SYSTEM LEVEL MODELLING AND PERFORMANCE ESTIMATION

EXPLORATION OF A DIGITAL AUDIO PLATFORM

ANDERS SEJER TRANBERG-HANSEN AND JAN MADSEN

DTU Informatics
Department of Informatics and Mathematical Modeling
OUTLINE

MOTIVATION
MOTIVATION
MOTIVATION

• Increased design complexity is leading to sub-optimal implementations due to time-to-market constraints of systems.

• Difficult to get feedback to the consequences of a design choice before the system is realized – implying that in the early design phases experience of designers are a key element.

• Efficiency goals dominate leading to systems with less flexibility due to insufficient exploration of the design space.
PRESENTATION OUTLINE

• Overview of the Framework.
• Service Models.
  – Architecture Modelling
• Case-study: A Mobile Audio Processing Platform.
• Conclusion and Future Work.
THE FRAMEWORK

A COMPOSITIONAL MODELLING FRAMEWORK FOR PERFORMANCE ESTIMATION
OBJECTIVES

• A framework for quantitative performance estimation for use in all design phases.

• The framework is a tool which can be used for exploration of the design space.

• Must be able to capture both applications and the platform onto which the application is executed.
OVERVIEW
BASICS

SERVICE MODELS
SERVICE MODEL

• The behaviour of a component is modelled by the availability of a set of services.

• A Service Model is composed of:
  1. One or more Service Model Interfaces
  2. One Service Model Implementation
SERVICE MODEL

• A Service Model defines:
  1. The services offered.
  2. The implementation of the behaviour of the services.
  3. The resource requirements of the services.
  4. The inter-connection possibilities of the model.
  5. The latency of the services.
SERVICE MODEL COMPOSITION

PROCESSOR SERVICE MODEL

ACTIVE

PASSIVE

BUS SERVICE MODEL

PROCESSOR SERVICE MODEL

ALU SERVICE MODEL

CACHE SERVICE MODEL
SERVICE MODEL COMPOSITION

SERIAL CONNECTION

PARALLEL CONNECTION
THE FRAMEWORK

APPLICATION MODEL
PRODUCER{
    ACTIVE_INTERFACE write;
    RUN()
    for(int i=0; i < 63; i++){
        IntVal res;
        // CALCULATE RESULT
        ...
        write.request("WT", res);
    }
}

CONSUMER{
    ACTIVE_INTERFACE read;
    RUN()
    for(int i=0; i < 63; i++){
        IntVal res :=
        read.request("RD");
        // PROCESS DATA
        ...
    }
}

APPLICATION MODEL
THE FRAMEWORK

PLATFORM MODEL
PLATFORM MODEL

PASSIVE INTERFACE

PASSIVE INTERFACE

PROCESSING ELEMENT

ACTIVE INTERFACE

MEMORY

PASSIVE INTERFACE

PASSIVE INTERFACE

PROCESSING ELEMENT

ACTIVE INTERFACE

PASSIVE INTERFACE
MODEL-OF-COMPUTATION

• Hierarchical Coloured Petri Nets.
  – Hierarchical.
  – Parallel activities.
  – Resource requirements and access to shared resources.
  – Data flow.
• Modified Execution Semantics.
• Quasi-static scheduling
• Two types of tokens only.
HCPN BASED SERVICE MODEL

