Technische Universität Chemnitz
Fakultät für Informatik
Professur Technische Informatik
Prof. Dr. Wolfram Hardt

HW/SW Codesign II

Estimation

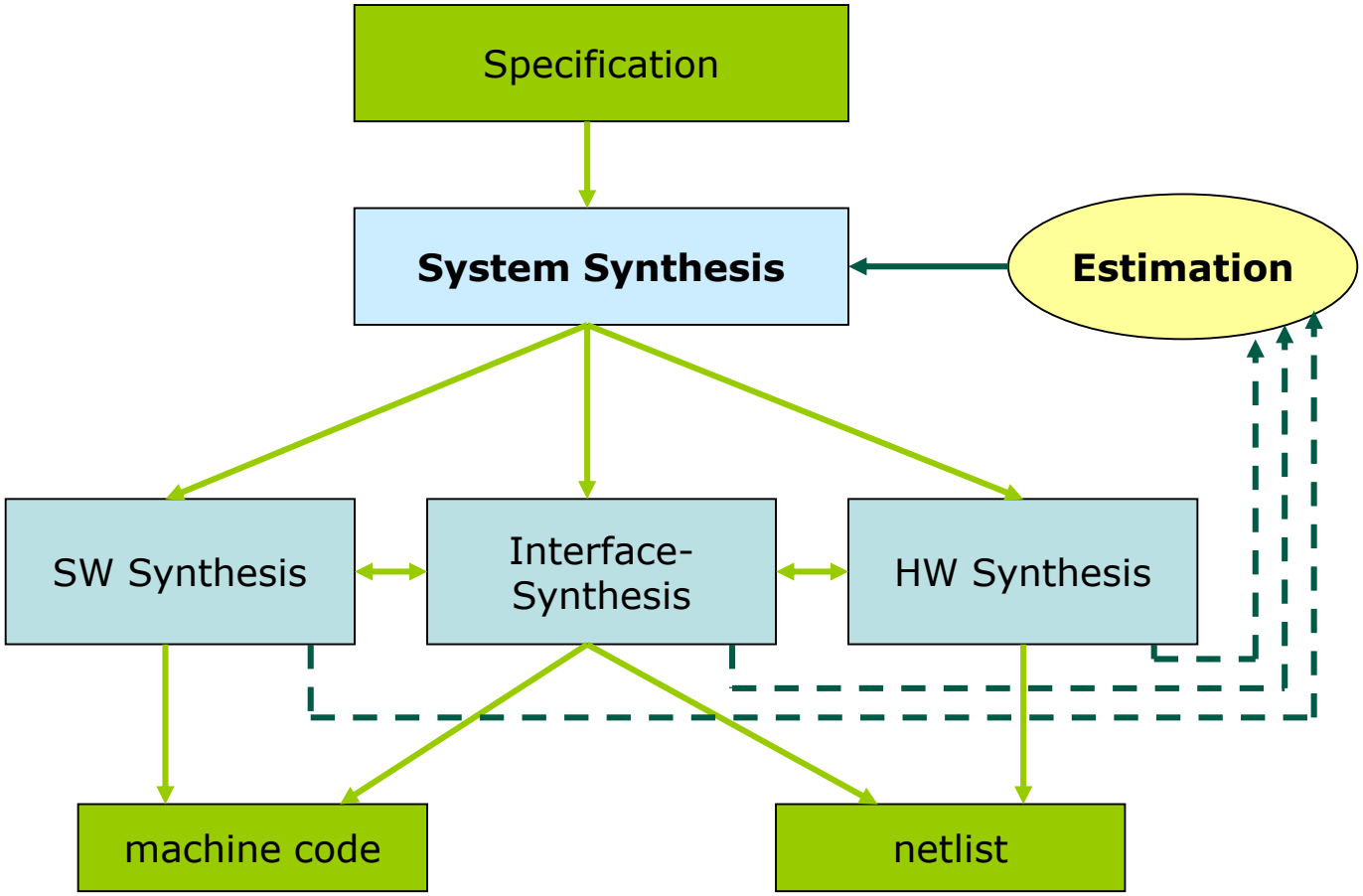
Prof. Dr. Wolfram Hardt
Dipl.-Inf. Michael Nagler

Sommersemester 2015
Version: 15.02.V.r-1.0-150407

Contents

- Parameters of Estimation Methods
- Estimation of Hardware Metrics
- Estimation of Software Metrics
- Estimation of Communication Metrics

Remember

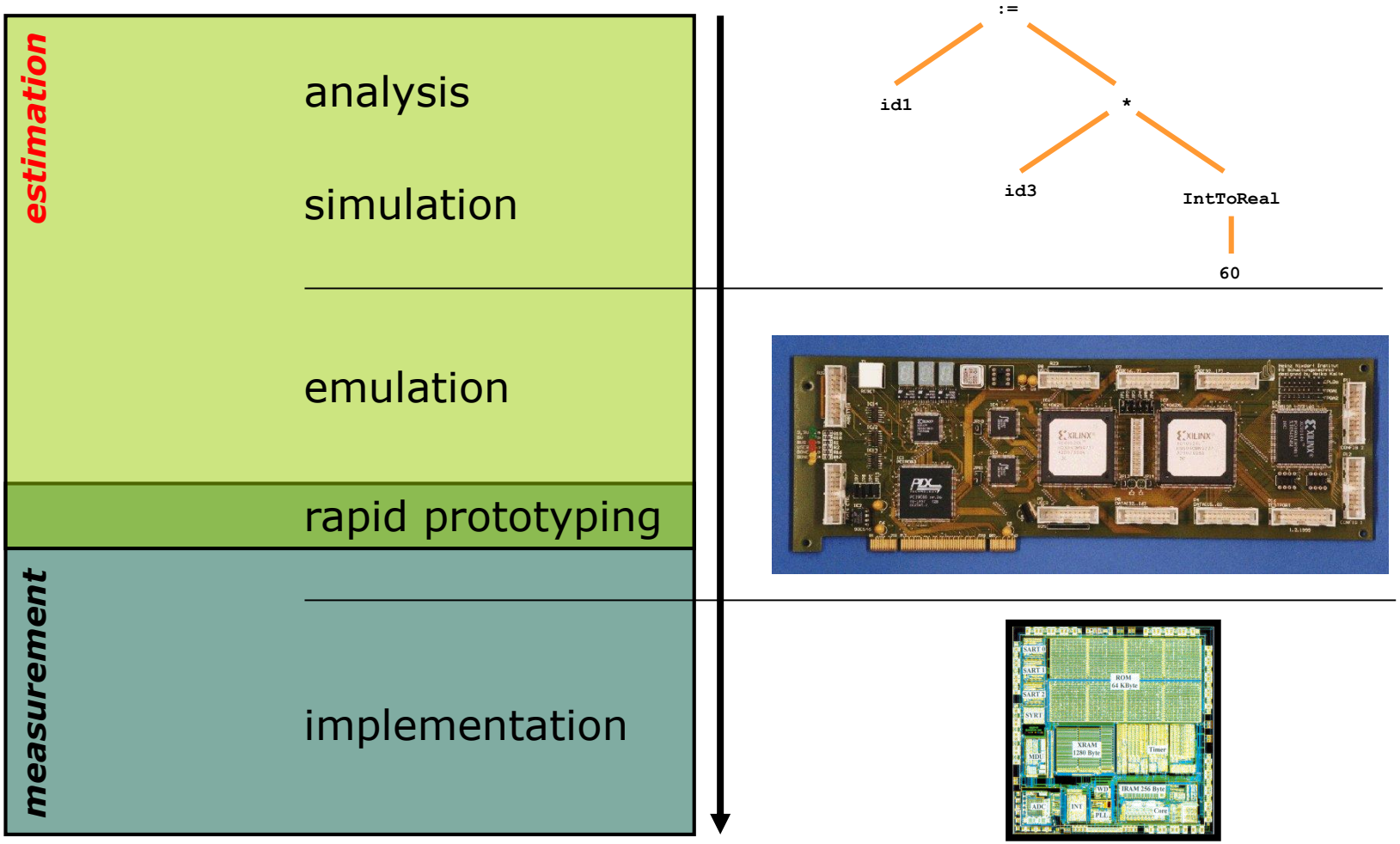

Specification
System Synthesis
Estimation
SW Synthesis
Interface-
Synthesis
HW Synthesis
machine code
netlist

Main Objective

- estimation (*Abschätzung*) is the determination of design parameters (characteristics) without _____

- benefits
  - support for design decisions
  - basis for design space exploration
  - basis for system optimization
  - reduction of time-to-market
  - cost reduction

Estimation in Design Process



Design Parameters/Metrics

- _____ metrics
  - performance
  - costs
  - power consumption
- _____ metrics
  - reliability
  - testability
  - maintainability
- business metrics
  - time-to-market
  - in-house-components

Accuracy

- **definition**

  Let $E(D)$ be an estimated and $M(D)$ an exact (measured) metrics of an implementation D. The accuracy (Exaktheit) A of the estimation is given by

---

$$if \ \left|E(D)\right| \leq \left|2 \cdot M(D)\right| \Rightarrow A \in [0,1]$$

Fidelity (I)

