Application of CADP to Hardware Validation

Abderahman KRIOUILE and Massimo ZENDRI
STMicroelectronics

Forum Méthodes Formelles
"Le Model-Checking en action"

Toulouse, France, Oct 2014
• 20 years of Hardware Validation with CADP

• Presentation of hardware case studies

• Four Types of Studies
  • Formal Modeling
  • Functional Verification
  • Model-based Testing
  • Performance Evaluation

• Conclusion
20 Years of Hardware Validation with CADP
20 Years of Hardware Validation with CADP

High-level
- NovaScale/FAME
- Polykid
- SCSI-2
- FAME2
- CC-NUMA
- STBus SoC
- xSTream
- Platform2012
- AMBA ACE SoC
- Utah NoC

Low-level
- Supercomputers
- Multiprocessor
- On-Chip level
- Asynchronous logic


High-level vs. Low-level

1995: Supercomputers
2000: Multiprocessor
2005: On-Chip level
2015: Asynchronous logic

Application of CADP to Hardware Validation
20 Years of Hardware Validation with CADP

Application of CADP to Hardware Validation

Powerscale
- multiprocessor architecture based on PowerPC microprocessors used in Bull’s Escala servers and workstations

Supercomputers
Multiprocessor
On-Chip level
Asynchronous logic
20 Years of Hardware Validation with CADP

Polykid
- multiprocessor architecture based on PowerPC
- CC-NUMA memory model
- 2 cache coherency levels

Application of CADP to Hardware Validation
20 Years of Hardware Validation with CADP

Supercomputers

Multiprocessor

On-Chip level

Asynchronous logic

High-level

NovaScale/FAME

Polykid

Powerscale

SCSI-2

FAME2

CC-NUMA

Blitter Display

STBus SoC

2D Mesh

Low-level

1995

2000

2005

2010

2015

SCSI-2

- SCSI-2 bus arbitration protocol
- bus grant based on fixed priorities (SCSI numbers)
- unexpected OS deadlocks reported by Bull

Application of CADP to Hardware Validation 15/10/2014
20 Years of Hardware Validation with CADP

NovaScale/FAME
- 64-bit high-end servers based on Intel's Itanium-2
- CC-NUMA architecture
- focus on most critical, asynchronous parts

Supercomputers
Multiprocessor
On-Chip level
Asynchronous logic

High-level

Low-level

20 Years of Hardware Validation with CADP

STBus SoC
- STBus interconnect protocol
- dedicated to high bandwidth SoCs
- audio-video processing

Asynchronous logic
On-Chip level
Multiprocessor
Supercomputers
High-level
Low-level

Application of CADP to Hardware Validation 15/10/2014
20 Years of Hardware Validation with CADP

Application of CADP to Hardware Validation

- multiprocessor architectures
- CC-DSM: cache coherent-distributed shared memory
- MPI benchmark: ping-pong protocol
- performance prediction
20 Years of Hardware Validation with CADP

Application of CADP to Hardware Validation

- Supercomputers
- Multiprocessor
- On-Chip level
- Asynchronous logic

High-level
- NovaScale/FAME
- Polykid
- SCSI-2
- FAME2
- CC-NUMA
- Blitter Display
- STBus SoC

Low-level
- On-Chip level
- Supercomputers
- Multiprocessor
- Asynchronous logic

DES
- Data Encryption Standard
- asynchronous circuit
- no clock: gates evolve concurrently and synchronize via handshake protocols
- no constraints on communication delays


DES
20 Years of Hardware Validation with CADP

FAUST/MAGALI

- GALS architecture
- asynchronous NoC
- CHP (communicating Hardware Processes) model
20 Years of Hardware Validation with CADP

xSTream
- multiprocessor dataflow architecture
- high performance embedded multimedia streaming applications
- expected Performance measures:
  - latency
  - throughput
  - resource utilization

Low-level

High-level

Application of CADP to Hardware Validation
20 Years of Hardware Validation with CADP

Blitter Display

- MULTIVAL project
- 2D graphics co-processor implementing BLIT (Block Image Transfer) and numerous graphical operators
- SystemC/TLM model

