System Level Specification for Multimedia Applications

Dohyung Kim and Soonhoi Ha

Codesign And Parallel Processing Lab
Seoul National University
System consists of diverse subsystems:

- Processor Core
- ASIC hardware
- Memory
System consists of diverse subsystems

- Control Module
- Computation Module
- Memory

I/O

Processor

ASIC

APCHDL’99

Soonhoi Ha
Content

- Introduction
- Related Works
  - STATEMATE
  - Ptolemy
  - COSY
- Codesign Backplane
  - specification of function modules (dataflow)
  - specification of control modules (fFSM)
  - cospecification of control and function modules
- Example
- Conclusion
System Specification

- Use of imperative languages (C, VHDL, …)
  - Describes how to do, not what to do
  - Virtually no mechanism to prove the correctness of design at compile time except syntax errors
  - Productivity is low

- Higher level formal specification is advocated.
  - Amenable to validation, simulation, and analysis by computers
  - Top-down refinement from specification to implementation
  - Design reuse and maintenance
  - Described what to do: design space exploration
  - Productivity is high
Specification method depends on application areas

Control-oriented applications: FSM
- POLIS approach: a network of Codesign FSM (CFSM).
- Esterel: Synchronous Languages

Signal processing applications: Dataflow
- Ptolemy, COSSAP, SPW: SDF and related dataflow models
- GRAPE II: CSDF

Control + signal processing applications
- Unified model: COSY, CDFG
- Heterogeneous model: Ptolemy, STATEMATE
Related Work: STATEMATE

- Control-oriented reactive applications

Conceptual model

Activity chart: functionality

Statechart: behavior

STRUCTURE
Module chart

STATEMATE

†
Statechart: Hierarchy and Concurrency

- Hierarchy: OR-composition

Reduce the number of transitions
Statechart

☐ Concurrency: AND composition

Solve the state explosion problem
Ambiguity: Instantaneous Transition

- $\beta$ is an internal event
- At the first transition, $\alpha$ is consumed.
- Need instantaneous inter-level transition

Violate compositionality
STATEMATE

- Innovative and Intuitive specification of control activity
- Ambiguities in the statechart
- Lack of formalism in activity chart and its interaction with statechart
Related Work: Ptolemy (* chart)

- Hierarchical FSM with multiple concurrency models
- Motivation
  - Decoupled FSM semantics from concurrency semantics unlike Statechart
- Candidate Concurrency Model: SDF, SR, DE, ...
  - Compositionality: composite modules can be treated as primitive modules
  - Heterogeneity: composite modules can be embedded within a foreign model of computation
Careful and precise interaction mechanism between SDF and FSM models is needed.

Subtlety: “absent” event appears explicitly as a token
Most straightforward combination from a control perspective.

Absence of token implies null event. - no transition should be made by null event.
Ptolemy * chart

- Subsystem has specific and intuitive specification
- Composition of formal models

- Some composition is difficult to understand
  - SR model + FSM
  - multirate SDF + FSM

- Execution mechanism of FSM subsystem is context-sensitive
Related Work: COSY

- IP-based real-time design methodology
- YAPI: extension of the Kahn process network model
  - Kahn process network + “select” operation
- Coarse grain mix-and-match of several IP blocks
COSY

- Unified approach
  - control module and computation module are inside a process: not distinguishable from the outside

- YAPI model provides a coordination method of different components

- COSY does not specify any formal model inside a component process - only coarse grain composition
Proposed Backplane Approach

- Hybrid approach of STATEMATE and Ptolemy
- Restricted composition of heterogeneous models of computations
  - Backplane is derived from discrete-driven model
  - fFSM: control logic specification
  - SPDF: computation (signal processing) specification
  - fFSM can interact with SPDF only through Backplane domain

Motivation

- Use models consistently from specification to synthesis
  - synthesis path from fFSM and SPDF is well established.
- Use natural combinations of heterogeneous models.
Codesign Backplane

- Central backbone for domain interactions
- Maintain global information of component models during the codesign process
- It supports
  - cospecification of control and function modules
  - cosimulation with HW/SW simulators
  - will support cosynthesis
Synchronous Dataflow Model

- Useful to describe DSP algorithms
- A node represents a function block (ex: FIR, DCT)
- An arc represents the flow dependency
- Multi-rate DSP applications can be naturally expressed.
- Statically scheduled at compile-time.