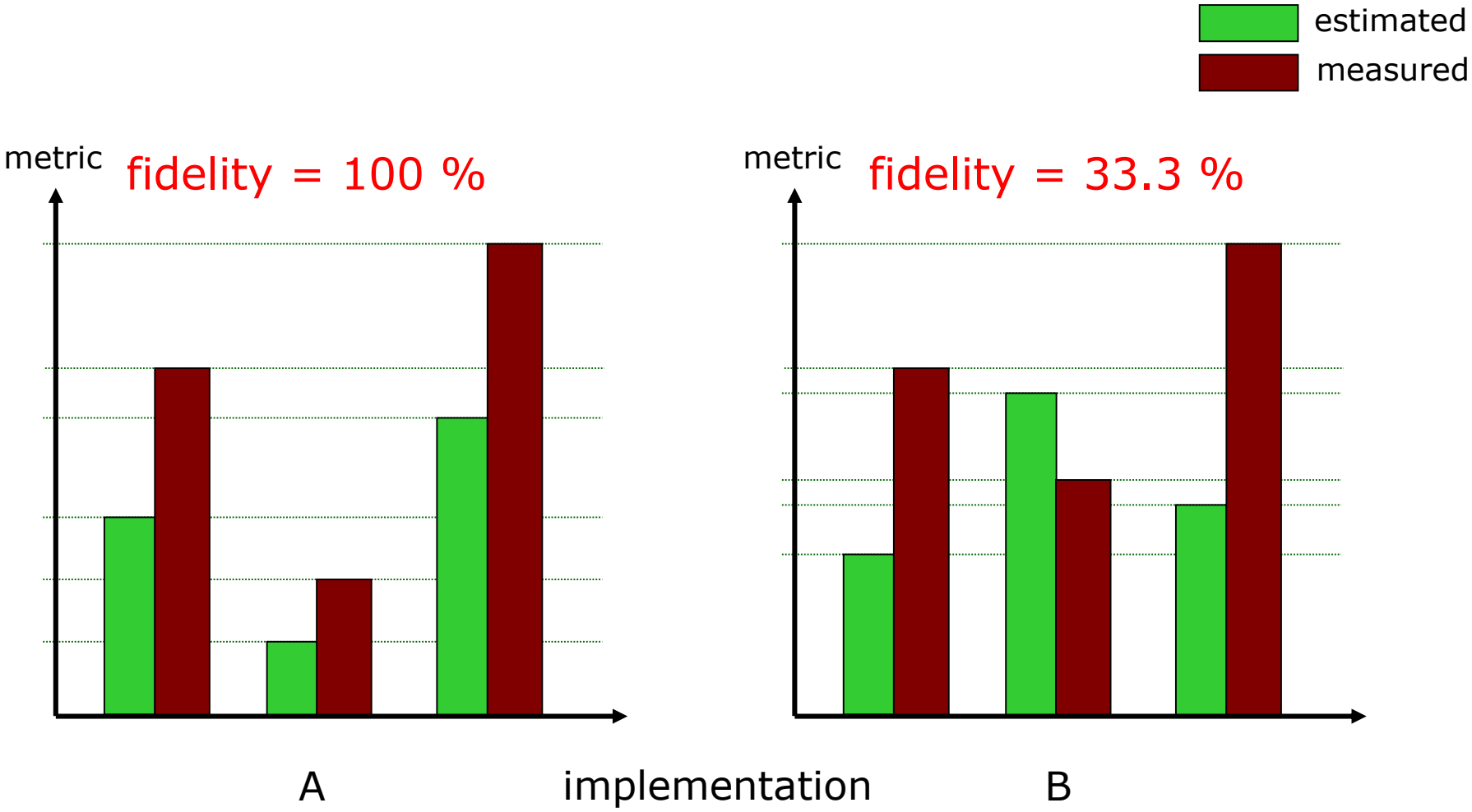
- **definition**

Let $D = \{D_1, D_2, \ldots, D_n\}$ be a set of implementations. The fidelity (Treue) $\boldsymbol{F}$ of an estimation method is given by

$$F = 100 \cdot \frac{2}{n(n-1)} \cdot \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{i,j}$$

$$\mu_{i,j} = \begin{cases} 1, & if \quad (E(D_i) > E(D_j) \wedge M(D_i) > M(D_j)) \vee \\ & \quad\;\; (E(D_i) < E(D_j) \wedge M(D_i) < M(D_j)) \vee \\ & \quad\;\; (E(D_i) = E(D_j) \wedge M(D_i) = M(D_j)) \\ 0, & else \end{cases}$$

Example


estimated
measured
metric
fidelity = 100 %
metric
fidelity = 33.3 %
A
implementation
B

Fidelity (II)

- very important to visualize trends
  - is the direction of development correct?

- problem
  - no quantification
  - mapping to *‘0’* or *‘1’*
  - so the fidelity is high even if a big increase of the estimated metric leads only to a small increase of the measured metrics

Quality Metrics

- performance metrics
  - hardware
    - clock period, latency, execution time, throughput
  - software
    - execution time
  - communication
    - maximal or average bit rate
- costs
- other metrics
  - power consumption
  - testability
  - reusability
  - …

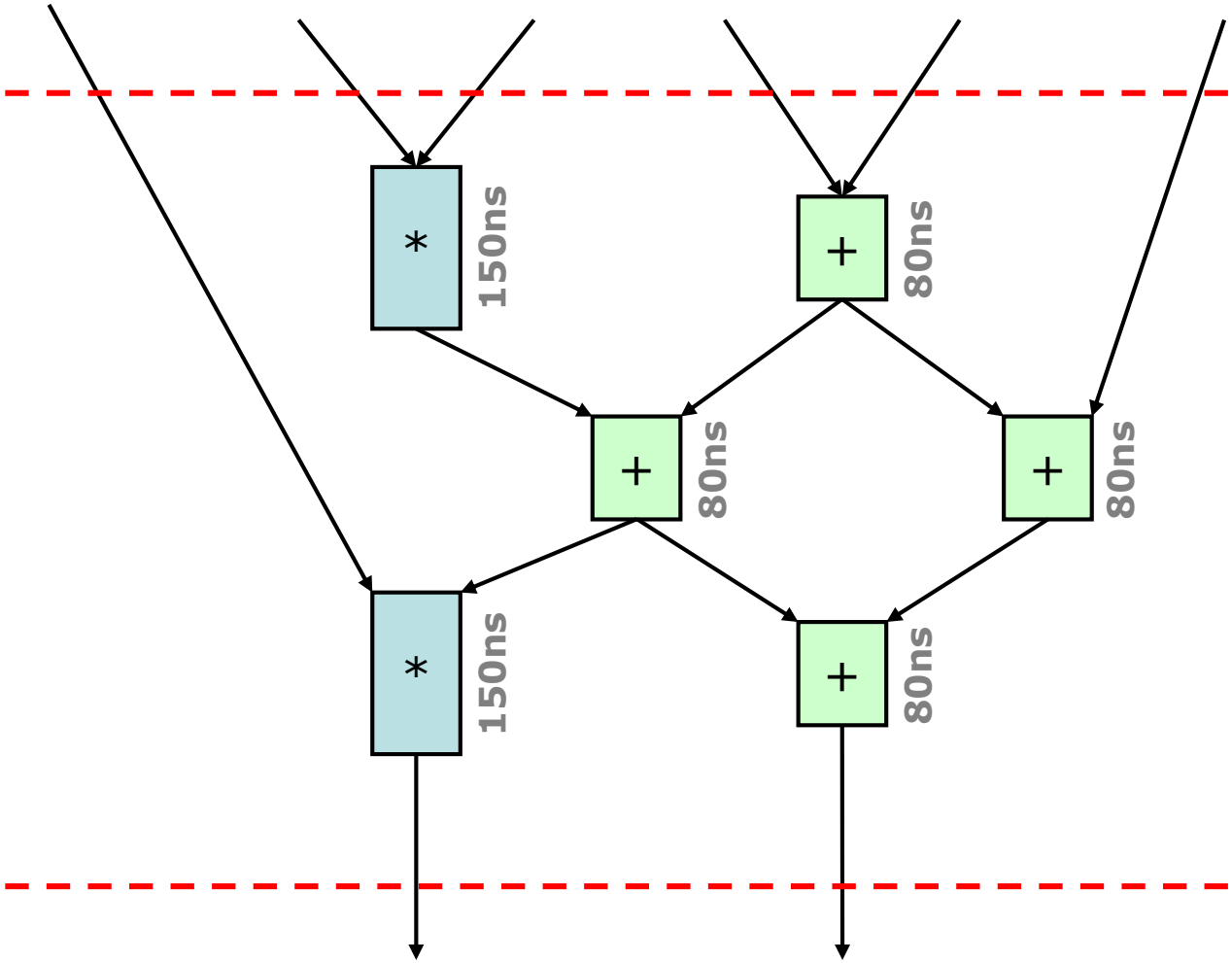
Contents

- Parameters of Estimation Methods
- Estimation of Hardware Metrics
- Estimation of Software Metrics
- Estimation of Communication Metrics

HW Performance

- clock period T
  - depends on technology resources
- _____
  - given by the number of clock steps
- execution time T_{ex}
  - $T_{ex} = T * L$
- throughput
  - $R=1/T_{ex}$ or $R=P/T_{ex}$ in case of pipelining with depth of P

