中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Load Store Unit of Xiangshan Processor

Huaqiang Wang 王华强

ICT, CAS BOSC

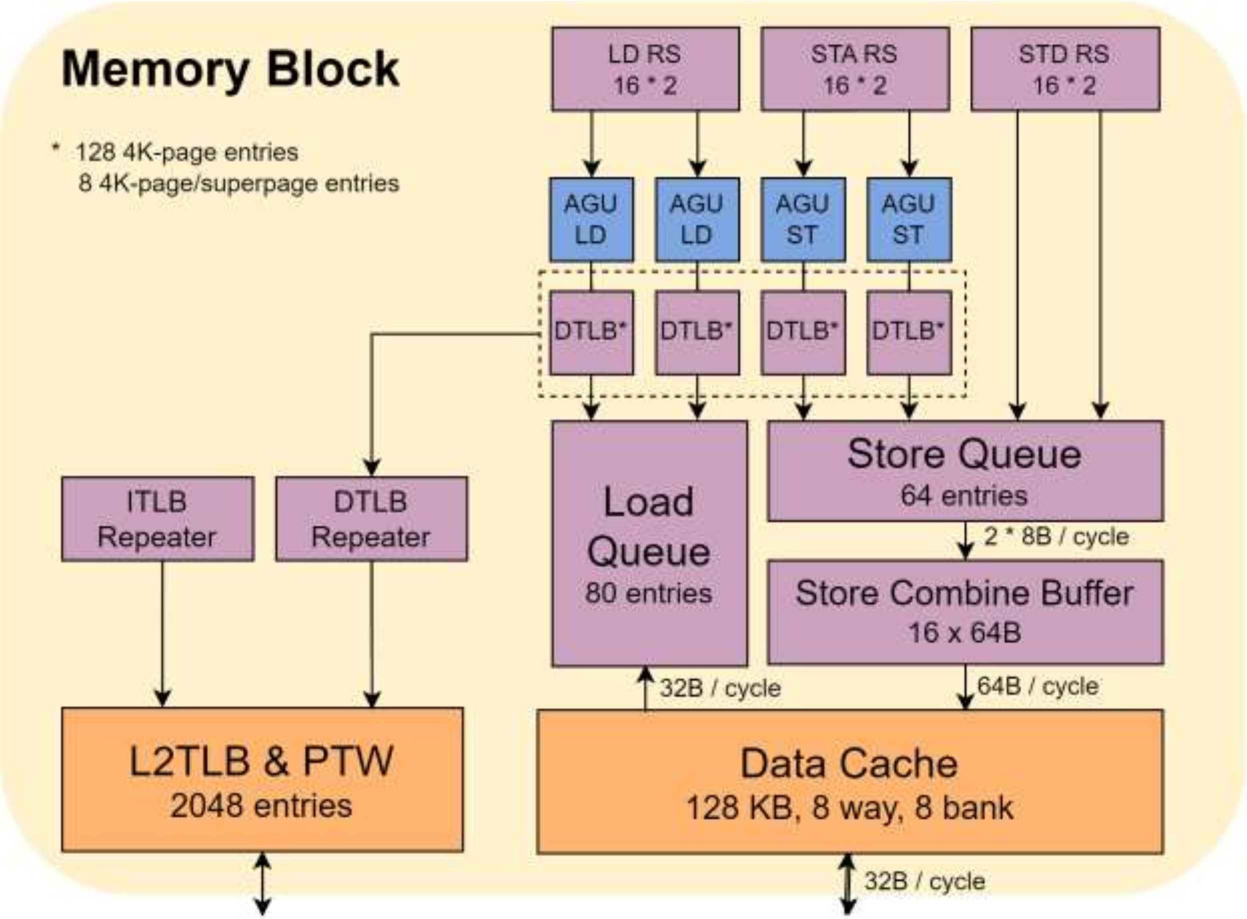
2023.4.26

Outline

- Xiangshan LSU overview
- LSU of Xiangshan Nanhu
  - Main modules
    - Load/store pipeline, load/store queue
    - Store combine buffer, Data cache
    - DTLB
  - Key Memory Access Mechanisms
    - Load miss, TLB miss, fence & atomic instructions etc.
- LSU of Xiangshan Kunminghu: a preview

LSU of Xiangshan: Overview

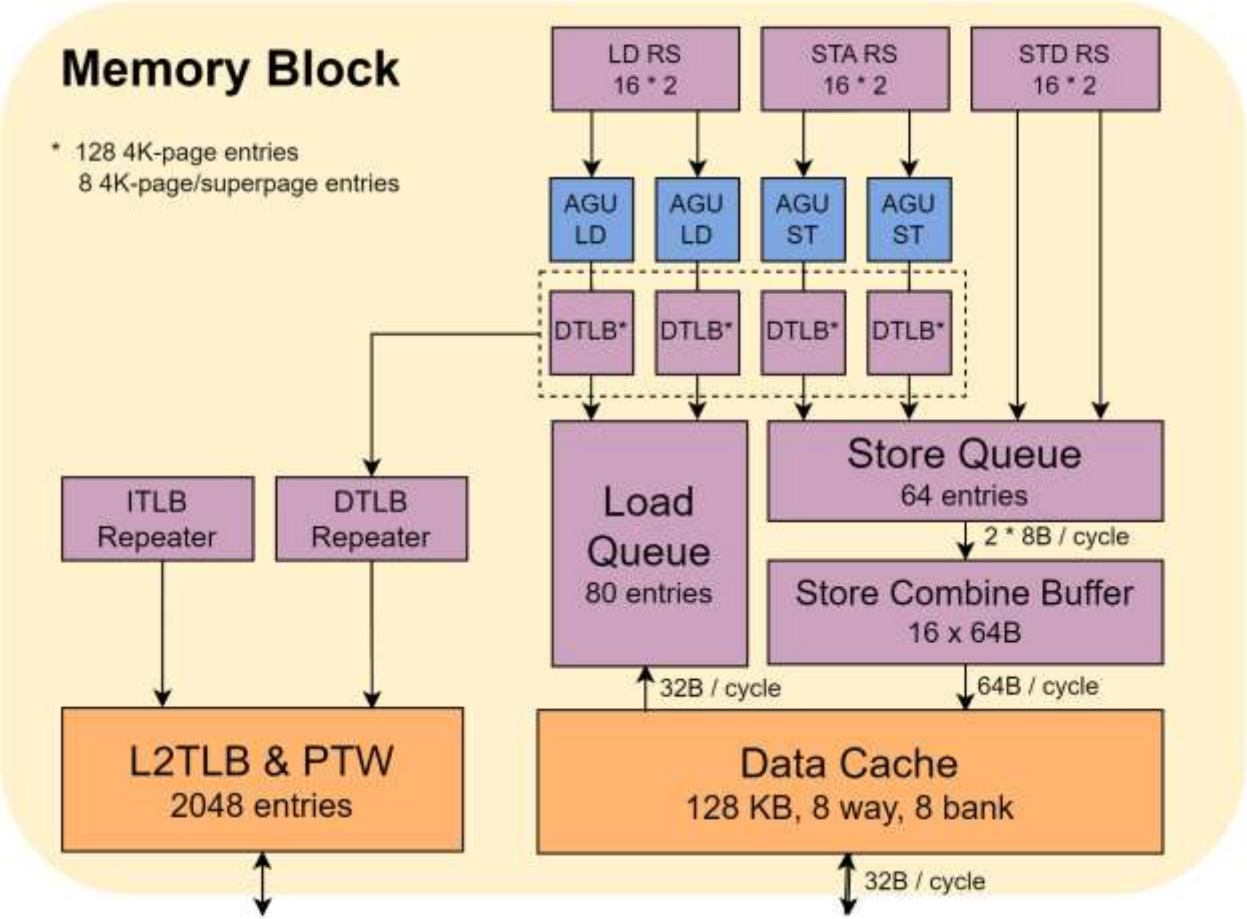
- Xiangshan Nanhu
  - Stable branch, **RTL frozen**
  - Pass **all** tests
- Xiangshan Kunminghu
  - Active develop branch
  - Pass **basic smoke tests**
  - Have many **potential bugs**
  - Many micro-architecture changes
    - compared with Nanhu
  - More like a modern commercial design
    - Read RF after issue
    - Decoupled load rar / raw / replay … queue
    - L2 hit signal directed miss replay
    - ……