- LOTOS
- CPU
- SDimg
- VTG

Functions extracted from the SDimg SystemC/TLM model

Low-level

20 Years of Hardware Validation with CADP

2D Mesh NoC

- 5x5 2D-mesh NoC
- Predict mean latency of end-to-end communication

Low-level

20 Years of Hardware Validation with CADP

Platform2012 DTD

- Dynamic Task Dispatcher
- tasks divided in concurrently executable sub-tasks (same code, different data)
- dedicated hardware to switch tasks in only few clock cycles

Low-level

20 Years of Hardware Validation with CADP

Utah NoC

- two-dimensional mesh
- routing algorithm tolerating link faults
- check absence of deadlocks

Low-level

20 Years of Hardware Validation with CADP

AMBA ACE SoC

- heterogeneous SoC
- ACE protocol: system level cache coherency standard
- support for ARM®@Big.LITTLE™
- integrated to STMicro set top box SoC for multiple Ultra HD

Application of CADP to Hardware Validation
Four Types of Studies

• Formal Modeling
• Functional Verification
• Model-based Testing
• Performance Evaluation
Formal Modeling

• Modeling languages used in these case studies
  • Before 2008-2009: LOTOS
  • Since then: LNT

• LOTOS vs LNT
  • Both are formal languages to describe asynchronously-concurrent systems
  • LNT more convenient for human users
  • LNT closer to programming languages and hardware languages (such as VHDL)

• Starting point for producing formal models:
  • Natural language descriptions (English text, tables, diagrams)
  • Programs in other hardware languages (CHP, SystemC/TLM, etc.)

• Guidelines must be followed when developing formal models:
  • Focus on complex parts of the system (parallelism, concurrency, etc.)
  • Use abstractions to hide irrelevant details
Some figures about modeling effort in past projects

<table>
<thead>
<tr>
<th>Case study</th>
<th>Company</th>
<th>Level</th>
<th>Modeling size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powerscale</td>
<td>Bull</td>
<td>system</td>
<td>720 lines of LOTOS</td>
</tr>
<tr>
<td>Polykid</td>
<td>Bull</td>
<td>system</td>
<td>4000 lines of LOTOS (model)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2000 lines of LOTOS (rules)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3,400 lines of LOTOS and 7,000 lines of C (emulation)</td>
</tr>
<tr>
<td>SCSI-2</td>
<td>Bull</td>
<td>system</td>
<td>220 lines of LOTOS</td>
</tr>
<tr>
<td>FAME1/CCS</td>
<td>Bull</td>
<td>system</td>
<td>1200 lines of LOTOS</td>
</tr>
<tr>
<td>FAME1/NCS</td>
<td>Bull</td>
<td>system</td>
<td>1200 lines of LOTOS</td>
</tr>
<tr>
<td>FAME1/B-SPS/FSS</td>
<td>Bull</td>
<td>system</td>
<td>5000 lines of LOTOS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4500 lines of LOTOS</td>
</tr>
<tr>
<td>Case study</td>
<td>Company</td>
<td>Level</td>
<td>Modeling size</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------</td>
<td>-------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>FAME1/ILU</td>
<td>Bull</td>
<td>unit</td>
<td>8900 lines of LOTOS 3400 lines of C</td>
</tr>
<tr>
<td>FAME1/PRR</td>
<td>Bull</td>
<td>block</td>
<td>7500 lines of LOTOS 200 lines of C</td>
</tr>
<tr>
<td>CC-NUMA</td>
<td>Bull</td>
<td>system</td>
<td>1800 lines of LOTOS 1000 lines of Murphi</td>
</tr>
<tr>
<td>DES</td>
<td>CEA-Leti/TIMA</td>
<td>unit</td>
<td>1700 lines of CHP 3800 lines of LOTOS</td>
</tr>
<tr>
<td>FAME2/PAB</td>
<td>Bull</td>
<td>block</td>
<td>3977 lines of LNT</td>
</tr>
<tr>
<td>FAUST/MAGALI</td>
<td>CEA-Leti</td>
<td>system</td>
<td>1200 lines of CHP</td>
</tr>
<tr>
<td>xStream</td>
<td>ST</td>
<td>unit</td>
<td>6800 lines of LOTOS</td>
</tr>
</tbody>
</table>
### Case study