S = Service
A = ArgumentList
R = RequestNumber

**SERVICE MODEL**

**IMPLEMENTATION**

**PASSIVE INTERFACE IN**

**ACTIVE INTERFACE IN**

**PASSIVE INTERFACE OUT**

**ACTIVE INTERFACE OUT**

**SERVICE REQUESTS**

**OFFERED SERVICES**

**REQUIRED SERVICES**

**SERVICE MODEL IMPLEMENTATION**

**SERVICE REQUESTS**
THE FRAMEWORK

SYSTEM MODEL
SYSTEM MODEL

```
PRODUCER()
  ACTIVE_INTERFACE sw;
  RUN()
  for(int i=0; i < 63; i++){
    IntVal res;
    // CALCULATE RESULT
    sw-request(...);
    ...
    sw-request("WT", sX, res);
  }
}
```

```
CONSUMER()
  ACTIVE_INTERFACE sw;
  RUN()
  for(int i=0; i < 63; i++){
    IntVal res;
    // CALCULATE RESULT
    sw.request(...);
    ...
    sw.request("WT", sX, res);
  }
}
```

```
BUFFER()
  PASSIVE_INTERFACE sw;
  RUN()
  for(int i=0; i < 63; i++){
    IntVal res;
    // CALCULATE RESULT
    sw.request("WT", sX, res);
  }
}
```

```
PROCESSING ELEMENT
  PASSIVE_INTERFACE sw;
  RUN()
  for(int i=0; i < 63; i++){
    IntVal res;
    // CALCULATE RESULT
    sw.request("WT", sX, res);
  }
}
```

DTU Informatics, Technical University of Denmark
PRODUCER{
  ACTIVE_INTERFACE sw;
  RUN()
  {
    for(int i=0; i < 63; i++){
      IntVal res;
      // CALCULATE RESULT
      sw.request(...);
      ...
      sw.request("WT", res);
    }
  }
}

PRODUCER{
  ACTIVE_INTERFACE sw;
  RUN()
  {
    for(int i=0; i < 63; i++){
      IntVal res :=
      sw.request("RD", 0xA);
      // PROCESS DATA
      sw.request(...);
      ...
    }
  }
}
CASE-STUDY

A MOBILE AUDIO PROCESSING PLATFORM
Bang & Olufsen ICEpower

Diagram:
- Energy Source (PE) to Power Management (PM)
- Audio Source to Signal Processing (SP)
- Amplifier (AMP) to Transducer (TR)
- Transducer (TR) to Acoustic Power (AP)

DTU Informatics, Technical University of Denmark
MOBILE AUDIO PROCESSING
PLATFORMS

• Available processing elements:
  – Dedicated hardware implementations (HW)
  – Application Specific Instruction-set Processor (SVF)
  – Digital Signal Processor optimized for Audio (DSP)

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>RELATIVE AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW (REFERENCE)</td>
<td>Dedicated hardware, ICEpower impl.</td>
<td>1</td>
</tr>
<tr>
<td>SVFx1</td>
<td>1 SVF ASIP.</td>
<td>0.6</td>
</tr>
<tr>
<td>SVFx2</td>
<td>2 SVF ASIP’s.</td>
<td>1.2</td>
</tr>
<tr>
<td>SVFx3</td>
<td>3 SVF ASIP’s.</td>
<td>1.8</td>
</tr>
<tr>
<td>SVFx4</td>
<td>4 SVF ASIP’s.</td>
<td>2.4</td>
</tr>
<tr>
<td>DSP</td>
<td>1 Audio DSP.</td>
<td>4</td>
</tr>
</tbody>
</table>
UTILIZATION OF PLATFORMS
ACCURACY

<table>
<thead>
<tr>
<th></th>
<th>MIXED LATENCY</th>
<th>COMPILED</th>
<th>RTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>84,034</td>
<td>102,034</td>
<td>102,034</td>
</tr>
<tr>
<td>B</td>
<td>123,041</td>
<td>129,384</td>
<td>129,384</td>
</tr>
<tr>
<td>C</td>
<td>33,011</td>
<td>33,011</td>
<td>33,011</td>
</tr>
<tr>
<td>D</td>
<td>153,051</td>
<td>168,056</td>
<td>168,056</td>
</tr>
<tr>
<td>TOTAL</td>
<td>393,137</td>
<td>429,143</td>
<td>429,143</td>
</tr>
</tbody>
</table>
OUTLOOK

CONCLUSION & FUTURE WORK
CONCLUSION

• Early performance estimation.
• Flexible construction of models.
• High-level of reusability, using component libraries.
• Support for multiple levels of abstraction to co-exists within a model.
• Excellent refinement possibilities of models.
• High level models to cycle accurate instruction set simulators.
FUTURE WORK

• SystemC / TLM 2.0 integration investigation.
• Better tool support.
  – Specification of models.
  – Specification of applications.
  – “Service Request Compiler”
• In depth case studies – larger and more complex models.
• Investigation of formal analysis possibilities using traditional HCPN methods.