

EA-based Test and Verification of Microprocessors

Giovanni Squillero
giovanni.squillero@polito.it

Copyright is held by the author/owner(s).
GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-131-6/08/07.

EA-based Test and Verification of Microprocessors

Part 1: Background

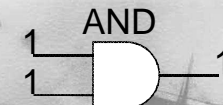
Giovanni Squillero
giovanni.squillero@polito.it

Outline

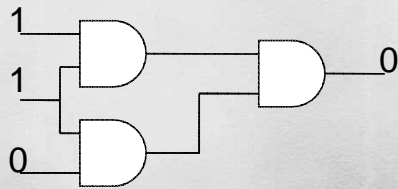
- Electronic circuit
- Levels of descriptions
 - Gate, RT, Behavioral
 - Comparison with programs
- Verification, Validation and Test

Digital Circuit

- Electric signals represent logical values
 - Discrete set of values (0, 1, X, ...)
 - Simplified timing information
- Logic gates
 - Logical operation on one or more inputs
 - Single output



Combinational Digital Circuit

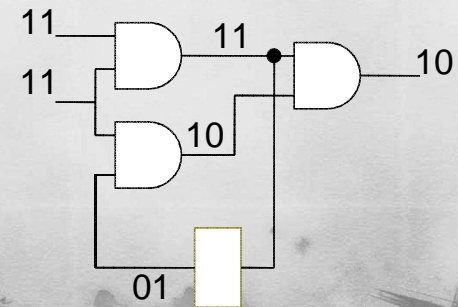


Part I - Background

G. Squillero

5

Sequential Digital Circuit

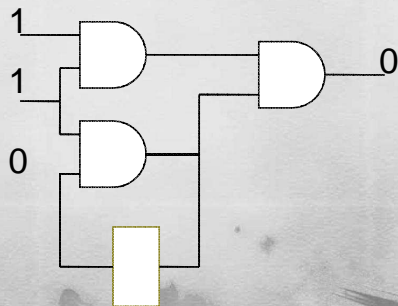


Part I - Background

G. Squillero

6

Sequential Digital Circuit



Part I - Background

G. Squillero

7

if anything can go wrong, it will...

Part I - Background

G. Squillero

8

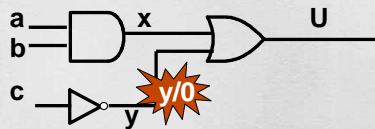
Non-functioning Circuits

- Industry goal:
 - Detect bad devices just after production
 - Apply a set of input stimuli able to discriminate malfunctioning devices from working ones
- Problems:
 - How to devise a suitable “set of input stimuli”?

Fault Models

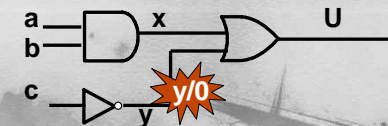
- Fault vs. Defect vs. Error
- Fault models
 - Stuck-at, stuck open
 - Bridging
 - Delay (path, gate, transitional, ...)
 - ...
- Single vs. Multiple
- Permanent vs. Transient

Single Permanent Stuck-at



Test Sequence

- Test Generation
 - Excitation: $y = 1$ $\{abc = -- 0\}$
 - Propagation: $x = 0$ $\{abc = 0 --, - 0 -\}$
 - Final test = *excitation* · *propagation*
- Test = $\{- - 0\} \cdot \{0 --, - 0 -\} = \{0 - 0, - 00\}$
 $= \{000, 100, 010\}$



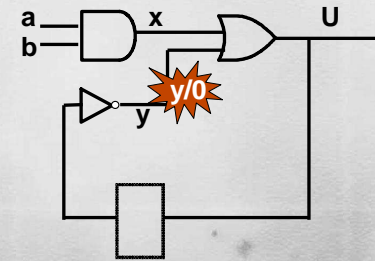
ATPG

(Automatic Test Pattern Generator)

An electronic design automation method/technology used to find an input sequence that, when applied to a digital circuit, enables testers to distinguish between the correct circuit behavior and the faulty circuit behavior caused by defects

(Wikipedia)

Sequential ATPG



Metrics

- Defect coverage (FC%)
- Fault coverage (FC%)
- Testable fault coverage (TC%)
- Fault efficiency
- ...

Levels of Abstraction

- Gate
- Register Transfer
- Behavioral

Hardware Description Language

- EDIF
- BLIF
- VHDL
- Verilog
- SystemVerilog
- SystemC
- ...

Levels of Abstraction

- Can be simulated?
- Can be synthesized?
 - How?

Register-Transfer VHDL

```
architecture RTL of MY_AND is
begin process(x, y)
begin
if ((x='1') and (y='1')) then
F <= '1';
else
F <= '0';
end if;
end process;
end foo;
```

Behavioral VHDL

```
architecture BEHAV of FOOBAR is
signal A, B: BIT_VECTOR(3 downto 0);
begin
A(0) <= X after 20ns;
A(1) <= Y after 40ns;
process(A)
variable P, Q: BIT_VECTOR(3 downto 0);
begin
P := fft(A);
B <= P after 10ns;
end process;
Z <= B;
end foo;
```

Behavioral VHDL

```
architecture BEHAV of FOOBAR is
signal A, B: BIT_VECTOR(3 downto 0);
begin
  A(0) <= X after 20ns;
  A(1) <= Y after 40ns;
  process(A)
    variable P, Q: BIT_VECTOR(3 downto 0);
  begin
    P := fft(A);
    B <= P after 10ns;
  end process;
  Z <= B;
end foo;
```

HDL

- Can be simulate
 - Not executed
- Description of a physical hardware
 - Not imperative language

HDL

```
if (rst='0') then
  REG1(j) <= (REG1(j)'range => '0');
  REG2(j) <= (REG2(j)'range => '0');
  COEF(j) <= (COEF(j)'range => '0');
  MULT16(j) <= (MULT16(j)'range => '0');
  SUM(j) <= (SUM(j)'range => '0');
elsif (clk'event and clk = '1') then
  REG1(j) <= REG2(j+1);
  REG2(j) <= REG1(j);
  if (coef_ld = '1') then
    COEF(j) <= COEF(j+1);
  end if;
  MULT8(j) <= signed(REG1(j))*signed(COEF(j));
```

High-Level ATPG

- Very active line of research
- Industrially relevant
- Missing an suitable fault model
- Missing correlation (high-level to lo low-level metrics)
- **A+B vs. A*B**

V&V

- Validation
 - Evaluate whether a system accomplishes its intended requirements
 - *Are we building the right system?*
- Verification
 - Evaluate whether a system complies with the conditions imposed at the start of a development phase
 - *Are we building the system right?*

V&V in CAD

- Verification
 - Formal verification (e.g., mathematical models and theorem proving)
 - Simulation and assertion (i.e., properties) checks
 - Comparison with a *golden model*
- Validation
 - Sometimes confused with verification