Example (I)



clock period

$T = 380ns$

latency

$L = 1$

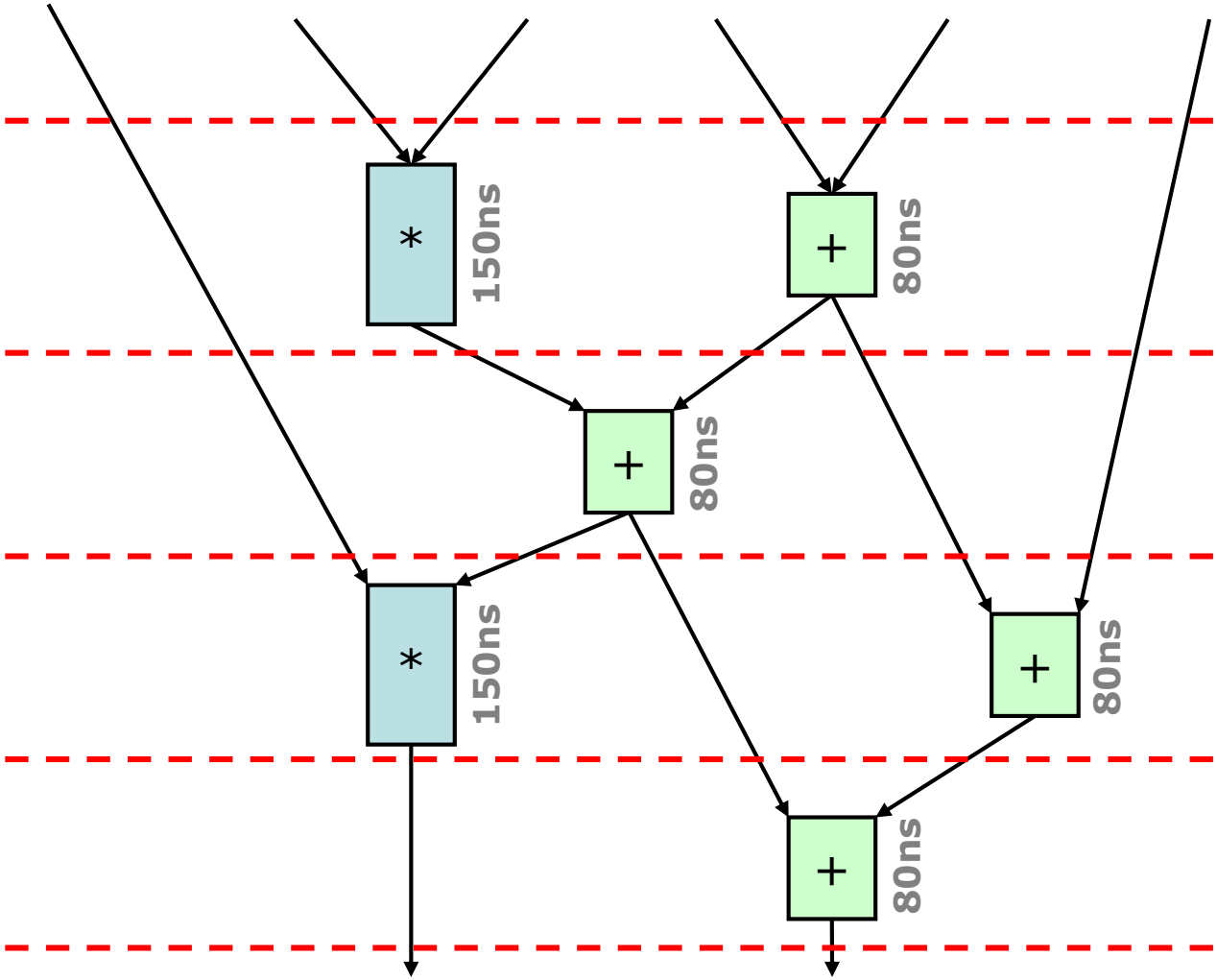
ex. time

$T_{ex} = 380ns$

resources

2 MUL, 4 ADD

Example (II)



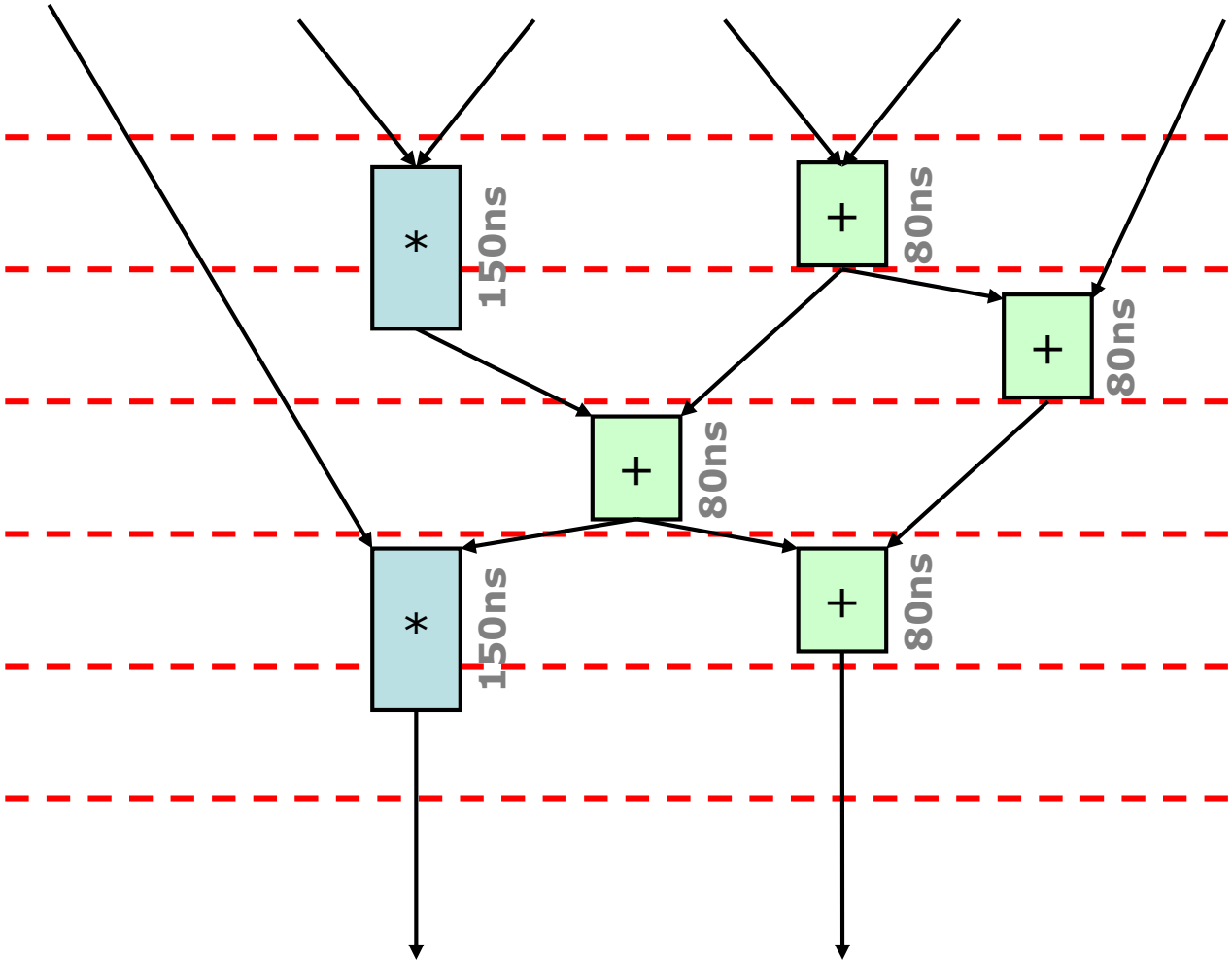
clock period
$T = 150ns$

latency
$L = 4$

ex. time
$T_{ex} = 600ns$

resources
1 MUL, 1 ADD

Example (III)



