Hardware-Software Codesign

8. Performance Estimation

Lothar Thiele
System Design

- Specification
- System synthesis
- Estimation
- SW-compilation
- Instruction set
- HW-synthesis
- Intellectual prop. code
- Machine code
- Net lists
- Intellectual prop. block
Motivation

Objective function values that guide the design space exploration are obtained through performance estimation.

Design space exploration may change

- **application** (algorithms and/or parallelization)
- **architecture** (hardware)
- **mapping** (binding and scheduling)

based on system (estimated) performance
Performance Estimation – Global Picture

**METRIC**
- time
- cost
- area
- power
- other: quality, SNR, ...

**ABSTRACTION LEVEL**
- HW subsystem
- SW subsystem
- CPU subsystem
- interconnect subsystem

**SUBSYSTEM TO ANALYZE**
- MPSoC

**METHOD**
- analytic
- simulation
- statistic

**x(y) = x0 * exp (-k0*y)**
- x0 = 105
- k0 = 1.2593

**Note:**
- RTL – Register Transfer Level
- ISA – Instruction Set Architecture
- TLM – Transaction-Level Model
- OS – Operating System
- HLL – High-Level Language
- APP – Application
Performance Estimation in Design Flow

- high-level
  - **advantages**: short estimation time, implementation details not necessary
  - **drawbacks**: limited accuracy, e.g. no info about timing

- low-level
  - **advantages**: higher accuracy
  - **drawbacks**: long estimation time, many implementation details need to be known
Why Do We Need Performance Estimation?

- Validation of non-functional aspects
  - equivalence between specification and implementation of non-functional properties (e.g., timing, power, energy, memory consumption)

- Design space exploration
  (guiding design decisions and optimization)
  - part of the feedback cycle (see global flow)
Performance Estimation – Global Picture

ABSTRACTION LEVEL

METHOD

METRIC

other: quality, SNR, ...
cost
time
area
power

x(y) = x0 * exp (-k0*y)

Note:
RTL – Register Transfer Level
ISA – Instruction Set Architecture
TLM – Transaction-Level Model
OS – Operating System
HLL – High-Level Language
APP – Application

HW IP
CPU
interconnect
intermediary level e.g., TLM, OS

low-level e.g., RTL, ISA

APP
task1

MPSoC

SW subsystem

interconnect subsystem

SW subsystem

CPU subsystem

HW subsystem

interconnect

task1
task2
task3

high-level, e.g., functional, HLL

Note:
RTL – Register Transfer Level
ISA – Instruction Set Architecture
TLM – Transaction-Level Model
OS – Operating System
HLL – High-Level Language
APP – Application
Why is MPSoC Performance Estimation Difficult?

- **Computation, communication, and memory**
  - (non-deterministic) computation in processing nodes
  - (non-deterministic) communication delays
  - (non-deterministic) memory accesses
  - complex resource interaction via scheduling/arbitration

- **Cyclic timing dependencies**
  - internal data streams interact on computation and communication resources
  - interaction determines stream characteristics

- **Uncertain environment**
  - different load scenarios
  - unknown (worst case) inputs
Illustration of Evaluation Difficulties

input stream

ab acc b

task communication

task scheduling

complex input:
- timing (jitter, bursts, ...)
- different event types
Illustration of Evaluation Difficulties

- task communication
- task scheduling
- complex input:
  - timing (jitter, bursts, ...)
  - different event types
- variable resource availability
- variable execution demand
  - input (different event types)
  - internal state (program, cache, ...)
Performance Estimation Requirements

- Estimation should be **composable** in terms of
  - subsystems and their *interactions*, i.e., HW, SW, interconnect
  - *computation, communication, memory, scheduling*

- Estimation should cover different **metrics**
  - e.g., delay, throughput, memory consumption, power, energy, temperature, cost, ...

