4X to 2.8X

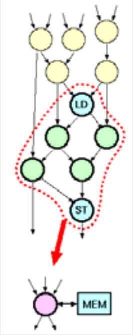
Many Related Problems

Automatic Identification of Instruction-Set Extensions



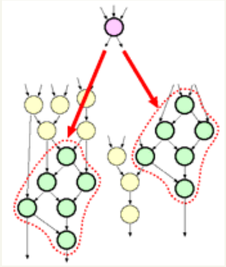
Atasu, Pozzi, Ienne (DAC 2003, BPA, and TCAD 2006)
Biswas, Pozzi, Ienne, Dutt, et al. (DATE 2005 and TVLSI)

Inclusion of Architecturally Visible Registers and Memory



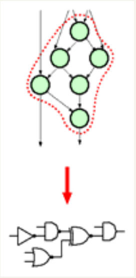
Biswas, Pozzi, Ienne, Dutt, et al. (DAC 2004)
Biswas, Pozzi, Ienne, Dutt, et al. (DATE 2006, BPA Nominee, and TCAD)

Symbolic Algebra for Instruction Selection



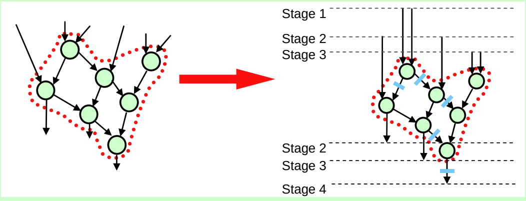
Peymandoust, Pozzi, Ienne, and De Micheli (ASAP 2003)

Arithmetic Optimisations



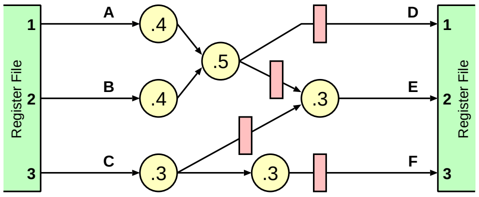
Verma and Ienne (ICCAD 2004)

Pipelining to Relax Port Constraints



Pozzi and Ienne (CASES 2005)

Not Enough RF Ports?

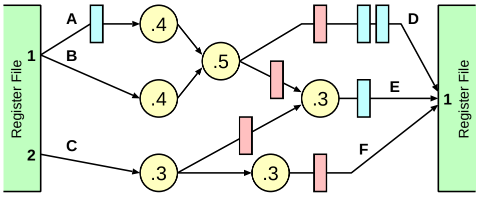


Rd			Wr		
1	2	3	1	2	3
A	B	C			
			D	E	F

2

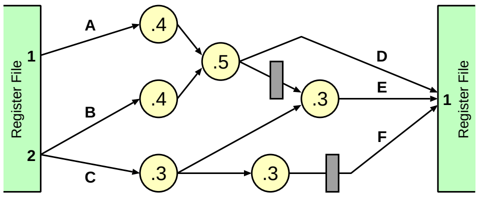
One wants to add **pipeline registers** to the application-specific functional unit...

...and one wants to add **delay registers** to transfer sequentially more values than the register file can in a cycle



