Abstract, Multifaceted Modeling of Embedded Processors for System Level Design

Gunar Schirner, Andreas Gerstlauer and Rainer Dömer

Center for Embedded Computer Systems
University of California, Irvine

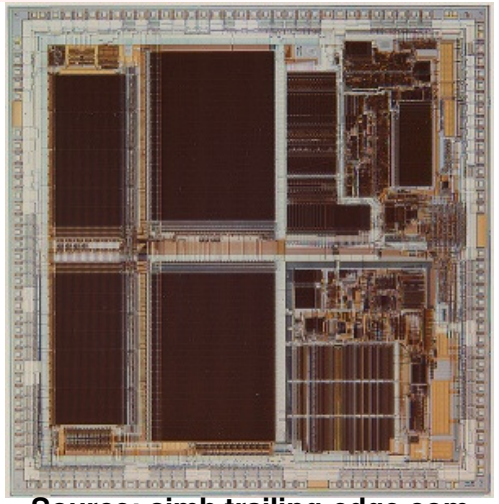
Acknowledge: Embedded Systems Methodology Group

http://www.cecs.uci.edu/~cad

UCIrvine
University of California, Irvine

Motivation

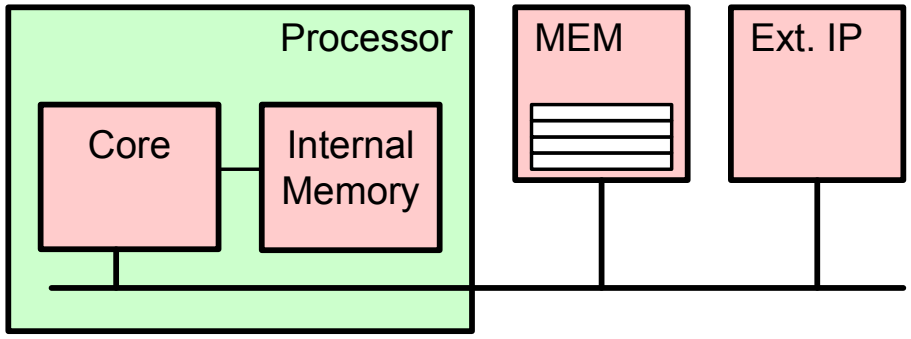
- System Level Design
  - Higher levels of abstraction
  - Meet productivity needs
- Embedded Software content growing
  - Flexible implementation of complex features
- Need processor models for System Level Design
  - Debugging
  - Validation
- Traditionally: Instruction Set Simulator (ISS)
  - Accurate
  - But slow



Source: simh.trailing-edge.com

Problem Definition

- **Devise Processor Model**
  - Simulate SW in execution environment
  - Fast and accurate
  - Identify required features
    - Accuracy, Observability
- Assume task executions delays are available
  - Own research topic, outside of scope for this publication
- Target Architecture
  - Single CPU/DSP
    - Internal memory
      - Code
      - Internal Data (stack …)
  - External memory
    - Shared data
  - External IP



Outline

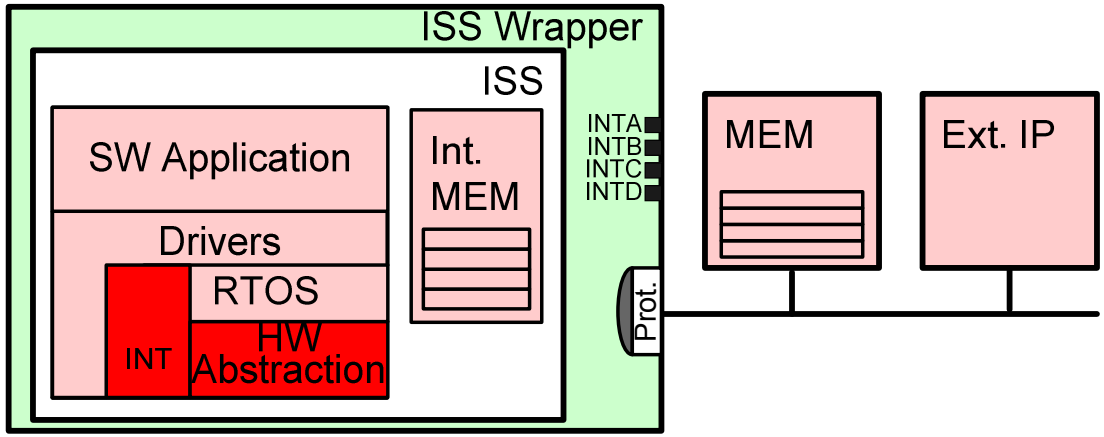
- Related Work
- Reference Model
- Processor Model
  - Feature Levels
  - TLM
  - BFM
- Example: GSM Transcoder
  - Speed / Accuracy
- Conclusions

