technische universität dortmund

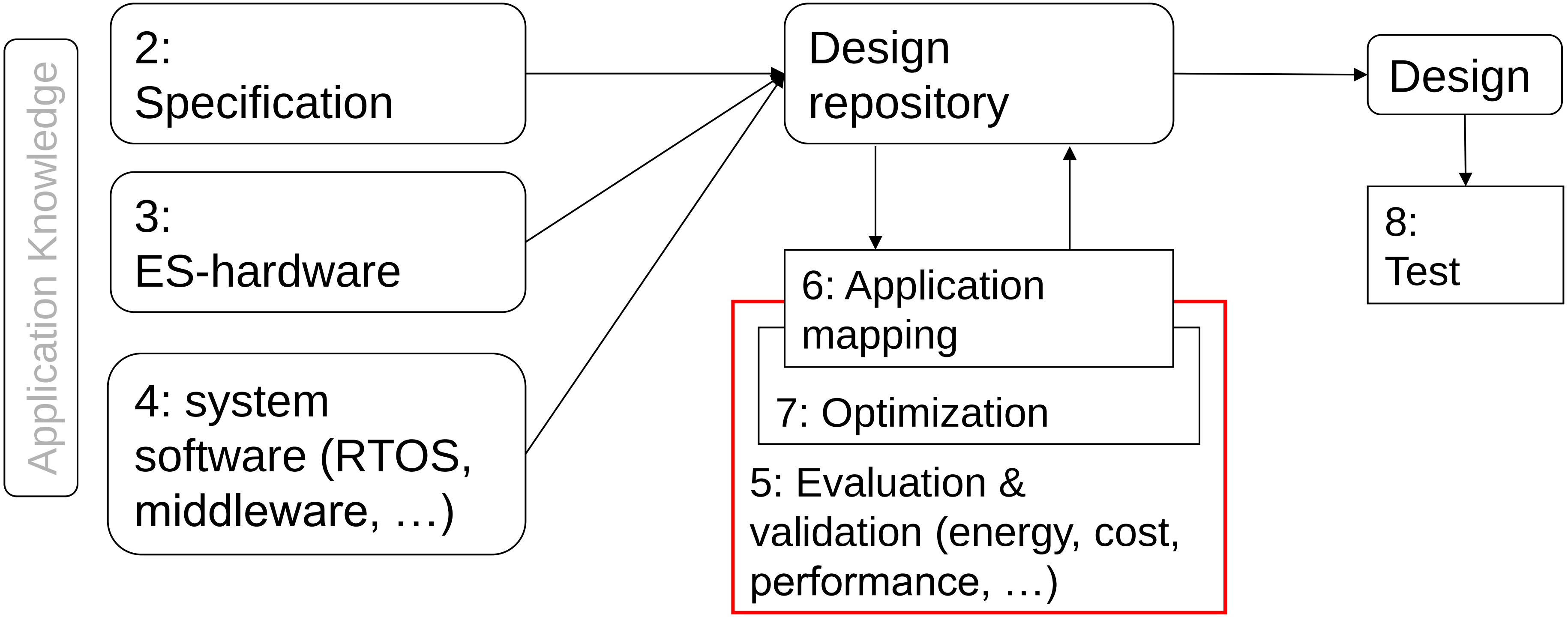
fakultät für informatik

Evaluation and Validation

Peter Marwedel
TU Dortmund, Informatik 12
Germany

2013年 12 月 02 日

Structure of this course



Numbers denote sequence of chapters

Validation and Evaluation

Definition: *Validation* is the process of checking whether or not a certain (possibly partial) design is appropriate for its purpose, meets all constraints and will perform as expected (yes/no decision).

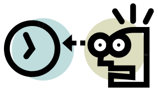
Definition: Validation with mathematical rigor is called *(formal) verification*.

Definition: *Evaluation* is the process of computing quantitative information of some key characteristics of a certain (possibly partial) design.

How to evaluate designs according to multiple criteria?

Many different criteria are relevant for evaluating designs:

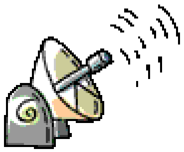
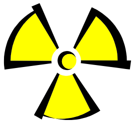
- Average & worst case delay
- power/energy consumption
- thermal behavior
- reliability, safety, security
- cost, size
- weight
- EMC characteristics
- radiation hardness, environmental friendliness, ..





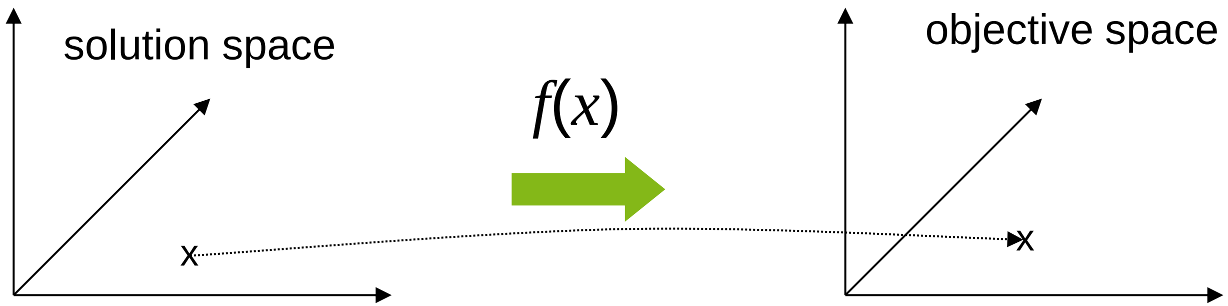



How to compare different designs?
(Some designs are "better" than others)

