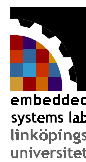
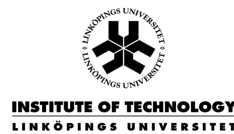
CoTest

COTEST Activities at LiU

Gert Jervan, Zebo Peng
Embedded Systems Laboratory (ESLAB)
Linköping University





Outline

- General Concept
- Structure
- Experimental Setup
- Experimental Results
- Conclusions

Activities at LiU

- An approach for early design space exploration to evaluate testability
- Based on Hierarchical Test Generation (HTG) methodology
  - Combining behavioral level information with gate-level accuracy
- Fully automated design flow
  - Combination of in-house, external academic and commercial tools
- Experimental results to demonstrate the efficiency of proposed approach

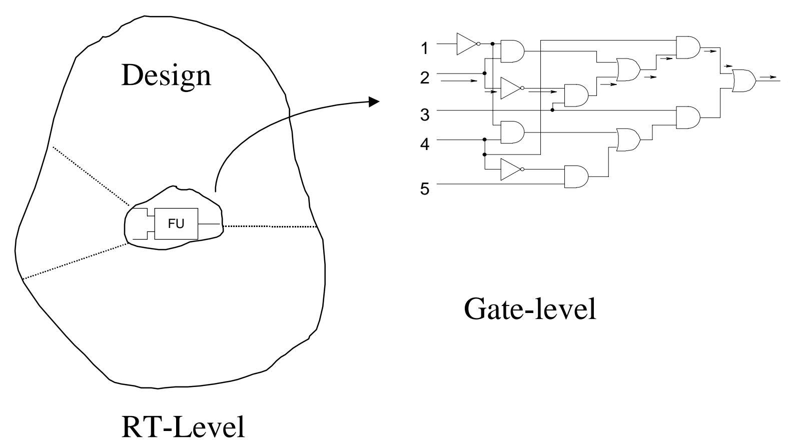
Activities at LiU

- Hybrid BIST Insertion
  - An architecture that supports the combination of pseudorandom and deterministic test patterns in cost effective way to improve design’s testability.
  - The architecture can be implemented only in software to avoid area overhead, not to affect system performance and to provide possibility for test reconfiguration and is therefore flexible and well suitable for SoC designs.
- BIST insertion and optimization at HLS

Earlier Work

- Modeling on a behavioral level by using Decision Diagrams (DDs)
- Hierarchical test generation algorithm based on code coverage metrics and mutation fault model

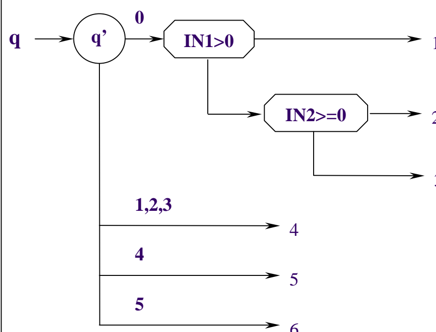
Hierarchical Test Generation



Behavioral DD Model

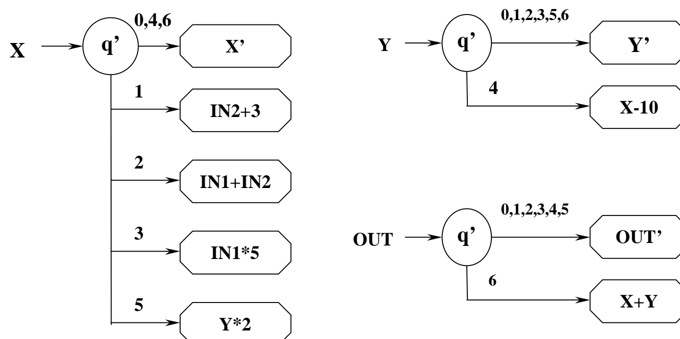
```
if (IN1 > 0)
    X=IN2+3;       --- q=1
else {
    if (IN2 >= 0)
        X=IN1+IN2; -- q=2
    else
        X=IN1*5;   --- q=3
}

Y=X-10;        -------- q=4
X=Y*2;         -------- q=5
OUT=X+Y;       -------- q=6
```

