Application of CADP to Hardware Validation

Abderahman KRIOUILE and Massimo ZENDRI
STMicroelectronics

Forum Méthodes Formelles
"Le Model-Checking en action"

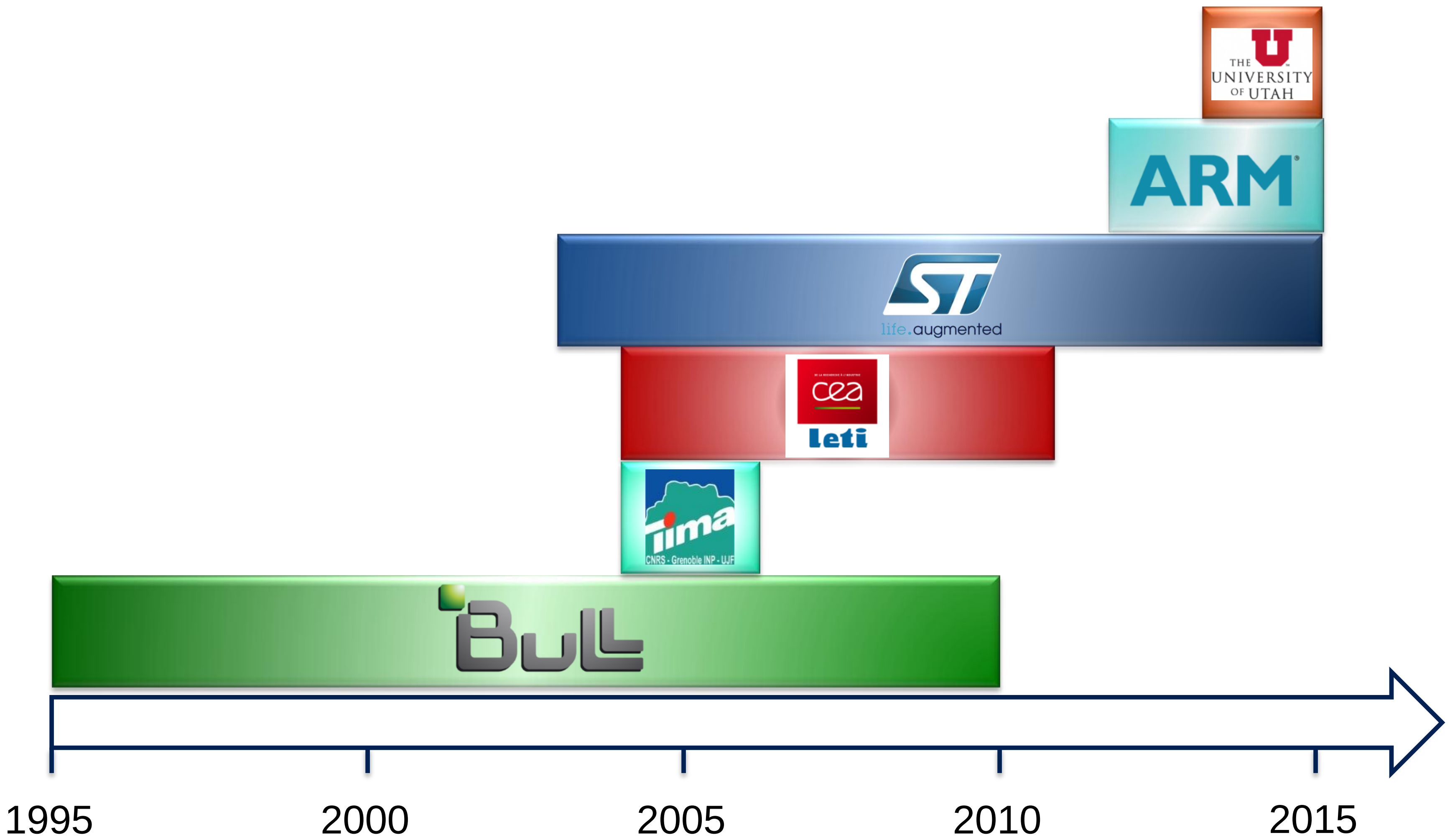
Toulouse, France, Oct 2014

life.augmented

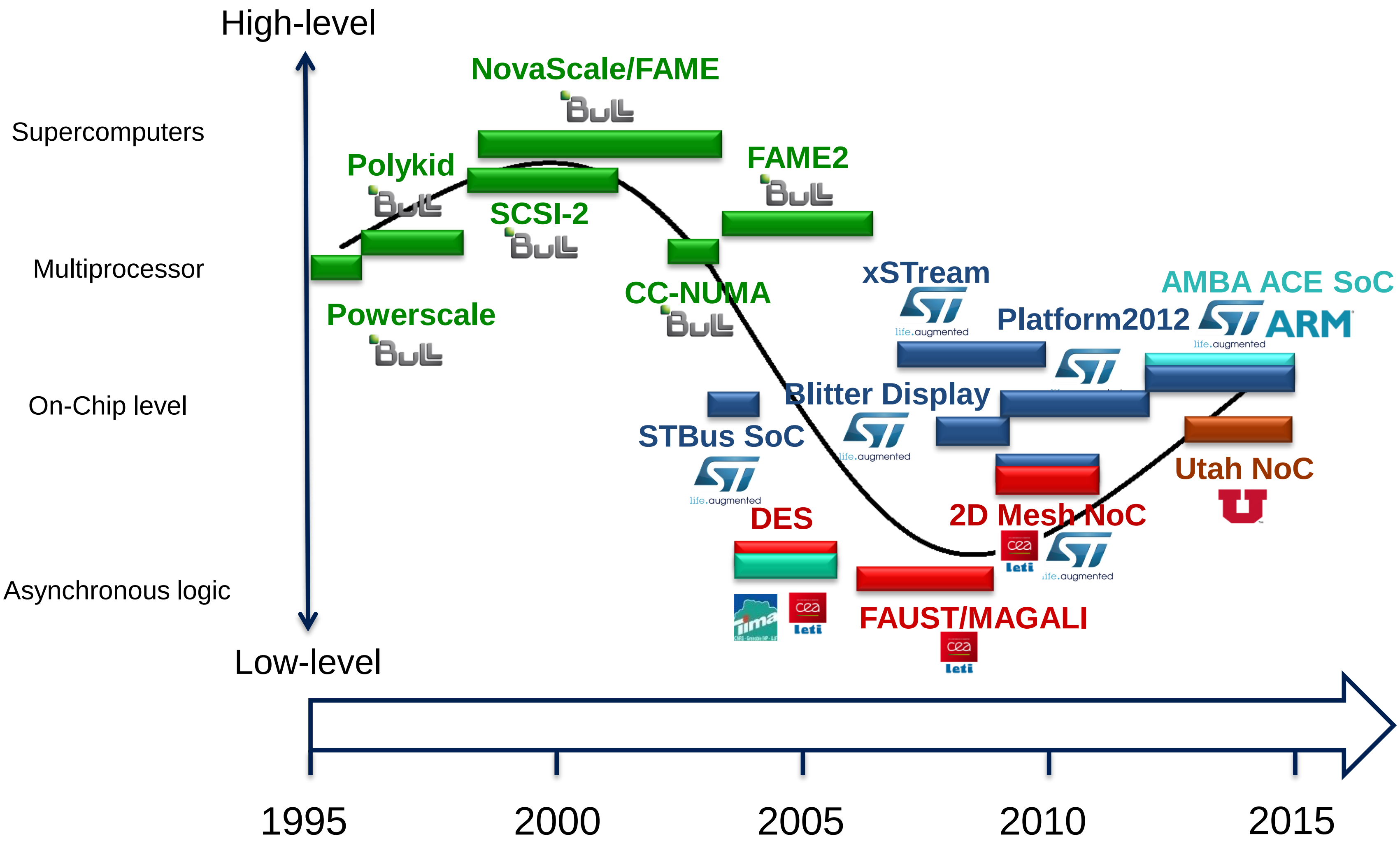
Agenda

- **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**

