Architectural/Micro-architectural Exploration on Virtual Platforms

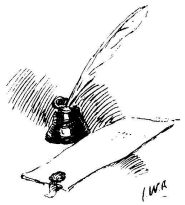
Qi Zhu, Michael Kishinevsky,
Zhu Zhou, Atul Kwatra
Intel Corporation

Outline

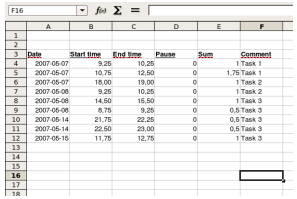
- Motivations
- Transaction-Level Modeling Methodology
- SOC Platform Exploration
- Summary and Future Directions

Motivations

pen&paper

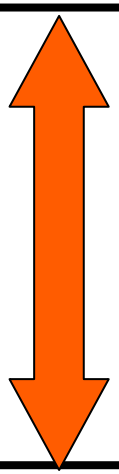


spreadsheets



	A	B	C	D	E	F
1						
2						
3	Date	Start time	End time	Pause	Sum	Comment
4	2007-05-07	9,25	10,25	0	1	Task 1
5	2007-05-07	10,75	12,50	0	1,75	Task 1
6	2007-05-07	18,00	19,00	0	1	Task 2
7	2007-05-08	9,25	10,25	0	1	Task 2
8	2007-05-08	14,50	15,50	0	1	Task 3
9	2007-05-08	8,75	9,25	0	0,5	Task 3
10	2007-05-14	21,75	22,25	0	0,5	Task 3
11	2007-05-14	22,50	23,00	0	0,5	Task 3
12	2007-05-15	11,75	12,75	0	1	Task 3
13						
14						
15						
16						
17						
18						

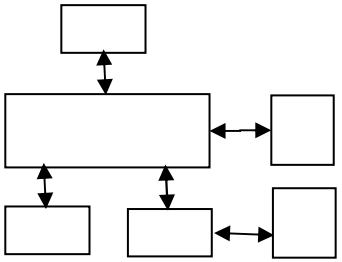
manual and abstract
(hard to explore complex designs)



Need a framework to

– explore multiple abstraction levels in between

– standard interfaces and modular designs for easier exploration and reusability



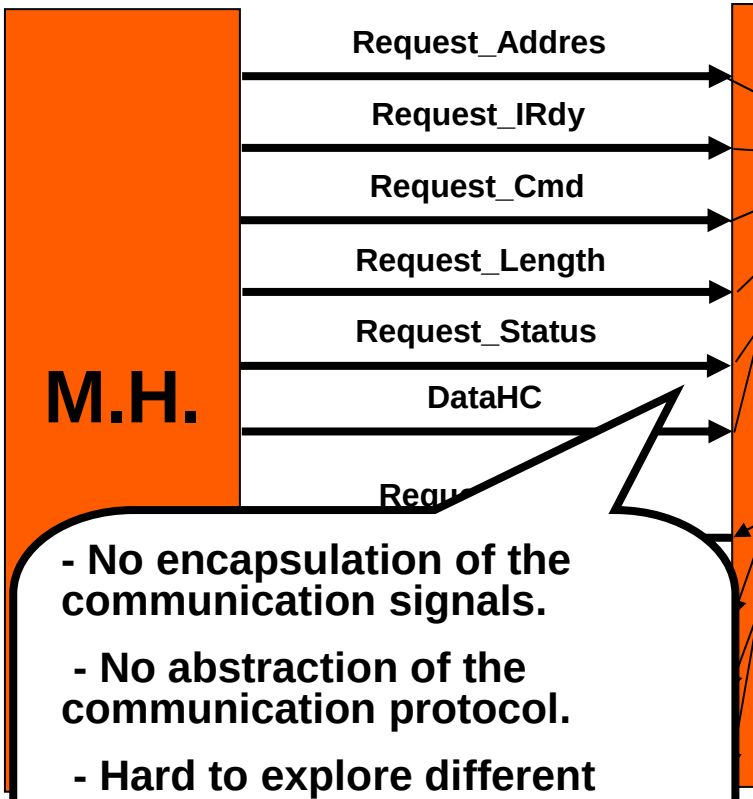
detailed perf. models
(take time to build, hard to extend)

Transaction-Level Modeling Methodology

- A system-level modeling framework (virtual platform) based on TLM
  - Separation of communication and computation for modular design and easier exploration.
  - Multiple abstraction levels of modeling for trade-off between complexity and accuracy.
  - Integration with external IPs.
  - Leverage of vendor tools for development, debugging and analysis capabilities.
  - Configurable domain-specific libraries with standard interfaces for design efficiency.
- Use SystemC 2.2 and TLM 2.0 library from OSCI
  - TLM 2.0 AT (approximately-timed) coding style is utilized. Validate the timing accuracy of TLM models with in-house detailed performance simulator.

