System Level Modeling and Simulation

Designing Future Computer Systems

Emad Arasteh

arasteh@chapman.edu

Fowler School of Engineering
Chapman University

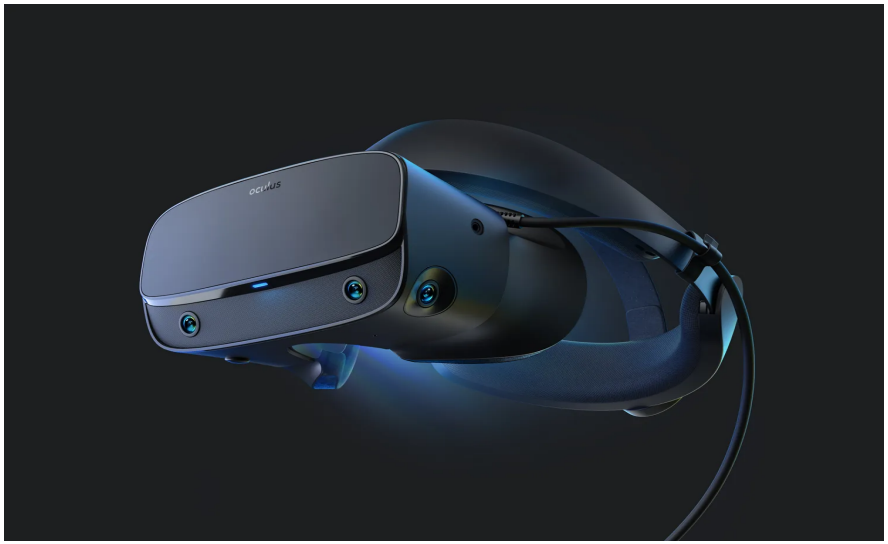
October 4, 2022

Outline

Computers come in various forms

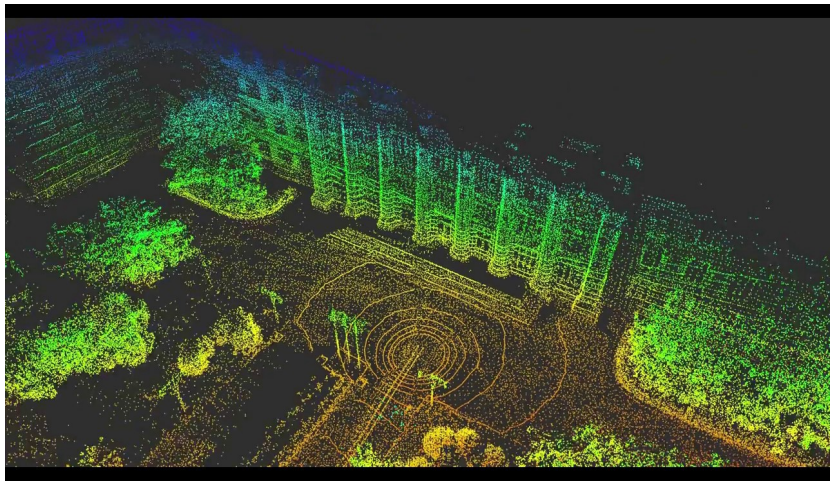
Computers come in various forms

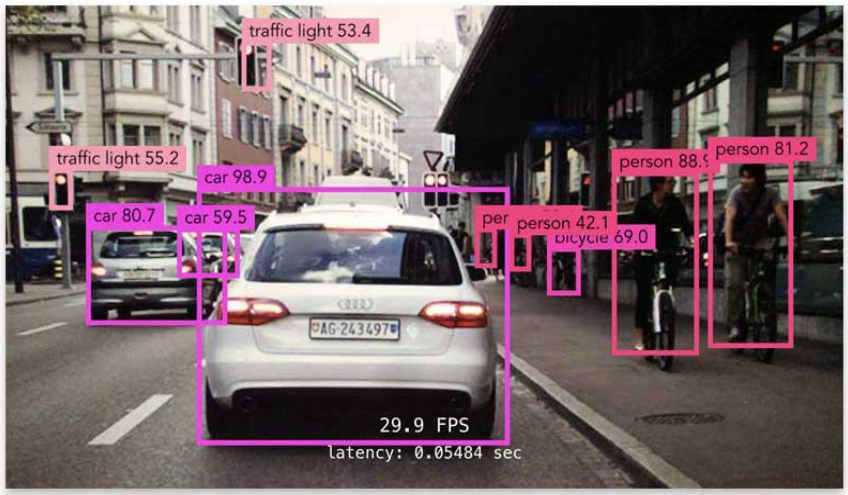
Computers come in various forms



Sources: Infinix, CMTC, Oculus, Volvo

Applications come in many flavors

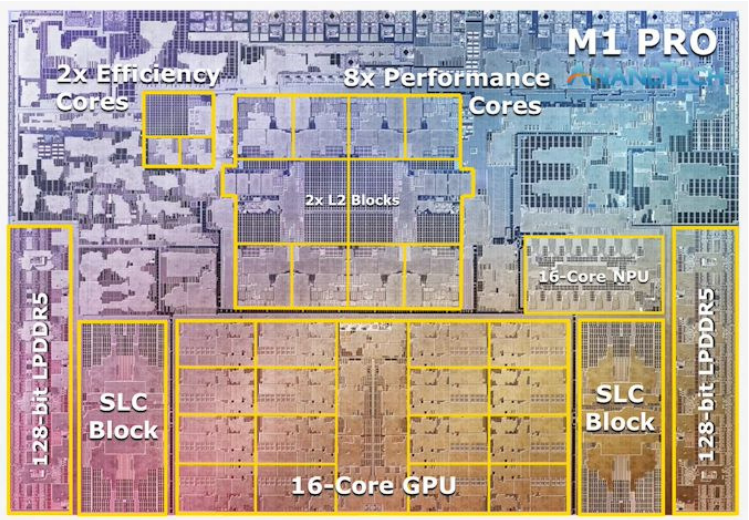




Sources: MyTechDecisions, GP-SLAM+, Second Life

Embedded Computer System Design

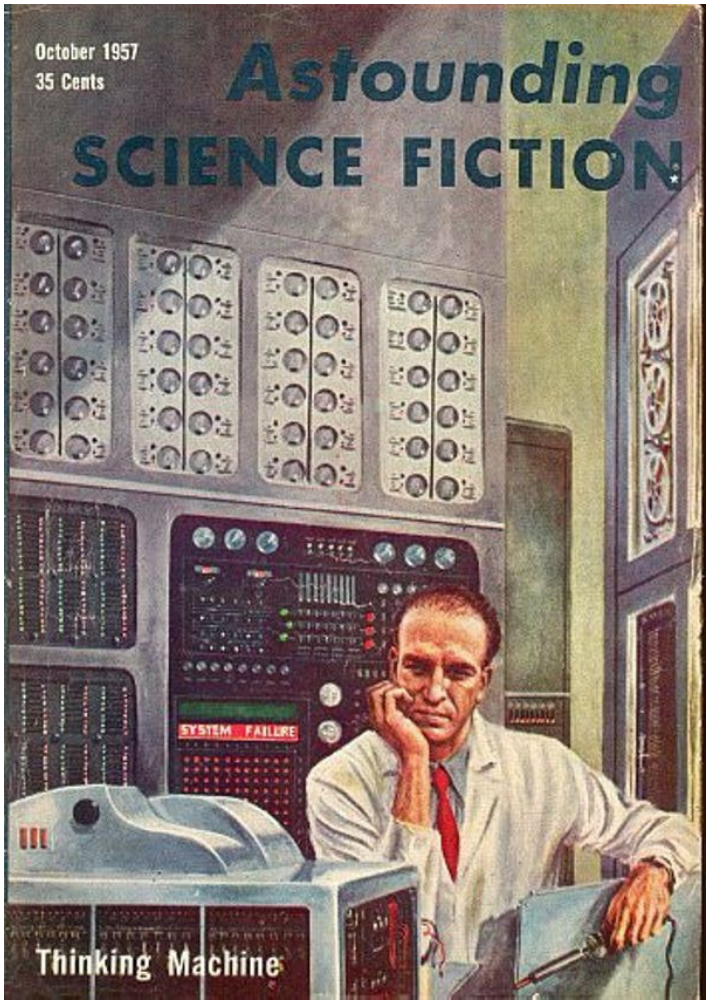
- Design challenges
  - Often mobile
    - Battery powered (low power)
  - Often highly reliable
    - Extreme environment (e.g. temperature)
  - High performance requirements
    - Often real-time requirements
  - High complexity
  - Tightly coupled
    - Software
    - Hardware
  - Rapid development for low price

