Industrial Applications of Evolutionary Algorithms

Giovanni Squillero
Politecnico di Torino
Torino, ITALY
giovanni.squillero@polito.it

http://www.sigevo.org/gecco-2013/

Copyright is held by the author/owner(s).
GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

Presenter

- **Giovanni Squillero** received his M.S. and Ph.D. degrees in computer science engineering, respectively, in 1996 and 2000, from Politecnico di Torino, Torino, Italy, and is presently an Assistant Professor at the same institution. The only recurring theme in his research activities is the exploitation of bio-inspired techniques for tackling *real* – or at least *realistic* – problems, usually in direct collaboration with industries. Squillero is the author of 3 books (one didactic); 15 journal articles; 9 book chapters and about 100 papers in conference proceedings. Since 2007, he co-organizes the *Workshop on Hardware Optimization Techniques* (EvoHOT) at EvoSTAR.
- **Acknowledgements:** these works have been realized thanks to the contribution of Stefano Di Carlo, Stefano Gandini, Marco Gaudesi, Danilo Ravotto, Walter Ruzzarin, Ernesto Sanchez, Alberto Scionti, Alberto Tonda.

Agenda

- Evolutionary algorithms
- Industrial problems
- Case studies
  - Drift correction in electronic noses
  - Cell phone software validation
  - Assembly program generation

Evolutionary Algorithms

Natural Evolution

"the great effect produced by the accumulation in one direction, during successive generations, of differences absolutely inappreciable by an uneducated eye"

— Charles Robert Darwin

Natural & Artificial Evolution

- Not a random process
- Sequence of different steps
  - Some mostly random (e.g., variation)
  - Some mostly deterministic (e.g., selection)

Natural vs. Artificial Evolution

- Natural Evolution
  - Does not have a goal
  - Is not an optimization process
  - Does not favor intelligence
- Artificial Evolution
  - Has a goal
  - Is exploited as an optimizer
  - Does create intelligent behavior?

Evolutionary Algorithms

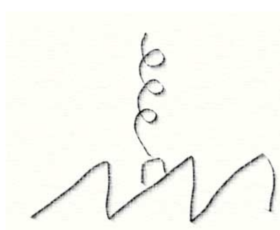
- EA vs. Local search
  - Step size, neighborhood
- Population
- Genetic operators
  - Inheritance
  - Single parent (mutation)
  - Recombination (crossover)

Historical perspective

- 1960s
  - Evolutionary programming (EP)
  - Evolution strategies (ES)
  - Genetic algorithm (GA)
- 1980s
  - Genetic programming (GP)

Why they works?

- Mutation, inheritance
  - "Small variation"
  - Locality
- Recombination, Xover
  - Individual encoding
- Non-Disruptive genetic operators
  - Operator design
  - Encoding



Industrial Problems

Industrial problems

- Motivation
- Industrial vs. toy problems

Case studies

- ❖ Drift correction in electronic noses
- ❖ Cell phone software validation
- ❖ Assembly program generation

Drift correction

- ❖ Creating an adaptive system

Underlying ideas

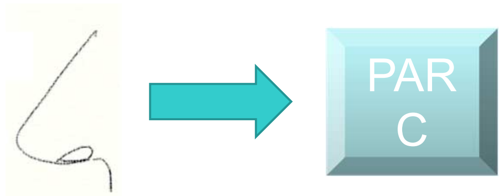
- ❖ Evolution and adaptation
  - Fogel, Holland, …
- ❖ On-line vs. off-line evolution
- ❖ Adaptive system
  - Embedding an EA

Artificial Olfaction

- ❖ Mimic human olfaction
- ❖ Arrays of gas chemical sensors
- ❖ Metal oxide semiconductors
  - Contact with volatile compounds
  - Adsorption changes sensor’s electrical properties
- ❖ Each sensor is sensitive to all volatile molecules in a specific way

Artificial Olfaction

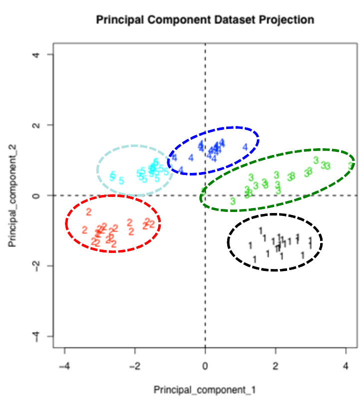
- Digital fingerprint
- Pattern recognition (PARC) techniques



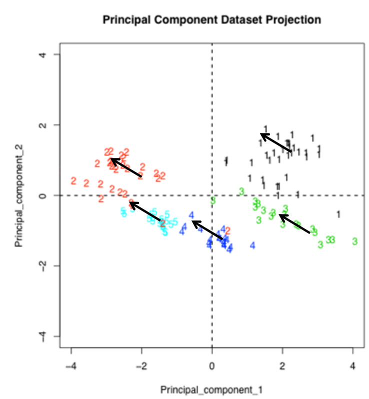
Artificial Olfaction

- Changes in sensors sensitivity and baseline
  - Irreversible phenomena (e.g., poisoning & aging)
  - Environmental factors (e.g., humidity variations)
  - Thermo-mechanical degradation
  - Unknown dynamic processes