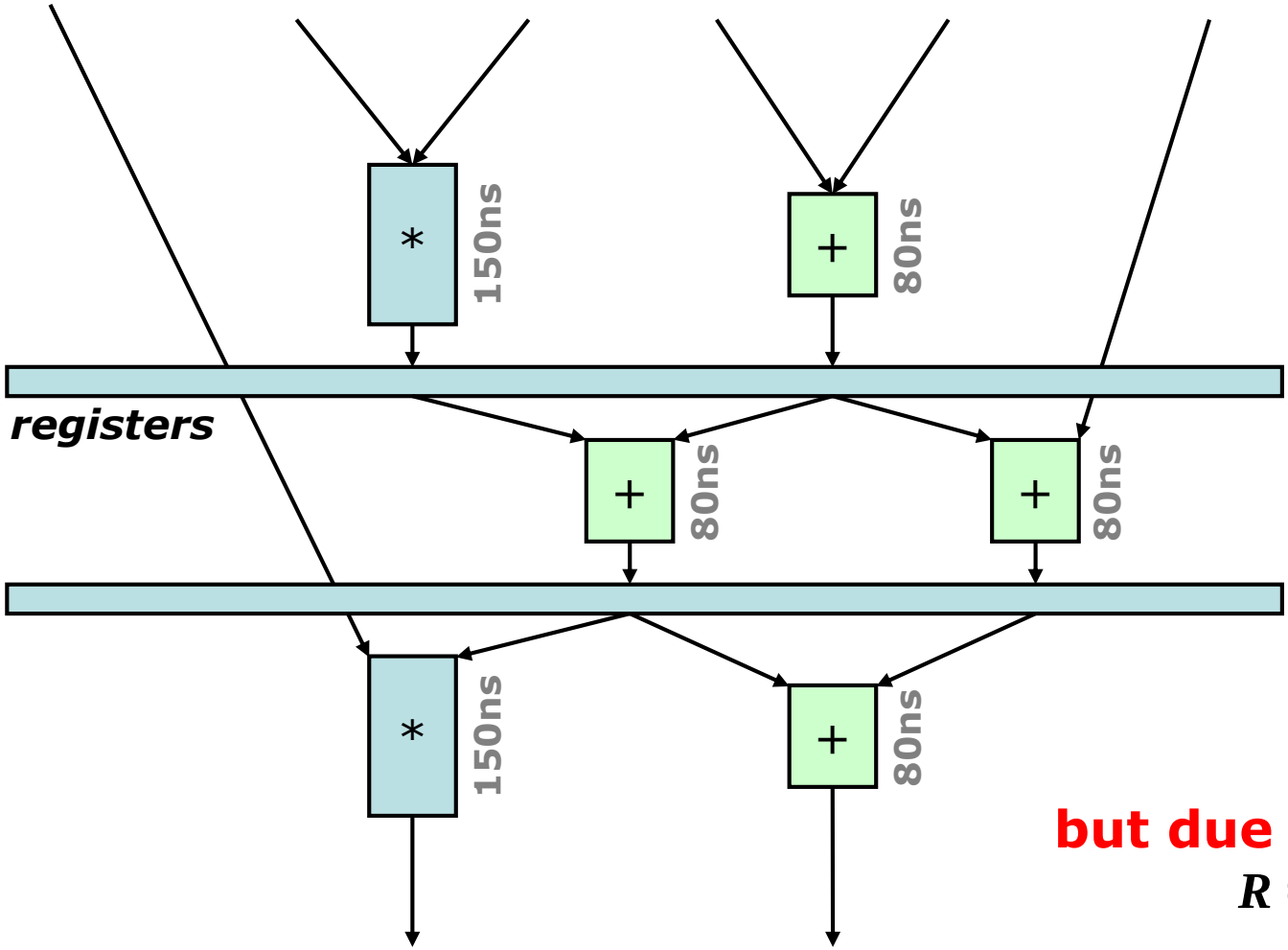
clock period
$T = 80ns$

latency
$L = 5$

ex. time
$T_{ex} = 400ns$

resources
1 MUL, 1 ADD

Pipelining



clock period

$T = 150ns$

latency

$L = 3$

ex. time

$T_{ex} = 450ns$

resources

2 MUL, 4 ADD

but due to pipelining:

$R = 3 * 1/T_{ex}$

Estimation of Maximal Clock

- functional units (operators) v_k with delays $del(v_k)$
  → **how to find "good" clock period T ?**

- method of the maximum operator delay

$$T_{min} = \max_k(del(v_k)) \qquad f_{max} = \frac{1}{T_{min}}$$

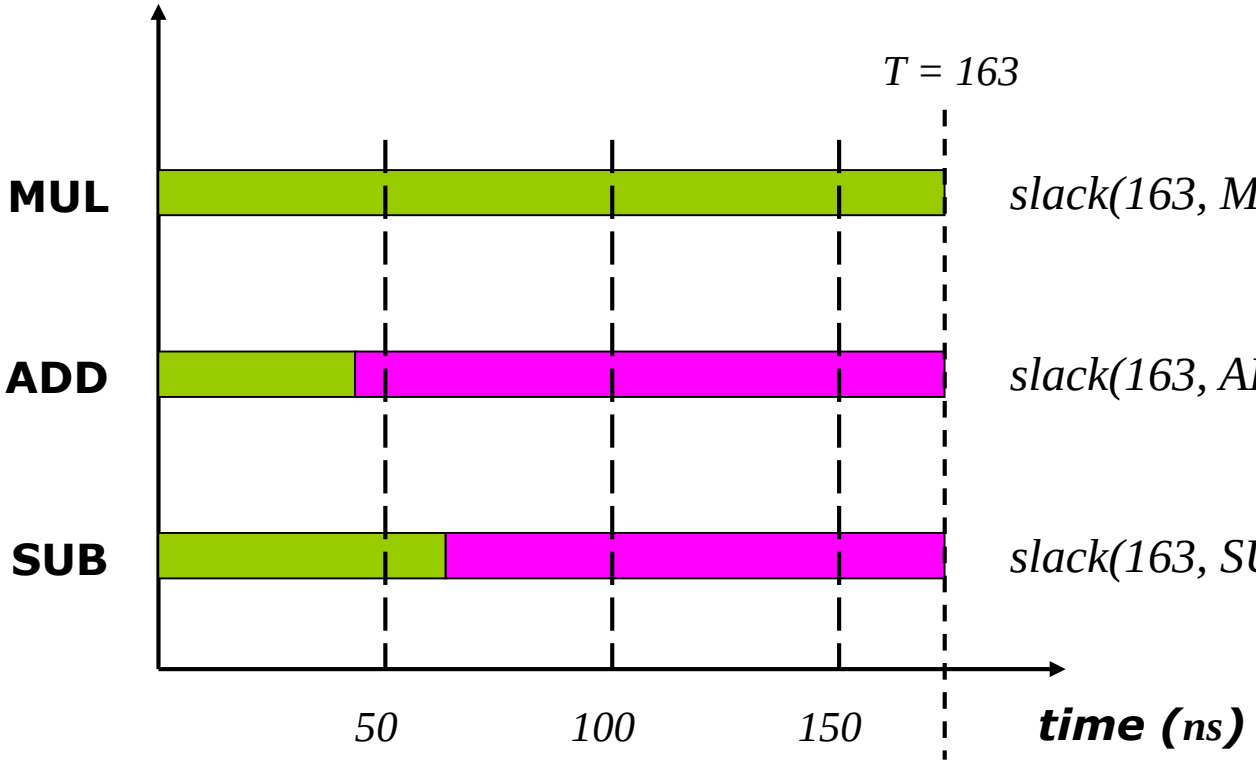
→ bad estimation

- method of clock slack minimisation
  - search in the interval $[T_{min} \ldots T_{max}]$ for the clock period T that maximises the utilisation (and minimises clock slack)

- Finite State Machine + Data Path model

- ILP search

Clock Slack



$$slack(T, v_k) = \left\lceil \frac{del(v_k)}{T} \right\rceil \cdot T - del(v_k)$$

busy unit

slack

Clock Slack Minimisation

- Let $occ(v_k)$ be the number of operations of type k and $|V_T|$ the number of different operation types. So the average clock slack is given by

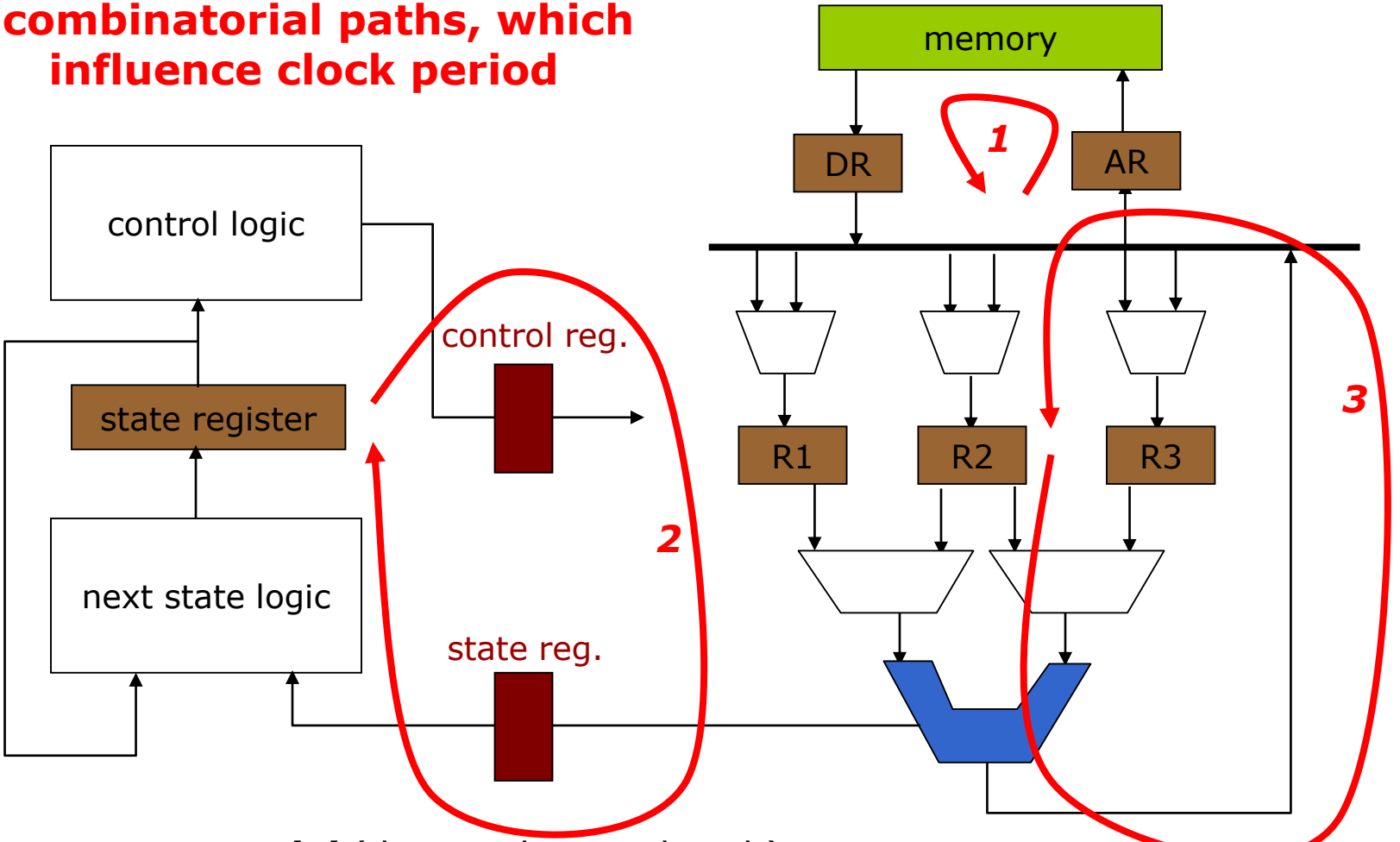
$$avgslack(T) = \frac{\sum_{k=1}^{|V_T|} (occ(v_k) \cdot slack(T, v_k))}{\sum_{k=1}^{|V_T|} occ(v_k)}$$