Xiangshan Nanhu LSU micro-architecture

LSU of Xiangshan Nanhu: Overview

- **2** load pipeline， **4** stages
- **2** store address pipeline， **2+2** stages
- **2** store data pipeline, **2** stages
- **80** entry unified load queue
- **64** entry store queue
- **16** entry store combine buffer
  - (store coalescing buffer)
- VIPT L1 cache
  - **128** KB, **8** way or **64** KB, **4** way
  - **16** MSHRs
- RISC-V Weak Memory Ordering (RVWMO)

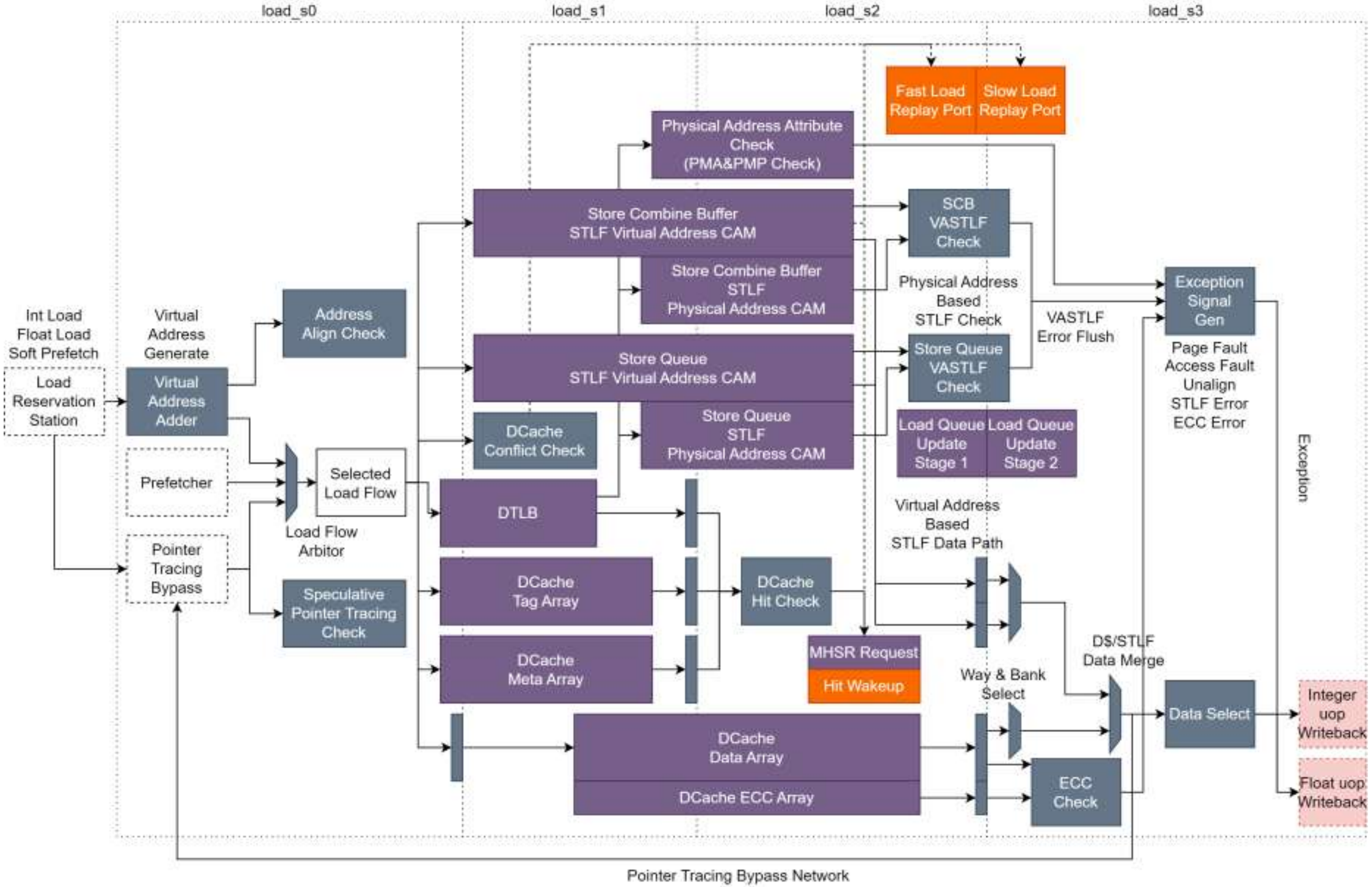


Xiangshan Nanhu LSU micro-architecture

Main modules

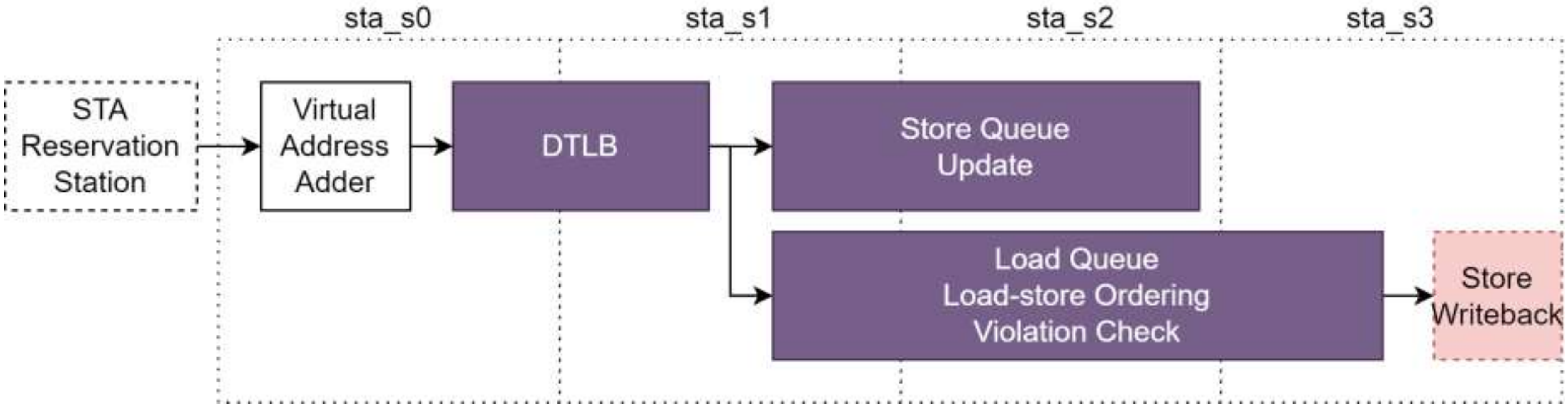
- Load/store pipeline
- Load/store queue
- Store combine buffer
- Data cache
- DTLB

