POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA

DRuiD: Designing Reconfigurable Architectures with Decision-making Support

ASP-DAC 2014

V.-M. Sima, R. Meeuws, K. Bertels

TU Delft, Delft (NL)

G. Palermo, C. Silvano, V. Zaccaria

Politecnico di Milano, Milano (IT)

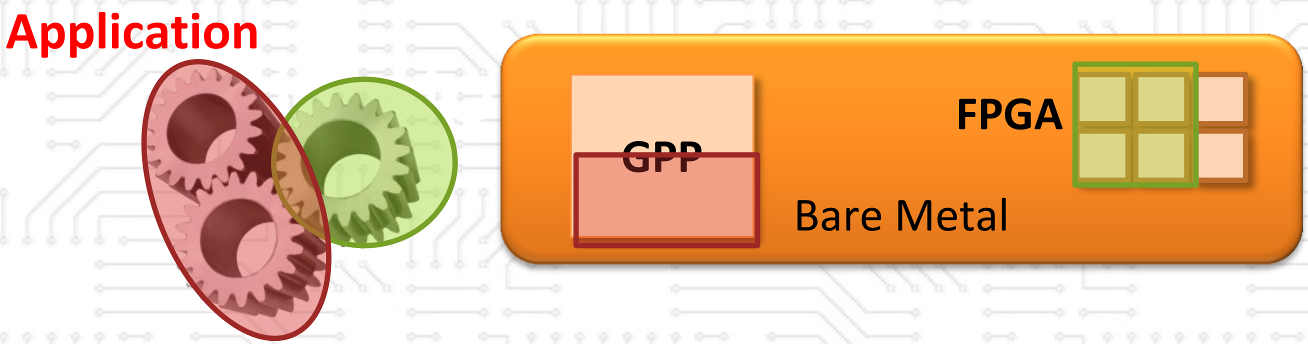
Giovanni Mariani ***giovannisiro.mariani@polimi.it***

ALaRI – University of Lugano, Lugano (CH); Politecnico di Milano

Table of Contents

Introduction and Motivation

- **Heterogeneous and reconfigurable architectures:**
  - **Coupling General Purpose Processors (GPPs)** with **reconfigurable hardware**
  - Accelerate computational intensive kernels using HW



- The considered baseline design approach:
  - An application implementation is available in SW
  - Some kernels shall be ported to HW

Automatic Mapping Approach

- Conventional **partitioning** and **mapping** approaches in literature:
  - Kernels have one or more costs (e.g. execution time and area)
  - Assume kernel costs are known
  - Map to minimize the costs and fit in the constraints

SW ex. time	HW ex. time	Area	Kernel id
2	1	2	K1
4	4	2	K2
6	4	4	K3

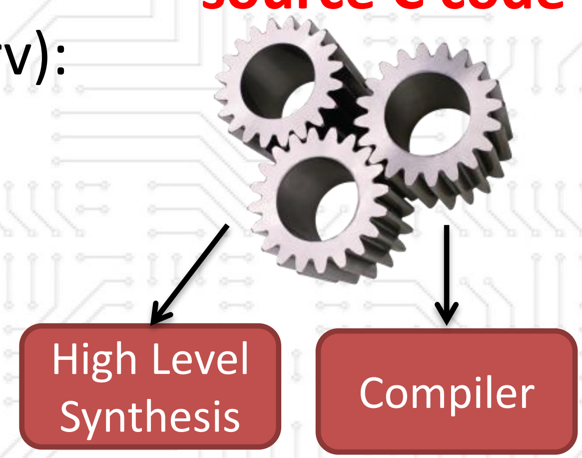
Automatic Mapping Approach

- Are kernel costs known?

SW ex. time	HW ex. time	Area	Kernel id
2	1	2	K1
4	4	2	K2
6	4	4	K3

- Considering the application is available in SW
- The kernels must first be ported to HW
  - High level synthesis tools help (e.g. dwarv):
    - Restrictions on the C code
    - Cleanup the code
    - Some manual optimization
  - Time consuming
  - Error prone