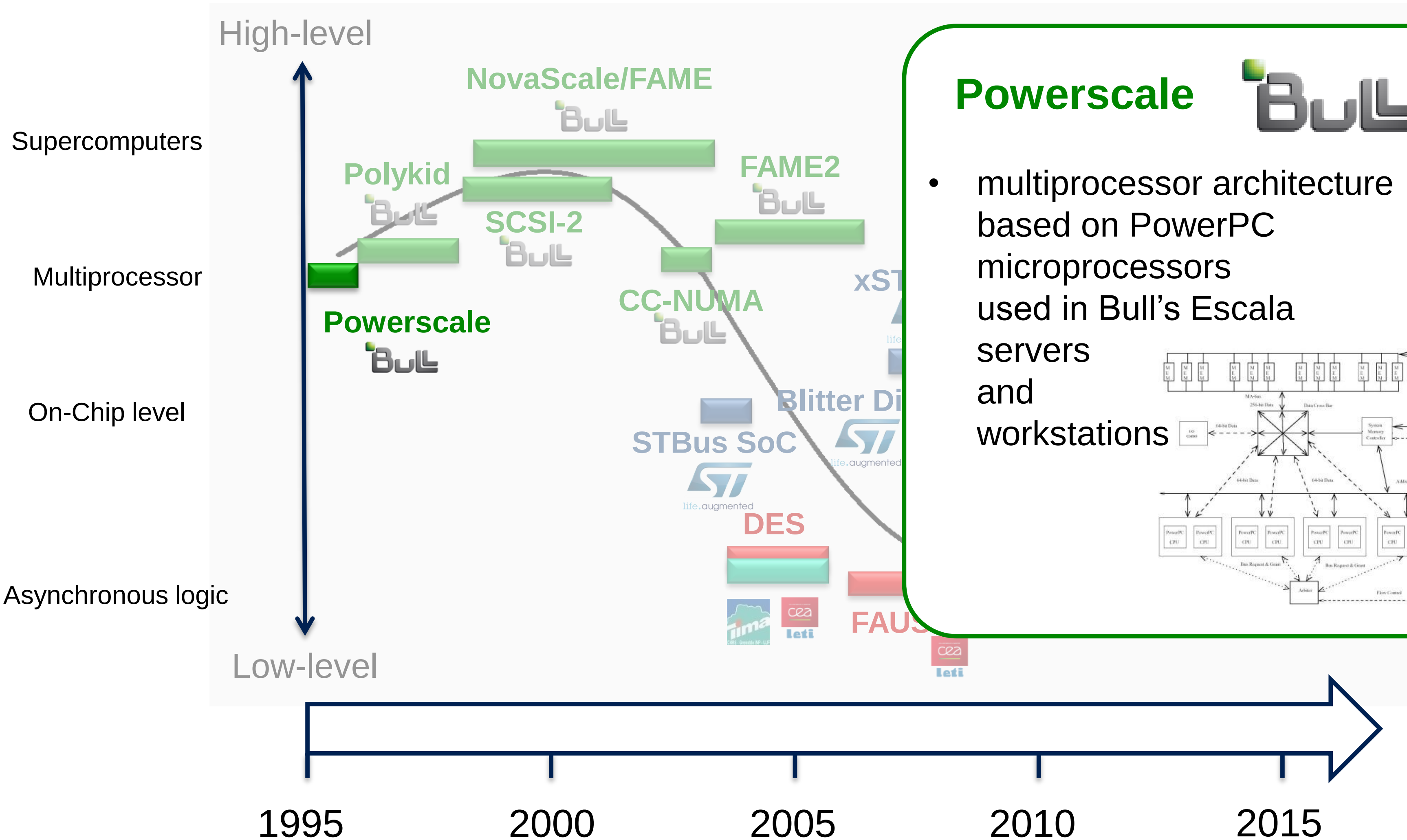
20 Years of Hardware Validation with CADP



20 Years of Hardware Validation with CADP



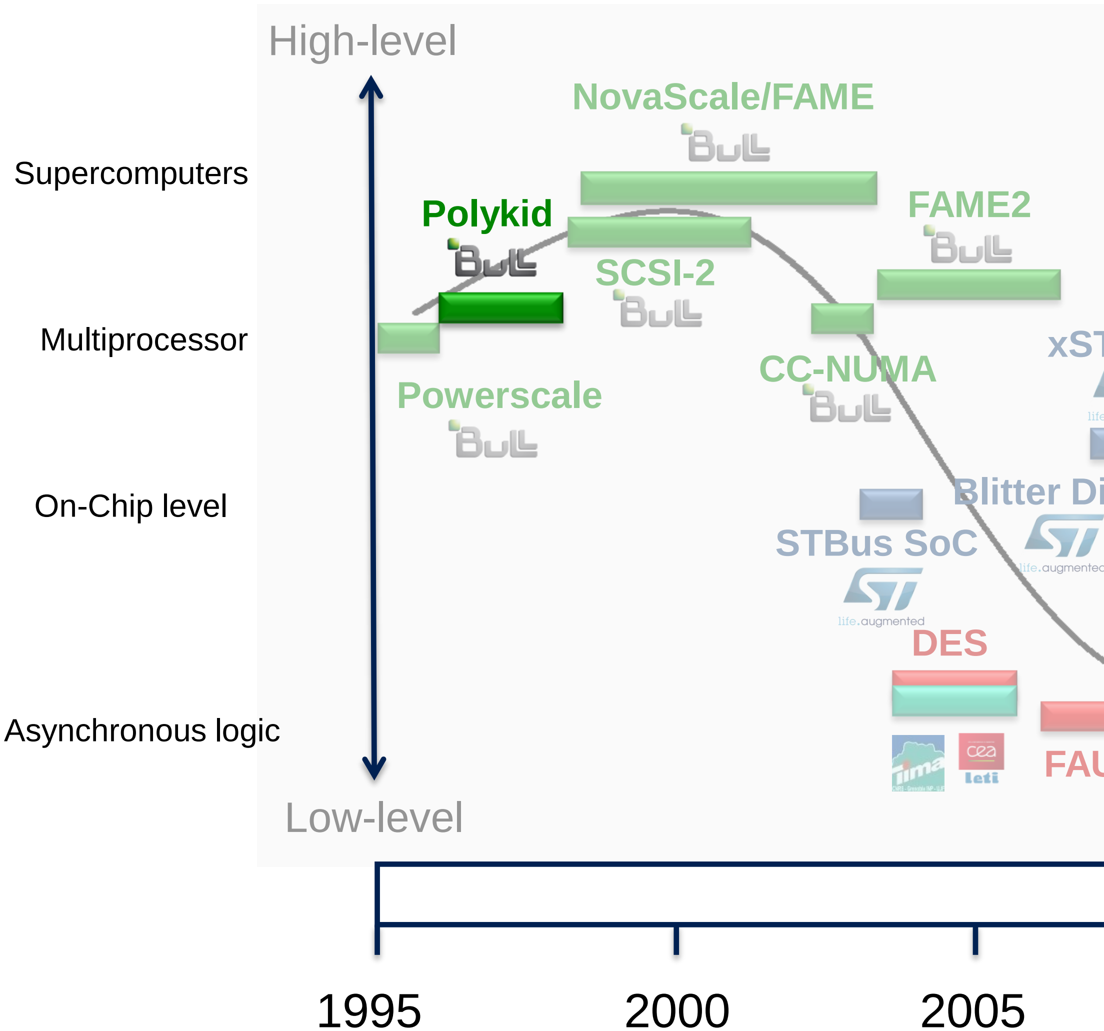
20 Years of Hardware Validation with CADP



Powerscale

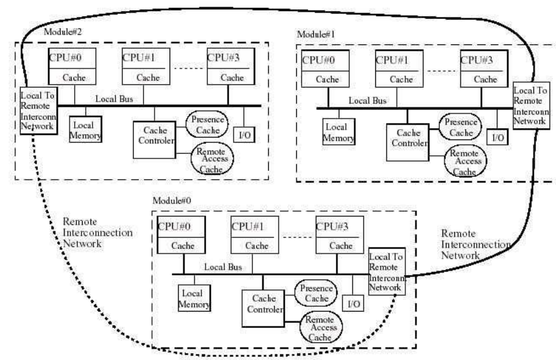
- multiprocessor architecture based on PowerPC microprocessors used in Bull's Escala servers and workstations

