Automatic Design Space Exploration and Prototyping from Behavioral Models

Joachim Falk, Jens Gladigau, Michael Glaß, Martin Lukasiewycz, Joachim Keinert, Felix Reimann, Martin Streubühr, Christian Haubelt, Jürgen Teich

Hardware/Software Co-Design
University of Erlangen-Nuremberg
JPEG Decoder Application

010010001001010111100101101010111111111111000000110000
1111110000011000011010101111111000000000000011110100100
0010101111001011010101111111111100000011000010010001
11111100000110000110101010010000000001111010010001

Some Questions:
- Run everything in software? How many processor cores?
- Design a hardware accelerator for this application?
- Compute only parts in hardware?
MPSoC Platform

- SDRAM
- MemCtrl
- \( \mu \text{Controller} \)
- \( \mu \text{Controller} \)
- Local RAM

Controller Bus

- Bridge
- DSP
- DSP RAM
- HW Accl

DSP Bus

- Shared RAM
- HW Accl
- Video Front End

Local Bus
Mapping

Application

Automatic Mapping

Platform
Agenda

- Overview
- Design Space Exploration
- Embedded Software Generation
- Prototyping
- Case study
- Conclusions
System Synthesis

System synthesis comprises:

- Resource allocation
- Actor binding
- Channel mapping
- Transaction routing

Idea:

- Formulate system synthesis problem as 0-1 ILP
- Use Pseudo-Boolean (PB) solver to find a feasible solution
- Use Multi-Objective Evolutionary Algorithm (MOEA) to optimize Decision Strategy of the PB solver
System Synthesis (Resource Allocation)

- $R$ denotes the set of resources
- Resource activation $\alpha: R \rightarrow \{0, 1\}$
- $\alpha(r) = 1$ denotes resource activation
- $\alpha(r) = 0$ denotes resource deactivation
System Synthesis (Actor Binding)

- $A$ denotes the set of actors
- Actor binding activation $\alpha : A \times R \rightarrow \{0, 1\}$
- $\alpha(a,r) = 1$ binds actor $a$ onto resource $r$
- $\forall a \in A : \sum \alpha(a,r) = 1$ (Each actor is bound exactly once)
System Synthesis (Channel Mapping)

- \( C \) denotes the set of channels
- Channel mapping activation \( \alpha : C \times R \rightarrow \{0, 1\} \)
- \( \alpha(c,r) = 1 \) maps channel \( c \) onto resource \( r \)
- \( \forall c \in C: \sum \alpha(c,r) = 1 \) (Each channel is mapped exactly once)
Additional variables for transactions (memory accesses)
\[ t_{a,c,r,n}, t_{c,a,r,n} \in \{0, 1\} \]

- \( n \) is the communication step

Set of constraints
- Transaction has to start or end at the resource a sending/receiving actor is bound to, respectively
- Each transaction must not visit a resource more than once
- Successive communication steps must be performed on adjacent resources
Design Space Exploration

PB Solver

Crossover

Mutation

Performance vs. Cost
Agenda

- Overview
- Design Space Exploration
- Embedded Software Generation
- Prototyping
- Case study
- Conclusions
SystemC Advantages

- Permits hardware/software co-design
- Actor-oriented design
  - Network graph
  - Channels
  - Actors
- Actors are only allowed to communicate via channels
SystemC Disadvantages

- Complex control flow and unstructured channel accesses prevent proper analysis leading often to suboptimal application implementations

```cpp
class PPMSink: public sc_module {
  void process() {
    while(1) {
      dimX = i2.read();
      dimY = i2.read();
      pixels = dimX*dimY;
      printHeader();
      for (int n=0; n<pixels; n++)
        printPixel(i1.read());
    }
  }
};
```
Single Processor Scheduling

\[
\text{schedule()} \{ \\
\quad \text{while(true)} \{ \\
\quad \quad \text{forall actors } \mathbf{a} \in \mathbf{A} \{ \\
\quad \quad \quad \text{if is_executable}(\mathbf{a}) \{ \\
\quad \quad \quad \quad \text{execute}(\mathbf{a}) \\
\quad \quad \quad \} \\
\quad \quad \} \\
\quad \} \\
\}
\]
SystemCoDesigner Modeling Approach

parser → Huffman decode → inv ZRL → DC decode → inv quant → Inv ZigZag

IDCT 2D → Shuffle → YCbCr → PPM Sink

PPM Sink

Transform:

| size       | pixel = i1[0];
| dimX = i2[0];
| dimY = i2[1];

Check:

if(x<dimX || y<dimY) return(false);
return(true);

Queue (FIFO Semantics)
SysteMoC

- SysteMoC – SystemC library for actor-based design
  - Actor functionality (member functions and variables)
  - Communication behavior (activation pattern)

SysteMoC
- Elementary channels separating functionality and communication
- C++ syntax for specifying actor communication behavior as FSMs
- Scheduler for (dynamic) dataflow MoC domain

Elementary Channels
- Signal, timer, mutex, semaphore, FIFO, etc.

