CACHE MODELING AND SCHEDULING ANALYSIS WITH AADL

HAI-NAM TRAN, FRANK SINGHOFF, STÉPHANE RUBINI, JALIL BOUKHOBZA

LAB-STICC, UNIVERSITY OF WESTERN BRITTANY (UBO)

Outline

1. Introduction
2. Related works
3. Modeling cache with AADL
4. Modeling control flow graph with AADL
5. Cache and scheduling analysis
6. Conclusion & future works.

1.1 Context: the Cheddar project

- **Cheddar**: a free real-time scheduling analysis and simulation tool, used to support the schedulability analysis of AADL models

- **Our plan**: take memory hierarchies into account.
  - Start from the cache memory.

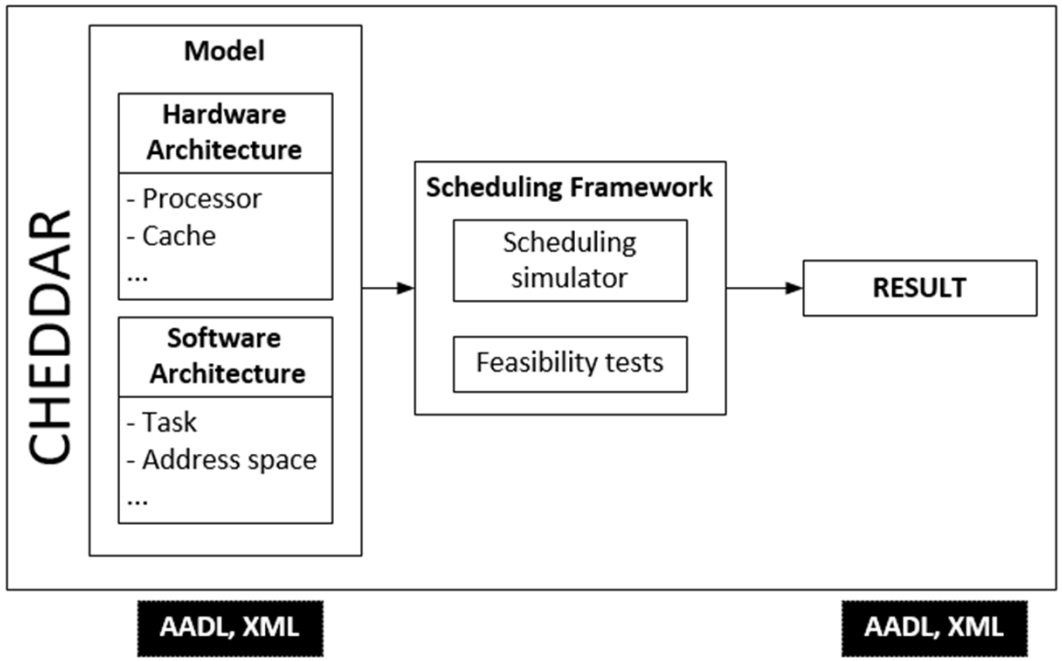


Figure: the framework of Cheddar

Definition

Cache is small high speed memory (usually SRAM) that contains the most recently accessed pieces of main memory (usually DRAM).

1.2 Problems and motivations

MOTIVATIONS

- Cache: a solution narrowing the speed gap between processor and memory

- The popularization of contemporary processors, which cache is integrated, in real-time systems.

PROBLEMS: Cache ‘s unpredictability.

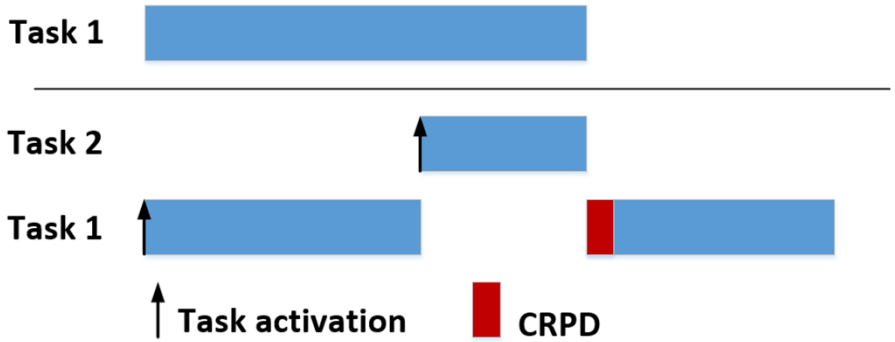
- Cache access time depends on the memory block is in the cache or not.

