Interface Synthesis in Heterogeneous System-Level DSP Design Tools

ICASSP '96
Special Session on Methods and Tools for Rapid Prototyping of DSP Systems

José Luis Pino
Michael C. Williamson
Edward A. Lee

pino@eecs.berkeley.edu
http://ptolemy.eecs.berkeley.edu/~pino
Goal

Promote the rapid prototyping of typical embedded signal processing systems.

Method

Enable the seamless interaction of high-level DSP synthesis environments with external simulation and hardware engines.
Typical Embedded Signal Processing System

- control panel
- ASIC
- microcontroller
- real-time operating system
- controller process
- user interface process
- system bus or switching fabric
- DSP assembly code
- host port programmable DSP memory interface
- host port programmable DSP memory interface
- dual-ported memory
- CODEC
- DSP assembly code
- analog interface

A real system may have many more processors and/or ASICs.
Simulation & Hardware Engines

- System-Level Simulation
- VHDL Simulation
- DSP Simulation
- Real-Time Hardware
Outline

• Disadvantages of using a discrete-event simulation backplane
• Synchronous Dataflow (SDF): guaranteeing non-deadlocking run-time behavior
• Synchronization protocol and interface construction
• Examples & performance
Discrete-Event Simulation Backplane

- System may deadlock at run-time
- Discrete-event semantics have large overhead
- Engines may not be mixed arbitrarily: discrete-event simulation backplanes are not well suited for fine-grain mixing of simulation and hardware engines
Synchronous Dataflow

- Multirate dataflow semantics: Actors consume and produce a fixed number of tokens per invocation.

- Complete run-time behavior is known at compile time.

- Compile time scheduling:
  - Repetitions vector: \( q(V) = [q(A), q(B), q(C), q(D)] = [1, 1, 2, 1] \)
  - Many valid schedules: ABCCD, ACBCD, or AB(2C)D.
Deadlock Avoidance

Execution in multiple threads must be coordinated:

Within the gray subsystem, A and C have no apparent dependencies. If they are scheduled as C then A, deadlock occurs!
SDF Multiprocessor Scheduling

SDF Precedence DAG (Directed Acyclic Graph)

Multiprocessor Schedule

<table>
<thead>
<tr>
<th>Time →</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A1</td>
<td>B1</td>
<td>D1</td>
</tr>
<tr>
<td></td>
<td>C1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Two pairs of communication actors are required for each hardware or software engine to be supported.

Provide necessary synchronization — self-timed implementation.

Fixed FIFO buffer size computed at compile time:
- Blocking write: Send actor suspends when buffer full
- Blocking read: Receive actor suspends when there is not enough data in FIFO
Restricting VHDL to SDF Semantics

VHDL simulator

sequential VHDL entity

VHDL entity

VHDL signals

Unix sockets

sequential C or C++
Algorithm Procedure

hierarchical dataflow graph

flatten the code generation subsystems

multiprocessor scheduler where mapping is user specified

multi-lingual, flat dataflow graph

VHDL code generator(s)
C code generator(s)
Assembly code generator(s)
Interface Construction: Between Engines

User Specification

Spliced-in send/receive pairs
Interface Construction: With Ptolemy

User Specification

Sim-SDF | External | Sim-SDF

Spliced-in simulation-SDF send/receive actors

SimIn C S-56X SimOut

C S-56X

VHDL

DSP

S-56X C

VHDL C

VHDL C

VHDL C
Filterbank Example: Ptolemy, S-56X & Synopsys VHDL

Top-level application specification

2x faster than using only VHDL
Sample Rate Conversion Example: Incremental Compilation

FIR polyphase filterbank subsystem (mapped to C code generator) is compiled into a monolithic simulation-SDF block.

CD (44.1 kHz) to DAT (48 kHz)

3x faster than using only simulation-SDF
Summary

- Enabled the **seamless interaction** of high-level DSP synthesis environment with external simulation and hardware engines.
- Realized **200% - 300% speedup** when using appropriate engines and communication infrastructure.
- Restricted the specification semantics to SDF, which **guarantees non-deadlocking run-time behavior**.
- Abstracted the communication protocol to easily support new simulation and hardware engines.