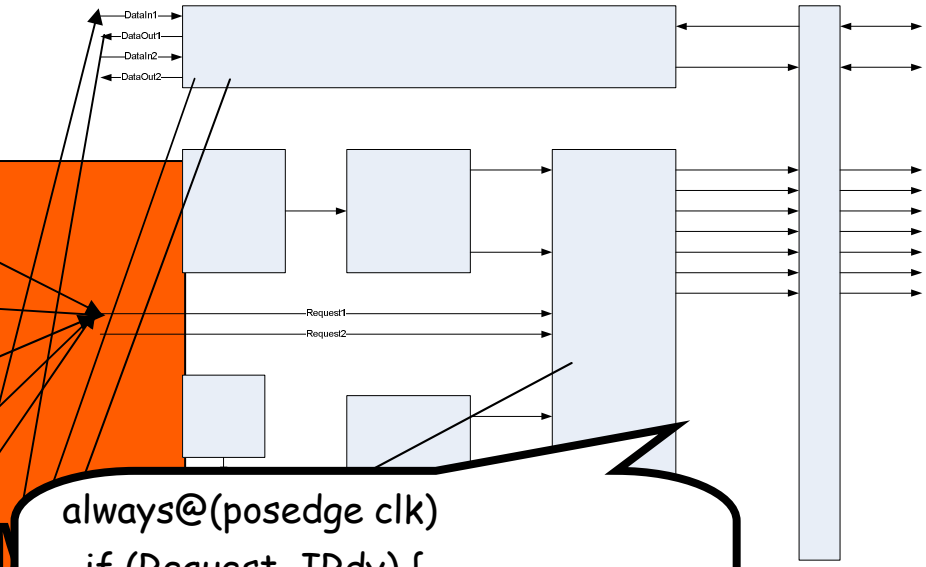
RT Level

1. Interface

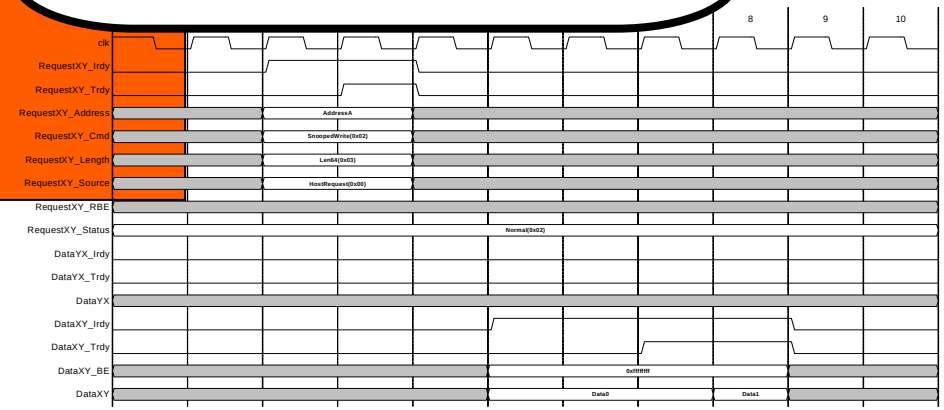


- No encapsulation of the communication signals.
- No abstraction of the communication protocol.
- Hard to explore different communication mechanisms.
- Hard to abstract the model to another level.

