Hardware-Software Codesign

7. Design Space Exploration

Lothar Thiele
System Design

specification

system synthesis

SW-compilation

instruction set

HW-synthesis

machine code

net lists

estimation

intellectual prop. code

intellectual prop. block
Optimization-Analysis Cycle

- decision vector $X$
- allocation
- binding
- schedule
- evaluation model (e.g., simulation, analytic)
- allocation
- binding
- schedule
- optimization algorithm
- make decisions only by knowing (and comparing) $f$
- cost
- throughput
- delay
- memory
- objective vector $f(X)$
Example: Simple Mapping Model

search algorithm

EA

1. selection
2. recombination
3. mutation

“chromosome” = encoded allocation + binding

solutions

allocation

binding

analysis of individual solutions

decode allocation

decode binding

scheduling

fitness evaluation

user constraints

fitness

design point (implementation)

binding $\beta$

scheduling $\tau$

allocation $\alpha$

fitness evaluation

search algorithm

solutions

analysis of individual solutions

user constraints
Remember ...

**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, \( V_S = V_P \cup V_A \), \( E_S = E_P \cup E_A \cup E_M \).
**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.
Challenges of EAs in DSE

- encoding allocation + binding
  - **simple encoding**
    - e.g., one bit per resource, one variable per binding
      - easy to implement
      - ... however, it may lead to (many) infeasible partitioning solutions
  - **encoding + repair**
    - e.g. simple encoding and repair for allocation
      - s.t. for each $v_p \in V_p$ there exists at least one $v_a \in \alpha$ with $(v_p, v_a) \in E_m$
      - reduces number of infeasible partitioning solutions

- (“smart”) generation of initial population

- (“smart”) neighborhood operations, e.g., mutation, crossover
Example Network Processors - Definition

- Typically, network processors serve as bridge between the network and the source/sink audio/video device (or set of devices)

- **implementation**: high-performance, programmable devices optimized for (real-time) network packet processing

- **features**: complex packet processing capabilities at high line speeds (routing; forwarding; de-/encryption; de-/compression; ...) and means to guarantee quality-of-service
Network Processor Architecture (*)

Network processor heterogeneous hardware/software architecture:

- available processing units
  - … are described in resource set \( R = \{\text{ARM9}, \text{PowerPC}, \text{DSP}, \text{MEngine}, \text{Classifier}, \text{Cipher}, \text{LookUp}, \text{CheckSum}\} \)
  - … have a relative implementation cost \( \text{cost}(r) \geq 0, \ r \in R \)
  - … and are selected for a specific architecture during the allocation step
    - with \( \text{alloc}(r) = 1 \) if a resource is selected and 0 otherwise

Network Processor Task Model

- **application structure**: set of streams $s \in S$ and set of tasks $t \in T$
  - each stream includes an ordered sequence of tasks $V(s) = [t_0, \ldots, t_n]$
- **example**:
  $S = \{RTSend, NRTDecrypt, NRTEncrypt, RTRecv, NRTForward\}$
Problem: Optimal Design of Network Processor

- M ⊆ T × R: all possible bindings of tasks
  - i.e., if (t, r) ∈ M, then task t could be executed on resource r
- request \( w(r, t) \geq 0 \)
  - i.e., execution of one packet in t would use \( w \) computing units of \( r \)
- resource allocation cost \( c(r) \geq 0 \)

⇒ binding \( Z \) of tasks to resources \( Z \subseteq M \) (leading to actual implementation)
  - subset of mappings \( M \) s.t. every task \( t \in T \) is bound to exactly one allocated resource \( r \in R \) and \( \text{alloc}(r) = 1 \) and \( r = \text{bind}(t) \)
NP Design Constraints

the design of network processors typically faces conflicting goals:

- **delay constraints**
  - e.g., maximal time a packet is processed within NP

- **throughput maximization**
  - e.g., maximum throughput of NP (packets per second)

- **cost minimization**
  - implementation with small amount of resources (e.g., processing units, memory, and communication networks)

- ... and conflicting usage scenarios
  - usually, a packet processor is used in several different systems (e.g., router or consumer multimedia processing device) and might have different implementations with different throughput/delay requirements
NP Design Space Exploration

issues to be considered during system-level design (and synthesis):

