Perspectives on System-level MPSoC Design Space Exploration

Andy D. Pimentel
System and Network Engineering Lab
University of Amsterdam
Embedded Systems Design

- Design of embedded systems becomes increasingly complex
- Heterogeneous Multi-Processor System-on-Chip architectures
  ✓ Different processor types, dedicated / reconfigurable hardware blocks, Network-on-Chip, etc.
Embedded Systems Design

- Design of embedded systems becomes increasingly complex
- Heterogeneous Multi-Processor System-on-Chip architectures
  - Different processor types, dedicated / reconfigurable hardware blocks, Network-on-Chip, etc.
- Many design requirements
  - High performance, low power, low cost, small form factor, high flexibility, high reliability, etc.
  - Typically conflicting requirements
Our Holy Grail...
Our Holy Grail...
Our Holy Grail...
Climbing the abstraction ladder
Climbing the abstraction ladder

Sequential program code (e.g., C, VHDL)
- Compilers (1960's, 1970's)
- Assembly instructions
- Assemblers, linkers (1950's, 1960's)
- Machine instructions

Behavioral synthesis (1990's)
- Register transfers
- RT synthesis (1980's, 1990's)
- Logic equations / FSM's
- Logic synthesis (1970's, 1980's)
- Logic gates

Implementation
- Microprocessor plus program bits: “software”
- VLSI, ASIC, or PLD implementation: “hardware”

Source: Vahid/Givargis
Climbing the abstraction ladder

Application specification(s) + constraints
System-level synthesis (±2005 - )

Sequential program code (e.g., C, VHDL)

Compilers (1960's, 1970's)
Assembly instructions
Assemblers, linkers (1950's, 1960's)
Machine instructions

Behavioral synthesis (1990's)
Register transfers
RT synthesis (1980's, 1990's)
Logic equations / FSM's
Logic synthesis (1970's, 1980's)
Logic gates

Microprocessor plus program bits: “software”
VLSI, ASIC, or PLD implementation: “hardware”

[IEEE TCAD'09]
Towards System-level Synthesis

Library of IP cores

System-level Synthesis

Sequential application(s)

Multi-processor System on Chip
(Synthesizable VHDL and C/C++ code for processors)
The context: Daedalus
A system-level synthesis framework

System-level synthesis

Library of IP cores
- High-level Models
- RTL-level Models

Common XML Interface

System-level design space exploration
- Platform specification
- Mapping specification
- Parallel application specification

Multi-processor System on Chip
(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS'07, DAC'08]
The context: Daedalus
A system-level synthesis framework

Library of IP cores
- High-level Models
- RTL-level Models

Common XML Interface
- Platform specification
- Mapping specification
- Parallel application specification

Explore, modify, select instances

Sequential application
Automatic Parallelization

System-level design space exploration

System-level synthesis

Multi-processor System on Chip
(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS’07, DAC’08]
The context: Daedalus
A system-level synthesis framework

[CODES+ISSS’07, DAC’08]
The context: Daedalus
A system-level synthesis framework

Library of IP cores
- High-level Models
- RTL-level Models

Common XML Interface
- Platform specification
- Mapping specification
- Parallel application specification

System-level design space exploration
- Explore, modify, select instances

Sequential application
- Automatic Parallelization

Multi-processor System on Chip
(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS’07, DAC’08]
The context: Daedalus
A system-level synthesis framework

- **Library of IP cores**
- **High-level Models**
- **RTL-level Models**

- **System-level design space exploration**
- **Platform specification**
- **Mapping specification**
- **Parallel application specification**

- **Explore, modify, select instances**

- **Sequential application**
- **Automatic Parallelization**

- **Multi-processor System on Chip**
  (Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS’07, DAC’08]
System-level Design
Space Exploration (DSE)