Definitions

- Let X: m-dimensional **solution space** for the design problem. Example: dimensions correspond to # of processors, size of memories, type and width of busses etc.
- Let F: n-dimensional **objective space** for the design problem. Example: dimensions correspond to average and worst case delay, power/energy consumption, size, weight, reliability, …
- Let $f(x)=(f_1(x),\ldots,f_n(x))$ where $x \in X$ be an **objective function.** We assume that we are using $f(x)$ for evaluating designs.



Pareto points

- We assume that, for each objective, an order < and the corresponding order ≤ are defined.
- **Definition**:
  Vector $u=(u_1,\ldots,u_n) \in F$ **dominates** vector $v=(v_1,\ldots,v_n) \in F$
  $\Leftrightarrow$
  u is "better" than v with respect to one objective and not worse than v with respect to all other objectives:

$$\forall i \in \{1,\ldots,n\} : u_i \leq v_i \wedge$$
$$\exists i \in \{1,\ldots,n\} : u_i < v_i$$

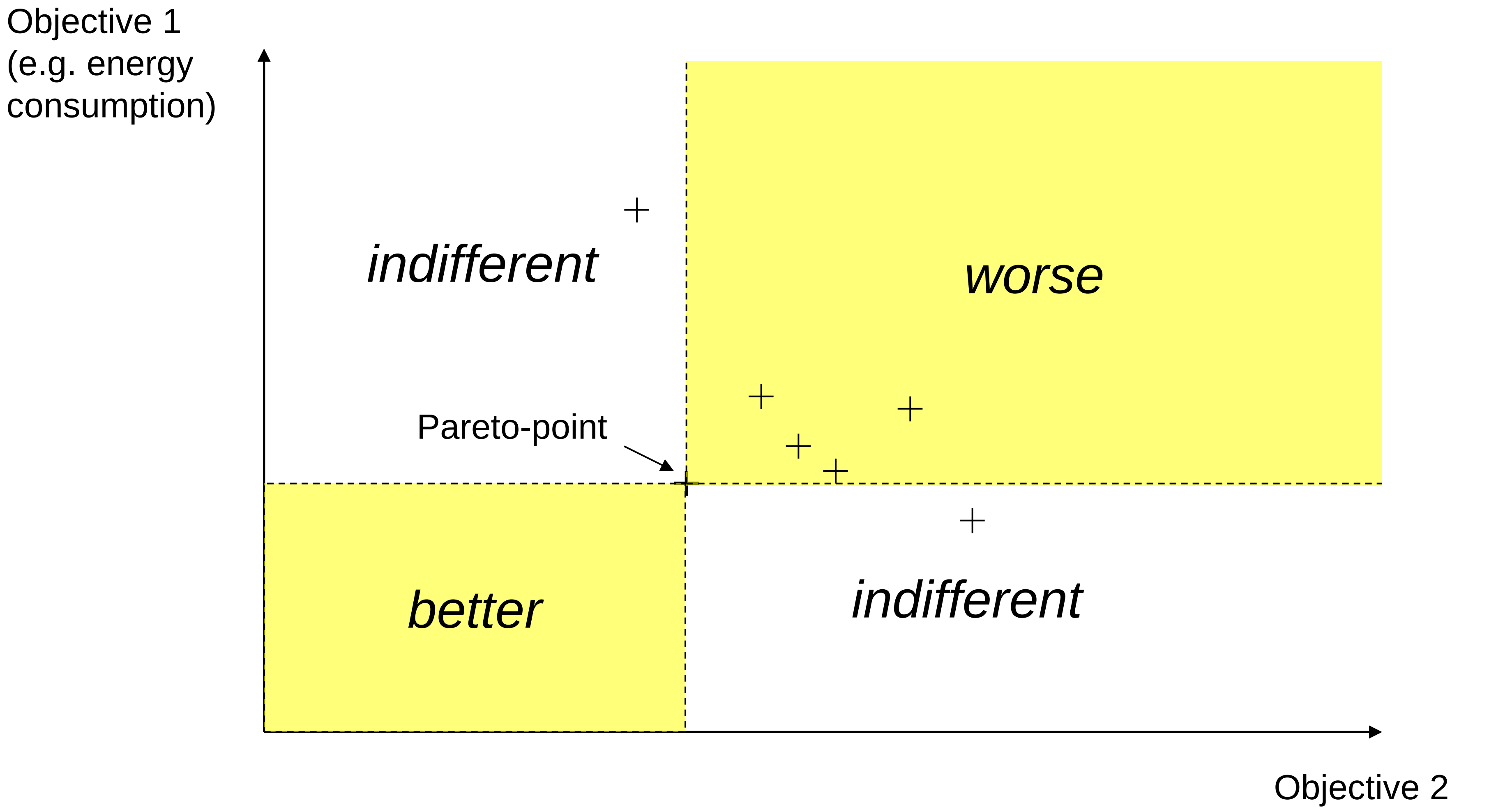
- **Definition:**
  Vector $u \in F$ is **indifferent** with respect to vector $v \in F$
  $\Leftrightarrow$ neither u dominates v nor v dominates u

