

Efficient system-level support for CHERI Capabilities

Mark Hill, Distinguished Engineer & Lead CPU Architect

→ System-level requirements for CHERI

- A system using CHERI (Capability Hardware Enhanced RISC Instructions) must maintain a tag bit for every memory location that can hold a capability
- Tag and all bytes of the data in a capability must appear, by all observers in the system, to be updated atomically (single copy atomicity)
- When tag and data can be co-located maintaining atomicity is straightforward
- But sometimes they cannot...
 - Integration of tag-aware subsystems (e.g. security islands) into non-tag aware systems
 - Integration with standard, non-tag aware, system IPs: such as Dynamic Memory Controllers and Last Level Caches
- Tag and data then need to be stored in separate locations
- This is the issue that the **Capability Management Unit (CMU)** has been designed to address

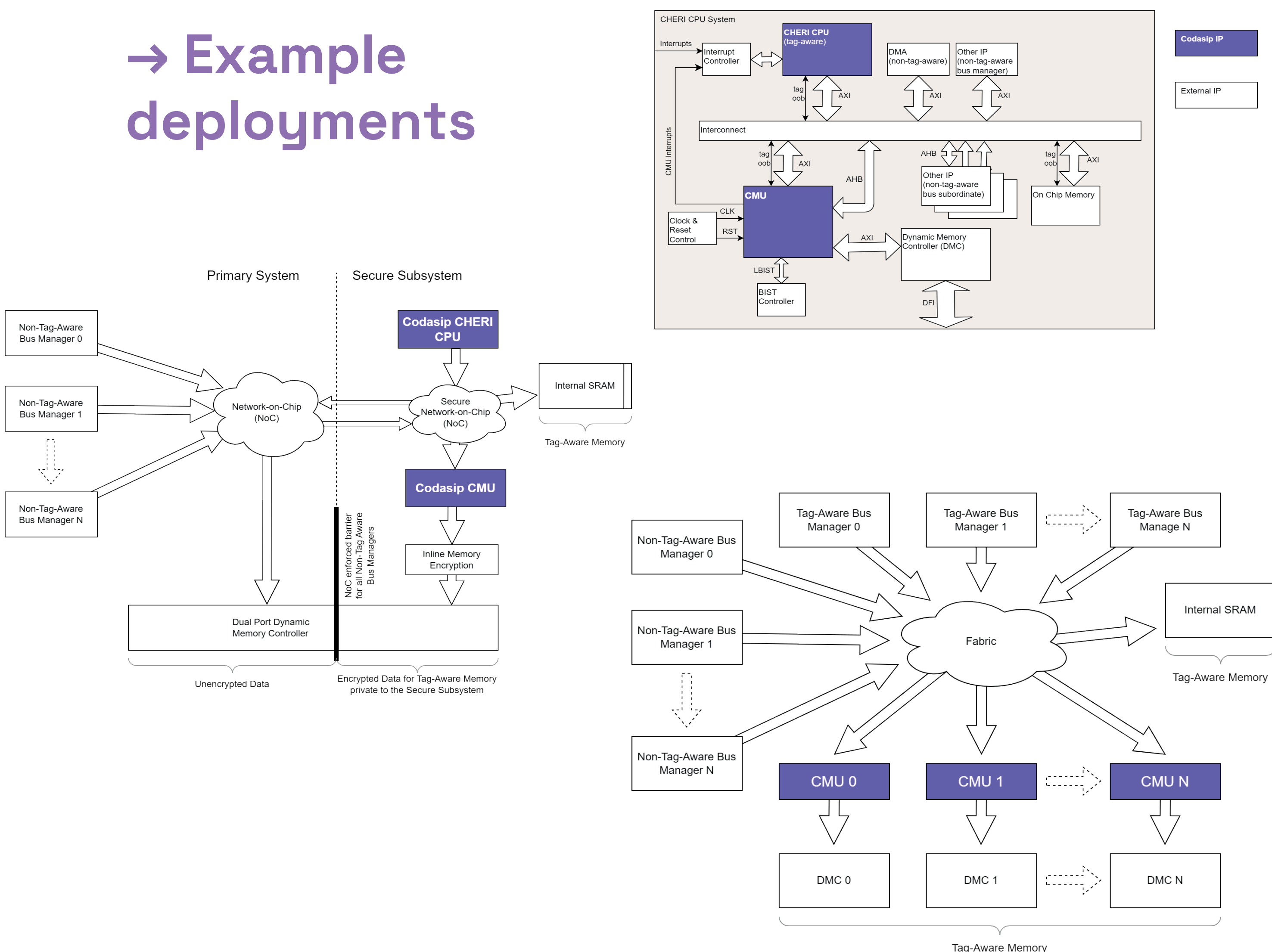
→ CMU challenges

- Preserving atomicity of tag and data stored in different locations
 - In a system with multiple bus managers and subordinates
 - Downstream bus fabric and IPs are aggressively reordering transactions (to optimize throughput)
- Minimizing overhead, both in terms of bandwidth and latency, to a level comparable to non-CHERI implementations
- Maximizing design re-use, by supporting a wide-range of use-cases

