Hardware-Software Codesign

8. Performance Estimation

Lothar Thiele
System Design

specification

SW-compilation

estimation

instruction set

machine code

intellectual prop. code

net lists

intellectual prop. block

system synthesis

HW-synthesis
**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

METHOD

METRIC

ABSTRACTION LEVEL

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

HW IP

CPU

SW subsystem

SW subsystem

SW subsystem

interconnect subsystem

MPSoc

SUBSYSTEM TO ANALYZE

HW subsystem

CPU subsystem

intermediary level e.g., TLM, OS

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

low-level e.g., RTL, ISA

other: quality, SNR, ...

cost

area

power

time

analytic

simulation

statistic

x

y

task1

task2

task3

x

y

HW IP

CPU

interconnect

APP

HW IP

HW IP

CPU

interconnect

HW IP

SW

SW

SW

API

API

API

API

HW IP

HW IP

HW IP

HW IP

HW IP

HW IP

HW IP

HW IP

HW IP

HW IP
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

**METHOD**
- HW IP
- CPU
- Interconnect
- Intermediary level e.g., TLM, OS
- High-level, e.g., functional, HLL

**METRIC**
- Statistic
- Simulation
- Analytic
- Time
- Cost
- Area
- Power
- Other: quality, SNR, ...

**ABSTRACTION LEVEL**
- Low-level e.g., RTL, ISA
- Intermediary level e.g., TLM, OS
- 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
Multi-Processor System-on-Chip

communication templates

computation & memory templates

MPSoc

scheduling and arbitration templates

EDF

priority

EDF

proportional share

TDMA

fixed priority

static
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, ...)

input stream

ab acc b
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

**METHOD**
- analytic
- simulation
- statistic

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

**ABSTRACTION LEVEL**
- **HW IP**
- **CPU**
- **SW**
- **API**

**SUBSYSTEM TO ANALYZE**
- **HW subsystem**
- **CPU subsystem**
- **SW 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

1970’s
- technology: transistors, layouts
- transistor model (t=RC)

1980’s
- signal: gate, schematic, RTL
- gate level model 1/0/X/U (D ns)

1990’s
- transaction: SW, HW systems
- register-transfer level model data[1011011] (critical path latency)

2000’s
- tokens: SW tasks, comm. backbones, IPs
- simulator: VHDL

2010+
- simulator: HW/SW codes./cosim. tools; formal methods

© 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

**METHOD**
- Analytic
- Simulation
- Statistic

**METRIC**
- Statistic
- Simulation
- Analytic

**SUBSYSTEM TO ANALYZE**
- HW IP
- SW

**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** [mm$^2$]
  - 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**

- HW IP
- CPU
- 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
- other: quality, SNR, ...

**SUBSYSTEM TO ANALYZE**

- HW IP
- SW
- SW subsystem
- CPU subsystem
- 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

- Worst-case
- Best-case
- Real system
- 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
spec. of inputs

parameters

parameters

data sheets
platform benchmarks
1. Static Analytic Models

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

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

+ 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
Ex. 2.1 – 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 )

![Diagram of a classical M/M/1 queuing system](source)
**Ex. 2.2 – 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 9

Model:
- program implementing a model of the system (application, hardware platform, and mapping)
- performance is evaluated by running the program
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 average-case behavior for one simulation scenario (i.e., specific input trace and system state)

- **Advantages/disadvantages**
  - (typically) complex set-up and extensive runtimes
  - ... but accurate results and good debugging possibilities
Ex. 3.1: 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. 3.1: 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**

**Diagram:**
- Application functional model → complete trace → abstract event graph → trace-based simulation → estimation results

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

- **Section 6: Simulation**
  - 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.