Intel® OpenSource HD Graphics Programmer's Reference Manual (PRM)
Volume 1 Part 7: L3$/URB
(Ivy Bridge)

For the 2012 Intel® Core™ Processor Family

May 2012

Revision 1.0

NOTICE:
This document contains information on products in the design phase of development, and Intel reserves the right to add or remove product features at any time, with or without changes to this open source documentation.

Doc Ref #: IHD-OS-V1 Pt 7 – 05 12

Creative Commons License

You are free to Share — to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

No Derivative Works. You may not alter, transform, or build upon this work.

Contents

1. L3$/URB

1.1 Overview

GFX L3 cache is introduced for GFX core as a large storage which backs up various L2/L1 caches on many clients. It provides a simple way based partitioning option for each or a cluster of clients to get a dedicated chunk of the cache. It also acts as a GFX URB and can be configured as highly banked memory for EUs/ROWs.

In order to provide the bandwidth needed L3 has been separated into 4x128KB structures which can be accessed concurrently. A 2x clocking is introduced to further enhance the bandwidth and cover the limitations of SRAM (6T) design.

- Formed as 4 (2 for GT1) individual banks each with 128KB in size
- Each logical bank consists of
    - Data Array
    - Tag Array
    - LRU Array (implements a Pseudo Least Recently Used algorithm)
    - State Array
    - SuperQ Data Buffer
    - Atomic Processing Unit
- The rest of the support logic around L3 are
    - SuperQ (main scheduler)
    - Ingress/Egress queues to L3/SQ (L3 arbiter)
    - CAM structures to maintain coherency.
    - Crossbars for data routing
- Use of 2x/1x clocking
- L3 operates in GFX coherent domain
- A portion of L3 can be allocated as highly banked memory

1.2 L3$ Cache Configuration

- 4x128KB cache, 64 logical ways
- 64B Cacheline with a portion capable of highly banked memory (with 16x4B capability)
- Interface 64B to SQDB for the fill/write path, 64B Read/Evict path to SQDB
- Data Array built via 6T cells
    - Data protection via parity
- TAG/LRU/STATE (using gen-ram via RLS flows)
    - 32-bit GFX addressing support in TAG

- 2 bit state
- Intel pseudo-LRU implementation for selecting the line to be replaced

- Repetition rates for each operation
  - All operations – 1 every 2x clock
    - With b2b restriction for same type of accesses (i.e. read to read or write to write)

1.3 Memory Object Control State on Cacheability

This 4-bit field is used in various state commands and indirect state objects to define MLC/LLC cacheability, graphics data type for memory objects.

Bit	Description
3	**Reserved**
2	**Graphics Data Type (GFDT)** This field contains the GFDT bit for this surface when writes occur. GFDT can also be set by the GTT. The effective GFDT is the logical OR of this field with the GFDT from the GTT entry. This field is ignored for reads. The GFDT bit is stored in the LLC and selective cache flushing of lines with GFDT set is supported. It is intended to be set on displayable data, which enables efficient flushing of data to be displayed after rendering, since display engine does not snoop the rendering caches. Format = U1
1	**LLC Cacheability Control (LLCCC)** This is the field used in GT interface block to determine what type of access need to be generated to uncore. For the cases where the LLCCC is set, cacheable transaction are generated to enable LLC usage for particular stream. 0: use cacheability controls from GTT entry 1: Data is cached in LLC
0	**L3 Cacheability Control (L3CC)** This field is used to control the L3 cacheability (allocation) of the stream. 0: not cacheable within L3 1: cacheable in L3 *Note: even if the surface is not cacheable in L3, it is still kept coherent with L3 content.*

2. Atomics

An atomic operation may involve both reading from and then writing to a memory location. Atomic operations apply only to either u# (Unordered Access Views) or g# (Thread Group Shared Memory). It is guaranteed that when a thread issues an atomic operation on a memory address, no write to the same address from outside the current atomic operation by any thread can occur between the atomic read and write.

If multiple atomic operations from different threads target the same address, the operations are serialized in an undefined order. This serialization happens outside of the L3 control logic.

Atomic operations do not imply a memory or thread fence. If the program author/compiler does not make appropriate use of fences, it is not guaranteed that all threads see the result of any given memory operation at the same time, or in any particular order with respect to updates to other memory addresses.

Atomicity is implemented at 32-bit granularity. If a load or store operation spans more than 32-bits, the individual 32-bit operations are atomic, but not the whole.

Limitation: Atomic operations on Thread Group Shared Memory are atomic with respect to other atomic operations, as well as operations that only perform reads (“load”s). However atomic operations on Thread Group Shared Memory are NOT atomic with respect to operations that perform only writes (“store”s) to memory. Mixing of atomics and stores on the same Thread Group Shared Memory address without thread synchronization and memory fencing between them produces undefined results at the address involved. This limitation arises because some implementations of loads and stores do not honor the locking semantics for implementing atomics. It turns out this has no impact on loads, since they are guaranteed to retrieve a value either before or after an atomic (they will not retrieve partially updated values, given they are all defined at 32-bit quanta). However store operations could find their way into the middle of an atomic operation and thus have their effect possibly lost.

In L3 or SLM, the atomic operation leads to a read-modify-write operation on the destination location with the option of returning value back to requester. The table below is defined as a list of atomic operations needed:

Atomic Operation	Description	New Destination Value	Return Value (optional)
Atomic_AND	Single component 32-bit bitwise AND of operand src0 into dst at 32-bit per component address dstAddress, performed atomically.	“old_dst” AND “src0”	old_dst
Atomic_OR	Single component 32-bit bitwise OR of operand src0 into dst at 32-bit per component address dstAddress, performed atomically.	“old_dst” OR “src0”	old_dst
Atomic_XOR	Single component 32-bit bitwise XOR of operand src0 into dst at 32-bit per component address dstAddress, performed atomically.	“old_dst” XOR “src0”	old_dst
Atomic_MOVE	Replacement of the dst with src0.	“src0”	old_dst
Atomic_INC	Single component 32-bit integer increment of dst back into dst	“old_dst + 1”	old_dst
Atomic_DEC	Single component 32-bit integer decrement of dst back into dst	“old_dst - 1”	old_dst
Atomic_ADD	Single component 32-bit integer add of operand src0 into dst at 32-bit per component address performed atomically. Insensitive to sign	“old_dst + src0”	old_dst