- In preemptive scheduling context, need to take into account the impact of *cache related preemption delay* (**CRPD**).

→ Need a solution to **handle cache during real time scheduling analysis**

1.3 Objective

- Estimate the cache related preemption delay (**CRPD**) - the additional time to refill the cache with the cache blocks evicted by the preemption.



- Model the system architecture, simulate the scheduling and export the result.
- Focus on analyze the direct-mapped *instruction cache* in the first steps.

Outline

1. Introduction
2. **Related works**
3. Modeling cache with AADL
4. Modeling control flow graph with AADL
5. Cache and scheduling analysis
6. Conclusion & future works.

2. Related works

- **SymTA/P** (Staschulat et al., 2005): a timing analysis tool, supports cache analysis based on the program's control flow graph.

- **SimSo** (Chéramy et al.,2013): a scheduling simulation tool, supports cache sharing on multi-processor system. Modeling the cache behavior based on Stack Distance Profile.

- **Our approach**
  - Focus on scheduling analysis.
  - Based on the cache behavior deduced from the Control Flow Graph of a program (Lee et al.,1998)
  - Take into account the memory-to-cache mapping of tasks (following the work of Busquets-Mataix et al., 1996)

Outline

1. Introduction
2. Related works
3. **Modeling cache with AADL**
4. Modeling control flow graph with AADL
5. Cache and scheduling analysis
6. Conclusion & future works.

3.1 Cache attributes

Definition

- **cache hit**: access to a memory block in the cache
- **cache miss**: access to a memory block not in the cache

Cache Attributes

- ***Cache_Size***: the size of the cache.
- ***Block_Size***: number of contiguous bytes that are transferred from main memory on a cache miss.
- ***Associativity***: the number of cache locations where a particular memory block may reside.
- ***Hit_Time***: access time when cache hit
- ***Miss_Time***: access time when cache miss

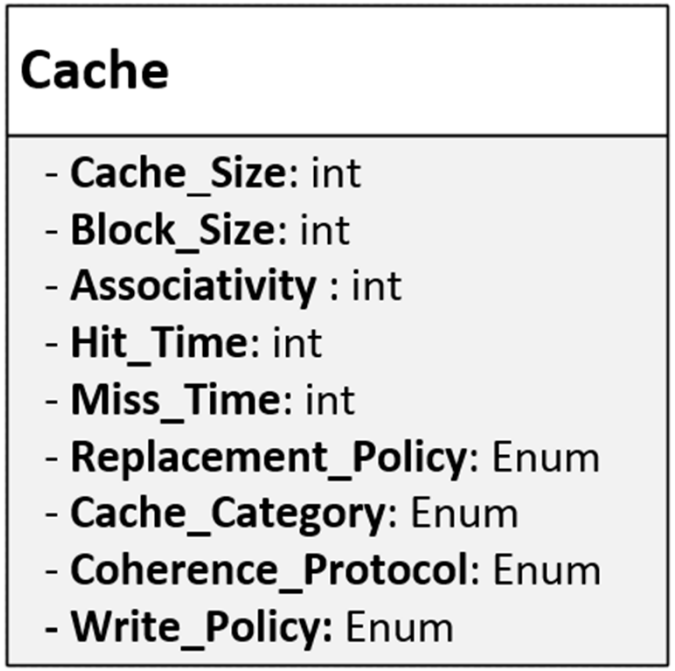


Figure: Cache model (UML)

3.1 Cache attributes

Cache Attributes

- ***Replacement_Policy***: decides which cache block should be replaced when a new memory block needs to be stored in the cache.
- ***Cache_Category***: specifies a cache is *instruction cache*, *data cache* or a *combined one*.
- ***Coherence_Protocol***: protocol maintains the consistency between all the caches in a system of distributed shared memory
- ***Write_Policy***: determines how to write of the data back to the main memory.