Rd		Wr
1	2	1
A		
B	C	
		F
		E
		D

5



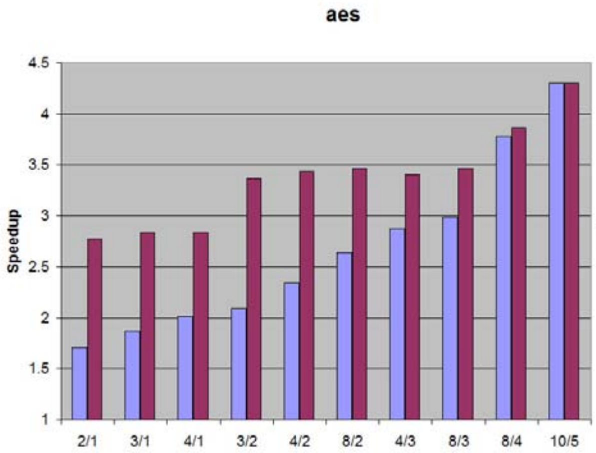
Rd		Wr
1	2	1
A	B	
	C	D
		E
		F

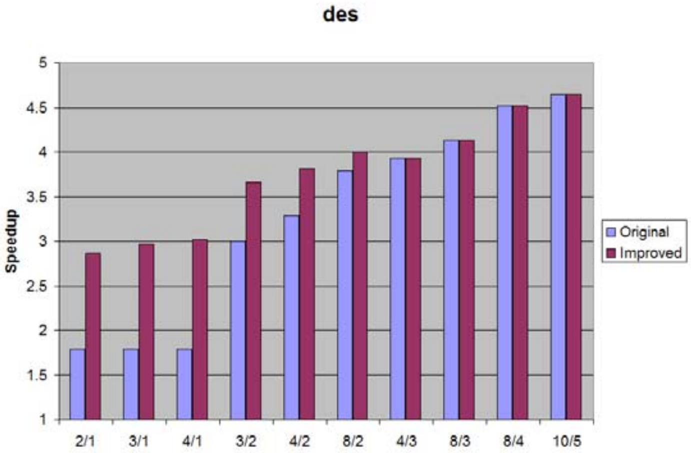
3

Doing both at once minimizes the cost (cycles and registers)

Pipelining and Sequentialization

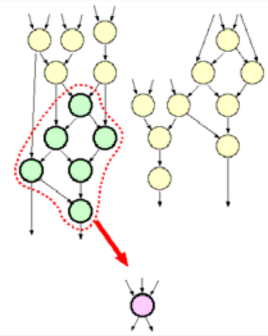
- Algorithm to solve the following problem
  - **Pipelining**: no path inside the application-specific functional unit longer than cycle time
  - **Legality**: same number of registers on every path
  - **I/O Schedulability**: no more than N_{in} input values nor more than N_{out} output values required per cycle





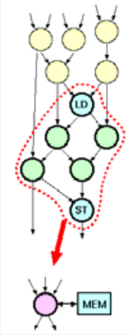
How Competitive Can We Be to Manual ISE Design?

Automatic Identification of Instruction-Set Extensions



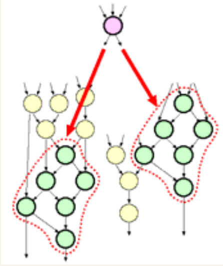
Atasu, Pozzi, Ienne (DAC 2003, BPA, and TCAD 2006)
Biswas, Pozzi, Ienne, Dutt, et al. (DATE 2005 and TVLSI)

Inclusion of Architecturally Visible Registers and Memory



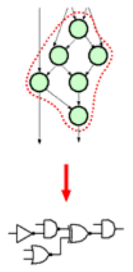
Biswas, Pozzi, Ienne, Dutt, et al. (DAC 2004)
Biswas, Pozzi, Ienne, Dutt, et al. (DATE 2006, BPA Nominee, and TCAD)

Symbolic Algebra for Instruction Selection



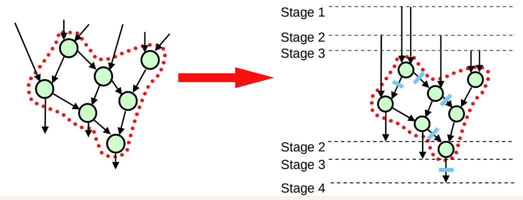
Peymandoust, Pozzi, Ienne, and De Micheli (ASAP 2003)

Arithmetic Optimisations



Verma and Ienne (ICCAD 2004)