Load Pipeline: Overview

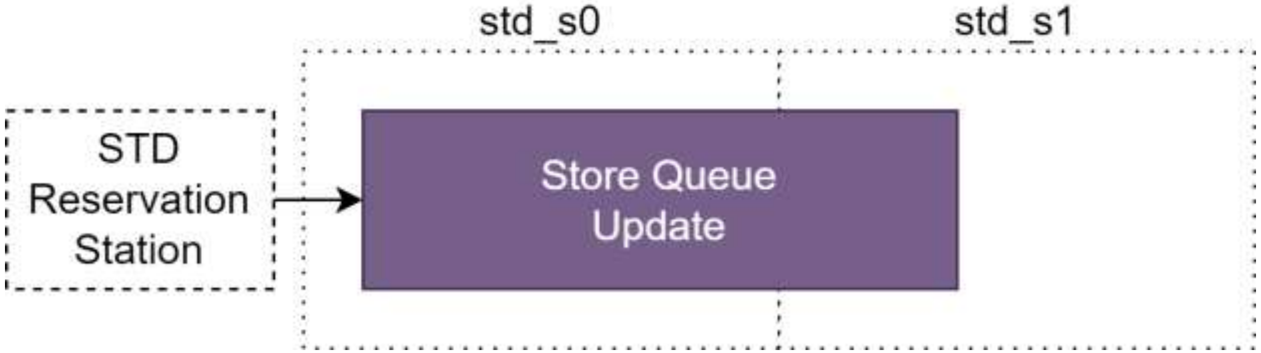


Store Pipeline: Overview

- Store address pipeline （STA）

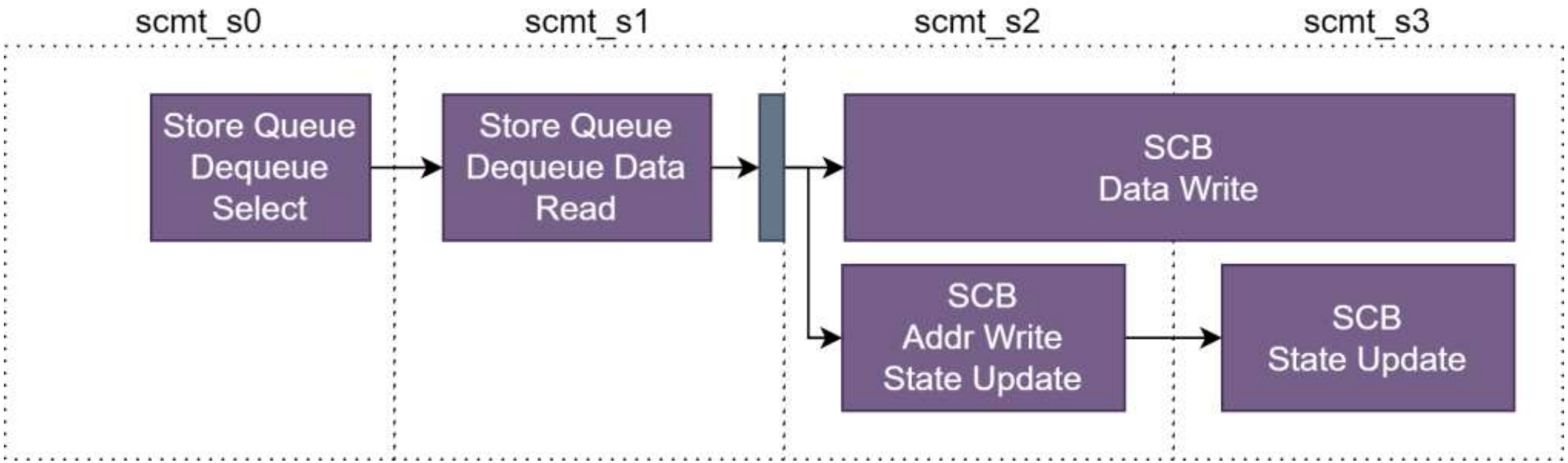


- Store data pipeline （STD）



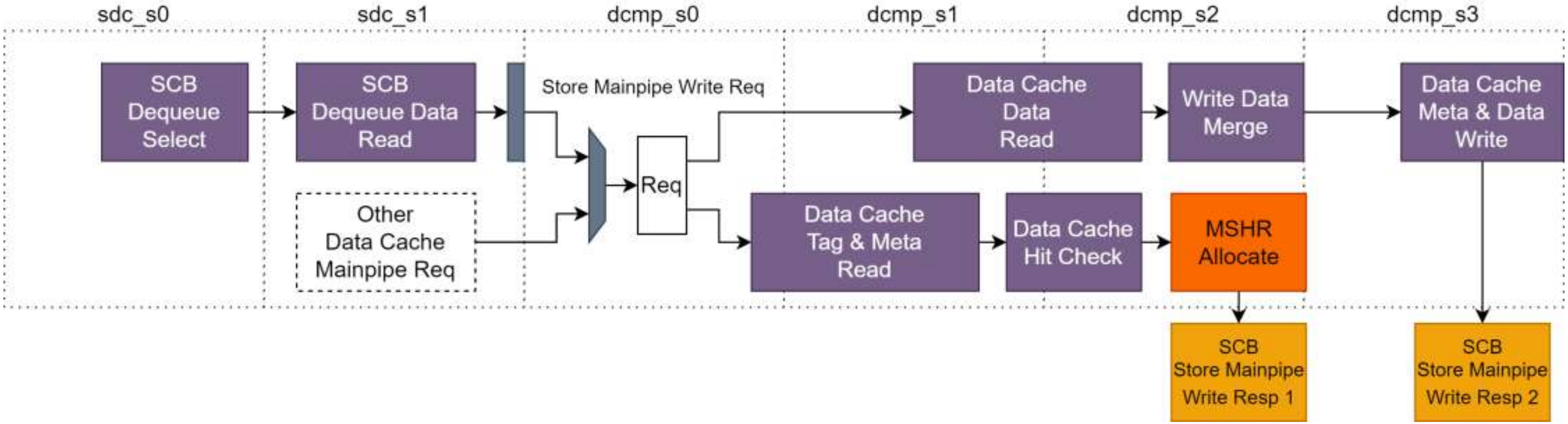
Store Pipeline: Overview

- SCB enq pipeline


scmt_s0
scmt_s1
scmt_s2
scmt_s3
Store Queue Dequeue Select
Store Queue Dequeue Data Read
SCB Data Write
SCB Addr Write State Update
SCB State Update

