Hardware-Software Codesign

3. Mapping Applications To Architectures

Lothar Thiele
System Design

specification

system synthesis

estimation

SW-compilation

instruction set

HW-synthesis

intellectual prop. code

machine code

intellectual prop. block

net lists
System Synthesis - Mapping

Mapping transforms behavior into structure and execution

- **Allocation**: select components
- **Binding**: assign functions to components
- **Scheduling**: determine execution order

... finally, synthesis results into **implementation**
Application Specification

... using an underlying model of computation

- some examples (see also next slides)
  - task graphs: data flow graph, control flow graph
  - process networks: Kahn process network, synchronous data-flow
  - state machine representations: SpecCharts, StateCharts [not covered in this course]

- for mapping, very often only the process network structure and its (abstract) properties are relevant (abstraction from detailed functionality)
data flow graph (DFG)

\[ x = 3a + b^2 - c; \]
\[ y = a + b^2x; \]
\[ z = b - c(a + b); \]
### Specification Example 2

**control flow graph (CFG)**

```plaintext
code
what_is_this {
  read(a,b);
  done = FALSE;
  repeat {
    if(a>b)
      a = a-b;
    elseif(b>a)
      b = b-a;
    else done = TRUE;
  } until done;
  write(a);
}
```

```
1     read(a,b);
2     done = FALSE;
3     repeat {
4       if(a>b)
5          a = a-b;
6       elseif(b>a)
7          b = b-a;
8       else done = TRUE;
9     } until done;
10    write(a);
```
Specification Example 3

Kahn process network

- example: hierarchical network of MJPEG application

![Diagram of Kahn process network with nodes and edges representing MJPEG application subnetworks.](image-url)
Architecture Specification

- reflects the **structure** and (key) **properties** of the underlying platform
- ... and can be done at **different abstraction levels**

**example: system-level architecture specification**

- usually a graph notation is used to describe **structure**
- annotations to graph elements reflect **properties** of the underlying platform, e.g., processing frequency

![Diagram of system-level architecture specification]

- processor
- memory
- hardware channel
- input / output /
- bidirectional port
- connection
Mapping Specification

- mapping relates application and architecture specifications
  - **binds** processes to processors
  - **binds** communication between processes to communication paths of the architecture
  - specifies **resource sharing disciplines** and **scheduling**

Mapping = binding + scheduling
Example DFG Application Model

basic model: data flow graph and static scheduling

data flow graph $G_P(V_P, E_P)$

Interpretation:

- $V_P$ consists of functional nodes $V_P^f$ (task, procedure) and communication nodes $V_P^c$.
- $E_P$ represent data dependencies
Example Architecture Model

Architecture graph $G_A(V_A, E_A)$:

- $V_A$ consists of functional resources $V_A^f$ (RISC, ASIC) and bus resources $V_A^c$. These components are potentially allocatable.
- $E_A$ model directed communication.
**Example Mapping**

**Definition:** A *specification graph* is a graph $G_S=(V_S,E_S)$ consisting of a *data flow* graph $G_P$, an architecture graph $G_A$, and edges $E_M$. In particular, $V_S=V_P \cup V_A$, $E_S=E_P \cup E_A \cup E_M$.
Example Mapping

Mapping involves

- **allocation** $\alpha$ (subset of $V_A$)

- **binding** $\beta$ (subset of $E_M$), i.e., reflecting the mapping of application nodes in $V_P$ (functional and communication) onto architectural nodes in $V_A$ (processors and buses)

- **scheduling** $\tau$, i.e., assigning an order among nodes (e.g., start time)

- ...and ultimately, **implementation of** $(\alpha, \beta, \tau)$ targeted to actual hardware and/or software modules
**Definition**: Given a specification graph $G_S$, an implementation is a triple $(\alpha, \beta, \tau)$, where $\alpha$ is a feasible allocation, $\beta$ is a feasible binding, and $\tau$ is a schedule.
**Definition:** Given a specification graph $G_S$, an implementation is a triple $(\alpha, \beta, \tau)$, where $\alpha$ is a feasible allocation, $\beta$ is a feasible binding, and $\tau$ is a schedule.

**Problem graph $G_P(V_P,E_P)$:**

- $V_P$ consists of functional nodes $V_P^f$ and communication nodes $V_P^c$.
- $E_P$ represents data dependencies.

**Interpretation:**

- $V_P^f$ represents tasks.
- $V_P^c$ represents procedures.
- $E_P$ represents data dependencies.

**Application & mapping:**

- The application and mapping of functions and procedures are visualized through the directed graph.

**Execution schedule:**

The execution schedule shows the allocation and binding of processes to hardware components along with the PTP bus and shared bus interactions.
System Exploration

specification

system synthesis

estimation

SW-compilation

instruction set

HW-synthesis

intellectual prop. code

machine code

intellectual prop. block

net lists
Mapping Optimization in the Design Flow

- functional validation
- application
- architecture

system-level design space exploration loop

- mapping
- search algorithm
- system-level performance analysis

- hardware/software synthesis
- final performance analysis

complete design space exploration loop (typ. manual)
Optimization via Design Space Exploration

- Often based on iterative interaction between optimization and performance analysis
- Based on quantitative parameters obtained through estimation, e.g., end-to-end delay, throughput, power, temperature
What is ahead?

- **Section 4: Partitioning**
  - Some basic methods to assign tasks to computing resources (or communication to networks and busses).

- **Section 5: Multiobjective Optimization**
  - Generic method to optimize H/S systems (and other engineering systems)

- **Section 7: Design Space Exploration**
  - Application of these methods to design space exploration