A Meta-modeling Framework for Dynamic Reconfiguration of Dataflow Graphs

Sankalita Saha, Dong-Ik Ko and Shuvra S. Bhattacharyya

Department of Electrical and Computer Engineering, and Institute for Advanced Computer Studies, University of Maryland, College Park MD, USA
Outline

• Introduction
• Background
• Dataflow Graphs
• Synchronous Dataflow
• Parameterized Dataflow
• Blocked Dataflow
• Dynamic Graph Topology
• Conclusions
Introduction: Dataflow

- Used widely in design tools for DSP.
- Application is modeled as a directed graph.
Introduction: Dataflow

• Data-driven execution model
  – A node can execute whenever it has sufficient data on its input edges.
  – The order in which nodes execute is not part of the specification.
  – The order is typically determined by the compiler, the hardware, or both.

• Iterative execution
  – Body of loop to be iterated a large or infinite number of times.
Background

- **Petri nets**
  - Depicts structure of a distributed system as a directed bipartite graph consisting of places and transitions. Arcs run between places and transitions.

- **Marked Graphs**
  - Special case of a Petri net where every place has exactly one incoming arc and one outgoing arc.

- **Computation Graphs**
  - A finite graph with a set of nodes, each associated with a function and a set of arcs. A branch is a queue of data directed from one node to another.
  - A function can be fired only when the number of tokens present on the arc exceeds a given threshold.

- **Kahn Process Networks**
  - Consists of nodes representing arbitrary sequential programs communicating via channels of the process networks with blocking read and non-blocking write operations.
Synchronous Dataflow (SDF)

- The number of tokens produced and consumed is restricted to be a positive integer known at compile time [1].
Dataflow: Data-triggered or Event-triggered?

- Dataflow graphs essentially represent data-triggered system.
Dataflow: Data-triggered or Event-triggered?

- Dataflow can model event triggered systems too
Dataflow: Pros and Cons

• Advantages
  – Static scheduling and compile-time predictability for useful restricted forms of dataflow
  – Exposes opportunities for coarse-grain optimizations
• Disadvantages
  – Limited expressive power of the static (restricted forms)
  – Lack of intuitive appeal for the dynamic (Turing complete) forms
Parameterized Dataflow

• (PDF) is a meta-modeling framework that can be applied to any underlying form of dataflow graph ("base model") that has a well-defined notion of graph iteration [2].
• Increases expressive power while allowing many static analysis techniques, and keeping much of the intuitive structure of the base model.
Parameterized SDF (PSDF)

- Parameterized SDF is obtained by parameterizing arbitrary aspects of an SDF graph (actors, edges, dataflow properties) and allowing parameters to be dynamically changed according to a structured hierarchical discipline [2].
  - *init* and *subinit* graphs are dataflow graphs that are responsible for reconfiguring parameters associated with a subsystem.
  - *init* executes less frequently but can perform less restricted reconfiguration operations compared to *subinit*.
- An actor is characterized by a set of parameters that can control the actor’s functionality, as well as its dataflow behavior.
Parameterized Cyclo-static SDF (PCSDF)

- Parameterized CSDF is obtained by parameterizing cyclo-static SDF.
- There are 2 fundamental dataflow properties which can be parameterized
  - period of the cycle of phases
  - data rates associated with each phase
Blocked Dataflow

- Extends parameterized dataflow: the set of parameters used to reconfigure sub-systems or sub-graphs are extracted from the input data stream [3]
Dynamic Graph Topology

- Extends PSDF to allow dynamic change of graph topology at run-time [4].
  - The *init* graph determines the transfer rate of each port of a graph.
  - The *subinit* graph determines the graph topology of the associated body graph before the invocation of the body graph.
Parameterized Dataflow and Event-triggered Systems
Experiments (DGT)