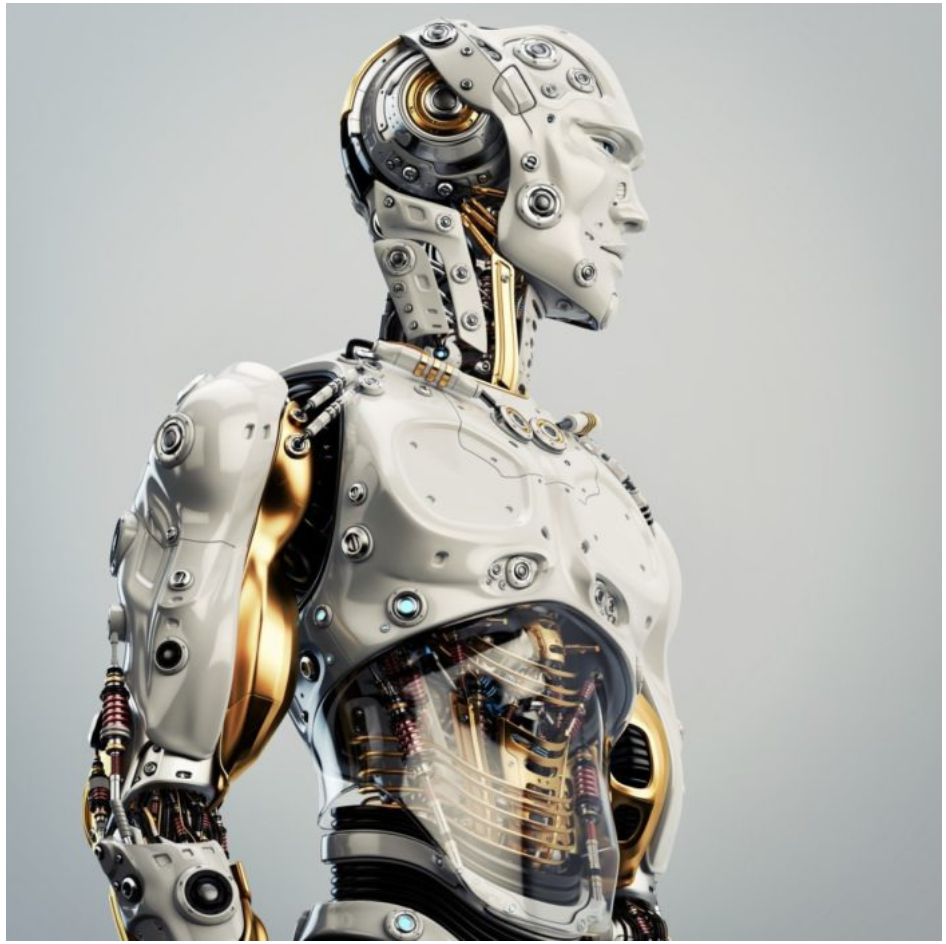


Source: Apple Silicon M1 (2020)

What if we want to build future computer systems?

- Run **efficient**, **reliable** and **safe** with the ever increasing emergence of new applications



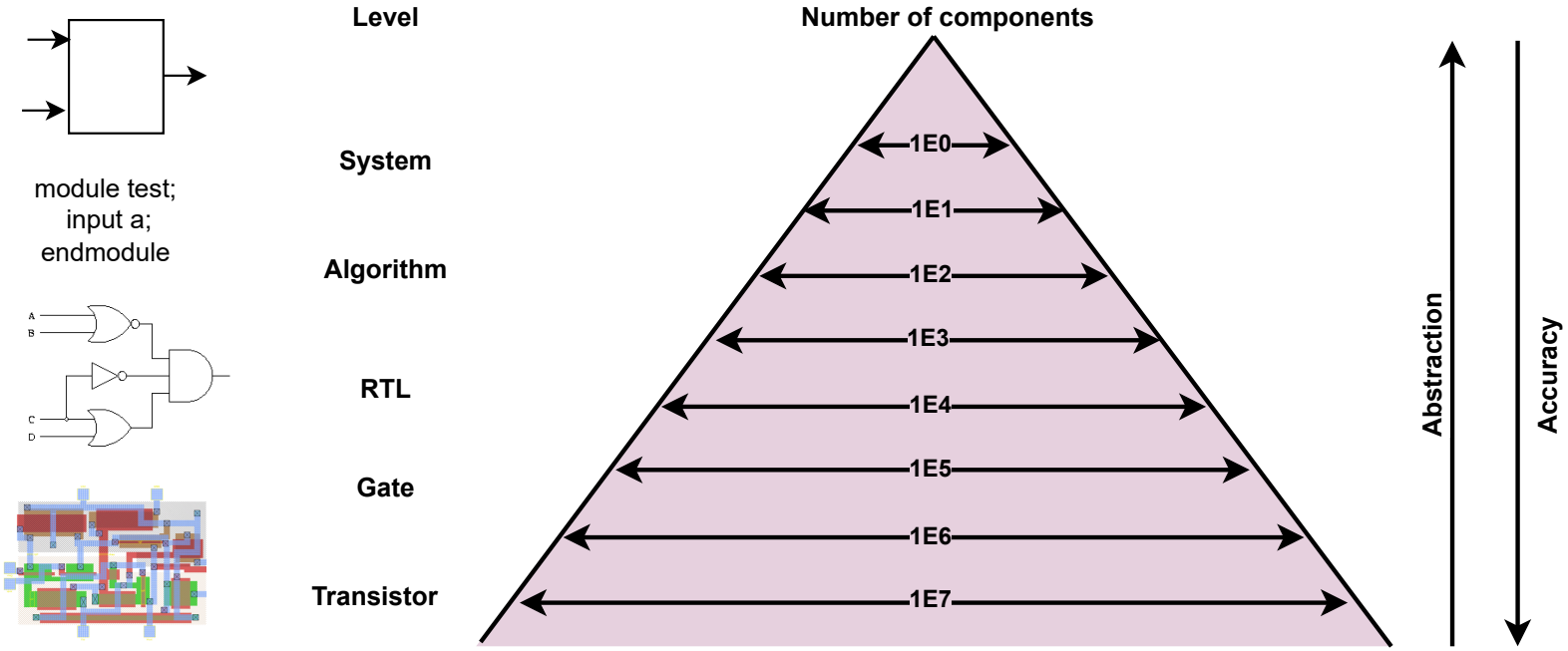


Research Approach

- Application trends
  - Data-intensive: machine learning, genome sequencing, etc
  - Require **high-capacity** and **high-bandwidth** memory
- Technological trends
  - Availability of massively **parallel** processors
  - Advances in integration of **memory-logic** fabrics
- Bridge the **software** and **hardware** design
- **Co-design** of hardware-software
  - System-Level Design Languages (SLDL)