Related Work

- Software Models in TLM
  - Abstract RTOS Model [Yu et. al, CODES+ISSS 2003]
  - Virtual Processing Unit [Kempf et. al, DATE 2005]
  - SoCOS [Desmet et. al, DAC 2000]
- Bus Models in TLM
  - [Sgroi et. al, DAC 2001], [Coppola et. al, DATE 2003] [Pasricha et. al, CODES+ISSS 2004], [Schirner et. al, DATE 2005]
- Processor Models
  - Traditional ISS-based
    - Commercial: [ARM], [CoWare], [VaST]
    - Academic: MPARM [Benini et. al, 2005]
  - High-level
    - Abstract CPU subsystem [Bouchhima et. al, ASPDAC 2005]
      - Our models are finer grained, representing more features

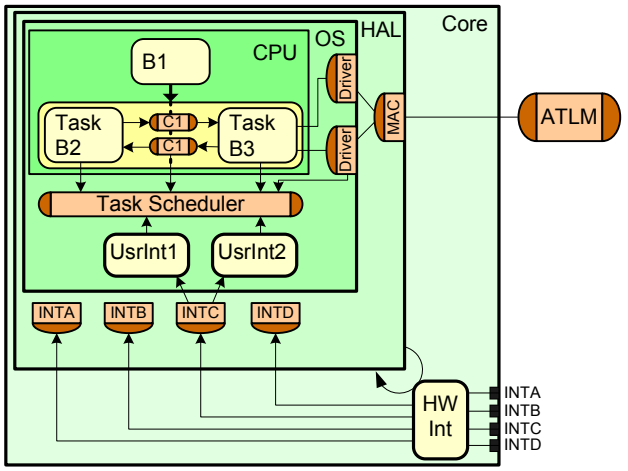
Reference Model

- Instruction Set Simulator (ISS)
  - Integrated into system-level model
- Features:
  - Target specific execution timing (up to cycle accurate execution)
  - RTOS, Task mapping and dynamic scheduling
  - Low level software drivers and interrupt handlers
  - Hardware interrupt handling
  - Bus communication



Processor Model: Overview

- Constructed model in layers
  - Each layer adds new features
  - Explain step-by-step from inside out
    - 3 Feature Levels
- Goal two processor models
  - Processor TLM
    - Note: we consider TLM as a class
  - Processor BFM