Pipelining to Relax Port Constraints



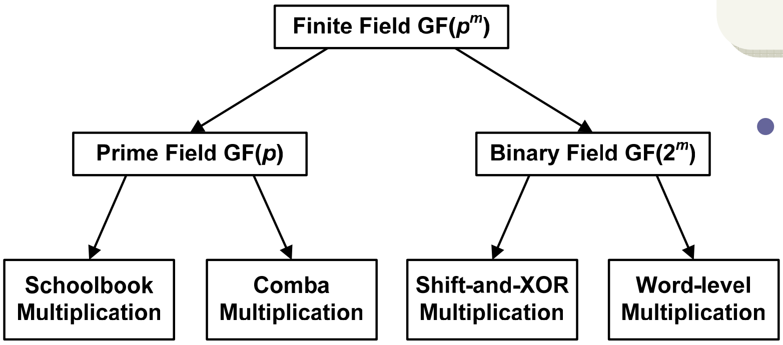
Pozzi and Ienne (CASES 2005)

Often Applications Offer Options in Algorithm Implementation

- **Elliptic curve cryptography**
  - Elliptic curves over a finite field *K*
  - Main operation: point multiplication $k \cdot P$
  - Realized by arithmetic operations in the field *K*
  - Field order: 160–256 bits

Problem:

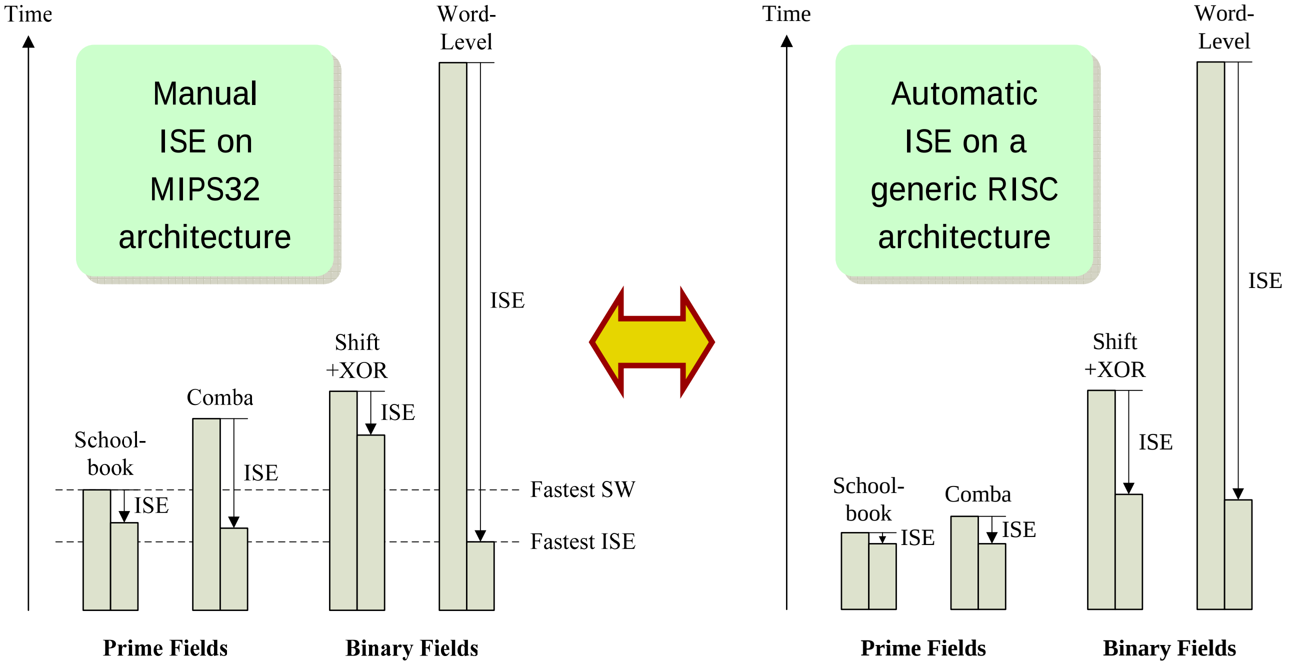
How to help the *algorithm* designer select the best options so that the *implementation* will be most efficient?