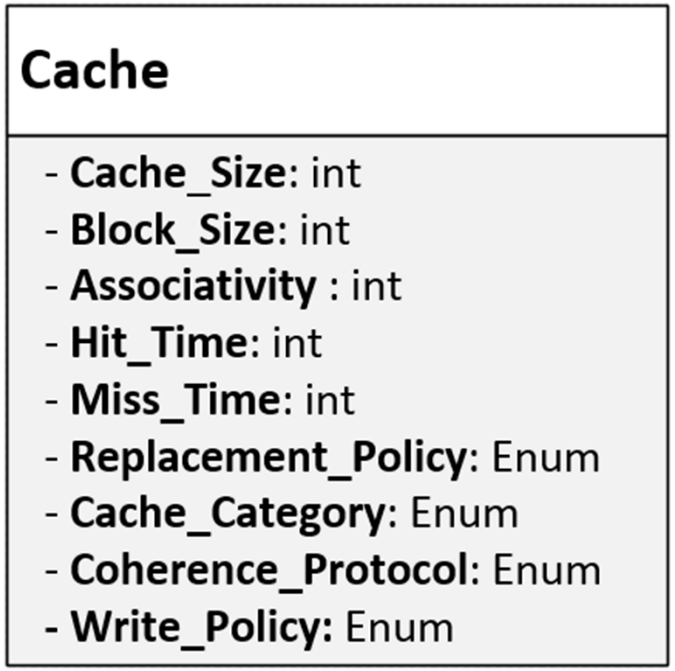


Figure: Cache model (UML)

3.2 Attributes comparison with AADL memory component

We expect to model cache with AADL memory component.

Cache	Memory (AADL)
	MemoryProtocol: Read/Write
Cache_Size	**Byte_Count**
Block_Size	**Word_Size**
Associativity	**No equivalent**
Hit_Time	**Read_Time**
Miss_Time	= Higher Level memory Read_Time + Cache Write_Time
Replacement_Policy	**No equivalent**
Cache_Category	**No equivalent**
Coherence_Protocol	**No equivalent**
Write_Policy	**No equivalent**

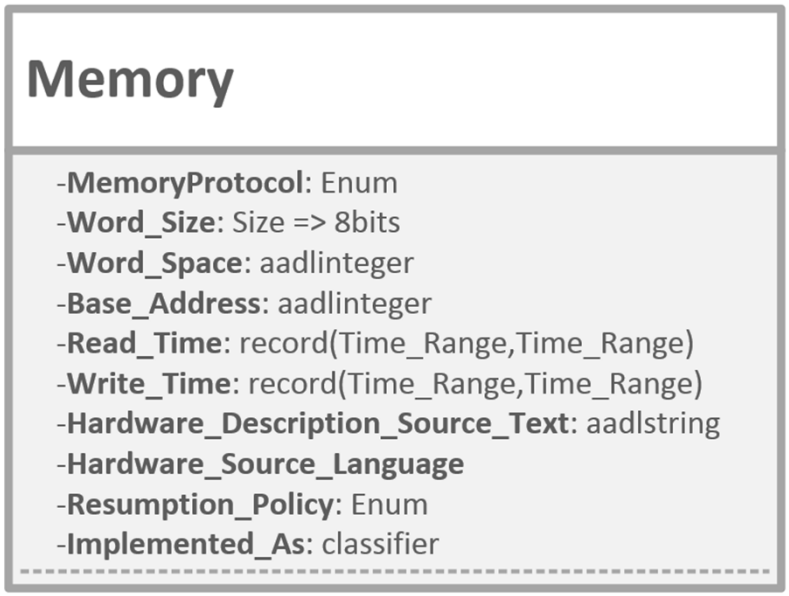


Figure: AADL memory component model (UML)

3.3 Defining cache attributes using AADL User Defined property

Cache = Memory + Added Properties

property set set_of_caches **is**

associativity: **type** *aadlinteger* ***applies to (memory)*;**

replacement_policy: **type enumeration** *(FIFO, LRU, PLRU, Random)*
applies to (memory);

coherence_protocol: **type enumeration** *(Shared_Cache_Protocol, Private_Cache Protocol,...)*
applies to (memory);

cache_category: **type enumeration** *(Data_Cache, Instruction_Cache, Unified_Cache)*
applies to (memory);

write policy: **type enumeration** *(Write_Back, Write_Through, Write_Through_Allocate)*
applies to (memory);

end set_of_caches;

Outline

1. Introduction
2. Related works
3. Modeling cache with AADL
4. **Modeling control flow graph with AADL**
5. Cache and scheduling analysis
6. Conclusion & future works.