- Estimation should represent a **reasonable trade-off** between
  - (1) effort in terms of computation time
  - (2) accuracy of performance estimates
  - (3) set-up time / modeling effort
Performance Estimation – Global Picture

**ABSTRACTION LEVEL**
- low-level, e.g., RTL, ISA
- intermediary level e.g., TLM, OS
- high-level, e.g., functional, HLL

**METHOD**
- analytic
- simulation
- statistic

**METRIC**
- cost
- area
- power
- time
- statistic
- simulation

**SUBSYSTEM TO ANALYZE**
- MPSoC
- SW subsystem
- HW subsystem
- interconnect subsystem

**Note:**
- RTL – Register Transfer Level
- ISA – Instruction Set Architecture
- TLM – Transaction-Level Model
- OS – Operating System
- HLL – High-Level Language
- APP – Application
Short History in Abstraction

technology: transistors, layouts

signal: gate, schematic, RTL

transaction: SW, HW systems

tokens: SW tasks, comm. backbones, IPs

simulator: SPICE

simulator: VHDL

simulator: SystemC /ISS

simulator: HW/SW codes./cosim. tools; formal methods

1970’s

gate level model
1/0/X/U (D ns)

transistor model
(t=RC)

1980’s

register-transfer level model
data[1011011] (critical path latency)

1990’s

1990’s

2000’s

SW
HW

SW Tasks
OS/drivers
CPU
HW adaptation
IPs

SW adaptation

on-chip
communication
Network

HW adaptation
IPs

© Adapted from Cadence Design Systems Inc.
Performance Estimation in Design Flow

- **high-level**
  - **advantages**: short estimation time, implementation details not necessary
  - **drawbacks**: limited accuracy, e.g. no info about timing

- **low-level**
  - **advantages**: higher accuracy
  - **drawbacks**: long estimation time, many implementation details need to be known
Performance Estimation – Global Picture

**ABSTRACTION LEVEL**

- **HW IP**
- **SW**
- **SW**
- **SW**
- **SW**
- **CPU**
- **AD**
- **I/O**
- **Mem**
- **HW subsystem**
- **CPU subsystem**
- **SW subsystem**
- **interconnect**
- **MPSoC**
- **SUBSYSTEM TO ANALYZE**

**METHOD**

- **analytic**
- **simulation**
- **statistic**

**METRIC**

- cost
- area
- power
- time
- other: quality, SNR, ...

**x(y) = x0 * exp(-k0*y)**

- **x0 = 105**
- **k0 = 1.2593**

**Note:**

- **RTL** – Register Transfer Level
- **ISA** – Instruction Set Architecture
- **TLM** – Transaction-Level Model
- **OS** – Operating System
- **HLL** – High-Level Language
- **APP** – Application
Performance Metrics

*Performance metric* = function defined on relevant non-functional properties of a system, which gives a quantitative indication on system execution.

**Examples (relevant for embedded systems)**

- **time** [seconds]
  - e.g., end-to-end delay, throughput, latency

- **power, energy, temperature** [mW, mJ, K]
  - e.g., power consumed by the network, energy to execute a task, maximal temperature

- **area** [mm²]
  - e.g., area of an integrated circuit

- **cost** [$]
  - e.g., cost of parts, labor, development cost

- **other metrics**
  - e.g., SNR (signal to noise ratio) - video/sound quality

**generally, performance metrics are conflicting!**
Performance Estimation – Global Picture

**ABSTRACTION LEVEL**

**METHOD**

- **analytic**
- **simulation**
- **statistic**

**METRIC**

- other: quality, SNR, ...
- cost
- area
- time
- power
- other: quality, SNR, ...

**SUBSYSTEM TO ANALYZE**

- HW IP
- SW subsystem
- interconnect subsystem
- MPSOC

**Note:**
- RTL – Register Transfer Level
- ISA – Instruction Set Architecture
- TLM – Transaction-Level Model
- OS – Operating System
- HLL – High-Level Language
- APP – Application
Performance Estimation Methods – Illustration

- real system
- worst-case
- best-case

e.g. delay

measurement | simulation | probabilistic estimation | worst case (formal) analysis

→ lecture 6
→ lectures 9-10
Performance Estimation Methods – Description

How to evaluate?