- An MPEG2 video encoder was implemented.
- Target platform was Texas Instruments C64xx series using Code Composer Studio.
- Two implementations were done
  - Separate graph approach using a combination of SDF and FSM.
  - DGT implementation.
- SAS (Single Appearance Schedule) and MAS (Multiple Appearance Schedule) and a combination of SAS and MAS was used.
- The DGT method selects different scheduling methods (SAS or MAS) depending on graph characteristics.
Experimental Results (DGT)

<table>
<thead>
<tr>
<th>Frame Size</th>
<th>&lt; DGT Graph &gt;</th>
<th></th>
<th></th>
<th>&lt; Separate Graph &gt;</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>26,469</td>
<td>31,946</td>
<td>26,469</td>
<td>63,341</td>
<td>79,773</td>
<td>63,341</td>
</tr>
<tr>
<td>Buffer</td>
<td>1,557</td>
<td>1,429</td>
<td>1,557</td>
<td>4,667</td>
<td>4,283</td>
<td>4,667</td>
</tr>
<tr>
<td>Total</td>
<td>28,026</td>
<td>33,375</td>
<td>28,026</td>
<td>68,008</td>
<td>84,056</td>
<td>68,008</td>
</tr>
<tr>
<td>Code</td>
<td>26,469</td>
<td>31,946</td>
<td>26,469</td>
<td>63,341</td>
<td>79,773</td>
<td>63,341</td>
</tr>
<tr>
<td>Buffer</td>
<td>6,173</td>
<td>5,661</td>
<td>6,173</td>
<td>18,515</td>
<td>16,979</td>
<td>18,515</td>
</tr>
<tr>
<td>Total</td>
<td>32,642</td>
<td>37,607</td>
<td>32,642</td>
<td>81,856</td>
<td>96,752</td>
<td>81,856</td>
</tr>
<tr>
<td>Code</td>
<td>26,469</td>
<td>44,903</td>
<td>31,393</td>
<td>63,341</td>
<td>118,645</td>
<td>94,180</td>
</tr>
<tr>
<td>Buffer</td>
<td>52,852</td>
<td>19,991</td>
<td>21,788</td>
<td>158,551</td>
<td>59,967</td>
<td>65,364</td>
</tr>
<tr>
<td>Total</td>
<td>79,321</td>
<td>64,894</td>
<td>53,181</td>
<td>221,892</td>
<td>178,612</td>
<td>159,544</td>
</tr>
<tr>
<td>Code</td>
<td>26,469</td>
<td>58,074</td>
<td>44,564</td>
<td>63,341</td>
<td>158,157</td>
<td>133,692</td>
</tr>
<tr>
<td>Buffer</td>
<td>130,680</td>
<td>45,320</td>
<td>49,397</td>
<td>392,035</td>
<td>135,955</td>
<td>148,192</td>
</tr>
<tr>
<td>Total</td>
<td>157,149</td>
<td>103,394</td>
<td>93,961</td>
<td>455,376</td>
<td>294,112</td>
<td>281,884</td>
</tr>
<tr>
<td>Code</td>
<td>26,469</td>
<td>58,074</td>
<td>50,041</td>
<td>63,341</td>
<td>158,157</td>
<td>150,124</td>
</tr>
<tr>
<td>Buffer</td>
<td>1,817,064</td>
<td>100,940</td>
<td>100,937</td>
<td>5,451,187</td>
<td>302,815</td>
<td>302,524</td>
</tr>
<tr>
<td>Total</td>
<td>1,843,533</td>
<td>159,014</td>
<td>150,978</td>
<td>5,514,528</td>
<td>460,972</td>
<td>452,648</td>
</tr>
</tbody>
</table>
Conclusions

• Parameterized dataflow is powerful modeling paradigm for signal processing applications.
• Various forms of dynamic reconfigurability can be achieved using parameterized dataflow and associated models of computation.
• Though traditional view of dataflow based systems is as data-triggered systems, they can be used to model event-triggered systems as well.
Thank you!
References