→ Maximising design re-use

- Highly configurable IP
- Supports Multiple Tag management strategies that can be configured in/out at build time and be selected at run time
- Address filtering allows strategy to be selected on a per region basis
- **Tag handling can be disabled for any regions not required to store capabilities**
- Performance, Power and Area (PPA) can be scaled and finely tuned to target system for use in everything from a small embedded systems to high performance servers

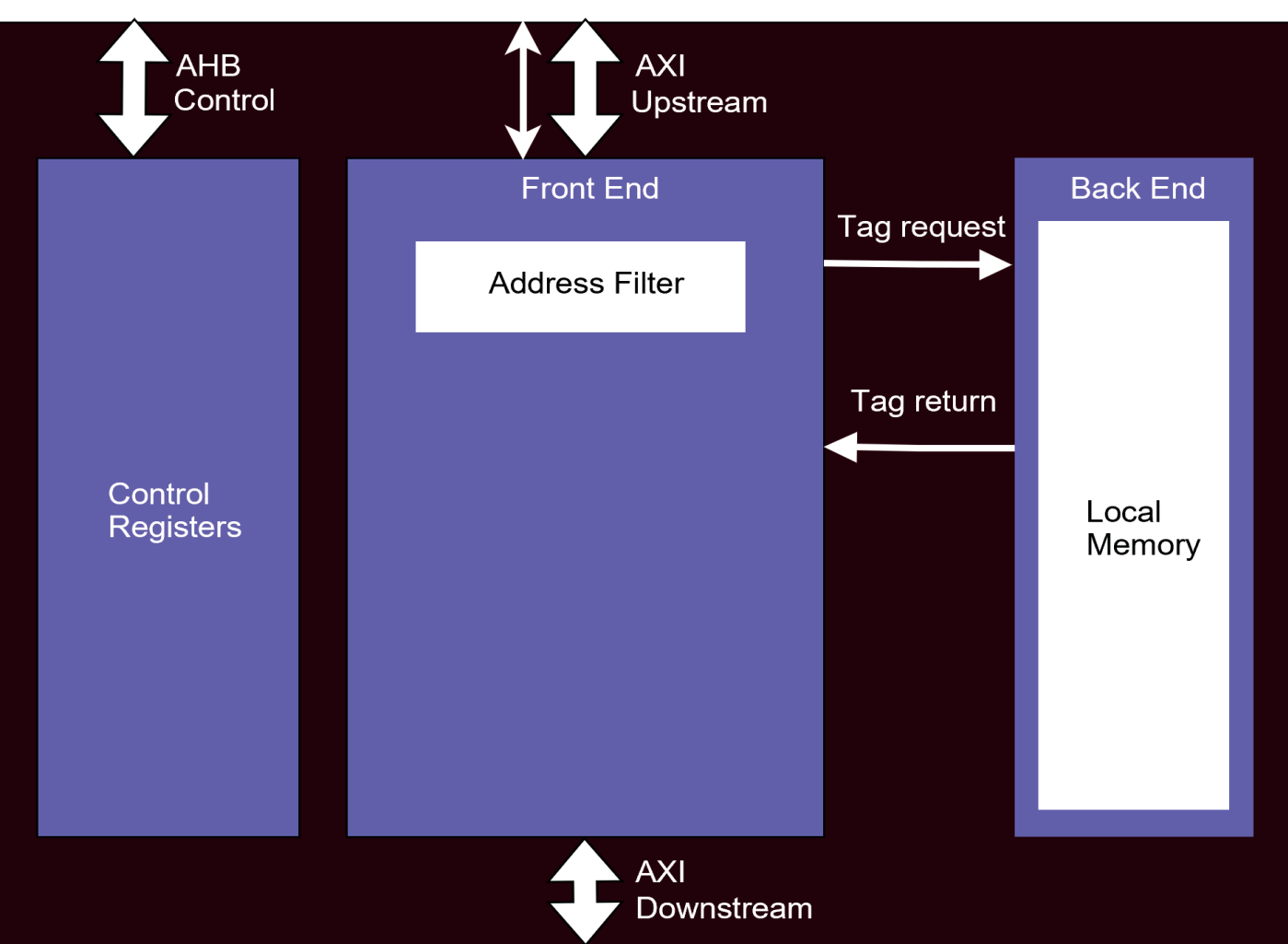
→ Example deployments



→ Tag Stash Mode

- **All tag bits held in local memory within the CMU**
- + No additional external memory requests ever generated
- Requires 1-bit of local memory for every location in external memory which can hold a capability
- On-chip memory demand gets too high for larger external memories, realistic limit is about 128Mbytes