Pareto points

- A solution $x \in X$ is called **Pareto-optimal** with respect to X $\Leftrightarrow$ there is no solution $y \in X$ such that $u=f(x)$ is dominated by $v=f(y)$. x is a **Pareto point**.
- **Definition**: Let $S \subseteq F$ be a subset of solutions.
  $v \in F$ is called a **non-dominated solution** with respect to S $\Leftrightarrow$ v is not dominated by any element $\in S$.
- v is called **Pareto-optimal**
  $\Leftrightarrow$ v is non-dominated with respect to all solutions F.
- A **Pareto-set** is the set of all Pareto-optimal solutions

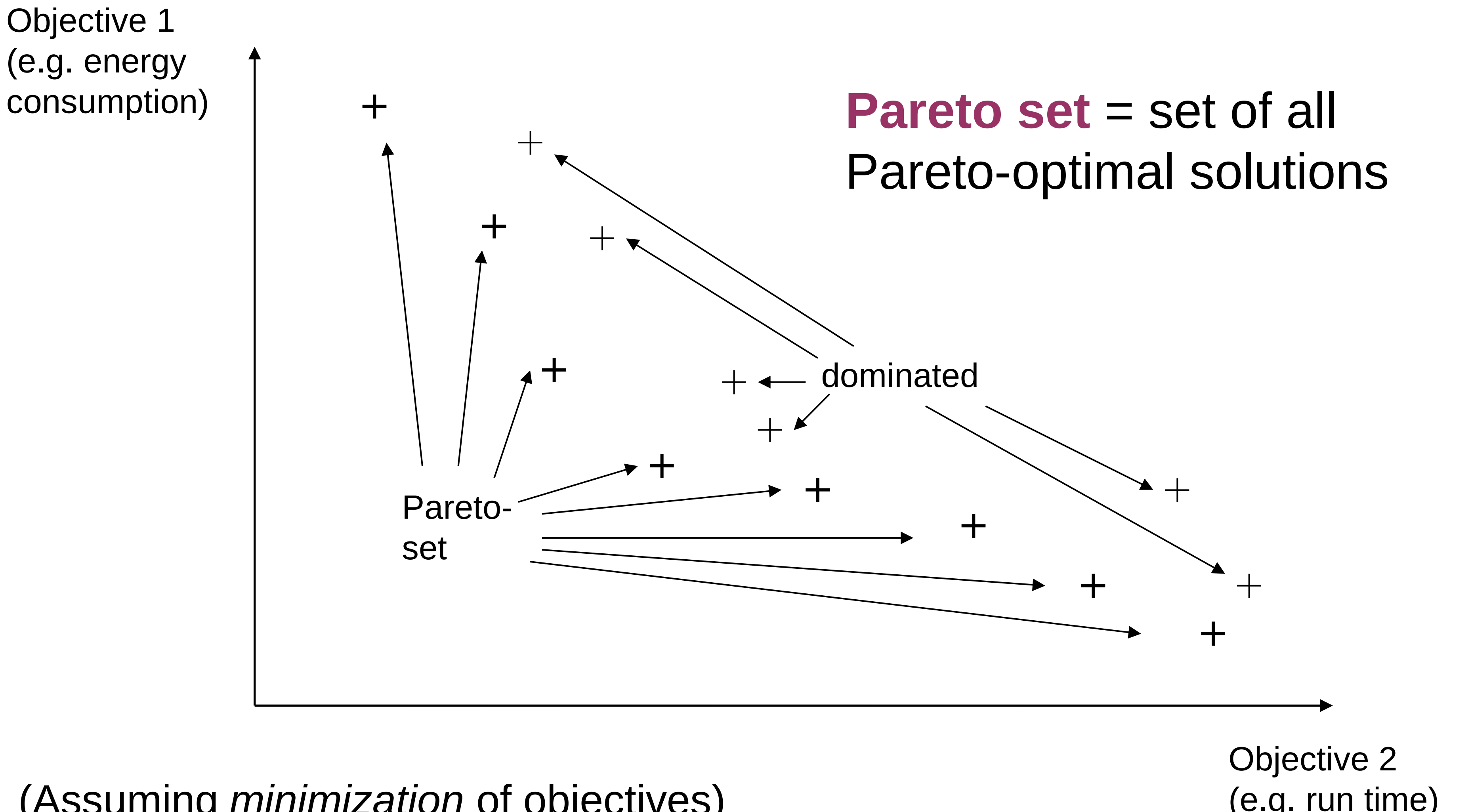
Pareto-sets define a **Pareto-front**
(boundary of dominated subspace)

Pareto Point



(Assuming *minimization* of objectives)