- Many implementation options
  - Binary field GF(2*m*)
  - Prime field GF(*p*)
  - Different multiplication algorithms

Use Automatic ISE to Get Early Feedback

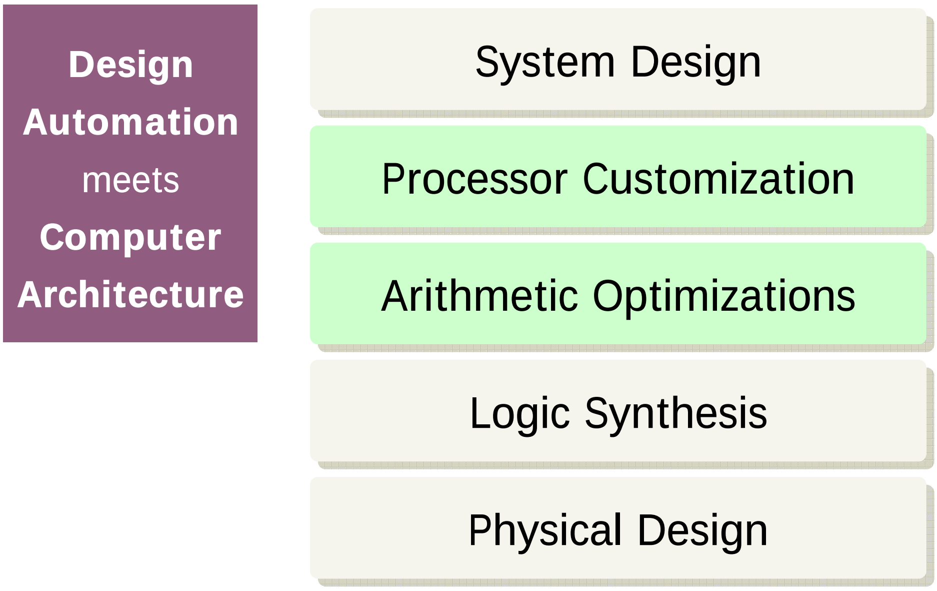
- Großschädl et al. (DATE 2006) have shown a very close qualitative matching between accurate microarchitectural ISE studies and automatic ISE