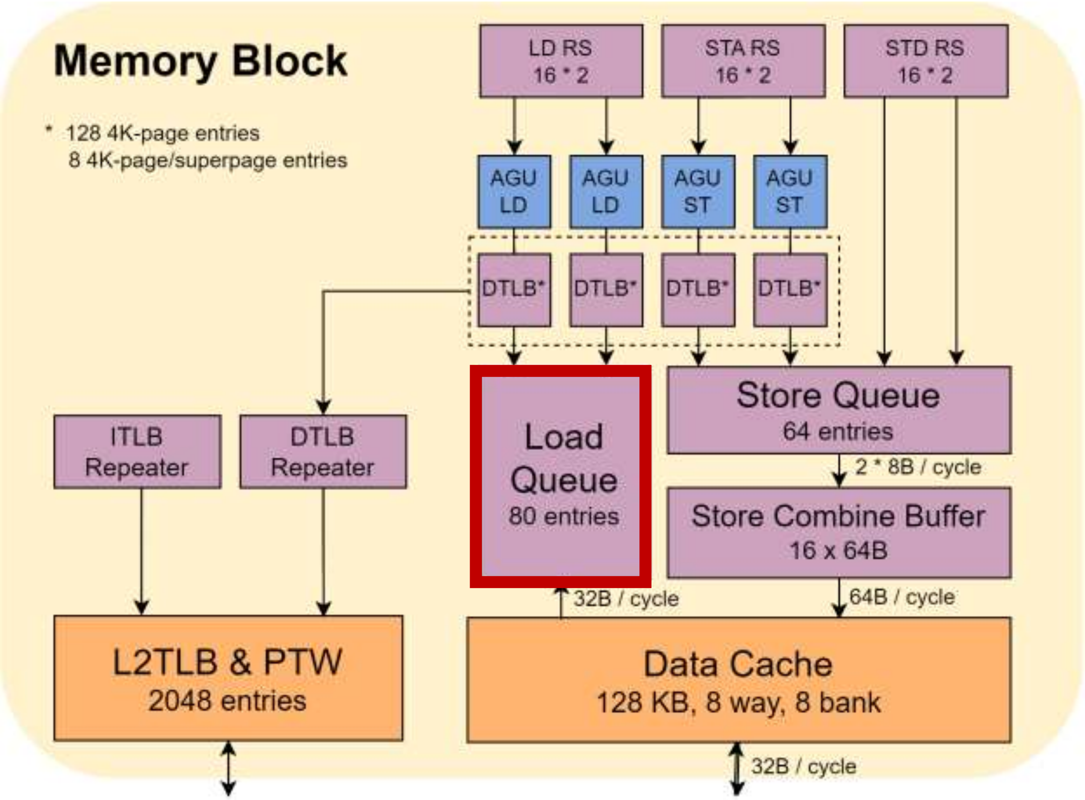
Store Pipeline: Overview

- SCB deq pipeline



Load Queue in Nanhu

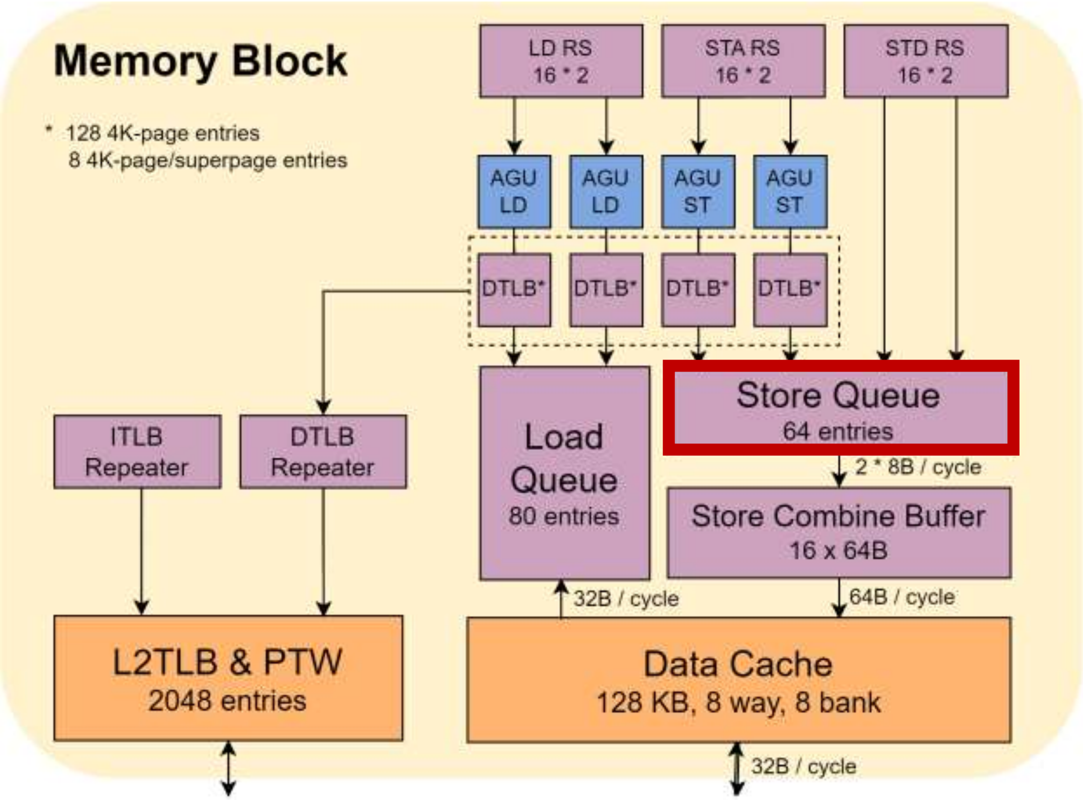
- **80** entry **unified** queue
  - Replay missed load
- For each cycle：
  - Allocate 2 entries @ dispatch
  - 2 insts from load pipe can **update** LQ
  - Replay 2 missed load insts
    - Missed, but now successfully refilled
    - Priority： LQ < load RS



- L1 data cache refill half a cache line to load queue every cycle
- Load queue parse refilled addr. & data, then replay missed load with