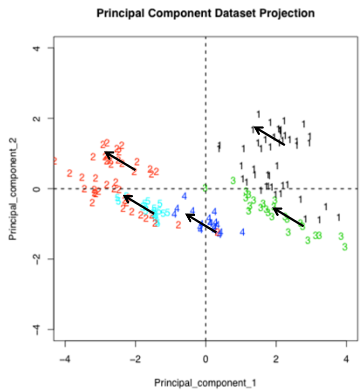
Pattern recognition



Drift effect

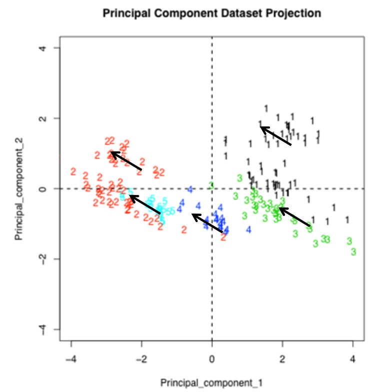


Drift effect



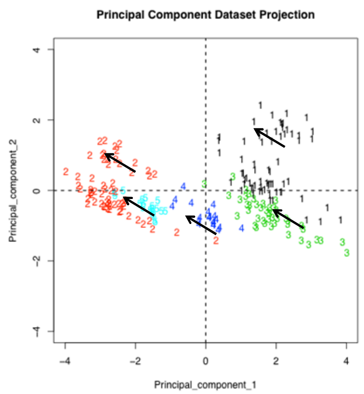
giovanni.squillero@polito.it 21

Drift effect



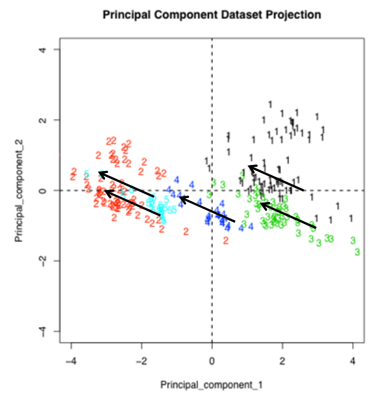
giovanni.squillero@polito.it 22

Drift effect



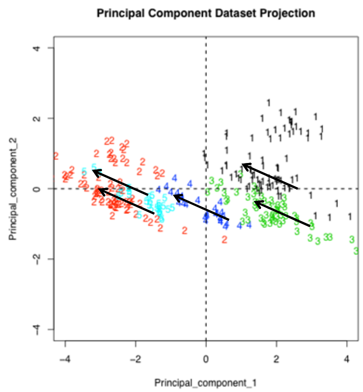
giovanni.squillero@polito.it 23

Drift effect



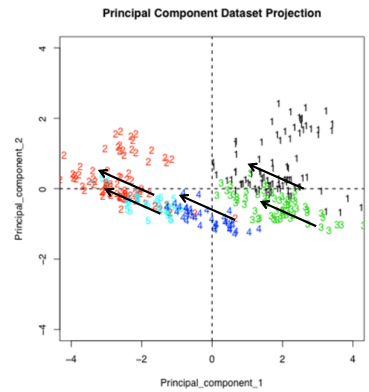
giovanni.squillero@polito.it 24

Drift effect



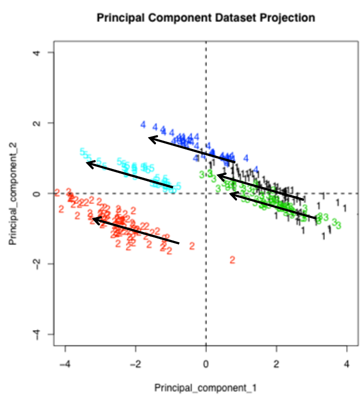
giovanni.squillero@polito.it

Drift effect



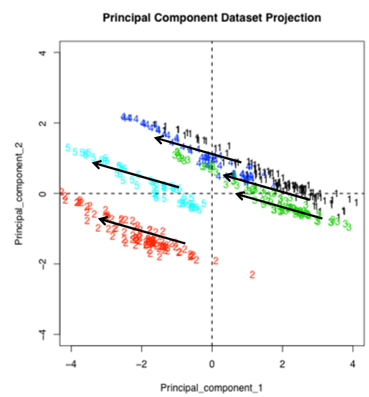
giovanni.squillero@polito.it

Drift effect



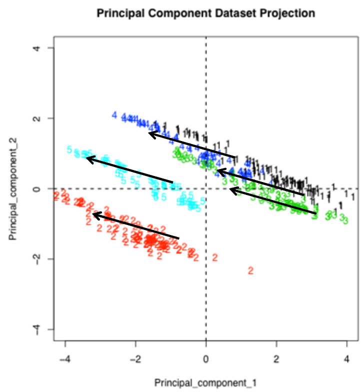
giovanni.squillero@polito.it

Drift effect



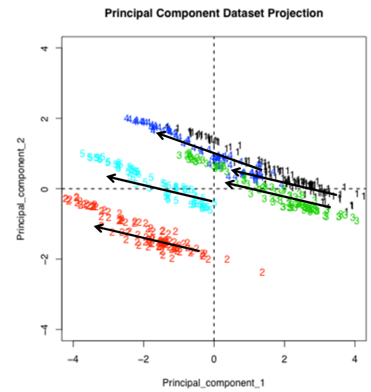
giovanni.squillero@polito.it

Drift effect



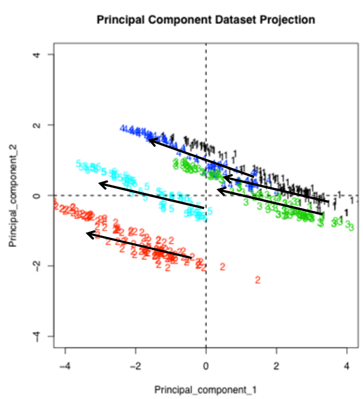
giovanni.squillero@polito.it

Drift effect



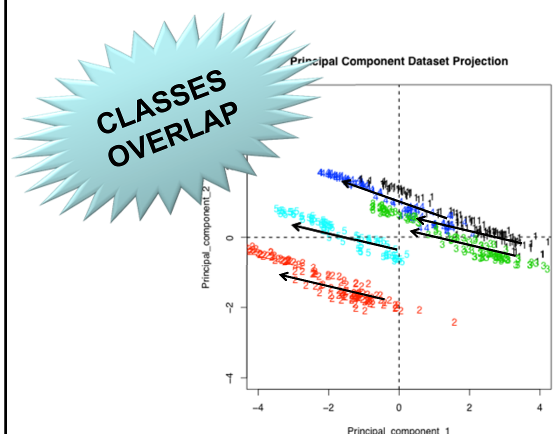
giovanni.squillero@polito.it

