IP-CHINOOK: IP-based Design for Distributed Embedded Systems

Presented by
Abdallah Tabbara
William Jiang

Introduction

- IP-CHINOOK: system design tool
- Model
  - Enables control and dataflow
- IP composition:
  - Software components
- Communication synthesis:
  - Distributed embedded system
- Rapid evaluation:
Design Flow

- **Input**
  - Behavioral description
  - Target description
  - Allocation function

- **Control synthesis**
  - Centralized mode manager
  - Distributed mode manager
Design Flow

- Communication and interface synthesis
  - Driver generation
  - Routing process synthesis
  - Port allocation
  - Glue logic insertion
- Output
  - Complete schematic
  - Target-dependent program code

Input

behavioral description  allocation function  target description
Input

- Behavioral description (function)
  - Modal processes
  - Control composition
  - Data composition
- Target description (architecture)
  - Heterogeneous distributed architecture
- Allocation function (mapping)
  - Process -> processor
  - Channel -> busses

Behavioral Description

control composition

modal process

modal process

modal process

data composition & timing constraints
Modal Processes

• Consist of:
  – Ports
  – Handlers
  – Modes

• Ports:
  – Connected by channels

Modal Processes

• Handlers:
  – Invoked by events on input ports
  – Sends messages to output ports
  – Run to completion semantics
Modal Processes

- Modes
  - Mapping from ports to handlers
  - Active or inactive
  - Multiple modes simultaneously active

![Diagram](image)

- Execution
  - Messages queued on input ports dispatched to handlers in active modes
  - Handlers execute sending one or more messages to output ports
  - Change active modes:
    - Vote collection: each handlers returns a set of votes
    - Vote reconciliation: by priority
    - Vote distribution
  - Notes:
    - One-step delay before receiving handlers can be triggered
    - Buffers at least one-deep
Control Composition

- Abstract control types (ACT):
  - Control coordination
  - Define protocol for interprocess composition
  - Can form hierarchical FSM similar to Statecharts

Digital Wrist-watch Example

- Statecharts & Esterel example
- Behavioral description:
  - Six modal processes
    - Watch + UI
    - Stopwatch + UI
    - Alarm + UI
  - ACT to control UI
Digital Wrist-watch Example

Input

behavioral description

allocation function

target description
Target Description

- Defines desired target architecture
- No automatic partitioning and allocation
- Defined by:
  - Processing elements: uC, DSP, FPGA
  - Topology
  - Operating systems
  - Communication protocols: CAN, SCSI, USB, IrDA, and Ethernet
Input

Allocation Function

- Maps:
  - Modal processes to processing elements (many-to-one)
  - Logical communication channels to architectural communication links (many-to-many)
    - Specification uses message passing
    - Can synthesis shared memory
Impressions

• Methodology:
  – Function
  – Architecture
  – Mapping
• Model of computation:
  – Modal processes with ACTs

• Model of computation:
  – Very low-level (DE)
  – Cannot prove model liveness and safety properties
• Implementation details included in specification:
  – Scheduling
• Only targeted towards software components:
  – No synthesis except for interfaces
• Generic RTOS
Synthesis and Simulation

- Transformation of design representation from high level to low level
  - Mode manager synthesis
  - Communication and interface synthesis
- Selective focus simulation

Mode-manager Synthesis

- Mode manager:
  - Part of the run-time system that manages control communication
  - Coordinate modal processes
- Uni-processor: centralized mode manager
  - Synchronous mode
  - All processes are blocked until mode changes are resolved
Mode-manager Synthesis

- Distributed architecture: distributed mode manager
  - Different synchrony provided to trade off space, performance and determinism
    - Mode synchrony
    - Event synchrony
    - Communication synchrony
    - Data flow synchrony

Comm. Synthesis Flow

- High level description
- Architecture
- Partitioning
- Mapping
- Communication Synthesis
- Evaluation
- end

- Abstract comm. protocols
- timing requirements.

- protocol parameters
- protocol formats
- deadline constraints
- message routing
- device driver
- glue logic
- etc.
Communication Synthesis

• Inputs:
  – High level process description
  – System architecture
  – Mapping (process $\rightarrow$ processor, comm. $\rightarrow$ Bus)

• Outputs:
  – Application specific RTOS for each processor
    • Protocol parameters and formats
    • Device drivers and glue logic
    • Message routing, timing constraints, etc

Layered Communication

![Layered Communication Diagram]
Communication Refinement

• **Abstract protocol:**
  – Input port: queuing semantics
    • queue size
    • overflow behavior
  – Output port:
    • blocking style
    • deadline constraints

• **Architecture dependent protocol:**
  – bus protocol (message passing)
  – routing requirements
  – timing constraints

Example
Comm. Synthesis Steps

- Multi-hop deadline distribution
- Bus protocol attribute synthesis
- Message router generation
- Device-driver instantiation
Communication Model

- Non-blocking message passing protocol
  - Asynchronously event driven
  - Interrupt handler: real-time kernel
- Unique message port for each message type
  - User defines
    - Queue size, over-write policy
    - Message attributes:
- Process handler
  - Run to complete
  - Many different handlers for one process

Multi-hop Deadline Distribution

- Users:
  - Explicitly enumerate routing paths
  - Explicitly define timing constraints
- Synthesis:
  - Distribute timing constraints among hops
  - Minimize bus traffic
  - Generate routing information
  - Create hop processes
  - Global communication synthesis
Bus Protocol Attribute Synthesis

• Message passing information
• Specific bus protocol attributes
  – Generates message ids
  – Generates processor ids
  – Generates queues
  – Assigns message or processor priorities
  – Meets bus specifications
  – Supports: CAN, F2C, SCSI, USB, IrDA, Ethernet

Message Router Generation

• Message segmentation and re-assembly
• Deliver messages to appropriate input ports
Device Driver Instantiation

- Protocol library
- HW/SW interface
  - Embedded microprocessor’s I/O ports
    - Connected to registers
    - Maximize sharing among peripheral devices
  - Memory mapped I/O
    - Multiplexing when necessary

Communication Synthesis

- Allow quick evaluation of:
  - Different architecture
  - Different bus topologies
  - Different process mapping
  - Different communication protocols
Simulation

- HW/SW co-simulation
- Selective focus simulation
  - Highest level of abstraction whenever possible to achieve speed
  - Detailed low level simulation only in sub-regions
  - Dynamically zoom in and out of regions
  - Implemented by run-level

Conclusion

- Comprehensive hardware/software co-synthesis framework for heterogeneous distributed embedded systems
- Automatic communication and interface synthesis for fast evaluation of different architectures, partitioning and process mapping
References

- P.Chou, etc. “IPChinook: an integrated IP-based design framework for distributed embedded systems” in DAC 1999