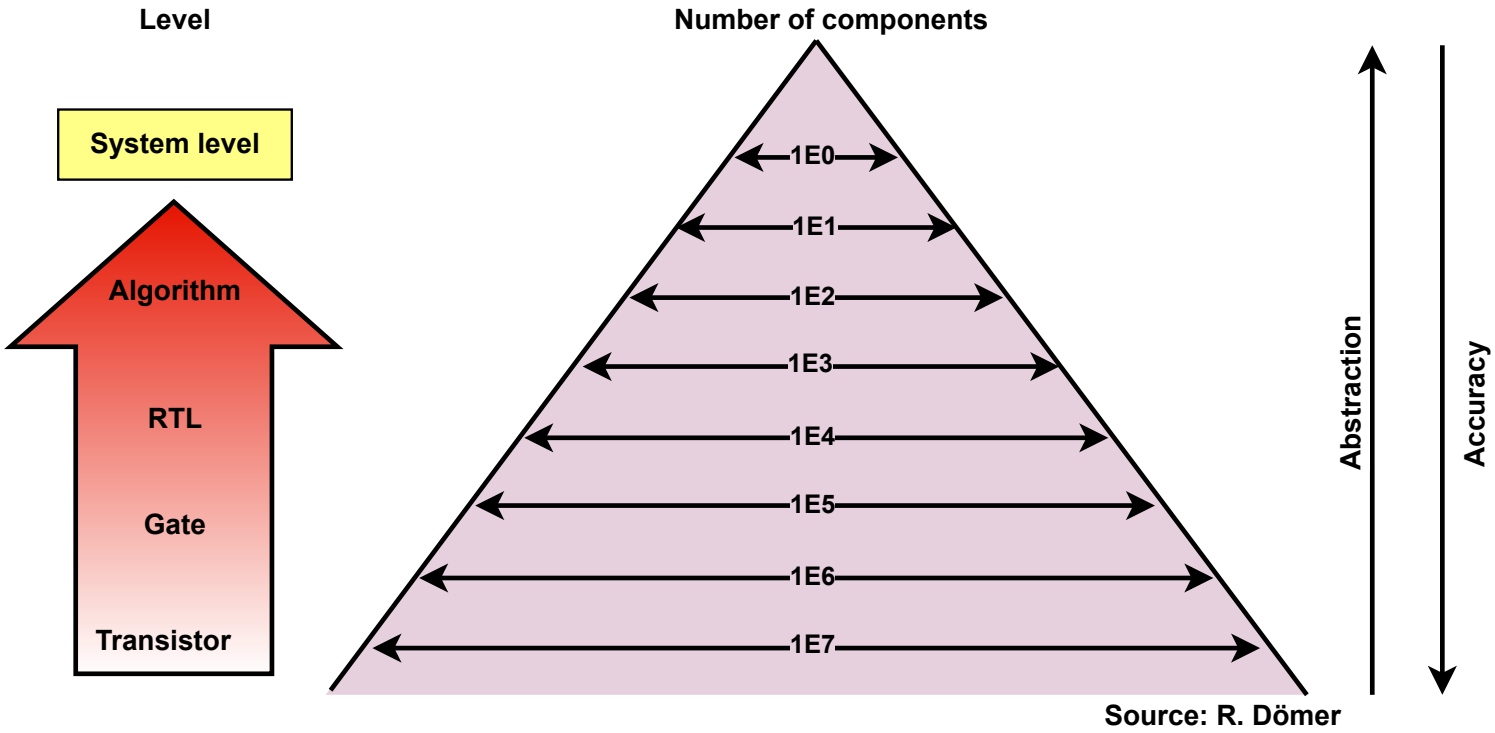
Abstraction Level

- Embedded system design faces tremendous increase in design complexity

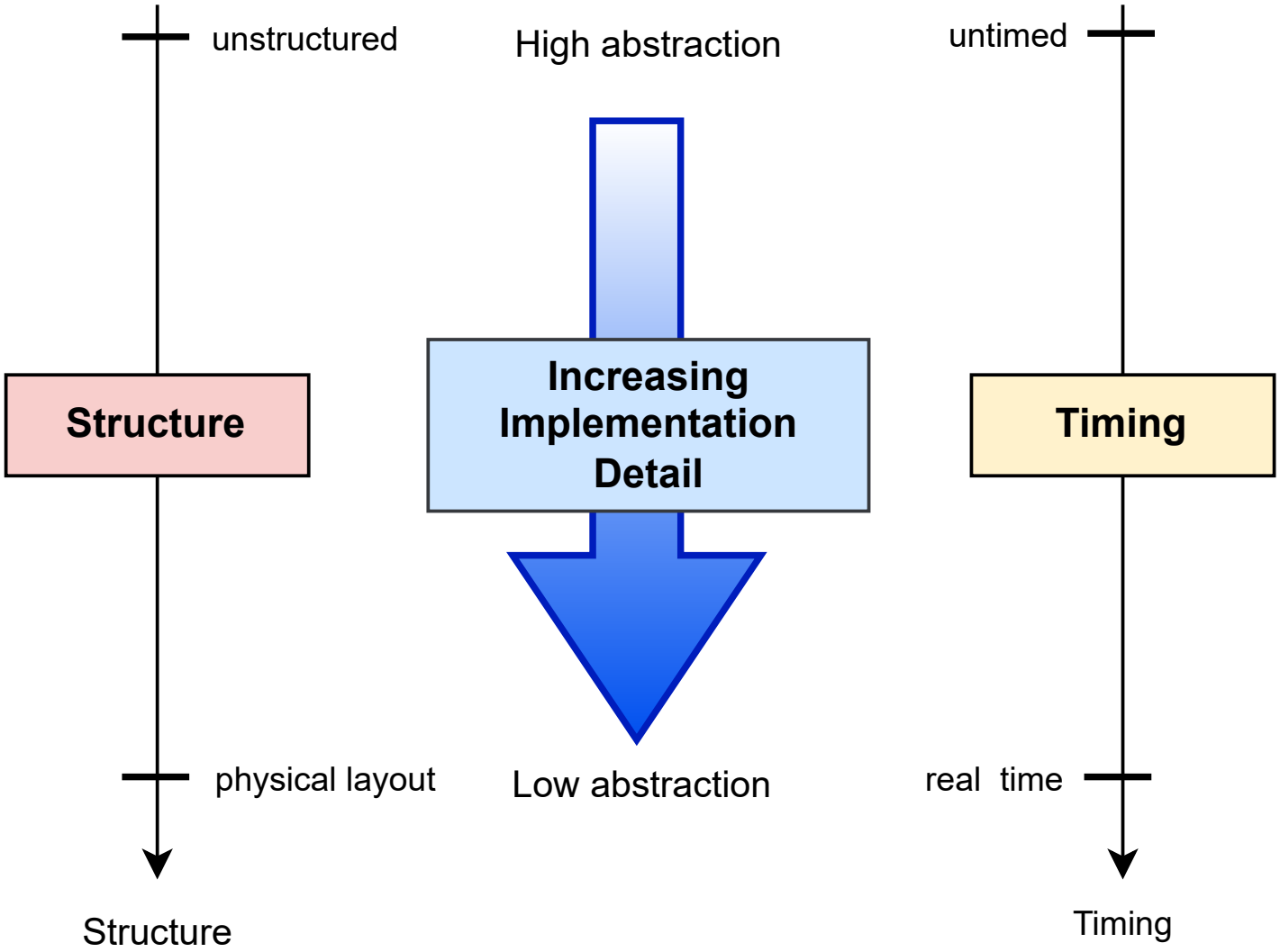


Abstraction Level

- Embedded system design faces tremendous increase in design complexity
  - Move to higher levels of abstraction!