• We need to automatically
  ✓ find the best decomposition of the (parallel) application(s)
  ✓ decide what application task to perform in SW or accelerate using HW
  ✓ choose the number and types of required processing elements in the (heterogeneous) system
  ✓ decide on how to interconnect the processors
  ✓ decide on how to map application tasks onto the selected processors
  ✓ and so on…
System-level Design
Space Exploration (DSE)

• We need to automatically
  ✓ find the best decomposition of the (parallel) application(s)
  ✓ decide what application task to perform in SW or accelerate using HW
  ✓ choose the number and types of required processing elements in the (heterogeneous) system
  ✓ decide on how to interconnect the processors
  ✓ decide on how to map application tasks onto the selected processors
  ✓ and so on…

• while simultaneously optimizing the system for cost, performance, energy consumption, reliability, etc.
System-level Design
Space Exploration (DSE)

• We need to automatically
  ✓ find the best decomposition of the application(s)
  ✓ decide what application task to perform in SW or accelerate using HW
  ✓ choose the number and types of required processing elements in the (heterogeneous) system
  ✓ decide on how to interconnect the processors
  ✓ decide on how to map application tasks onto the selected processors
  ✓ and so on…

• while simultaneously optimizing the system for cost, performance, energy consumption, reliability, etc.

Major challenge: develop DSE techniques that efficiently and effectively handle the vast design space, with sufficient accuracy
System-level Mapping DSE

- Exploring different
  - Resource allocations
    ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)
**System-level Mapping DSE**

- Exploring different
  - Resource allocations
    - ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)
System-level Mapping DSE

- Exploring different
  - Resource allocations
    - Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)
System-level Mapping DSE

- Exploring different
  - Resource allocations
    ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)
System-level Mapping DSE

• Exploring different
  • Resource allocations
    ✓ Number and type of processors, memories, interconnect(s), etc.
  • Application to Resource bindings (spatial binding)
  • Task scheduling (temporal binding)
System-level DSE: two elements

Evaluating a design point

Searching the design space

Source: Teich et al.
System-level DSE: two elements

Source: Teich et al.

Evaluating a design point

Searching the design space
System-level DSE: two elements

Evaluating a design point

Source: Teich et al.

$G_V$  $M$  $G_A$

- problem graph
- mapping set
- architecture graph
System-level DSE: two elements

![Diagram showing system-level DSE with two elements: Specification and Alternative realizations. The diagram illustrates the trade-offs between accuracy/modeling cost and evaluation speed for different model types, including Analytical models, Cycle-approximate TLM models, Cycle-accurate models, and RTL models.]

Source: Teich et al. [IEEE Computer’01]
System-level DSE: two elements

System-level

Specification

Analytical models
Cycle-approximate TLM models
Cycle-accurate models
RTL models

Alternative realizations

Accuracy/Modeling cost
Low
High

Evaluation speed
Low
High

Source: Teich et al.

[IEEE Computer’01]

Evaluating a design point
System-level DSE: two elements

- Exhaustive search usually is not feasible
- Typically, metaheuristics are used to search the design space
  - Only visit a relatively small number of design points
  - Single-objective or multi-objective optimization
  - Do not guarantee finding the global optimum
System-level DSE: two elements

- Exhaustive search usually is not feasible
- Typically, metaheuristics are used to search the design space
  - Only visit a relatively small number of design points
  - Single-objective or multi-objective optimization
  - Do not guarantee finding the global optimum
Evaluating a single design point
The Sesame simulation framework

Application model
(Kahn Process Network)

Mapping model

Cycle approximate,
TLM MPSoC architecture model
The Sesame simulation framework

Process A → Process C → Process D

Process B

Mapping layer
(mapping, scheduling and event refinement)

Processor 1 → Processor 2 → Processor 3

Shared memory

[IEEE TC’06]
The Sesame simulation framework

Process A → Process C → Process D

Process B

Execute(), Read(), and Write() events

(mapping, scheduling, and event refinement)