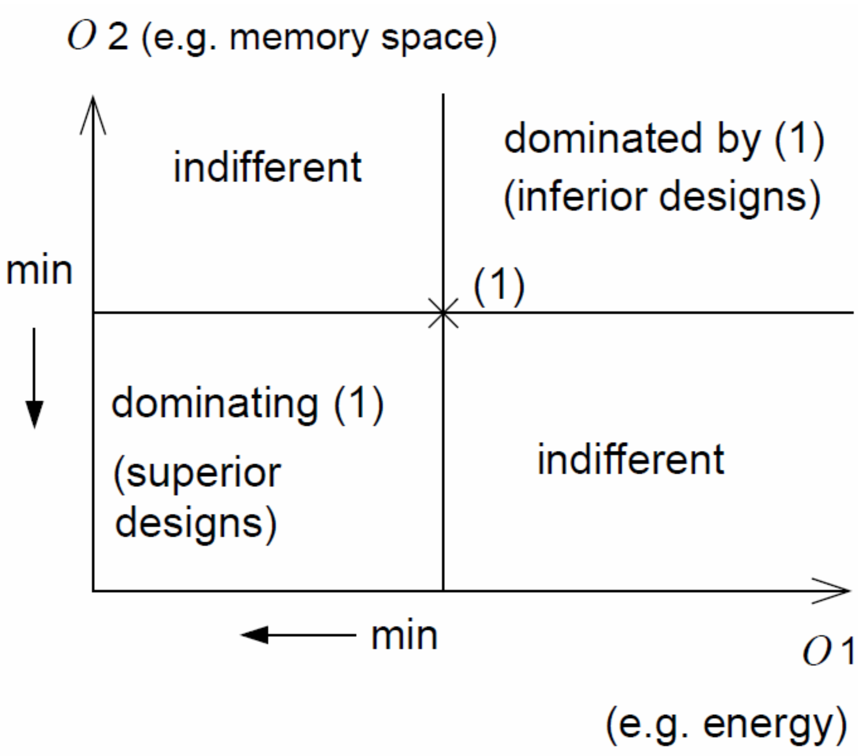
Pareto Set



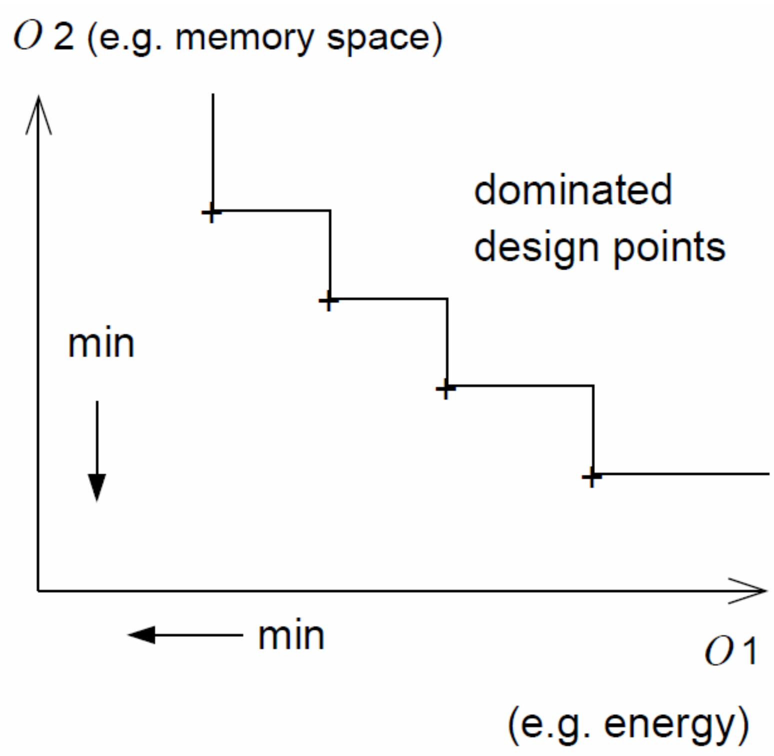
(Assuming *minimization* of objectives)

One more time …

Pareto point



Pareto front



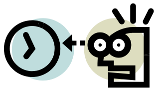
Design space evaluation

Design space evaluation (DSE) based on Pareto-points is the process of finding and returning a set of Pareto-optimal designs to the user, enabling the user to select the most appropriate design.

How to evaluate designs according to multiple criteria?

Many different criteria are relevant for evaluating designs:

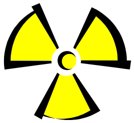
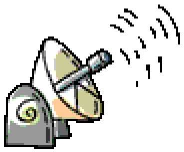
- Average & worst case delay
- power/energy consumption
- thermal behavior
- reliability, safety, security
- cost, size
- weight
- EMC characteristics
- radiation hardness, environmental friendliness, ..