Atomic Operation	Description	New Destination Value	Return Value (optional)
Atomic_SUB	Single component 32-bit integer subtraction of operand src0 into dst at 32-bit per component address performed atomically. Insensitive to sign	“old_dst - src0”	old_dst
Atomic_RSUB	Single component 32-bit integer subtraction of operand dst from src0 into dst at 32-bit per component address performed atomically. Insensitive to sign	“src0 - old_dst”	old_dst
Atomic_IMAX	Single component 32-bit signed MAX of operand src0 into dst at 32-bit per component address dstAddress, performed atomically.	IMAX (old_dst, src0)	old_dst
Atomic_IMIN	Single component 32-bit signed MIN of operand src0 into dst at 32-bit per component address dstAddress, performed atomically.	IMIN (old_dst, src0)	old_dst
Atomic_UMAX	Single component 32-bit unsigned MAX of operand src0 into dst at 32-bit per component address dstAddress, performed atomically.	UMAX (old_dst, src0)	old_dst
Atomic_UMIN	Single component 32-bit unsigned MIN of operand src0 into dst at 32-bit per component address dstAddress, performed atomically.	UMIN (old_dst, src0)	old_dst
Atomic_CMP/WR	Single component 32-bit value compare of operand src0 with dst at 32-bit per component address dstAddress If the compared values are identical, the single-component 32-bit value in src1 is written to destination memory, else the destination is not changed The entire compare+write operation is performed atomically	(src0 == old_dst)? src1: old_dst	old_dst
Atomic_PREDEC	Single component 32-bit integer decrement of dst back into dst	“old_dst - 1”	new_dst

Atomics for unstructured data types will take place in L3each via a single DW atomics engine. Each L3 bank will have 2 independent atomics engines, one for L3 and one for SLM. Both L3 or SLM atomic requests are going to be reduced to a single index (1 DW) request by DC before getting pushed to L3. In L3, a single 32bit atomics engine is suffice to service DC requests.

The DC request for atomic will have the proper DW only byte enables set for the 32 bit of interest. The address down to bit[2] (dword address) will also be provided to point to correct DW out of 16 lanes in 64Bytes.

The processing of atomics will follow 2 separate pipelines of operation (either SLM or L3) depending on the destination of the access.

2.1 Atomics in L3

Atomics in L3 are handled separately in each bank, to achieve this function 2 Atomics blocks are instantiated along with each bank. Each operand being moved to SQ also moves its data (up to 2 DWs) into an assigned atomics block to be used later on (when he destination data is available).

A separate credit is given to L3 arbiter for atomics, once an atomic request is moved from L3 arbiter to SQ – both the SQ credits and Atomics credits need to be deducted to regulate the number of atomic requests in SQ. For GT2, this process allows upto 8 atomics to be performed in a given clock.

The request interface allows only 1 DW of atomics per request, data from DC will be given on DW0 (also in DW1 if src1 is given) for all atomic operations regardless of the address of byte enables. Cacheline address will be provided on the interface with proper Byte Enables singling the DW location of the destination.

If final data is returned to client (optional), the DW of interest will be given in the same position pointed out by byte enables (in fact the same DW will be replicated over 16 positions).

2.2 Atomics in SLM

SLM pipeline has a mechanism to handle atomics similar to L3/URB pipeline. There is only 1 ALU per SLM bank. The protocol between DC and L3 allows one atomics to be performed at a given time, the SLM controller will stall the interface if needed. Per atomics request from the DC, only ONE DW can be active on one SLM bank. SLM pipeline can execute b2b atomics request (1 every 1x clock) as long as b2b operations do not conflict on the same bank. If conflict is detected a single clock of bubble is inserted into pipeline in order to update the corresponding bank with SLM output before next operation can be performed (*see SLM pipeline details*)

Data from DC will be always given on DW0 and DW (if needed) and VALIDs will point to the bank of interest out of 16 banks of SLM. Correct set of byte enables should be provided which is active for the valid bank.

DW of interest is returned to DC on the byte enable corresponding lane of the cacheline.

2.3 Atomics in URB

Simple atomics are possible to be processed for URB locations as well. The process should fall out from the L3 path of the atomics and is restricted similar to L3 atomics.

3. L3 Allocation and Programming

L3 Cache allocation is done on a per way basis which should be consistent across all 4 banks (2 banks for GT1). The way allocation between URB and any of the L3 clients can only be changed post pipeline flush where L3 contains no data. This is required for stream based flushes to be dependent on the way allocation of these corresponding streams. S/W should not be removing ways under a particular stream and expect a later pipelined stream flush to target all the corresponding locations. The stream based flush will be performed on the existing way allocation of that stream, there is no history of previous way allocation tracked in the hardware.

L3 Cache has been divided into following client pools:

- **Shared Local Memory**: When enabled its size is always fixed to 128KB (64KB for GT1)
- **URB**: Local memory space, provides a flexible allocation on per 8KB granularity
- **DC**: Data Cluster Data type
- **Inst/State**: Both instructions and state allocation is combined
- **Constants**: Pull constants for EUs
- **Textures**: texture allocation to back-up L2$

In addition to these sub-groups, a collection of groups are generated to bundle multiple clients under the same allocation set:

- **All L3 Clients**: DC, Inst/State, Constants & Textures
- **Read-Only Clients**: Inst/State, Constants & Textures

Each of the L3 way allocations are managed via pLRU, hence best performance can be attained via assigning a **power-of-2 number** of ways. This is to ensure pLRU to distribute the ways w/o hot spotting within that client's group. Even though design provides a flexible (per way basis) programming model for way allocation for each client following table is given for validation and s/w programming models. The programming options in the following table represents most likely cases for different operation modes.

For GT1, hardware will retain 2 of the L3 banks hence all following allocations will be reduced half the size.

3.1 Non-SLM mode Allocation

Normal L3/URB mode (non-SLM mode), uses all 4 banks of L3 equally to distribute cycles. The following allocation is a suggested programming model. Note all numbers below are given in KBytes.

Normal Banked - No SLM									
Normal Bank									
	SLM	URB	Rest	DC	RO	I/S	C	T	Sum
0	0	256	0	0	256	0	0	0	512
1	0	256	0	128	128	0	0	0	512
2	0	256	0	32	0	64	32	128	512
3	0	224	0	64	0	64	32	128	512
4	0	224	0	128	0	64	32	64	512
5	0	224	0	64	0	128	32	64	512
6	0	224	0	0	0	128	32	128	512
7	0	256	0	0	0	128	0	128	512

3.2 SLM mode Allocation

With the existence of Shared Local Memory, a 64KB chunk from each of the 2 L3 banks will be reserved for SLM usage. The remaining cache space is divided between the remaining clients. SLM allocation is done via reducing the number of ways on the two banks from 64 to 32.

