WCET-aware Scratchpad memory allocation

Sireesha R Basavaraju
Embedded Systems Group, Technical University of Kaiserslautern
Overview

- Introduction
- WCET of program – ILP Formulation
- Requirement
- SPM allocation for code
- SPM allocation for data
- Conclusion
- References
Introduction

- Real-time (RT) systems: Embedded systems whose correctness depends on timeliness of results
- RT systems application: automotive engine control, aircraft flight control
- Worst-case execution time (WCET) analysis
  - Used to guarantee RT system safety is met
- To bridge gap between processor speed and memory
  - On-chip memories typically caches are used
- But caches are unpredictable in nature
  - Go for Scratchpad memories (SPMs) for RT systems
Scratchpad memories (SPM) are small on-chip memories that are mapped into the address space of the processor.

Advantages of SPM
- Smaller in size compared to caches
- Energy efficient
- Access latency is predictable

When SPMs are used
- The decision of placing data and code objects lies in hands of compiler
Overview

- Introduction
- WCET of program – ILP Formulation
- Requirement
- SPM allocation for code
- SPM allocation for data
- Conclusion
- References
A program P’s WCET is the maximal time P’s execution can ever take.

WCET analysis of program is performed on control flow graph (CFG) of program.

Goal: find WCET of program
- Start from functions in the program
- For each function, for each loop, the innermost loop is an acyclic path
- Obtain the WCET of the innermost loop and multiply it with loop’s worst case loop count
Cost of each basic block is given by:
\[ c_i = C_{\text{min}}^i * (1 - x_i) + C_{\text{spm}}^i * x_i \]

With \( x_i \), allocation variable defined as:
\[ x_i = \begin{cases} 
1 & \text{If basic block is SPM allocated} \\
0 & \text{otherwise} 
\end{cases} \]

Start from basic block at the exit and then moving upwards to basic block at the entry of loop.

WCET of basic block at exit will be cost of basic block itself:
\[ W_{\text{exit}}^L = C_{\text{exit}}^L \]

WCET of any basic block \( b_i \) is obtained as:
\[ W_i \geq W_{\text{succ}} + c_i \]
WCET of program – ILP formulation (contd.)

- The WCET of all paths of loop body of $L$ can be obtained by modeling the basic block at the entry of loop:
  \[ W_{entry}^L \]

- Replace the innermost loop by a single node in graph
- Repeat the procedure iteratively
- main function is the unique entry point of an entire program
- Thus WCET of entire program: \[ W_{entry}^{main} \]
Overview

- Introduction
- WCET of program – ILP Formulation
- Requirement
- SPM allocation for code
- SPM allocation for data
- Conclusion
- References
Requirement

Requirement:
- Optimal SPM allocation which is WCET-aware i.e., it reduces the WCET of application

Problem:
- An allocation optimal for average case execution time is not optimal for WCET reduction
- Locally optimizing the current worst-case path may not lead to the globally optimal solution
- Allocation must consider changes in worst case path
Overview

- Introduction
- WCET of program – ILP Formulation
- Requirement
- SPM allocation for code
  - WCET-aware static allocation
  - Instruction SPM allocation for PRET architecture
- SPM allocation for data
- Conclusion
- References
WCET-aware static allocation for program code [1]

- Minimizes WCET by placing the most beneficial parts of a program’s code in SPM

- This method:
  - captures a program’s current WCEP and its possible switches
  - considers jump penalties and variable basic block sizes based on jump scenarios
ILP modeling

- Goal: ILP has to minimize a program’s WCET by assigning basic blocks to the SPM
- The objective function: $w_{main}^{\text{entry}}$
- Extend WCET of each basic block $b_i$ by
  - Jump penalty $j_{p_i}$
  - Call penalty $c_{p_i}$
ILP modeling: Allocating of consecutive basic blocks

Why jump penalty?

- Consider unconditional jump scenario:
  - After $b_i$, $b_j$ has to execute
    - If they are placed in different memories
    - If they are placed in same memory but not adjacently
    - If they are placed in adjacently in same memory
Modeling global control flow:

- A function call to F inside a basic block $b_i$ of F'
- Use $W_{entry}^F$ to model worst case cost of $b_i$
- In addition a call penalty to model a jump to function scenario
- Thus WCET of basic block is:

$$w_i \geq w_{succ} + c_i + jp_i + cp_i$$
Advantages:
- Considers Jump penalties and variable basic block sizes based on jump scenario
- It is optimal WCET-centric allocation method for program code as it is based on ILP

Disadvantage:
- WCET analysis required to find cost of basic block is expensive
Overview

- Introduction
- WCET of program – ILP Formulation
- Requirement
- SPM allocation for code
  - WCET-aware static allocation
  - SPM allocation for PRET architecture
- SPM allocation for data
- Conclusion
- References
Instruction allocation to SPM for PRET architecture [2]

- Precision timed (PRET) architecture has predictable timing and repeatable temporal behaviors.
- It provides precise timing control using timing instructions.
- PRET is a multi-threaded architecture.
- CFG used for ILP formulation is extended with timed blocks.
ILP modelling