EA-based Test and Verification of Microprocessors

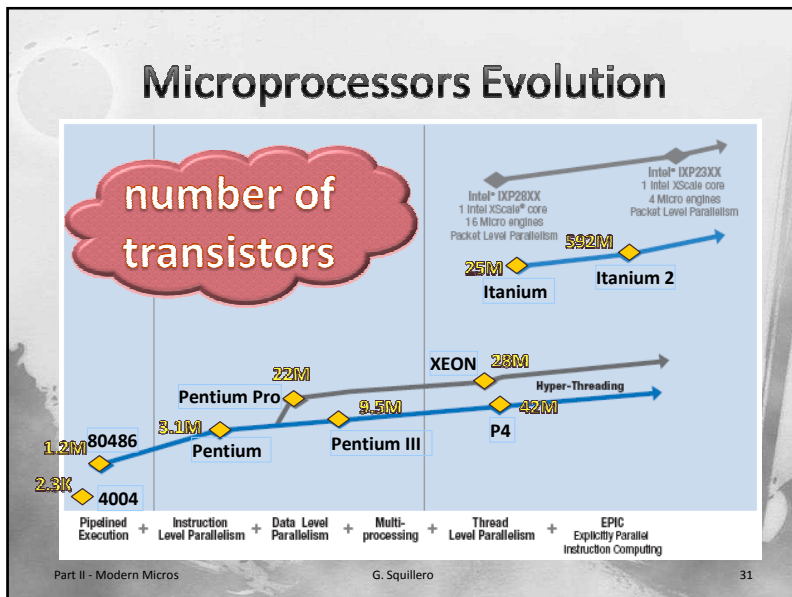
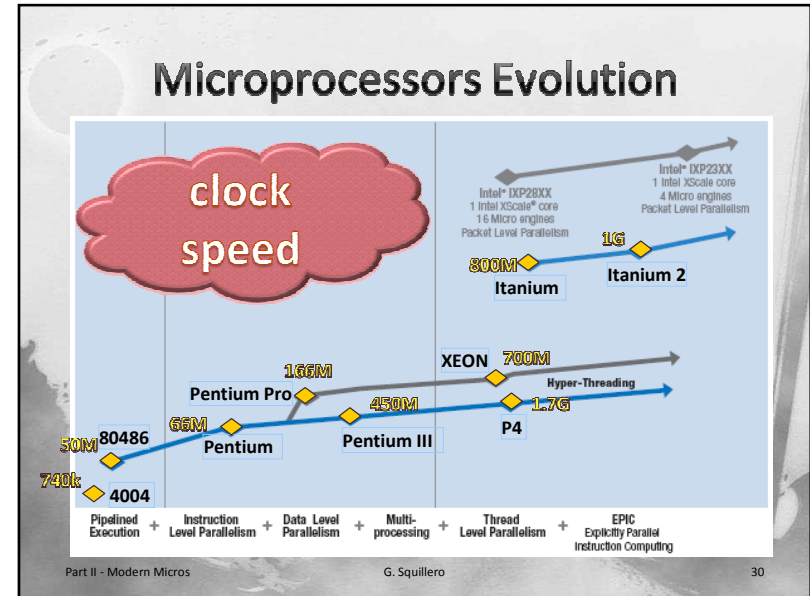
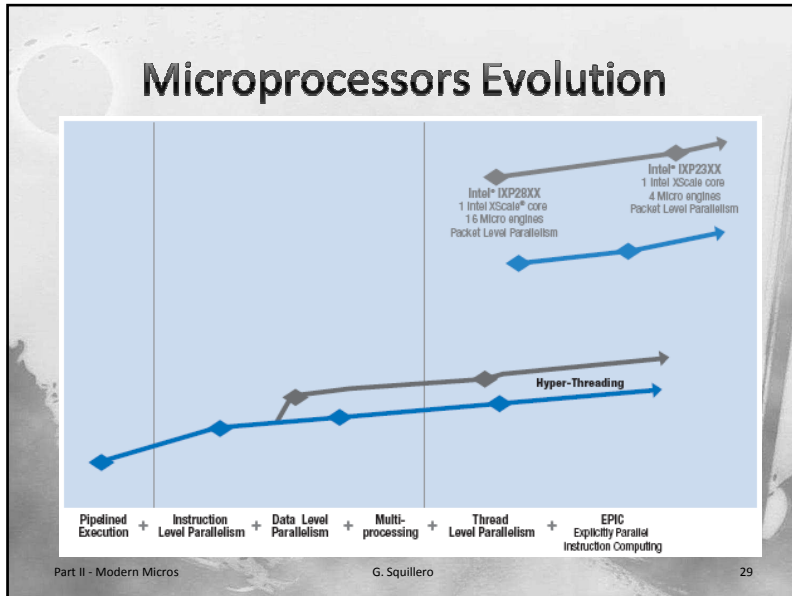
Part 2: Modern Micros

Giovanni Squillero

giovanni.squillero@polito.it

Outline

- Micros challenges
 - Size
 - Complexity
 - Competitive pressure



- ### Old Microprocessor
- Intel 4004 (world's first commercial microprocessor)
 - Released in late 1971
 - Discontinued in 1981
 - 4-bit CPU
 - 2,300 transistors
 - 740 kHz
 - Execute approx 92,000 instructions/sec
- Part II - Modern Micros G. Squillero 32

Modern Microprocessor

- Pentium 4
 - Released in late 2000
 - 32-bit CPU
 - 42,000,000 to 55,000,000 transistors
 - 1.40 GHz (initial) to 3.40 GHz (Northwood C, 2004)
 - Execute up to 10,000,000,000 instructions/sec

4004 vs. P4

- 20,000 times bigger
- 100,000 times faster

Modern Microprocessors

- Scalar architecture
- Superscalar architecture

Modern Microprocessors

- Strategies
 - Cache
 - Branch prediction
 - Parallelism
 - Pipeline
 - Out-of-order execution
 - Speculative execution
 - Simultaneous multithreading
 - Multi-core design

High-End Micros

- Relatively small volumes
- Complex structure, innovative design
- Unstable technology
- PC, Laptop
- High cost (hundreds of Euros)

Low-Cost Micros

- High volumes
- Relatively simple
- Stable technology
- Embedded in other systems (e.g., USB controllers)
- Low cost (usually less than €1)

Between the two Extremes?