Shared Local Memory Mode - highly banked - aka MMemory									
Shared Loca									
	SLM	URB	Rest	DC	RO	I/S	C	T	Sum
0	128	128	0	128	128	0	0	0	512
1	128	128	0	64		64	64	64	512
2	128	128	0	32		64	32	128	512
3	128	128	0	32		128	32	64	512

Given the reduction on the 2 banks for L3, we have a unique problem of how to manage HASH between 4 un-equal size banks. The way to address that issue is to identify “Low Bandwidth” clients and allocate them into un-even ways of the large banks and handle them via a 2-way HASH. The remaining clients are allocated between 4 banks with the equal number of ways on each bank. The shaded clients in corresponding table are supposed to correspond to low bandwidth clients and their total should be 128KB.

Note that, having mixture of 2 hashes need to ensure “there are no coherency requirements between the high b/w and low b/w clients”. This solution prevents any producer/consumer models cross groups within L3. When programming the low b/w vs high b/w client profile, the need for coherency with the L3 fabric has to be considered.

Note that URB needs to be set as low b/w client in SLM mode, else the hash will fail. This is a required s/w model.

4. L3 Invalidation and Flush Flows

In, L3 fabric will support two different invalidation/flush flows. One for Read only streams and it could be per stream basis and the other one is the pipelined flush for modified lines in L3.

4.1 Read Only Stream Invalidations

All read-only stream invalidations are done at the TOP of the pipe and communicated to L3 fabric directly from the main command streamer. Once send to L3, command streamer can kick off the next workload to re-use the same memory space. There are four type streams that can be covered individually or overlapped:

- Texture Invalidation
- Instruction Invalidation
- Constant Invalidation
- State Invalidation

For all types of invalidations the flows are the same from L3 arbiter perspective: first invalidation will directly come from the command streamer and the “state invalidation” is sent via state arbiter unit.

L3 arbiter will propagate the invalidation only after the corresponding streams’ requests are retired to superQs. Once a particular invalidation is received, L3 arbiter will put a marker where all the existing requests from that stream. This will allow the already existing requests to be sent to superQ while accepting new requests, however new requests should never be sent to SuperQ until the existing invalidation is complete. Once all existing (marked requests) moved to SQ, L3 arbiter will propagate the invalidation request to SuperQ and keep the corresponding ingress FIFOs blocked.

Note: There may be two sources of the corresponding stream (i.e. Textures from either half-slice arbiters…). L3 arbiter needs to ensure both streams are serviced before propagating the “invalidation to SuperQ”.

Once SuperQ receives the invalidation, it will start monitoring particular streams transactions that could be still in SuperQ and wait for them to retire. If there is no such requests, this process would be immediate. The invalidation will be forwarded to L3 as SuperQ ensures there is no more requests in its slots with the matching clientID assignment.

Once L3 cache receives the invalidation, it is guaranteed that none of its requests in the TAG pipeline belongs to OLD marked requests from the stream getting the invalidation. SuperQ already guarantees this process by waiting for the retirement of the marked requests. TAG pipeline controller will stall the operations and wait for the TAG pipeline to clear than send the indication to STATE Array controller. For the ways that are corresponding to invalidation type, all ways for matching stream type will be updated in one shot. Once invalidation is complete, each L3 bank will send the indication to Pixel Arbiter (GAP). Note that multiple invalidations will be serialized in L3.

GAP will collect all invalidation requests from 4 banks (2 banks for GT1) and make sure all are complete before sending this completion back to corresponding L3 arbiters. L3 arbiters will ungate the ingress FIFOs of the completed invalidation as they complete the last step required.

This is exception part of the Read-Only Stream invalidations where state arbiter needs to stop sending new requests to L3 arbiters and ensure all pre-committed requests already sent to L3 arbiters:

- Stop and discard any prefetch requests that may be ongoing

- Finish any state requests with the length fields.

Once above conditions are satisfied the “state invalidation” will be forwarded to L3 arbiters and state arbiter will stop sending processing any other state requests while processing the invalidation. The ingress FIFOs will be kept blocked towards the arbiter until L3 arbiters send an indicator (all 4 of them) to state arbiter for the completion of the state invalidation. Note that this is an extra step for L3 arbiters only for the case of state invalidation within Read-Only streams.

Note that state arbiter needs to accumulate any new state requests that falls behind the invalidation event and not process them until the state invalidation complete indicator is seen from GAP. It needs to ensure old state data is cleared before sending the new state requests from clients.

There is a possibility where multiple invalidations could exist for a given stream. This is where L3 arbiters need to co-ordinate the when the invalidation requests would be processed. As L3 arbiter receives a particular invalidation to the time response is received from GAP, there could be yet another invalidation for the same stream. However since the ingress FIFOs are blocked, new invalidation request should not be processed before the prior request is complete. When L3 arbiter receives the completion from GAP, it will start processing the new invalidation as if it was received right at the same cycle.

Same rules apply for state arbiter as well where while the completion for a prior invalidation is pending, there may be another invalidation request from the main command streamer. State arbiter will hold off the execution of the newer invalidation until the completions are seen from the L3 arbiters.

Note that there could be third, forth invalidations while the very invalidation is being processed. All invalidations for a given stream could be collapsed while the prior one is being processed.

4.2 Pipelined Flush for Writes

Pipelined flush for data cluster writes will be propagated from DC directly to L3 arbiters as their buffers are flushed. L3 arbiters will be getting a flush indicator from each DC independently, there is no point of acting on the first flush indicator, it is easier to wait for both DCs to send their flush request and process them together. Once both flush indicators are seen, L3 arbiter will flush all its ingress FIFOs and block outlet of them being processed. L3 arbiter should not be sending any other requests to SuperQs until a completion is seen.

SuperQ as a response to Write Flush will wait for all its slots to retire, this is to prevent any boundary cases and ensure all writes are retired to L3 (if any). Once emptied, SQ will start accumulating “Flush” requests to the defined sets and ways (defined as data cluster reserved ways) and walk through each entry of the L3. These flush requests are to invalidate and evict any modified lines that may be present in L3s.

After all defined ways are walked with FLUSH requests, SuperQ should wait for empty indicator once again. This is to make sure all evicted data is retired towards GAP.

As GAP receives the flush complete indicator from each bank of L3 (4 for GT2 and 2 for GT1), it will ensure the eviction path is retired towards GAM, Once all done, it will send an indicator to PSD units in each half-slice. Same signal will be received by all L3 arbiters and used to un-block their interfaces towards SuperQ.

Note that SuperQ should not release any credits to L3 arbiter when retiring an internally generated flush request.