2. Internal components

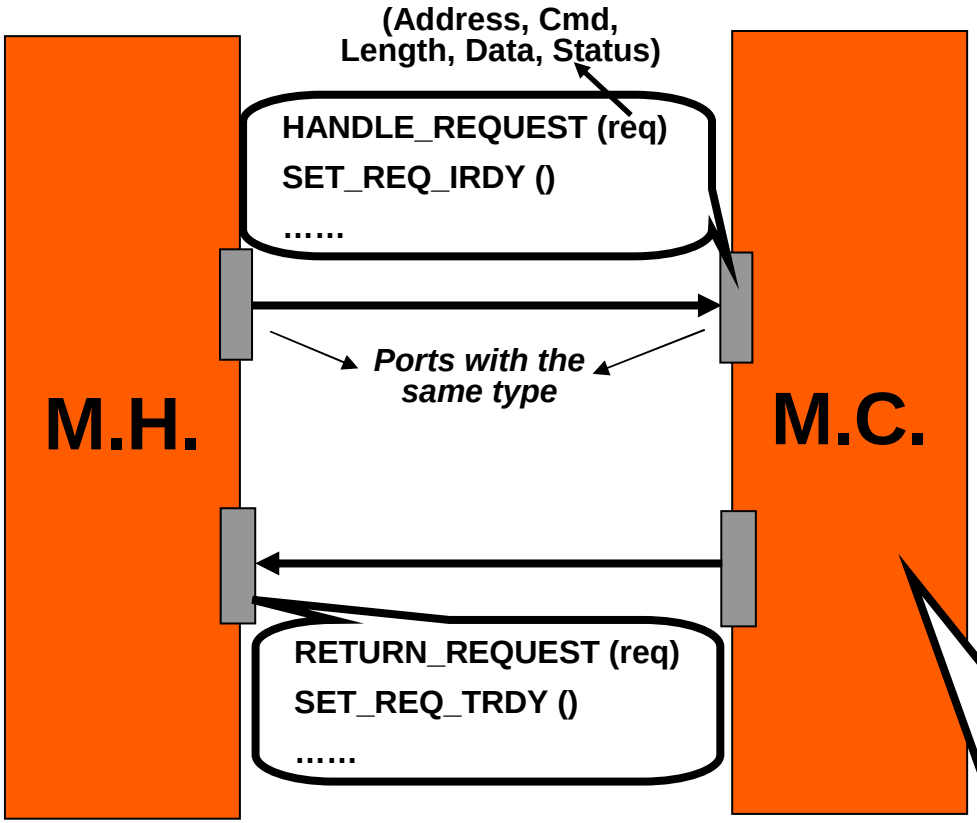



```
always@(posedge clk)
  if (Request_IRdy) {
     ……
  }
```



Signal Abstraction – toward higher level

1. Interface



2. Internal components and 3. Execution semantics are similar to previous case.

- **Internal computation components actively checking "firing" conditions every clock cycle**

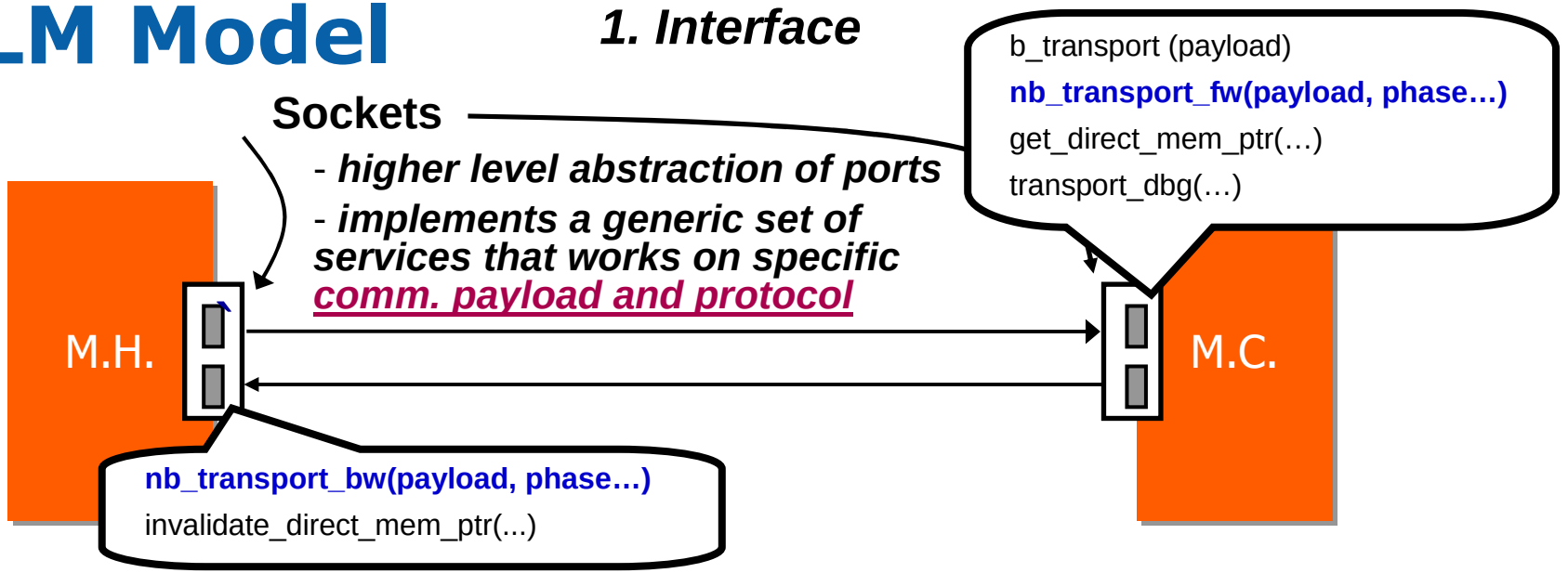
```
Clock() {
  if (!request_buffer.empty()) {
      ……
  }
}
```