- A wide range of solutions
 - 8- to 32- bit microcontrollers
 - Variable clocks and performances
 - Variable memory
 - Variable costs
- Today high-end micros will be the core of tomorrow microcontrollers...
 - E.g., Freescale *Power Architecture* for microcontrollers

Open Problems

- Semantic does matter!
- Input stimuli must be regarded as *programs* and not simply as *binary data*

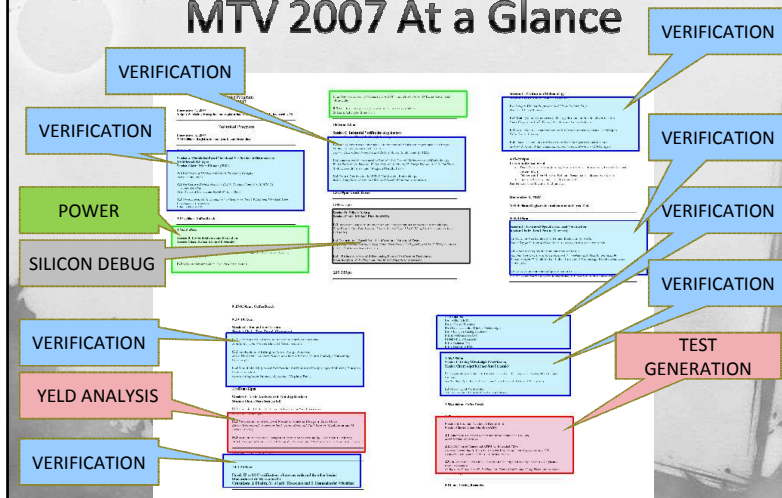
Open Problems: Test

- Design are too complex to run logic simulation
 - One logic simulation is required to evaluate the effect of each fault
 - The number of faults is roughly two times the number of gates
 - Which fault model?

Open Problems: Verification

- Design are too complex for exact verification
 - Simplified models?
 - Simulation-based approaches?
 - Instructions randomizers
- Designs are too complex for running extensive simulations
 - Pre-synthesis vs. post-synthesis verification

MTV 2007 At a Glance



EA-based Test and Verification of Microprocessors

Part 3: Proposed Methodology

Giovanni Squillero
giovanni.squillero@polito.it

Outline

- Design choices
- Proposed methodology
 - Stimuli
 - System
 - Stimuli generator
 - Feedback

Design Choices

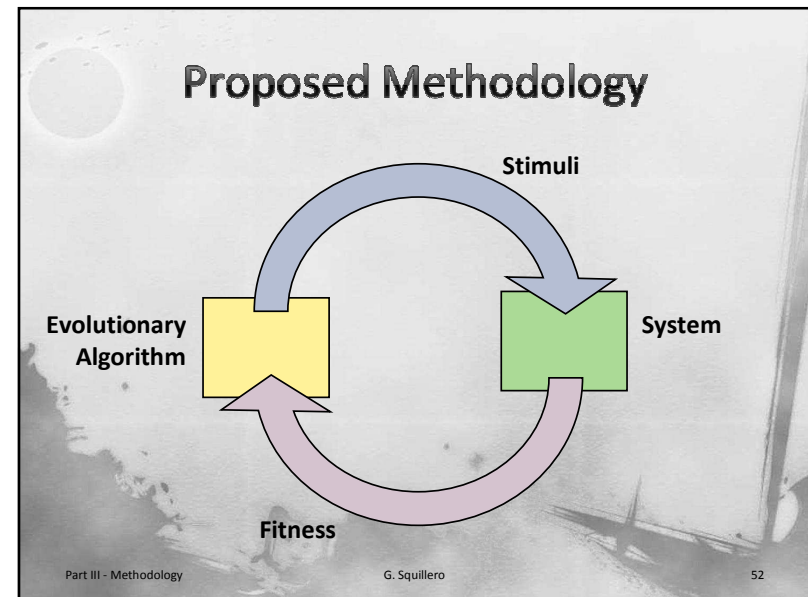
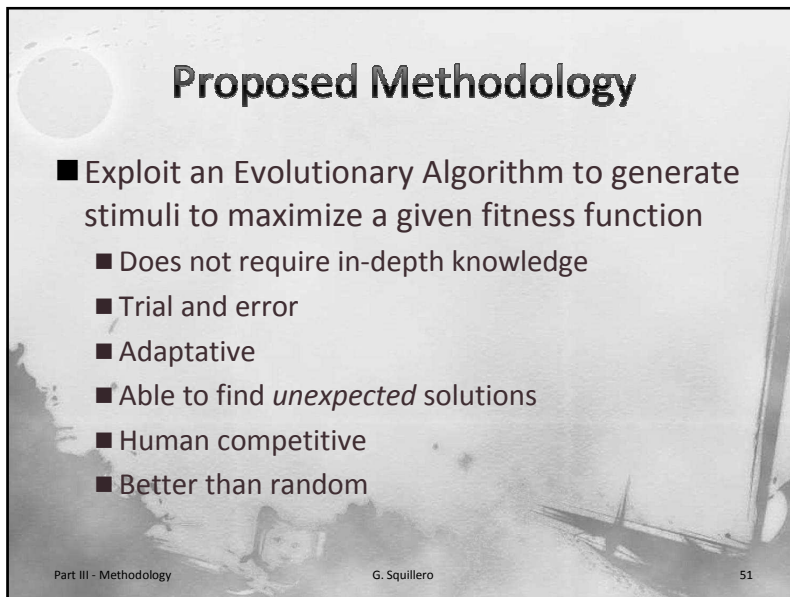
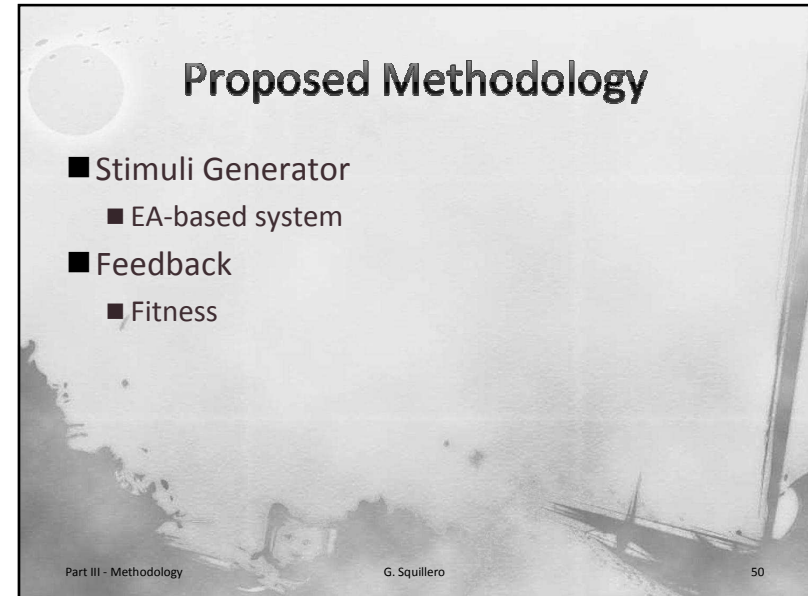
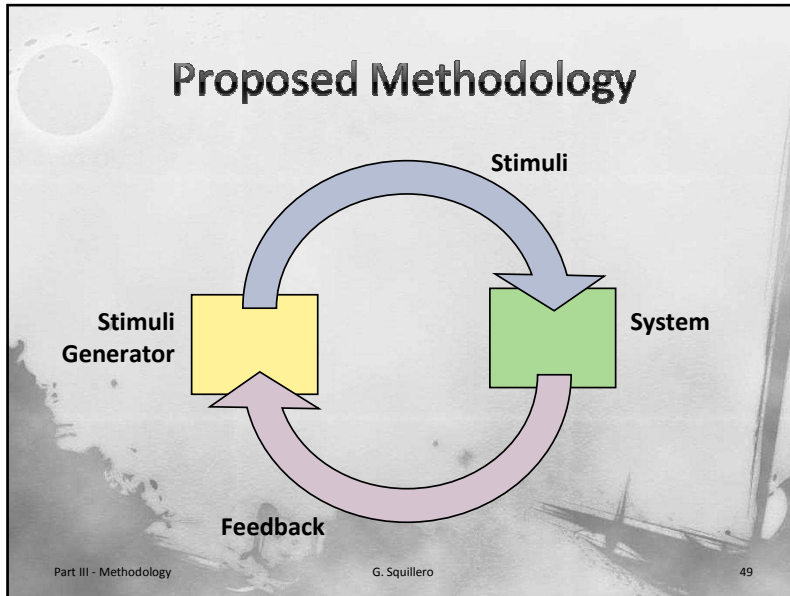
- Being able to tackle real problems
- Uniform approach
 - Exploit underlying common aspects
 - Minimize effort to change goal/target