Similar to RO invalidations, the status of the write flush can also be tracked via register space. The indication of each bank will be reported via its corresponding L3 arbiter to a register space.

4.3 Global Invalidation

Once written a global indicator will be sent to L3 arbiters and state arbiter which will kick-off all invalidations at once. The same register space will collect all completion from L3 arbiters and state arbiter to clear the same bit.

Register bit can be polled by s/w to track to completion of all invalidations.

Bit	Access	Default Value	Description
0	RW/C	0	**L3 Global Invalidations:** Once written it will kick off a global invalidation for L3 both on RO and WR streams. S/W is expected to write logic1 to kick-off the invalidation and H/W will clear the bit once all invalidations are complete. Meanwhile S/W can poll the bit to track the completion of invalidation.

5. Shared Local Memory (SLM)

Shared local memory (aka highly-banked memory) is a portion of L3 which will be dedicated to EUs as a local memory when enabled. The accesses are only possible through data cluster with the destination flag set as SLM. In order to support a highly banked design, 2 of the L3 banks are structured to have 16x4KB portion which could be accessed independently per clock. This part of the L3 can support 16 dw size accesses (per SLM) in a given clock cycle.

These 16 banks can either be used as L3/URB or used as shared local memory with parallel accesses to all banks. The choice of enabling SLM mode is done through MMIO programming

Bit	Access	Default Value	Description
0	RW/C	0	**Enable Shared Local Memory:** When set, it enables the use of 2 banks of L3 as shared local memory which allows 64KB of L3 to be banked as 16x4KB and allows independent accesses to all banks within the same clock cycle. ***Note:*** *This mode can only be enabled once L3 content is completely flushed.*

In, SLM is structured as 64KB per half slice (6 EUs) and each 64KB allow up to 16 accesses to a particular 4KB bank. Each 4KB bank is addressed separately where their requests are placed on a 160bit address bus where each 10-bit correspond to a bank sequentially. Each bank gets addressed with 10bits and provides 4B access with a total of 4KB per bank. When SLM mode is not enabled, SLM banks are considered a part of L3 and used for cache or URB.

SLM requests are forked around the L3 arbiter, post ingress FIFOs for DC. L3 arbiter delivers request/data to SLM controller upon the availability of credits. Request will be crossed to 2x clock domain routed to corresponding banks. Individual bank controls are managed via SLM controller which are muxed with L3/URB accesses. Note that SLM accesses do carry byte enables and needs to be honored towards the banks. If the request has atomic requirements, SLM controller will provide the data to ALU along with the atomic type. Output data is again managed with SLM controller towards the output cross bars.

5.1 Protocol

The interface to access the Shared Local Memory is very similar to DC accesses to L3/URB with some additional fields:

- **Bank Valids**: A 16bit vector that points to which request in each bank's lane being active. DC is required to set the corresponding valid bit to perform a read or write operation for a corresponding bank.
- **Bank DW offsets**: 10-bit address is provided per bank basis. Each bank can be accessed with a different address within the same clock cycle
- **Byte Enables** could be set for any format or combination

The protocol also restricts some of the capabilities

- In a given clock cycle same type of command have to be used for all banks. This allows to read or write but not to combine different cycle operation within the same clock.

- Only 1 valid could be set if the SLM access requires atomics.
- In case of atomics, the data is provided always on the lower DW lanes (src0=DW0 and src1=DW1). The read return from SLM will have the DW of interest in the natural aligned location within 64B

SLM requests are routed through the same L3/URB arbiter using the protocol credits that are available between DC and L3, however right after the ingress queues SLM packets are routed to SLM controller w/o going through the arbitration.

6. Control Field Definition

For the state control fields are re-defined to comprehend L3 addition as following:

Bit[2]	GFDT	GFDT flag to color the displayable surfaces in LLC. This field is later used by GT to poll the LLC during query/flush mechanism and used to push the data to DRAM prior to Display Accesses
Bit[1:0]	Cacheability Controls	Control for LLC and L3 cacheabiltiy fields Bit[1]: LLC cacheability Bit[0]: L3 cacehability 0: Access is NOT cacheable 1: Access is cacheable

7. Dynamic Parity Feature for GFX L3 Cache

This document is meant to outline and describe dynamic parity detection function in the graphics L3 cache.

7.1 Feature Definition

The concept of DPF is to provide a run-time protection for graphics L3 cache via parity detection and redundant rows. Parity errors are considered to be an extremely rare occurrence, but this mechanism provides a means to address them should they occur.

In order to mitigate error detection needs of graphics L3 cache, parity detection capability has been added to each sub-bank of the L3. A sub-bank is defined as a 16KB entity of the total 512KB cache. This would translate into 32 independent sub-banks which all have 2 independent redundant rows for a total of 64 possible replacement rows. Redundant rows can be activated to replace a row with identified parity errors via writing the address of the parity error row into hardware registers.

DPF allows the HW to notify SW (driver) when any single bit graphics L3 parity error has occurred and also provide a mechanism to allow SW to fix persistent bit errors where a susceptible bit fails multiple times.

In order to support robust GPGPU / compute workloads on graphics, Intel recommends that driver developers implement this suggested DPF flow. Regardless, all usages of the graphics L3 cache can benefit from the added parity protection.

7.2 Hardware and Software Flows

7.2.1 Parity Generation & Detection

The graphics L3 cache will generate 1-bit parity as data is stored in the cache. The parity bit will be written along with data for future verification. As the same content is accessed later in time, HW re-calculates the parity based on read content and compares with the stored parity value.

Once a mismatch is detected, HW generates a parity interrupt for the graphics driver to service. Meanwhile, HW continues forward progress in execution. There is no implicit halt or execution stopping for the IVB / Gen7 HW.

Along with the interrupt, the HW will update a set of registers to indicate which bank/sub-bank/row in which the error has been detected.

7.2.2 Correction Using Parity Error data and Redundant Rows

Each sub-bank contains 2 redundant rows which could be used to replace the rows with reported parity errors. The graphics driver, once servicing the parity interrupt, will access the reporting registers and record the bank/sub-bank/row information. This information should be stored in a permanent (non-volatile memory: ie: disk, registry or similar) location for future use by the graphics driver.

The graphics driver will then reset the render engine (i.e. render specific reset) to prevent propagation of the data with the parity error. Even though the graphics driver will terminate the context via resetting the GPU upon a parity interrupt, there is a possibility that the parity interrupt may be observed by the driver after the graphics context is complete or about to switch to a new context. For non-graphics workloads that require high data integrity, such as GPGPU computing, the driver should prevent this possible boundary case by polling the error reporting registers when a context completion or context switch interrupt is registered by the driver. Upon identifying that the error reporting registers are active, the driver should follow the same steps as servicing a parity interrupt.

