YAMM

Yet Another Memory Manager

Ionut Tolea
AMIQ Consulting

June 23, 2016
Munich, Germany

Agenda

- Theory
  - Memory Management Introduction
  - YAMM Overview: Features, Algorithm, Data types, API
- Comparison with UVM_MAM
  - Feature-wise
  - Performance-wise
- Examples

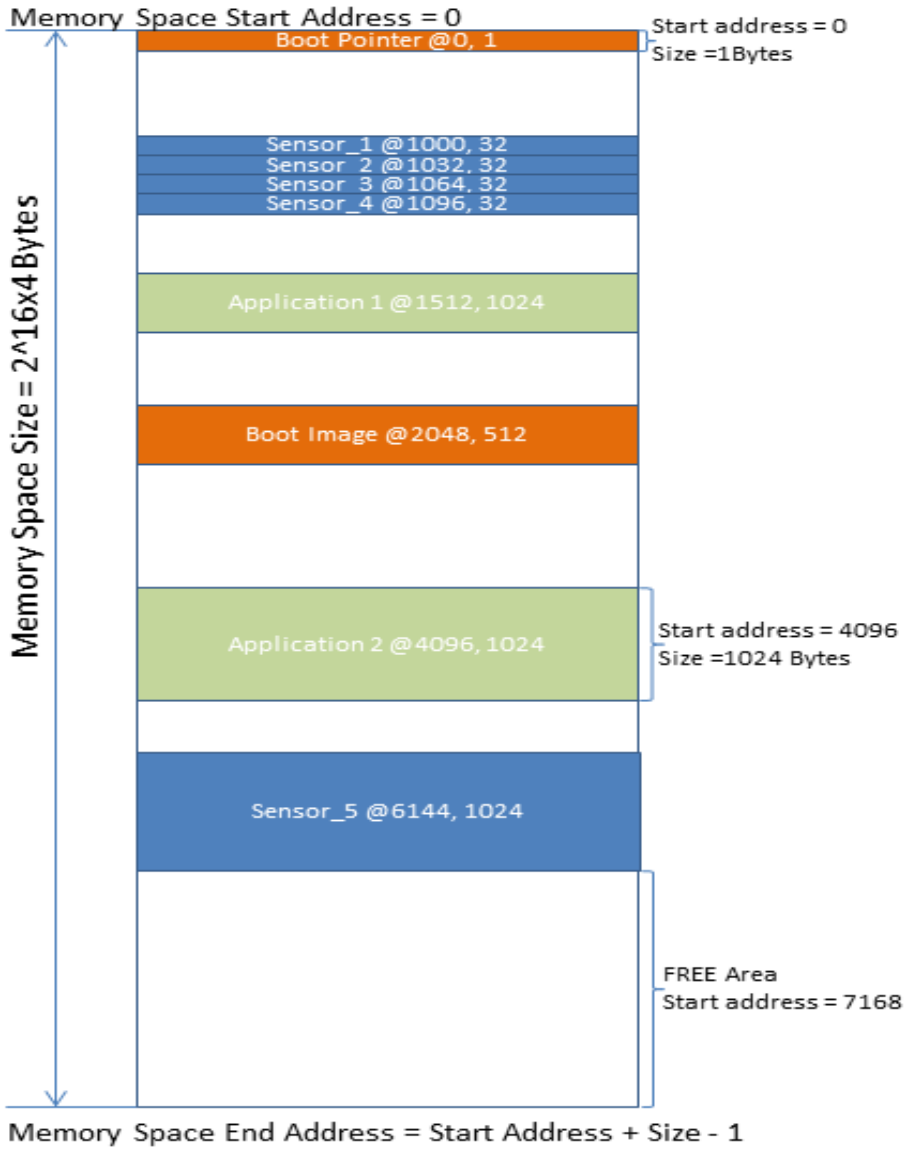
AMIQ Consulting

snug
Synopsys Users Group
Europe

Memory Management

Introduction

Memory Management – What is it?



Memory Management Requirements

Real Life	Verification
Provide memory buffers to programs	Support real life use cases
Prevent memory corruption	Provide randomization support
Reduce fragmentation	Provide debug support
	Not necessarily a memory model

YAMM

Overview

YAMM

Features (1)

- Everything is a buffer, either it’s a free one or a user defined one
- YAMM provides API for retrieval of allocated buffers
- Buffers can be allocated using a specific allocation mode
- User can set different granularities and address alignments
- YAMM has the ability to store contents or generate random contents inside used buffers

YAMM

Features (2)

- Buffers represent address spaces themselves
- Memory can be dumped to file or can be returned in a readable form as a string
- It provides functions to check the statistics of a specific memory space
- Can be extended for specific use cases

