EECS249

Designing with POLIS

Harry Hsieh
Alberto Sangiovanni-Vincentelli
Outline

- POLIS Design Methodology
- Seatbelt Alarm Controller
  - Specification
  - Co-simulation
  - HW & SW synthesis
- Dashboard Controller
  - Simulation & architectural selection
  - HW & SW synthesis
Embedded Systems

- Microprocessors are **embedded** to
  - Replace circuits or mechanical devices
  - Achieve flexibility and faster time to market
- Some hardware or DSP may be used
- Examples
  - Consumer electronic (microwave oven, smartquill)
  - Automotive (engine control)
  - Telecommunication (cellular phone)
Polis Co-Design Framework

- Control-dominated embedded systems
- Unified System Representation (CFSM)
- Input: Languages with FSM semantics
  - Esterel, Esterel-C, synthesizable HDL
Polis (continue)

- **Analysis:**
  - Formal verification (VIS)
  - System simulation (Ptolemy, VHDL)
    - Abstract timing model of architectures
    - Trade-off in mapping
- **System design with user input**
- **Automatic synthesis**
  - SW, HW, Interface, and OS
Co-Design Finite State Machine

- Extended Finite State Machine/ FSMD
- Globally asynchronous
  - communication between CFSMs
  - unbounded delay to represent HW/SW
    - HW-SW: different delay
- Locally Synchronous
  - non-zero delay to ensure compositionality
  - Synthesis procedure refine (or fix) delays
- Reactive processing: efficient synthesis
Communicating CFSM

- Semi-synchronizing
- Private one-place buffer
- Non-overwriting ensure by
  - Explicit hand shaking in design
  - Synthesis directives
    - Scheduling
    - HW implementation
Embedded System Design with Polis

- Graphical EFSM
- ESTEREL
- Partitioning
- Formal Verification
- Compilers
- CFSMs
- SW Synthesis
- SW Code + RTOS
- Intfc Synthesis
- HW Synthesis
- Logic Netlist
- SW Code + RTOS
Designing Seat Belt Alarm Controller

- $POLIS/examples/esterel/belt
- Design from scratch
  - start from esterel files (CFSMs)
  - generated Ptolemy files (make pto...)
  - connecting CFSMs in Ptolemy
  - design exploration
  - automatic hw and sw synthesis (blif file and c-code)
CFSM Example: Car Belt Controller

1. Write an Esterel file
2. Simulate the controller

If no condition is satisfied, implicit self-loop in the current state
Timer

- Functionality: “Every time a start_timer occurs, wait for count_5 msec and output end_5, wait for another count_10 msec and output end_10. If these operation is not completed when the new start_timer comes in, start over and discard the current computation”

1. Write the Esterel File

2. Simulate the timer
Ptolemy Simulation

Some useful Ptolemy commands:

- Create Star *
- Select s
- Edit Parameter e
- Open Pallette O
- Unselect u
- Run R
- Create/Copy c
- Undo U
- Get Info, Get Info, Get Info,
- Full Screen f
- Move m
- Save window S
- Pan p
- Delete D
- Zoom -in z
- Create-icon @
- Zoom-out Z
- Open-Facet F
- Close-window CTL-D
- Look-inside i

1. Create the Ptolemy Simulation Testbed
Create the Seatbelt Galaxy
Create the Testbed Universe
High Level Co-simulation

- High level Co-simulation allows:
  - Hardware/software partitioning
  - Architecture selection (CPU, scheduler,...)
  - Can not be used to validate the final implementation
    ➔ Need a much more detailed model of HW and SW architecture
Our Co-Simulation Approach

- Benchmarked software timing estimates
- Synthesized C code annotated w/ clock cycles required on different processors
- Clock cycle accumulation during simulation to synchronize the software
  - With the hardware & environment
- Uses Ptolemy or VHDL simulators as:
  - Graphical interface
  - Simulation engine
Performance and cost estimation

- Example: 68HC11 timing estimation
- Cost assigned to s-graph edges
  - (Different costs for taken/not taken branches)
- Estimated time:
  - Min: 26 cycles
  - Max: 126 cycles
- Accuracy: within 20% of profiling
Trade-off Evaluation

- Parameters associated with each hierarchy level:
  - Changed on the fly (no recompilation)
    - Implementation, CPU type, clock speed, scaling factors, schedulers, priority
- Hierarchical inheritance
- Analysis on buffer overflow, CPU usage
- Automatically transmitted to synthesis steps
Embedded System Design with Polis

Formal Verification

Partitioning

Simulation

Graphical EFSM

ESTEREL

Compilers

CFSMs

SW Synthesis

Intfc Synthesis

SW Code + RTOS

HW Synthesis

Logic Netlist
Software Synthesis Procedure

Specification, Partitioning

S-graph Synthesis

Timing Estimation

not feasible

feasible

Scheduling, Validation

fail

feasible

Code Generation

Compilation

Testing, Validation

Production

pass
S-graph Example: Car Belt Controller
Hardware Synthesis
Car dashboard example

- Implemented a dashboard controller:
  - Speedometer and odometer
  - Safety functions (seat-belt alarm)
  - Fuel gauge control

- 23 interacting CFSMs, of 13 different sorts:
  - Speed, Odo, RPM (speedometer, odometer)
  - Belt (safety)
  - Crossdisp (fuel)
  - FRC, Timer
Car dashboard example

- Control functions specified with Esterel
  - automated translation from Esterel to CFSMs
- CFSM interconnection specified graphically

fuel
key, belt
clock
wheel
ingine

fuel

fuel_disp

speed_disp

RPM_disp

odo_disp

Crossdisp

Crossdisp

Crossdisp

Crossdisp

RPM

Speed

Belt

FRC

Timer

Odo
Designing Car Dashboard Controller

- $POLIS/examples/demo/dac_demo
- Design Exploration
  - CPU usage
  - buffer overflow
- Automatic synthesis
Finding POLIS

- WWW Homepage
  - www-cad.eecs.berkeley.edu/~polis
  - users’ manual / tutorial

- Using POLIS
  - on IC cluster
    - `setenv POLIS /projects/hwsw/hwsw/$ARCH`
      - or for version 0.3: `/projects/hwsw/hwsw-0.3/$ARCH`
    - add `$POLIS/bin` to path
  - others
    - download most recent release from homepage

- A set of makefile for design management