Post any reset event (graphics reset, power up, etc..), the graphics driver retrieve the parity failure row information from non-volatile storage. It will then program the parity failure row information into corresponding bank/sub-bank registers and start normal graphics operation. Parity failure rows would then be effectively replaced with the extra redundant rows until next system reset.

In the extremely rare probability that redundant rows themselves have a parity errors, the parity error will be reported as the row they have replaced. SW drivers should recognize the use of redundant row and skip the replacement.

7.2.3 Number of Corrections

Given DPF is designed to deal with persistent errors, graphics drivers need to be able to identify which sub-bank rows are producing the most number of errors. Hence, the driver should keep a list of the reported parity error rows and record the number of times each row reports a parity error. If there are more than 2 parity error rows identified for a given sub-bank, the driver should replace the top two rows first (decided by historical error count). This will force rows with consistent parity errors to bubble up to the top of the list to be replaced.

SW driver tracking of parity error rows should be saved in non-volatile memory, so the driver can keep track of parity failure rows across reboot/reset.

7.2.4 Summary

Basic Algorithm:

- Each sub-bank has two redundant rows. (By HW design)
- Driver SW keeps track of each sub-bank's parity failure rows and keeps a count of failures of each row.
- Error count row information is saved to non-volatile memory, so it persists across reboot and graphics resets.
- At all times, the two redundant sub-bank rows are used to replace the highest count parity failure rows.
- SW always forces reset (graphics render reset is sufficient) on L3 errors.
- SW programs all necessary replacement rows after any reset.

7.2.5 Sub-banks with more than two persistent parity error rows

While not expected during normal lifetime operation, a problematic case could occur when the HW reports more than 2 rows on a particular sub-bank are causing parity errors. For this case, the graphics driver should keep replacing the rows, always selecting the two rows with the highest parity error failure count.

7.2.6 Interrupt Enabling

In order to enable proper DPF notification, the graphics driver must enable the correct interrupt paths from render command streamer (x20A8) as well as rest of the interrupt structures around GTISR (x44010), GTIMR (x44014), GTIIR (x44018), GTIER (x4401C). Bit5 has been selected for L3 parity interrupts. Please refer to the appropriate register sections for generic graphics interrupt enabling details.

7.2.7 Clearing the Error Reporting Registers

Clearing error status registers for the L3 cache should be done by writing to L3CD Error Status Register bit[13] with a logical value of “1”. This should be done after an error is reported and a row has been replaced. The graphics driver can do this in two different ways:

1. Via batch buffer executed in HW (via LRI or LRM mechanisms)
2. Direct writes to graphics HW MMIO space

Direct reads and writes using graphics MMIO (for both error log and status registers) requires DOP level clock gating to be turned off as HW might have finished execution, which will result in a hang when accessing the L3 parity registers. This requires graphics MISCCPCTL bit 0 (x9424[0]) to be cleared prior to register updates. It must be set again post register updates. Do not leave the DOP clock gating bit cleared. Doing so will significantly affect graphics power.

Note that the HW parity registers will clear with any reset, hence error and row replacement registers will have to be re-programmed any time the driver performs a HW reset or driver is re-loaded.

7.3 Hardware Registers

Various registers are used to report and contain the parity error failing row information:

7.3.1 Error Report Registers

This is the lead section for this topic and will be updated over time with introductory comments and links to related information.

Use the “Next Topic” (right arrow) button above to proceed to the first detail section.

7.3.1.1 L3CDERRST1 - L3CD Error Status register 1

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B008-B00Bh
Default Value: 00000000h
Access: RW; RO; WO;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:25	RO	0000000b	Core	**Reserved (RSVD):** Reserved
24:14	RWC	00000000000b	Core	**Parity row address error (PRTYROWNUM):** Data array address which has parity B1: Report the data array address which has the Error ltcd_sarb_parity_err_rownum[10:0] Once set by HW, it can be cleared only by MMIO Write of 1 to this register bit 13 .
13	RWC	0b	Core	**Parity Error Valid (PRTYERRVLD):** Parity Error valid Report the Parity Error ltcd_sarb_parity_err_valid Once set by HW, it can be cleared only by MMIO Write of 1 to this register bit 13 .
12:11	RWC	00b	Core	**Parity error bank number (PRTYBNKNUM):** bank number which has parity error Report the bank no. which has the Error ltcd_sarb_parity_err_banknum[1:0] Once set by HW, it can be cleared only by MMIO Write of 1 to this register bit 13 .
10:8	RWC	000b	Core	**Parity Error sub-bank no (PRTYSBNKNUM):** Parity Error in sub bank: ltcd0_sarb_parity_err_subanknum[2:0] Once set by HW, it can be cleared only by MMIO Write of 1 to this register bit 13 .
7	RW	1b	Core	**Parity report enable (LCPRTYRPTEN):** sarbcf_csr_lc_parity_report_en

				this is the parity reporting enable, by default it is enabled. When enabled parity will be reported by ltcd to sarb. When disabled by driver, ltcd should not send out any parity error to SARB.
6:0	RO	00h	Core	**Reserved (RSVD):**

7.3.2 Row Replacement Registers

The range of row replacement registers is addresses xB070 (bank0/sub-bank0) to xB0EC (bank3/sub-bank7). Only the first register format is given below - all registers have the same format.

7.3.2.1 L3B0REG0 - L3 bank0 reg0 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B070-B073h
Default Value: 00000000h
Access: RW; RO
Size: 32 bits

The log error registers of the L3 will maintain the bad row information for each of the 16KB sub-bank groups. The log should be programmed by the driver before any workloads are submitted if it is necessary to replace parity failing rows. The contents of the log register are included in any context save and restore by HW for RC6 power events.

All L3 error registers are put as part of HW context which could lead the same error to be reported multiple times (i.e. when context gets reloaded).