YAMM

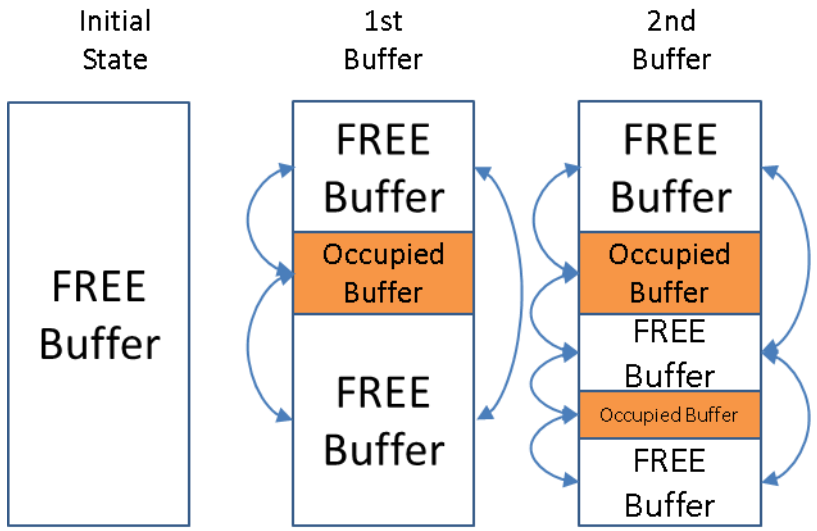
Data Types

- yamm_buffer
  - Contains all data and functions
- yamm
  - Top level class in the hierarchy
  - Inherits from yamm_buffer and implements specific functionality required for top level
- yamm_access
  - Optional usage
  - Used to model a basic access (start address, size)

YAMM Overview

Algorithm

- After initialization the memory map will contain a single free buffer
- All buffers in a memory map are chained in a double linked list



