Application Driven Optimization of VLIW Architectures: a Hardware–Software Approach

Alberto Ferrante

DTI,

University of Milan

E-mail: ferrante@dti.unimi.it

Giuseppe Piscopo, Stefano Scaldaferri

ALaRI Institute,

University of Lugano

E-mail: {piscopo, scaldaferri}@a.alari.ch

Presentation Outline

1. Motivations and Goals;
2. Optimization Steps;
3. Case Study;
4. Conclusions and Future Work.

1/4. Motivations and Goals

- We want to optimize an architecture for a specific task:
  - a detailed description of a methodology is lacking;
  - it is well known in industrial environment;
  - goal is to provide a *step-by-step* description through a case study.

2/4. Main Optimization Steps (1)

- Preliminary Analysis:
  - profiling;
  - preliminary simulations;
- design space and cost function definition;
- simulations and study of the results obtained.

2/4. Main Optimization Steps (2)

- Addictional steps:
  - code tuning for the optimized architecture;
  - custom instructions.

3/4. Case Study

- We considered the *Imaging Pipeline* benchmark:
  - provided by HP Labs;
  - main operations:
    - decompression of a *jpeg* image into a *RGB* one;
    - resizing of *RGB* image;
    - color-space conversion from *RGB* to *CMYK* ;
    - dithering to obtain a half-toned image.

3/4. C.S.: Reference System (1)

- VLIW architecture;
- VEX toolset:
  - developed by HP Labs;
  - compiler simulator derived from the *Multiflow C compiler*;
  - clustered VLIW processors, HP Lx ISA.

3/4. C.S.: Reference System (2)

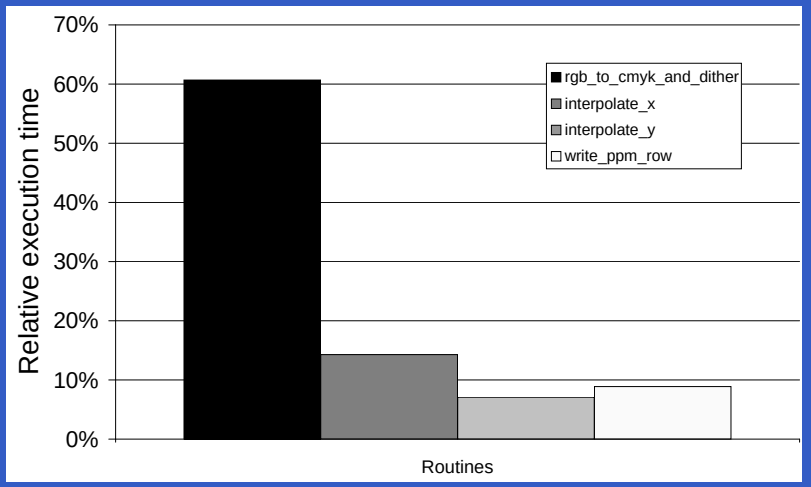
- *Architectural parameters*:
  - visible to the compiler;
  - num. of clusters, issue width, data cache ports, num. of reg. files, gen. purpose/branch regs, int ALUs/MULs.

3/4. C.S.: Reference System (3)

- *Microarchitectural parameters*:
  - not visible to the compiler;
  - Dcache and Icache size, associativity, line size, miss penalty.

3/4. C.S.: Preliminary Analysis (1)

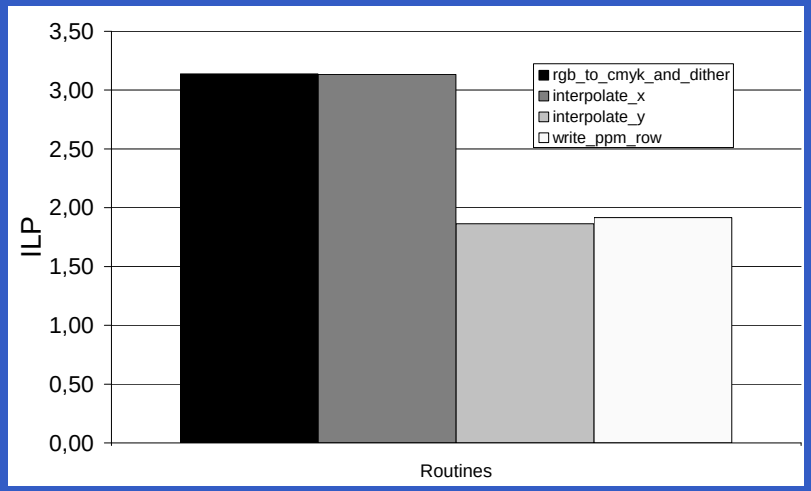
Dynamic profiling:

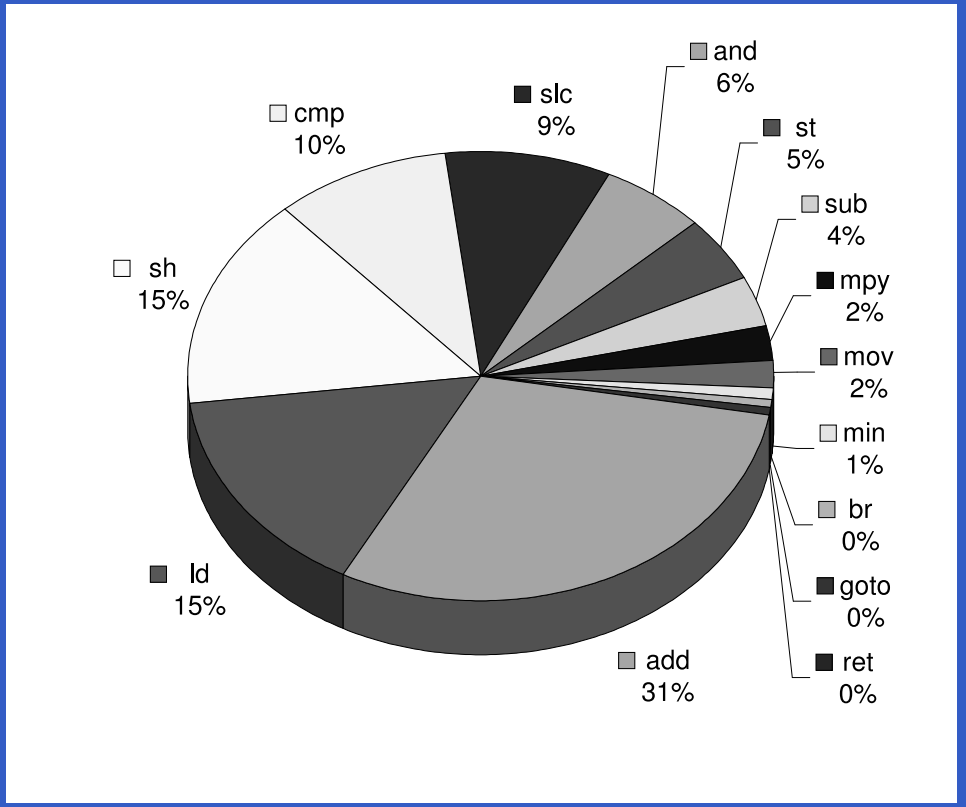


we focus on *rgb_to_cmyk_and_dither*

University of Lugano
ALaRI
DTI

3/4. C.S.: Preliminary Analysis (2)





3/4. C.S.: Design Space Definition (1)

Preliminary simulations: relevant parameters?
Architectural configurations

Parameter	Range	Combin.
Issue Width (= n)	4, 8, 16, 32	4
# clusters	1, 2, 4	3
ALU per cluster	n	1
MemLoad (= p)	n/2, n	2
MemStore	p	1
MemPorts per cluster	p	1
Multipliers per cluster	2	1
Total Architectural Combinations		24

3/4. C.S.: Design Space Definition (2)

Microarchitectural configurations

Parameter	Explored Range	Combin.
DCache Size (kbyte)	32, 64, 128	3
DCache Associativity	4, 8	2
DCache Line Size (bit)	32, 64	2
ICache Size (kbyte)	32, 64, 128	3
ICache Associativity	1	1
ICache Line Size (bit)	64	1
Total Microarchitectural Combinations		36