<table>
<thead>
<tr>
<th>Case study</th>
<th>Company</th>
<th>Level</th>
<th>Modeling size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blitter Display</td>
<td>ST</td>
<td>block</td>
<td>5550 lines of SystemC/TLM, 920 lines of LOTOS, 2250 lines of C</td>
</tr>
<tr>
<td>Platform2012/HWS</td>
<td>ST</td>
<td>unit</td>
<td>300 lines of LNT</td>
</tr>
<tr>
<td>Platform2012/DTD</td>
<td>ST</td>
<td>block</td>
<td>1200 lines of LNT</td>
</tr>
<tr>
<td>Utah NoC</td>
<td>Univ. of Utah</td>
<td>system</td>
<td>1350 lines of LNT</td>
</tr>
<tr>
<td>AMBA ACE SoC</td>
<td>ST/ARM</td>
<td>system</td>
<td>3400 lines of LNT (model), 990 lines of LNT (checks)</td>
</tr>
</tbody>
</table>
Formal Modeling

• Detect ambiguities
  • The initial specification is usually not formal
  • Many problems are discovered just by modeling, before running any tool
  • Formal specification triggers discussions with architects

• Debugging the model
  • Remove errors introduced during modeling
  • Architects are not interested in false positives

• How?
  • Compile with CADP tools
  • Simulate step by step with the OCIS simulator
  • Check simple properties (absence of deadlocks, etc.)
Functional Verification

• Looking for “real” bugs in the specification (and not in the model)

• Need to formalize the properties
  • Equivalence checking: properties expressed in the same language as the model (LOTOS, LNT, etc.)
  • Model checking: properties expressed in a dedicated languages (MCL, XTL, etc.)
  • A new source of bugs
  • How to debug properties?

• At some point, good confidence is reached in both the model and the properties

• Then, if a verification reports an error, it can be
  • Either an error in the verification tool (rare, to be fixed by tool developers)
  • Or a “real” bug in the specification is detected
# Functional Verification Results

<table>
<thead>
<tr>
<th>Case study</th>
<th>Functional Verification Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Powerscale</td>
<td>Hidden bug found in a few minutes FORTE'96 [Chehaibar-Garavel-Mounier-Tawbi-Zulian-96]</td>
</tr>
</tbody>
</table>
| Polykid      | Phase 1: 55 questions  
Phase 2: 20 questions, 7 serious issues  
Phase 3: 13 serious issues  
IWTCS'98 [Kahlouche-Viho-Zendri-98] |
| SCSI-2       | SCSI-2 bus arbiter starvation problem confirmed (avoided in SCSI-3 standard) |
| FAME         | Critical parts of FAME design verified using CADP  
10 issues raised, 2 ambiguities pointed out |
| STBus SoC    | Error in the design discovered  
MEMOCODE'03 [Wodey-Camarroque-Baray-et-al-03] |
| FAME2 / MPI  | Formally verified               |
# Functional Verification Results

<table>
<thead>
<tr>
<th>Case study</th>
<th>Functional Verification Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAUST/MAGALI</td>
<td>Routing problem detected in the CHP description ASYNC’07 [Salaum-Serwe-Thonnart-Vivet-07]</td>
</tr>
</tbody>
</table>
| Blitter Display     | Avoids complete translation of SystemC/TLM to LOTOS:  
|                     | - reduced translation effort  
|                     | - better integration of formal verification in the design flow MEMOCODE’09 [Garavel-Helmstetter-Ponsini-Serwe-09]                                                                                 |
| xSTream             | Two design issues detected very early                                                                                                                                                             |
| Platform2012/DTD     | Problematic configurations with livelocks found  
|                     | Further investigation by co-simulation FMICS’11 [Lantreibecq-Serwe-11]                                                                                                                           |
| AMBA ACE SoC        | Reproduction of a known bug of a previous specification  
|                     | “Proof” that the protocol is valid FMICS’13 [Kriouile-Serwe-13]                                                                                                                             |
| Utah NoC            | Found flaws in the original arbiter design FMICS’14 [Zhang-Serwe-Wu-et-al-14]                                                                                                                  |
Model-based Testing

• Offline approach: Test Generation
  • Step 1: generate test cases
  • Step 2: run test cases on the implementation

• Online approach: Co-simulation
  • Mutual cross-check between the model and the implementation

• Coverage-oriented methods
  • Use coverage metrics to generate tests
  • Can be applied offline or online

• Emulation
  • Replacement of a hardware component by a software program generated from a LOTOS/LNT model
## Model-based Testing Results

