
DTU

DTU

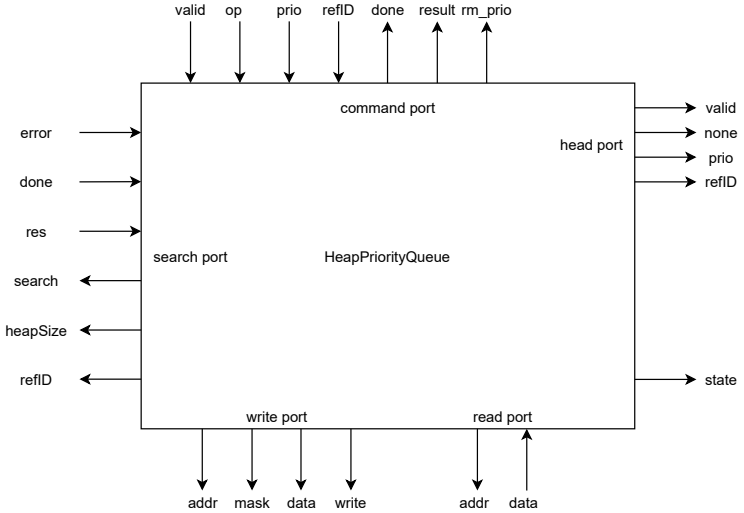
use case

Heap sort based priority queue in Chisel

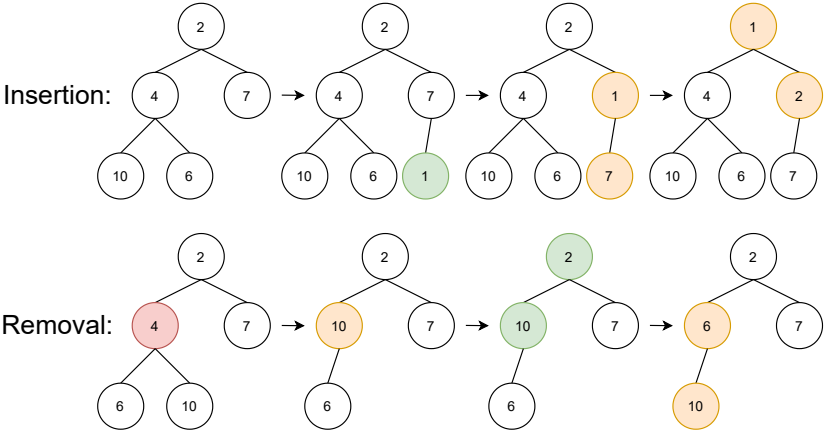
Priority Queue - The context

- Should sort elements so that the highest priority is in front of the queue using heap sort
- Usage:
  - Scheduling in time critical systems e.g. time-out stamps
  - A unique reference ID generated by the host is associated with every element
  - Elements can be removed by providing the appropriate reference ID
- Component should support queue sizes from 16 to 256 elements