- The clock utilisation is given by

$$util(T) = 1 - \frac{avgslack(T)}{T}$$

Finite State Machine + Data Path (I)

3 combinatorial paths, which influence clock period



processor model (data and control path)

Finite State Machine + Data Path (II)

- clock period has to be bigger than the highest delay in the influencing combinatorial paths

$$T \geq del(SR) + del(CL) + del(RF) + del(Mux) + del(FU) + del(NS) + setup(SR) + \sum_i del(n_i)$$

SR ...	state register	FU ...	functional units
CL ...	control logic	NS ...	next state logic
RF ...	register file	$setup$ ...	setup time
Mux ...	multiplexer	n_i ...	wires

- very complex

Cost Metrics (HW)

- metrics proportional to silicon area
  - chip area in mm²
  - number of transistors, number of gates
  - number of logic blocks (FPGA)
- package
- number of I/O pins

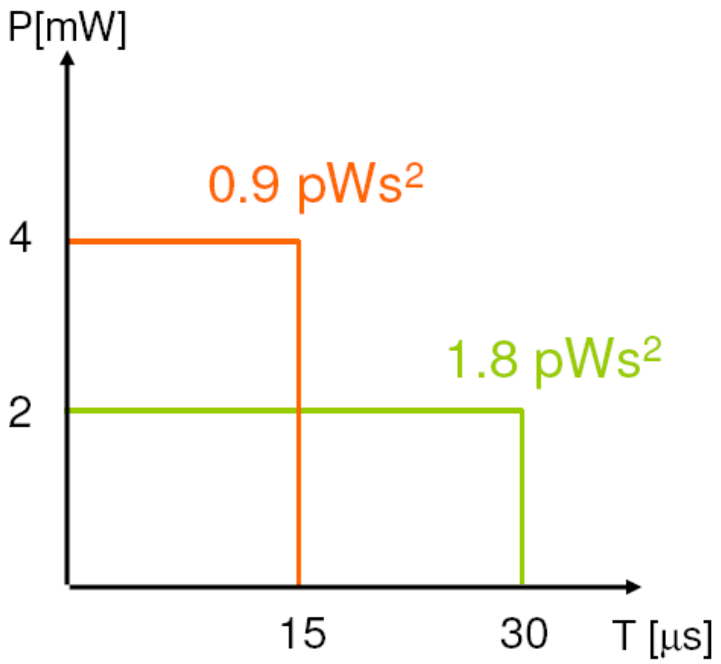
Energy Consumption (I)

- power dissipation
  - P in $[W]$
  - important for dimensioning of packaging, power supply and cooling

- energy
  - $E = P_{avg} * T_{execution}$ in $[Ws]$
  - important for mobile devices (battery life time)
  - metrics for systems that operates at a fixed rate
  - power normalised to clock period in $[\mu W/MHz]$
  - for processor also available
    - $[\mu W/MIPS]$ or $[MIPS/\mu W]$
    - $[\mu W/SPEC]$ or $[SPEC/\mu W]$

Energy Consumption (II)

- _____
  - $EDP = E * T_{exe}$ in $[Ws^2]$
  - metrics for systems that operates at maximum rates



  - for processors also:
    - $[MIPS^2/\mu W]$
    - $[SPEC^2/\mu W]$

Example: Mobile Processors

Processor (Vendor)	Techn. [µ]	V_{DD} [V]	Clock [MHz]	Power [mW]	MIPS	MIPS/W	$MIPS^2/mW$
StrongARM (Intel)	0.35	2	230	360	268	744	200
ARM710 (VLSI)	0.8	3.3	25	120	30	250	8
ARM940T (VLSI)	0.35	3.3	150	675	(e)160	(e)237	(e)38
MMC2001 (Motorola)	0.35	2	34	80	31	387	12
TR4102 (LSI)	0.25	1.8	80	40	(e)90	(e)2250	(e)203
SH7708 (Hitachi)	0.5	3.3	25	95	25	263	7
SH7750 (Hitachi)	0.25	1.8	200	1600	300	188	56

MIPS ratings from Dhrystone Benchmark

(e)… estimated

Example: VLIW Processors

Processor (Vendor)	Techn. [µ]	V_{DD} [V]	Clock [MHz]	Power [mW]	MIPS	MIPS/W	$MIPS^2$/mW
'C6201 (Texas Instr.)	0.25	2.5	200	4600	1600	348	557
SC140 (Mot. /Lucent)	0.13	1.5	300	500	3000	6000	18000
TM1000 (Philips)	0.35	3.3	100	4000	2500	625	1563
Merced (HP/Intel)	0.18	?	800	(e)70000	6400	(e)91	(e)585

MIPS ratings are peak number

(e)… estimated

Contents

- Parameters of Estimation Methods
- Estimation of Hardware Metrics
- Estimation of Software Metrics
- Estimation of Communication Metrics

SW Performance (I)

- execution time T

$$T = I_C \cdot CPI \cdot \tau = \frac{I_C \cdot CPI}{f}$$

  - I_C … instruction count for a given program
  - CPI … cycles per instruction (averaged value)
  - τ … _____
  - f … _____

- estimation on basis of source-/ intermediate-/ target code

- example
  - $I_C = 2000, CPI = 0.4, f = 400\ MHz \rightarrow T = 2\mu s$

- "worst-case execution time" important for real-time systems

SW Performance (II)

- MIPS rate (million instructions per second)

$$MIPS = \frac{I_C}{T \cdot 10^6} = \frac{f}{CPI \cdot 10^6}$$

- MFLOPS (million floating-point operations per second)
  - parallel operations considered
  - optimal utilisation assumed
- MACS (million multiply & accumulates per second)
  - important for DSPs, see HW/SW Codesign I
- MOPS (million operations per second)
  - counts alls possible operations (ALU, address calculation, DMA, …)
  - optimal utilisation assumed

Example (I)

- processor with $f = 500\ MHz$
- 3 instruction classes A, B, C with $CPI_A = 1, CPI_B = 2, CPI_C = 3$
- 2 different compilers generate (for the same program) following instruction mixes (in 10^9 instructions)

	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

Example (II)

- clock cycles

Compiler 1	$(5 * 1 + 1 * 2 + 1 * 3) * 10^9 = 10 * 10^9$
Compiler 2	$(10 * 1 + 1 * 2 + 1 * 3) * 10^9 = 15 * 10^9$

- execution times

Compiler 1	$(10 * 10^9) / (500 * 10^6) = 20s$
Compiler 2	$(15 * 10^9) / (500 * 10^6) = 30s$

- MIPS

Compiler 1	$((5 + 1 + 1) \times 10^9) / (20 * 10^6) = 350$ MIPS
Compiler 2	$((10 + 1 + 1) \times 10^9) / (30 * 10^6) = 400$ MIPS

program of compiler 2 has a higher MIPS rate,
but program of compiler 1 runs faster

Worst-Case Execution Time (WCET)

- WCET cannot be determined by _____
- _____ with program analysis techniques
  - program path analysis
    - which sequence of instructions is executed in the worst-case (longest runtime)
    - problem: the number of possible program paths grows exponentially with the program length
  - modelling of the target architecture
    - computation of the estimated WCET for a specific processor model
    - problem: modelling/ computation of compiler optimisations and dynamic effects like pipelining or caches

Program Path Analysis

- the estimated WCET
  - always has to be higher the than the actual WCET
  - good estimation method closely approximates the actual WCET

- assumptions for processor model
  - one processor
  - no interrupts
  - no operating system

