A Dynamic Memory Allocation Library for High-Level Synthesis

Nicholas V. Giamblanco and Jason H. Anderson
University of Toronto, Canada
Dept. of Electrical and Computer Engineering
FPL 2019
Dynamic Memory Allocation in HLS: Current Problems

No Obvious Way TO include it!

Where and How Big should the Arena(Heap) Be?  Which Allocator?  Performance & Area Problems
Dynamic Memory Allocation in HLS: Why Include it?

- No More Code-Refactoring!
- No More Memory Over-Provisioning
- Portability
- Marginal Performance and Area Impacts!!!
The Allocators

**gnumem**  Linked-List Allocator.

**bitmem**  Bitmap Allocator.

**linmem**  Linear Allocator.

**budmem**  Buddy Allocator.

**lutmem**  Look-Up Table Allocator.
Our Approach

Implement Algorithms in HLS-friendly C Library
- Arena (heap) implemented as BRAM

Automate Transform with LLVM Pass
- User can select
  - Allocator Algorithm
  - Heap Size

Available on Github: https://github.com/ngiambla/libmem
Example:

// USER PROGRAM
void check_this_out() {
    int * arr = (int*)malloc(SIZE);
    //… do stuff here
    free(arr);
}

#TCL PARAMETERS FOR USER
set_parameter HEAP_SZ 65536
set_parameter ALLOC_S gnu

// USER PROGRAM
void check_this_out() {
    int * arr = (int*)gnu_malloc(SIZE);
    //… do stuff here
    gnu_free(arr);
}
Results: Area

Number of ALMs
Results: Performance

- linmem: 526.59 MHz
- gnumem: 451.47 MHz
- bitmem: 406.83 MHz
- lutmem: 394.17 MHz
- budmem: 410.51 MHz
Benchmarks

Typical Memory Request Patterns

Random: random request, random release
Square: request-do-release
Triangular: iterative-request do iterative-release

Real world apps

list  hash
priq  dfs
stack

Available on Github: https://github.com/ngiambla/dmbenchhls
Suggest an allocator based on **Memory Pattern** AND **User Requirements**

<table>
<thead>
<tr>
<th>Memory Pattern</th>
<th>Area Efficient</th>
<th>Latency Sensitive</th>
<th>Fast Clock Frequency</th>
<th>Exe. Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Pattern 1]</td>
<td>bitmem</td>
<td>gnumem, lutmem</td>
<td>gnumem</td>
<td>lutmem, gnumem</td>
</tr>
<tr>
<td>![Pattern 2]</td>
<td>bitmem</td>
<td>gnumem</td>
<td>lutmem, bitmem</td>
<td>lutmem</td>
</tr>
<tr>
<td>![Pattern 3]</td>
<td>linmem*, bitmem</td>
<td>linmem*, lutmem</td>
<td>linmem*, bitmem</td>
<td>linmem*, lutmem</td>
</tr>
</tbody>
</table>
Conclusions

- One Allocator does not ‘rule them all’
- Performance and area are marginally affected by allocators!
- Allocators within HLS work and are useful
THANKS!

SEE ME AT THE POSTER

Downloads:
https://github.com/ngiambla/libmem
https://github.com/ngiambla/dmbenchhls