20 Years of Hardware Validation with CADP

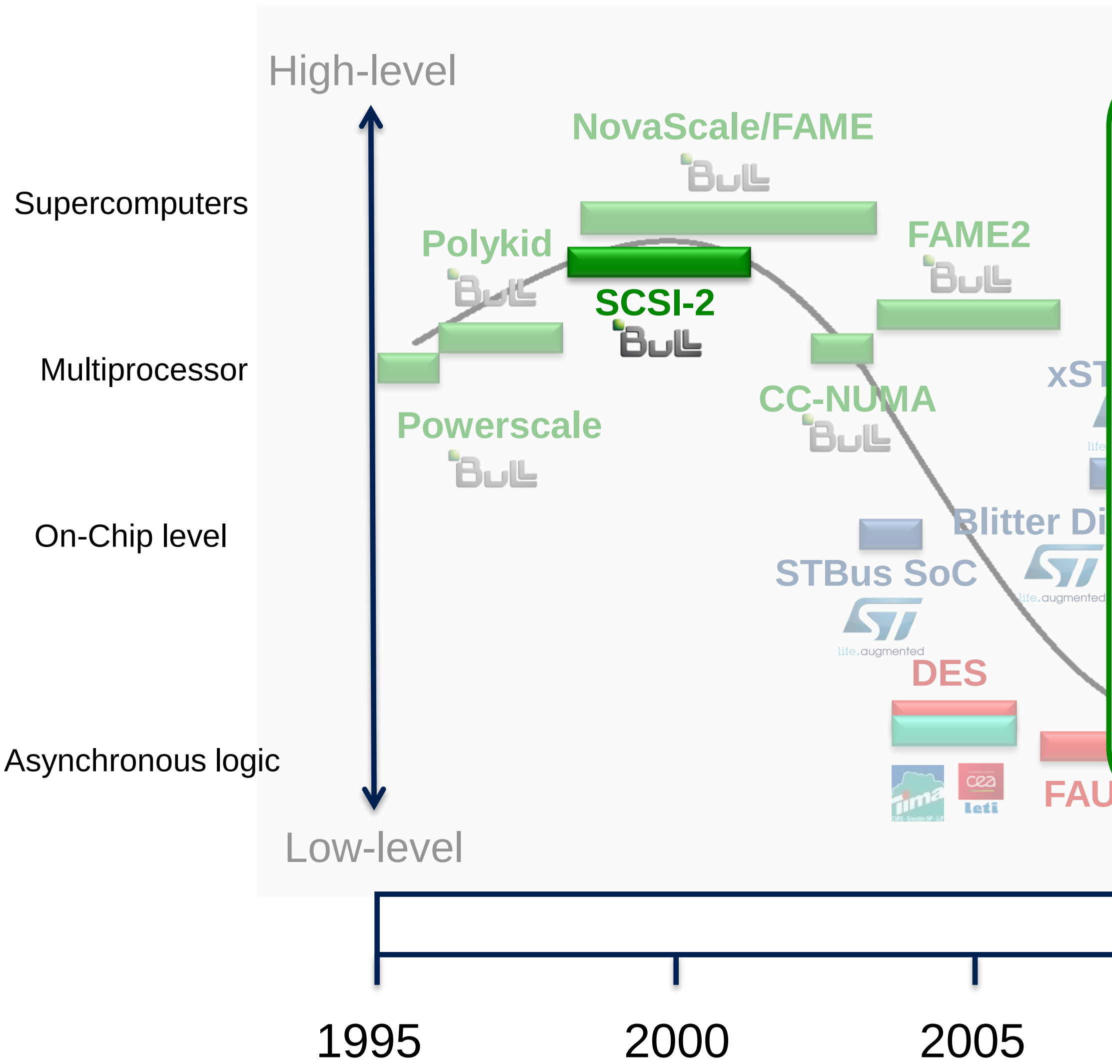


Polykid

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

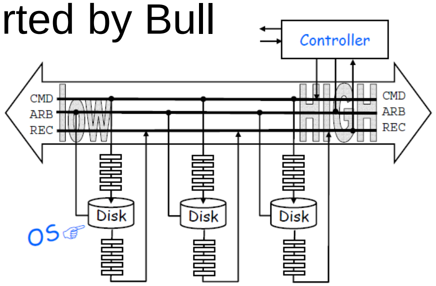


20 Years of Hardware Validation with CADP

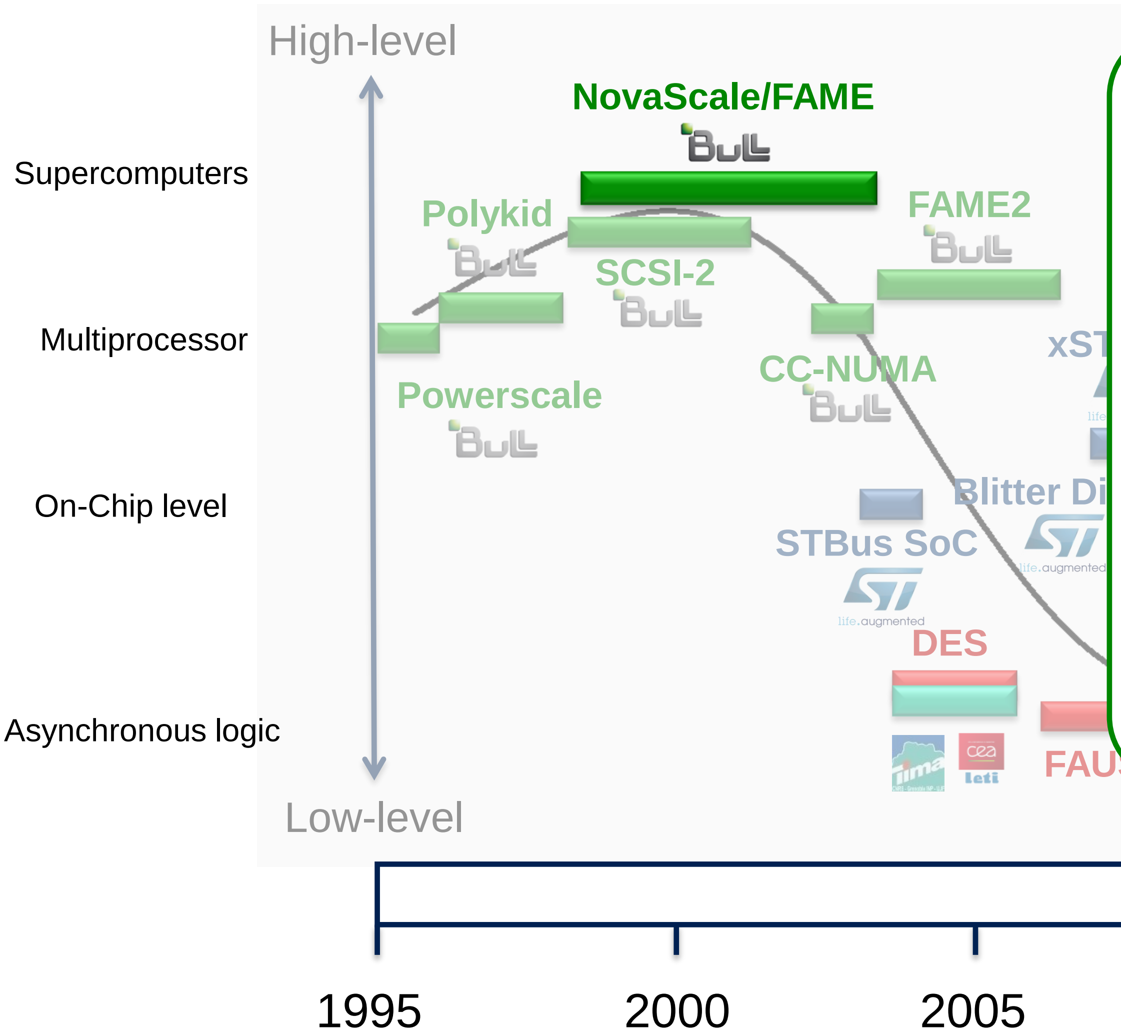


SCSI-2

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

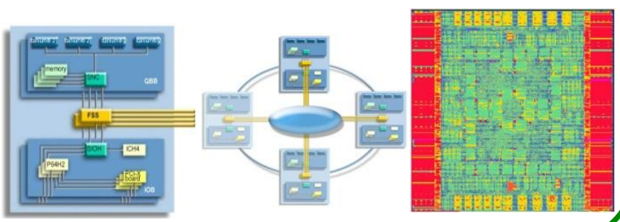


