Hardware-Software Codesign

7. Design Space Exploration

Lothar Thiele
Optimization-Analysis Cycle

- decision vector $X$
- allocation
- binding
- schedule
- evaluation model (e.g., simulation, analytic)
- optimization algorithm
- make decisions only by knowing (and comparing) $f$
- objective vector $f(X)$
- cost
- throughput
- delay
- memory
- CPU0 CPU1 CPU2 CPU3
- bus
- p0 p1 p2 p3
- CPU0 CPU1 CPU2 CPU3
- bus
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

fitness

user constraints

design point (implementation)

binding \( \beta \)

scheduling \( \tau \)

allocation \( \alpha \)
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 modify
      - s.t. for each $\mathbf{v}_p \in \mathbf{V}_p$ there exists at least one $\mathbf{v}_a \in \mathbf{V}_A$ with $\beta(\mathbf{v}_p) = \mathbf{v}_a$
      - 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 $R = \{\text{ARM9, PowerPC, DSP, MEngine, Classifier, Cipher, LookUp, CheckSum}\}$
  - ... 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**:
  
  $S = \{\text{RTSend}, \text{NRTDecrypt}, \text{NRTEncrypt}, \text{RTRecv}, \text{NRTForward}\}$
Problem: Optimal Design of Network Processor

- mappings $M \subseteq T \times R$: all possible bindings of tasks
  - i.e., if $(t, r) \in M$, 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 $r \in R$ $\text{alloc}(r) = 1$ 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

**Allocation**
- $\text{alloc}(r) = 0/1$
- $r = \text{bind}(t)$
- $\text{prio}(s) > 0$

**Selected Architectures**
- $c_{\text{MAX}}$
- $w_{\text{MAX}}$

**Multiobjective Evolutionary Selection**
- Constraints Opt. Criteria

**Performance Analysis**
- $c(\text{all } r)$
- $w(\text{all } t)$

**Variation**
- **Selection**

**HW Architecture**

**HW/SW Architecture**

**HW/SW Architectures Cost and Performance**
Tools and a Small Demo

**EXPO - A Tool for Design**

- Initialisation sequence started.
- Static parameters read.
- Problem specification read.
- Initial population constructed.
- Initial population written to file.
- Population written to file.
- Generation counter set to 1.
- Generation 1
- All active gene IDs read.
- Population before cleaning.
- Population after cleaning: 101 elements.
- Clean of population finished.
- Population written to file.
- Genes for variation read.
- Variation finished.
- Generation 2
- All active gene IDs read.

**Implementation Nr. 60641**

- **Scenario: Scen2**
  - Optimal Scaling Factor: 0.530
  - Total Memory: 8.295
  - **DSP**
    - Utilization: 79%
  - **Checksum**
    - Utilization: 4%
  - **LockUp**
    - Utilization: 7%

**Flow: RTSend**
- Priority: 5
- Acc. Waiting Time in Queue: 0.000
- RTPtx
- VoiceEnc
- LinkTx
- Schedule
- Decrypt
- AHVerify
- Classify
- LinkRx

**Flow: NRTDecrypt**
- Priority: 4
- Acc. Waiting Time in Queue: 0.000
- ESPDecaps
- ProcessIP
- IFModify
- LinkTx
- Schedule
- Decrypt
- AHVerify
- Classify
- LinkRx

**Flow: RTRecv**
- Priority: 1
- Acc. Waiting Time in Queue: 0.000
- Dejitter
- VoiceDec
- ProcessIP
- RTPtx
- Classify
- LinkRx

**Flow: NRTForward**
- Priority: 3
- Acc. Waiting Time in Queue: 23.068
- ProcessIP
- VerifyIP
- UDPrx
... Some Results

performance of encryption/decryption

performance of RT voice processing

downward arrow from cost

downward arrows from DSP

- DSP
  - NRT: 64%
  - RT: 39%

circular arrow from Cipher

- Cipher
  - NRT: 71%
  - RT: 0%

circular arrow from LookUp

- LookUp
  - NRT: 15%
  - RT: 6%

circular arrow from Classifier

- Classifier
  - NRT: 27%
  - 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 \( p \)) \( \triangleright \) initialization
2: initialize local data structures
3: end procedure
4: procedure FIRE(DOLProcess \( p \)) \( \triangleright \) execution
5: DOL_read(INPUT, size, buf) \( \triangleright \) blocking read
6: manipulate
7: DOL_write(OUTPUT, size, buf) \( \triangleright \) 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

heterogeneous architecture

design space exploration
(performance analysis & mapping optimization)

software synthesis
Simple Analysis Model

\[ obj_1 = \max_{c \in \mathcal{C}} \sum_{\forall p \text{ mapped to } c} n(p) \cdot r(p, c) \]

- Processor \( c \) with worst total runtime
- Number of activations of process \( p \)
- Runtime of process \( p \) on processor \( c \)

\[ obj_2 = \max_{g \in \mathcal{G}} \sum_{\forall s \text{ mapped onto } g} \frac{b(s)}{t(g)} \]

- Communication link with worst load
- Communication request from channel \( s \)
- Bandwidth of communication link \( g \)

\( \mathcal{C} \): Set of processors
\( \mathcal{G} \): Set of communication links
\( n(p) \): Number of activations of process \( p \)
\( r(p, c) \): Runtime of process \( p \) on processor \( c \)
\( b(s) \): Bandwidth of communication link \( s \)
\( t(g) \): Total runtime of processes on processor \( g \)
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