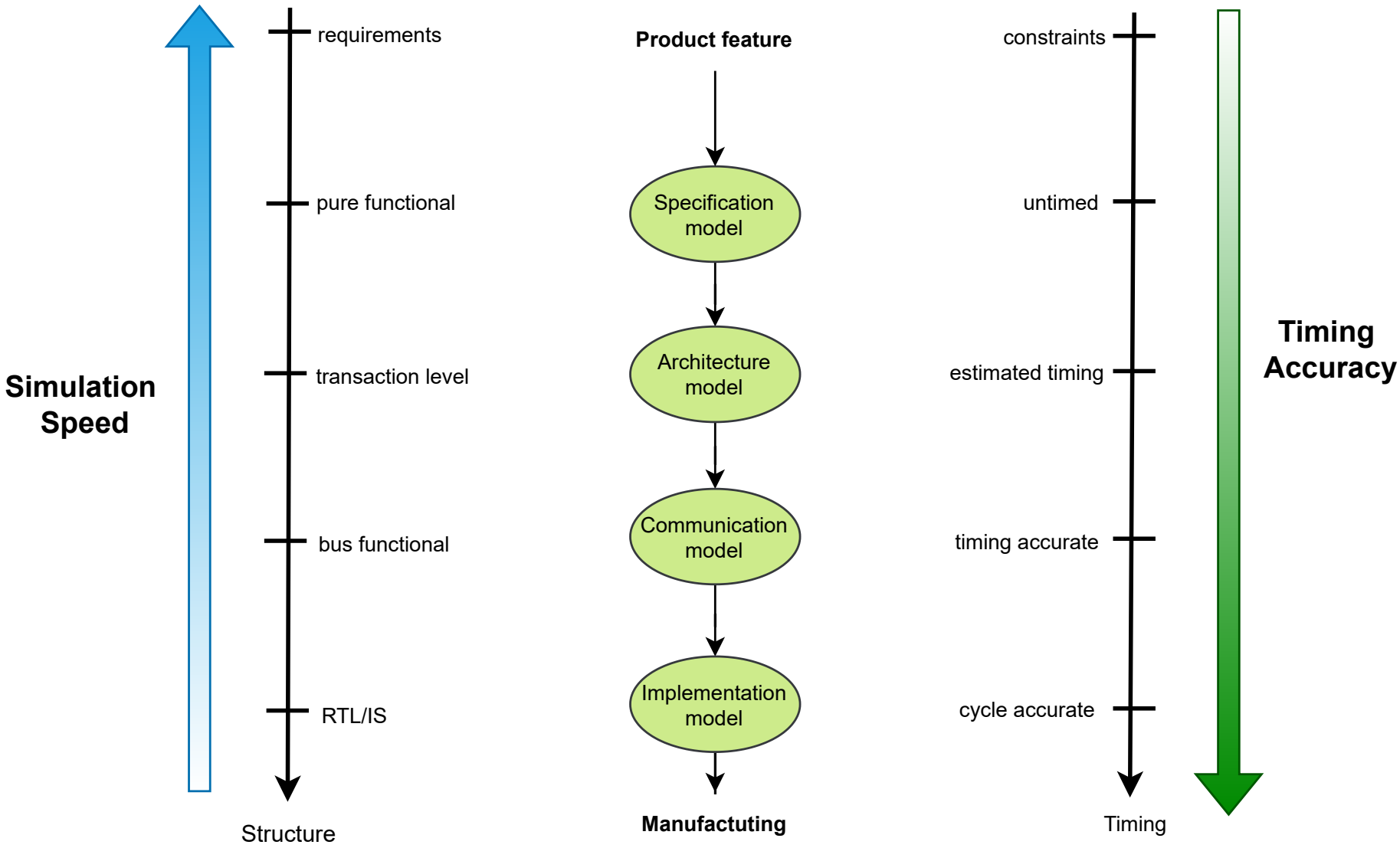


Top-Down Design Flow



Source: R. Dömer

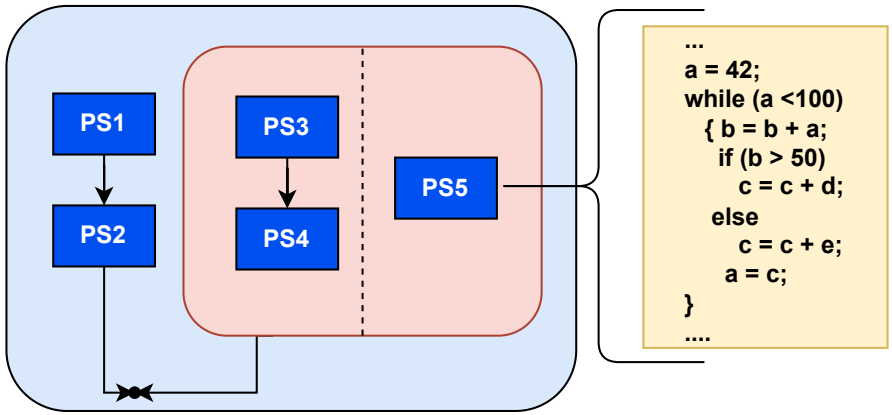
Top-Down Design Flow



Source: R. Dömer

Abstract Modeling

- Model of Computation
  - Formal description of a system model at high abstraction level
    - Specification, documentation, reasoning, validation, synthesis
- Model for Hardware and Software design
  - State-based models of computation
  - from Finite State Machine (FSM)
  - to Program State Machine (PSM)
    - States described by procedures in a programming language



Source: R. Dömer

System-Level Description Languages

- Goals and requirements
  - Behavioral hierarchy, structural hierarchy, concurrency, synchronization, exception handling, timing, state transitions
- Requirements were not fully supported by existing languages
  - C, C++, Java, VHDL, Verilog, StateCharts, etc.
- SpecC and SystemC were born in UC Irvine in early 2000s
- Today SystemC is the IEEE standard for system-level modeling and simulation (HW/SW co-design)

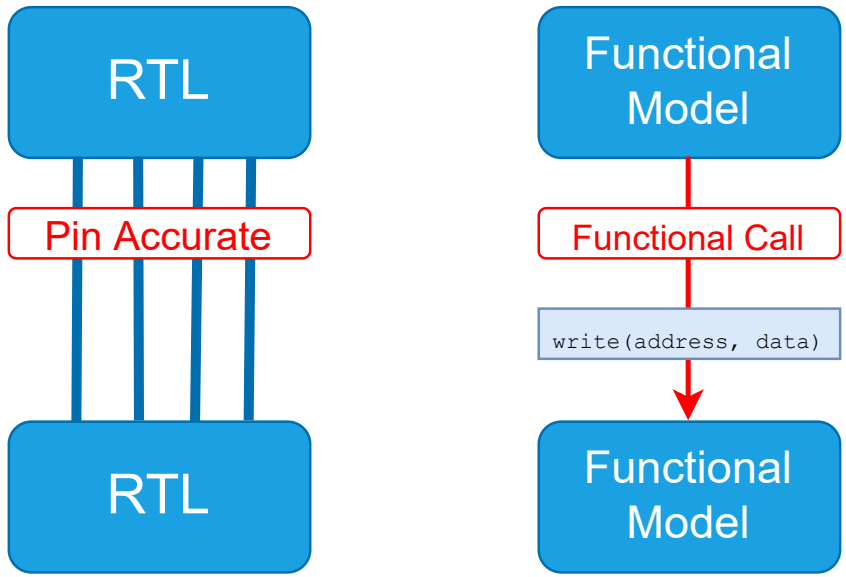
accellera SYSTEMS INITIATIVE

SYSTEMC™

Source: Qualcomm

Transaction Level Modeling

- Separation of Concerns
  - System design principle to address each *concern* separately
  - TLM focuses on separating computation from communication in the model
- TLM can speed up simulation significantly by replacing many pin-level events in detailed RTL model

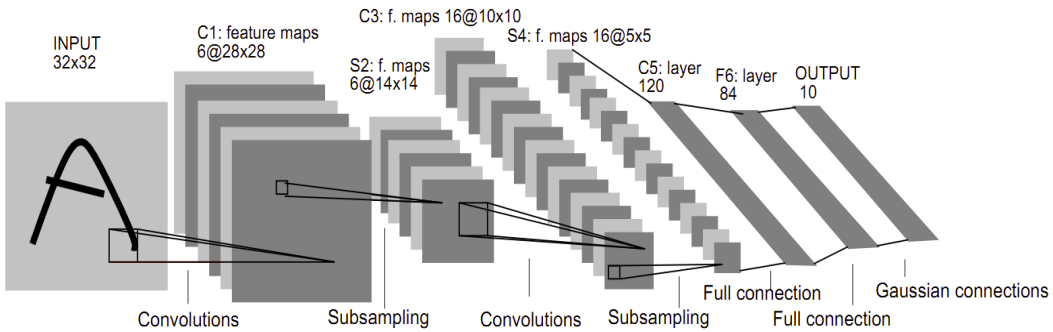


Simulate every event!

100-1000x faster simulation!

Deep Learning and CNN

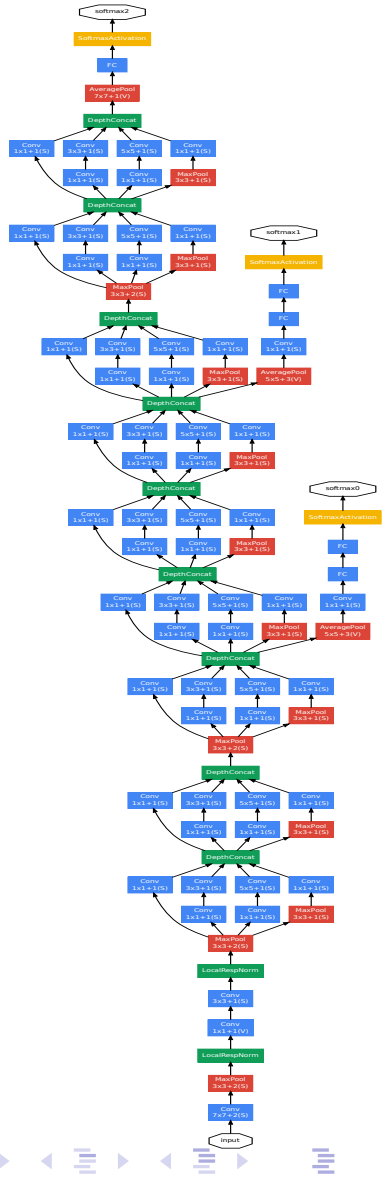
- DL, class of machine learning methods
  - Extract useful features from input data, perform data transformations, and arrive at a final meaningful representation
- DL applications
  - Visual recognition, in particular image classification
- DL success
  - Growth of computing power, availability of huge datasets for training and rapid innovation in DL architectures
- Convolutional neural network (CNN)
  - Alternating convolution and pooling (sub-sampling) layers