4. Modeling control flow graph with AADL

Definition

- *Control flow graph* (CFG): a representation, using graph notation, of all paths that might be traversed through a program during its execution. Each node represents a *basic block*.

Why need the CFG ?

- A requirement for cache analysis methods. (Detailed description in Section 5)
- Which the cache analysis in Cheddar we use an abstract level of CFG.

How to get it ?

- Extract CFG using binaries analysis tool such as aIT (by AbsInt),

Bound-T (by Tidorum).

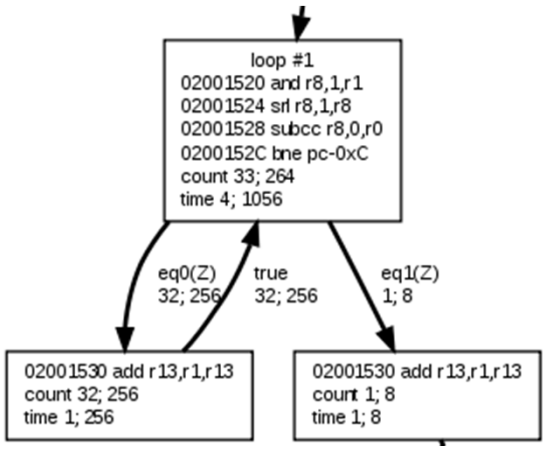


Figure: Control Flow Graph example

4. Modeling control flow graph

An abstract model of *CFG* used for cache analysis in Cheddar:

- Control flow graph: a set of basic blocks

- Basic block attributes:
  - ***PreviousBlock***, ***NextBlock***: indicate the position of the basic block
  - ***InstructionOffset***, ***InstructionCapacity*** : position and capacity of the assembly instruction in main memory
  - ***LoopBound***: additional properties used for CRPD analysis.
  - ***Type***: specifies a block is START, MIDDLE or TERMINATE block.

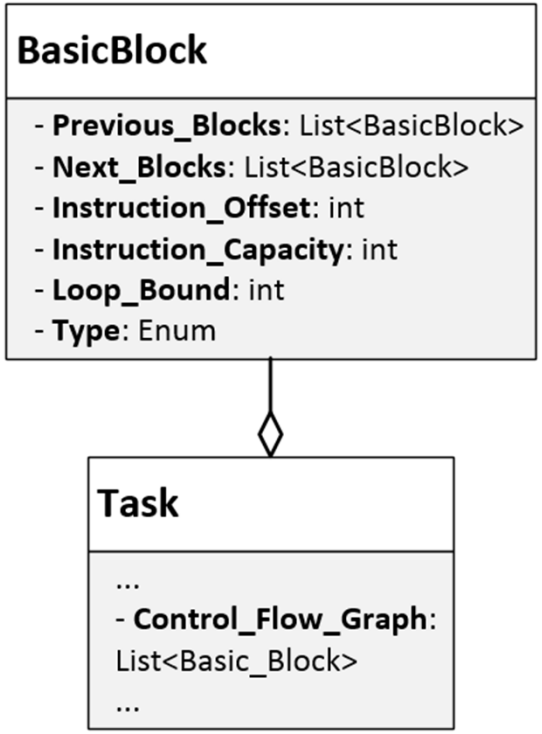


Figure: Control Flow Graph model in Cheddar

4. Modeling control flow graph

Solutions to model the control flow graph in AADL.

- Could we use AADL Behavior Annex to model it ?