- **Data signals are encapsulated. Some separation of comm. and comp.**
- **Still no abstraction of the communication protocol. Hard to explore different protocols.**
- **Hard to explore different abstraction levels.**

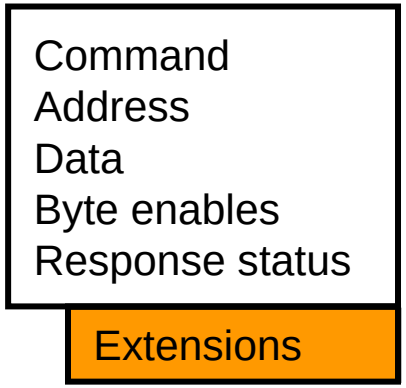
A port defines a set of services (through interface functions) provided by the component.

TLM Model

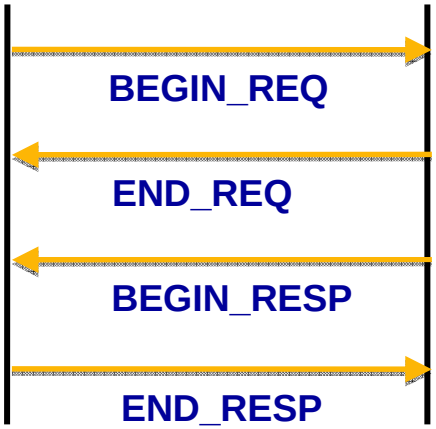
1. Interface



Generic payload



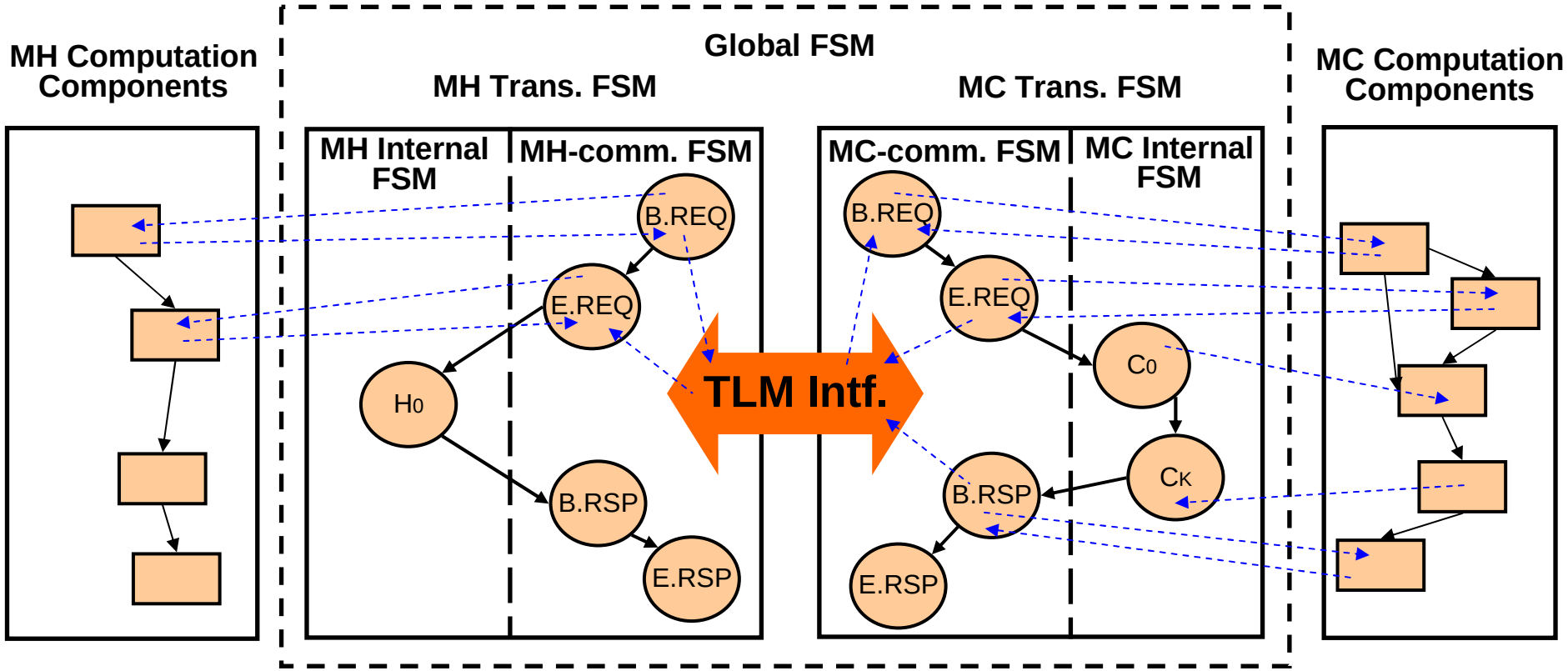
Protocol phases



TLM Model Contd.

2. Internal components and 3. Execution Semantics

In TLM, computation components are passively waiting to be triggered by Transaction FSMs that handle transaction phase transitions (including communication protocol phases and internal phases).

