The Ptolemy Project

Shuvra Bhattacharyya
Joseph T. Buck
Wan-Teh Chang
Michael J. Chen
Brian L. Evans
Soonhoi Ha
Paul Haskell
Chih-Tsung Huang
Wei-Jen Huang
Christopher Hylands
Asawaree Kalavade
Alan Kamas
Allen Lao
Edward A. Lee
Seungjun Lee
David G. Messerschmitt
Praveen Murthy
Thomas M. Parks
José Luis Pino
S. Sriram
Michael C. Williamson
Kennard White
This highly multidisciplinary project addresses system-level design and implementation of reactive and real-time systems.
Implementation Technology

The design philosophy in Ptolemy is heterogeneous, allowing for effective use of specialized design tools within a general system-level design environment.
Multiple models of computation may be used in the same system. Here, dataflow is used for signal processing, while a timed discrete-event system models a communication network.
Synchronous Dataflow (SDF)

Possible Schedules
ABCCCD
ACB(C)CD
AB(2C)D

Processor 1

C

\[
q(V) = \begin{bmatrix}
2 & 0 & -1 & 0 \\
1 & -1 & 0 & 0 \\
0 & 0 & 1 & -2 \\
0 & 1 & 0 & -1 \\
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
\end{bmatrix}
\]

\[
q(V) = \begin{bmatrix}
q(A) \\
q(B) \\
q(C) \\
q(D) \\
\end{bmatrix} = \begin{bmatrix}
1 \\
1 \\
2 \\
1 \\
\end{bmatrix}
\]
Hierarchical Scheduling Framework
Software Synthesis for Multiple Embedded Processors
SDF Multiprocessor Scheduling

Past SDF multiprocessor scheduling techniques begin with precedence DAG

Exponential Translation

NP-complete, use Polynomial Heuristics
Exponential DAG Expansion Problem

Precedence DAG

$O(M^{N-1})$ Nodes

N levels
Chains with Mutually Prime Numbers

\[ \sum_{j=1}^{N} \left( \prod_{i=1}^{j-1} n_i \right) \left( \prod_{i=j}^{N-1} m_i \right) \]

Precedence DAG Nodes

CD to DAT conversion (44.1 kHz to 48 kHz)
Hierarchical Scheduling Framework

Initialization
Simple clustering

Main Loop
Apply clustering heuristics

Wrapup
Invoke independent schedulers

Multiprocessor Scheduler
Uniprocessor Scheduler 1
Uniprocessor Scheduler 2
Uniprocessor Scheduler 3
Hierarchical Scheduling — Clustering

Control growth of nodes by clustering SDF graph

• Simple Heuristics
  — User specified
  — Resource constrained
  — Well-ordered uniform repetition count (URC)

• Interprocessor communication cost (IPC) reduction heuristic
Hierarchical Scheduling — Schedulers

Make use of heterogeneous schedulers:

- Uniprocessor schedulers
- Multiprocessor schedulers
- Specialized schedulers
Clustering an SDF Graph

SDF lacks the **composition** property. Thus a cluster of SDF dataflow nodes may change the graph semantics.
The SDF Composition Theorem

Four sufficient conditions to compose two adjacent SDF nodes into an SDF cluster without changing the semantics of the overall graph eliminate the possibility of:

1. Introducing a cycle into the SDF graph
2. Hiding a delay in a cycle
3. Improperly shifting the precedences (1)
4. Improperly shifting the precedences (2)
SDF Composition Theorem
C1: SDF Cycle Introduction

Invalid SDF clustering which introduces SDF cycle

Precedence DAG showing introduced cycle
SDF Composition Theorem
C2: Hidden Delay in an SDF Cycle

Invalid SDF clustering

Precedence DAG showing introduced cycle
SDF Composition Theorem

C3 & C4: Precedence Shift Conditions

Invalid SDF clustering

Precedence DAG showing introduced cycle
Well-Ordered URC SDF Clustering

N Node URC Dataflow Specification

1 Precedence DAG Node
DAG Multiprocessor Scheduling

Common technique for reducing IPC

1. Clustering phase – Reduce the parallel time

2. Processor assignment phase
Hierarchical Scheduling Main Loop

1. Apply one step of modified Sarkar’s multiprocessor clustering heuristic on the acyclic SDF graph

2. Test cluster to verify it does not introduce deadlock, and update the repetitions vector and reachability matrix

3. Repeat 1,2 until we reach stopping condition

$$\sum_{v_i' \in V} q_{G'}(v_i') < K_{\text{max}}(|V|, P)$$
# Performance

<table>
<thead>
<tr>
<th>System</th>
<th>SDF Graph Size (nodes)</th>
<th>Precedence DAG Size (nodes)</th>
<th>Scheduling Time (seconds)</th>
<th>Parallel Time (clock cycles)</th>
<th>P1: DSP Code Size Assembly (bytes)</th>
<th>P2: Sparc Code Size C (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FM-Synthesis 128 pt. spectrum</td>
<td>44</td>
<td>14 / 806 57 x smaller</td>
<td>0.47 / 4.35 9.25 x faster</td>
<td>28832 / 28832 no difference</td>
<td>408 / 408 same</td>
<td>34K / 420K 12 x smaller</td>
</tr>
<tr>
<td>BPSK (530 bps)</td>
<td>31</td>
<td>9 / 2628 292 x smaller</td>
<td>0.37 / 14.71 40 x faster</td>
<td>41566 / 41368 &lt; 1% difference</td>
<td>424 / 32045 75 x smaller</td>
<td>14K / 56K 4 x smaller</td>
</tr>
<tr>
<td>4-QAM (320 bps) eye diagram</td>
<td>59</td>
<td>15 / 9267 618 x smaller</td>
<td>0.91 / 80.87 87 x faster</td>
<td>150123 / 150123 no difference</td>
<td>1421 / 87533 62 x smaller</td>
<td>38K / 63K 1.7 x smaller</td>
</tr>
<tr>
<td>4-QAM (640 bps)</td>
<td>52</td>
<td>10 / 3490 349 x smaller</td>
<td>0.69 / 20.1 29 x faster</td>
<td>40037 / 39707 &lt; 1% difference</td>
<td>848 / 29720 35 x smaller</td>
<td>35K / 56K 1.6 x smaller</td>
</tr>
</tbody>
</table>
Conclusions

• Presented Ptolemy a heterogeneous system-level design environment for reactive and real-time systems

• Developed a hierarchical scheduling framework which uses heterogeneous schedulers to synthesize an application from a high-level dataflow description

• Improved scheduling time and resource usage by 1 - 2 orders of magnitude using user specified clustering

• Enabled application designers to synthesize applications previously not possible