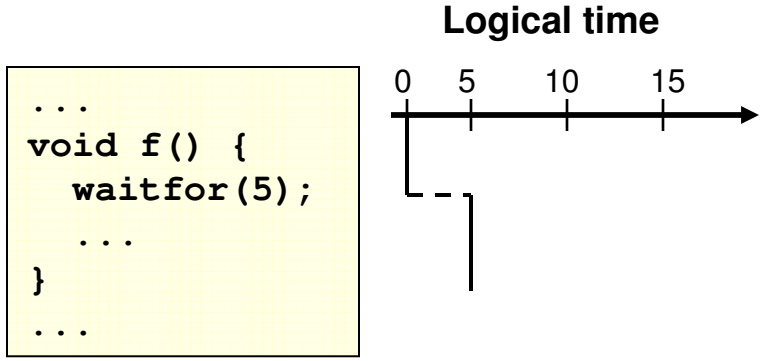


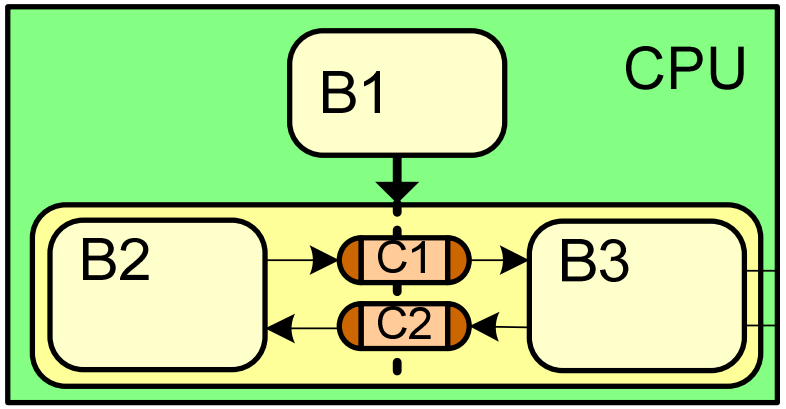
Features	
Target approx. computation timing	Appl.
Task mapping, dynamic scheduling Task communication, synchronization	Task
Interrupt handlers, low level SW drivers	Firmware
HW interrupt handling, int. scheduling	TLM
Cycle accurate communication	BFM
Cycle accurate computation	BFM - ISS

Processor Model: Application (1/5)

- Goal: Timed execution

- **Application Level**
  - User code in SLDL
    - Hierarchical set of behaviors
    - Channels
      - High-level, typed message passing
  - Timed execution
    - Back annotate C code with wait-for-time statements
    - At function level
      - Many other possibilities
  - Execute natively on simulation host
    - Discrete event simulator
    - Fast execution speed

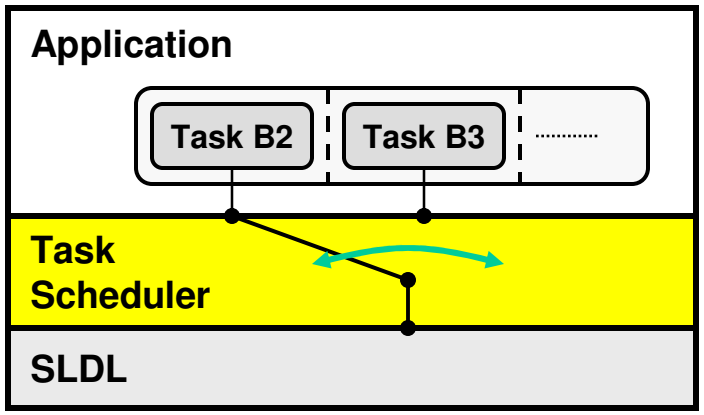


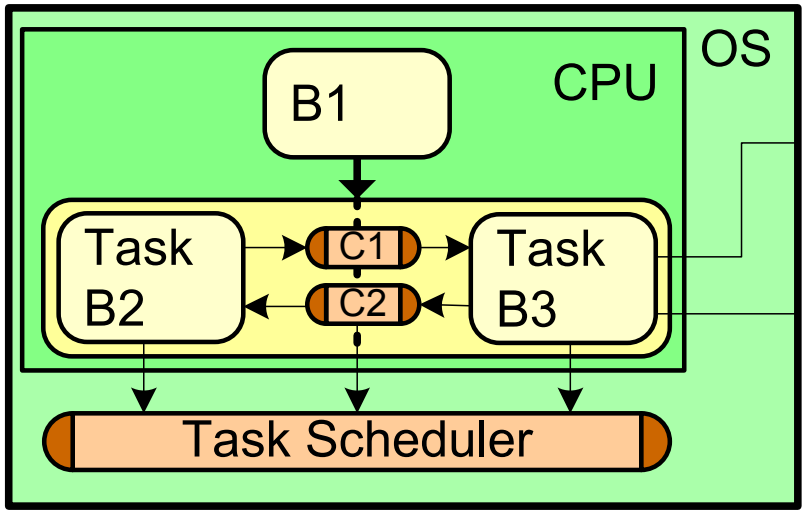


Processor Model: Task (2/5)

- Goal: Model dynamic scheduling effects

- **Task Level**
  - Group behaviors to tasks
  - Schedule tasks by abstract scheduler
    - Channel in SLDL
  - Wrap primitives that could trigger scheduling
    - Task start
    - Channel communication
    - wait-for-time