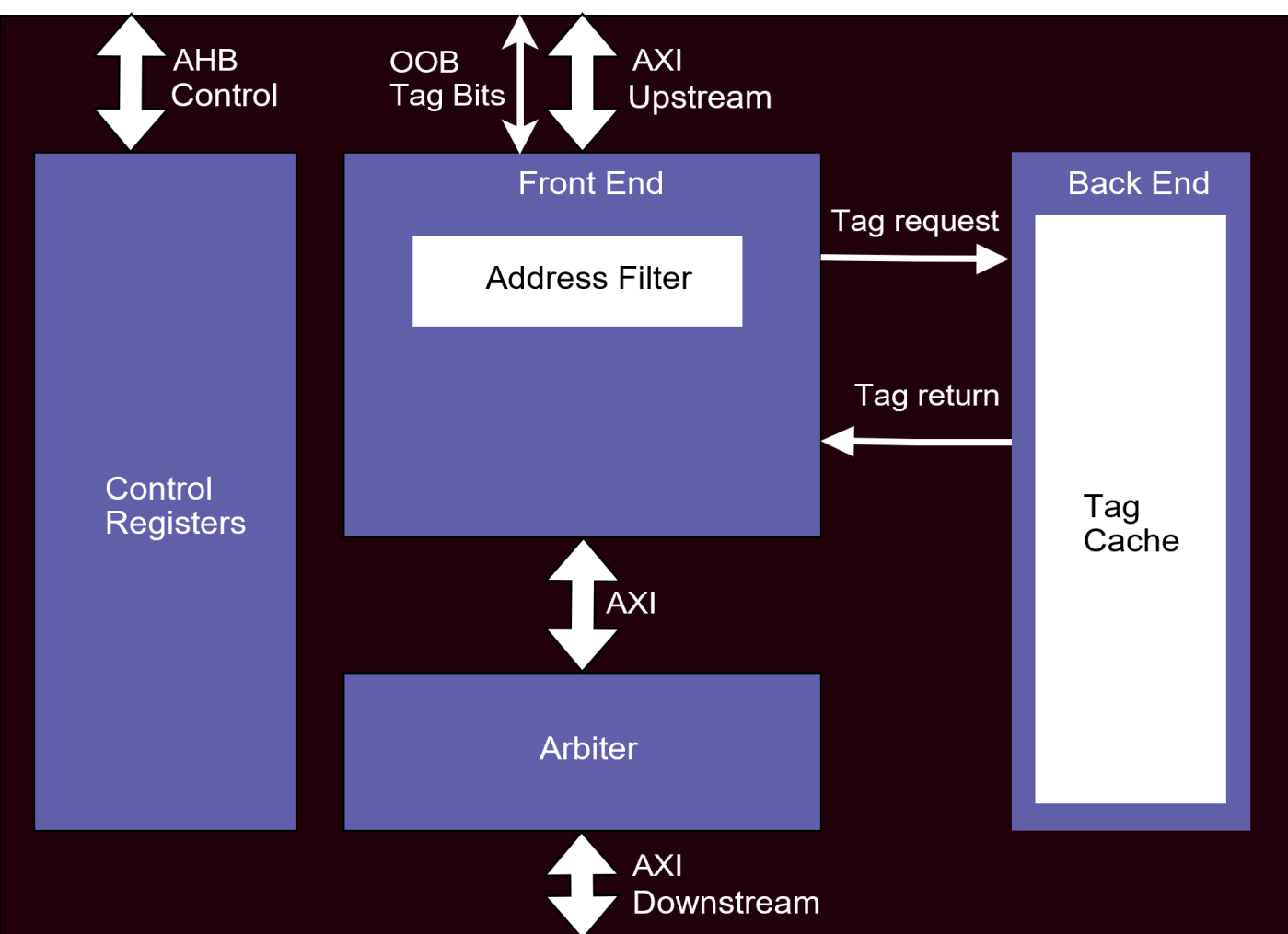
- Example use case: Tag bits for a security island requiring some off-chip capability storage



→ Tag Cache Mode

- **Tags bits for recently accessed memory stored in a tag cache**
- Each Tag Cache Line caches the tags for nominally a 4K memory block (when the line size is 256-bit and CLEN is 128-bit)
- + No limit on size of memory on which cache operates.