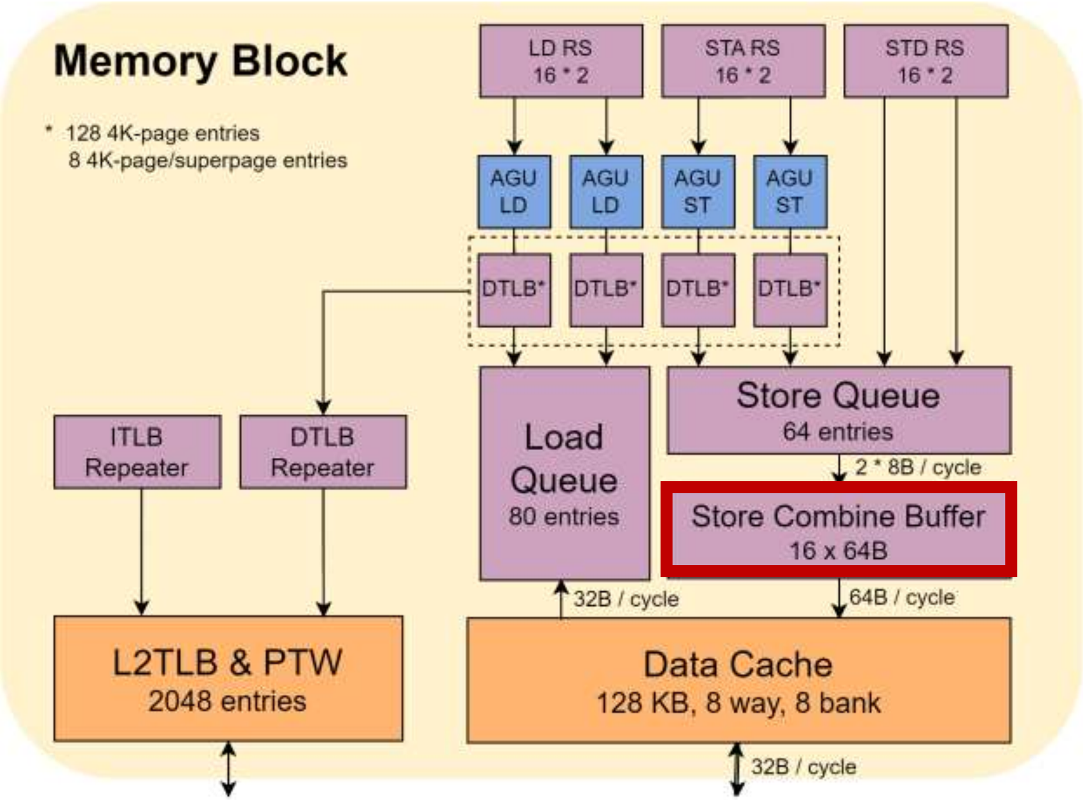
Store Queue in Nanhu

- **64** entry queue, 64 bit per entry
- For each cycle:
  - Allocate 2 entry @ dispatch stage
  - Update info for 2 insts from store pipe
  - Write up to 2 store insts to SCB (2x64 bit)
- Provide STLF result for 2 load pipelines



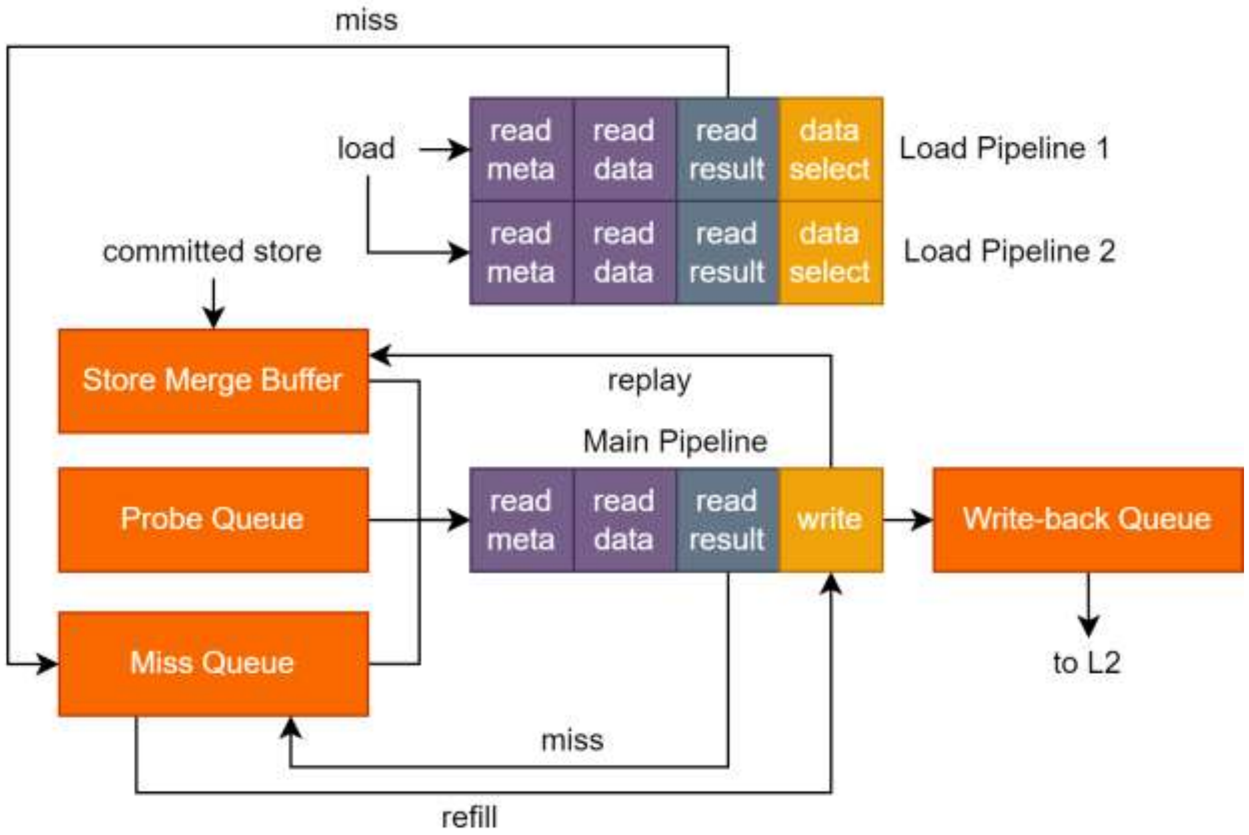
Store Combine Buffer in Nanhu

- **16** entry
- 512 bit (cacheline size) per entry
- For each cycle:
  - Receive up to 2 store (2x64 bit)
  - Write 1 entry (512 bit) to dcache
  - (Support “set cacheline to 0” inst.)
- Provide STLF result for 2 load pipelines
- Missed store will wait in SCB if dcache MSHR is full