Total: 864 configurations

3/4. C.S.: Design Space Definition (3)

Default architecture

Machine Model Parameters	
Number of Clusters	1
Issue Width	4
Data Cache Ports	1
Register Files	2
General-Purpose Registers (32 bit)	64
Branch Registers (1 bit)	8
Integer ALU	4
Integer Multiplier	2

Latencies of Main Operations (cycles)	
Load	2
Store	0
ALU	0
MPY	1

3/4. C.S.: Cost Function

$$C(A,T) = \frac{A}{A_{def}} \times \frac{T}{T_{def}}$$

- A: area estimation;
- A_{def}: area estimation of default architecture;
- T: simulated time (in clock cycles);
- T_{def}: simulated time for the default architecture.

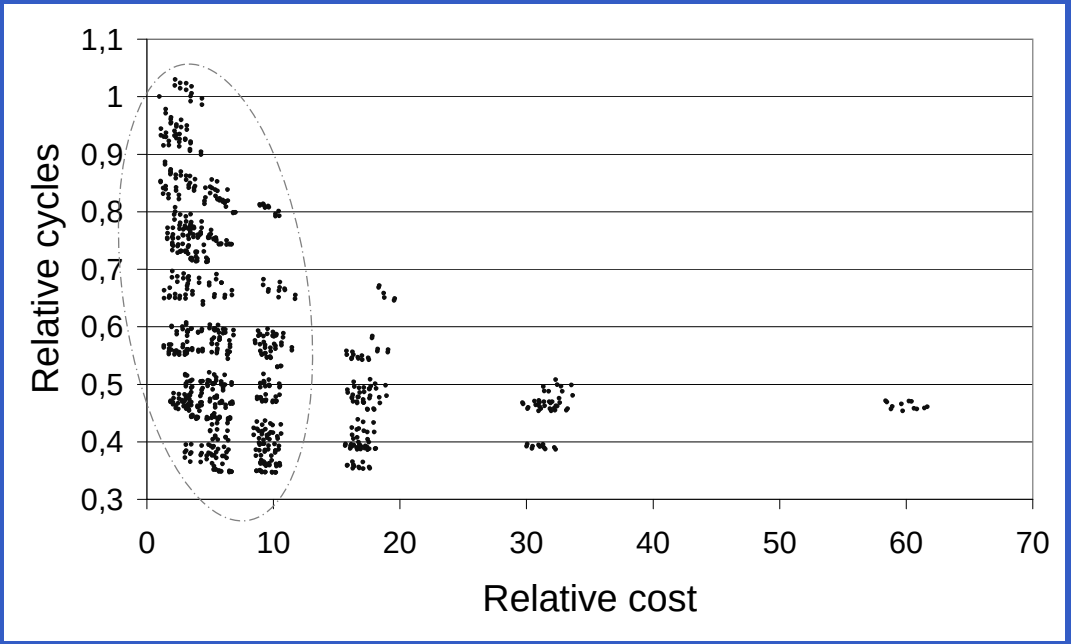
3/4. C.S.: Area Estimation

$$A = A_{mem} + N_{cluster}(A_{reg} + N_{alu}A_{alu} + N_{mul}A_{mul})$$

- measured in *register bit equivalent area* (rbe);
- related to the underlying technology.

3/4. C.S.: Design Space Exploration (1)

Results of the simulations:

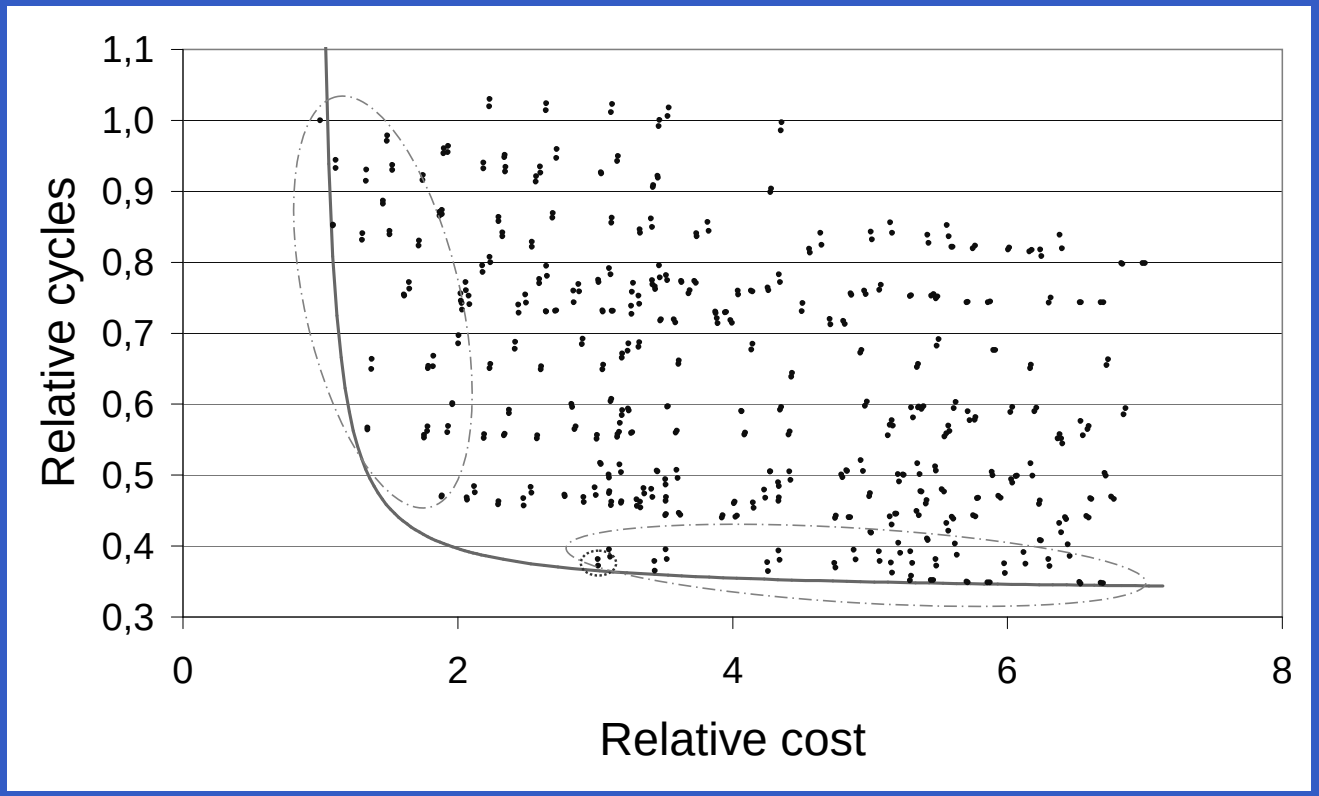


Overall simulation time: 8 hours.

University of Lugano
ALaRI
DTI

3/4. C.S.: Design Space Exploration (2)

Zoomed version of the DSE results:



3/4. C.S.: Optimal Point

- Previous analysis allowed to identify an optimal point among the explored ones;
- benchmark runs almost 3 times faster than on the base architecture;
- cost is 3 times higher than base configuration.

3/4. C.S.: Code Optimization

- to improve performance with the optimal architecture;
- original code is already highly optimized;
- providing compiler directives for selection loop unrolling of 16 furtherly improves performance by 5%.

3/4. C.S.: Custom Instructions

- Custom instructions have been introduced for the most time consuming parts ot the code:
  - 4 instructions like:

```
int dx_c=(ex_c*7+ey_c*5+eyp_c*3+eym_c)/16;
```

  - were transformed into:

```
int dx_c=(_DITH75(ex_c,ey_c)+_DITH31(eyp_c,eym_c))/16;
```

- performance are improved by another 6%.

4/4. Conclusions

- Through an example, we have shown a methodology allowing to optimize an architecture for a specific task;
- we have obtained an optimal performance–area tradeoff point in a reasonable amount of time:
  - different tradeoffs can be optimized, by changing the cost function.

4/4. Future Work

- Study of a model for evaluating the influence of architectural changes on the clock cycle time is ongoing.