Manual ISE on MIPS32 architecture

Automatic ISE on a generic RISC architecture

Complex Integrated Systems Still Need Research in Design Automation



Design Automation meets **Computer Architecture**

System Design

Processor Customization

Arithmetic Optimizations

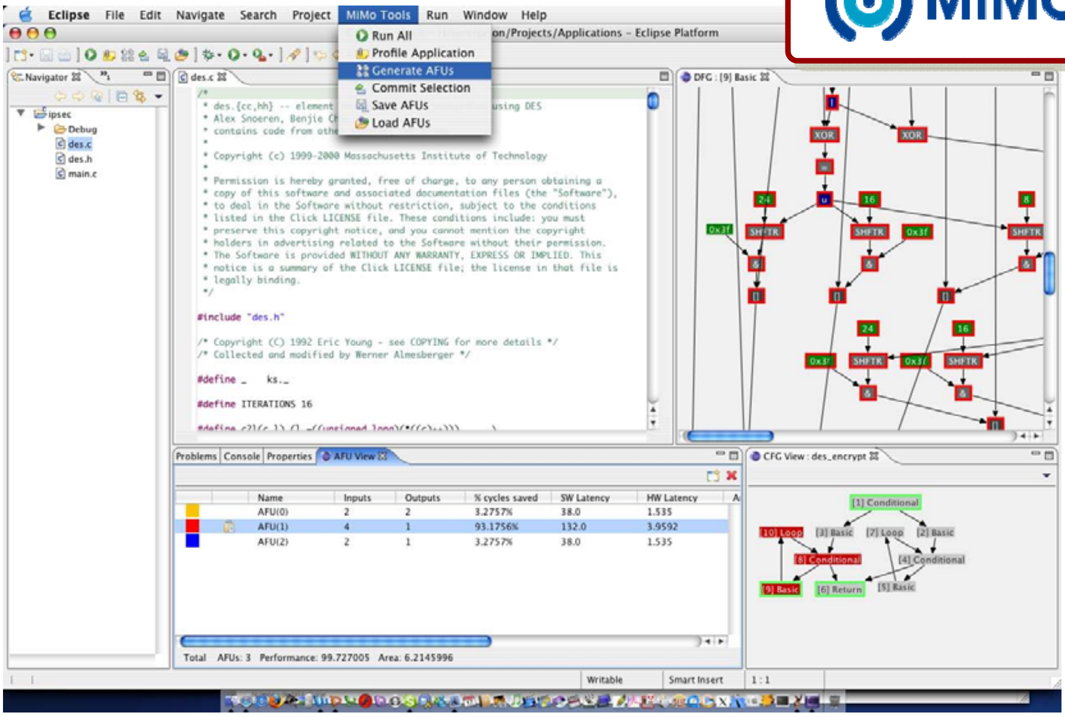
Logic Synthesis

Physical Design

- Customization marks a turn in the history of processors
- Design tools for high-quality datapaths must be to the reach of all designers

You Can Now Buy It...

MIMOSYS



```
/*
 * des.{cc,hh} -- element                         using DES
 * Alex Snoeren, Benjie Ch
 * contains code from othe
 *
 * Copyright (c) 1999-2000 Massachusetts Institute of Technology
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, subject to the conditions
 * listed in the Click LICENSE file. These conditions include: you must
 * preserve this copyright notice, and you cannot mention the copyright
 * holders in advertising related to the Software without their permission.
 * The Software is provided WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED. This
 * notice is a summary of the Click LICENSE file; the license in that file is
 * legally binding.
 */

#include "des.h"

/* Copyright (C) 1992 Eric Young - see COPYING for more details */
/* Collected and modified by Werner Almesberger */

#define _    ks._

#define ITERATIONS 16
```

		Name	Inputs	Outputs	% cycles saved	SW Latency	HW Latency
🟨		AFU(0)	2	2	3.2757%	38.0	1.535
🟥		AFU(1)	4	1	93.1756%	132.0	3.9592
🟦		AFU(2)	2	1	3.2757%	38.0	1.535

Total AFUs: 3 Performance: 99.727005 Area: 6.2145996

Available for

- Xilinx Virtex4 PowerPC
- Altera Nios II
- ARC 6xx and 7xx
- CoWare LisaTek

...and You Can Read About It!

1. From Prêt-à-Porter to Tailor-Made – *Ienne and Leupers*
2. Opportunities for Application-Specific Processors: The Case of Wireless Communications – *Ascheid and Meyr*
3. Customizing Processors: Lofty Ambitions, Stark Realities – *Fisher, Faraboschi, and Young*
4. Architecture Description Languages – *Mishra and Dutt*
5. C Compiler Retargeting – *Leupers*
6. Automated Processor Configuration and Instruction Extension – *Goodwin, Leibson, and Martin*
7. Automatic Instruction-Set Extensions – *Pozzi and Ienne*
8. Challenges to Automatic Customization – *Topham*
9. Coprocessor Generation from Executable Code – *Taylor and Stewart*
10. Datapath Synthesis – *Brisk and Sarrafzadeh*
11. Instruction Matching and Modeling – *Parameswaran, Henkel, and Cheung*
12. Processor Verification – *Große, Siegmund, and Drechsler*
13. Sub-RISC Processors – *Mihal, Weber, and Keutzer*
14. Application Specific Instruction Set Processor for UMTS-FDD Cell Search – *Puusaari, Yli-Pietilä, and Rounioja*
15. Hardware/Software Tradeoffs for Advanced 3G Channel Decoding – *Schmidt and Wehn*
16. Application Code Profiling and ISA Synthesis on MIPS32 – *Leupers*
17. Designing Soft Processors for FPGAs – *Bilski, Mohan, and Wittig*

DESIGN TECHNOLOGIES AND APPLICATIONS
CUSTOMIZABLE EMBEDDED PROCESSORS
SYSTEMS ON SILICON
Edited by PAOLO IENNE RAINER LEUPERS
MK

New
July 2006