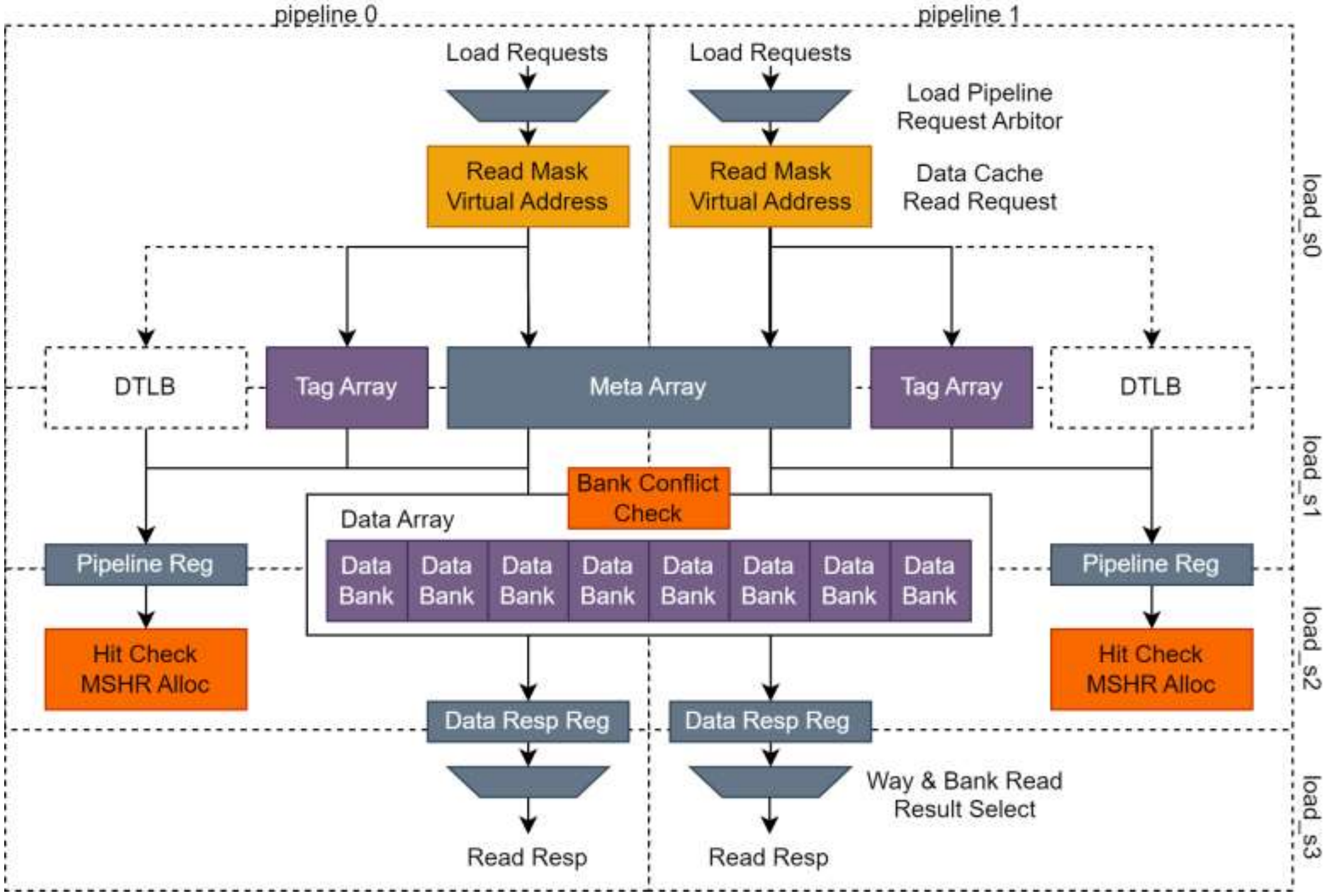


Data Cache

- Parameters
  - 64KB， 4-way or 128KB， 8-way
  - 8 banks
  - 2x64 bit load port
  - 1x512 bit store port (for SCB)
  - 16 MSHRs (Miss Queue)
- Load pipeline
  - Load and dcache prefetch
- Main pipeline
  - Replace, probe(snoop), store etc.



Load Pipeline in Data Cache

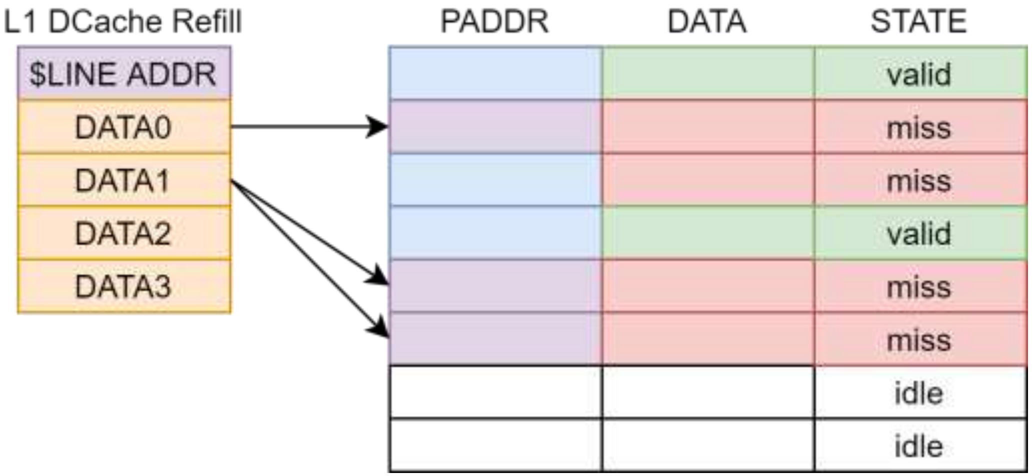


Key Memory Access Mechanisms

- Data cache miss: replay from LQ, refill 256 bits/cycle
- TLB miss: replay from reservation station
- STLF: vaddr based STLF (VASTLF, VACAM)
  - Vaddr based fast path for forwarding
  - Paddr based slow path for result check & recovering
- Memory dependency predict
  - Store Sets like MDP
- Atomic instructions / fence
  - Block backend, always do atomic operation at dcache
  - No barrier-load OoO