Drift effect



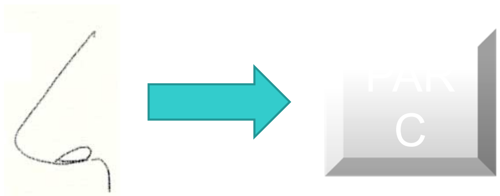
giovanni.squillero@polito.it

Drift effect



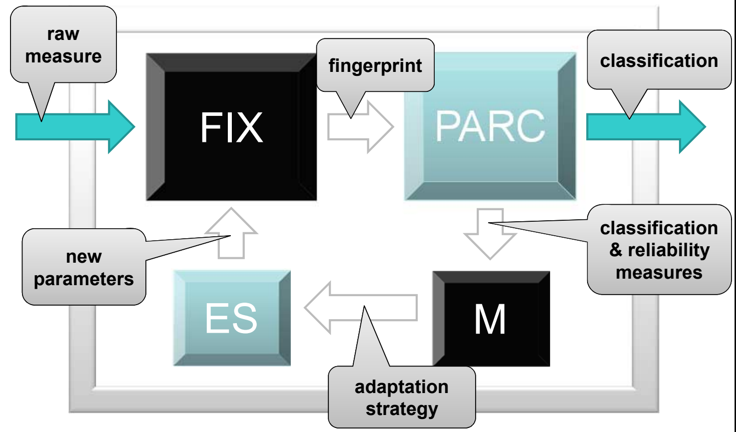
giovanni.squillero@polito.it

Drift-resistant Classifier



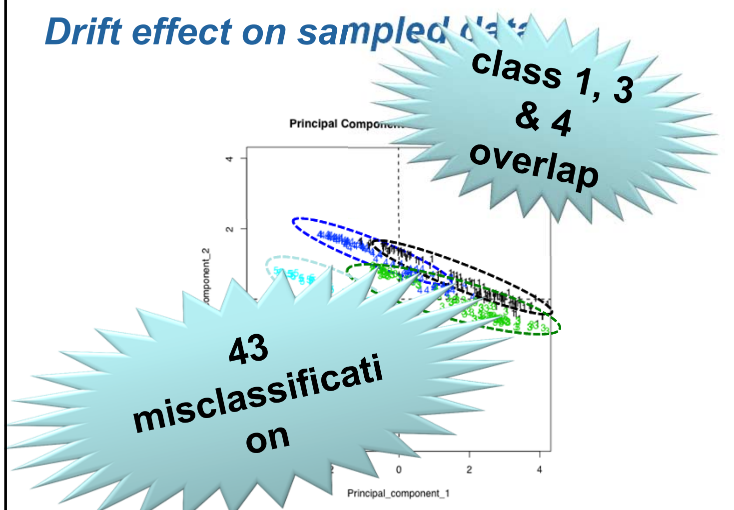
giovanni.squillero@polito.it 33

Drift-resistant Classifier



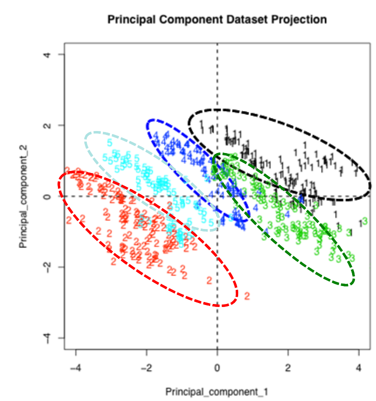
giovanni.squillero@polito.it 34

Drift effect on sampled data



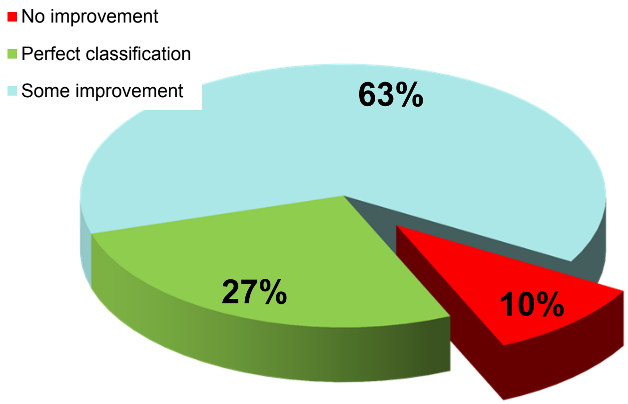
giovanni.squillero@polito.it 35

Drift compensation (2010)



giovanni.squillero@polito.it 36

First experimental results



Dissemination

- "Drift Correction Methods for gas Chemical Sensors in Artificial Olfaction Systems: Techniques and Challenges" (in Advances in Chemical Sensors)
- "Increasing pattern recognition accuracy for chemical sensing by evolutionary based drift compensation" (Pattern Recognition Letters)
- Conferences
  - International Bionic Engineering Conference
  - International Symposium on Olfaction and Electronic Nose