20 Years of Hardware Validation with CADP

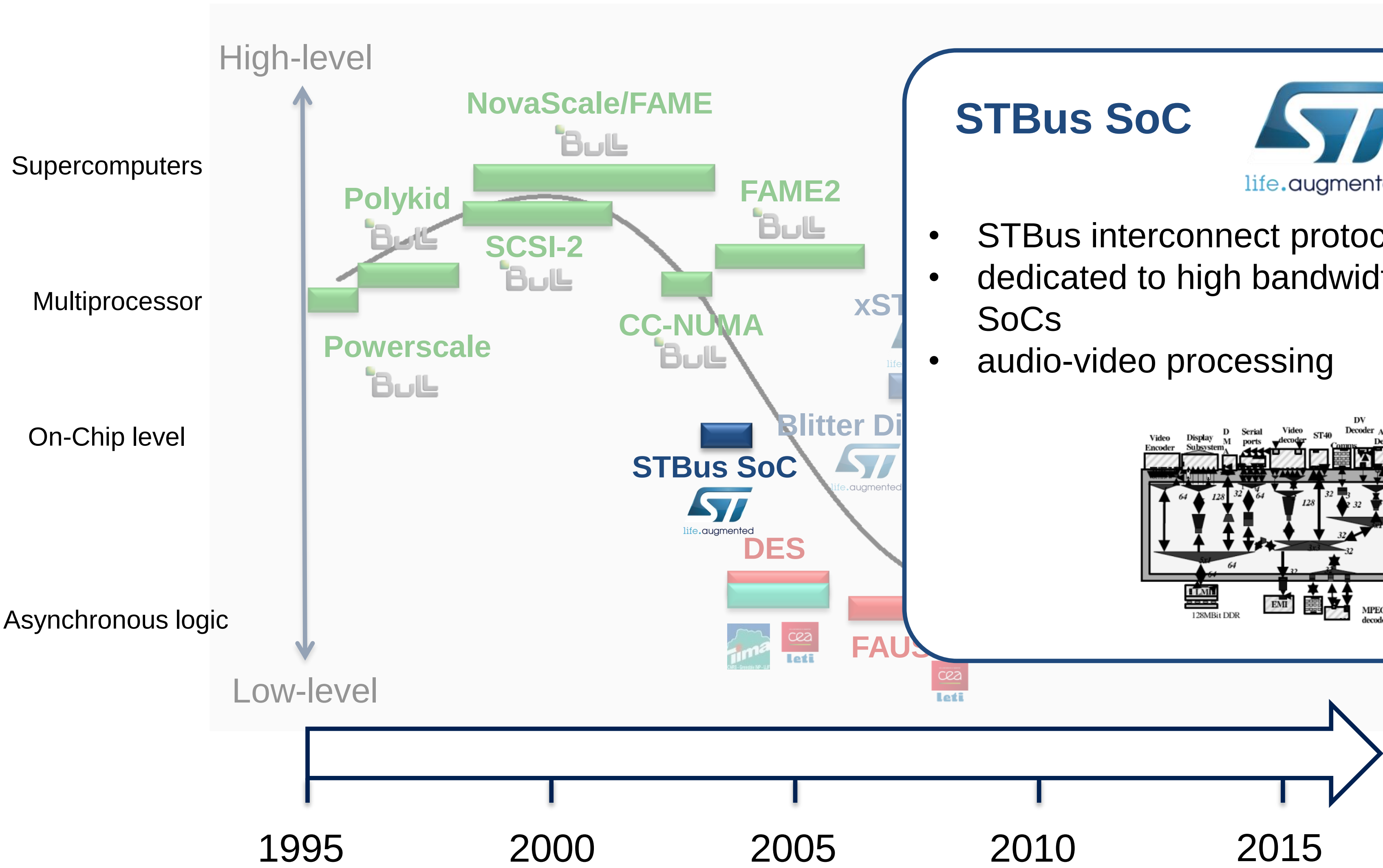


NovaScale/FAME

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



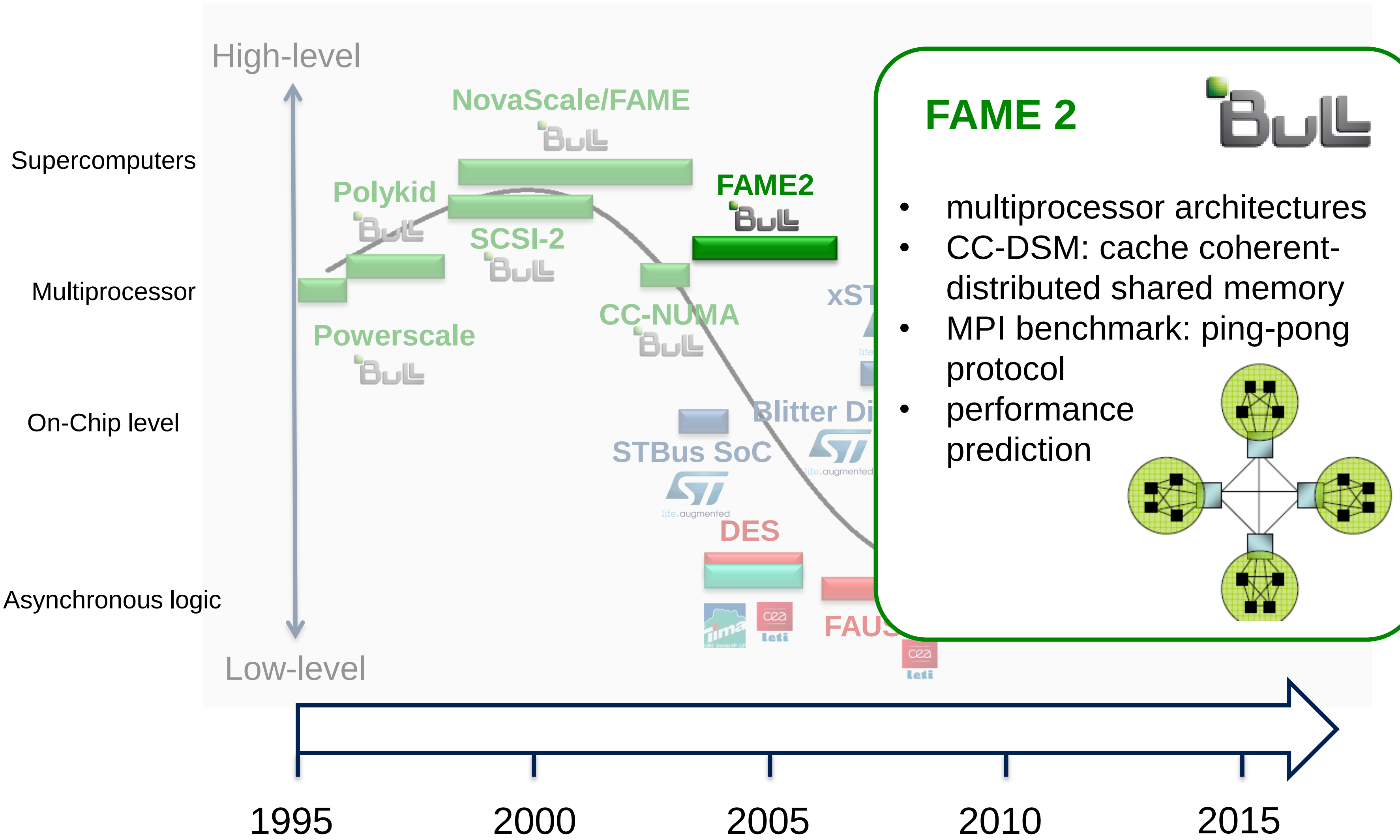
20 Years of Hardware Validation with CADP



STBus SoC

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

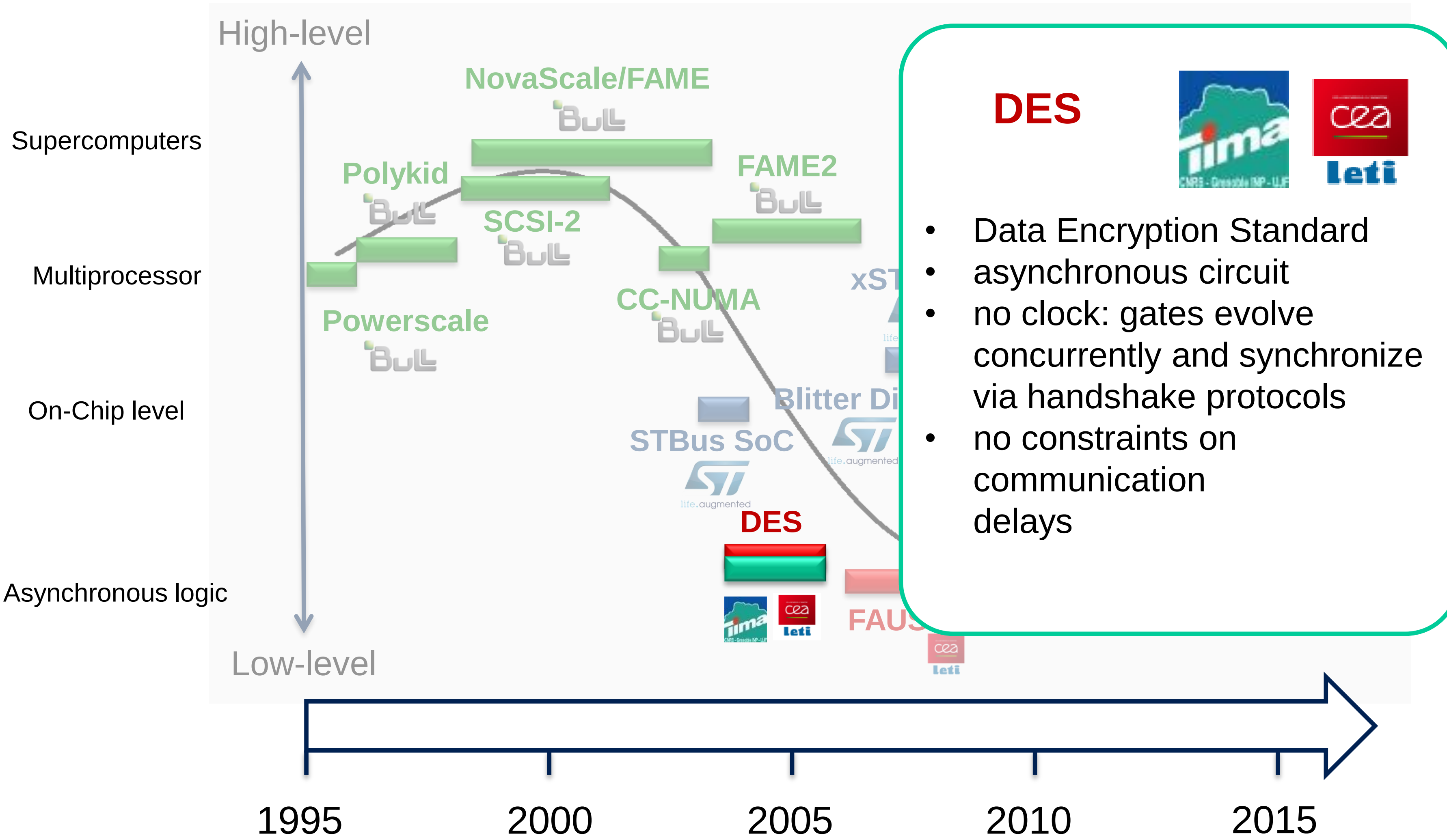
20 Years of Hardware Validation with CADP