Application source C code



Practical Mapping Approach

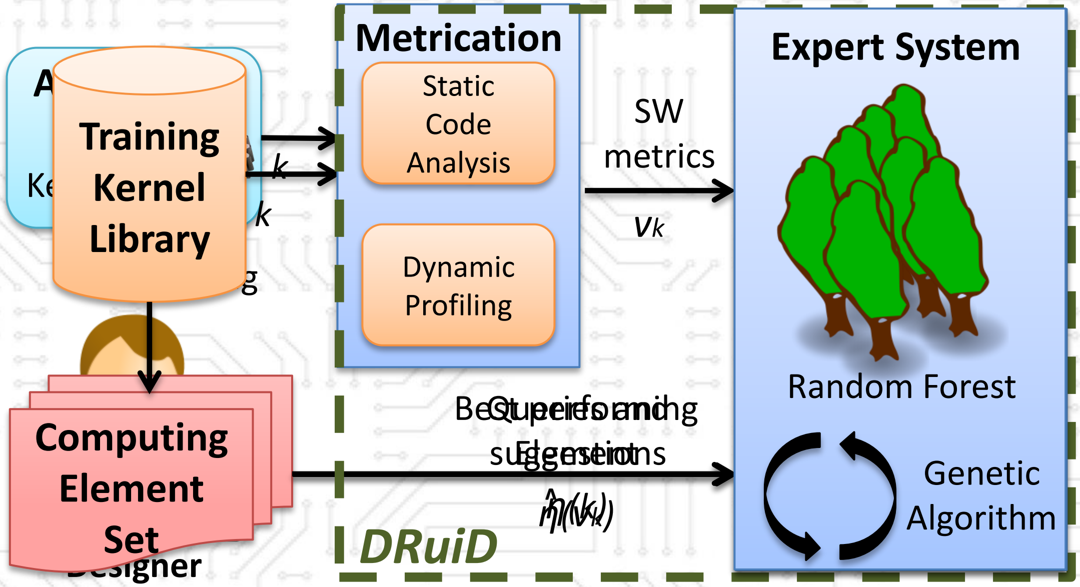
SW ex. time	HW ex. time	Area	Kernel id
2	1	2	K1
4	4	2	K2
6	4	4	K3

- When porting an application to a heterogeneous and reconfigurable platform:
  - Analyze the application:
    - Which kernel is expensive in SW?
    - Which kernel is suitable for HW?
      - Computational intensive?
      - Inherently sequential?

Application source C code

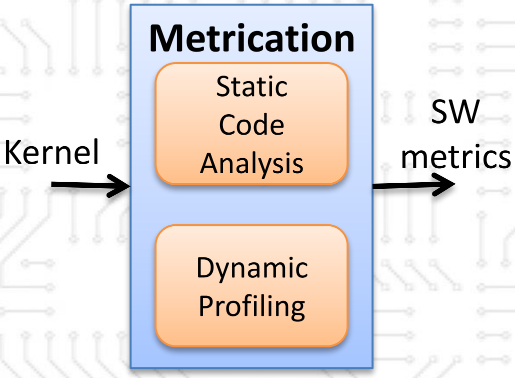
Decision Making Support

- Given the application source C code
- An expert system
  - Given a parametric representation of a kernel
  - Classifies the kernel
  - Learning the behavior of a set of known training kernels



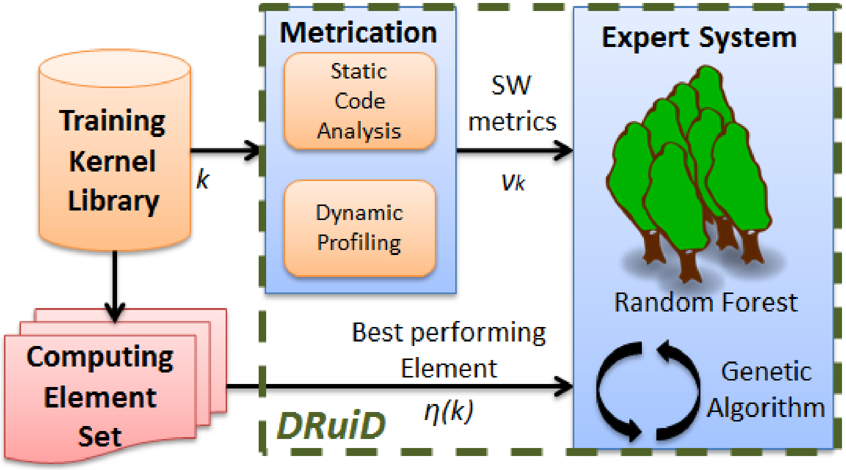
Kernel Representation

- Static Code Analysis (SCA)
  - Instruction types
  - Control flow
  - Data flow
  - Variable scope and location



- Pin based dynamic profiling, Microarchitectural Independent Characterization (MICA)
  - Instruction types
  - Instruction level parallelism
  - Memory and register access pattern
  - Branch predictability