A Behavioral Specification



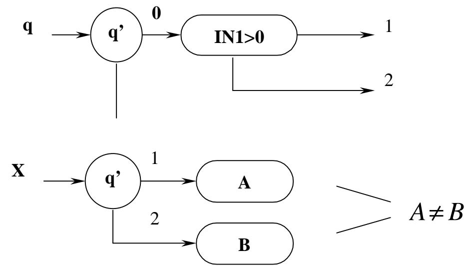
Control-flow DD

Behavioral DD Model

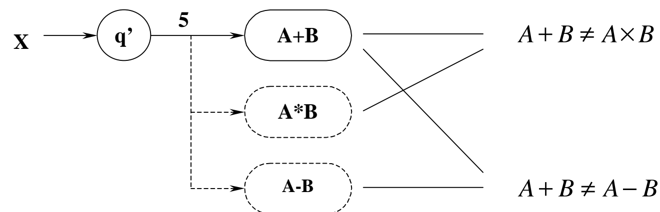


Datapath DDs

Branch Activation Fault Model



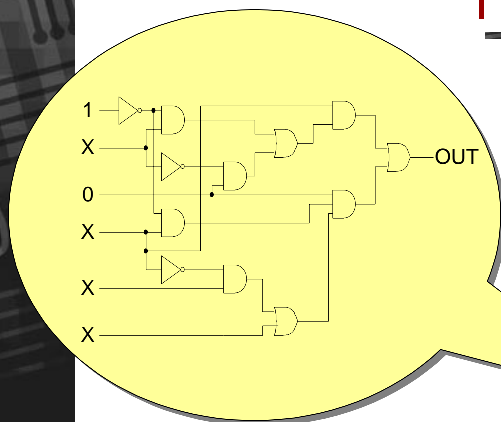
Operator Mutation Fault Model



New Approach

Usage of structural information in a behavioral test environment

Hierarchical Test



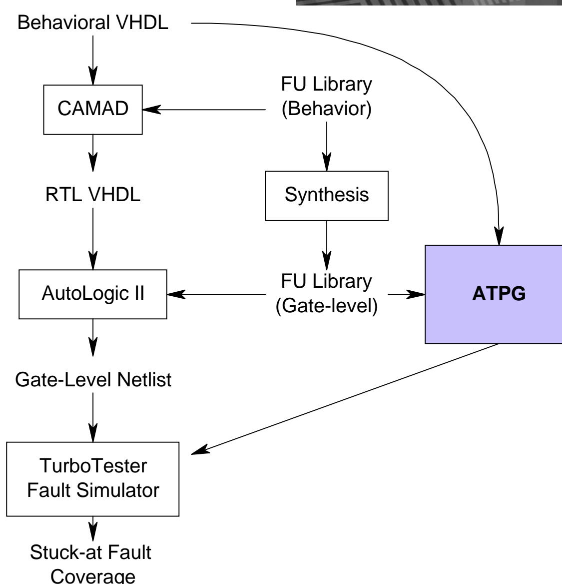

```
if (IN1 > 0)
    X=IN2+3;      --- q=1
else {
    if (IN2 >= 0)
        X=IN1+IN2; -- q=2
    else
        X=IN1*5;  --- q=3

          -------- q=4
X = Y * 2;
OUT=X+Y;    -------- q=6
if (IN1 > 0)
    X=IN2+3;      --- q=1
else {
    if (IN2 >= 0)
        X=IN1+IN2; -- q=2
    else
        X=IN1*5;  --- q=3
}

Y=X-10;     -------- q=4
X=Y*2;      -------- q=5
OUT=X+Y;    -------- q=6
```