The interface



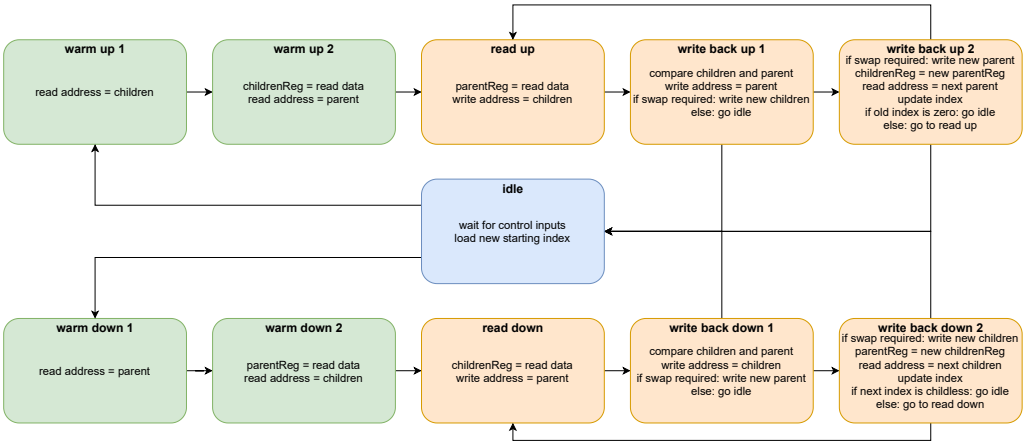
The heap sort algorithm



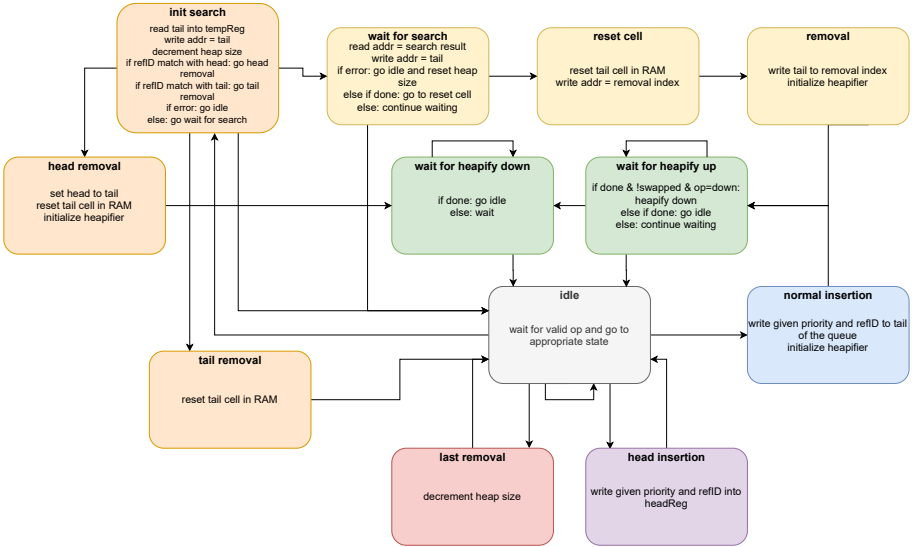
Insertion and removal take at most $\log_k(N)$ swap operations, where k is the number of children per node and N is the number of elements in the heap.

Heapifier

Component implementing the sorting:



The Top state machine



Testing

- A reference model described in scala
- A wrapper class for the DUT:
  - Abstracts interaction with the DUT
  - Simulates memory in scala so that it is completely accessible
- Random inputs are driven onto DUT and model
- Both are compared at the end of every operation
- The developed coverage report was employed to evaluate coverage of the random tests

A test run

```
====================Report====================
Heapsize = 17, children count = 4
66.70% of operations were valid
342 insertions which took on average 8.30 cycles
325 removals which took on average 11.61 cycles
==============================================


============ COVERAGE REPORT ============
============== GROUP ID: 1 ==============
COVER_POINT PORT NAME: operation
BIN insertion COVERING Range 0 to 0 HAS 1 HIT(S)
BIN removal COVERING Range 1 to 1 HAS 1 HIT(S)
========================================
COVER_POINT PORT NAME: cmd.prio.cycl
BIN cyclic COVERING Range 0 to 3 HAS 4 HIT(S)
========================================
COVER_POINT PORT NAME: cmd.prio.norm
BIN lower half COVERING Range 0 to 127 HAS 116 HIT(S)
BIN upper half COVERING Range 127 to 255 HAS 116 HIT(S)
```

```
========================================
COVER_POINT PORT NAME: head.prio.cycl
BIN cyclic COVERING Range 0 to 3 HAS 4 HIT(S)
========================================
COVER_POINT PORT NAME: head.prio.norm
BIN lower half COVERING Range 0 to 127 HAS 116 HIT(S)
BIN upper half COVERING Range 127 to 255 HAS 116 HIT(S)
========================================
CROSS_POINT cyclics at ops FOR POINTS operation AND cmd.prio.cycl
BIN insertion COVERING Range 0 to 0 CROSS Range 0 to 3 HAS 4 HIT(S)
BIN removal COVERING Range 1 to 1 CROSS Range 0 to 3 HAS 4 HIT(S)
========================================
CROSS_POINT normals at ops FOR POINTS operation AND cmd.prio.norm
BIN insertion lower half COVERING Range 0 to 0 CROSS Range 0 to 127 HAS 79 HIT(S)
BIN insertion upper half COVERING Range 0 to 0 CROSS Range 127 to 255 HAS 82 HIT(S)
BIN removal lower half COVERING Range 1 to 1 CROSS Range 0 to 127 HAS 116 HIT(S)
BIN removal upper half COVERING Range 1 to 1 CROSS Range 127 to 255 HAS 116 HIT(S)
========================================
```

Results

- Best and worst cases:

Head insertion	2 cycles
Normal insertion	min 7 cycles and max $5 + 3 \cdot \log_k(N)$
Head removal	min 8 cycles and max $6 + 3 \cdot \log_k(N)$
Tail removal	3 cycles
Normal removal*	min 12 cycles and max $13 + 3 \cdot \log_k(N)$

*with a reference ID look up time of 1 cycle

- Average insertion and removal times of a random run:

Size	$k = 2$	$k = 4$	$k = 8$	$k = 16$
16+1	9.38/14.56	7.86/11.51	7.47/10.42	-
32+1	9.9/15.15	7.96/12.29	7.34/11.22	7.54/11.00
64+1	9.79/16.34	8.01/13.80	7.41/12.38	7.32/11.18
128+1	9.96/17.37	8.14/14.47	7.54/13.14	7.26/11.62
256+1	9.73/17.39	8.15/15.54	7.53/14.21	7.34/12.89

Questions and links

- Find the code: `https://github.com/chisel-uvm/chisel-verify/tree/master/src/main/scala/heappriorityqueue`
- Find the tests: `https://github.com/chisel-uvm/chisel-verify/tree/master/src/test/scala/heappriorityqueue`
- Questions?