- assumptions for programming model
  - no recursive function calls (direct or indirect)
  - no pointer operations
  - loops have to be bounded

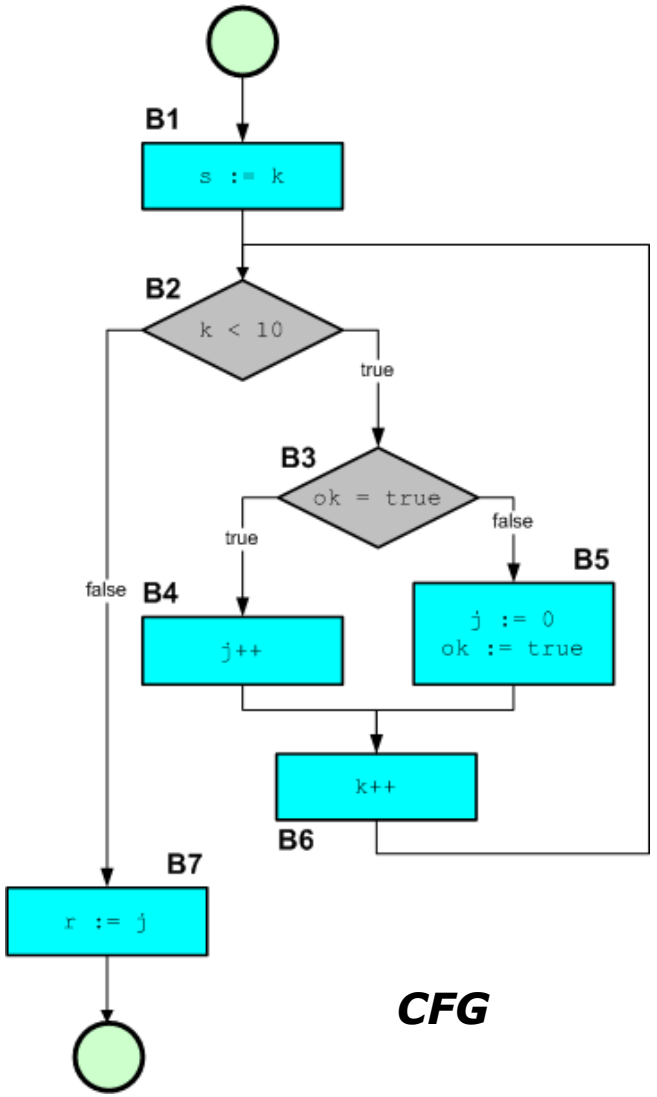
Example

```
/* k >= 0

s = k;
while (k < 10)
{
   if (ok) j++;
   else
   {
      j = 0;
      ok = true;
   }

   k++;
}
r = j;
```

program



CFG

Calculation of WCET

- **definition**

  A program consists of N basic blocks, where each basic block B_i has a $WCET$ c_i and is executed for exactly x_i times. Then, the $WCET$ is given by

---

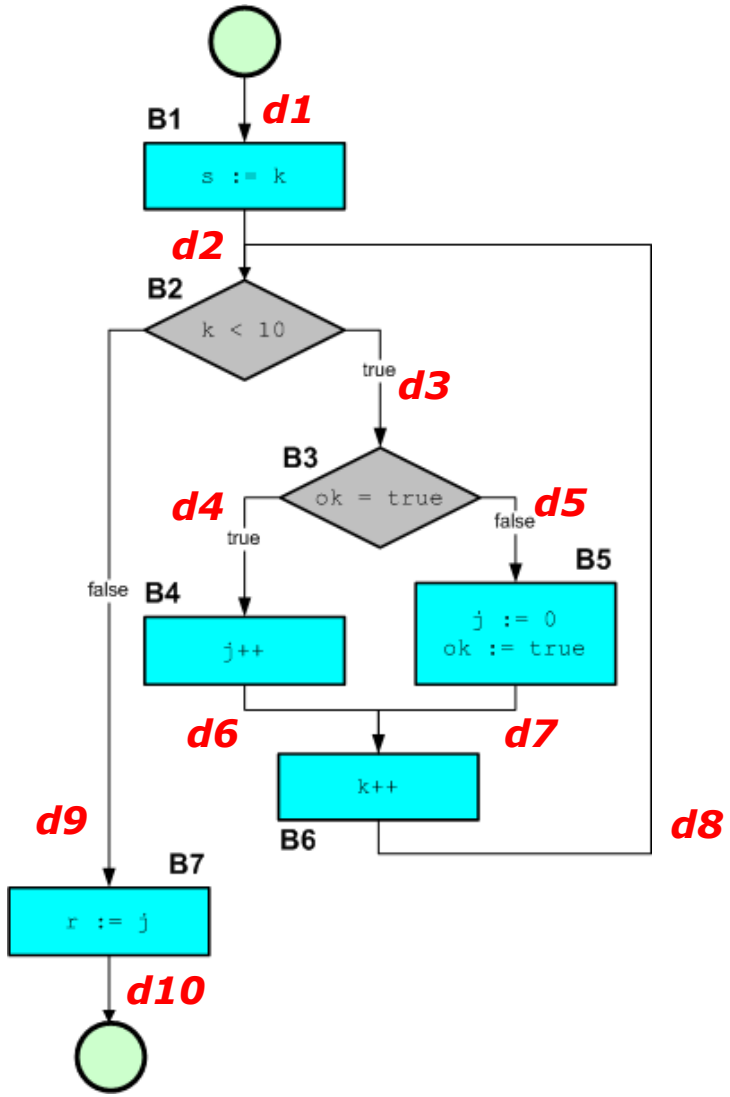
- c_i can easily be estimated, because the sequence of executed instruction is known (by definition of basic block)

- how to determine x_i?
  - structural constraints given by program structure
  - functional constraints provided by the programmer

Example – Structural Constraints

flow equations:

$$d1 = d2 = x_1$$
$$d2 + d8 = d3 + d9 = x_2$$
$$d3 = d4 + d5 = x_3$$
$$d4 = d6 = x_4$$
$$d5 = d7 = x_5$$
$$d6 + d7 = d8 = x_6$$
$$d9 = d10 = x_7$$



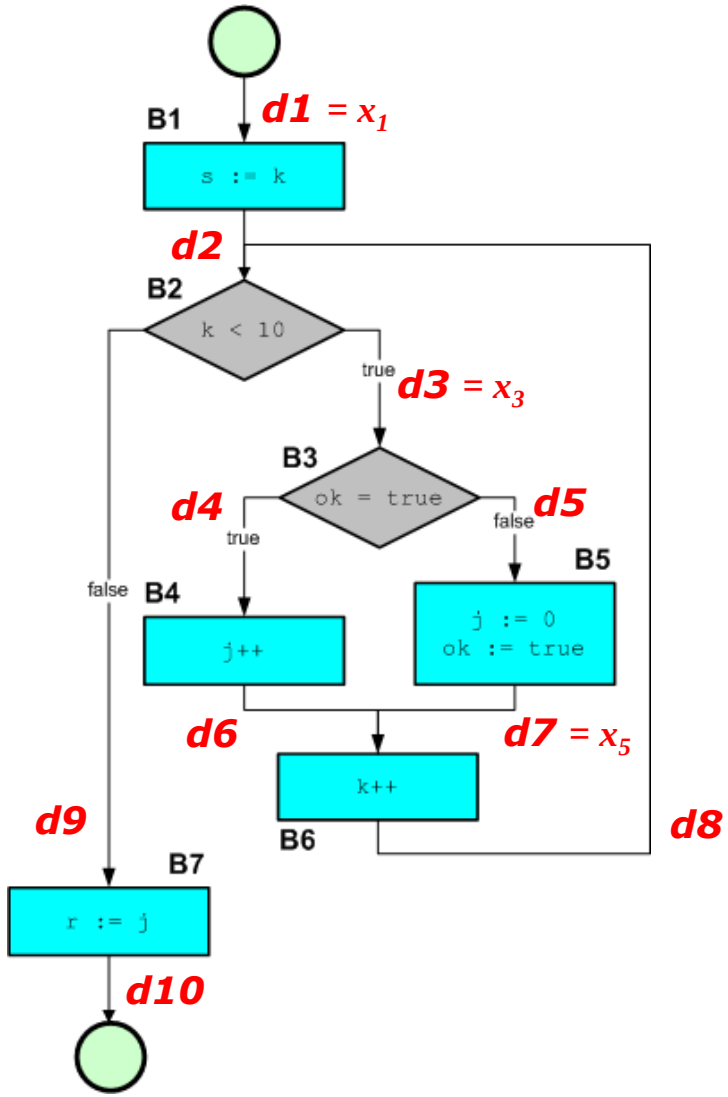
Example – Functional Constraints (I)

loop is executed for at most 10 time:

$$x_3 \leq 10 * x_1$$

B5 is executed for at most one time:

$$x_5 \leq x_1$$



Functional Constraints

- functional constraints are provided by the programmer
  - bounds for loop counters, …
  - based on knowledge of the program context