Software validation

- Finding the needle in the haystack

Underlying ideas

- Fitness singularities
- Mesas
- Differential survivals

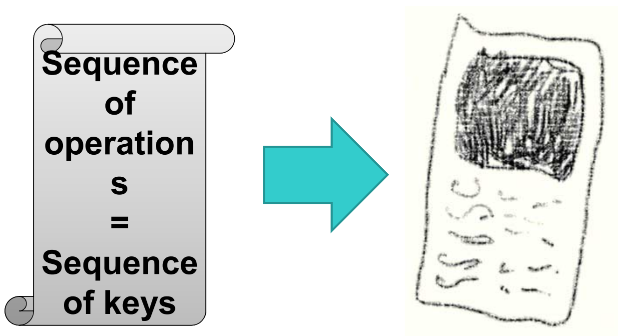
Background

- Complexity of cell phones
- Software applications
  - Different developers
  - Unreliable methodologies
- Time to market
- Power-related bugs
- Deep sleep

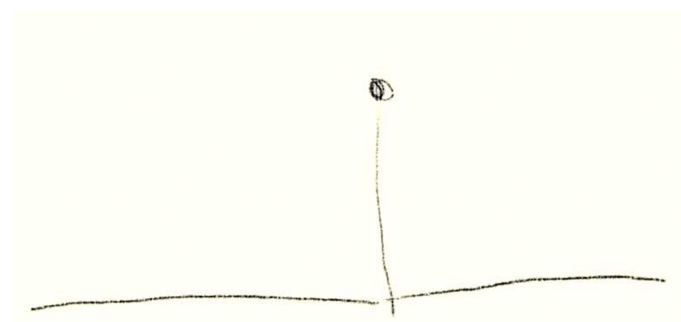
Goal

- Found a sequence of operations able to cause an incorrect behavior (excessive power consumption) in deep sleep

Individual



Fitness landscape



“Flat” fitness landscape

- Mesa
  - Differential survival?
  - Evolution?
- Spurious fitness
  - Favor exploration

Favor exploration

- Ad-hoc “battery”
- Measure power consumption
- Autocorrelation

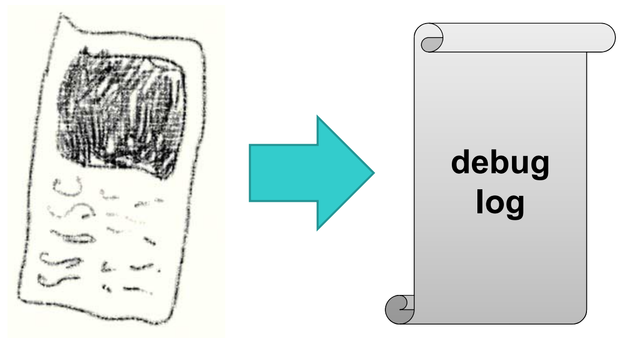
Favor exploration

- FSM modeling the whole system
  - Telephone + all applications
  - FSM state $\Rightarrow$ global system state
- Maximize state transition!

Problems

- The FSM is <u>not</u> known in advance
  - Fitness?
  - Current state?

FSM Definition



FSM Definition

- Extract debug log messages
- Parse (raw, heuristic)
- Define the state (fuzzy)

Space exploration

- Use autocorrelation
- Use FSM transition coverage
- Discover new states during the process
- Transition coverage $\Rightarrow$ Number of transitions

Results

- Two power-related bugs that escaped <u>all</u> previous checks
  - Video recording bug
  - Voice call bug
- One bug in the interface
  - Incorrect menu representation

Dissemination

❖ "A Framework for Automated Detection of Power-related Software Errors in Industrial Verification Processes" (*Journal of Electronic Testing*)

giovanni.squillero@polito.it

The lesson

❖ Exploration does work
- Exploit every information from the target system

❖ Do not assume your fitness to be perfect

giovanni.squillero@polito.it

Similar applications

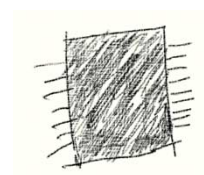
❖ Exploration
- Disprove equivalency between two implementations of a circuit 
- Crash a DNS server 

❖ FSM Approach
- CAD Problems (find a set of *control registers*) 

giovanni.squillero@polito.it

Assembly program generation

❖ Tackling complex individual structures



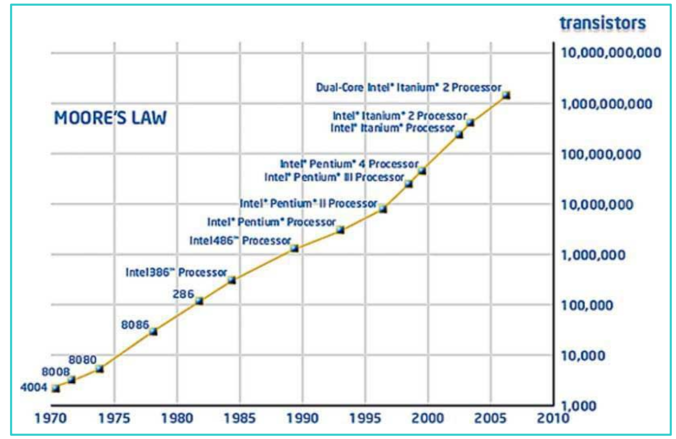
giovanni.squillero@polito.it

Underlying ideas

- ❖ Bit strings are elegant
  - … but complex structures may be useful
- ❖ Complex individual structure
  - Locality
  - Meaningful recombination
- ❖ Complex fitness evaluation