- The objective function:
  - Overall execution time of all the timed blocks of all the threads
- This is minimized under constraints:
  - Timing constraint: Execution time of individual time blocks is just met
  - SPM size constraint: the total sum of size of all basic blocks must lie within SPM size
Instruction allocation to SPM for PRET architecture

- Advantages:
  - incorporates instructions from multiple hardware threads of the PRET architecture
  - Allocation utilizes unused space of one thread for another thread’s timed blocks and hence better than dedicated segmented SPM for each thread
  - It allocates basic blocks so that explicit timing specified in the program is met
Overview

- Introduction
- WCET of program – ILP formulation
- Requirement
- SPM allocation for code
- SPM allocation for data
  - WCET-aware static allocation based on ILP
  - SPM allocation using Branch and Bound Algorithm
  - Dynamic SPM allocation for data
- Conclusion
- References
WCET-aware static allocation based on ILP for data

- This allocation considers both scalar variables and array
- The method analyses CFG of program to find the objective function
  \[ w_{main} \]
  \[ w_{entry} \]
- Allocate variables to SPM to reduce WCET of program
- Objective function is minimized under SPM size constraint
WCET-aware static allocation based on ILP for data

- **Advantages:**
  - Optimal method since it is based on ILP
  - WCET reduction of up to 80% is possible for different benchmarks

- **Disadvantage:**
  - Doesn’t consider infeasible path information in programs
Overview

- Introduction
- WCET of program – ILP formulation
- Requirement
- SPM allocation for code
- SPM allocation for data
  - WCET-aware static allocation based on ILP
  - SPM allocation using Branch and Bound Algorithm
  - Dynamic SPM allocation for data
- Conclusion
- References
WCET-centric SPM allocation for data using Branch-and-bound algorithm[3]

- Search space = power set of all possible allocations

- Measure the potential WCET reduction of a variable using its maximum contribution over all execution paths
  - sort the variables in decreasing order of this measured value
WCET-centric SPM allocation for data using Branch-and bound algorithm (contd.)

- Traverse the search tree and keep the maximum WCET reduction achieved so far at any leaf node bound.
- Heuristic function computes the maximum possible WCET reduction at any leaf node in the sub-tree rooted at node m’.
- The search space corresponding to the sub-tree rooted at m’ is pruned based on the maximum WCET reduction that can be obtained on that path.
Procedure for finding WCET path

1. **General idea**:
   - Find the infeasible path patterns by conservatively identifying pairs of branches and/or assignments which are guaranteed to “conflict”
     - Example: $x=2$ conflicts with $x>3$
   - WCET calculation same as before but using the conflict list for each path list to avoid exhaustive path enumeration

2. **WCET-aware Scratchpad memory allocation**

9/6/2015
Advantages:
- Provides optimal allocation
- Considers infeasible path information in program
- Reduction obtained in WCET is higher than that using ILP based static allocation

Disadvantage:
- Complexity is exponential with respect to the number of data variables to be allocated
Overview

- Introduction
- WCET of program - ILP formulation
- Requirement
- SPM allocation for code
- SPM allocation for data
  - WCET-aware static allocation based on ILP
  - SPM allocation using Branch and Bound Algorithm
  - Dynamic SPM allocation for data
- Conclusion
- References
WCET centric Dynamic allocation for data [4]

- Enables eviction and placement of data to the SPM at run-time

Steps:
1. Determine the possible targets for load–store instruction
2. ILP formulation to obtain an allocation
3. Greedily refine the allocation to get the WCET–centric allocation
WCET centric Dynamic allocation for data (contd.)

1. Determine the possible targets for load–store instruction:
   Done by applying pointer analysis
   ◦ To whole text of program
   ◦ To all pointer definitions

2. Compute WCET of application

3. Compute data access information on worst–case execution path
4. ILP formulation to obtain an allocation
   ◦ 0/1 ILP is formulated to compute allocation
   ◦ Formulation is applied on flow graph

5. Apply the process iteratively till no more reduction in WCET can be obtained
Advantages:
- For small SPM highly beneficial
- Outperforms the static allocation from 12% to 85%
- Useful to systems with a SPM shared among several RT tasks

Disadvantages:
- Allocation quality depends on granularity of flow graph
- Run time of ILP solver is a limitation for the largest benchmarks
Overview

- Introduction
- WCET of program
- Requirement
- SPM allocation for code
- SPM allocation for data
- Conclusion
- References
Conclusion

- WCET is a key metric in RT embedded systems
- WCET-centric SPM allocation techniques for program code and data objects
- ILP based method give optimal allocation reducing WCET
- Instruction allocation for PRET architecture allocates basic blocks from multiple threads and meet the explicit timing requirements
- Dynamic allocation approach for data allows efficient use of small SPM
Overview

- Introduction
- WCET of program
- Requirement
- SPM allocation for code
- SPM allocation for data
- Conclusion
- References
References


Thank you 😊