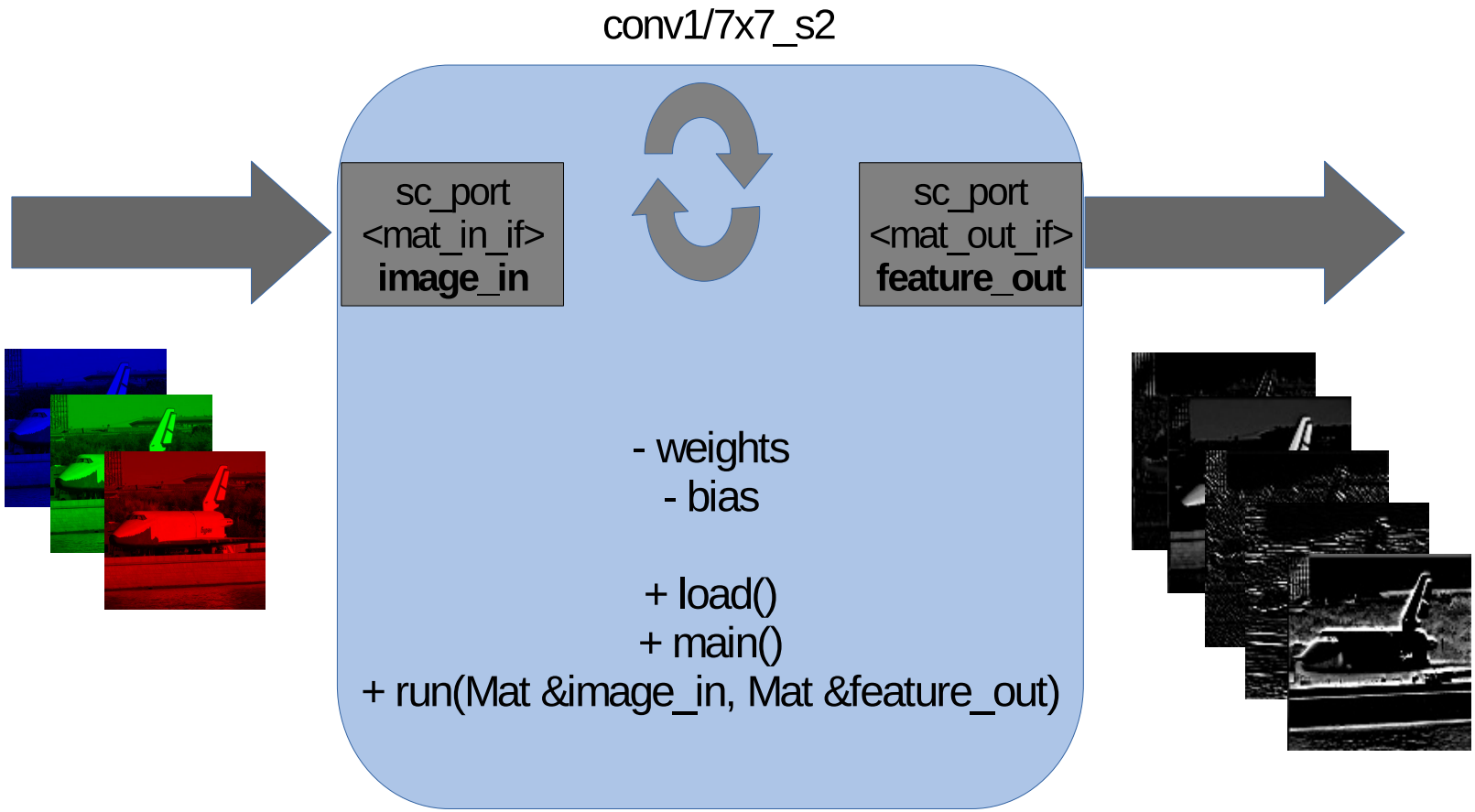
GoogLeNet

- Deep CNN for image classification
- 142 independent distinct layers
- Convolution Pooling Concat Classifier
- TLM modeling enables investigation of the parallelism and memory bottleneck

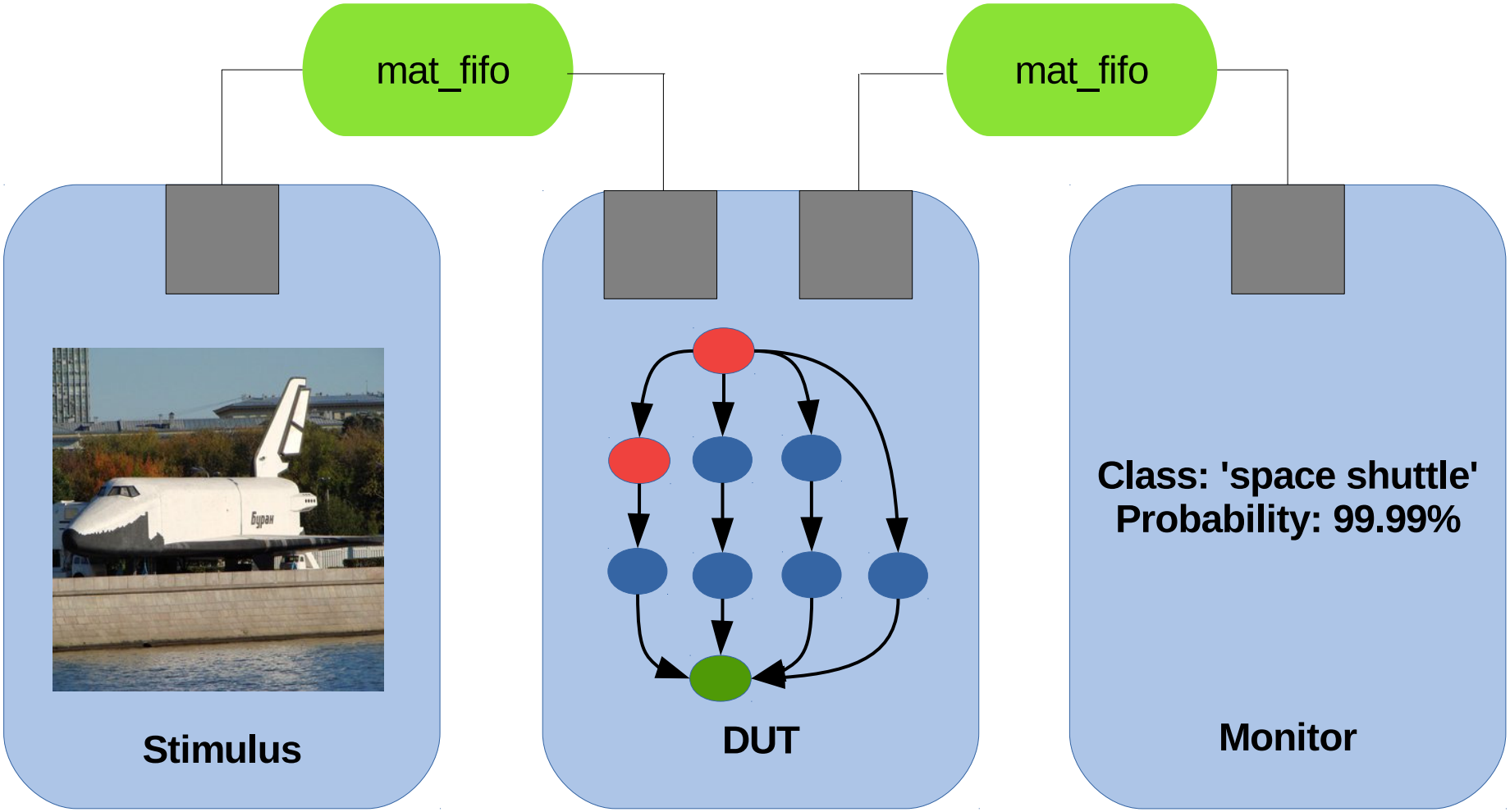
Layer type	Count
Convolution	57
ReLU	57
Pooling	14
LRN	2
Concat	9
Dropout	1
InnerProduct	1
Softmax	1
Total	142