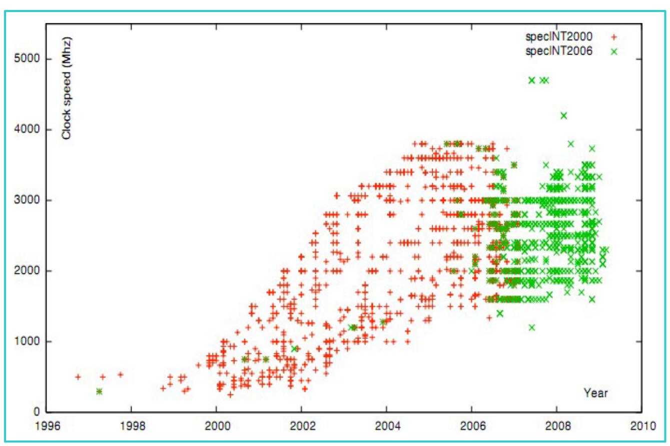
giovanni.squillero@polito.it

Background



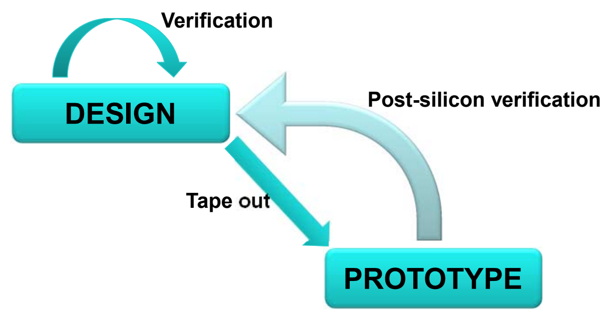
giovanni.squillero@polito.it

Background



giovanni.squillero@polito.it

Background



giovanni.squillero@polito.it

Post-silicon

- “Very few chips ever designed function or meet their performance goal the first time”
  - R. McLaughlin, S. Venkataraman, C. Lim (2009)

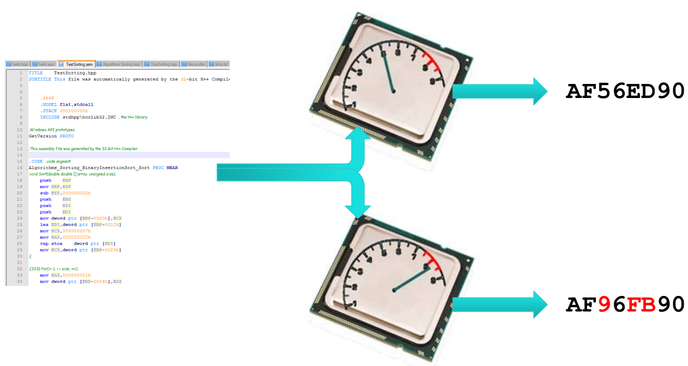
giovanni.squillero@polito.it 61

Post-silicon

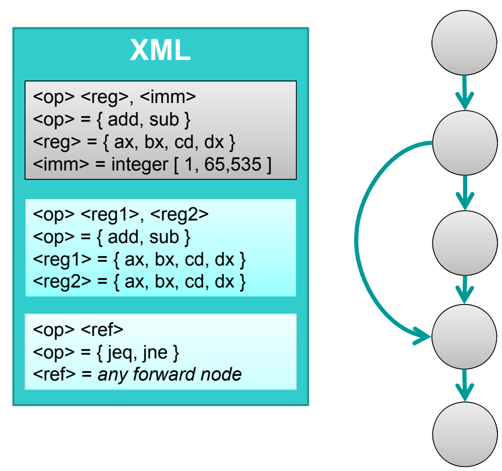
- Frequency pushes
- Speed-path
  - Pre-silicon critical path
  - Speed-path debug and analysis
- Failing test