Simulation-based approach

- Pros:
 - Uncover design errors by detecting incorrect behaviors when tests are applied
 - May be usable on under-specified models
 - May require limited computational resources
- Cons:
 - Only consider a limited range of behaviors
 - Never achieve 100% confidence of correctness

Feedback-Based Approach

- Exploits feedback from simulation
- Incremental improvement/refinement of the solutions (trial-and-error)
- Trade-off between computational resources and confidence
- May exploit heuristics (e.g., evolutionary core) or problem-specific knowledge



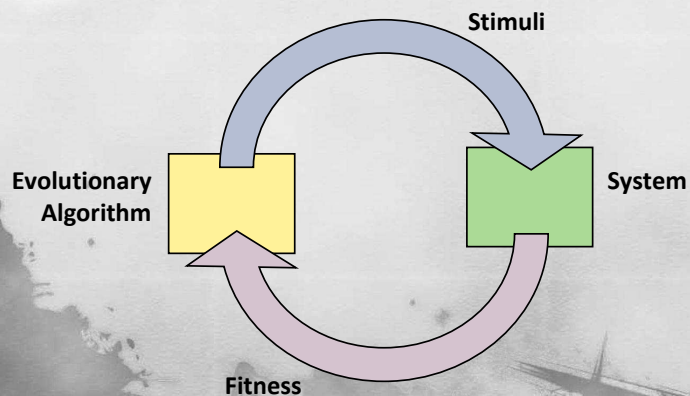
Class of Problems

- Maximization
 - Test (maximize FC%)
 - Verification (maximize tested functionalities)
 - ...
- Needle in a haystack
 - Find a counter-example
 - Find a bug
 - ...

Class of Problems

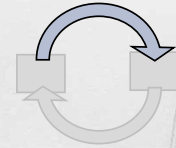
- Transform needle-in-a-haystack problems into maximization problems
- Smooth fitness landscape
- Intermediate goal
- Heuristic
- Problem-specific knowledge
- Favor exploration

Proposed Methodology



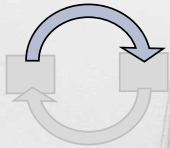
Stimuli

- Valid assembly language programs
 - Exploit all syntax
 - Instruction asymmetries
 - Subroutines/Interrupt handlers
 - Microprocessor peculiarities
 - Register windows on SPARC
 - Global Descriptor Table and protected mode in IA86



Stimuli

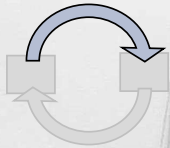
- Lower than assembly?
 - Even assembly is a *high-level language* (e.g., in x86 the same opcode corresponds to different machine code instructions)



Part III - Methodology G. Squillero 57

Stimuli

- External world?
 - In order to check a device (e.g., a I/O block) *external stimuli* must be considered
 - Highly correlated



