

Determining the Optimum Extended Instruction-Set Architecture for Application Specific Reconfigurable VLIW CPUs

**C.Alippi, W.Fornaciari,
L.Pozzi, M.Sami**

**Politecnico di Milano
Milan, Italy**

Outline of the Presentation

- ❑ Introduction
- ❑ The Target architecture
- ❑ Identification of Candidate FPGA-opcodes
- ❑ FPGA-opcodes selection by Genetic Algorithms
- ❑ Experimental Results

Structure Set-up and Background

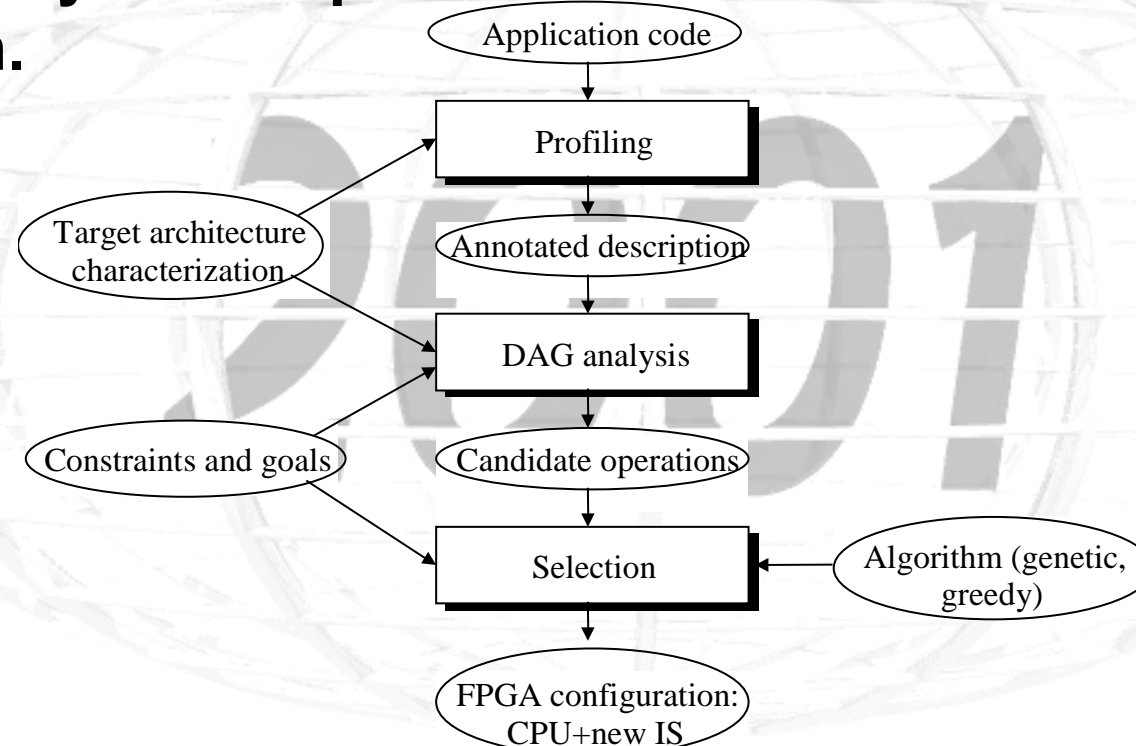
- ❑ We consider a core CPU with several Functional Units some of which can be reconfigured
- ❑ Interesting small “segments” of the computation can be identified on the Data Flow Graph
- ❑ Such segments can be associated with FPGA-based opcodes
- ❑ A run-time reconfiguration can be considered to solve the FPGA area limitation

The Target Architecture

- ❑ **A VLIW processor augmented with Reconfigurable Functional Units**
- ❑ **FUs are organised in clusters each with FUs and a register file**
- ❑ **the ISA provides specific instructions for copying data from different register files in different clusters**
- ❑ **The RFU constitutes a cluster possessing a local memory and a register file**
- ❑ **A new op-code is associated with the RFUs;**
- ❑ **Reconfiguration necessities are evaluated during compile-time**
- ❑ **The compiler issues a SW Interrupt when a reconfiguration is needed**

Identification of Candidate op-codes

- The application code is translated into an intermediate **Data Acyclic Graph** associated with the **Control Flow Graph**.