giovanni.squillero@polito.it 62

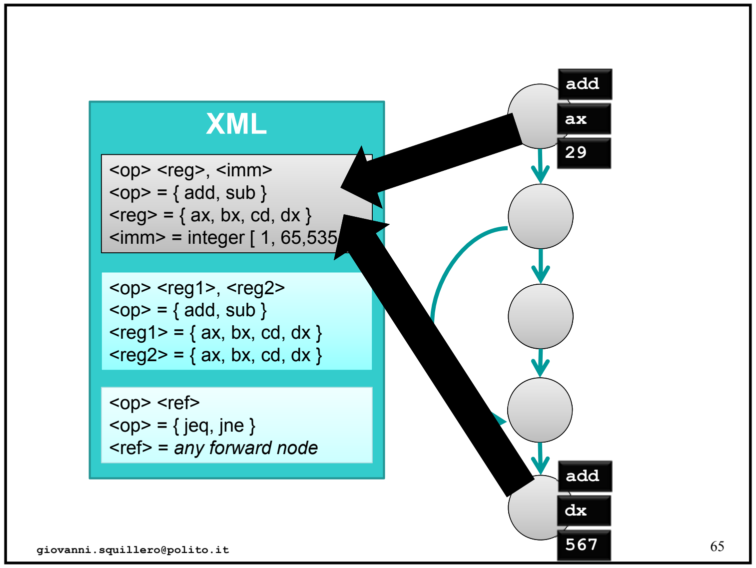
Functional failing test

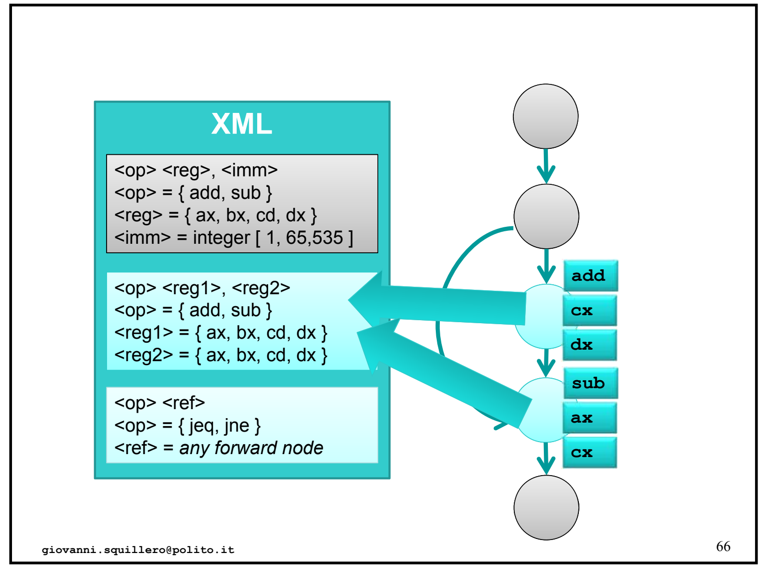


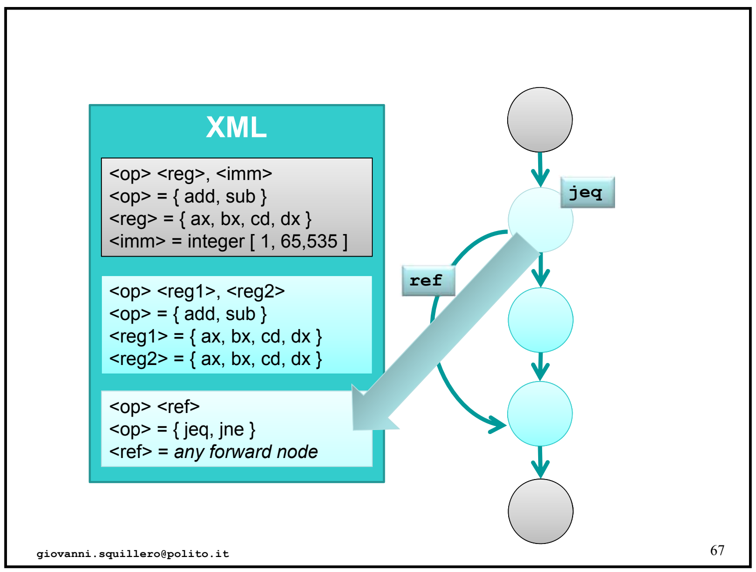
giovanni.squillero@polito.it 63

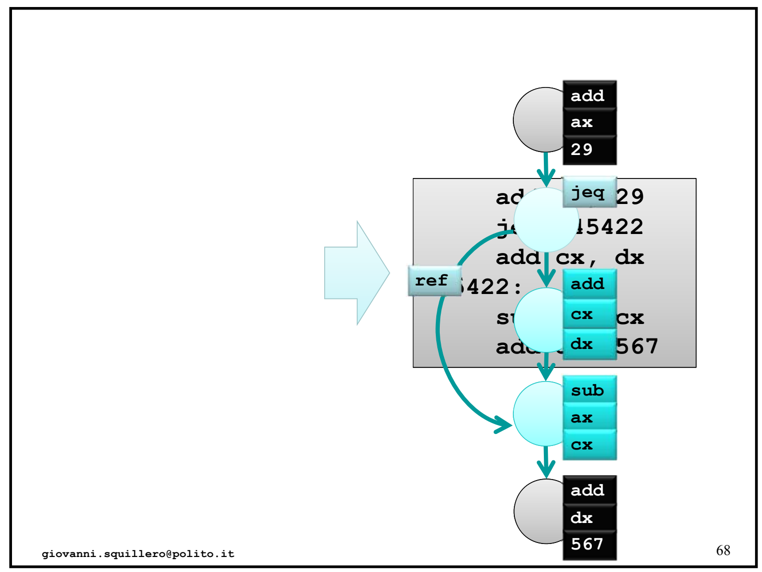


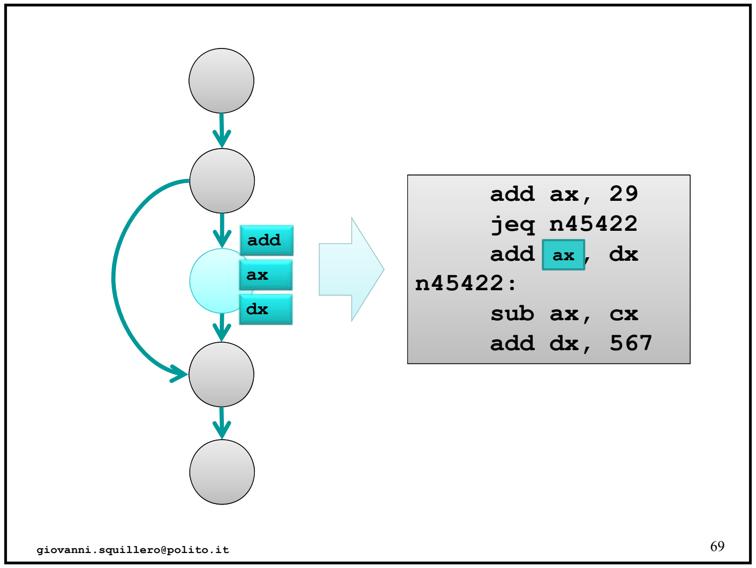
giovanni.squillero@polito.it 64


XML
<op> <reg>, <imm>
<op> = { add, sub }
<reg> = { ax, bx, cd, dx }
<imm> = integer [1, 65,535
<op> <reg1>, <reg2>
<op> = { add, sub }
<reg1> = { ax, bx, cd, dx }
<reg2> = { ax, bx, cd, dx }
<op> <ref>
<op> = { jeq, jne }
<ref> = any forward node
add
ax
29
add
dx
567
giovanni.squillero@polito.it
65