FAME 2

- multiprocessor architectures
- CC-DSM: cache coherent-distributed shared memory
- MPI benchmark: ping-pong protocol
- performance prediction

20 Years of Hardware Validation with CADP



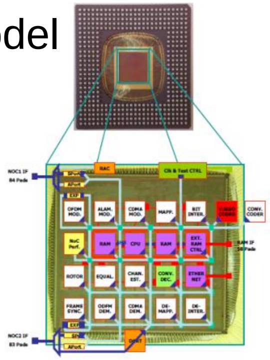
DES

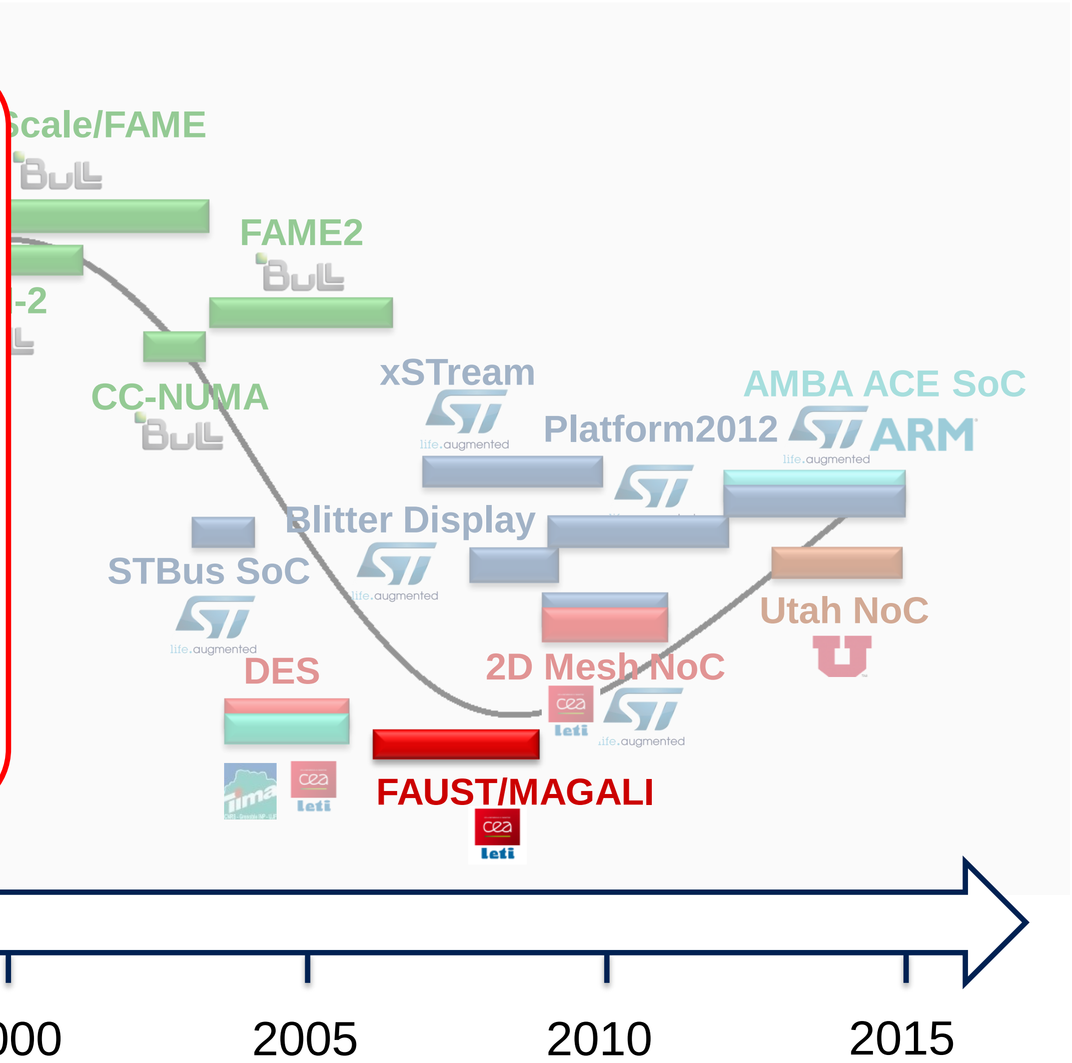
- Data Encryption Standard
- asynchronous circuit
- no clock: gates evolve concurrently and synchronize via handshake protocols
- no constraints on communication delays

20 Years of Hardware Validation with CADP

FAUST/MAGALI

- GALS architecture
- asynchronous NoC
- CHP (communicating Hardware Processes) model