Processor 1 → Processor 2 → Processor 3

Shared memory

[IEEE TC’06]
The Sesame simulation framework

Mapping layer
(mapping, scheduling and event refinement)

<table>
<thead>
<tr>
<th>Op.</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>750</td>
</tr>
<tr>
<td>Y</td>
<td>150</td>
</tr>
<tr>
<td>Z</td>
<td>1500</td>
</tr>
</tbody>
</table>

[IEEE TC’06]
Exploring the design space
Exploring the design space: GAs

Selection based on fitness

Population

Reproduction

Variation
Exploring the design space: GAs

![Diagram showing the design space of GAs]

- Population
- Fitness evaluation + Selection
- Parents for producing offspring
- Crossover
- Mutation
- New offspring with new genetic material
- Update population
- New offspring
Exploring the design space: GAs

Crossover

Mutation

Parents → Uniform → Children
One-point
Two-point
Parents → Child

Population
Fitness evaluation + Selection
Parents for producing offspring
Crossover
New offspring
New offspring with new genetic material
Update population
Analyzing the DSE process and its results

- Visualization support for three aspects:
  - Help algorithm developers to find the best optimization algorithm for their specific problem
  - Help designers to analyze the DSE results
  - Help decision makers to choose the most preferred solution
Analyzing the DSE process and its results (cont’d)
Analyzing the DSE process and its results (cont’d)
Analyzing the DSE process and its results (cont’d)
Exploiting domain knowledge

• For example, making the search process aware of “mapping symmetries”
  • GA encoding: [0,1,2,3,0,0]

• A “Mapping distance” (δ) metric to maintain diversity and prevent evaluating duplicates
  • δ(a,a) = 0  (equality)
  • δ(a,b) = #transformations needed to achieve equality
  • δ([0,1,2,3,0,0] , [0,1,0,0,2,3] ) = 4
Exploiting domain knowledge

• For example, making the search process aware of “mapping symmetries”
  • GA encoding: [0, 1, 2, 3, 0, 0]

\[ \delta = 1 \]

• A “Mapping distance” (\(\delta\)) metric to maintain diversity and prevent evaluating duplicates
  • \(\delta(a,a) = 0 \) (equality)
  • \(\delta(a,b) = \#\) transformations needed to achieve equality
  • \(\delta([0, 1, 2, 3, 0, 0], [0, 1, 0, 0, 2, 3]) = 4\)
Exploiting domain knowledge

• For example, making the search process aware of “mapping symmetries”
  • GA encoding: [0,1,2,3,0,0]

  \[ \delta = 2 \]

• A “Mapping distance” (\( \delta \)) metric to maintain diversity and prevent evaluating duplicates
  • \( \delta(a,a) = 0 \) (equality)
  • \( \delta(a,b) = \# \text{transformations needed to achieve equality} \)
  • \( \delta([0,1,2,3,0,0], [0,1,0,0,2,3]) = 4 \)
Exploiting domain knowledge

• For example, making the search process aware of “mapping symmetries”
  • GA encoding: [0, 1, 2, 3, 0, 0]

\[ \delta = 3 \]

• A “Mapping distance” (\( \delta \)) metric to maintain diversity and prevent evaluating duplicates
  • \( \delta(a,a) = 0 \) (equality)
  • \( \delta(a,b) = \text{#transformations needed to achieve equality} \)
  • \( \delta([0,1,2,3,0,0] , [0,1,0,0,2,3] ) = 4 \)
Exploiting domain knowledge

• For example, making the search process aware of “mapping symmetries”
  • GA encoding: [0, 1, 2, 3, 0, 0]
    \[ \delta = 4 \]

• A “Mapping distance” (\( \delta \)) metric to maintain diversity and prevent evaluating duplicates
  • \( \delta(a,a) = 0 \) (equality)
  • \( \delta(a,b) = \# \text{transformations needed to achieve equality} \)
  • \( \delta([0,1,2,3,0,0] , [0,1,0,0,2,3] ) = 4 \)
A small example

