MPSoC Design Flow: Case Study for H.264†

Kai Huang
huangk@vlsi.zju.edu.cn

Institute of VLSI Design, Zhejiang University, China
August, ICDFN 2007

† "Simulink-Based MPSoC Design Flow: Case Study of MJPEG and H.264" Published at DAC 2007
Motivation

System level design: Key solution to complex MPSoC

- Algorithm/Architecture mapping (System Level)
- Protocol Accurate (Virtual Architecture Level)
- Synchronization Accurate (Transaction Level)
- Cycle Accurate (Virtual Prototype Level)

- SystemC
  - A preferred HW/SW codesign language
    - Within a wide range of abstraction levels
  - Intrinsic low level language
    - NOT easy to specify the complex system at algorithm level

- Simulink
  - The prevailing environment
    - Modeling and simulating complex systems at algorithm level
    - An open issue: Algorithm/Architecture mapping for MPSoC
Objective

1) System level MPSoC design flow
   - Combine Simulink with SystemC
     - Simulink for high-level algorithm modeling
     - SystemC for low-level HW/SW design
   - Concurrent HW/SW design
     - Seamless refinement at different level abstraction models
     - Systematic and automated HW/SW code generation

2) Case study for multimedia applications
   - Feasibility and efficiency of proposed design flow
     - System functional validation
     - HW/SW co-design/co-verification
     - Performance analysis for architecture exploration
Content

- Motivation & Objective
- Simulink-Based MPSoC Design Flow
  - HW and SW Mixed Model
  - Design Steps
- Case Study
  - H.264 Baseline Decoder
- Experiment
- Conclusions & Future Works
Overall Design Flow

1. Simulink modeling
2. Application/Architecture mapping
3. Simulink parsing
4. HW Architecture Gen.

Mixed HW and SW Model

Step i
HW and SW Mixed Models

- Seamless refinement at four abstraction levels:
  - Simulink Combined Algorithm and Architecture Model (Simulink CAAM)

![Diagram showing HW and SW Mixed Models]

- System level model (Simulink CAAM)
- Virtual architecture Model (SystemC)
- Transaction accurate (SystemC)
- Virtual prototype (SystemC)
Step 1 : Simulink Modeling

- **Application C/C++** Into a set of modular functions:
  - *User-defined Simulink blocks* (e.g. S-function)
  - pre-defined Simulink blocks (e.g. mathematical operation)

Application C++ Codes
1: ...
2: ...
3: for (i=0; i<99; i++)
4: { ...
5: ...
6: ...
7: ...
8: F_{11}(\cdots, \&\text{Out10});
9: ...
10: }
11: ...
12: ...
13: ...

Simulink Modeling
Step 2 : Application/Architecture mapping

1) Architecture layer:
   CPU SSs, Inter-SS comm.

2) Subsystem layer:
   Threads, Intra-SS comm.

3) Thread layer:
   Simulink blocks, links
Step 3 : Simulink Parsing

Colif is a XML-based meta-model
- well-defined data structures
- Modules, channels and ports
- One-to-one correspondence (Simulink)
- Simulink Port to Send/Receive Block
Step 4: HW Architecture Generation

1. Component inst.
2. Interconnect inst.

CPU SS generation

System model

HW Library

VP HW platform

CPU1 (ARM7) (4MB~8MB)

ARM7 ISS
Mem
Timer
Bus bridge
Mailbox

CPU2 (Xtensa) (8MB~12MB)

Xtensa ISS
Mem
Timer
Bus bridge
Mailbox

CPU3 (Xtensa) (12MB~16MB)

Xtensa ISS
Mem
Timer
Bus bridge
Mailbox

Global shared bus
Step 5: Multithreaded Code Generation

1. Copy removal
2. Scheduling
3. Buffer sharing
4. Code generation

Step 4: HW Library
   - Comp. Subsystems
   - Comm. Channels

Step 3: Simulink CAAM

Step 2: Simulink parsing

Step 1: Application
   - Simulink modeling
   - Simulink application model

Application/Architecture mapping

Virtual Architecture Model

Transaction Accurate Model

Virtual Prototype Model

Multithread Code Gen.