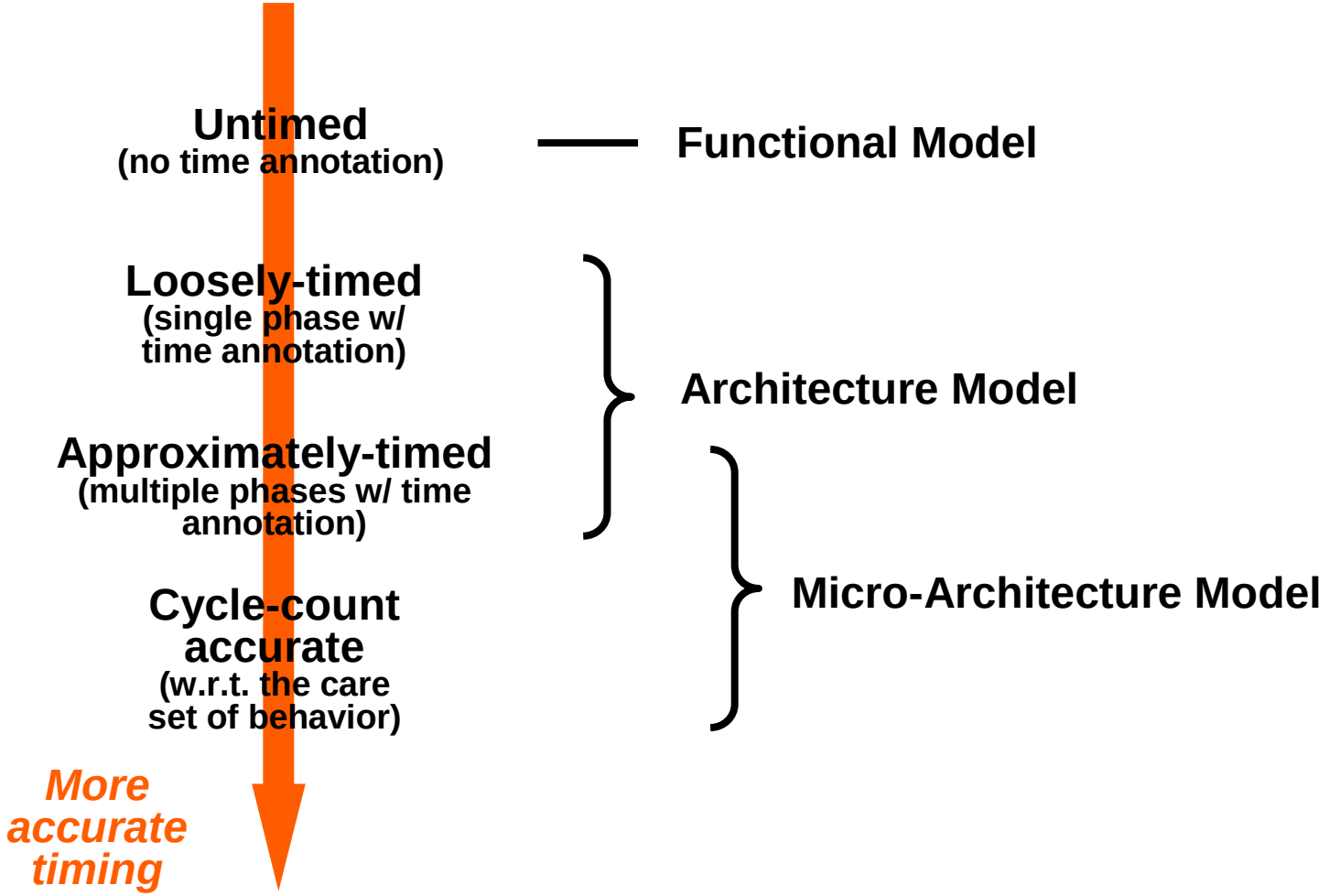


TLM Benefits Summary

- Communication is explicitly modeled by transactions and separated from computation
  - Two aspects of transactions: data and protocol.
  - Data is encapsulated in generic payload and its extensions.
  - Protocol can be described by FSMs whose states are phases of transactions.
- Multiple abstraction levels of the design can be explored easier