- 11-process application, 4-processor crossbar architecture
- Design space: $4^{11} = 4M$ design points (175275 unique)
- Summary of results of repeated GA experiments
  (dominating lines show better GA performance)
Multi-functional embedded systems

• Modern embedded systems need to support multiple applications and standards

• Multiple applications can be active simultaneously, contending for system resources

• Application workload may change over time
  ✓ System demands change over time
Multi-functional embedded systems

- Modern embedded systems need to support multiple applications and standards
- Multiple applications can be active simultaneously, contending for system resources
- Application workload may change over time
  ✓ System demands change over time

How to perform DSE for multi-application workloads?
How to deal with dynamic workload behavior?
DSE for multi-application systems: scenario-based DSE
Scenarios: they are exponential
Scenarios: they are exponential

Infeasible to evaluate design points using all possible application workload scenario’s!
Scenarios: they are exponential

Use a small, representative subset of application scenarios to evaluate designs!
Scenario-based DSE

Inter and intra application scenarios

Scenario detection

Scenario database

Workload generation

Parameterized MP-SoC platform specification

Scenario and mapping/platform co-exploration

Pareto front for average scenario behavior

[ICCD’10, IEEE TCAD’13]
The need for system adaptivity

• Cope with changing (demands of) application workloads
• Dynamic QoS management allowing to trade off different system qualities like performance, precision and power consumption
• Cope with transient and/or permanent system faults
The need for system adaptivity

• Cope with changing (demands of) application workloads

• Dynamic QoS management allowing to trade off different system qualities like performance, precision and power consumption

• Cope with transient and/or permanent system faults

• Types of adaptivity:
  • Component reconfiguration (e.g., DVFS, reconfigurable HW, reconfigurable network, etc.)
  • Run-time (re-)mapping of application tasks
Run-time adaptive systems

Design-time optimization

Run-time reconfiguration
Run-time adaptive systems

How to find optimal system configurations at runtime using light-weight algorithms? When to migrate tasks?
Adaptive MPSoCs

- Re-mapping (migration) of tasks not always beneficial!
  - Dependent on workload scenario duration
Adaptive MPSoCs

• Re-mapping (migration) of tasks not always beneficial!
  • Dependent on workload scenario duration
• This leads to a need for adaptivity throttling
  • Predict whether or not it is beneficial to re-map

\[
(T_{\text{Current\_mapping}} - T_{\text{New\_mapping}}) \times \text{duration} > \text{overhead of remapping}
\]
Adaptive MPSoCs

• Re-mapping (migration) of tasks not always beneficial!
  • Dependent on workload scenario duration
• This leads to a need for adaptivity throttling
  • Predict whether or not it is beneficial to re-map

\[
\left( T_{\text{Current\_mapping}} - T_{\text{New\_mapping}} \right) \times \text{duration} > \text{overhead of remapping}
\]

Needs prediction
Incorporating additional optimization objectives
Reliability-aware DSE

Incorporating fault-tolerance as design objective
Reliability-aware DSE

Incorporating fault-tolerance as design objective
Reliability-aware DSE

Incorporating fault-tolerance as design objective

- Detection
- Recovery
  - E.g. trade-off checkpoint overhead / restart overhead
- Design options
  - Different effects on reliability
  - Affects other objectives (like performance, power and costs)

[CODES+ISSS’12]
Security-aware DSE?

• Increasing ubiquity and connectivity of embedded systems \(\Rightarrow\) security!

• At this moment, security mostly an afterthought in the design process

• Security must be an objective in early DSE!
  • Security mechanisms affect other design objectives
Security-aware DSE?

- Increasing ubiquity and connectivity of embedded systems → security!
- At this moment, security mostly an afterthought in the design process
- Security must be an objective in early DSE!
  - Security mechanisms affect other design objectives

BIG CHALLENGE:
how do you quantify the level of security?