TLM of Convolution

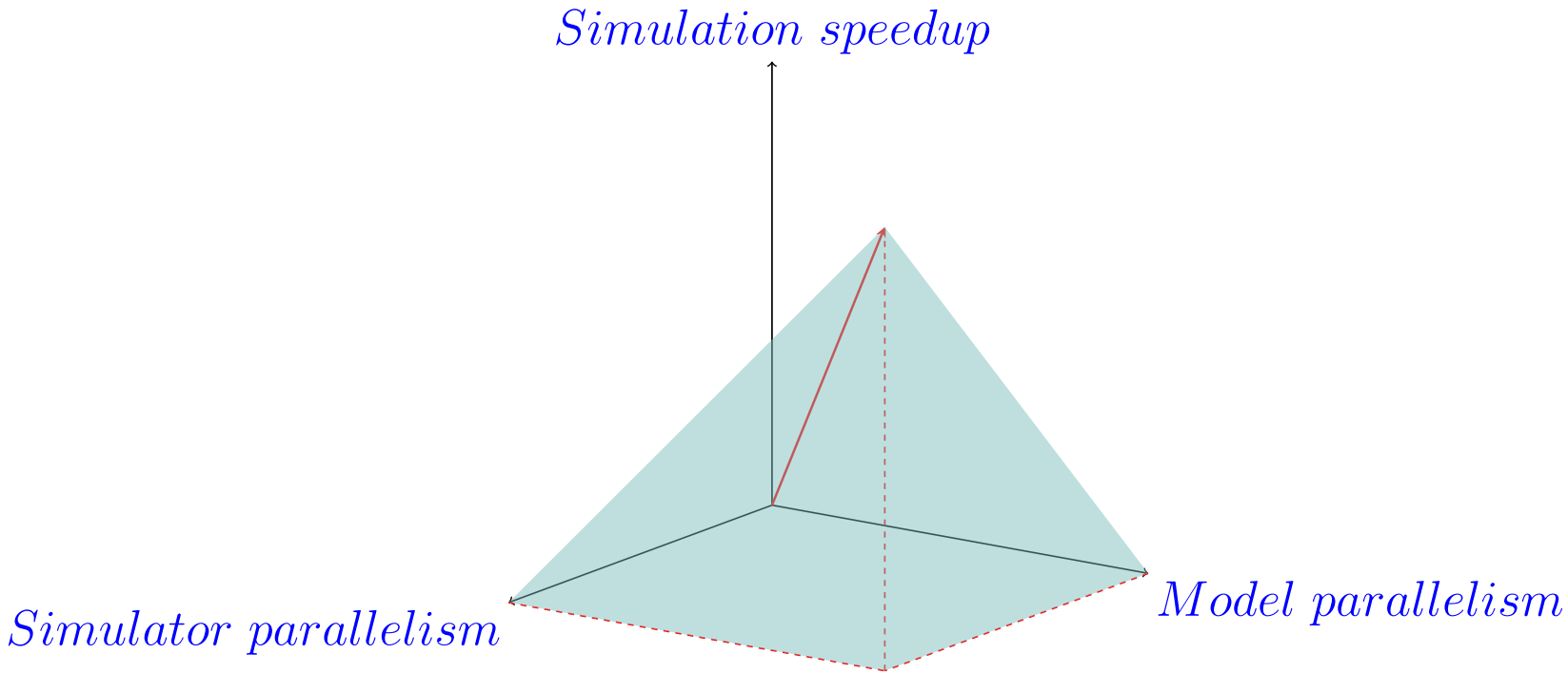


SystemC Test Bench



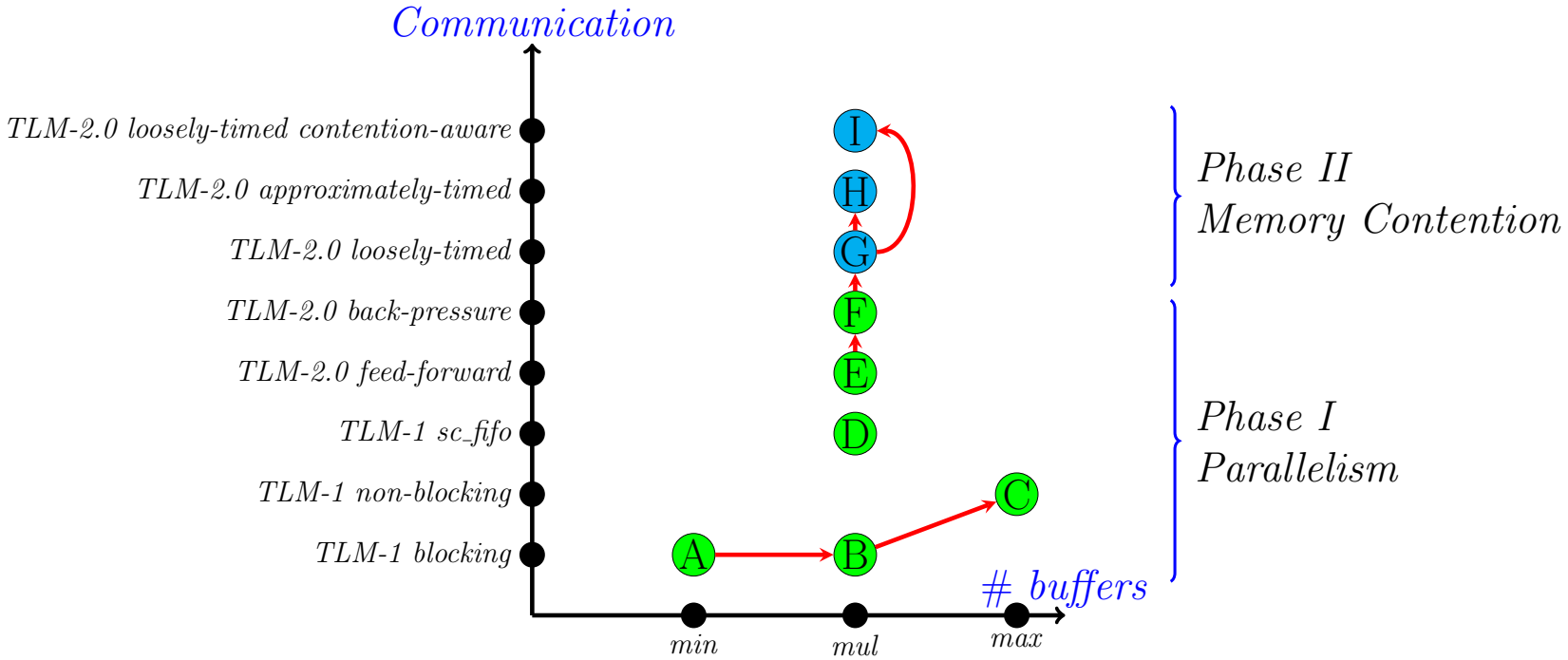
Model Parallelism

- TLM must expose any available **parallelism** in the application
- The explicit parallelism in the model can be exploited by **parallel discrete event simulation** (PDES)
- *Can we improve level of parallelism in TLM models by assessing the PDES performance?*



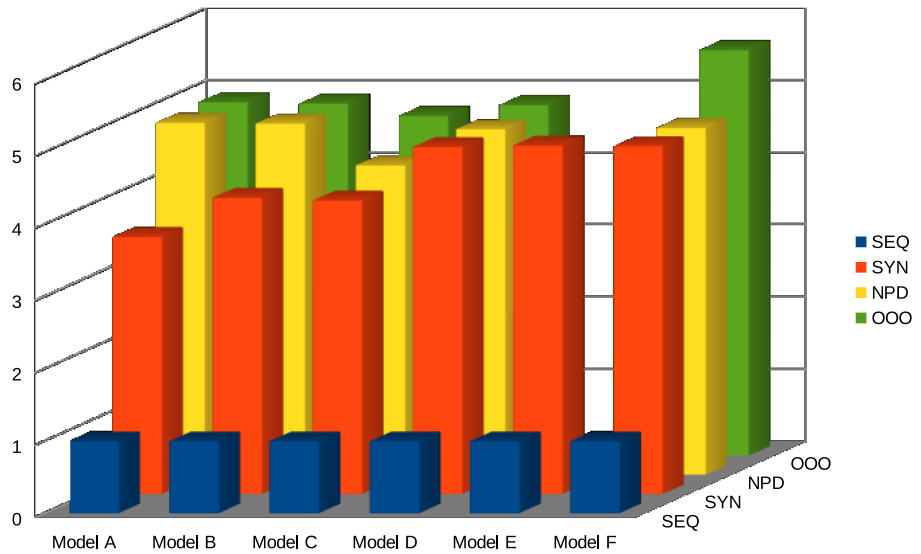
Parallelism and Communication

- Choice of **communication** and **synchronization** between concurrent modules has a significant impact on the available **parallelism**