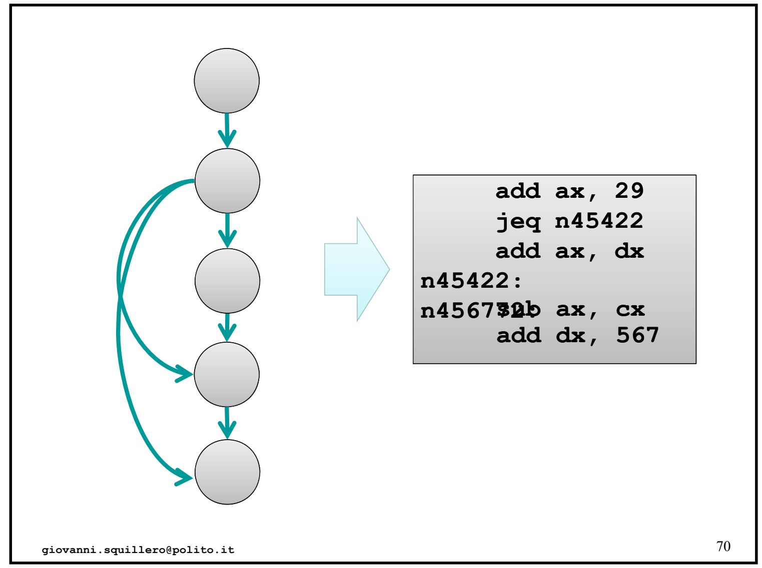

XML
<op> <reg>, <imm>
<op> = { add, sub }
<reg> = { ax, bx, cd, dx }
<imm> = integer [1, 65,535]
<op> <reg1>, <reg2>
<op> = { add, sub }
<reg1> = { ax, bx, cd, dx }
<reg2> = { ax, bx, cd, dx }
<op> <ref>
<op> = { jeq, jne }
<ref> = any forward node
add
cx
dx
sub
ax
cx
giovanni.squillero@polito.it
66


XML
<op> <reg>, <imm>
<op> = { add, sub }
<reg> = { ax, bx, cd, dx }
<imm> = integer [1, 65,535]
<op> <reg1>, <reg2>
<op> = { add, sub }
<reg1> = { ax, bx, cd, dx }
<reg2> = { ax, bx, cd, dx }
<op> <ref>
<op> = { jeq, jne }
<ref> = any forward node
jeq
ref
giovanni.squillero@polito.it
67

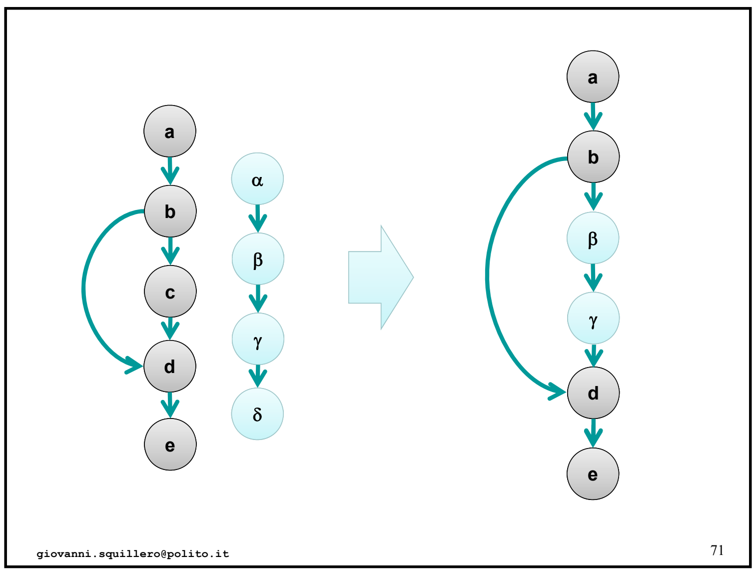

add
ax
29
jeq
ref
add
cx
dx
sub
ax
cx
add
dx
567
add cx, dx
giovanni.squillero@polito.it
68



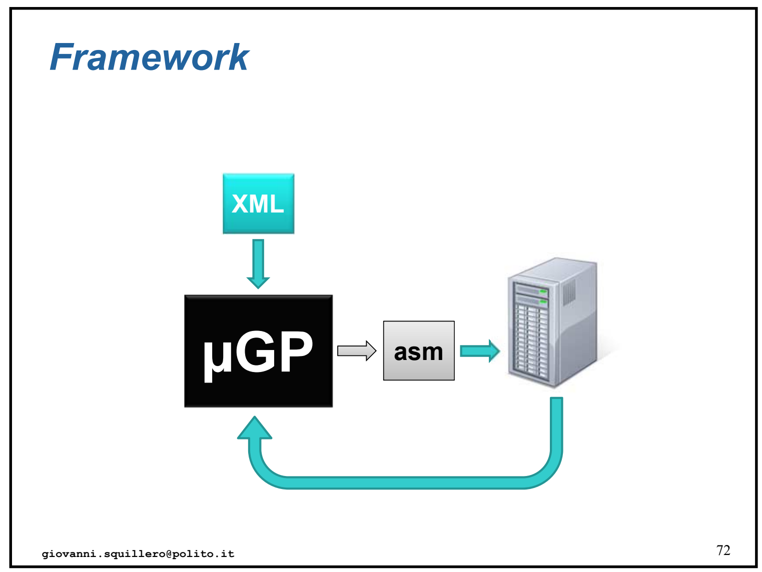
add
ax
dx
add ax, 29
jeq n45422
add ax, dx
n45422:
sub ax, cx
add dx, 567
giovanni.squillero@polito.it
69