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error.
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error.
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8. L3 Register Space (Bspec)

8.1 config space for L3

Register Name	Register Symbol	Register Start	Register End	Default Value	Access
SARB Error Status	SARERRST	B004	B007	00000000h	RO;
L3CD Error Status register 1	L3CDERRST1	B008	B00B	00000080h	RW; RO;
L3CD Error Status register 2	L3CDERRST2	B00C	B00F	00000000h	RO;
L3 SQC registers 1	L3SQCREG1	B010	B013	01730000h	RW; RO;
L3 SQC registers 2	L3SQCREG2	B014	B017	00004567h	RO; RW;
L3 SQC registers 3	L3SQCREG3	B018	B01B	00001ABFh	RO; RW;
L3 Control Register1	L3CNTLREG1	B01C	B01F	8C47FF80h	RW; RO;
L3 Control Register2	L3CNTLREG2	B020	B023	00080040h	RW; RO;
L3 Control Register3	L3CNTLREG3	B024	B027	00000000h	RO; RW;
L3 SLM Register	L3SLMREG	B028	B02B	40000000h	RO; RW;
Arbiter Control Register	GARBCNTLREG	B02C	B02F	29000000h	RO; RW;
L3 SQC register 4	L3SQCREG4	B034	B037	88000000h	RW; RO;
L3 bank0 reg0 log error	L3B0REG0	B070	B073	00000000h	RW; RO;
L3 bank0 reg1 log error	L3B0REG1	B074	B077	00000000h	RW; RO;
L3 bank0 reg2 log error	L3B0REG2	B078	B07B	00000000h	RW; RO;
L3 bank0 reg3 log error	L3B0REG3	B07C	B07F	00000000h	RW; RO;
L3 bank0 reg4 log error	L3B0REG4	B080	B083	00000000h	RW; RO;
L3 bank0 reg5 log error	L3B0REG5	B084	B087	00000000h	RW; RO;
L3 bank0 reg6 log error	L3B0REG6	B088	B08B	00000000h	RW; RO;
L3 bank0 reg7 log error	L3B0REG7	B08C	B08F	00000000h	RW; RO;
L3 bank1 reg0 log error	L3B1REG0	B090	B093	00000000h	RW; RO;
L3 bank1 reg1 log error	L3B1REG1	B094	B097	00000000h	RW; RO;
L3 bank1 reg2 log error	L3B1REG2	B098	B09B	00000000h	RW; RO;
L3 bank1 reg3 log error	L3B1REG3	B09C	B09F	00000000h	RW; RO;
L3 bank1 reg4 log error	L3B1REG4	B0A0	B0A3	00000000h	RW; RO;
L3 bank1 reg5 log error	L3B1REG5	B0A4	B0A7	00000000h	RW; RO;
L3 bank1 reg6 log error	L3B1REG6	B0A8	B0AB	00000000h	RW; RO;
L3 bank1 reg7 log error	L3B1REG7	B0AC	B0AF	00000000h	RW; RO;
L3 bank2 reg0 log error	L3B2REG0	B0B0	B0B3	00000000h	RW; RO;
L3 bank2 reg1 log error	L3B2REG1	B0B4	B0B7	00000000h	RW; RO;
L3 bank2 reg2 log error	L3B2REG2	B0B8	B0BB	00000000h	RW; RO;
L3 bank2 reg3 log error	L3B2REG3	B0BC	B0BF	00000000h	RW; RO;
L3 bank2 reg4 log error	L3B2REG4	B0C0	B0C3	00000000h	RW; RO;
L3 bank2 reg5 log error	L3B2REG5	B0C4	B0C7	00000000h	RW; RO;
L3 bank2 reg6 log error	L3B2REG6	B0C8	B0CB	00000000h	RW; RO;
L3 bank2 reg7 log error	L3B2REG7	B0CC	B0CF	00000000h	RW; RO;
L3 bank3 reg0 log error	L3B3REG0	B0D0	B0D3	00000000h	RW; RO;
L3 bank3 reg1 log error	L3B3REG1	B0D4	B0D7	00000000h	RW; RO;
L3 bank3 reg2 log error	L3B3REG2	B0D8	B0DB	00000000h	RW; RO;
L3 bank3 reg3 log error	L3B3REG3	B0DC	B0DF	00000000h	RW; RO;
L3 bank3 reg4 log error	L3B3REG4	B0E0	B0E3	00000000h	RW; RO;
L3 bank3 reg5 log error	L3B3REG5	B0E4	B0E7	00000000h	RW; RO;
L3 bank3 reg6 log error	L3B3REG6	B0E8	B0EB	00000000h	RW; RO;
L3 bank3 reg7 log error	L3B3REG7	B0EC	B0EF	00000000h	RW; RO;

Register Name	Register Symbol	Register Start	Register End	Default Value	Access
SARB config save msg	SARBCSR	B1FC	B1FF	00000000h	RWHC; RO;

8.1.1 SARERRST - SARB Error Status

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B004-B007h
Default Value: 00000000h
Access: RO
Size: 32 bits

Reports the error if any has occurred for certain sarb features.

Bit	Access	Default Value	RST/PWR	Description
31	RO	0b	Core	**Error if general bound is zero (ERRGENBDZO):** Error if general bound is zero set by sarbunit 1: general bound address is 0 sarbcf_csr_gen_bnd_zero_err
30	RO	0b	Core	**Error if dynamic bound is zero (ERRDYDNZO):** Error if dynamic bound is zero- set by sarbunit 0:no error 1: dynamic address is 0 sarbcf_csr_dyn_bnd_zero_err
29	RO	0b	Core	**Reserved (RSVD):**
28	RO	0b	Core	**General Bound Check Overflow Error (GENBNDOVF):** General Bound Check Overflow Error - set by sarbunit 1: overflow for general bound check sarbcf_csr_gen_bnd_ovflw_err
27	RO	0b	Core	**Dynamic Bound Check Overflow Error (DYNBDOVF):** Dynamic Bound Check Overflow Error -set by sarbunit 1: overflow for dynamic bound check sarbcf_csr_dyn_bnd_ovflw_err
26	RO	0b	Core	**Lower Bound Check Overflow Error (LWRBDOVF):** Lower Bound Check Overflow Error-set by sarbunit lower bound overflow sarbcf_csr_lower_bnd_err
25:21	RO	00000b	Core	**INVALIDATION FLUSH STATUS REPORTING (INVSTRPT):**

				invalidation status for l3 is reported in this register.
20:18	RO	000b	Core	**SARB invalidation Status reporting (SARBINVSTRPT):** invalidation status of sarb is reported in this register.
17:0	RO	00000h	Core	**Reserved (RSVD):** Reserved