- **Compilation and Profiling** has been carried out with **SUIF**

Basic Blocks oriented analysis

- ❑ **Search for interesting segments of computation within Basic Blocks**
 - the segment of the computation must have one entry point (one starting address)
 - one exit point (one ending address) and the update of PC would occur at different instant of time
 - one constant latency
- ❑ **We look for subgraphs that are**
 - suitable to collapse into a single instructions
 - easily identifiable (acceptable complexity due to the presence of isomorphism)
- ❑ **Multiple Input Single Output MISO subgraphs**

MISOs and Max-MISOs

- ❑ Search for Max-MISOs can be carried out with a complexity linear in the number of nodes in the graph
- ❑ Each Max-MISO must be checked for feasibility (e.g., complexity)
- ❑ EMISOs: when possible loops can also collapse and become an opcode
 - This reduces inter-traffic
- ❑ The variables which can be “hidden” within an EMISO are local variables, e.g., whose use is limited to that loop
- ❑ Only one variable is allowed not to be local

SW vs. FPGA Implementation

- ❑ Dynamic information are obtained through profiling
- ❑ Static information (e.g., area, speed-up) can be obtained by synthesis tools
- ❑ An estimate of the number of CPU cycles saved is

$$\Delta T = (T_{SW} - T_{HW}) \cdot OCC_{st} \cdot OCC_{dyn}$$

- T_{SW} and T_{HW} represent the estimated time for execution of the application-specific opcode in SW instructions on the core-processor or in HW (i.e. on the RFU as a single operation)
- occ_{st} : is the number of static recurrences (search tool)
- occ_{dyn} : is the number of dynamic occurrences (profiling)
- T_{SW} is bounded by the *critical path* of the subgraph
- T_{HW} comes from a *high-level synthesis* of the given subgraph, adopting a time-constrained approach

Optimising performance by Genetic Algorithms

❑ The goal is to cover the application graph with the candidate MISOs/EMISOs and select the best covering

❑ Definitions

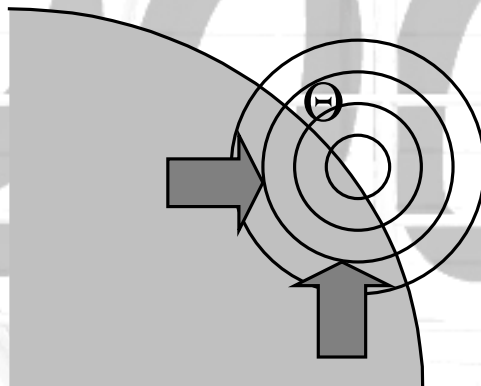
- The covering Θ set of a graph G (a procedure) is the set containing the native IS as well as the (E)MISOs associated with the graph (a procedure).
- A graph cover is a graph covered with disjoint elements of Θ

❑ the problem can be formalised as:

$$C^o(G) = \max_{C(G) \in \Phi} J(C(G))$$

Genetic Algorithms: Chromosomes and Population

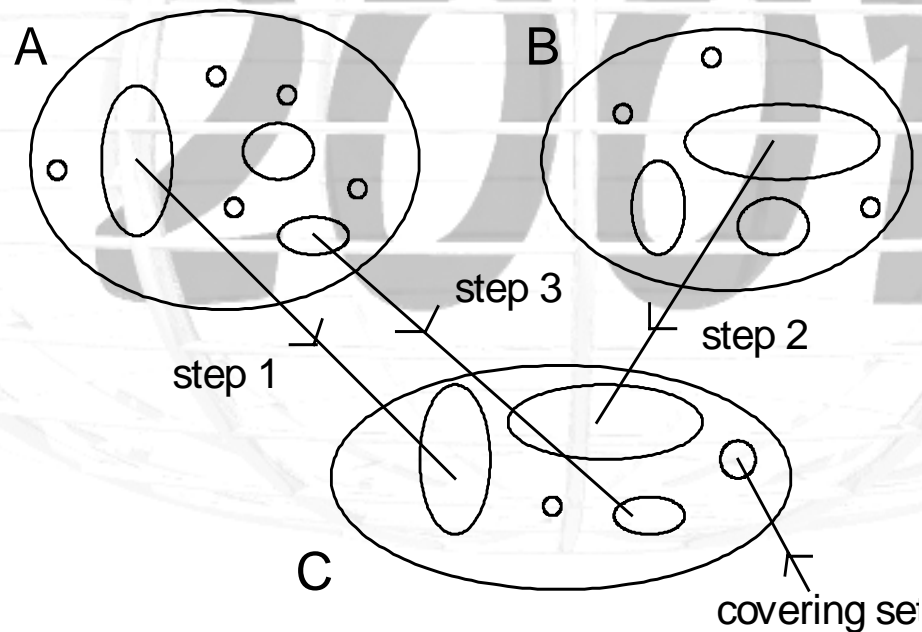
- ❑ The Chromosome is a graph covered with EMISOs and instructions
- ❑ A set of genetic operators have been considered
 - they must provide feasible offsprings