Core Language
- Modules
- Ports
- Processes
- Events
- Interfaces
- Channels

Data-Types
- 4-valued logic types (01zx)
- 4-valued logic vectors
- Bits and bit-vectors
- Arbitrary-precision integers
- Fixed-point numbers
- C++ user-defined types

Event-driven Simulation Kernel

C++ Language Standard
Hierarchy of Streaming MoCs

- BDF and larger: Turing complete
- FunState ≈ RPN

RPN: Reactive Process Network
DDF: Dynamic Dataflow
KPN: Kahn Process Network
BDF: Boolean Dataflow
CSDF: Cyclo-Static Dataflow
SDF: Synchronous Dataflow
HSDF: Homogeneous SDF

[Basten@MoCC2008]
Improvement

JPEG Decoder example
- speedup by a factor of two.

Synthetic benchmarks
- speedup up to a factor of 10.
The Situation

- We got a dataflow graph
  - Some actors, or even the topology, are unknown
  - But a subgraph of SDF actors is known
  - How to find a good quasi-static schedule?
Clustering (SDF)

\[ \tau = (2a_1, 1a_2, 2a_3) \]
Clustering (CSDF)

$$\tau = ((1*a_1, 1*a_3), (1*a_1, 1*a_2, 1*a_3))$$
Clustering 2 (CSDF)

\[ \tau = ((1*a_2),(2*a_1,2*a_3)) \]
Clustering (dynamic dataflow)

\[ \#i_1 \geq 1 \land \#i_2 \geq 1 \land \#o_1 \geq 2 \]

\[ \#i_1 \geq 1 \land \#i_2 \geq 1 \land \#o_2 \geq 2 \]

\[ \#i_1 \geq 2/(a_2) \land \#i_2 \geq 1/(a_1,a_3) \]

\[ \#i_2 \geq 1/(a_1,a_3) \land \#i_1 \geq 2/(a_2) \]
Clustering Results

- Real world example [EMSOFT2008]
  - JPEG QCIF (176x144) decoding
  - Dynamic scheduling obtained 88 ms per frame
  - After clustering of the IDCT and the InvZigZag actor 42 ms per frame were obtained.
  - Whole system speedup by a factor of two.

- Synthetic benchmarks
  - Ten random graphs with 60 nodes and 5000 clusters
  - Maximal speedup found per graph is by a factor of 10.
  - Selection of actors for clustering has a huge impact on the obtained speedup.
Agenda

- Overview
- Design Space Exploration
- Embedded Software Generation
- Prototyping
- Case study
- Conclusions
Actor Synthesis

Hardware Synthesis

Software Synthesis

delay, area, power

DSP
Communication Synthesis

Queue (FIFO Semantics)

Memory

FIFO Control

event handling

commit policy

Memory

Memory

HW/SW Performance Estimation
Design Flow

1. **Model**: Select CPUs, busses, hw accelerators, etc. from component library.
2. **Explore Model**: Specify mapping.
3. **Automatic Design Space Exploration**: Select implementation.
4. **Optimized Solutions**: Use Forte Cynthesizer for behavioral synthesis.
5. **Rapid Prototyping**: Component library for final implementation.
Agenda

- Overview
- Design Space Exploration
- Embedded Software Generation
- Prototyping
- Case study
- Conclusions
Motion JPEG (1/2)

- Baseline profile without subsampling (interleaved/non-interleaved)
- 5650 SysteMoC LoC
  - 4 PDs specification and interface definition
  - 16 PDs module implementation
  - 14 PDs integration and debugging (~ 3 PDs integration)
  - 4 PDs SCD SysteMoC code adaptation
- Any HW/SW implementation on Xilinx Virtex II FPGA (50 MHz)
  - HW only implementation QCIF@65frames/second
## Motion JPEG (2/2)

<table>
<thead>
<tr>
<th># SW Actors</th>
<th>Latency</th>
<th>Throughput</th>
<th>LUTs</th>
<th>FFs</th>
<th>BRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>15.63 ms (12.61 ms)</td>
<td>65.0 fps (81.1 fps)</td>
<td>40 467 (44 878)</td>
<td>14 508 (15 078)</td>
<td>47 (72)</td>
</tr>
<tr>
<td>1</td>
<td>23.49 ms (25.06 ms)</td>
<td>43.0 fps (40.3 fps)</td>
<td>35 033 (41 585)</td>
<td>11 622 (12 393)</td>
<td>72 (96)</td>
</tr>
<tr>
<td>8</td>
<td>6 275 ms (4 465 ms)</td>
<td>0.16 fps (0.22 fps)</td>
<td>15 064 (17 381)</td>
<td>7 540 (8 148)</td>
<td>63 (63)</td>
</tr>
<tr>
<td>all</td>
<td>10 030 ms (8 076 ms)</td>
<td>0.10 fps (0.13 fps)</td>
<td>1 893 (2 213)</td>
<td>1 086 (1 395)</td>
<td>29 (29)</td>
</tr>
</tbody>
</table>

7,600 solutions evaluated
366 non-dominated solutions
evaluation time 30.44s/solution
Agenda

- Overview
- Design Space Exploration
- Embedded Software Generation
- Prototyping
- Case study
- Conclusions
Conclusions

- SystemCoDesigner automatically maps applications to MPSoC platforms
- SystemCoDesigner application model permits an automatic detection of restricted MoC
- Thus, making available design methods applicable
- SystemCoDesigner uses a combination of PB solver and MOEA to perform automatic design space exploration
Project Partners

- Alcatel-Lucent AG
- Audi AG
- Cadence Design Technologies
- Daimler AG
- Forte Design Systems
- Fraunhofer Institute for Integrated Circuits
- IBM Germany, GmbH
- Infineon Technologies AG
- VaST Systems