Our New Approach

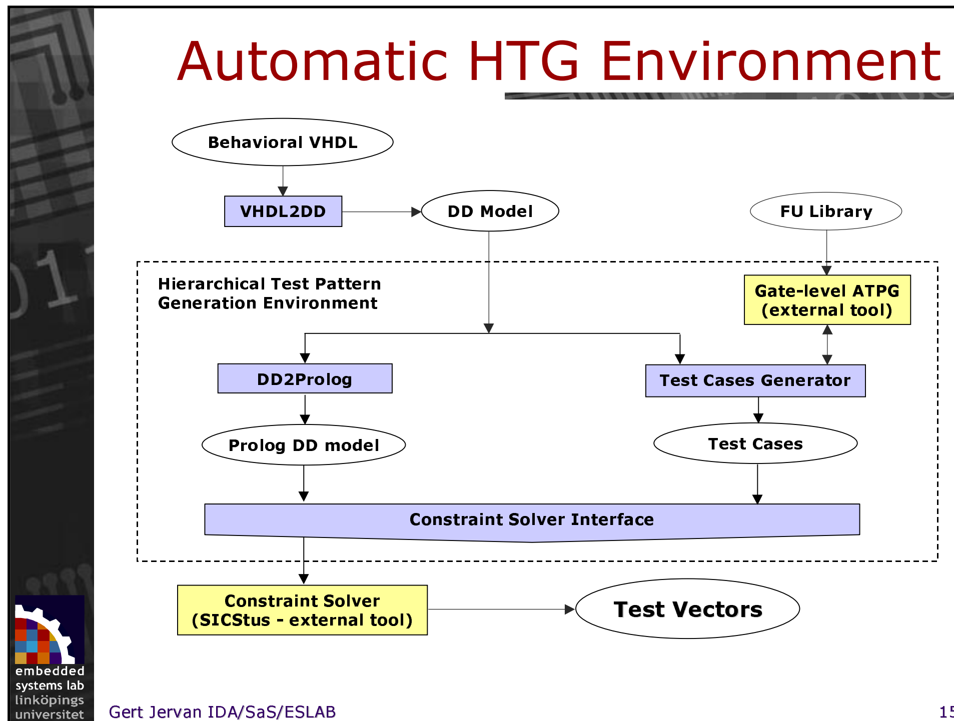
- All functional units (FUs) are tested by using their actual gate-level netlists and stuck-at fault model.
- Test pattern propagation and justification is carried out on a behavioral level
- The best possible test set under given constraints will be calculated
- Final fault coverage is measured on a structural level against stuck-at faults