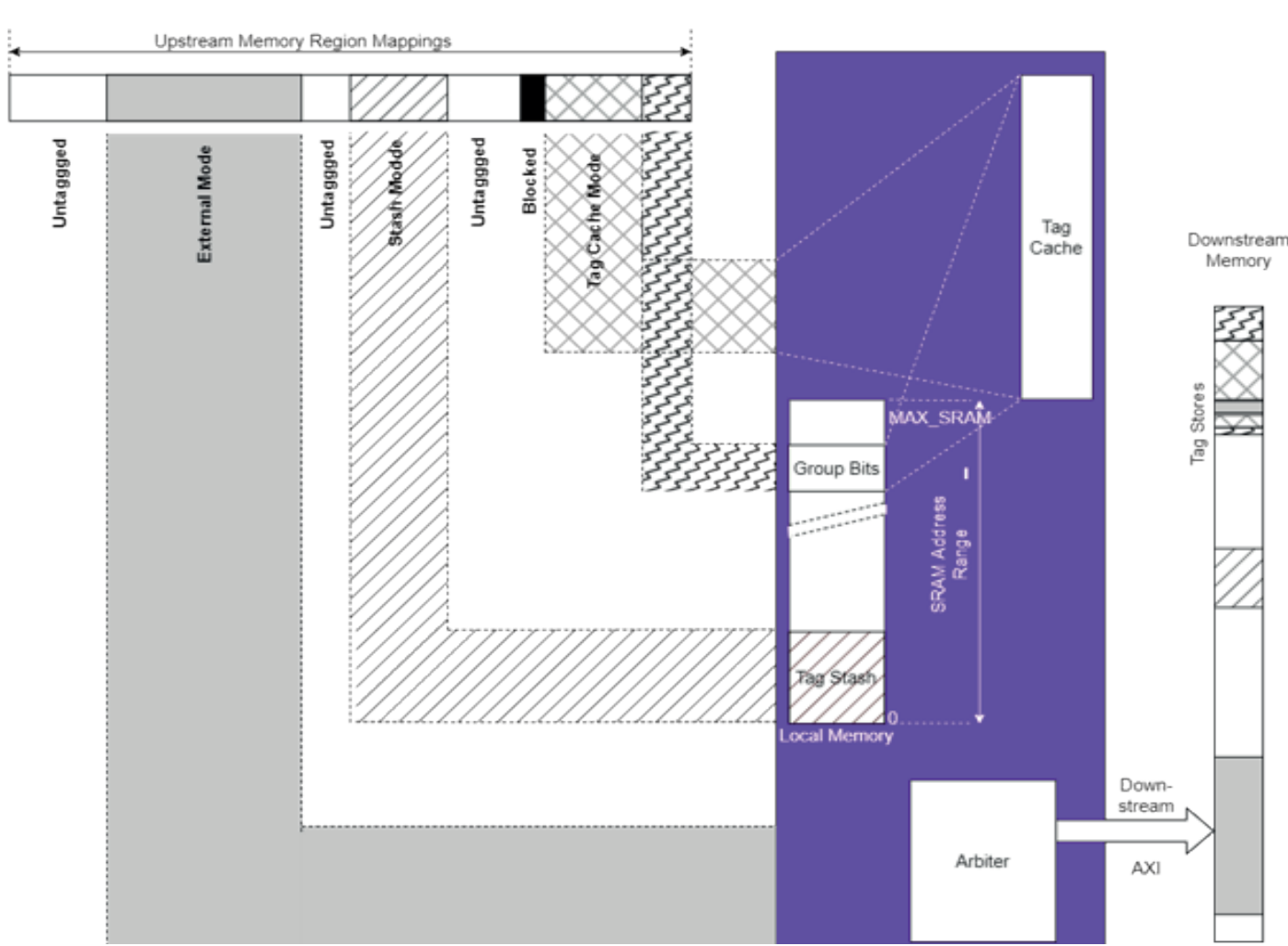
- General issues associated with caching e.g. non-determinism compared to Tag Stash Mode



→ Combining Modes: Address Filtering

- CMU has a configurable number of address filters
- Each Address Filter Table Entry (AFTE) defines a region using a NAPOT encoding scheme similar to that used in the PMP
- Unmapped regions are handled as untagged
- Tag handling strategies can be assigned on a per region basis at run-time

- Memory used as tag storage can, by default, only be accessed by the CMU



→ Status and acknowledgements

- Release now available for partner evaluation
- Evaluation Platform
 - Integrates CMU with Codasip X730 core on an FPGA board
 - Booting Linux in Tag Group Cache Mode and running Doom!
 - Platform release for partners available imminent
 - Next step integrating performance counters into Linux profiler we can perform detailed performance study
- This work acknowledges the research done at Cambridge University in this area (Efficient Tagged Memory by Alexandre Joannou et al.)