- **measurements**
  - use existing instance of the system to measure performance

- **simulation**
  - develop a program which implements a model of the system and evaluate performance by running the program

- **statistics**
  - develop a statistical abstraction of the system and derive statistic performance via analysis or simulation

- **formal analysis**
  - develop a mathematical abstraction of the system and compute formulas which describe the system performance
System Compositional Performance Estimation

- designers experience
- component simulation
- parameters
- model of application
- model of environment
- system model
- model of architecture
- estimation tool (method)
- estimation results

- input traces
- parameters
- spec. of inputs

- data sheets
- platform benchmarks
- parameters
1. Static Analytic Models

- Describe computing, communication, and memory resources by *algebraic equations*.
- Describe system properties by *parameters*, e.g., data rate.
- Combine relations.

\[
\text{comm\_delay} = \frac{\#\text{words}(M1, M2)}{\text{burst\_size}} \text{ comm\_time}
\]

+ *fast and simple estimation*
- *generally inaccurate modeling* (e.g., resource sharing not modeled)
2. Dynamic Analytic Models

- Combination between
  - **static models** possibly extended by non-determinism in run-time and event processing
  - **dynamic models** for describing, e.g., resource sharing mechanisms (scheduling and arbitration)

- Existing approaches
  - **classical real-time scheduling theory**
  - **stochastic queuing theory**
    - (statistical bounds) – example 1
  - **analytic (non-deterministic) queuing theory**
    - (worst case/best case bounds) – example 2
Example – Queuing Systems

- **Example:** clients request some service from a server over a network
- **Analysis goals:**
  - performance of the server
  - performance of the network

![Diagram of a system model and real system](image)
Ex. 1 – Stochastic Queuing Systems

- A **stochastic model** of queuing systems is described by probability density functions (distributions) of:
  - arrival rates
  - service mechanisms
  - queuing disciplines

- **Performance measures** are stochastic values (functions):
  - average delay in queue
  - time-average number of customers in queue
  - proportion of time server is busy

**classical M/M/1 queuing system:**
(M = Markovian (exponential) distribution)
Ex. 2 – Worst-Case/Best-Case Queuing Systems

A worst/best-case queuing system is described by worst/best-case bounds on system properties:
- w/b-case bounds on arrival times
- w/b-case bounds on server behavior
- resource interaction

**Performance measures**
- worst/best-case delay in queue
- worst/best-case number of customers in queue
- worst/best-case system delay

*discussed in lecture 10*
3. Simulation

*discussed in lecture 6*

**Model:**
- program implementing a model of the system (application, hardware platform, and mapping)
- performance is evaluated by running the program

![Model diagram]

- **input trace**
- **model (virtual)**: application, hardware platform, mapping
- **output trace**
- **performance estimations**
3. Simulation (contn.)

Simulation

- considers hardware platform and mapping of application onto that platform (by means of a virtual platform)
- combines functional simulation and performance data
- evaluates behavior for one simulation scenario (i.e., specific input trace and initial system state)

Advantages/disadvantages

- (typically) complex set-up and extensive runtimes
- ... but accurate results and good debugging possibilities
**Ex. 3: Trace-Based Simulation**

- **Trace-based simulation**: abstract system-level simulation (without timing)
  - faster than low-level simulation
  - ... but still based on a single input trace

- **Abstraction**
  - **application**: abstract execution traces
    - graph of events: read, write, and execute
  - **architecture**: “virtual machines” and “virtual channels”
    - calibrated with non-functional properties (timing, power, energy)

---

e.g. [Lahiri et al., 2001], [Pimentel et al., 2006], [Huang et al., 2009]
Ex. 3: Trace-Based Simulation

- Trace-based simulation steps
  - **build application abstract model**
    → execution trace determined by functional application simulation
  - **extend abstract model with architecture and mapping**
    → event graph extended by non-functional properties of virtual architecture elements
  - **simulation of extended model**

---

e.g. [Lahiri et al., 2001], [Pimentel et al., 2006], [Huang et al., 2009]
What is ahead?

- **Section 6: Simulation (already done...)**
  - Simulation-based approaches to system estimation.

- **Section 9: Worst Case Execution Time Analysis**
  - Analytic method to bound the execution time of tasks.

- **Section 10: Modular Performance Analysis**
  - Analytic method to bound properties of distributed hardware-software systems.

- **Section 11: Thermal-aware Design**
  - Simulation-based and analytic methods for temperature estimation