Training Overview



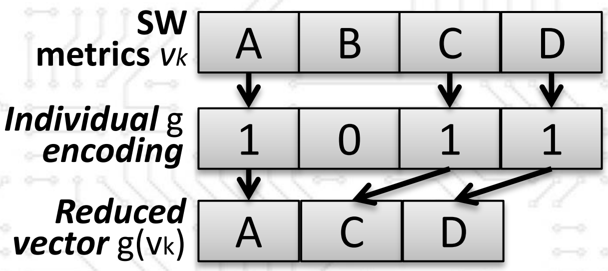
- *Random Forest* (RF):
  - Many *decision trees* are trained
  - More robust than a single *decision tree*
- Reduce the SW metrics by using a *Genetic Algorithm* (GA)

SW metrics

Source	Group name and acronym	Number of metrics
SCA	Instruction type (IT)	45
	Control flow (CF)	18
	Data flow (DF)	15
	Variables scope and location (VSL)	14
MICA	Instruction type (IT)	12
	Instruction level parallelism (ILP)	4
	Memory and register access pattern (MRAP)	66
	Branch predictability (BP)	15
Total		189

Training (GA + RF)

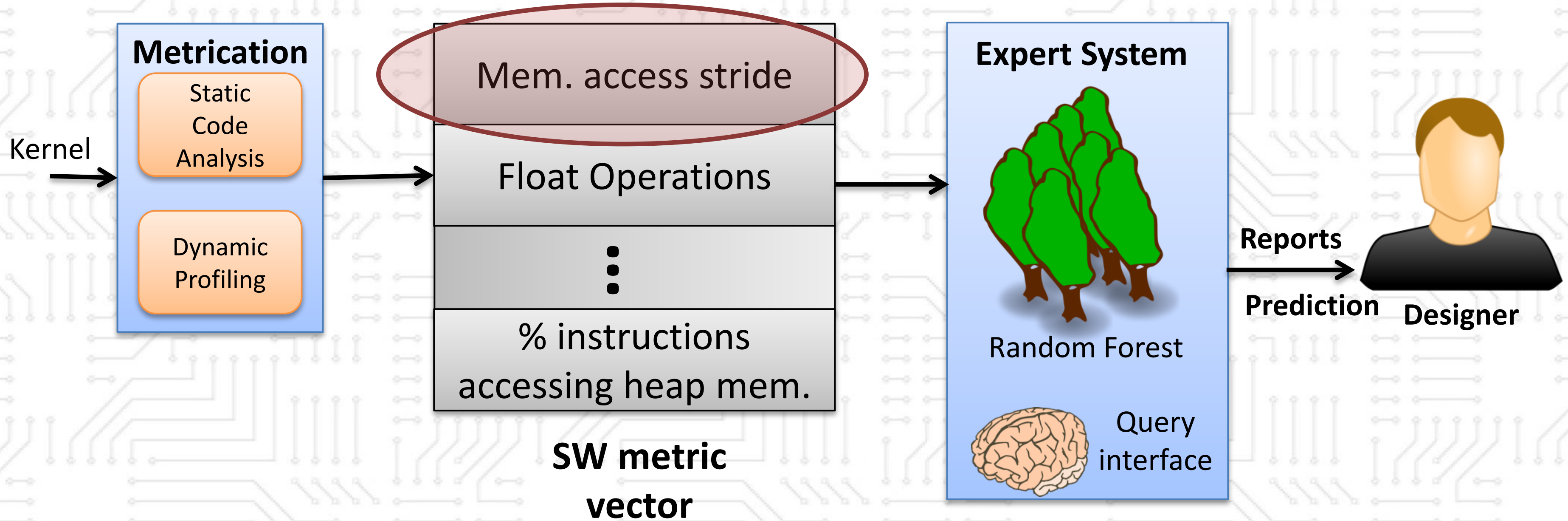
- A GA individual identifies the SW metrics to use



- A RF is trained on the reduced SW metric vector
- GA evolves a population of RFs to:
  - Maximize RF accuracy
  - Minimize the number of metrics

Prediction and Query

- A RF is a collection of *decision trees*
- For a given kernel, each tree votes either for FPGA or for GPP mapping
- Majority voting is used
- How modifications on the SW metrics impact on votes?