Part III - Methodology G. Squillero 58

Stimuli Generator

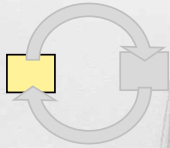
- Our tool: MicroGP



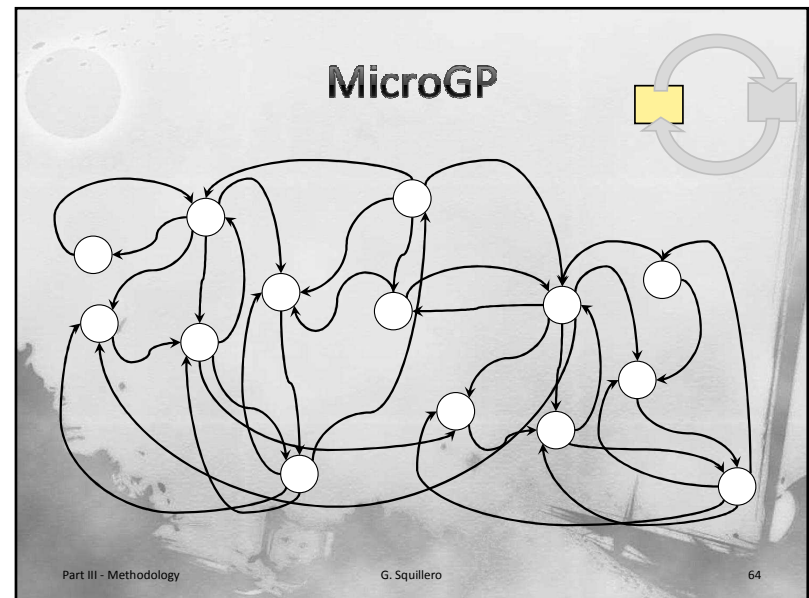
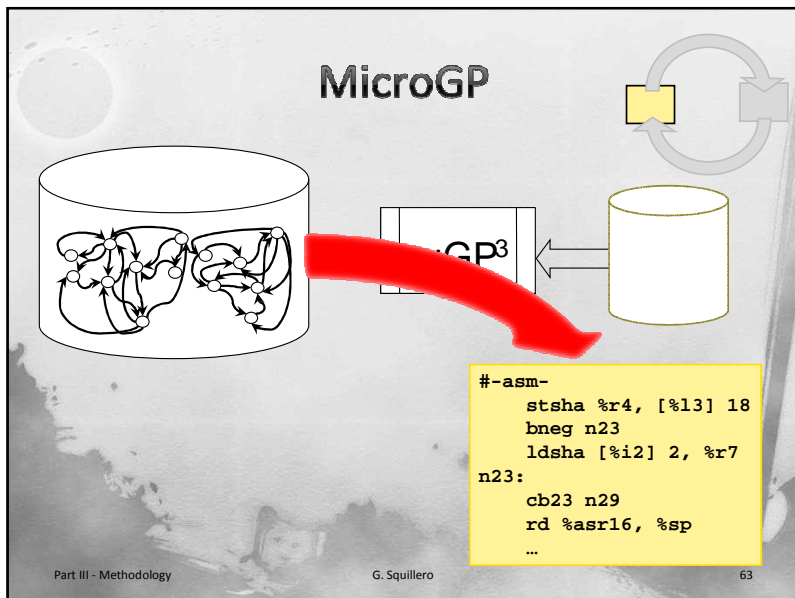
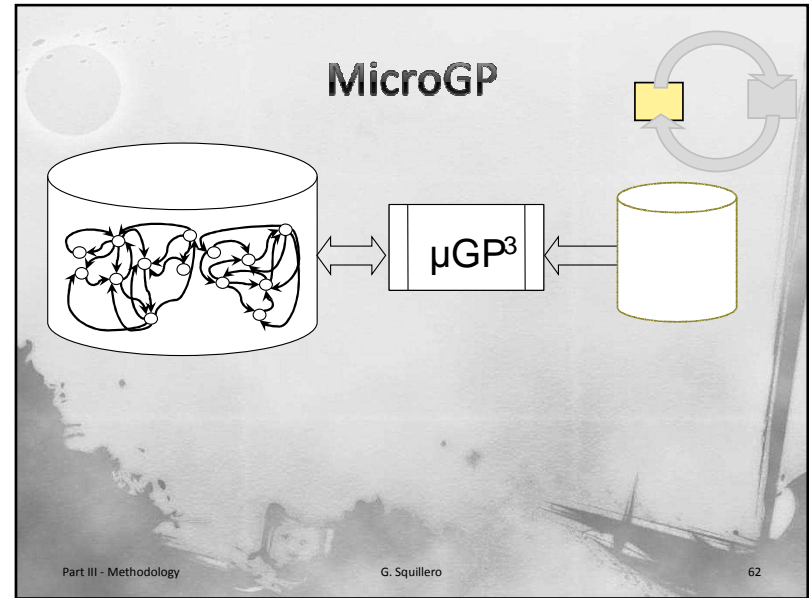
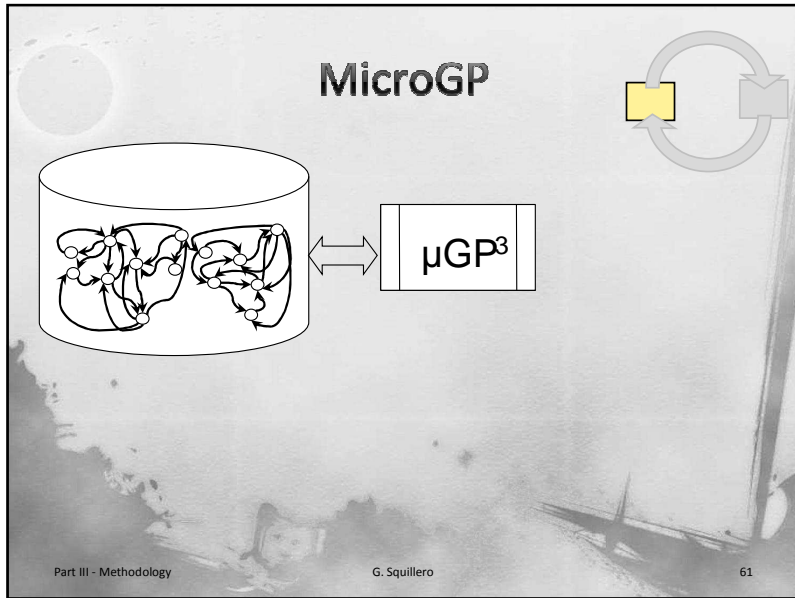
Part III - Methodology G. Squillero 59

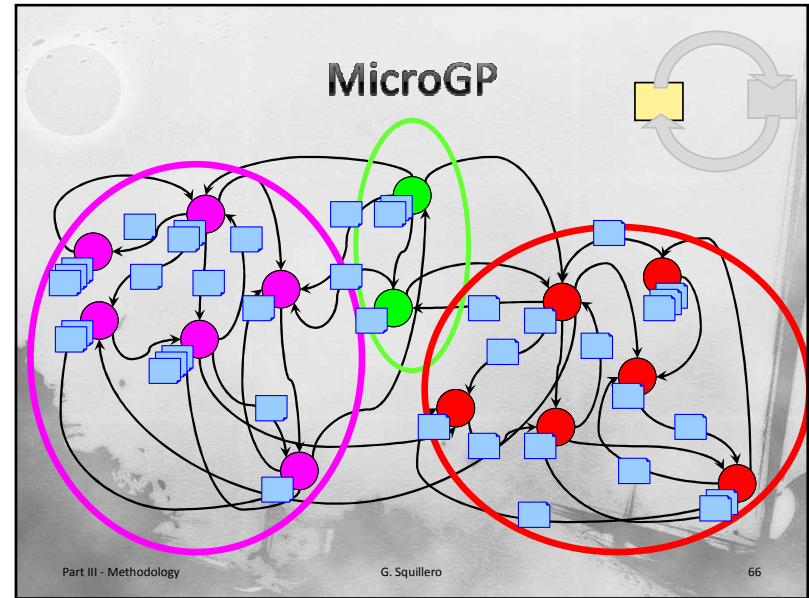
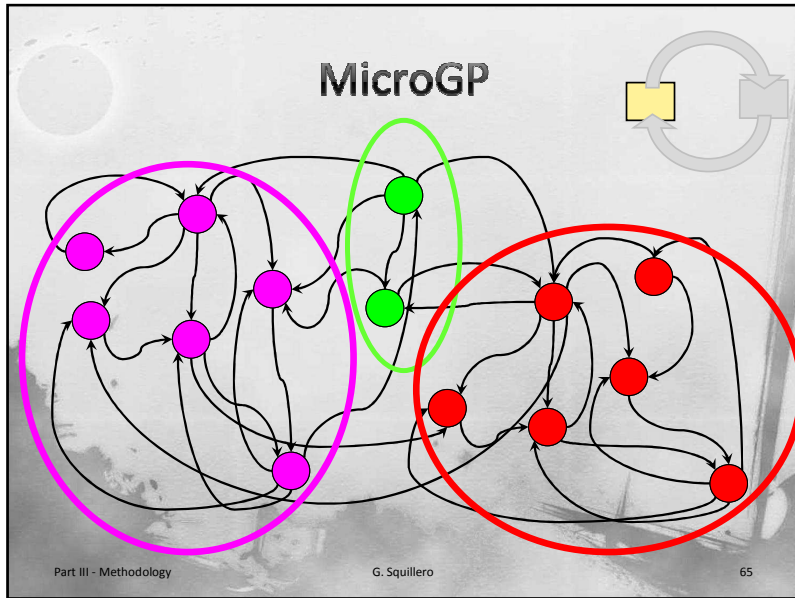
MicroGP

- MicroGP++ (μGP^3)
 - CAD Group general-purpose evolver
 - Project started in 2002
 - 3 *versions* (only 2 released through sourceforge)
 - 8 developers, plus several contributors
 - Actual version: 3.1
 - $\approx 15,000$ lines in C++
 - Plus some utilities in different languages
 - MOEA in version 3.1 (stable in late 2008?)