- ❑ The initial population is generated by randomly extracting (E)MISOs from

Selection and Crossover

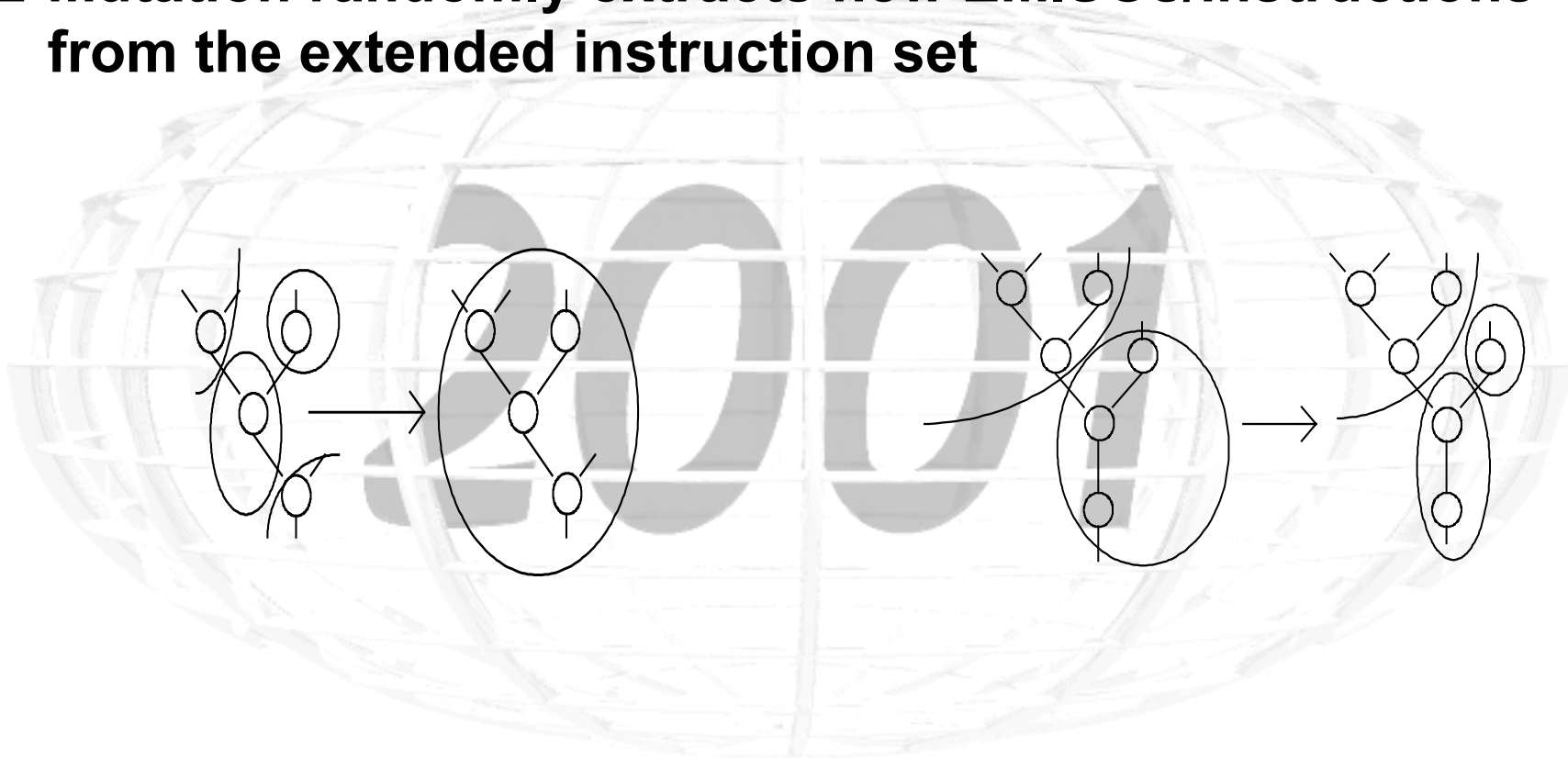
- ❑ Selection is carried out with the roulette strategy (most fitted elements are most likely to be selected)
- ❑ Crossover implements genes exchange



- ❑ feasibility is a crucial issue

Mutation

- ❑ Mutation randomly extracts new EMISOs/instructions from the extended instruction set



- ❑ Growing/reduction aspect
- ❑ Elitism

Experimental Results

❑ Applications

- cryptography - DES
- image processing - SVD
- function optimisation - BFGS

❑ Profiling provided the MaxEmisos/Emisos

Application	MAXMISO count	Max number of nodes	Static recurrence	MAXEMISO count	Max number of nodes	Static recurrence
Cryptography	122	10	32	2	2	1
SVD	524	8	2	241	12	6
BFGS	13	7	1	6	5	3
JPEG	2673	15	4	43	8	7