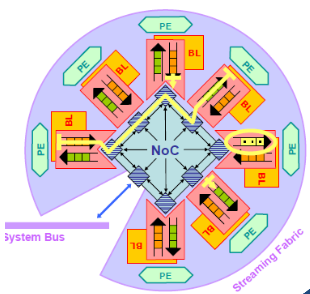


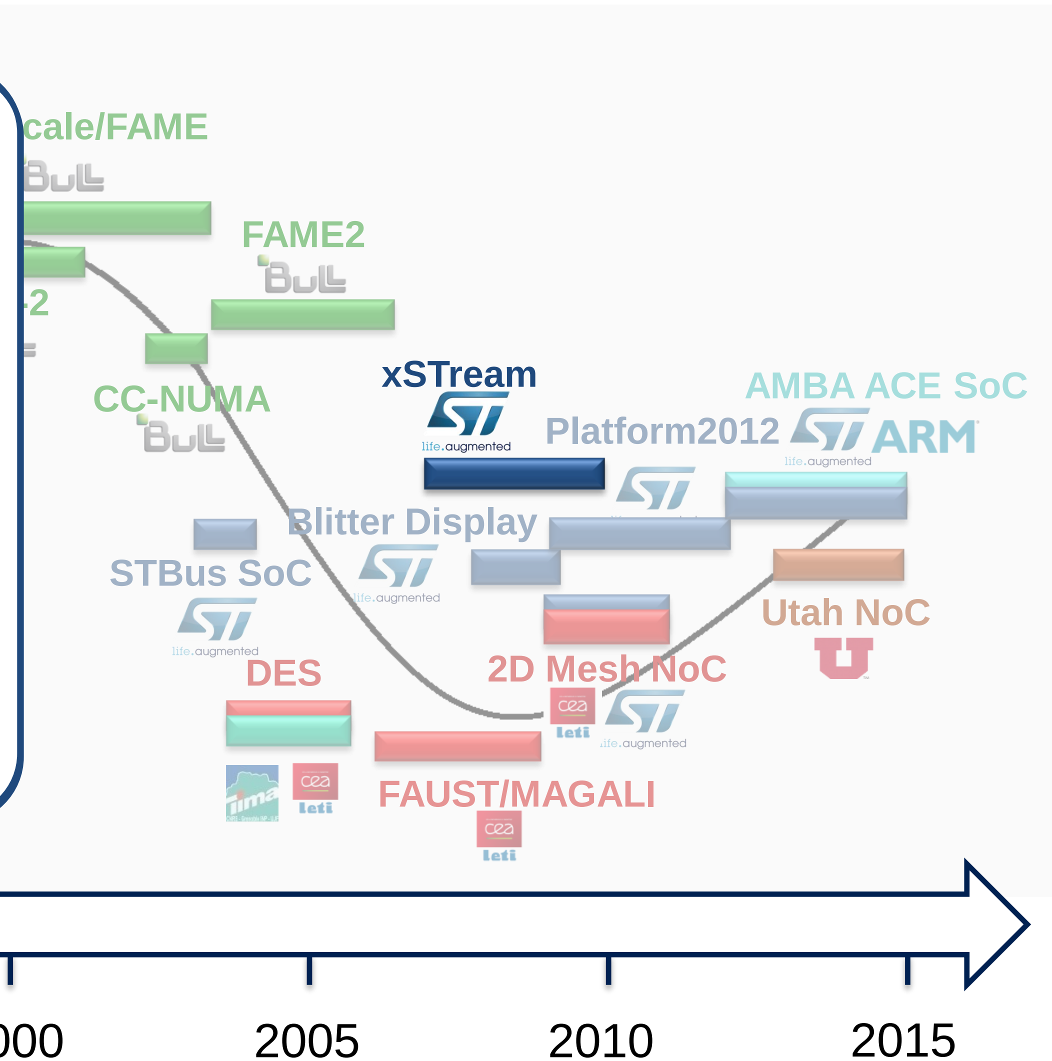


20 Years of Hardware Validation with CADP

xSTream

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

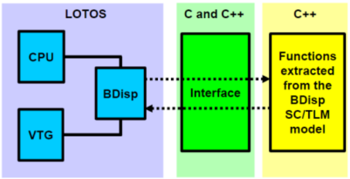


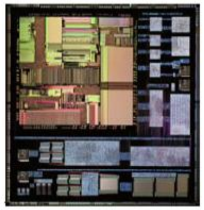


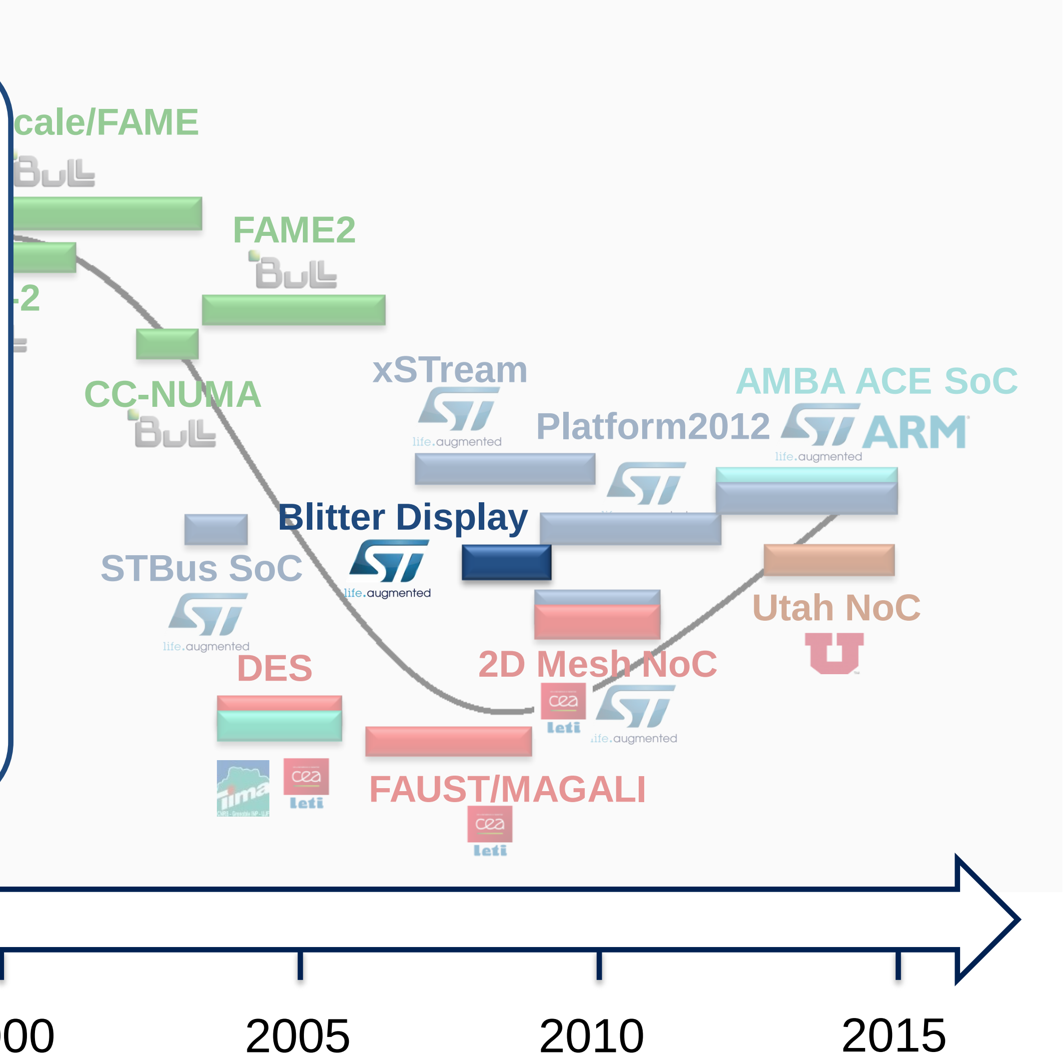
20 Years of Hardware Validation with CADP