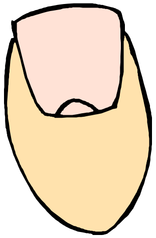



How to compare different designs?
(Some designs are “better” than others)

Average delays (execution times)

- **Estimated** average execution times **:**
  Difficult to generate sufficiently precise estimates;
  Balance between run-time and precision

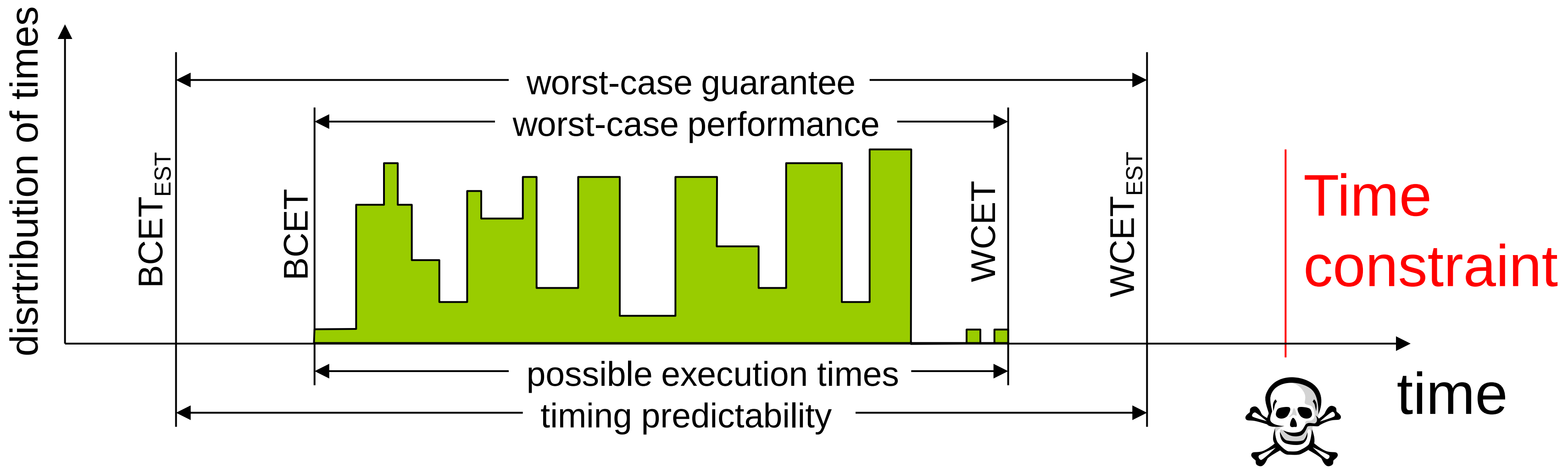
π x 

- **Accurate** average execution times**:**
  As precise as the input data is.

We need to compute **average** and **worst case** execution times

Worst case execution time (1)

Definition of worst case execution time:



$WCET_{EST}$ must be

1. safe (i.e. ≥ WCET) and
2. tight ($WCET_{EST}$-WCET≪$WCET_{EST}$)

Worst case execution times (2)

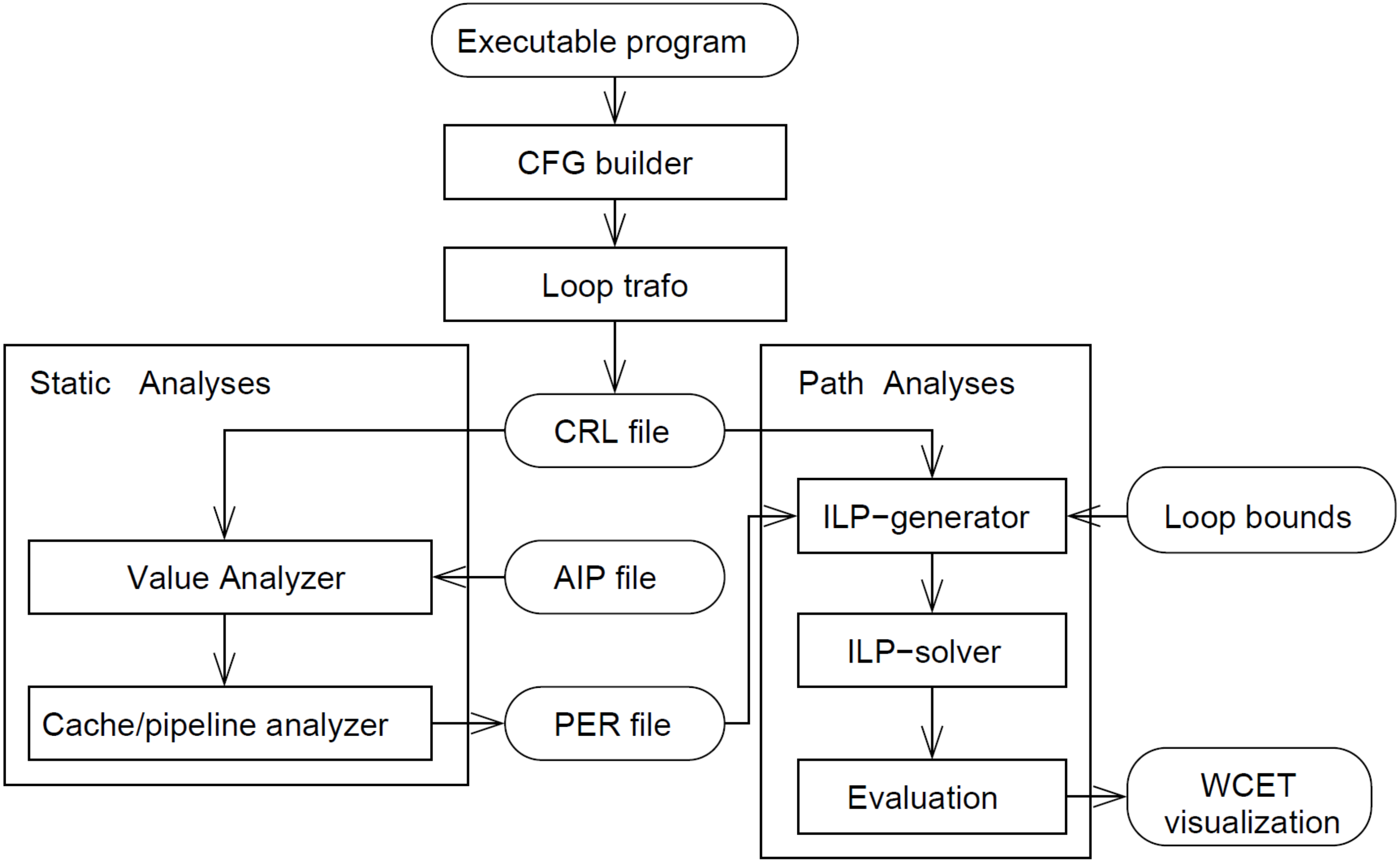
Complexity:

- in the general case: undecidable if a bound exists.
- for restricted programs: simple for “old“ architectures, very complex for new architectures with pipelines, caches, interrupts, virtual memory, etc.

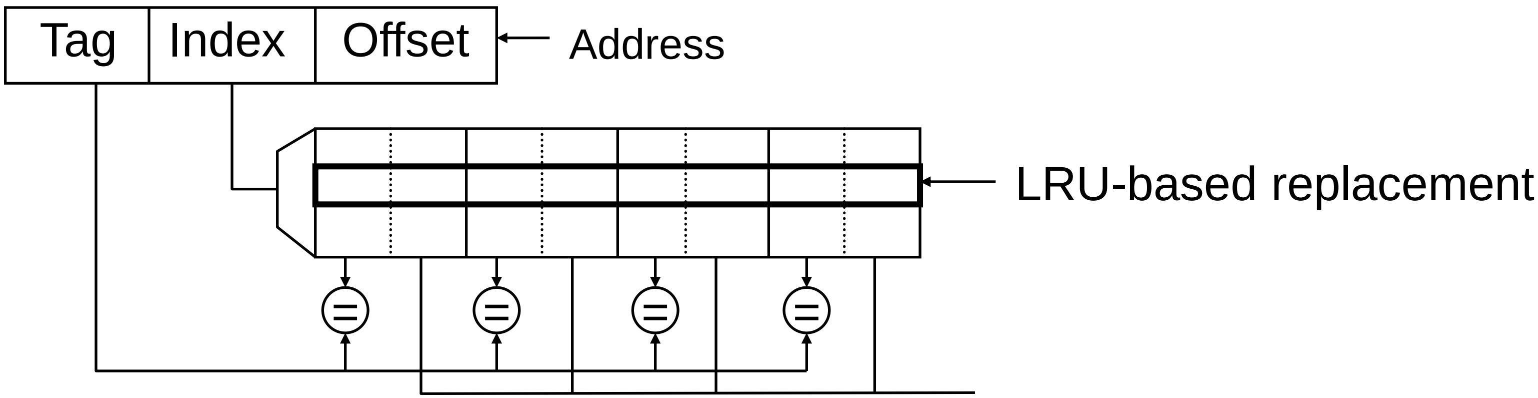
Approaches:

- for hardware: requires detailed timing behavior
- for software: requires availability of machine programs; complex analysis (see, e.g., www.absint.de)

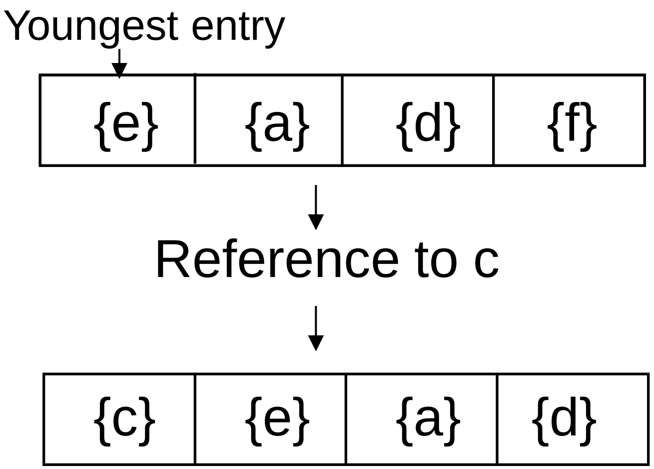
WCET estimation: AiT (AbsInt)



