Modeling Concurrency in NOC for Embedded Systems

Ankur Agarwal, Ravi Shankar
Center of System Integration,
Dept of Computer Science and Engineering,
Florida Atlantic University, Boca Raton, FL 33431
ankur@cse.fau.edu, ravi@cse.fau.edu

High Performance Embedded Computing (HPEC)
Workshop
19-21 September 2006
CONCURRENCY MODELING

• Concurrency Trend
  – Multi-core Architectures

• Advantages of Concurrency Modeling
  – Better Performance
  – Lower Power Dissipation
  – Higher Reuse

Deadlock/Livelock

Concurrency Issues
CONCURRENCY FAILURES

• Pitfalls of Concurrency
  – Intermittent & Catastrophic failures
  – Not a Traditional Coding Style
  – Poor Specification – Large Integration Time

Example of Concurrency Failures

– Denver Airport Baggage System
– Airplane Software Glitch
CONCURRENCY MODELING WITH FSP

- Goals: Reduce Concurrency time, concurrency failures & ease architectural selection
- FSP: A Systematic approach to concurrency modeling
- LTSA: Used for exhaustive analysis
- UML to Concurrency Modeling
- Powerful methodology for concurrency modeling
CONCURRENCY MODELING
FLOW DIAGRAM

1. High Level System Specification
2. Identify Concurrent Processes
3. Model Process Interactions
4. Analyze System for Deadlock/Livelock/Starvation
5. Re-Do Component Specification
NETWORK ON CHIP ARCHITECTURE (NOC)

- Multi-core Architecture
- Reusable Communication Sub-System
- Enhances Productivity
- Manages Complexity
CONCURRENCY IN NOC ARCHITECTURE

• 3×3 NOC Mesh Architecture includes
  – Forty Five Input Buffers
  – Forty Five Output Buffers
  – Nine Network Interfaces
  – Nine Producers
  – Nine Consumers
  – Nine Schedulers
  – Nine Routers

• Will there be concurrency issue among these component interactions?
HIGH LEVEL SUB-SYSTEM SPECIFICATION

1. Data will be received in serialized packet format
2. Several data paths will be available arranged in a matrix fashion for data forwarding
3. Data packets may be buffered at each intersection
4. Further routing will be available based on the availability & congestion of links at the destination
5. Packet will contain destination address & Priority
6. Links may be unidirectional (2 links for each direction) or bi-directional
COMPONENT SPECIFICATION

1. Link Specification
   1. Collects the data from the source
   2. Forwards the data to the buffer

2. Buffer Specification
   1. Store Input Data: Data sent by Link
   2. Forward the data to the output: Data sent to the node
   3. Inform about the buffer status: buffer is empty, buffer is full
   4. Forward the high priority data first
3. **Scheduler Specification**
   1. Receives the request from the buffer for forwarding the data to the node
   2. Forwards the request for transmitting data
   3. Checks the availability of data path for a data packet

4. **Node Specification**
   1. Determines the route information
   2. Gets the data from buffer
   3. Forwards the data to a buffer

5. **Producer/Consumer Specification**
   1. Forwards the data packet to the buffer based on buffer availability
   2. Accepts the data packet from the buffer
MODEL PROCESS INTERACTIONS ONLY (NOT INTERNAL ACTIONS)

Example:
PRODUCER = (buffAvail -> {hiPriDataOut,loPriDataOut} -> PRODUCER).

Process Abstraction
DEVELOP MODEL INCREMENTALLY

1. 1st we analyzed Producer (P) and Buffer (B) Process
2. Introduced Link (L) Process
3. Analyzed the interaction among P, B & L
4. Added Node (N) & scheduler (S) processes
5. Analyzed Interaction among P, B, L, N & S processes
6. Later: Addition of consumer (C) process
MODEL ABSTRACTION

• Initial Model

• Abstracted Model
RESULTS

Compiled: BUFFER
Compiled: SCHEDULER
Compiled: PRODUCER
Composition:
FINAL1 = b1:BUFFER || b2:BUFFER || s1:SCHEDULER || p1:PRODUCER
State Space:
24 * 24 * 9 * 2 = 2 ** 15
Composing...
-- States: 351 Transitions: 882 Memory used: 3043K
Composed in 110ms
FINAL1 minimising........
Minimised States: 351 in 46ms
No deadlocks/errors
Progress Check...
-- States: 351 Transitions: 882 Memory used: 3397K
No progress violations detected.
Progress Check in: 31ms
CONCLUSION

✓ Concurrency Modeling will ensure that there are no deadlocks/livelocks in a system
✓ Easier & quicker to integrate components into a subsystem, and a subsystem into a system
✓ Concurrency modeling at an early stage will speed up system modeling and analysis
✓ Enhanced Component Reuse via subsystem Design Patterns