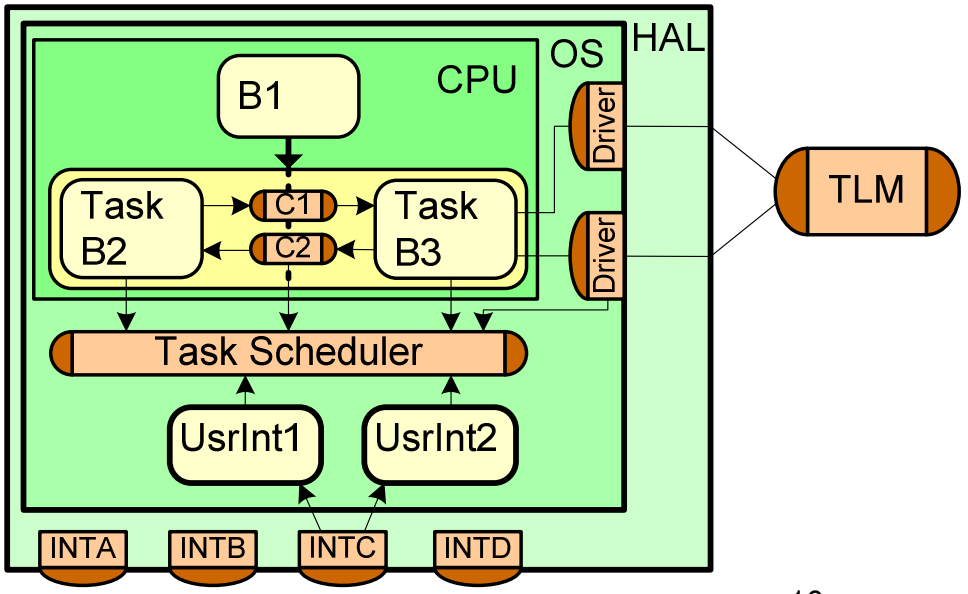


Processor Model: Firmware (3/5)

- Goal: External Communication
- **Firmware Level**
  - Software Drivers
    - Presentation, Session, Packeting
    - Synchronization (e.g. Interrupts)
  - TLM Bus model
    - User transactions
  - However, interrupts are unscheduled

App.
```
sample.send(v1);
```

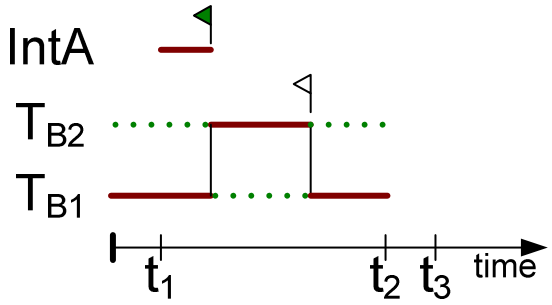
Driver
```
void send(…) {
  intr.receive();
  bus.masterWrite(0xA000,
                  &tmp,
                  len);
}
```



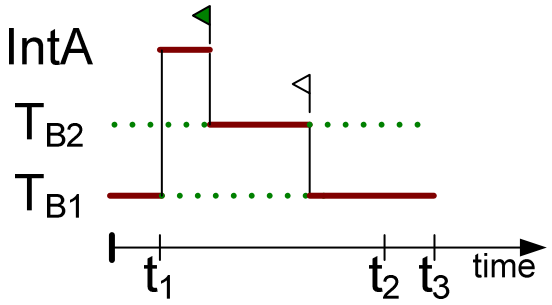
Processor Model: TLM (4/5)

- Goal:
  Interrupt Scheduling

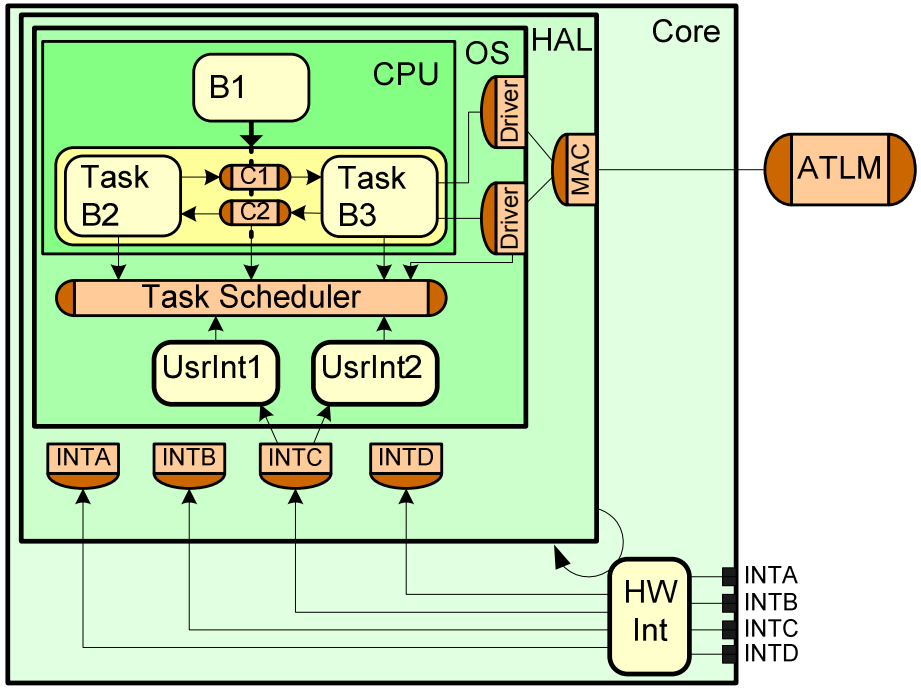
Unscheduled:



Scheduled:

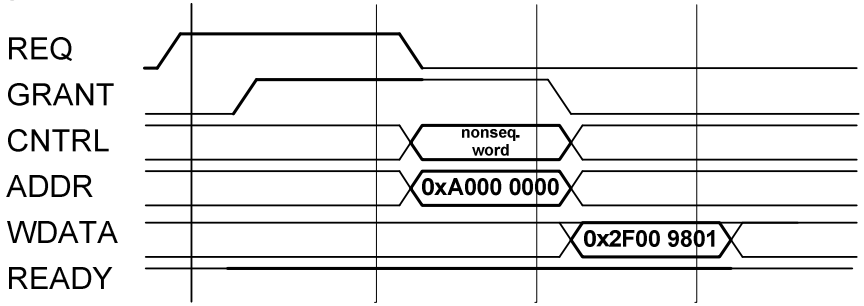


- **Processor TLM**
  - Hardware Interrupt Handling
    - Interrupt Scheduling
      - Suspend user code
      - Priority, Nesting
  - Media Access Control (MAC) for bus interface
    - Split user transaction into bus transaction
  - Arbitrated TLM bus model
  - Complete model!

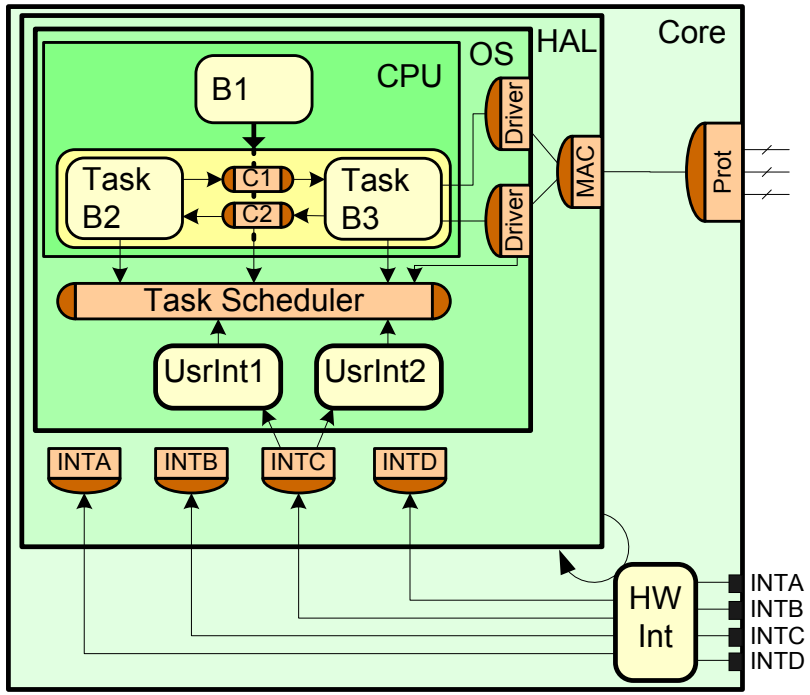


Processor Model: BFM (5/5)

- **Goal**: Accurate Bus Model



- **Processor Bus Functional Model**
  - Pin-accurate model of processor
    - Cycle approximate for SW execution
  - Bus model
    - Pin-accurate
    - Cycle-Accurate