Or

- Store the CFG on a separate file.
- User defined property: put a reference to the control flow graph file in the annotated AADL model: *cfg_source_file*: **type** *string applies to (subprogram)*;

Outline

1. Introduction
2. Related works
3. Modeling cache with AADL
4. Modeling control flow graph with AADL
5. **Cache and scheduling analysis**
6. Conclusion & future works.

5.Cache and scheduling analysis

Our approach:

1. Model the cache memory and program's control flow graph (section 3,4)
2. Apply cache analysis methods, calculate the cache related preemption delay.
3. Put the result of cache related preemption delay in scheduling analysis.

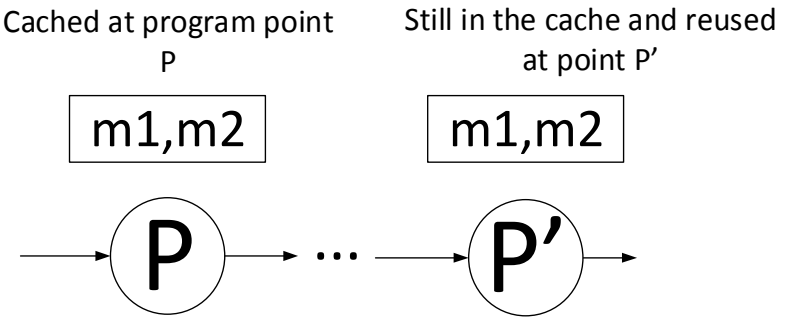
5.1 Cache related preemption delay analysis methods overview

Two basic analysis methods in the domain of cache related preemption delay (CRPD):

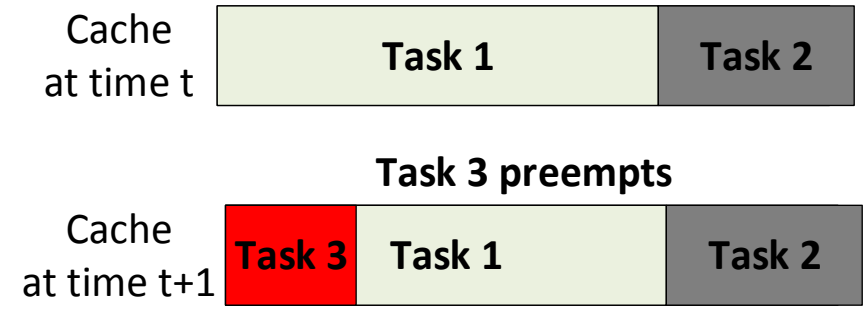
- **Useful cache blocks (UCB)**: find the number of cache blocks that hold memory blocks, which are:
  - Cached
  - Reused in the execution of a task.

- **Evicting cache blocks (ECB)**: find the number of memories blocks that are accessed/placed in the execution of the preempting task.

The CRPD is calculated based on the number of cache blocks, which are: useful, evicted or both.



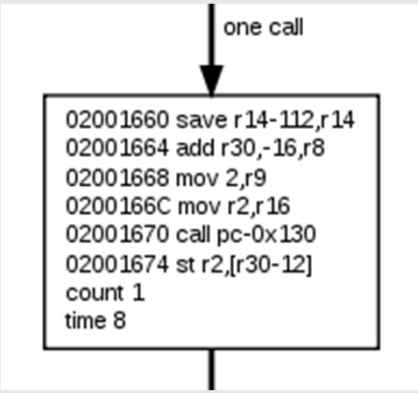
UCB Example: memory block 1 and 2 are useful

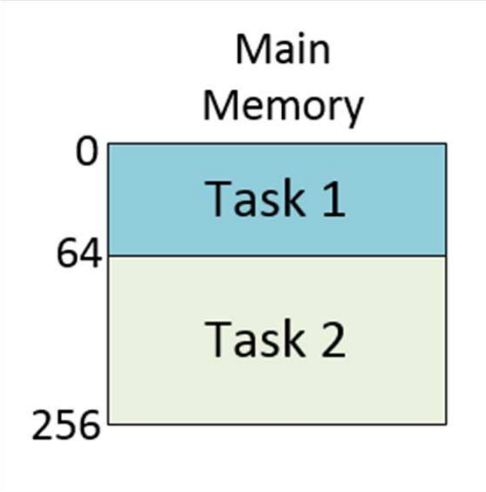


ECB example: Task 2 evicted the content of Task 1 in the cache

5.2 CRPD analysis methods - Requirements

UCB requirements	ECB requirements
- Cache configuration. - Task's memory-to-cache mapping. - Task's control flow graph.	- Cache configuration. - Task's memory-to-cache mapping.