  - Data: payload extensions.
  - Communication protocol: protocol phases.
  - Computation: transaction phases (including protocol phases and internal phases).

Timing Modeling in TLM

Timing accuracy is decided by the choice of transaction phases



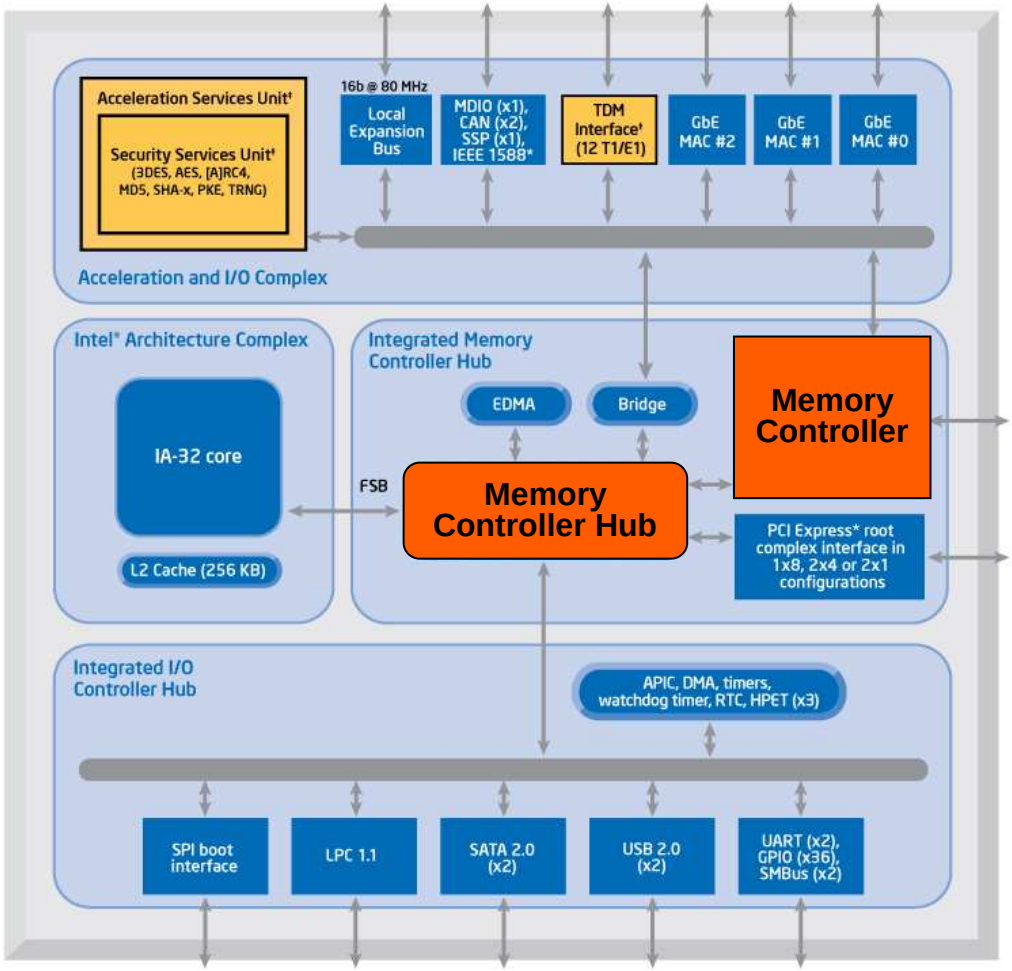
Outline

- Motivations
- Transaction-Level Modeling Methodology
- SOC Platform Exploration
- Summary Future Directions