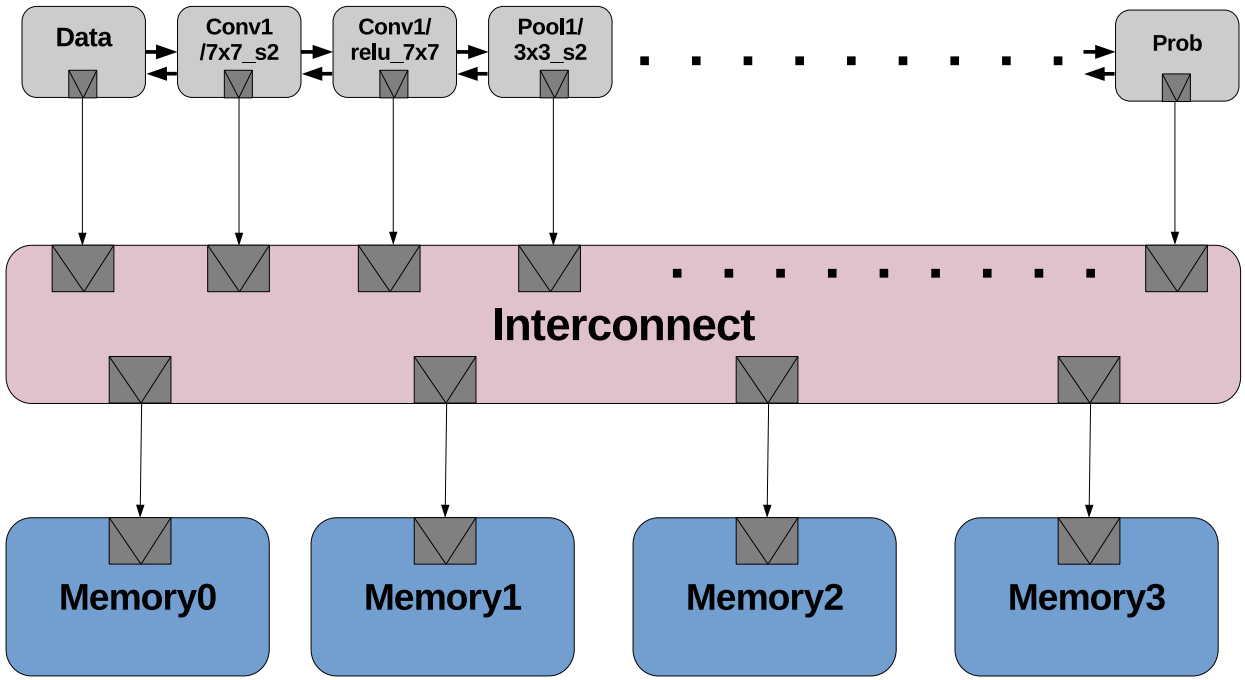
Results: Parallel Simulation

- Increase simulation speed up to 5.6x on a 16-core host machine using aggressive out-of-order parallel simulation in RISC
- Model F has shown the highest level of parallelism available
- Increased parallel simulation performance indicates better models with higher amount of parallelism exposed

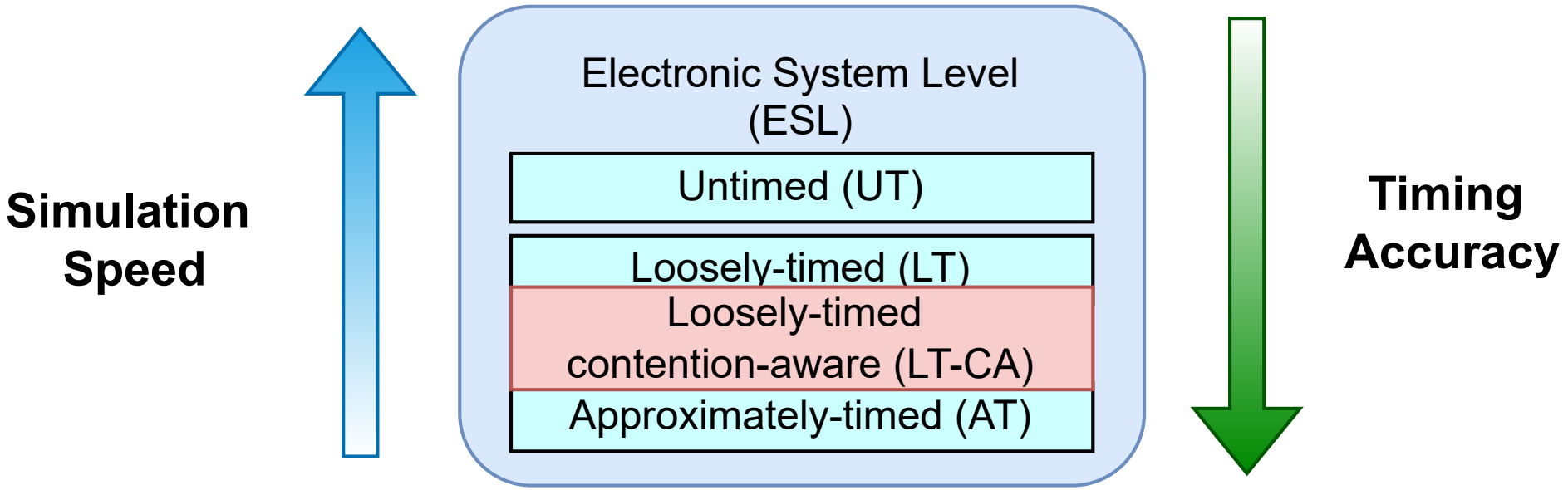


Memory Bottleneck

- Data-intensive application exacerbates the well-known memory bottleneck in computer systems
- Memory bottleneck is often detected late in the design cycle
- A memory bottleneck detected at such late stage can severely limit the available design choices or even require costly redesign



Memory Contention in TLM (1)

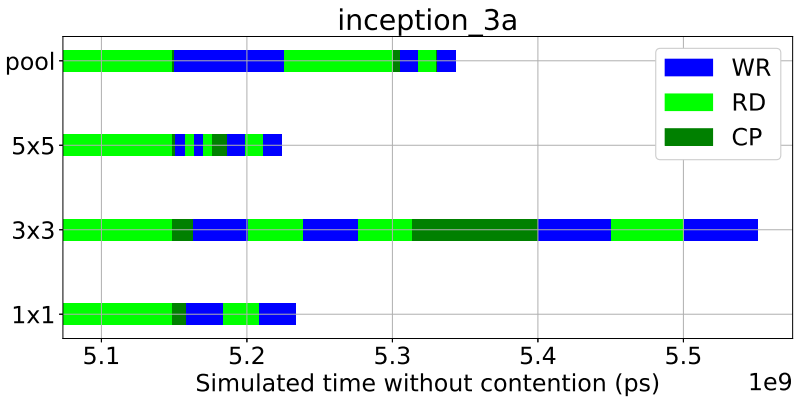


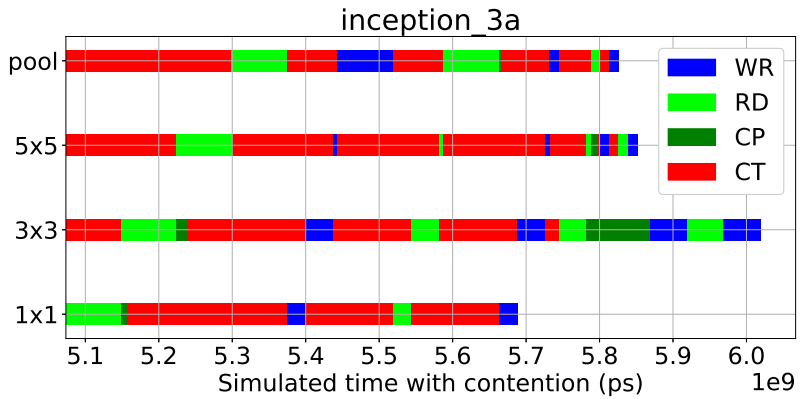
Memory Contention in TLM (2)

- Untimed
  - Early feedback on parallelism and local communications
  - No notion of memory contention ☹
- Loosely-timed
  - Programmer's view for early software development
  - Simulates fast but no notion of memory contention ☹
- Approximately-timed
  - Architecture exploration and detailed performance analysis
  - Model memory contention but simulates significantly slower than LT ☹
- Loosely-timed contention-aware
  - Models memory contention with fast LT simulation speed and same AT accuracy ☺