DCache Miss Refill in Nanhu

- If load miss, listen for L1 DCache refill result in load queue



L1 DCache Refill

$LINE ADDR
DATA0
DATA1
DATA2
DATA3

PADDR	DATA	STATE
		valid
		miss
		miss
		valid
		miss
		miss
		idle
		idle

Nanhu load queue refill

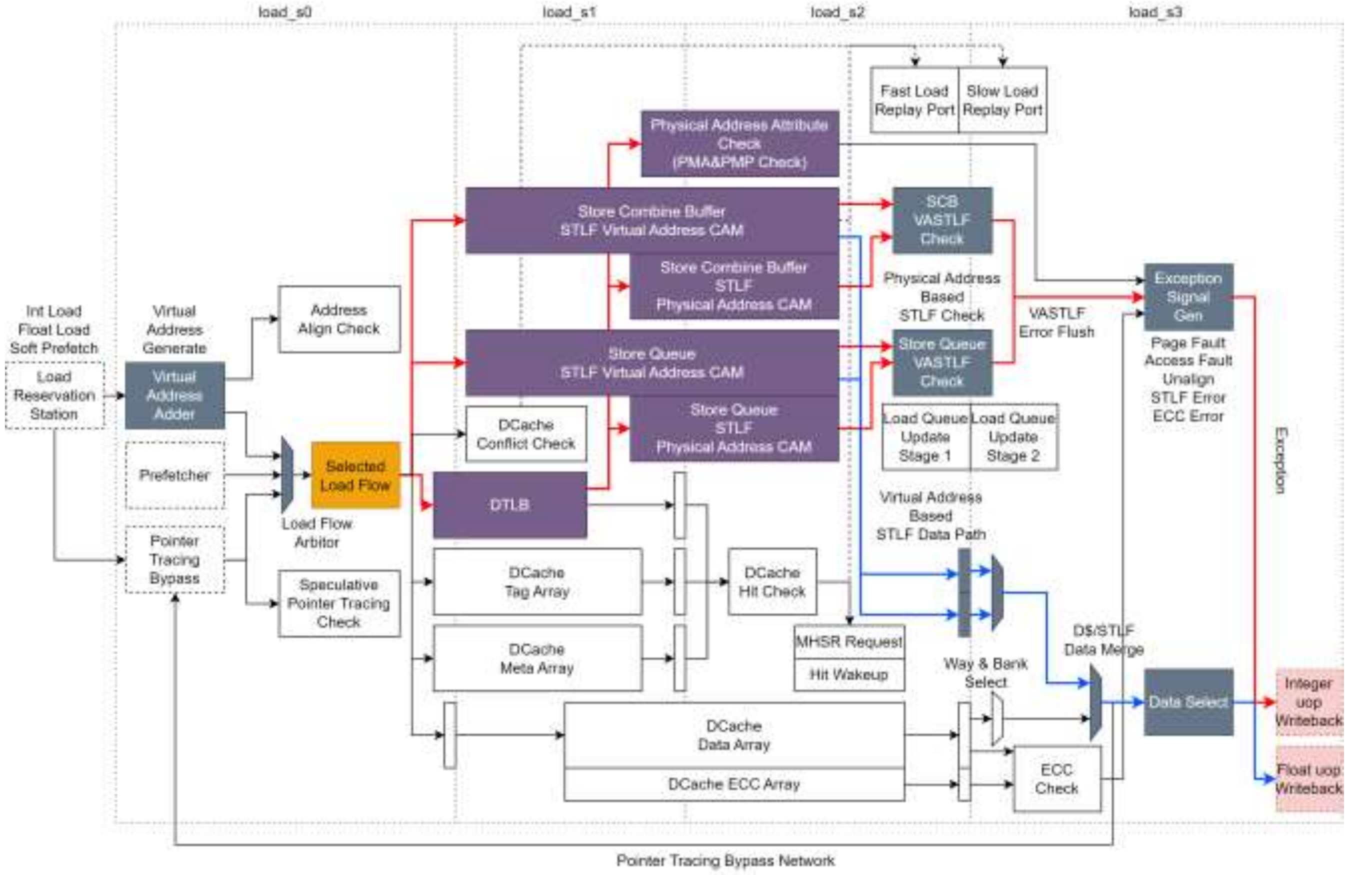
Replay from reservation station

- Instructions remain in reservation station (RS) after issue until:
  - Load / store pipeline report replay will not be needed
- Nanhu LSU request replay from reservation station when:
  - TLB miss
  - L1 data cache MSHR full / bank conflict
  - Store address valid but data is still invalid yet
- Otherwise, if there nothing wrong:
  - Nanhu LSU inform RS that inst. can leave it safely

Store to Load Forward (STLF)

- Virtual address based STLF
  - (VASTLF / VACAM)
- Virtual address based address match & data gen.
  - ***Vaddr Match, Paddr Fix***
- Use physical address to check if forward is right later

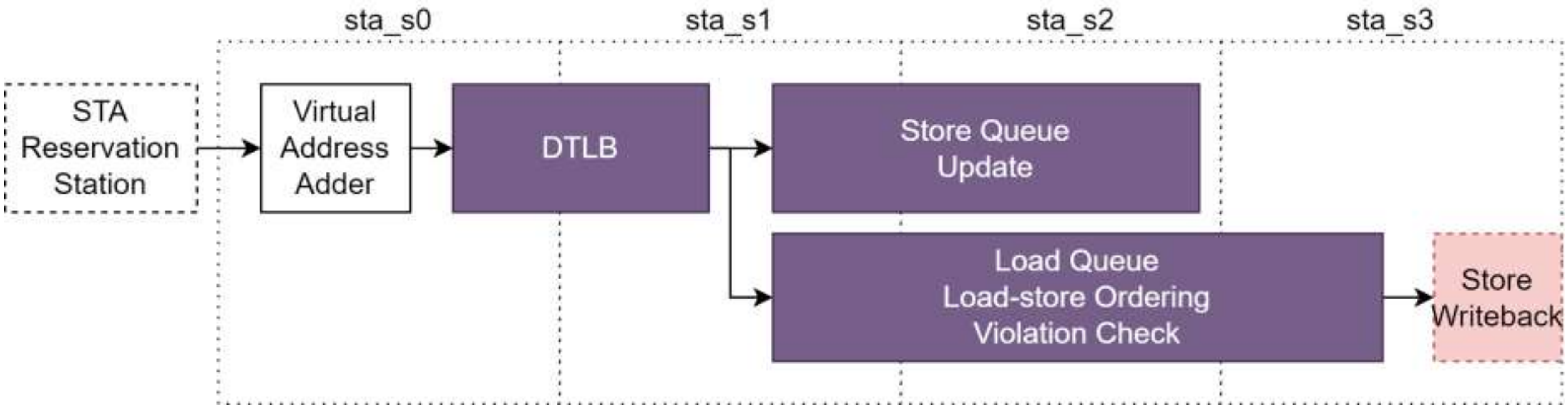
- No TLB lookup in STLF data path
- TLB lookup is in STLF control path, which is less critical



STLF in Nanhu load pipeline.
Red line: data path. Blue line: control path.

Load-store ordering Violation

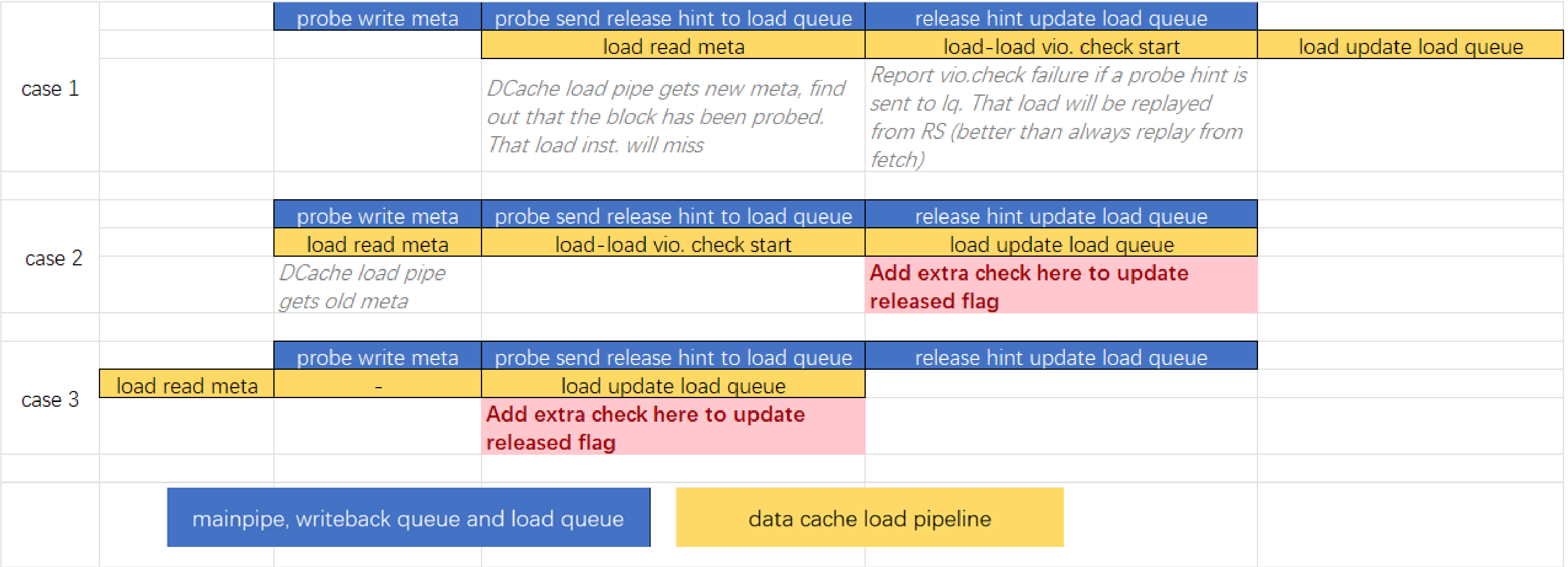
- Check
  - Start check when store arrives at store address stage 1



- Error recover
  - Rollback **immediately** if we found a conflict
  - No need to wait for instruction commit

Load-load ordering Violation

- load-load ordering check uses a released flag in LQ
- *Released* flag in LQ update is pipelined for better timing