- **allocation**
  - determine hardware components of the network processor

- **binding**
  - for each process of the software application choose an allocated hardware unit which executes it

- **scheduling**
  - for the set of tasks mapped onto a specific resource choose scheduling policy/parameters – from available run-time environment, e.g., a fixed priority for each stream \( s: \text{prio}(s) > 0 \)
Design Space Exploration Flow

Hardware Architecture Template

Software Application

Run-Time Environment

Application Scenarios

VARIATION

Allocation

HW Architecture

Binding

HW/SW Architecture

Scheduling

Performance Analysis

Multiobjective Evolutionary Selection

Selected Architectures

Constraints Opt. Criteria

HW/SW Architectures Cost and Performance

SELECTION

alloc(r) = 0/1

r = bind(t)

prio(s) > 0

alloc(r) = 0/1
Tools and a Small Demo

![Graph and diagram showing various data points and utilization percentages.]

- **Scenario: Scen2**
  - Optimal Scaling Factor: 0.530
  - Total Memory: 8.295

- **Flow: RTRsend**
  - Priority: 5
  - Acc. Waiting Time in Queue: 0.000

- **Flow: NRTDecrypt**
  - Priority: 4
  - Acc. Waiting Time in Queue: 0.000

- **Flow: RTRrecv**
  - Priority: 1
  - Acc. Waiting Time in Queue: 0.000

- **Flow: NRTForward**
  - Priority: 3
  - Acc. Waiting Time in Queue: 23.098
... Some Results

- **Performance of encryption/decryption**
  - **DSP**
    - NRT: 64%
    - RT: 39%
  - **Cipher**
    - NRT: 71%
    - RT: 0%

- **Performance of RT voice processing**
  - **LookUp**
    - NRT: 15%
    - RT: 6%
  - **Classifier**
    - NRT: 27%
    - RT: 11%
  - **LookUp**
    - NRT: 1%
    - RT: 6%
  - **Classifier**
    - NRT: 1%
    - RT: 11%
Example: Wave Field Synthesis

What is wave field synthesis (WFS)?
- high quality spatial sound reproduction system for huge listening areas
- 32 sound sources and 300 loudspeakers for medium sized reproduction rooms
System Specification: WFS Application

Parallel application modeled as Kahn process network

structure: XML

functionality: ANSI C & DOL(*) API

Algorithm 1 Process Model
1: procedure INIT(DOLProcess p) ▷ initialization
2: initialize local data structures
3: end procedure
4: procedure FIRE(DOLProcess p) ▷ execution
5: DOL_read(INPUT, size, buf) ▷ blocking read
6: manipulate
7: DOL_write(OUTPUT, size, buf) ▷ blocking write
8: end procedure

(*) DOL – distributed operation layer: http://www.tik.ee.ethz.ch/~shapes/dol.html
System Specification: Architecture

- Architecture is modeled at abstract level in XML format
- Modeled elements:
  - processors, buses, memories
  - communication paths between these elements
  - … parameters are included in the model
Application-to-Architecture Mapping

parallel application

correlation

sum

loudspeakers

heterogeneous architecture

microphones

DSP

SSC

SREG

DXM

DDM

DMA

RDM

RISC

design space exploration

(performance analysis & mapping optimization)

software synthesis
Simple Analysis Model

\[
\text{obj}_1 = \max_{c \in C} \left\{ \sum_{\forall p \text{ mapped to } c} n(p) \cdot r(p, c) \right\}
\]

max processor load

number of activations of process p

runtime of process p on processor c

\[
\text{obj}_2 = \max_{g \in G} \left\{ \sum_{\forall s \text{ mapped onto } g} \frac{b(s)}{t(g)} \right\}
\]

max bus load

communication link with worst load

communication request from channel s

bandwidth of communication link g
Where Are Data Obtained From?

- **Static parameters**: bandwidth of buses \( t(g) \)
- **Functional simulation**: number of activations for each process \( n(p) \), amount of data for each channel \( b(s) \)
- **Instruction-set simulation**: runtime of each process on different processors \( r(p,c) \) by using benchmark mappings
Design Space Exploration Cycle – An Example
EXPO: Example

- microphones
- convolution
- sum
- loudspeakers

max. bus load

search direction

single processor mapping

max. processor load