Arch/uArch Exploration on SOC Platform

- SOC characteristics
  - Communication is the focus – important to explicitly model communication.
  - Time-to-market is crucial (while resources are usually limited)
    - System-level modeling for early exploration.
    - Multiple abstraction levels for various design purposes.
    - Standard interfaces for IP integration.
  - Many derivatives from a base platform
    - Build a library of configurable component models.
    - Mixed-level modeling to support exploration on part of the platform.

Intel Future-Generation SOCs



Block Diagram for the Intel® EP80579 Integrated Processor and Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology

(from Intel.com)

Focus is on communication between various agents and memory.

A Virtual SOC Platform with approximate-timing is being built for arch/uarch exploration.

Exploration Example: DDR Memory Controller

- Configurable memory controller model (DDR, DDR2, DDR3, LP-DDR, etc.)
  - Number of incoming ports
  - Arbitration policy (RR, priority-based)
  - Scheduling
    - In-order vs. out-of-order scheduling of requests
    - Out-of-order request queue size
    - In-order request queue size
    - Out-of-order scheduling policy
    - SDRAM command level scheduling
  - Auto refresh (refresh burst)
  - SDRAM configuration
    - # of ranks, banks, row bits, column bits
    - Timing specification
  - Address mapping
  - Data width
  - Burst length

Exploration of Memory Controller

- Impact of priority-based scheduling between two request agents A1 and A2

	Avg. A1 latency	Avg. A2 latency
A1 always over A2	13.1	17
50% A1 over A2	14.3	15.5
A2 always over A1	16.8	13.5

- Impact of hitting an open page (with all request from A1)

	Avg. A1 latency
Always hits open page	6.2
50% hits open page	8
10% hits open page	9.1
Never hits open page	9.3

Simulation Complexity

- Denotation:
  - *P*: # of processes (components)
  - *C*: simulation length in cycles
  - *E*: average # of events per process during simulation
  - t_o: overhead of switching processes
  - t_p: average runtime for one invocation of a process
- Discrete-event simulation in our TLM model

  $E * P * (t_o + t_p)$
- Clock-driven simulation in reference performance model

  $C * P * (t_o + t_p)$
- Simulation speed is decided by *E*, which relates to the modeling abstraction level.
- Our current model can simulate >100K transactions per second, with accuracy within 5% of RTL.

Model Development

- 4 man-month for memory controller and memory controller hub model, including
  - Study the architecture documents and reference models.
  - Develop the TLM models.
  - Validate the timing accuracy of TLM with reference models.
  - Study and use external and internal tools for development and analysis.
- Time could be much shorter if
  - There are library components available.
  - Modeling methodology is well defined.
  - Developers are more familiar with the architecture and reference models.
  - Abstraction level is higher.

Outline

- Motivations
- Transaction-Level Modeling Methodology
- SOC Platform Exploration
- Summary and Future Directions

What we learned

- TLM is the right way to build Virtual Platforms for architecture exploration
  - Separation of communication and computation.
  - Support multiple abstraction levels and mixed-level modeling. Approximately-timed TLM model can provide relatively accurate timing with fast simulation speed.
  - Interoperability through standard modeling methodology and communication interfaces.
  - Enable modular and configurable designs.

What we need for Exploration

- Models:
  - Library of configurable components with well-defined interface at various abstraction levels.
- Methodology
  - Methodology for exploring abstraction levels (through transaction phases). Trade-off analysis between complexity and accuracy.
  - Modeling accuracy estimation. Validation with reference models.
  - Mixed-level modeling of components at multiple abstraction levels. Dynamic switching between abstraction levels.
  - Methodology for guided simulation to search design space. Combine simulation-based exploration with analytical methods.
  - Transaction-level power modeling for trade-off analysis between performance and power.
- Tools
  - How to leverage vendor tools - analysis, debug, etc.

Unified System-Level Framework based on TLM