5.3 CRPD analysis methods – Models

UCB based analysis method:

- Cache and control flow graph model in section 3 and 4.

ECB based analysis methods:

- Cache model in section 3
- Use the basic block to hold task's memory usage.

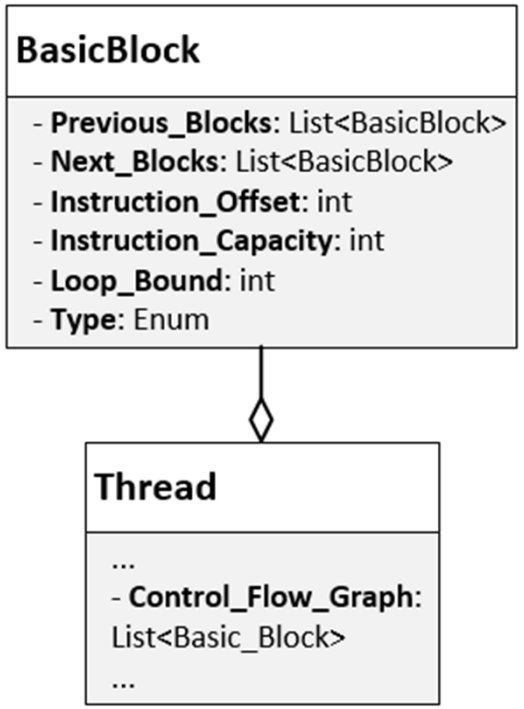


Figure: Control flow graph model (UML)

5.4 Implementation in Cheddar

- Software Architecture: Extended task model to model the control flow graph.

- Hardware Architecture: Added cache model.

- Implemented analysis methods:
  - Useful Cache Block (Lee et al.,1998): used for static analysis of a single task.
  - Evicting Cache Block (Busquet-Mataix et al, 1996): used in combination with ECB when perform scheduling simulator

- Update the scheduling simulator to add the effect of CRPD to the capacity of the task.

5.5 Experiment & result

Objective

- Evaluate the effect of CRPD.

Configuration

- LEON v3 processor, clock speed 400 MHz, 1KB instruction cache, 16 bytes block size, WCET + CFG generated by aiT (AbsInt), task sets from Malardaren benchmark suite.

Task	WCET (µs)	WCET with cache (µs)	Max UCB	Max CRPD (µs)	Max CRPD (%)
bs	6.1	4.5	14	0.35	7.78
fac	5.9	4.9	10	0.25	5.10
fdct	80.9	80.2	49	1.23	1.53
fibcall	8.1	4	7	0.18	4.38
insertsort	41.07	22.22	11	0.28	1.24
ns	545.3	273.2	20	0.50	0.18
prime	6.6	6.8	24	0.6	8.82

Table: experiment result

5.5 Experiment & result

- Results:
  - Cache related preemption delay: 1%-7% WCET – for one preemption.
  - Important to minimize the number of preemption → need a priority assignment algorithm.

Outline

1. Introduction
2. Related works
3. Modeling cache with AADL
4. Modeling control flow graph with AADL
5. Cache and scheduling analysis
6. **Conclusion & future works.**

6.1 Conclusion

Problem

- In preemptive scheduling context, we need to take into account the impact of *cache related preemption delay* (CRPD).

Our approach

- Model the cache memory and program's control flow graph.
- Apply CRPD analysis methods.
- Scheduling simulation with Cheddar.

From the result of experiment

- Cache related preempted delay (CRPD) from 1% to 7% of task's WCET.
- Should not ignored the impact of CRPD.

6.2 Future works

- Integrate the analysis into an AADL tool chain to perform scheduling analysis with cache.
- Take *the cache related preemption delay* into account when assigning priority to tasks.
- Exercise solutions to model the *control flow graph* with AADL.