Copy Removal and Buffer sharing are used for memory optimization
Content

- Motivation & Objective
- Simulink-Based MPSoC Design Flow
  - HW and SW Mixed Models
  - Design Steps
- Case Study
  - H.264 Baseline Decoder
- Experiment
- Conclusions & Future Works
H.264 Baseline Decoder

- Receives an encoded video bit stream and performs iterative executions of Macroblock level functions:
  - **VLD**: Variable Length Decoder
  - **IQ**: Inverse Quantization
  - **IT**: Inverse Transform
  - **SC**: Spatial Compensation
  - **MC**: Motion Compensation
  - **REC**: Reconstruction
  - **DF**: Deblock Filter

- **VLD**: Variable Length Decoder
  - **16x16**
  - Global ctrl
  - Macroblock VLD
  - Luminance VLD
  - **16x16**

- **8x8**
  - Chroma U VLD

- **16x16**

- **MC/SC**

- **Chroma Decoding**

- **Luma Decoding**

- **Chroma U**

- **Chroma V**

- **Luminance**
A Simulink CAAM Example

4 CPU SS
4 Threads
Inter-SS COM: GFIFO
Processor: ARM7
Experiment: Simulation Time

- An experiment for decoding 10 frames QCIF FOREMAN:
  - Four ARM7 Processors (VP)
  - GFIFO (TA, VP)
  - RTW (A sequential program running on host machine)

<table>
<thead>
<tr>
<th>RTW</th>
<th>Simulink</th>
<th>VA</th>
<th>TA</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5s</td>
<td>20.0s</td>
<td>2.8s</td>
<td>325s</td>
<td>11066s</td>
</tr>
<tr>
<td>146M/s</td>
<td>3.6M/s</td>
<td>26.0M/s</td>
<td>224K/s</td>
<td>6.5K/s</td>
</tr>
</tbody>
</table>

- VP is too long to debug the whole system.
- It’s necessary to make use of TA for HW/SW co-simulation.
Experiment: Performance Optimization with Different Architectures

A popular and simple task partition strategy was used in this experiment

- ✓ Computation-based
- ✓ Step by Step

Low computation load
High computation load
F1 Function Block
## Experiment Result of H.264 Baseline Decoder with Three Architectures

<table>
<thead>
<tr>
<th>CPU</th>
<th>2ARM</th>
<th>4ARM</th>
<th>6ARM</th>
<th>Function Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>CPU1</td>
<td>CPU1</td>
<td></td>
<td>Global control and MB VLD</td>
</tr>
<tr>
<td>CPU2</td>
<td>CPU2</td>
<td></td>
<td></td>
<td>Chroma decoding</td>
</tr>
<tr>
<td>CPU3</td>
<td>CPU3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU4</td>
<td>CPU4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Tradeoff between performance, cost and flexibility:
- High-performance dedicated processor?
- Fine-granularity task partition?

### Graphs

#### (a) Execution Time
- **Total execution cycles**
- **CPU1**
- **CPU2**
- **CPU3**
- **CPU4**
- **CPU5**
- **CPU6**

#### (b) H264_GFIFO_Computation Load
- **CPU1**
- **CPU2**
- **CPU3**
- **CPU4**

#### (c) H264_GFIFO_Computation Load
- **CPU1**
- **CPU2**
- **CPU3**
- **CPU4**

#### (d) H264_GFIFO_Computation Load
- **CPU1**
- **CPU2**
- **CPU3**
- **CPU4**

---

**fine-grained task partition:**
- more communication cost,
- more idle time for data sync.
Content

- Motivation & Objective
- Simulink-Based MPSoC Design Flow
  - HW and SW Mixed Models
  - Design Steps
- Case Study
  - H.264 Baseline Decoder
  - Experiment
- Conclusions & Future Works
Conclusions & Future Works