Blitter Display

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



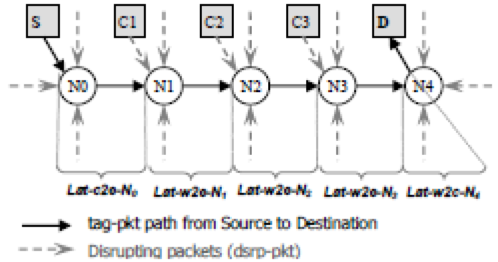


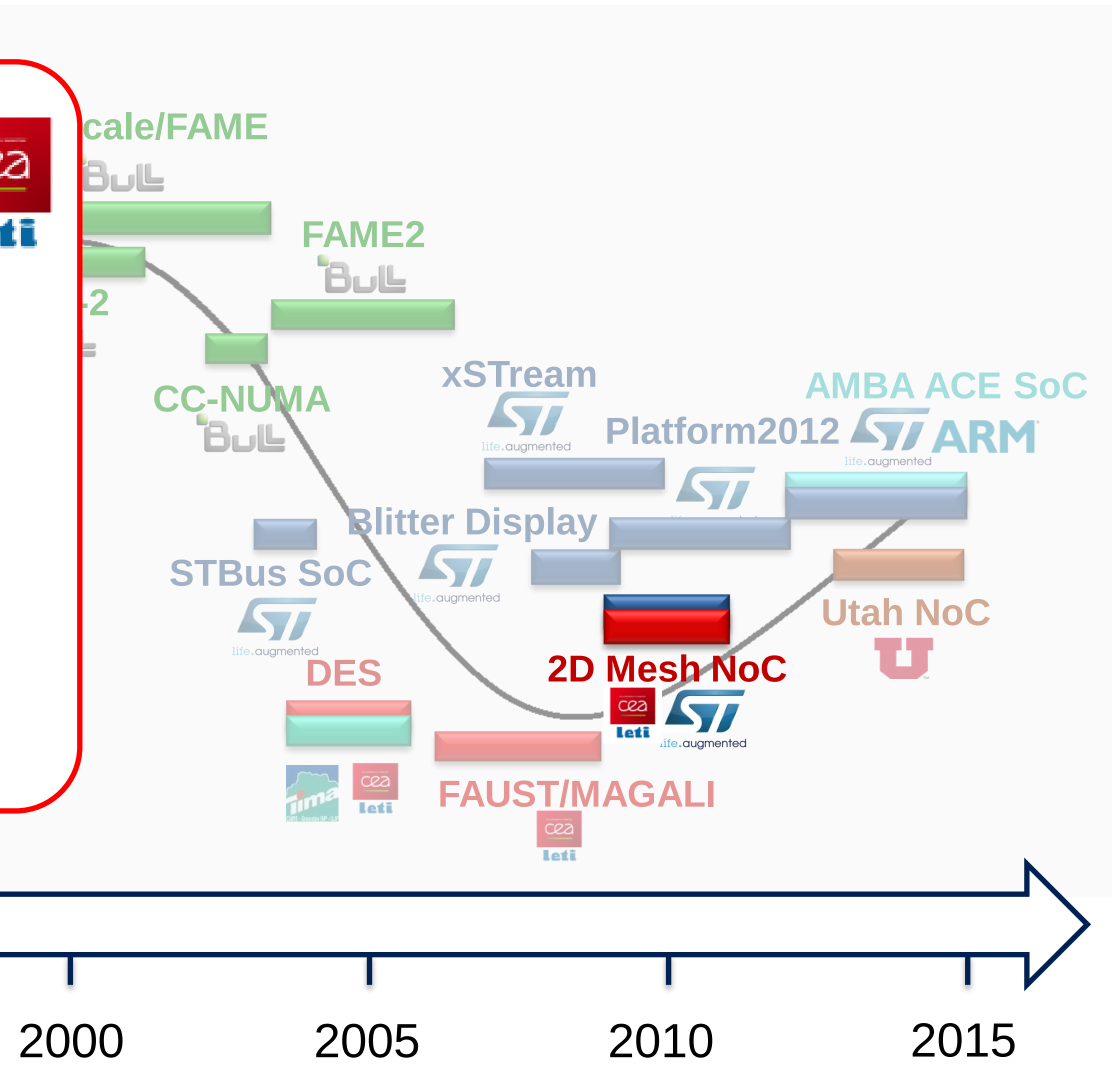


20 Years of Hardware Validation with CADP

2D Mesh NoC

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

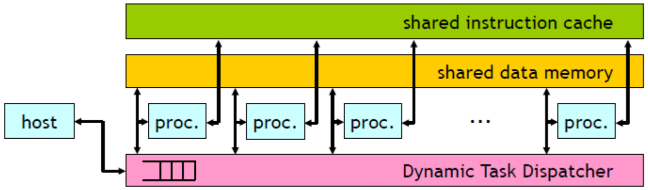


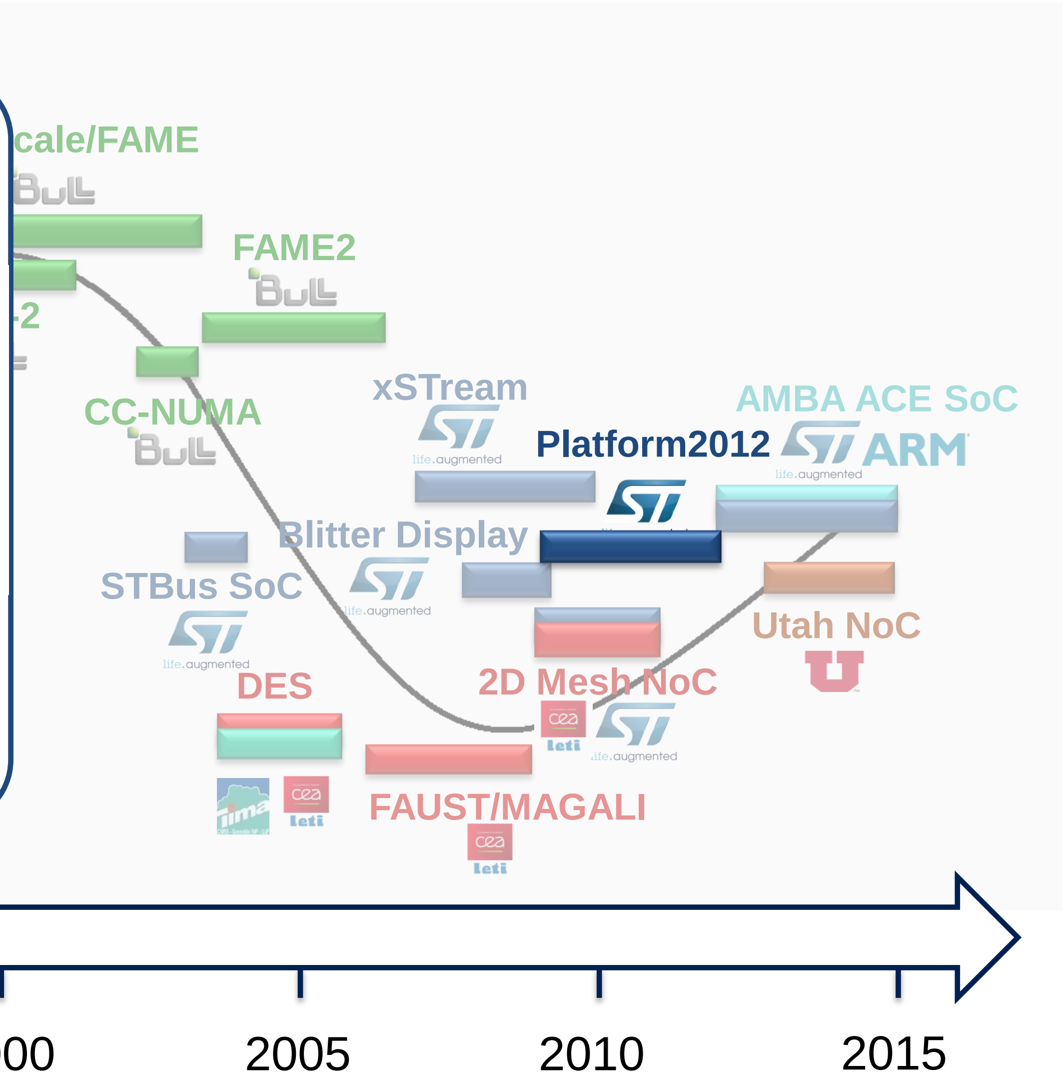


20 Years of Hardware Validation with CADP

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

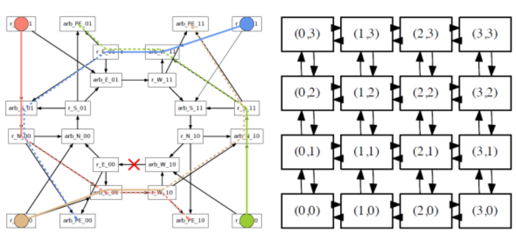


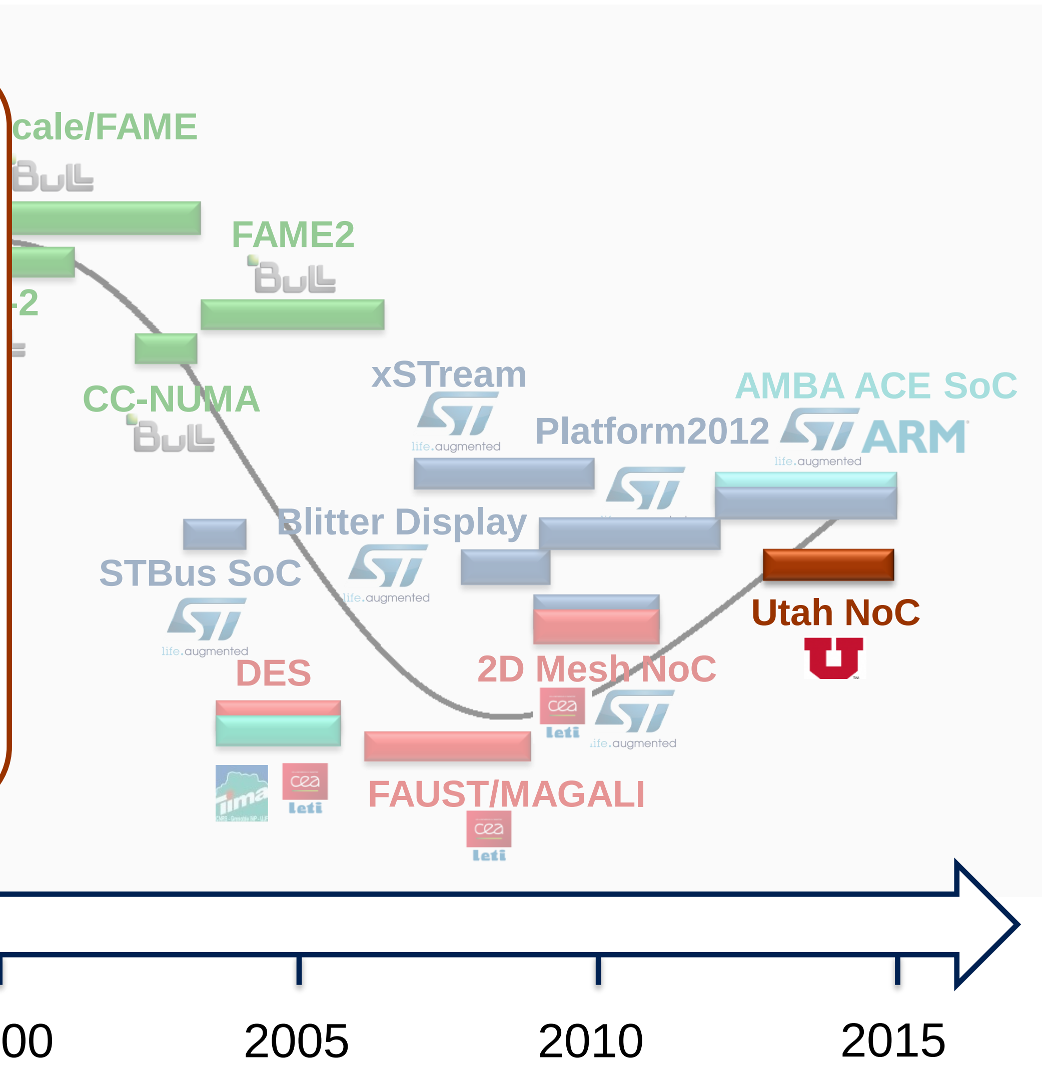


20 Years of Hardware Validation with CADP

Utah NoC

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

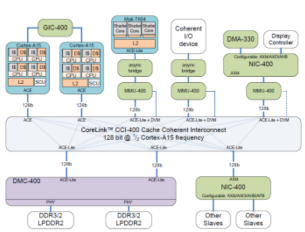


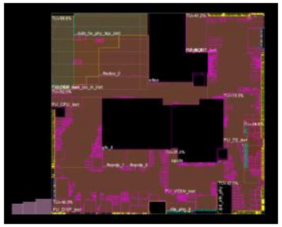


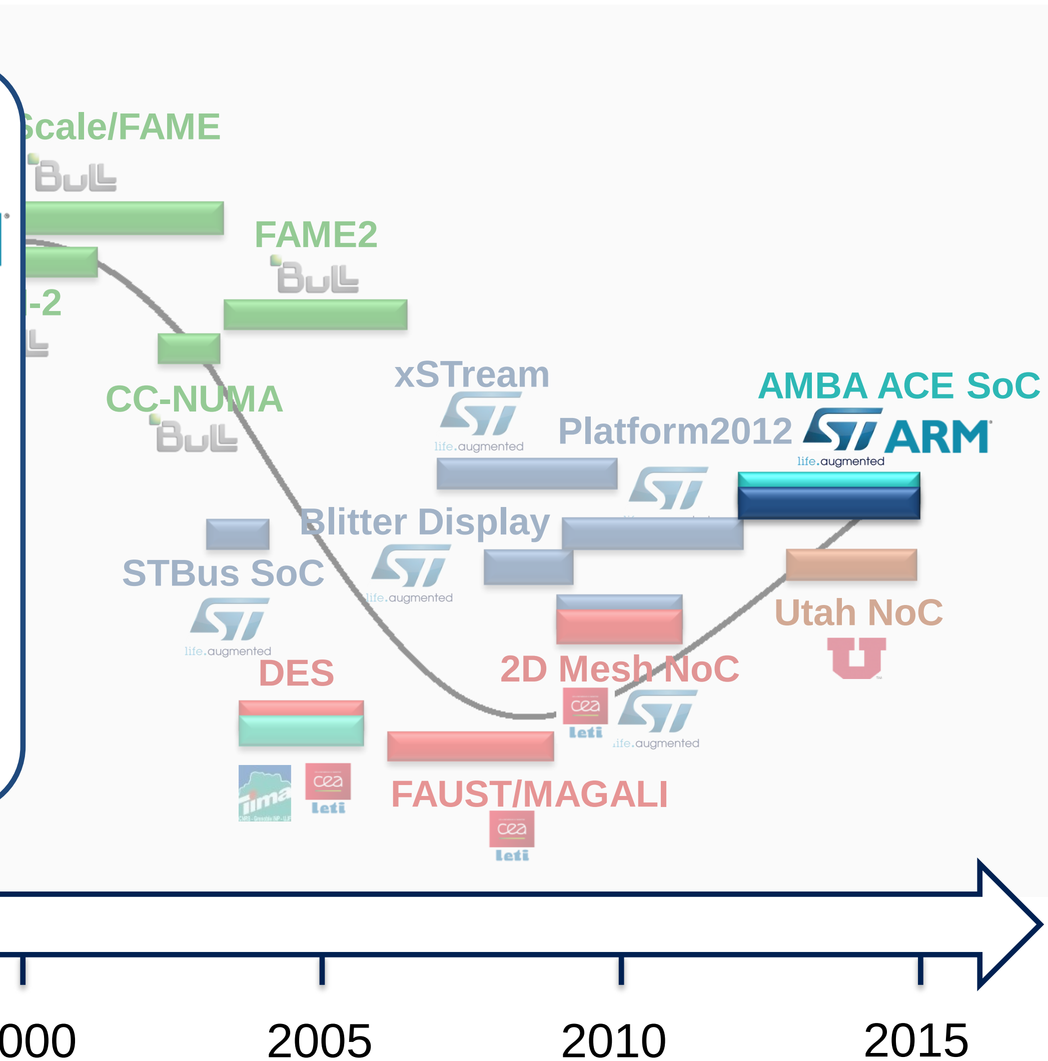
20 Years of Hardware Validation with CADP

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







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 programing 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

Formal Modeling

- Some figures about modeling effort in past projects

Case study	Company	Level	Modeling size
Powerscale	Bull	system	720 lines of LOTOS
Polykid	Bull	system	4000 lines of LOTOS (model) 2000 lines of LOTOS (rules) 3,400 lines of LOTOS and 7,000 lines of C (emulation)
SCSI-2	Bull	system	220 lines of LOTOS
FAME1/CCS	Bull	system	1200 lines of LOTOS
FAME1/NCS	Bull	system	1200 lines of LOTOS
FAME1/B-SPS/FSS	Bull	system	5000 lines of LOTOS 4500 lines of LOTOS

life.augmented

Formal Modeling

Case study	Company	Level	Modeling size
FAME1/ILU	Bull	unit	8900 lines of LOTOS 3400 lines of C
FAME1/PRR	Bull	block	7500 lines of LOTOS 200 lines of C
CC-NUMA	Bull	system	1800 lines of LOTOS 1000 lines of Murphi
DES	CEA-Leti/TIMA	unit	1700 lines of CHP 3800 lines of LOTOS
FAME2/PAB	Bull	block	3977 lines of LNT
FAUST/MAGALI	CEA-Leti	system	1200 lines of CHP
xStream	ST	unit	6800 lines of LOTOS

Formal Modeling

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

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

Case study	Functional Verification Results
Powerscale	Hidden bug found in a few minutes FORTE'96 [Chehaibar-Garavel-Mounier-Tawbi-Zulian-96]