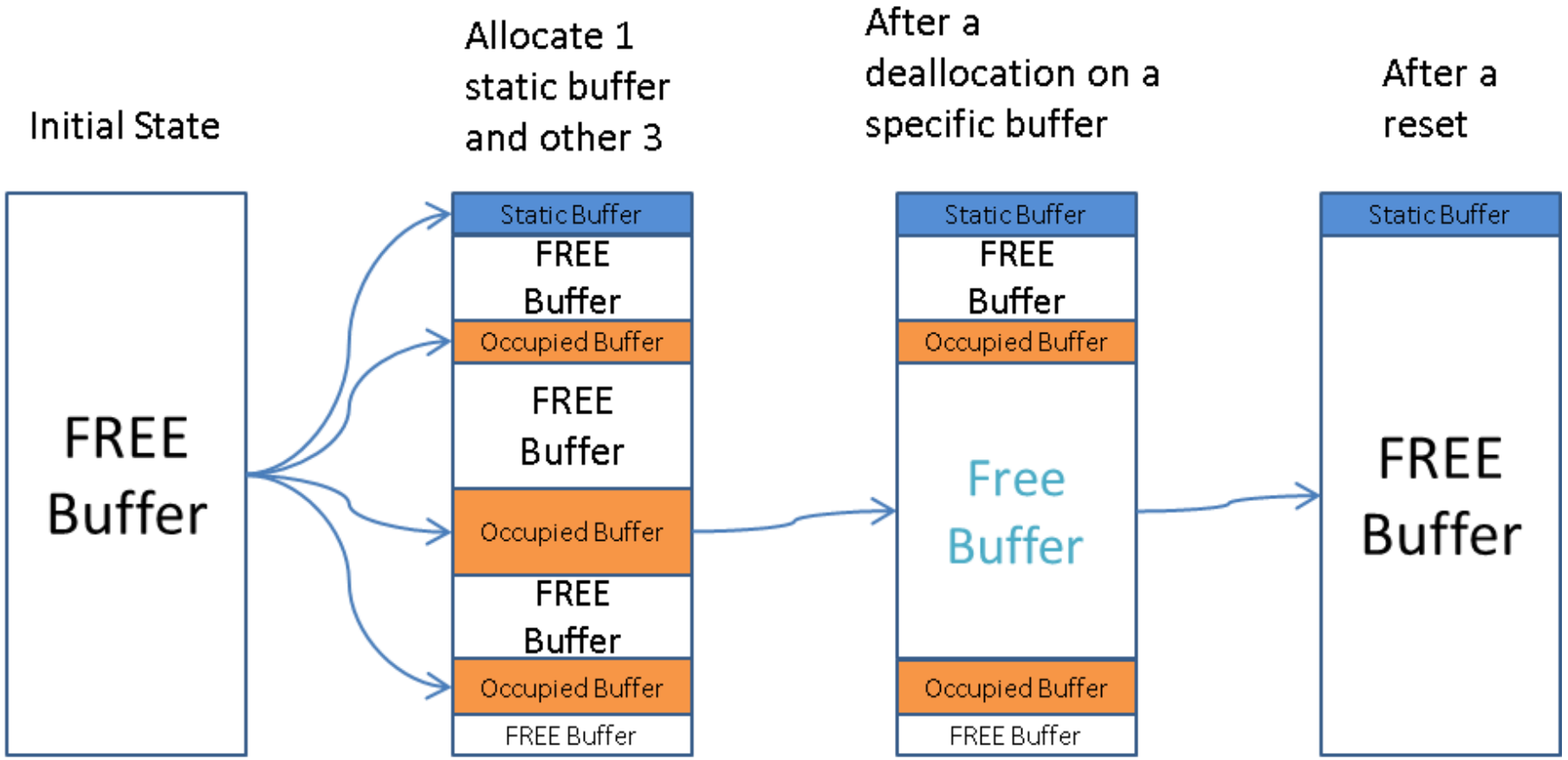
YAMM Overview

API

- Allocation
  - Manual allocation: insertion
  - Automatic allocation rules:
    - RANDOM_FIT
    - FIRST_FIT (and FIRST_FIT_RND)
    - BEST_FIT (and BEST_FIT_RND)
    - UNIFORM_FIT
- Deallocation
  - It can be done on a specific buffer or address

YAMM Overview

API - Allocation/Deallocation



YAMM Overview

API

- Search functions
  - Retrieve buffers by name
  - Retrieve buffer by address
  - Retrieve buffers in address range
  - Retrieve buffers by access
- Debug functions
  - Return the memory map allocation as a string
  - Dump memory map to file
  - Provide usage and fragmentation statistics

Comparison with UVM MAM

MAM

UVM solution

- The solution is oriented more towards memory modeling than memory management
- MAM plays a small role and is used with uvm_mem to reserve specific memory regions
- Supports only “greedy” allocation mode
- Specific regions can be freed or entire memory can be wiped
- Memory state can be returned as a string

MAM vs YAMM

Feature wise (1)

- MAM
  - Memory
    - uvm_mam is linked to uvm_mem which provides the mam memory locations used for storing data
  - Allocation
    - Can only allocate on previously unallocated memory
    - Has only 2 allocation modes
  - Deallocation
    - Releases the specific region

- YAMM
  - Memory
    - YAMM top level as well as every individual buffer contains a memory map composed of multiple buffers that can store simple data
  - Allocation
    - Permits allocation in previously unallocated memory or inside an already allocated buffer
    - Has 6 allocation modes
  - Deallocation
    - Can display a warning if said buffer contains other buffers

MAM vs YAMM

Feature wise (2)

- MAM
  - Finding buffers
    - Provides an iterator that user must use for any needs
  - Ease of use
    - It’s complex and rather hard to use
    - For features beyond reserving and freeing regions user has to go to objects higher in the hierarchy

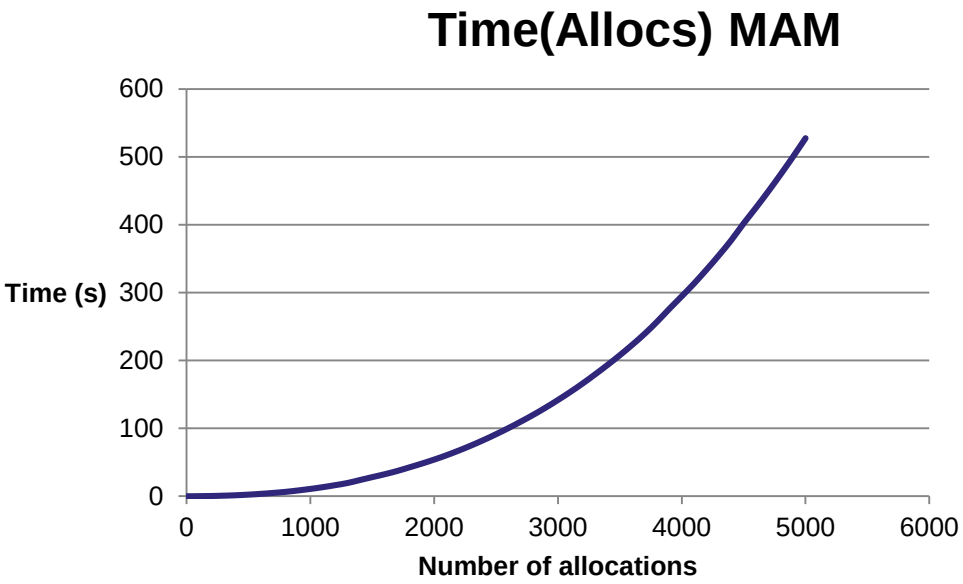
- YAMM
  - Finding buffers
    - Provides support for finding and modifying buffers by different criteria
  - Ease of use
    - Has a more user friendly API
    - Memory map can be accessed by calling functions on the top level
    - Specific regions can be accessed by calling same functions on the chosen buffers