- Proposed a Simulink based MPSoC design flow
  - For automated concurrent hardware-software design and verification.
  - Refine Simulink CAAM to three different abstraction level models (VA, TA, VP)
- In the case study of H.264 decoder
  - The feasibility and efficiency of proposed design flow
    - Functional evaluation
    - HW/SW codesign
    - Performance analysis for architecture exploration
- Plan to improve the current design flow:
  - Dedicated instruction sets.
  - Communication protocol with DMA.
  - Automatic design space exploration.
Acknowledgment

- **Thanks to:**
  - Prof. Ahmed Jerraya  
    (CEA-LETI, MINATEC, France)
  - Sang-II Han  
    (Seoul National University, Korea)
  - Katalin Popovici  
    (TIMA Laboratory, France)
  - Lisane Brisolara  
    (Federal University of Rio Grande do Sul, Brazil)
Thank you very much for your attention!
APPENDIX
Simulink CAAM of M-JPEG Decoder

- Four threads and three CPUs
- ARM processor and GFIFO/SWFIFO

Architecture Layer

Subsystem Layer

Thread Layer

7 S-Functions
7 Delays
26 Links
4 IASs
# Inter/Intra-Subsystem Communication

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Data buffer</th>
<th>Sender sync. addr</th>
<th>Receiver sync. addr</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWFIFO</td>
<td>Local memory</td>
<td>@ local memory</td>
<td>@ local memory</td>
<td>Software FIFO</td>
</tr>
<tr>
<td>SHM</td>
<td>Local memory</td>
<td>@ local memory</td>
<td>@ local memory</td>
<td>Shared memory</td>
</tr>
<tr>
<td>HWFIFO</td>
<td>Hardware queue</td>
<td>@ hardware FIFO</td>
<td>@ hardware FIFO</td>
<td>Hardware FIFO</td>
</tr>
<tr>
<td>GFIFO</td>
<td>Shared memory</td>
<td>@ mailbox</td>
<td>@ mailbox</td>
<td>Software FIFO via global shared memory</td>
</tr>
<tr>
<td>LFIFO</td>
<td>Local memories</td>
<td>@ mailbox</td>
<td>@ mailbox</td>
<td>Software FIFO via local memories</td>
</tr>
<tr>
<td>DMS</td>
<td>Local memories</td>
<td>@ MSAP</td>
<td>@ MSAP</td>
<td>Distributed memory server with DMA</td>
</tr>
</tbody>
</table>
Code and Data memory size of H.264 decoder

<table>
<thead>
<tr>
<th></th>
<th>Code memory size (Kbyte)</th>
<th>Data memory size (Kbyte)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>App. library</td>
<td>Hds library</td>
</tr>
<tr>
<td></td>
<td>Buffer</td>
<td>Channel</td>
</tr>
<tr>
<td>RTW</td>
<td>100 (79.0K)</td>
<td>100 (27.0K)</td>
</tr>
<tr>
<td>S1</td>
<td>97.7 (97.2K)</td>
<td>98.8 (26.6K)</td>
</tr>
<tr>
<td>S2</td>
<td>79.0 (62.4K)</td>
<td>58.1 (15.7K)</td>
</tr>
<tr>
<td>S3</td>
<td>78.7 (62.1K)</td>
<td>29.1 (7.9K)</td>
</tr>
<tr>
<td>M1</td>
<td>105.3 (83.2K)</td>
<td>29.1 (7.9K)</td>
</tr>
<tr>
<td>M2</td>
<td>105.9 (83.6K)</td>
<td>74.4 (20.0K)</td>
</tr>
<tr>
<td>M3</td>
<td>125.9 (99.5K)</td>
<td>110.3 (29.7K)</td>
</tr>
</tbody>
</table>

S1   Single-thread without optimization options
S2   Single-thread with **copy removal**.
S3   Single-thread with **copy removal** and **buffer sharing**
M1   Multi-thread without optimization options
M2   Multi-thread with **copy removal**.
M3   Multi-thread with **copy removal** and **buffer sharing**