add ax, 29
jeq n45422
add ax, dx
n45422:
n456732: sub ax, cx
add dx, 567
giovanni.squillero@polito.it
70



a
b
c
d
e
α
β
γ
δ
a
b
β
γ
d
e
giovanni.squillero@polito.it
71



Framework
XML
µGP
asm
giovanni.squillero@polito.it
72

Language Specification

- Integer instructions
- Legacy x87
- Single-instruction/multiple-data (SIMD)
  - MMX, SSE, SSE2, SSE3, SSSE3, SSE4, …
- Threads
- L1 cache hit/miss

giovanni.squillero@polito.it 73

Feedback

- Undervolting vs. Overclocking
  - Involves different phenomena in the physical world
  - Indistinguishable from the point of view of the optimizer
- Built-in CPU features to control voltage
  - Intel SpeedStep
  - AMD PowerNow! or Cool'n'Quiet
  - VIA Technologies LongHaul

giovanni.squillero@polito.it 74

Feedback

- Main loop:
  - Execute the program at full voltage (safe condition) and record signature
  - Decrease voltage and execute the program until a different signature is detected
  - → Record critical voltage
- Repeat main loop a N times
  - Maximum critical voltage
  - Percentage% of failures at max critical voltage

giovanni.squillero@polito.it 75

Experimental evaluation

- **Intel Core 2 Duo E2180**
  - 2007
  - dual-core (2 threads)
  - SSSE3

- **Intel Core i7-950**
  - 2009
  - quad-core (8 threads)
  - simultaneous multithreading
  - SSE 4.2

giovanni.squillero@polito.it 76

Experimental evaluation

❖ Possible comparison:
- State of the art Stability Tests from the overclockers community

Experimental evaluation

❖ Stability tests
- SuperPI
- CPU BurnIn
- Prime95
- IntelBurnTest
- LinX
- OCCT

Experimental results (E2180)

(single thread)

CORE V	SuperPI	CPU BurnIn	µGP
1.2625	7”	5’	≤ 1”
1.2750	10’	> 10’	≤ 1”
1.2875	> 10’	> 10’	> 10’

Experimental results (E2180)

CORE V	Prime95	IntelBurn Test	LinX	OCCT	µGP
1.2625	1”	2’	2’	3”	≤ 1”
1.2750	6”	2’	2’	4”	2”
1.2875	4’	4’	2’	7’	2”
1.3000	> 10’	7’	7’	> 10’	10”
1.3125	> 10’	> 10’	> 10’	> 10’	8’
1.3250	> 10’	> 10’	> 10’	> 10’	> 10’

Experimental results (i7)

CORE V	Prime95	IntelBurn Test	LinX	OCCT	µGP
1.21250	6'	1'	2'	4'	<1"
1.21875	>10'	>10'	4'	5'	<1"
1.22500	>10'	>10'	>10'	>10'	<1"
1.23125	>10'	>10'	>10'	>10'	<1"
1.23750	>10'	>10'	>10'	>10'	<1"
1.24375	>10'	>10'	>10'	>10'	<1"
1.25000	>10'	>10'	>10'	>10'	<1"
1.25625	>10'	>10'	>10'	>10'	1"
1.26250	>10'	>10'	>10'	>10'	1"
1.26875	>10'	>10'	>10'	>10'	3"
1.27500	>10'	>10'	>10'	>10'	3"
1.28125	>10'	>10'	>10'	>10'	3"
1.28750	>10'	>10'	>10'	>10'	5"
1.29375	>10'	>10'	>10'	>10'	30"
1.30000	>10'	>10'	>10'	>10'	2'
1.30625	>10'	>10'	>10'	>10'	5'
1.31875	>10'	>10'	>10'	>10'	>10'

Experimental results (addenda)

CPU CK [GHz]	FREQ. [GHz]	CORE V	µGP (~ 1')	IBT	LinX	Prime95
i7 920 / 2.66	3.20	1,000	FAIL	PASS		
i7 950 / 3.06	4.03	1,310	PASS	PASS		
i7 950 / 3.06	4.03	1,280	FAIL	FAIL		
i7 920 / 2.66	4.02	1,270	FAIL		PASS	
i7 950 / 3.06	4.00	1,256	FAIL			PASS
i7 950 / 3.06	4.00	1,262	PASS			PASS
i7 860 / 2.80	4.30	1,400	FAIL		FAIL	
i7 860 / 2.80	4.25	1,400	PASS		FAIL	
i7 950 / 3.06	4.20	1,300	FAIL			FAIL

Detected after 60'

Conclusions

- Fully automatic
- Easy to set-up
- Able to exploit specific micro-architectural features
  - Works on i7-950
  - Does not work on i7-860

Dissemination

- "Post-Silicon Functional Failing-Test Generation through Evolutionary Computation"
- *ETS'11 - 16th IEEE European Test Symposium*