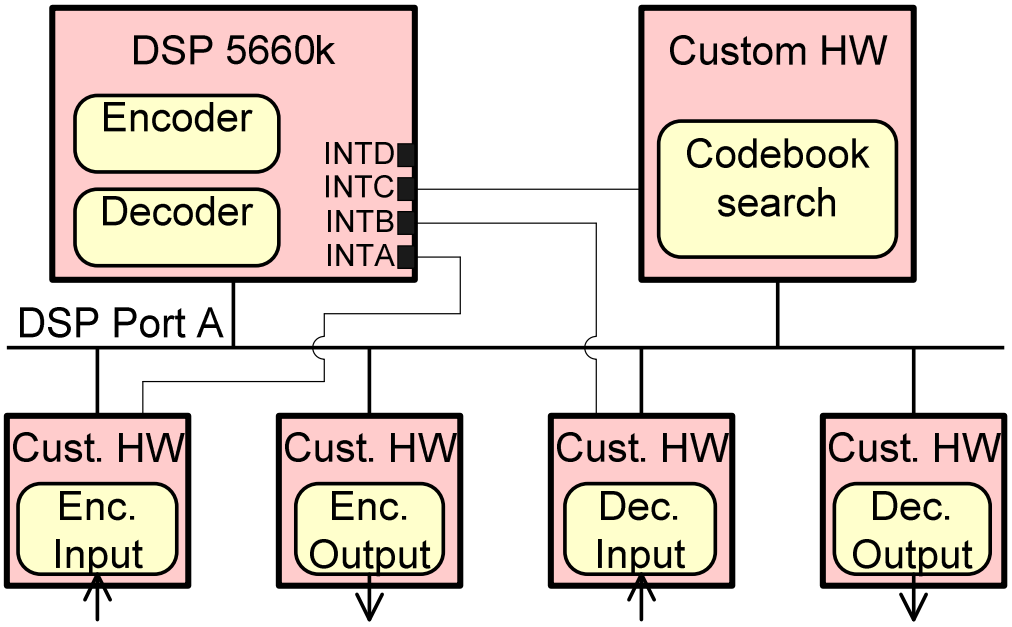
Processor Model

- Summary of features:

Features	Level
Target approx. computation timing	Appl.
Task mapping, dynamic scheduling Task communication, synchronization	Task
Interrupt handlers, low level SW drivers	Firmware
HW interrupt handling, int. scheduling	TLM
Cycle accurate communication	BFM
Cycle accurate computation	BFM - ISS

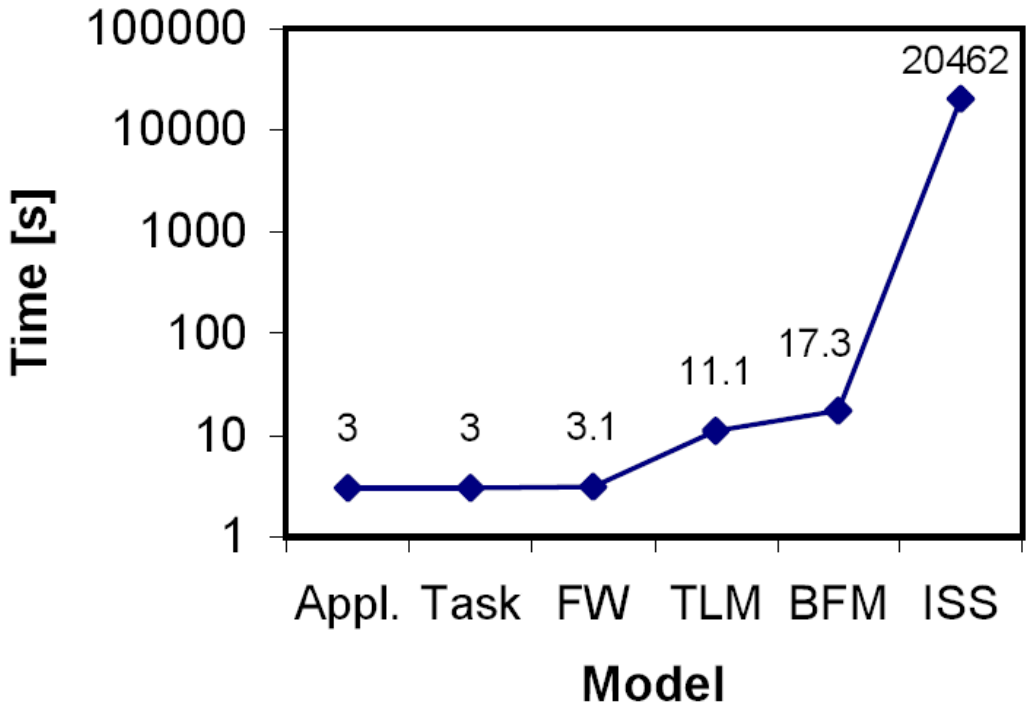
Experimental Results

- GSM 06.60 voice encoding and decoding
  - Motorola DSP 56600
  - Custom I/O blocks
  - Codebook search as custom HW
- Implemented in SLDL
  - SpecC 2.2.0
- Reference
  - Motorola proprietary ISS