Experimental Results

- ❑ Estimate of the speed-up gain without area requirements

Application	T for core processor	ΔT_{tot} (8 MISO)	Speedup (8 MISO)	ΔT_{tot} (16 MISO)	Speedup (16 MISO)	ΔT_{tot} (32 MISO)	Speedup (32 MISO)
Jpeg	5,295,127	618,552	1.132	1,098,348	1,261	1,377,722	1.351
Cryptography	476,865	61,228	1.147	84,625	1.213	152,115	1.468

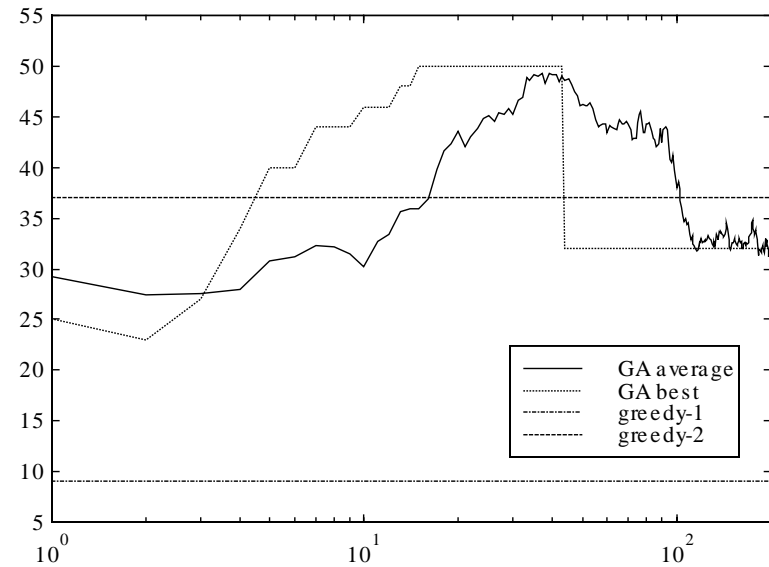
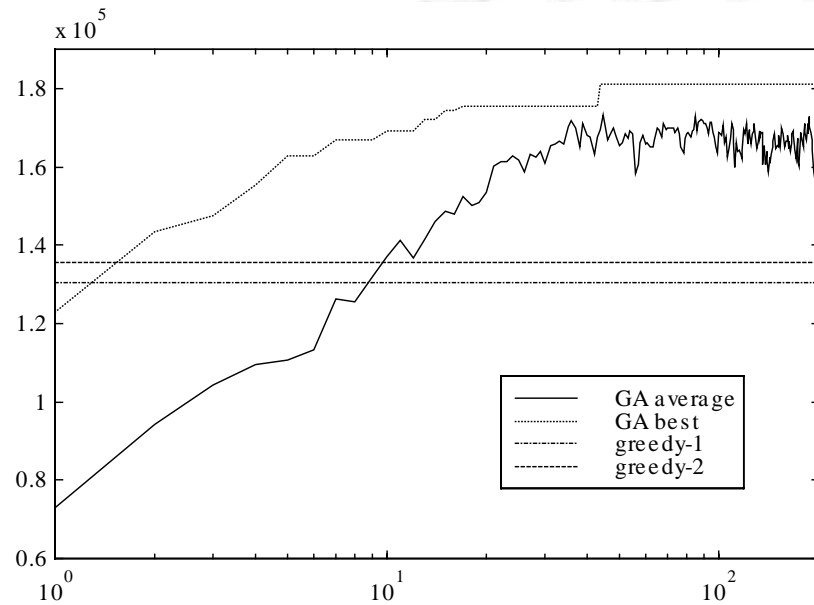
- ❑ We consider that the FPGA can accommodate 8, 16 and 32 max(E)MISOs
- ❑ The simulation was carried out with the Play Doh architecture and the Trimaran compiler and simulator
- ❑ the gain is in the number of cycles

Genetic search: area constraints

- ❑ **Greedy:**
 - **Fitting with the most effective Emisos**
 - **covering elements are randomly extracted**
- ❑ **benchmark the inverse DCT**
- ❑ **57 emisos found (mostly with 7 and 8 basic instructions) most of them executed more than 6800 time on a 227x149 image**
- ❑ **Crossover 0.6**
- ❑ **Mutation 0.001**
- ❑ **Population size 100 elements**
- ❑ **Evolution: 200 epochs**

Genetic search

area fixed to 50 eua



a) the fitness evolution over epochs, b) the FPGA area use

Conclusions

- ❑ We have presented a methodology for the automatic design of an extended Instruction Set, exploiting reconfigurable logic
- ❑ Extensions to EMISOs can be performed so that a whole loop body can be included in the FPGA section
- ❑ A genetic search has been considered and dedicated covering operators developed