Part III - Methodology G. Squillero 60





- ### MicroGP
- Multiple populations
 - Variable & self adapted
 - Offspring and Population size
 - Selective pressure
 - Operation probabilities
 - Behavior range smoothly from pure steady state to pure generational
 - Entropy-based technique to favor diversity
 - Clone detection
 - ...
- Part III - Methodology G. Squillero 67

MicroGP

<http://ugp3.sourceforge.net/>

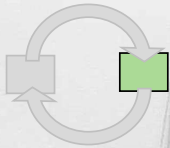
MicroGP++ (aka. `ugp3`, μGP^3)

- Information
- Download
- Credits

Part III - Methodology G. Squillero 68

System

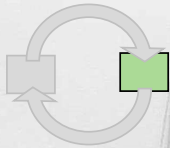
- System
 - The microprocessor
 - Helper module



Part III - Methodology G. Squillero 69

Microprocessor

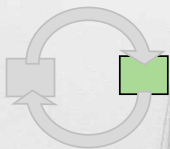
- (Obviously) problem dependant
- Model via simulation/emulation
 - HDL (netlist to high-level)
 - HW accelerated (e.g., exploiting FPGA)
 - Architectural simulator
 - ISA simulator
- Real device



Part III - Methodology G. Squillero 70

Helper Module

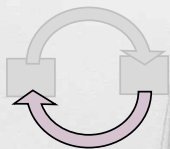
- Usually a collection of scripts
- Apply stimuli
- Analyze behavior
- Translate output to fitness
 - e.g., a file containing a list of real numbers



Part III - Methodology G. Squillero 71

Fitness

- From simulation
 - Code coverage metrics (instruction, branch, ...)
 - HW specific metrics (toggle coverage)
 - High-level information (FSM coverage)
- From running the real microprocessor
 - Performance counters
 - Physical measures (temperature, time)

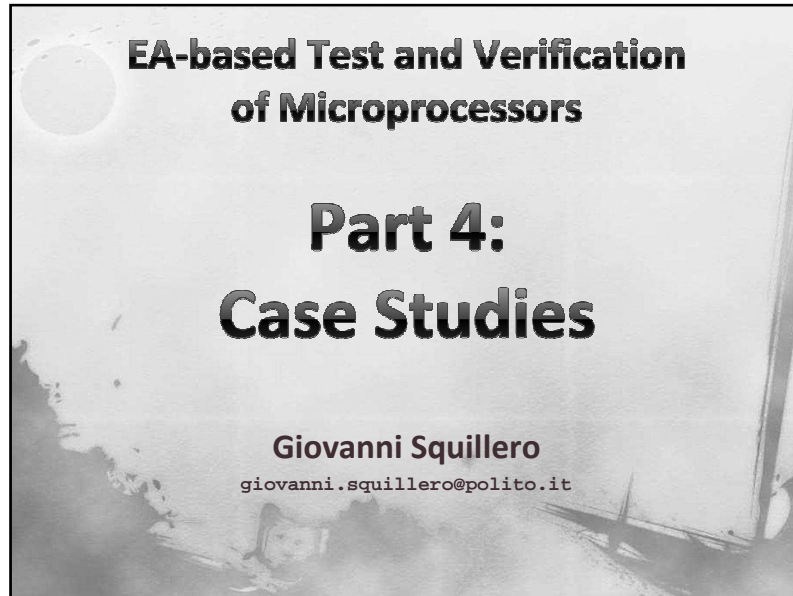


Part III - Methodology G. Squillero 72

EA-based Test and Verification of Microprocessors

Part 4: Case Studies

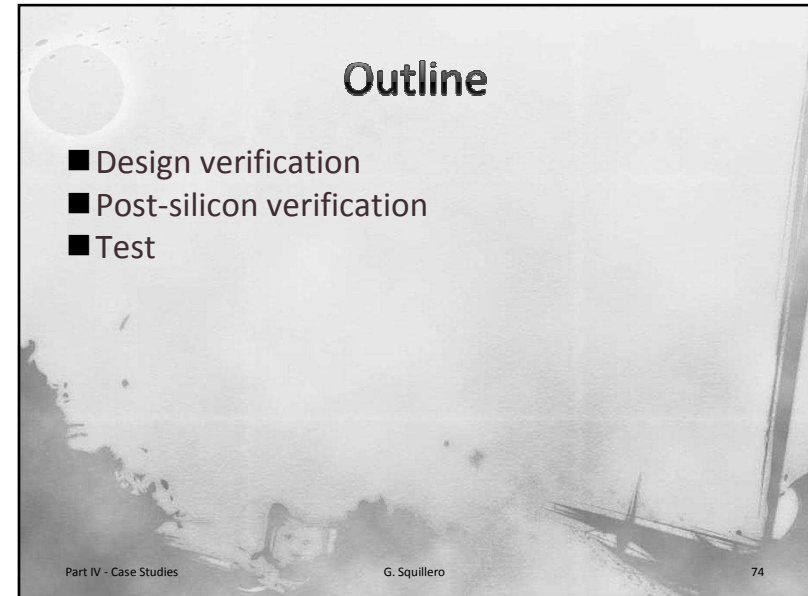
Giovanni Squillero
giovanni.squillero@polito.it



Outline

- Design verification
- Post-silicon verification
- Test

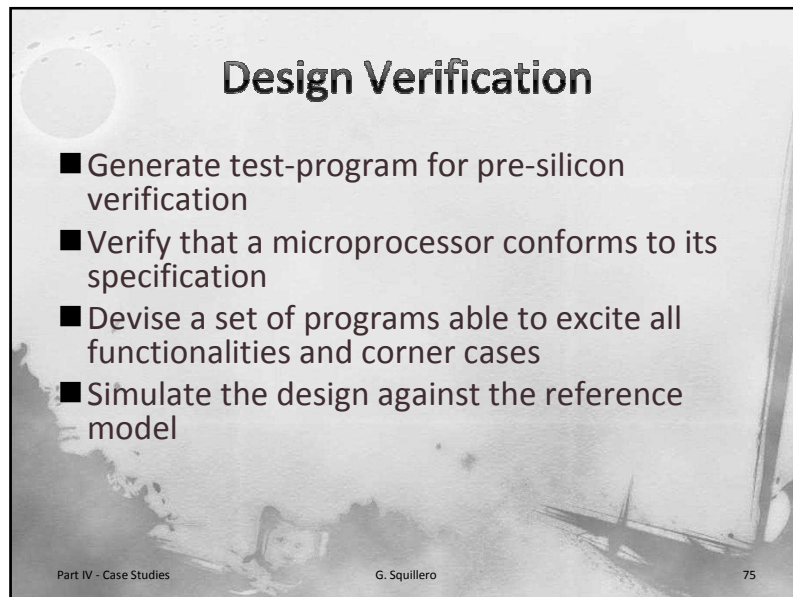
Part IV - Case Studies G. Squillero 74



Design Verification

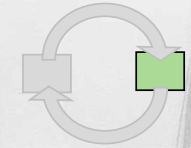
- Generate test-program for pre-silicon verification
- Verify that a microprocessor conforms to its specification
- Devise a set of programs able to excite all functionalities and corner cases
- Simulate the design against the reference model