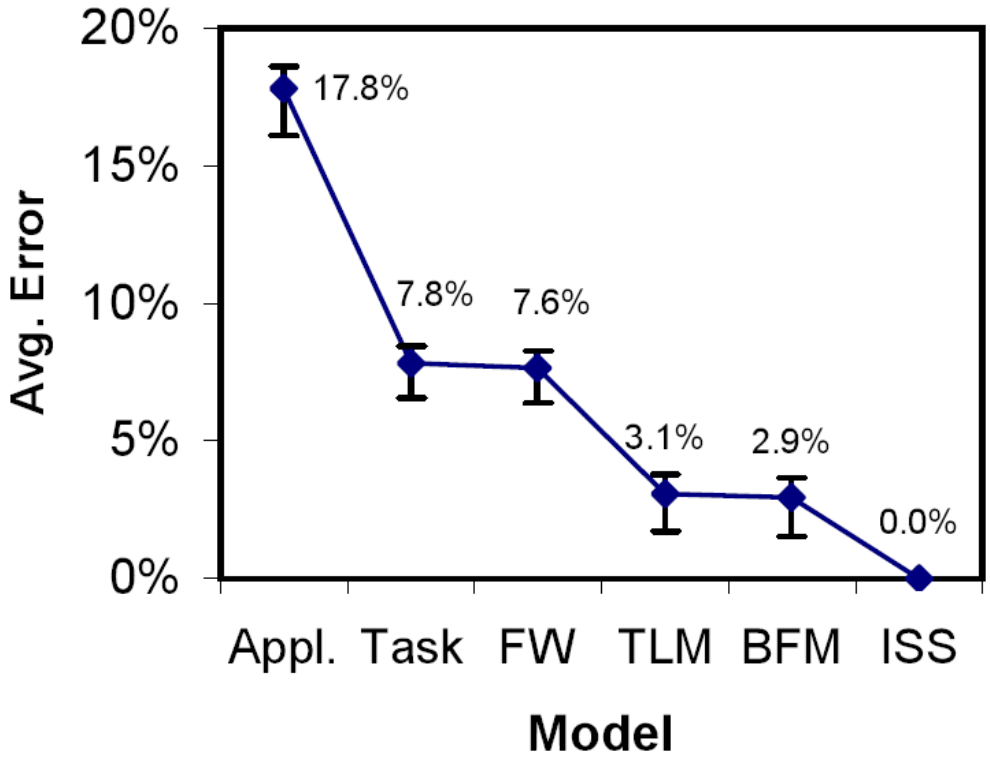
Experimental Results: Performance

- Execute on Sun Fire V240 (1.5 GHz)
  - 163 speech frames
- Analyze each feature level individually
  - Simulation time
- Dramatic increase with increasing detail
  - TLM:
    - 1800x speedup
  - FW Model:
    - 6600x speedup



Experimental Results: Accuracy

- Accuracy highly depends on timing back annotation
  - Processor modeling
    - Abstracted feature -> error ?
  - Use “perfect” timing back annotation
    - Obtained by ISS simulation
- Compare simulated time with ISS reference timing
  - Average Transcoding Frame Delay Error
- Error reduces with increased detail
  - Scheduling
  - Communication, Synchronization
- Jitter due to data dependent execution
- Remaining error:
  - Function call overhead
  - OS + Context switching



Summary and Conclusion

- Presented processor modeling approach
  - Dynamic scheduling
  - SW Drivers
  - Low level firmware
  - Hardware interrupt handling
- GSM 06.60 on DSP 56600 and custom hardware
  - FW: 6600x speed w/ 8% error
  - TLM: 1800x speed w/ 3% error
- Viable alternative to ISS based co-simulation
  - Fast and accurate
    - Exploration, Validation
  - Feature rich
    - Observability for debugging