8.1.2 L3CDERRST2 - L3CD Error Status register 2

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B00C-B00Fh
Default Value: 00000000h
Access: RO; RWC;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:29	RO	000b	Core	**Reserved (RSVD):** reserved
28	RWC	0b	Core	**URB High Limit Error on B3 (URBHLB3):** URB High Limit Error on B3: Report the URB High Limit Error- Address Bound check Once set, it can be cleared only by MMIO Write to this register. A write of value 1 will clear it (LTCC generates a Pulse to SARB Config , Sarb Config sets and reflect it in the MMIO as Error status. This can be only cleared by MMIO Write to that Bit.) ltcc3_sarb_urboff_error
27	RWC	0b	Core	**URB High Limit Error on B2 (URBHLB2):** URB High Limit Error on B2: Report the URB High Limit Error- Address Bound check Once set, it can be cleared only by MMIO Write to this register. (LTCC generates a Pulse to SARB Config , Sarb Config sets and reflect it in the MMIO as Error status. This can be only cleared by MMIO Write to that Bit.) ltcc2_sarb_urboff_error
26	RWC	0b	Core	**URB High Limit Error on B1 (URBHLB1):** URB High Limit Error on B1: Report the URB High Limit Error - Address Bound check Once set, it can be cleared only by MMIO Write to this register. (LTCC generates a Pulse to SARB Config , Sarb Config sets and reflect it in the MMIO as Error status. This can be only cleared by MMIO Write to that Bit.)

Bit	Access	Default Value	RST/PWR	Description
				ltcc1_sarb_urboff_error
25	RWC	0b	Core	**URB High Limit Error on B0 (URBHLB0):** URB High Limit Error on B0 : Report the URB High Limit Error - Address Bound check Once set, it can be cleared only by MMIO Write to this register. (LTCC generates a Pulse to SARB Config , Sarb Config sets and reflect it in the MMIO as Error status. This can be only cleared by MMIO Write to that Bit.) ltcc0_sarb_urboff_error
24:0	RO	0000000h	Core	**Reserved (RSVD):**

8.1.3 L3SQCREG1 - L3 SQC registers 1

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B010-B013h
Default Value: 01730000h
Access: RW; RO;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:28	RO	0000b	Core	**Reserved (RSVD):** Reserved
27	RW	0b	Core	**Convert L3 cycle for texture to uncachable (CON4TXTUNC):** Convert L3 cycle for texture to uncachable 1: texture has no way allocation in L3 0: texture has atleast 1 way allocated in L3 (default) sarbcf_csr_lsqc_cnvt_txt_unchble This bit should be set/cleared according to L3 configuration in registers at offset 0xB020 and 0xB024
26	RW	0b	Core	**Convert L3 cycle for constant to uncachable (CON4CONSUNC):** Convert L3 cycle for constant to uncachable 1: constant has no way allocation in L3 0: constant has atleast 1 way allocated in L3 (default) sarbcf_csr_lsqc_cnvt_const_unchble This bit should be set/cleared according to L3 configuration in registers at offset 0xB020 and 0xB024
25	RW	0b	Core	**Convert L3 cycle for Inst/State to uncachable (CON4INSSTUNC):**

<table>
<tr><th>Bit</th><th>Access</th><th>Default Value</th><th>RST/PWR</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>Convert L3 cycle for Inst/State to uncachable
1: Inst/State has no way allocation in L3
0: Inst/State has atleast 1 way allocated in L3 (default)
sarbcf_csr_lsqc_cnvt_ins_st_unchble
This bit should be set/cleared according to L3 configuration in registers at offset 0xB020 and 0xB024</td></tr>
<tr><td>24</td><td>RW</td><td>1b</td><td>Core</td><td>Convert L3 cycle for DC to uncachable (CON4DCUNC):
Convert L3 cycle for DC to uncachable
1: DC has no way allocation in L3 (default)
0: DC has atleast 1 way allocated in L3 sarbcf_csr_lsqc_cnvt_dc_unchble
This bit should be set/cleared according to L3 configuration in registers at offset 0xB020 and 0xB024
If 0xB020[26:21] (DC Way Assignment) field is 000000, this bits needs to be set (1). For all other values of 0xB020[26:21], this bit needs to be cleared (0)</td></tr>
<tr><td>23:16</td><td>RW</td><td>01110011b</td><td>Core</td><td>L3SQ General / High Priority Credit Initialization (SQGHPCI):
L3SQ General / High Priority Credit Initialization (SQGHPCI)Number of general and high priority credits that SQ presents to L3 Arbiter blocks. This inherently also determines the depth of the SQ; reduce the number of credits and SQ will use fewer slots.Any value not listed here, is considered Reserved.
<table>
<tr><th>Value</th><th># General Credits</th><th># High Pri Credits</th><th>SQ de</th></tr>
<tr><td>00010000</td><td>2</td><td>0</td><td>2</td></tr>
<tr><td>00010001</td><td>2</td><td>2</td><td>4</td></tr>
<tr><td>00100000</td><td>4</td><td>0</td><td>4</td></tr>
<tr><td>00100001</td><td>4</td><td>2</td><td>6</td></tr>
<tr><td>00100010</td><td>4</td><td>4</td><td>8</td></tr>
<tr><td>00110000</td><td>6</td><td>0</td><td>6</td></tr>
<tr><td>00110001</td><td>6</td><td>2</td><td>8</td></tr>
<tr><td>00110010</td><td>6</td><td>4</td><td>10</td></tr>
<tr><td>00110011</td><td>6</td><td>6</td><td>12</td></tr>
<tr><td>01000000</td><td>8</td><td>0</td><td>8</td></tr>
<tr><td>01000001</td><td>8</td><td>2</td><td>10</td></tr>
<tr><td>01000010</td><td>8</td><td>4</td><td>12</td></tr>
<tr><td>01000011</td><td>8</td><td>6</td><td>14</td></tr>
<tr><td>01000100</td><td>8</td><td>8</td><td>16</td></tr>
<tr><td>01010000</td><td>12</td><td>0</td><td>12</td></tr>
<tr><td>01010001</td><td>12</td><td>2</td><td>14</td></tr>
<tr><td>01010010</td><td>12</td><td>4</td><td>16</td></tr>
<tr><td>01010011</td><td>12</td><td>6</td><td>18</td></tr>
<tr><td>01010100</td><td>12</td><td>8</td><td>20</td></tr>
<tr><td>01010101</td><td>12</td><td>12</td><td>24</td></tr>
<tr><td>01100000</td><td>16</td><td>0</td><td>16</td></tr>
<tr><td>01100001</td><td>16</td><td>2</td><td>18</td></tr>
<tr><td>01100010</td><td>16</td><td>4</td><td>20</td></tr>
<tr><td>01100011</td><td>16</td><td>6</td><td>22</td></tr>
<tr><td>01100100</td><td>16</td><td>8</td><td>24</td></tr>
<tr><td>01110000</td><td>18</td><td>0</td><td>18</td></tr>
</table></td></tr>
</table>