Part IV - Case Studies G. Squillero 75

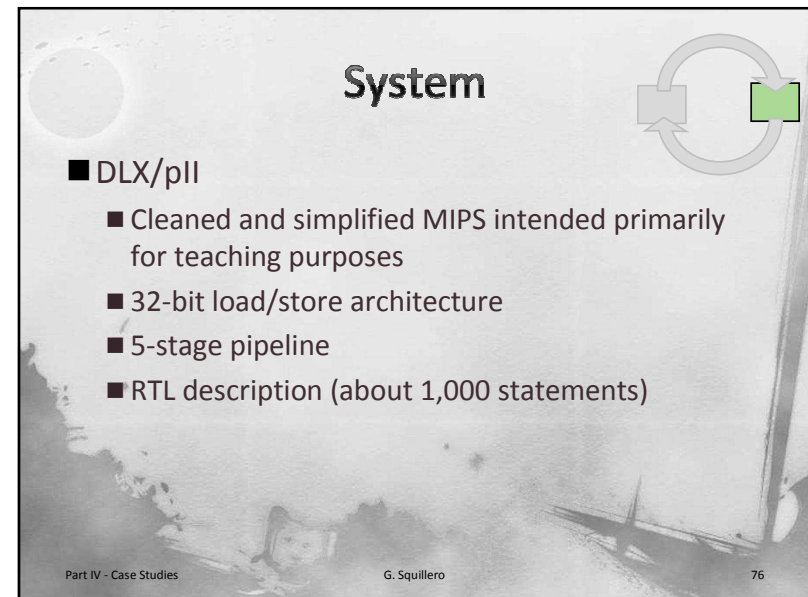


System

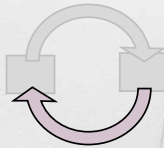
- DLX/pII
 - Cleaned and simplified MIPS intended primarily for teaching purposes
 - 32-bit load/store architecture
 - 5-stage pipeline
 - RTL description (about 1,000 statements)



Part IV - Case Studies G. Squillero 76



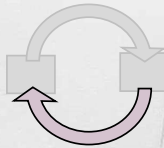
Feedback



- Code coverage metrics
 - Parts of the description that the have been *evaluated* by the HDL simulator
 - Caveat: it is not a program
- HW specific metric

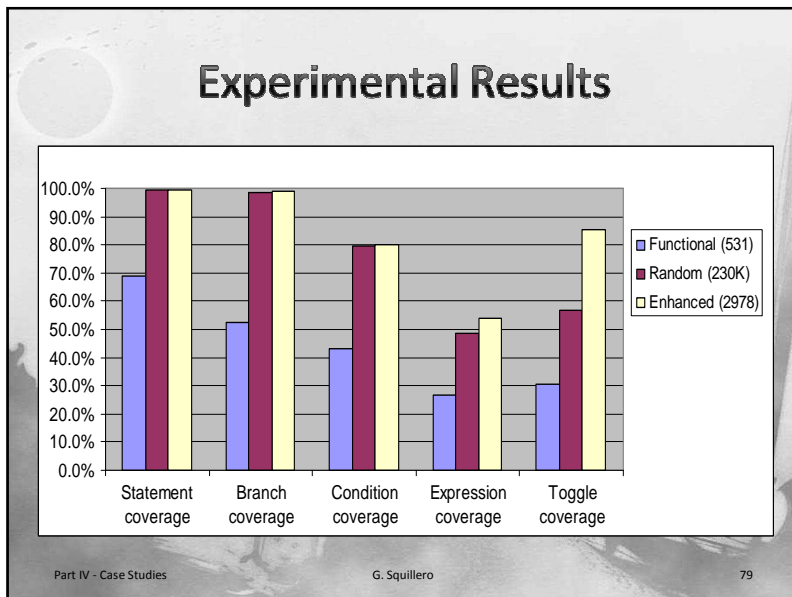
Part IV - Case Studies G. Squillero 77

Feedback



- Code coverage metrics
 - Statement coverage
 - Branch coverage
 - Condition coverage
 - Expression coverage
- HW specific metric
 - Toggle coverage

Part IV - Case Studies G. Squillero 78



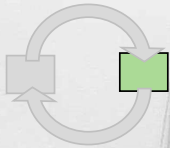
Post-silicon Verification

- Generate functional test-program for post-silicon verification
- The generated test programs
 - could be added as new content to improve existing validation suites
 - can be used to perform regression testing on future processor models
- Activity performed in collaboration with the *Embedded Test Methodology Group, Intel*

Part IV - Case Studies G. Squillero 80

System

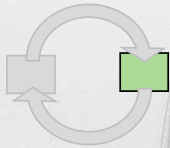
- Intel Pentium 4
 - 55 millions transistor
 - 5 millions gate (my unreliable estimate)
 - 2GHz clock
 - *NetBurst* architecture



Part IV - Case Studies G. Squillero 81

System

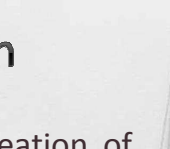
- Performance counters
 - Introduced in 1993 in IA-32 architecture
 - P4 Counters architecture:
 - 48 event detectors
 - 18 event counters
 - 18 counter configuration control registers
 - Instruction-tagging (for discriminating non-speculative performance events)