WCET estimation for caches



Old state

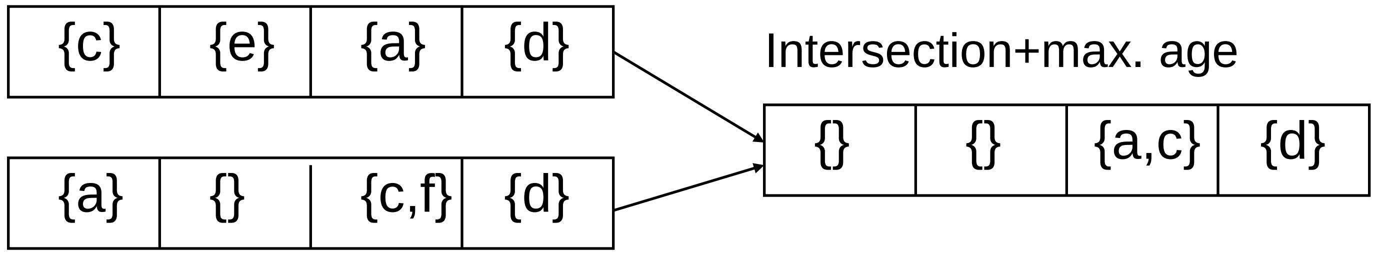


Variables getting older

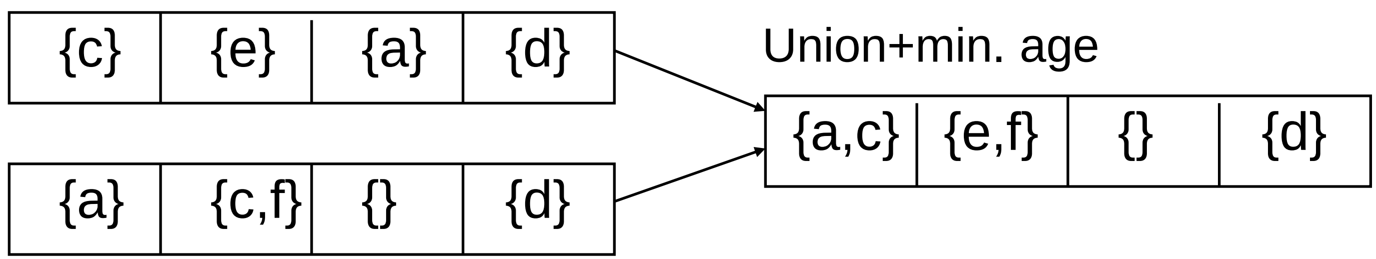
New state

Behavior at program joins

Worst case

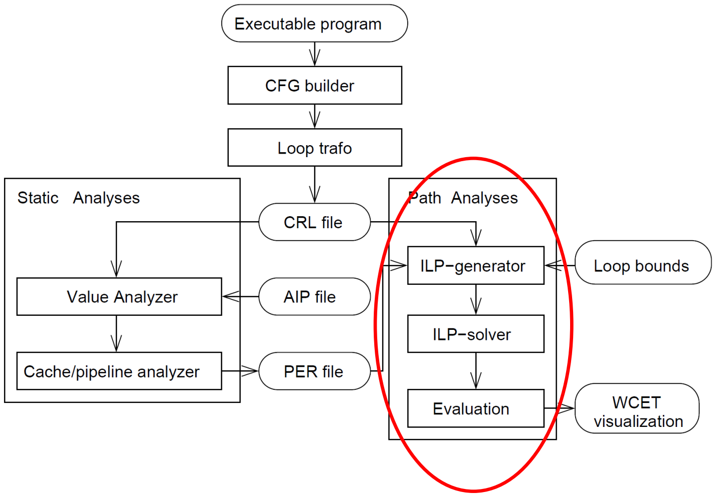


Best case



☞ Possibly several variables per entry

ILP model



- Objective function reflects execution time as a function of the execution time of blocks. To be **maximized**.
- Constraints reflect dependencies between blocks.
- Avoids explicit consideration of all paths

☞ Called **implicit path enumeration** technique.

Example (1)

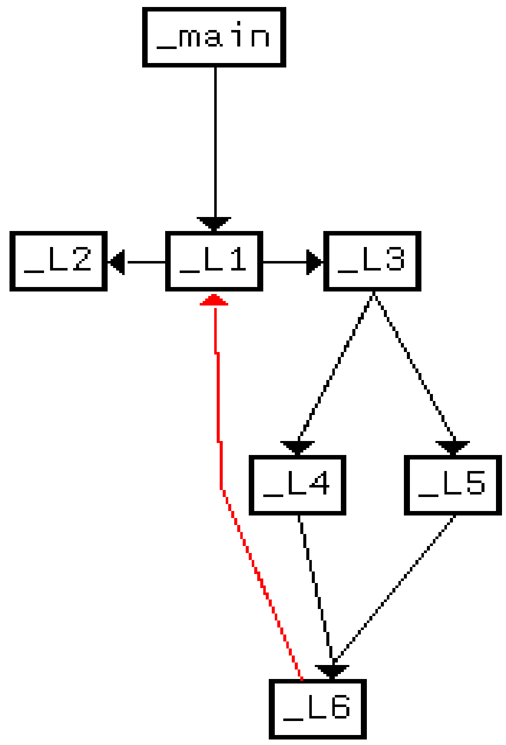
Program

```
int main()
{
  int i, j = 0;

  _Pragma( "loopbound min
                  100 max 100" );
  for ( i = 0; i < 100; i++ ) {
    if ( i < 50 )
      j += i;
    else
      j += ( i * 13 ) % 42;
  }

  return j;
}
```