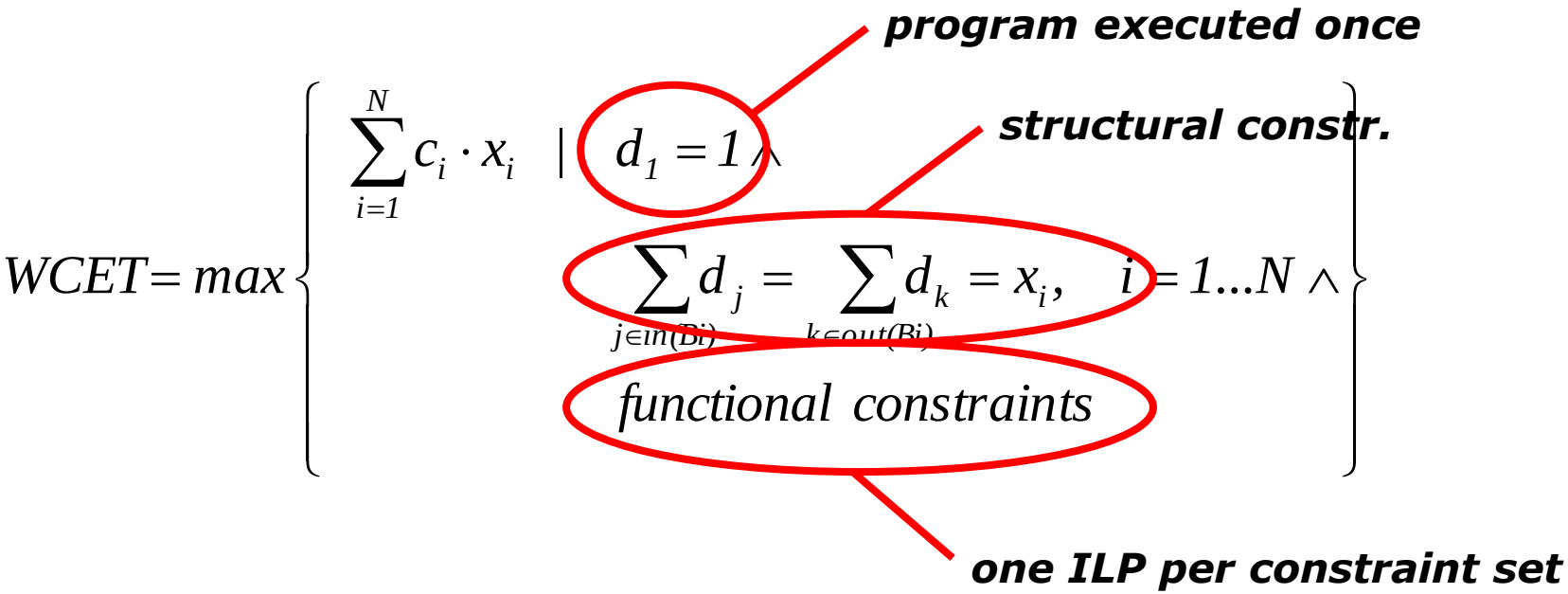
- functional constraints can be quite complex
  - e.g. "if the else-branch in the loop is executed, then the loop is executed exactly 5 times"

$$(x_5 = 0) \;||\; (x_5 \geq 1) \;\&\&\; (x_3 = 5 * x_1)$$

- problem: if constraints are propagated manually errors are possible

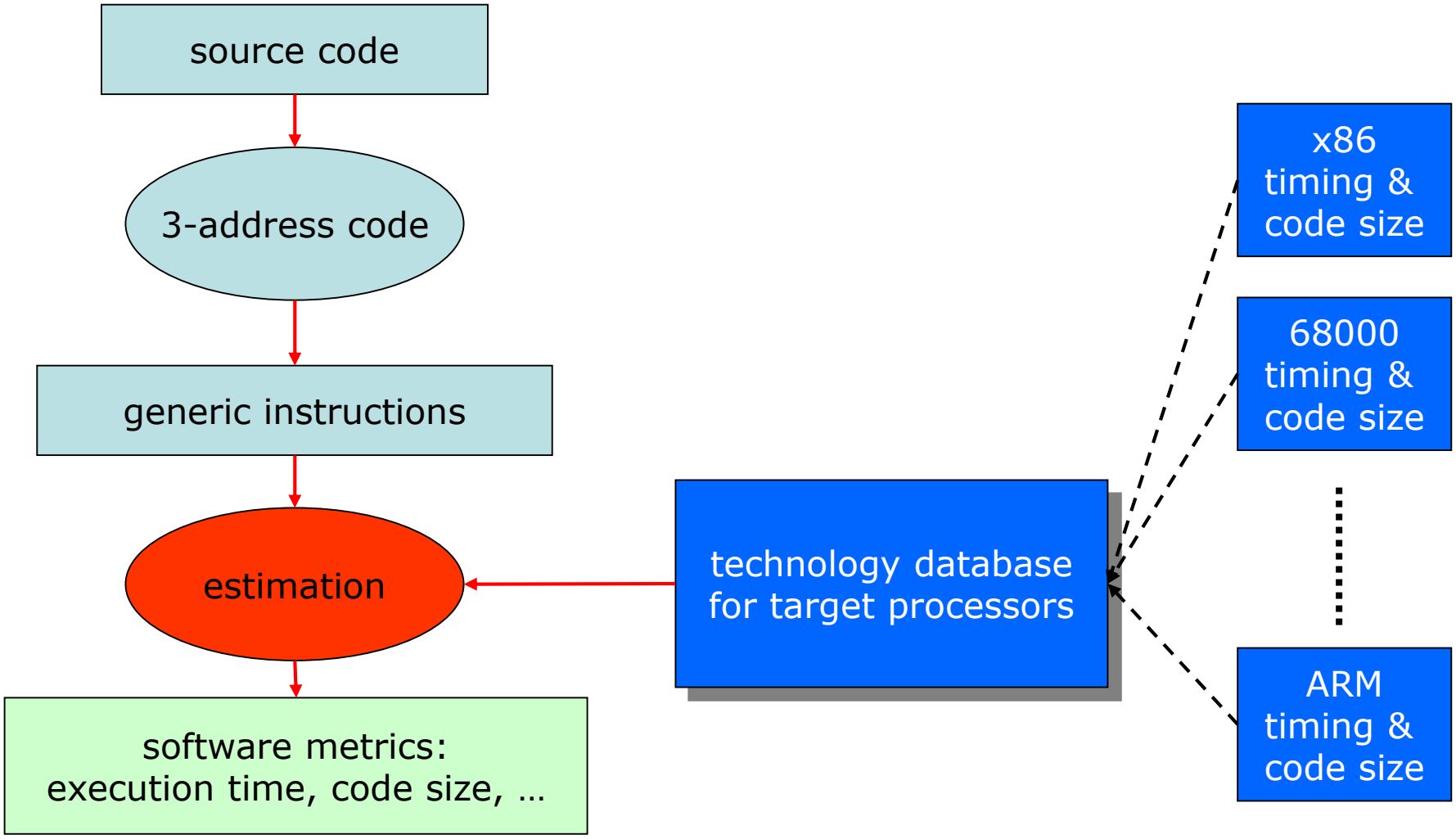
WCET ILP

- ILP including structural and functional constraints



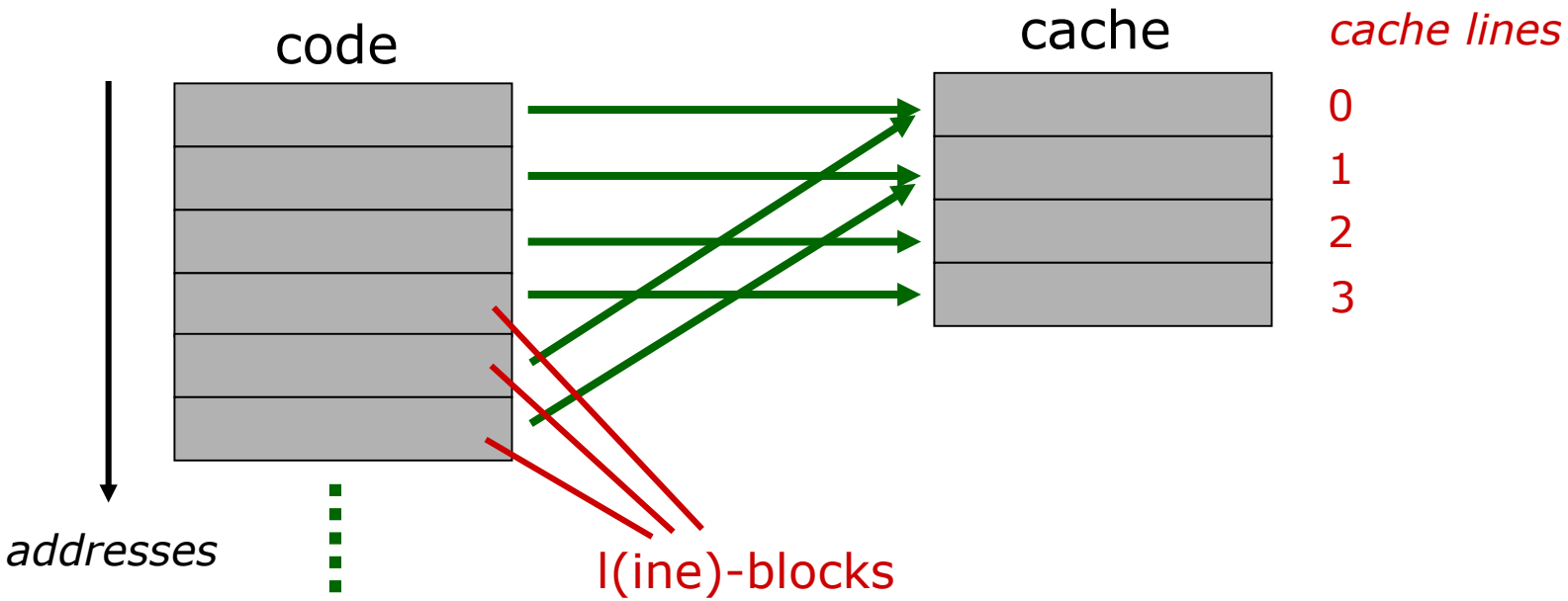
$$WCET = max\left\{ \sum_{i=1}^{N} c_i \cdot x_i \;\middle|\; d_1 = 1 \wedge \sum_{j \in in(B_i)} d_j = \sum_{k \in out(B_i)} d_k = x_i, \quad i = 1...N \wedge \text{functional constraints} \right\}$$