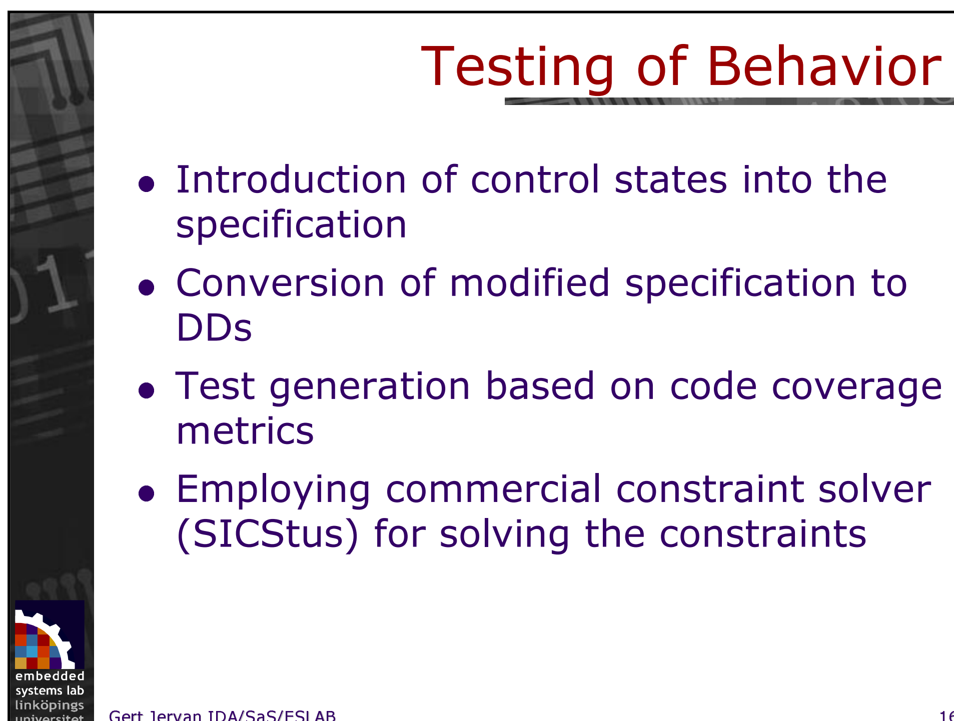
Gert Jervan IDA/SaS/ESLAB 13

Design Flow



Gert Jervan IDA/SaS/ESLAB 14


Automatic HTG Environment
Behavioral VHDL
VHDL2DD
DD Model
FU Library
Hierarchical Test Pattern
Generation Environment
Gate-level ATPG
(external tool)
DD2Prolog
Test Cases Generator
Prolog DD model
Test Cases
Constraint Solver Interface
Constraint Solver
(SICStus - external tool)
Test Vectors
embedded
systems lab
linköpings
universitet
Gert Jervan IDA/SaS/ESLAB
15


Testing of Behavior
Introduction of control states into the specification
Conversion of modified specification to DDs
Test generation based on code coverage metrics
Employing commercial constraint solver (SICStus) for solving the constraints
embedded
systems lab
linköpings
universitet
Gert Jervan IDA/SaS/ESLAB
16

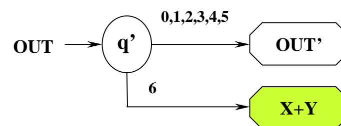
Testing FU-s

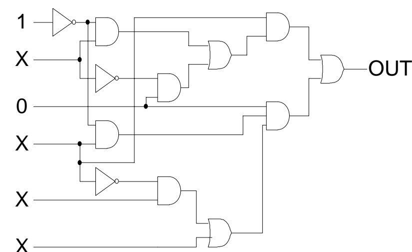
- Library of FU-s
  - VHDL models on a behavioral/structural level
  - DD models on a structural level
- PODEM algorithm for gate-level test generation (external tool)
- “Intelligent” method to handle X values and constants
- Justification and propagation is performed on a behavioral level

Testing FUs

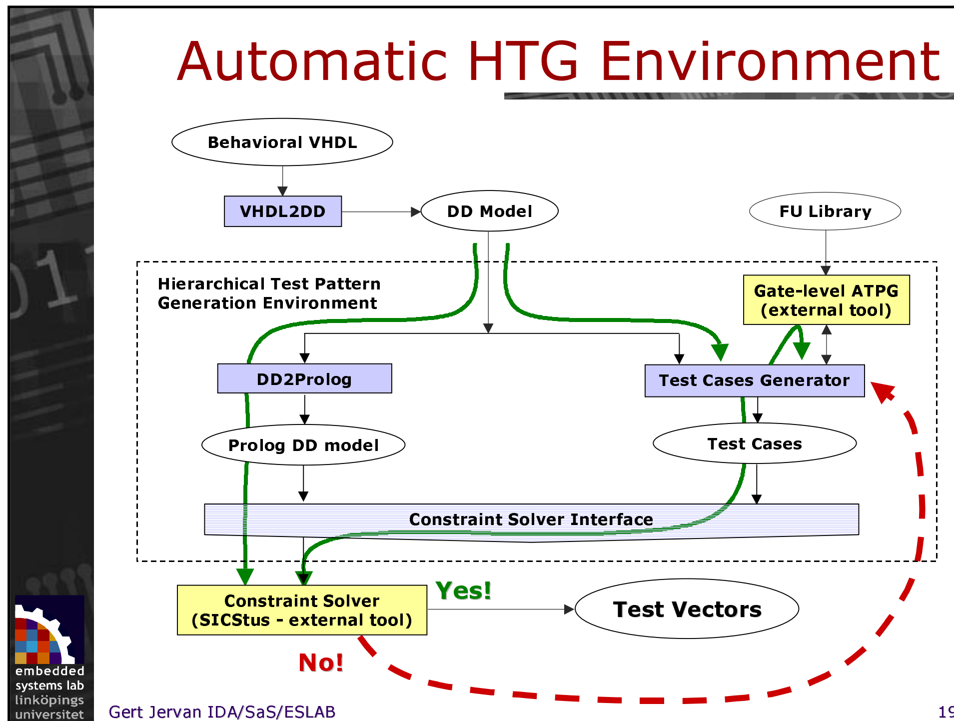
```
if (IN1 > 0)
    X=IN2+3;        --- q=1
else {
    if (IN2 >= 0)
        X=IN1+IN2; -- q=2
    else
        X=IN1*5;   --- q=3
}

Y=X-10;         -------- q=4
X=Y*2;          -------- q=5
OUT=X+Y;        -------- q=6
```