Schedule: A-C-A-C-B-D
Buffer size: AB(2), AC(1), CD(2), BD(1)
flexible FSM (f FSM)

- New FSM extension used in PeaCE

- Motivation
  - as intuitive and convenient as Statechart
  - possess formal properties like CFSM

- fFSM properties
  - concurrency
  - hierarchy
  - internal event
  - variable state
Internal Event

- Support communications between concurrent states and across hierarchy

Diagram:

- reset → reset / tostart
- start → run counter → tostart → halt
- start → toggle
- halt → toggle
Variable State

- Support a memory in FSM semantics
- Have the same meaning with concurrency

```
input : time, start
output : timeout
state variable : remain
```

```
init

start / remain=start

wait

time & remain>0 / remain=remain-1

time & remain=0 / timeout

input : time, start
output : timeout
state variable : remain
```
Communication between fFSM and SDF

- Synchronous interaction

1. Data transfer

2. Execution

3. Result (flag) transfer
Asynchronous Interaction

- Change the scheduling state of a dataflow module
  1. signal transfer
  fFSM
  2. change run status

- Change a block parameter
  1. signal transfer
  fFSM
  2. change block parameter
  source
  decode
  variable
  gain
  Dataflow
Synchronous Interaction

Event source

"a"

Backplane

fFSM

a/out

flag

X

Y

Display

"out"

"flag"

SDF

A

B

C

APCHDL’99

Soonhoi Ha
# Asynchronous Interaction

<table>
<thead>
<tr>
<th>Scripts</th>
<th>actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>run n_name</code></td>
<td>Resume the <code>n_name</code> block</td>
</tr>
<tr>
<td><code>suspend n_name</code></td>
<td>Suspend the <code>n_name</code> block</td>
</tr>
<tr>
<td><code>stop n_name</code></td>
<td>Stop the <code>n_name</code> block</td>
</tr>
<tr>
<td><code>set n_name parameter value</code></td>
<td>Update the <code>parameter</code> with <code>value</code> in the <code>n_name</code> block</td>
</tr>
</tbody>
</table>

## Backplane

- **Event source**
- **fFSM**
  - States: X, Y
  - Transitions: a
  - Reset: X
- **SDF**
  - Nodes: A, B, C
  - Edges: A → B, B → C, C → A

### Scripts

- **Script:** run "foo"
- **Script:** stop "foo"
Example (1) : Traffic Light
Example (1): Top-level FSM design
Example (1) : Traffic Light
Example (2): MPEG I Layer 3 Player

MPEG I Layer 3 Decoder

- File
- Bitstream unpacking
- Frequency sample reconstruction
- Frequency to time mapping
- Speaker

Tcl

TclScript

control FSM

File

Bitstream unpacking

Frequency sample reconstruction

Frequency to time mapping

Speaker

run mp3

stop

stop mp3

start

suspend

suspend mp3

up & gain<10 / gain+=1 & vol=gain

down & gain>0 / gain-=1 & vol=gain

set mp3 volume gain

volume

Tcl

TclScript

control FSM
Cosimulation between Synthesized code and Backplane

Codesign backplane

Event source

FSM

a

X

reset

Y

C code process

node_name: foo

dataflow

TCP/IP

non-blocking IO

A

B

C

script: run foo

script: stop foo

Soonhoi Ha
C Process associated with Dataflow

init_variable:

```c
variable initialization
while (true) {
    if (status==RUN) {
        non-blocking read for a control signal
        if (control packet==SUSPEND) {
            status=SUSPEND;
            continue;
        } else if (control packet==STOP) {
            // status==STOP
            status=STOP;
            goto init_variable;
        } else process parameter update
    }
    else { // status==SUSPEND || STOP
        blocking read for control a signal
        if (control packet==RUN) status=RUN;
        else if (control packet==STOP) {
            if (status==STOP) continue;
            status=STOP;
            goto init_variable;
        } else process parameter update
    }
    // synthesized C code for a dataflow module
}
```

APCHDL’99 -------------------------------------------------------- Soonhoi Ha
Conclusion

- Formal specification is important for codesign of complex system-on-chip.
- Specifies control and function modules by using formal languages
- Supports inter-model communication between control and function modules on the codesign backplane
- On-going work
  - automatic synthesis from fFSM to VHDL and C
  - automatic interface synthesis

- The tough problem we have to attack is not to make the cosynthesis task doable, but to perform it as efficiently as manual synthesis.

---

APCHDL’99

Soonhoi Ha
Thank You !!