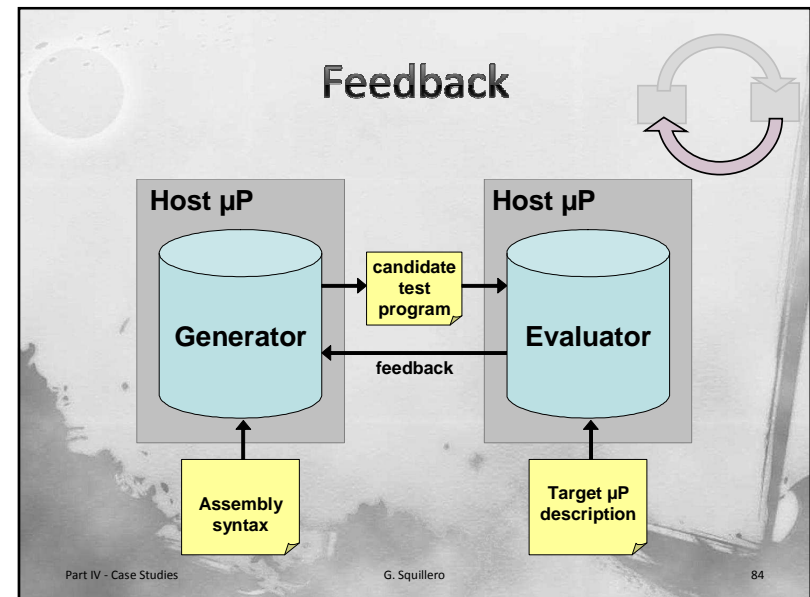
Part IV - Case Studies G. Squillero 82

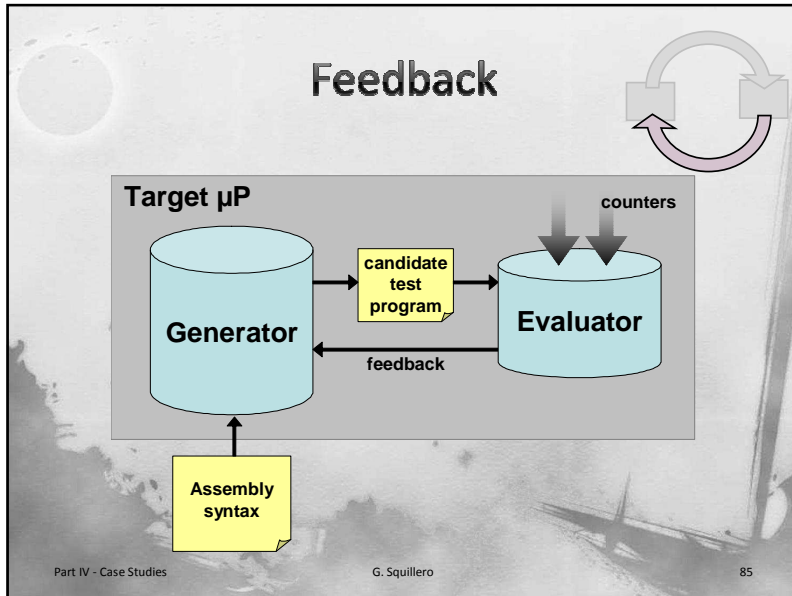
Post-silicon Verification

- Use monitors as a proxy for the creation of certain μ architectural events to
 - stress specific features
 - excite subtle corner cases



Part IV - Case Studies G. Squillero 83





- ### Mispredicted ratio
- Maximize/minimize the ratio of mispredicted branches over the total branches
 - only non-speculative (retired) instructions are considered.
 - controlling the branch prediction rate is challenging (the approach is, by definition, random)
 - could generate interesting code for exciting corner-case events
 - may cover flaws that would be hardly detected by manually-written targeted tests
- Ba QjWllErasE Studies 86

Mispredicted ratio

Program	#INST	Sampling Type	
		Time	Event
Random [max]	278	6.01	5.93
Random [min]	426	0.1	0
Random [avg]	353.87	1.63	1.33
Random [std]	91.91	2.11	2.19
μGP (maximizing)	442	49.34	49.55
μGP (minimizing)	266	0.02	0.01

Ba QjWllErasE Studies 87

- ### Trace cache deliver-mode ratio
- Max/Min ratio of clock cycles in which the trace cache is delivering μops to the execution unit instead of decoding or building traces
 - intrinsic feature of the μarchitectural design
 - tests programs not biased to any specific solutions
 - likely cover multiple cases, while an architect would target specific features
 - hard metric
- Ba QjWllErasE Studies 88

Trace cache deliver-mode ratio

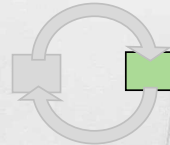
Program	#INST	Sampling Type	
		Time	Event
Random [max]	6	99.32	98.13
Random [min ^T]	5	85.39	
Random [min ^E]	53		81.46
Random [avg]	32.13	91.68	88.58
Random [std]	21.52	4.38	4.75
μGP (maximizing ^T)	36	99.49	
μGP (maximizing ^E)	40		93.94
μGP (minimizing ^T)	5	48.13	
μGP (minimizing ^E)	55		23.33

Test

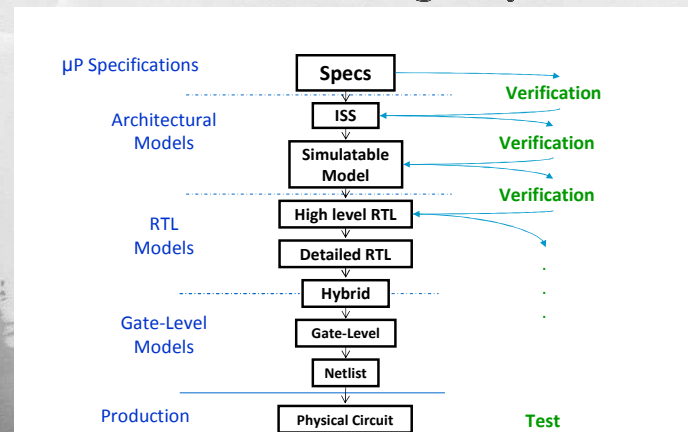
- Devise a test-set suitable for post-production test (i.e., a program that enables testers to distinguish between the correct circuit behavior and the faulty circuit behavior caused by defects)

System

- PLASMA (MIPS I)
 - 3-stage pipelined processor
 - Processor design models
 - Architectural level
 - RTL
 - Gate - level



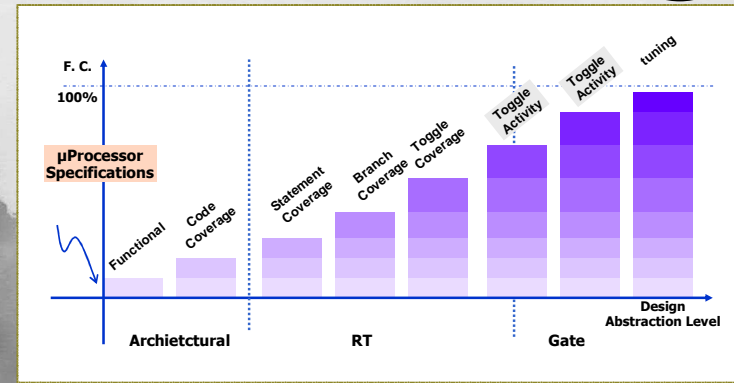
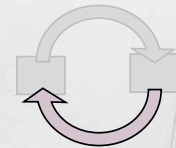
Processor Design-Cycle



Cumulative Methodology

- In each step a test-set is generated
- Already available test set are the starting point for current step
- Incremental generation
- MicroGP exploits the *Borg*
 - Designer functional testbenches can be exploited by the automatic tool
- Manual tuning

Feedback



Results