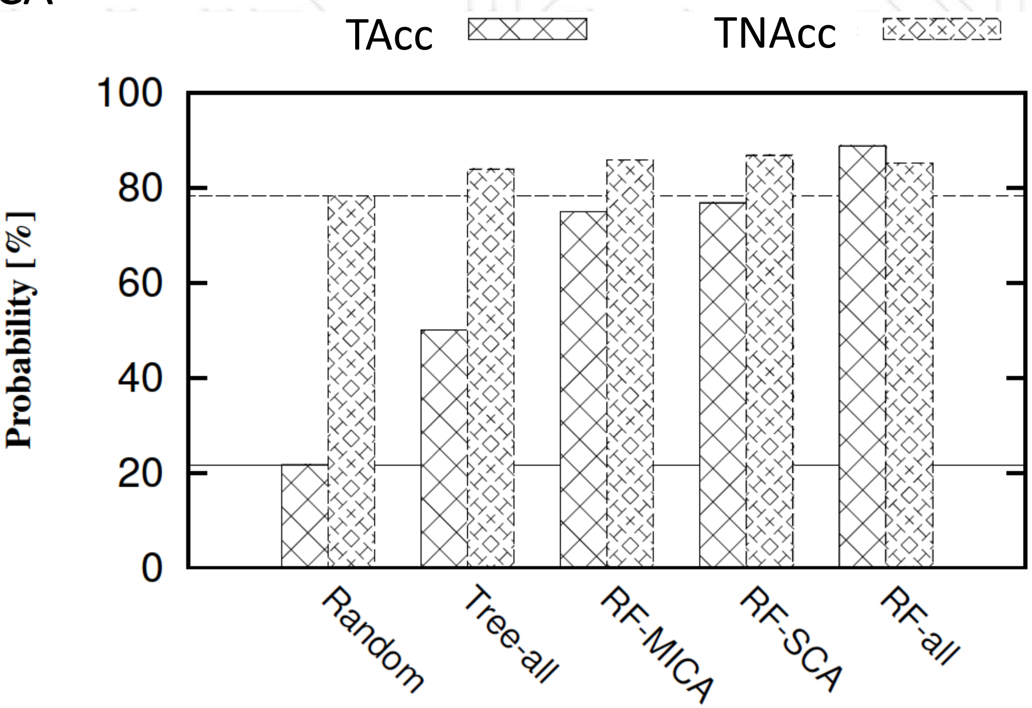
Experimental set up

- Target architecture is a **Xilinx ml510** board
- The GPP is a **PowerPC**
- The reconfigurable HW is a **Virtex5 FPGA**
- 97 kernels taken from different sources (MediaBench, MiBench, CHStone, …)
- Delft workbench toolsets:
  - C to VHDL using *dwarv*
  - FPGA logic area estimation from C using *Quipu*

Model Accuracy

- Probability of a kernel predicted as **FPGA-accelerated** to be truly accelerated (TAcc)
- Probability of a kernel predicted as **NON FPGA-accelerated** to be truly non-accelerated (TNAcc)
- Comparing different expert systems:
  - Random
  - Decision tree using SCA & MICA
  - RF using MICA metrics
  - RF using SCA metrics
  - RF using SCA and MICA

Crossvalidated Prediction Accuracy



Case Study

- Susan edge application from automotive MiBench
- RF suggests not to use the FPGA due to non local memory

```
susan_principle(unsigned char *in,int *r,char *bp,int max_no,int x_size,int y_size){
  int   i, j, n,x;
  unsigned char *cp;
  unsigned char cU[3],cC[3],cD[3];// Cache registers
  for (i=1;i<y_size-1;i++){
    for(x=0;x<3;x++){// Cache data into registers
      cU[x] = in[(i-1) * x_size + x];
      cC[x] = in[(i  ) * x_size + x];
      cD[x] = in[(i+1) * x_size + x];
    }
    for (j=1;j<x_size-1;j++){
      n=100;
      cp=bp + cC[1];   // Register read center

      n+=*(cp-cU[0]);  // Register read left
      n+=*(cp-cU[1]);  // Register read up
      n+=*(cp-cU[2]);  // Register read up right

      n+=*(cp-cC[0]);  // Register read left
      n+=*(cp-cC[2]);  // Register read right

      n+=*(cp-cD[0]);  // Register read down left
      n+=*(cp-cD[1]);  // Register read down
      n+=*(cp-cD[2]);  // Register read down right

      // Shift registers
      cU[0] = cU[1];cC[0] = cC[1];cD[0] = cD[1];
      cU[1] = cU[2];cC[1] = cC[2];cD[1] = cD[2];
      cU[2] = in[(i-1) * x_size + j+2];// Mem read next up
      cC[2] = in[(i  ) * x_size + j+2];// Mem read next
      cD[2] = in[(i+1) * x_size + j+2];// Mem read next down

      if (n<=max_no)
        r[i*x_size+j] = 730 - n;
    }
  }
}
```

Original version (3.96x speedup)

Conclusions

- We proposed a **decision making support for the design of heterogeneous platforms**
- A **Random forest classification** system is capable to learn what are the characteristics that make a kernel suitable for different computational elements
- Given an architecture composed of a PPC and a Virtex5 FPGA, the proposed methodology identifies the best computational element the 90% of the times

Thank You

Questions?