Results: Contention Visualization

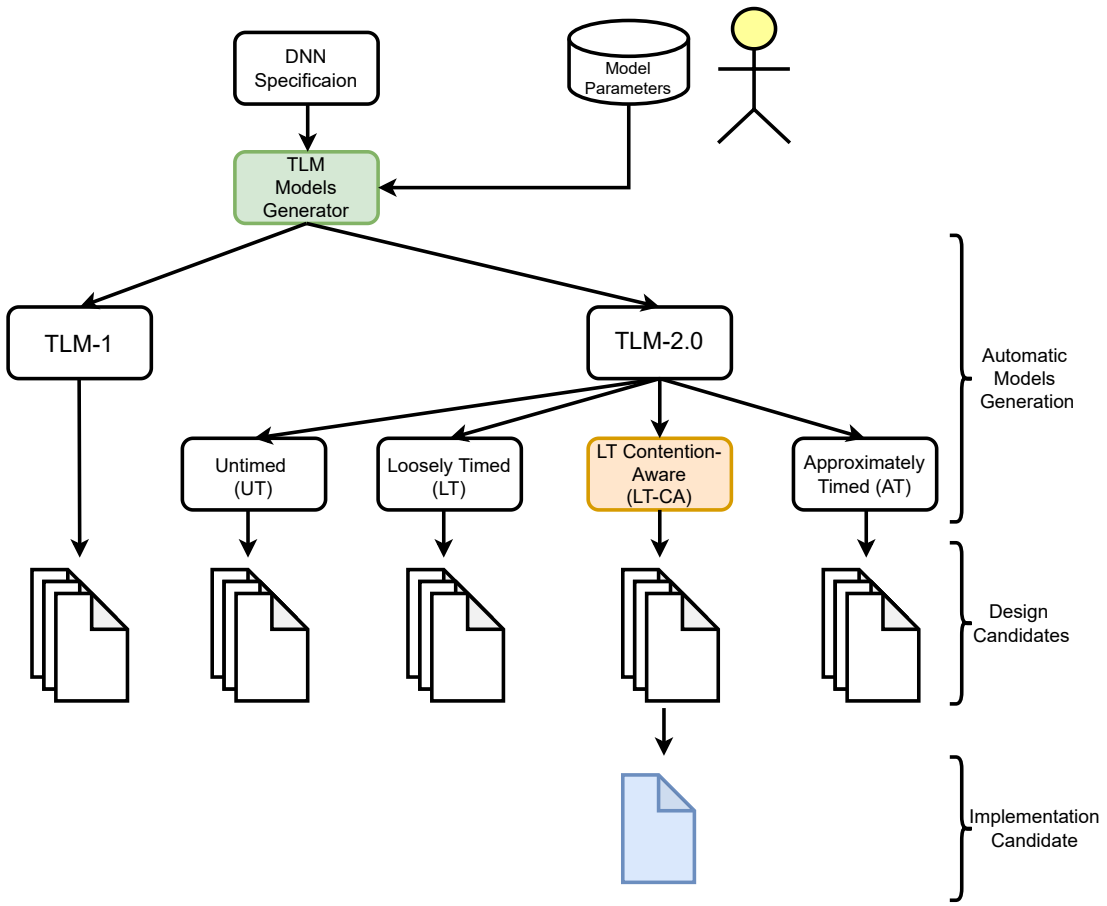
- Analyze memory access patterns and the effect of memory contention
- Transaction-level timing diagrams for the first inception module with pass of 1 image
- LT does not model contention so layers in parallel tracks access memory without blocking each other
- In LT-CA model, layers are blocked and wait until access becomes available red areas





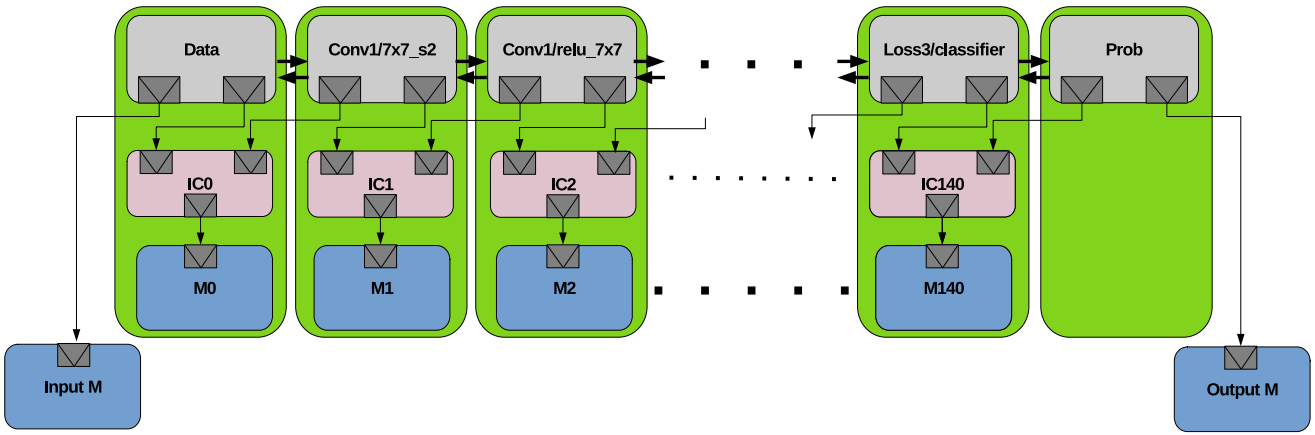
DNN TLM Modeling Framework

- Automatic TLM model generation to explore parallelism and memory contention
- Improve model parallelism (5.6x on a 16-core machine)
- Early contention modeling with high accuracy and fast simulation speed (46x with 1% error)



Processing-in-Memory

- New processor architectures that place private local memories close to computing units
- Reduce average bandwidth usage on global shared memory
- However, contention will emerge between local memories
- An unexplored simulation framework in this context is TLM
- TLM is an attractive approach that enables efficient **codesign** of both **hardware** and **software** solutions with fast and accurate memory contention modeling approach



Programming Models for Processing-In-Memory

- Define new programming models based on TLM for processing-in-memory computer systems
- A number of emerging areas will need to be explored:
  - Hardware simulation models
    - Grid of Processing Cell (GPC)
  - Software stack
  - Compilation flows
  - OS and run-time system support

Acknowledgment

- Prof. Rainer Dömer, UC Irvine
- Dr. Zhongqi Cheng, Apple
- Daniel Mendoza, Stanford
- Vivek Govindasamy, UC Irvine
- Yutong Wang, UC Irvine
- This research was made possible by the support from Intel Corporation

References

- [TECS'22] E. Arasteh, R. Dömer: "Fast Loosely-Timed System Models with Accurate Memory Contention", Journal of ACM Transactions on Embedded Computing Systems (under review), 2022.
- [FDL'21] E. Arasteh, R. Dömer: "Improving Parallelism in System Level Models by Assessing PDES Performance", Proceedings of Forum on Specification and Design Languages, Antibes, France, Sep. 2021.
- [CECS'21] E. Arasteh, R. Dömer: "Systematic Evaluation of Six Models of GoogLeNet using PDES", Center for Embedded and Cyber-Physical Systems, Technical Report 21-03, 2021, Sep. 2021.
- [Springer'20] R. Dömer, Z. Cheng, D. Mendoza, E. Arasteh: "Pushing the Limits of Parallel Discrete Event Simulation for SystemC", in "A Journey of Embedded and Cyber-Physical Systems" by Jian-Jia Chen, Springer Nature, Switzerland, August 2020.
- [DATE'20] D. Mendoza, Z. Cheng, E. Arasteh, R. Dömer: "Lazy Event Prediction using Defining Trees and Schedule Bypass for Out-of-Order PDES", Design, Automation and Test in Europe Conference, Grenoble, France, March 2020.
- [ASPDAC'20] Z. Cheng, E. Arasteh, R. Dömer: "Event Delivery using Prediction for Faster Parallel SystemC Simulation", Asia and South Pacific Design Automation Conference, Beijing, China, Jan. 2020.
- [IESS'19] E. Arasteh, R. Dömer: "An Untimed SystemC Model of GoogLeNet", Proceedings of the International Embedded Systems Symposium, Friedrichshafen, Germany, Sep. 2019.