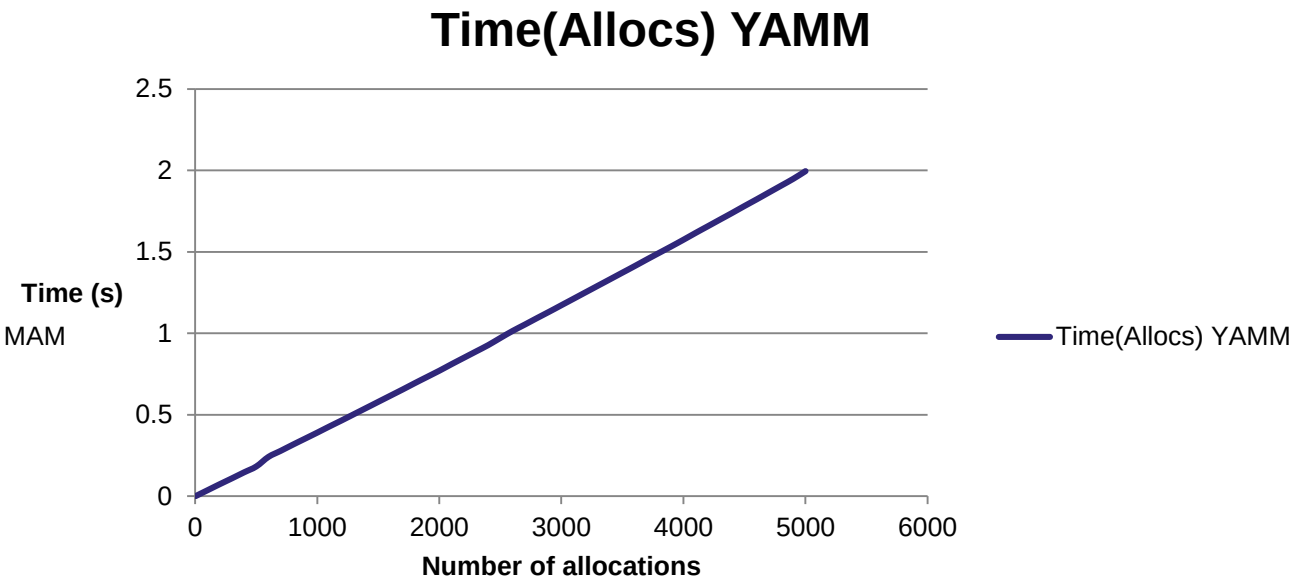
Performance test

Parameters

- Memory space of 1G
- Allocate 5000 buffers of size 100
- Measure the time taken for allocation every 100 allocations
- For MAM, `request_region()` with default policy was used (it will randomly allocate only memory that wasn't previously allocated)
- For YAMM, `allocate_by_size()` with RANDOM_FIT allocation mode was used

Performance - MAM vs YAMM





more than 200x speed

Examples

Examples

Configuration Example

```
yamm new_memory;
yamm_size_width memory_size = 10000;
new_memory = new;
new_memory.build("memory_name", memory_size);
// Now you can allocate static buffers. The memory was created but buffers
// can't be allocated yet. After reset(), normal buffers can be allocated but static
// buffers can't.
new_memory.reset();
```

Examples

Sequence Example

```
class user_sequence extends uvm_sequence;
   rand int unsigned access_size;
   ...
   task body();
      yamm_buffer buffer = p_sequencer.user_memory.allocate_by_size(access_size);
      `uvm_do_with(user_item, {
         address == buffer.start_addr;
         size == buffer.size;//or access_size
         data == buffer.get_contents();
      })
   endtask
endclass
```

Examples

Scoreboard Example

```
class user_scoreboard;
   yamm user_memory;
   ...
   //function checks if the current access is done to a previously allocated address
   function void check_access(user_item item);
      if(user_memory.get_buffer(item.addr) == null)
         `uvm_error(get_name(), "Access detected to a non-allocated memory address!")
   endfunction
endclass
```

YAMM Availability

- Free under Apache 2.0 license
- Available after SNUG conferences
- Blog: www.amiq.com/consulting/blog
- Github: github.com/amiq-consulting

AMIQ Consulting

snug
Synopsys Users Group
Europe

Questions?

AMIQ Consulting

snug
Synopsys Users Group
Europe

Thank You