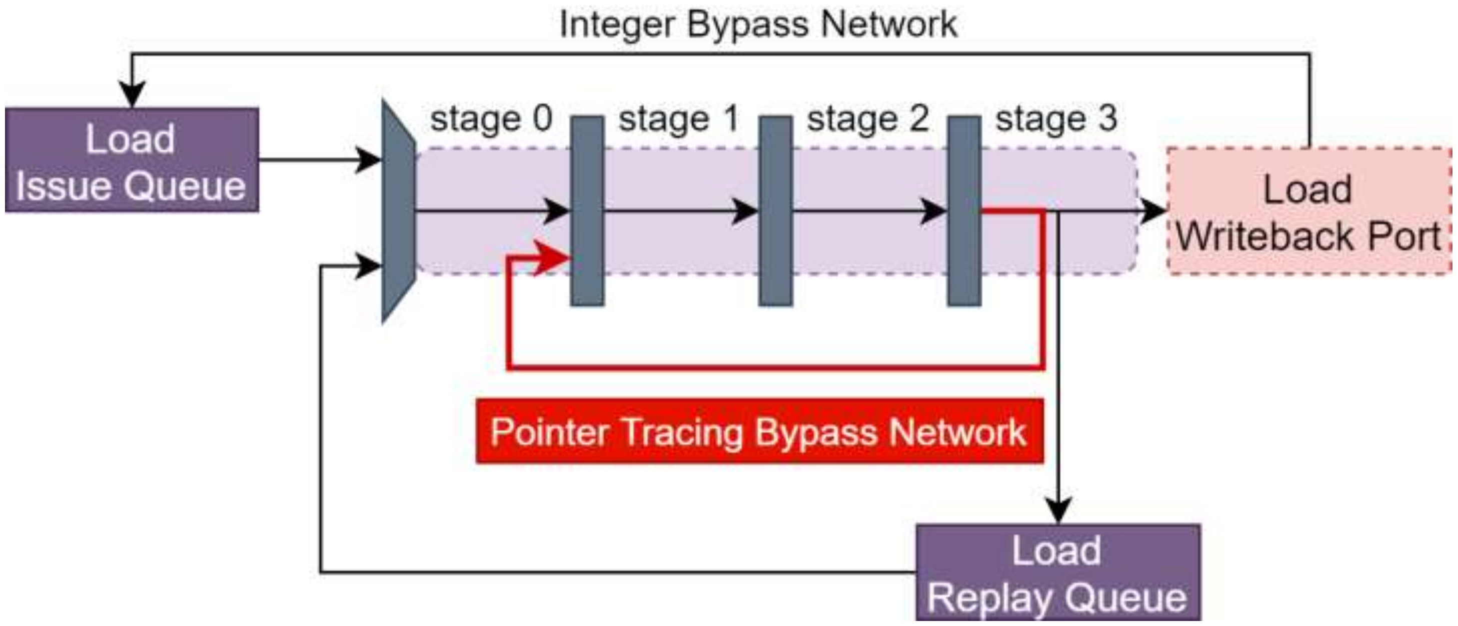
case 1		probe write meta	probe send release hint to load queue	release hint update load queue	
			load read meta	load-load vio. check start	load update load queue
			DCache load pipe gets new meta, find out that the block has been probed. That load inst. will miss	*Report vio.check failure if a probe hint is sent to lq. That load will be replayed from RS (better than always replay from fetch)*	
case 2		probe write meta	probe send release hint to load queue	release hint update load queue	
		load read meta	load-load vio. check start	load update load queue	
		DCache load pipe gets old meta		**Add extra check here to update released flag**	
case 3		probe write meta	probe send release hint to load queue	release hint update load queue	
	load read meta	-	load update load queue		
			Add extra check here to update released flag		

🟦 mainpipe, writeback queue and load queue

🟨 data cache load pipeline

Pointer Tracing

- Pointer Tracing Bypass Network reduces load to load latency from 4 to 3



Memory Dependence Prediction

- Predict memory dependence near dispatch stage using PC
  - *Store Sets like MDP*
- Predictor supported
  - Load Wait Table[1]
  - Store Sets[2]
- If the predictor believes that load violation may happen
  - load wait in reservation station until all former store is finished

[1] Kessler R E . The Alpha 21264 Microprocessor[J]. IEEE Micro, 1999, 19(2):24-36.
[2] Chrysos G Z , Emer J S . Memory Dependence Prediction using Store Sets[J]. ACM SIGARCH Computer Architecture News, 2002, 26(3):142-153.

Atomic / fence Instructions

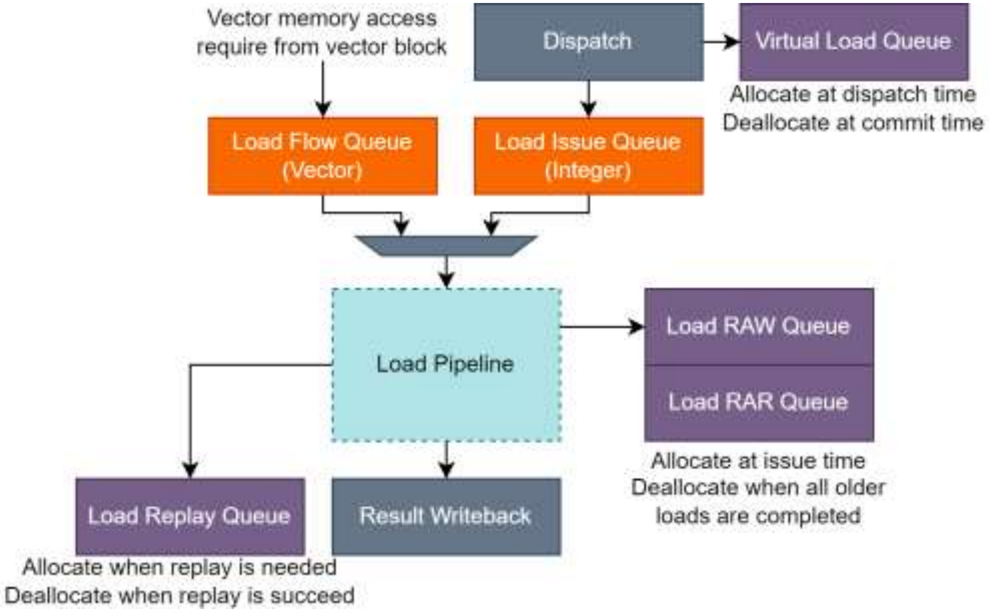
- For now, there is **no performance optimization** for atomic instructions / fence in Xiangshan

- Fence will **always drain** SB & SCB
- Atomic inst. / fence will **not issue** until all former insts has been committed
- Issued atomic inst. / fence will **block** backend

- Xiangshan Nanhu **always** do atomic operation at data cache
- No out of order barrier-load execution

Optimization in Kunminghu

- LQ split
- L2 hit signal directed miss replayVector



Vector memory access require from vector block
Dispatch
Virtual Load Queue
Allocate at dispatch time
Deallocate at commit time
Load Flow Queue (Vector)
Load Issue Queue (Integer)
Load Pipeline
Load RAW Queue
Load RAR Queue
Allocate at issue time
Deallocate when all older loads are completed
Load Replay Queue
Result Writeback
Allocate when replay is needed
Deallocate when replay is succeed

Q & A

- Load Queue / Store Queue / Store Combine Buffer
- Store to Load Forwarding (STLF)
- Load violation check
- Load violation predict / delay
- Uncached memory access （MMIO）
- Dcache miss / TLB miss
- Memory inst. replay
- Commit / exception