Evaluation and Validation

Peter Marwedel
TU Dortmund, Informatik 12
Germany

2011年 06 月 17 日

These slides use Microsoft clip arts. Microsoft copyright restrictions apply.
Structure of this course

Application Knowledge

2: Specification
3: ES-hardware
4: system software (RTOS, middleware, ...)

Design repository

6: Application mapping
7: Optimization
5: Evaluation & validation (energy, cost, performance, ...)

Design

8: Test

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?

In practice, many different criteria are relevant for evaluating designs:

- (average) speed
- worst case speed
- power consumption
- cost
- size
- weight
- 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 speed, cost, power 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, a total order < and the corresponding order ≤ are defined.

**Definition:**
Vector $u = (u_1, \ldots, u_n) \in F$ dominates vector $v = (v_1, \ldots, v_n) \in F$ if

$u$ is “better” than $v$ with respect to one objective and not worse than $v$ with respect to all other objectives:

\[
\forall i \in \{1, \ldots, n\}: u_i \leq v_i \land \\
\exists i \in \{1, \ldots, n\}: u_i < v_i
\]

**Definition:**
Vector $u \in F$ is indifferent with respect to vector $v \in F$ if

neither $u$ dominates $v$ nor $v$ dominates $u$
Pareto points

- A solution $x \in X$ is called **Pareto-optimal** with respect to $X$ $\iff$ there is no solution $y \in X$ such that $u = f(x)$ is dominated by $v = f(y)$

- **Definition**: Let $S \subseteq F$ be a subset of solutions. $v$ is called a **non-dominated solution** with respect to $S$ $\iff v$ is not dominated by any element $\in S$.

- $v$ is called **Pareto-optimal** $\iff v$ is non-dominated with respect to all solutions $F$. 
Pareto Point

Objective 1 (e.g. energy consumption)

Objective 2 (e.g. run time)

indifferent

better

Pareto-point

worse

(Assuming minimization of objectives)
Pareto Set

Objective 1 (e.g. energy consumption)

Pareto set = set of all Pareto-optimal solutions

(Assuming minimization of objectives)

Objective 2 (e.g. run time)
One more time …

Pareto point

- dominated by (1)
  (in inferior designs)

min

indifferent

(1)

dominating (1)
(supior designs)

min

(e.g. energy)

O 1

O 2 (e.g. memory space)

Pareto front

- dominated design points

min

indifferent

(1)

dominating (1)
(supior designs)

min

(e.g. energy)

O 1

O 2 (e.g. memory space)
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.
Evaluation of designs according to multiple objectives

Different design objectives/criteria are relevant:

- Average performance
- Worst case performance
- Energy/power consumption
- Thermal behavior
- Reliability
- Electromagnetic compatibility
- Numeric precision
- Testability
- Cost
- Weight, robustness, usability, extendibility, security, safety, environmental friendliness
Performance evaluation

- Estimated performance values:
  Difficult to generate sufficiently precise estimates;
  Balance between run-time and precision

- Accurate performance values:
  As precise as the input data is.

We need to compute average and worst case execution times
Worst/best case execution times

Requirements on WCET estimates:

- **Safeness:** WCET ≤ WCET_{EST}!
- **Tightness:** WCET_{EST} – WCET → minimal
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

Behavior at program joins

Worst case

Intersection+max. age

Best case

Union+min. age
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

```c
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

/* 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; ....

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

Performance 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