CFG



WCETs of BB (aiT 4 TriCore)

_main:	21 cycles
_L1:	27
_L3:	2
_L4:	2
_L5:	20
_L6:	13
_L2:	20

Example (2)

- Virtual start node
- Virtual end node
- Virtual end node per function

Variables:

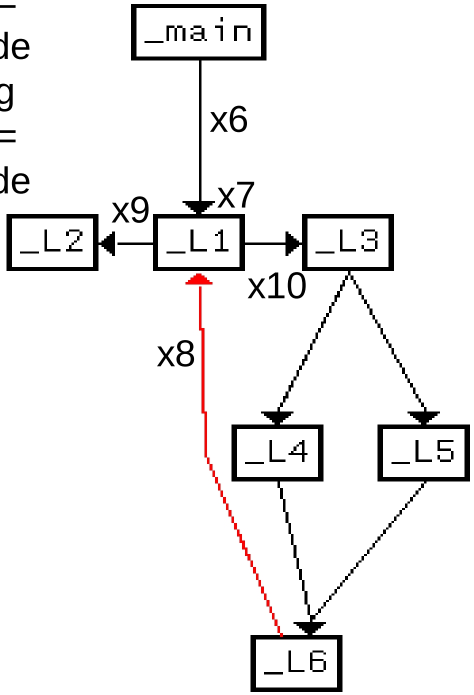
- 1 variable per node
- 1 variable per edge

Constraints: „Kirchhoff“ equations per node

- Sum of incoming edge variables = flux through node
- Sum of outgoing edge variables = flux through node

ILP

_main:	21 cycles
_L1:	27
_L3:	2
_L4:	2
_L5:	20
_L6:	13
_L2:	20



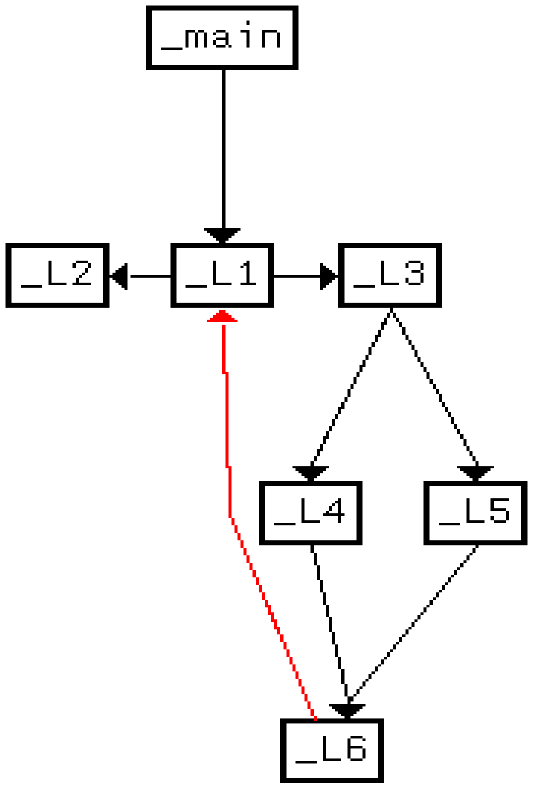

```
/* Objective function = WCET to be maximized*/
21 x2 + 27 x7 + 2 x11 + 2 x14 + 20 x16 + 13 x18 + 20 x19;
/* CFG Start Constraint */ x0 - x4 = 0;
/* CFG Exit Constraint */  x1 - x5 = 0;
/* Constraint for flow entering function main */
x2 - x4 = 0;
/* Constraint for flow leaving exit node of main */
x3 - x5 = 0;
/* Constraint for flow entering exit node of main */
x3 - x20 = 0;
/* Constraint for flow entering main = flow leaving main */
x2 - x3 = 0;
/* Constraint for flow leaving CFG node _main */
x2 - x6 = 0;
/* Constraint for flow entering CFG node _L1 */
x7 - x8 - x6 = 0;
/* Constraint for flow leaving CFG node _L1 */
x7 - x9 - x10 = 0;
/* Constraint for lower loop bound of _L1 */
x7 - 101 x9 >= 0;
/* Constraint for upper loop bound of _L1 */
x7 - 101 x9 <= 0; ….
```

Example (3)

Value of objective function: 6268

Actual values of the variables:

x2	1
x7	101
x11	100
x14	0
x16	100
x18	100
x19	1
x0	1
x4	1
x1	1
x5	1
x3	1
x20	1
x6	1
x8	100
x9	1
x10	100
x12	100
x13	0
x15	0
x17	100



Summary

Evaluation and Validation

- In general, multiple objectives
- Pareto optimality
- Design space evaluation (DSE)
- Execution time analysis
  - Trade-off between speed and accuracy
  - Computation of worst case execution times
    - Cache/pipeline analysis
    - ILP model for computing WCET of application from WCET of blocks