Target Architecture Modeling



Cache Modelling

- precondition: compilation into target code
- assumption: directly mapped instruction caches
- l-block = sequence of instructions stored in one cache line



WCET ILP – Cache Extension

- each basic block consists of n_i l-blocks
- an execution of an l-block results either in a cache hit or a cache miss

$$x_i = x_{i,j} = x_{i,j}^{hit} + x_{i,j}^{miss}$$

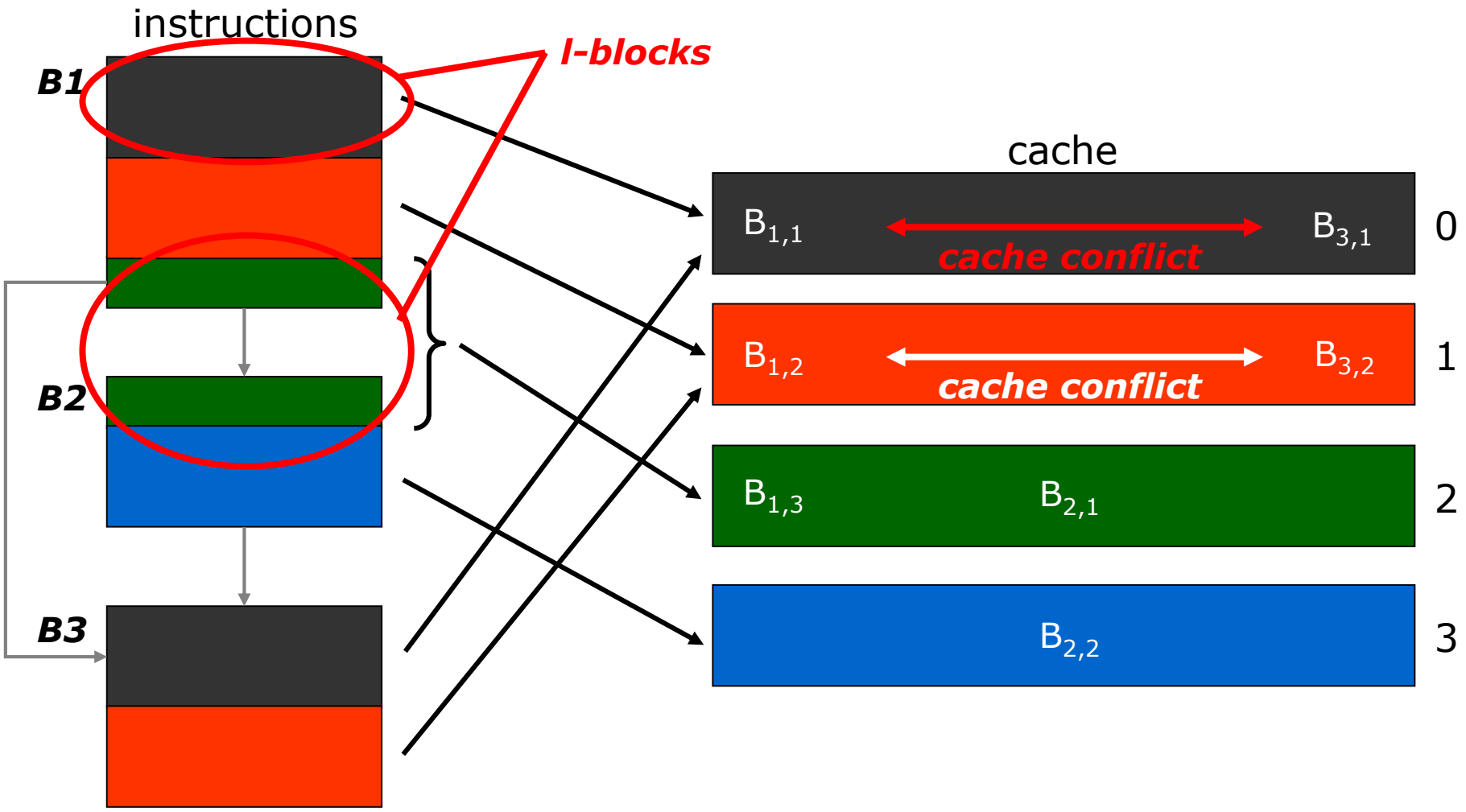
$x_{i,j}$ ... *number of execution of l-Block* $B_{i,j}$

- WCET

$$WCET = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \left(c_{i,j}^{hit} \cdot x_{i,j}^{hit} + c_{i,j}^{miss} \cdot x_{i,j}^{miss} \right)$$

$c_{i,j}$ ... *costs for hit or miss*

Example


instructions
I-blocks
B1
B2
B3
cache
B1,1
B3,1
cache conflict
0
B1,2
B3,2
cache conflict
1
B1,3
B2,1
2
B2,2
3

Cache Conflict Constraints

- **definition**

  Two l-blocks are in conflict, if the execution of one l-block leads to the replacement of the other l-block in the cache.

- example
  - $B_{2,2}$ is not in conflict with any other l-block:

  - $B_{1,3}$ and $B_{2,1}$ are not in conflict – a cache miss for one block automatically loads the other one to the cache:

$$x_{1,3}^{miss} + x_{2,1}^{miss} \leq 1$$

  - $B_{1,1}$ and $B_{3,1}$ are in conflict → ILP constraints necessary

Cost Metrics (SW)

- size of basic block i
  - *instr_size(j)* is the memory requirement for the generic instruction j

$$progsize_{B_i} = \sum_{j \in B_i} instr\_size(j)$$

- size of data memory
  - a program contains a set D of declarations
  - *data_size(d)* is the memory requirement for the declaration d

$$datasize = \sum_{d \in D} data\_size(d)$$

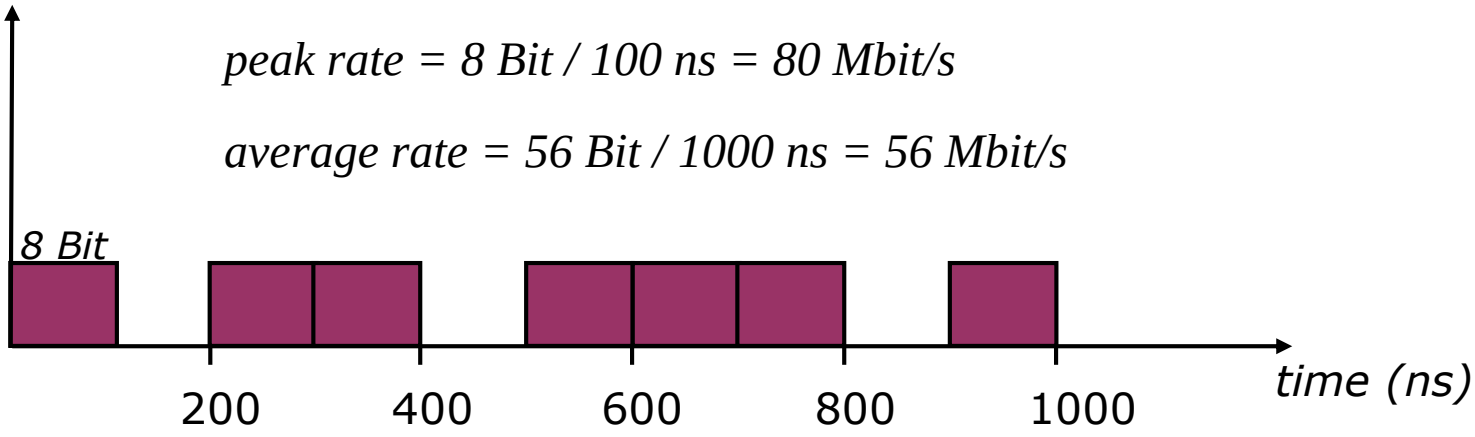
Contents

- Parameters of Estimation Methods
- Estimation of Hardware Metrics
- Estimation of Software Metrics
- Estimation of Communication Metrics

Communication Performance (I)

- processes transmit information via channels

- possible metric: how many information per time



Communication Performance (II)

- communication time T_{com}

T_{offset} ... time for initializing

messagesize ... in bit

bitrate ... in bit/sec