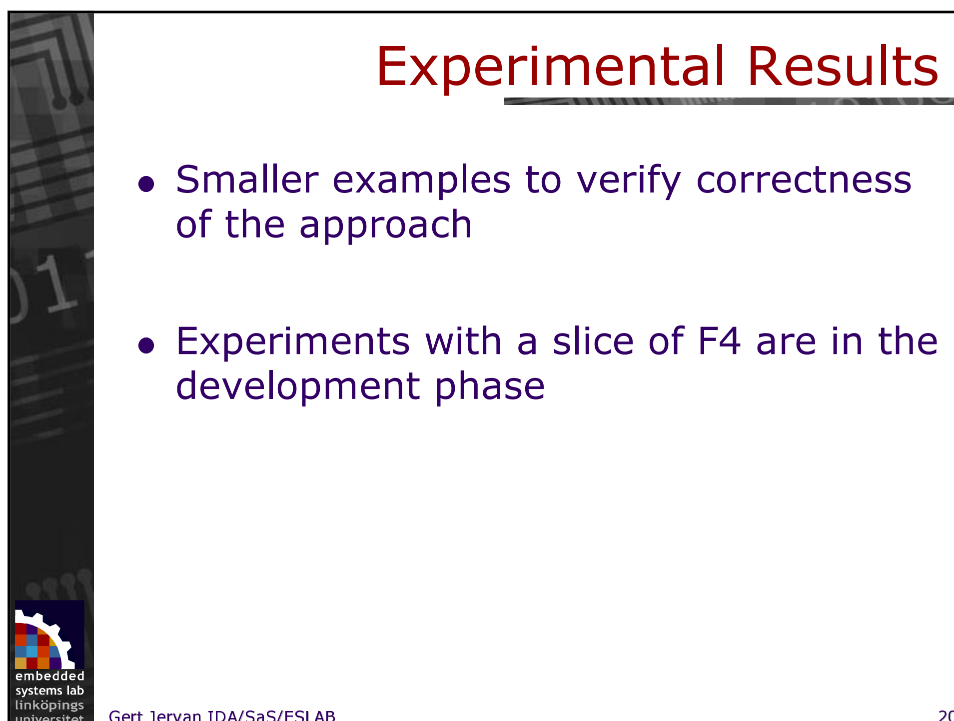
Behavioral Description





Gate-level netlist


Automatic HTG Environment
Behavioral VHDL
VHDL2DD
DD Model
FU Library
Hierarchical Test Pattern
Generation Environment
Gate-level ATPG
(external tool)
DD2Prolog
Test Cases Generator
Prolog DD model
Test Cases
Constraint Solver Interface
Constraint Solver
(SICStus - external tool)
Yes!
Test Vectors
No!
embedded
systems lab
linköpings
universitet
Gert Jervan IDA/SaS/ESLAB
19


Experimental Results
• Smaller examples to verify correctness of the approach
• Experiments with a slice of F4 are in the development phase
embedded
systems lab
linköpings
universitet
Gert Jervan IDA/SaS/ESLAB
20

Experimental Results

- DIFFEQ
  - # of lines (Behavioral VHDL): 59
  - # of gates: 4081
  - # of faults: 16222
  - # of untestable faults: 384
  - # of detected faults: 15277
  - Test generation time: 468.36 sec
  - # of unique test vectors: 199
  - # of applied test vectors: 20896 (includes vectors for reset etc.)
  - Fault efficiency: 96.46 % (gate-level stuck-at coverage)

Conclusions

- Hierarchical test generation environment for early design space exploration to evaluate testability
- Uses behavioral specifications for propagation and justification while retains gate-level accuracy
- Results are comparable with gate-level ATPG-s while our approach avoid synthesis step for testability evaluation
- Preliminary experimental results show feasibility of our approach

Thank You!