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

Case study	Functional Verification Results
FAUST/MAGALI	Routing problem detected in the CHP description ASYNC’07 [Salaum-Serwe-Thonnart-Vivet-07]
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]

life.augmented

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

Case study	Functional Verification Results
Polykid/Test generation	5 new bugs discovered in VHDL design IWTCS'98 [Kahlouche-Viho-Zendri-98]
Polykid/Emulation	Replacement of a missing ASIC by a software emulation running on a PowerPC microprocessor STTT'01 [Garavel-Viho-Zendri-01]
FAME/CCS	Directed test generation using TGV 21 base tests (1 mn per test) 50 collision tests (15 mn per test) 1 generalized test (1 day)
FAME/NCS	Directed test generation using TGV 50 base tests (30 sec per test)
FAME/PRR	Random test generation using Executor Detection of a non-conformity between LOTOS and Verilog codes for PRR v1 (not detected using commercial tools)

Model-based Testing Results

Case study	Functional Verification Results
FAME/ILU	Co-simulation using Exec/Caesar
FAME/B-SPS/FSS	Trace validation with coverage Major bug found (ambiguity in informal specification) Insufficient coverage found (3 missing tests added) SPIN'04 [Garavel-Mateescu-04]
FAUST/MAGALI	Co-simulation: LOTOS-SystemC / VHDL netlist Detection of spurious inputs generated by LOTOS model: Constraints added to generate only valid inputs
Plateform2012/DTD	Co-simulation: C++ / LNT Found C++ incorrect for some particular scenarios Science of Computer Prog. [Lantreibecq-Serwe-14]
AMBA ACE SoC	Model-based test generation using counterexamples targeted at corner cases Early detection of 10 errors in commercial verification IPs

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

Case study	Formalism	Functional Verification Results
SCSI-2	IMCs	Steady-state analysis suggested strategies to avoid starvation and increase throughput FME'02 [Garavel-Hermanns-02]
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]

life.augmented

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