<table>
<thead>
<tr>
<th>Case study</th>
<th>Functional Verification Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polykid/Test generation</td>
<td>5 new bugs discovered in VHDL design IWTCS'98 [Kahlouche-Viho-Zendri-98]</td>
</tr>
<tr>
<td>Polykid/Emulation</td>
<td>Replacement of a missing ASIC by a software emulation running on a PowerPC microprocessor STTT’01 [Garavel-Viho-Zendri-01]</td>
</tr>
<tr>
<td>FAME/CCS</td>
<td>Directed test generation using TGV 21 base tests (1 mn per test) 50 collision tests (15 mn per test) 1 generalized test (1 day)</td>
</tr>
<tr>
<td>FAME/NCS</td>
<td>Directed test generation using TGV 50 base tests (30 sec per test)</td>
</tr>
<tr>
<td>FAME/PRR</td>
<td>Random test generation using Executor Detection of a non-conformity between LOTOS and Verilog codes for PRR v1 (not detected using commercial tools)</td>
</tr>
<tr>
<td>Case study</td>
<td>Functional Verification Results</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FAME/ILU</td>
<td>Co-simulation using Exec/Caesar</td>
</tr>
<tr>
<td>FAME/B-SPS/FSS</td>
<td>Trace validation with coverage Major bug found (ambiguity in informal specification) Insufficient coverage found (3 missing tests added) SPIN'04 [Garavel-Mateescu-04]</td>
</tr>
<tr>
<td>FAUST/MAGALI</td>
<td>Co-simulation: LOTOS-SystemC / VHDL netlist Detection of spurious inputs generated by LOTOS model: Constraints added to generate only valid inputs</td>
</tr>
<tr>
<td>Plateform2012/DTD</td>
<td>Co-simulation: C++ / LNT Found C++ incorrect for some particular scenarios Science of Computer Prog. [Lantreibecq-Serwe-14]</td>
</tr>
<tr>
<td>AMBA ACE SoC</td>
<td>Model-based test generation using counterexamples targeted at corner cases Early detection of 10 errors in commercial verification IPs</td>
</tr>
</tbody>
</table>
Performance Evaluation

• High degree of concurrency
  • Communication latencies may appear
  • Time constraints have to be respected

• Quantitative issues occurring with high degree of concurrency

• Advantage of CADP
  • Both qualitative and quantitative aspects studied on the same formal model

• Formalisms used
  • CTMCs (Continuous-Time Markov Chains)
  • IMCs (Interactive Markov Chains)
  • IPCs (Interactive Probabilistic Chains)
## Performance Evaluation Results

<table>
<thead>
<tr>
<th>Case study</th>
<th>Formalism</th>
<th>Functional Verification Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSI-2</td>
<td>IMCs</td>
<td>Steady-state analysis suggested strategies to avoid starvation and increase throughput FME’02 [Garavel-Hermanns-02]</td>
</tr>
</tbody>
</table>
| FAME2 / MPI      | IMCs      | Numerical prediction were close to experimental measures:  
|                  |           | - Estimation of the number of caches misses  
|                  |           | - Selection of the most performant configuration QuEST’09 [Chehaiber-Zidouni-Mateescu-09]                                                                   |
| xStream          | IPCs      | Prediction of latencies, throughputs, and queue occupancy CAV’09 [N.Coste’PhD thesis]                                                                       |
| 2D Mesh NoC      | CTMCs     | Results were close (< 5%) to SystemC CABA simulation IPDPSW’10 [Foroutan-Thonnart-Hersemeule-Jerraya-10]                                                        |
Conclusion

• CADP has been applied to many different hardware problems

• Formal modelling requires expertise and can be time-consuming
  • Often, the first model is not the best, and several iterations are required
  • Knowledge and experience must be capitalized
  • Once the model exists, it can be profitably exploited in multiple ways

• Functional verification and model-based testing are effective
  • Non-trivial issues ("high quality bugs") are often detected
  • Limitations in scalability due to state-explosion problem
  • Focus on the most complex parts, and use appropriate abstractions
  • Use “clever” verification strategies, such as compositional verification

• Performance evaluation is industrially relevant
  • CADP enables one to use similar models for functional verification and performance evaluation
  • Quantitative analyses allow design-space exploration very early in the development flow