Bit	Access	Default Value	RST/PWR	Description
				01110001 18 2 20 01110010 18 4 22 01110011 18 6 24(default) 10000000 20 0 20 10000001 20 2 22 10000010 20 4 24 10010000 22 0 22 10010001 22 2 24 10100000 24 0 24 sarbcf_csr_lsqc_hp_gen_credit_init[7:0]
15:14	RO	00b	Core	**Reserved (RSVD):** Reserved
13:12	RW	00b	Core	**L3SQ Atomics Credit Initialization (SQACI):** L3SQ Atomics Credit Initialization (SQACI)Number of atomics credits that SQ presents to L3 Arbiter blocks. 00 = 2 Credits (default) 01 = 1 Credit1X = Reserved sarbcf_csr_lsqc_atom_credit_init[1:0]
11:10	RW	00b	Core	**L3SQ Data Credit Initialization (SQDCI):** L3SQ Data Credit Initialization (SQDCI)Number of data credits that SQ presents to L3 Arbiter blocks. 00 = 4 Credits (default) 01 = 1 Credit 10 = 2 Credits 11 = 3 Credits sarbcf_csr_lsqc_data_credit_init[1:0]
9	RW	0b	Core	**L3SQ Read Once Enable for Sampler Client (SQROE):** L3SQ Read Once Enable for Sampler Client (SQROE): Enables Read Once indications to L3 Cache from SQ. Once enabled, any reads from Sampler client (MT) will be sent as Read Once 0 = Reads from Sampler clients issue Read to L3 Cache (default) 1 = Reads from Sampler clients issue Read Once to L3 Cache sarbcf_csr_sampler_readonce_en
8:6	RW	000b	Core	**L3SQ Outstanding GAP Reads (SQOUTSGAP):** L3SQ Outstanding GAP Reads (SQOUTSGAP): Identifies the number of Pixel Arbiter Reads that can be outstanding before SQ throttles the puts to GAP. This is not an exact limit, but instead it is used as a threshold to throttling; once the read count is greater than or equal to the threshold, then no reads will be issued until data returns are received to bring the outstanding count back below the

Bit	Access	Default Value	RST/PWR	Description
				threshold. 000 = No limit (default) 001 = 1 read 010 = 2 reads 011 = 4 reads100 = 8 reads 101 = 16 reads11X = Reserved sarbcf_csr_lsqc_outs_gaprd[2:0]
5:3	RW	000b	Core	**L3SQ Outstanding L3 Fills (SQOUTSL3F):** L3SQ Outstanding L3 Fills (SQOUTSL3F): Identifies the number of L3 Fills that can be outstanding before SQ throttles the fill requests to L3 Cache. This is not an exact limit, but instead it is used as a threshold to throttling; once the fill count is greater than or equal to the threshold, then no fills will be issued until the fill responses are received to bring the outstanding count back below the threshold. 000 = No limit (default) 001 = 1 fill 010 = 2 fills 011 = 4 fills 100 = 8 fills 101 = 16 fills 11X = Reserved sarbcf_csr_lsqc_outs_fill[2:0]
2:0	RW	000b	Core	**L3SQ Outstanding L3 Lookups (SQOUTSL3L):** L3SQ Outstanding L3 Lookups (SQOUTSL3L): Identifies the number of L3 lookups that can be outstanding before SQ throttles the lookup requests to L3 Cache. This is not an exact limit, but instead it is used as a threshold to throttling; once the lookup count is greater than or equal to the threshold, then no lookups will be issued until the lookup responses are received to bring the outstanding count back below the threshold. 000 = No limit (default) 001 = 1 lookup 010 = 2 lookups 011 = 4 lookups 100 = 8 lookups 101 = 16 lookups 11X = Reserved sarbcf_csr_lsqc_outs_lookup[2:0]

8.1.4 L3SQCREG2 - L3 SQC registers 2

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B014-B017h
Default Value: 00004567h
Access: RO; RW;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:17	RO	000000000000000b	Core	**Reserved (RSVD):** Reserved
16	RW	0b	Core	**L3SQ Priority Selection Disable (SQPRIDIS):** L3SQ Priority Selection Disable (SQPRIDIS)Enables the use of priority selection based on client ID decodes. If disabled, all cycles in SQ will be treated as same priority 0 = Priority selection is enabled (default) 1 = Priority selection is disabled sarbcf_csr_priority_cnt_disable
15	RW	0b	Core	**L3SQ Priority 3 Pool Count Disable (SQPRI3CNTDIS):** L3SQ Priority 3 Pool Count Disable (SQPRI3CNTDIS): When set, priority3 pool becomes unlimited. And priority3 pool count value should not be used in reset of the remaining counters. 0 = Priority 3 pool count is enabled (default) 1 = Priority 3 pool count is disabled sarbcf_csr_priority3_cnt_disable
14:12	RW	100b	Core	**L3SQ Priority 3 Pool Counter (SQPRI3CNT):** L3SQ Priority 3 Pool Counter (SQPRI3CNT): The count of cycles will be selected from priority3 pool before switching to other priority pools. Count is used as the power of 2. 000 = 1 request 001 = 2 requests 010 = 4 requests 011 = 8 requests .. 111 = 128 requests sarbcf_csr_priority3_cnt[2:0]
11	RW	0b	Core	**L3SQ Priority 2 Pool Count Disable (SQPRI2CNTDIS):** L3SQ Priority 2 Pool Count Disable (SQPRI2CNTDIS): When set, priority2 pool becomes unlimited. And priority2 pool count value should not be used in reset of the remaining counters.

Bit	Access	Default Value	RST/PWR	Description
				0 = Priority 2 pool count is enabled (default) 1 = Priority 2 pool count is disabled sarbcf_csr_priority2_cnt_disable
10:8	RW	101b	Core	**L3SQ Priority 2 Pool Counter (SQPRI2CNT):** L3SQ Priority 2 Pool Counter (SQPRI2CNT): The count of cycles will be selected from priority2 pool before switching to other priority pools. Count is used as the power of 2. 000 = 1 request 001 = 2 requests 010 = 4 requests 011 = 8 requests .. 111 = 128 requests sarbcf_csr_priority2_cnt[2:0]
7	RW	0b	Core	**L3SQ Priority 1 Pool Count Disable (SQPRI1CNTDIS):** L3SQ Priority 1 Pool Count Disable (SQPRI1CNTDIS): When set, priority1 pool becomes unlimited. And priority1 pool count value should not be used in reset of the remaining counters. 0 = Priority 1 pool count is enabled (default) 1 = Priority 1 pool count is disabled sarbcf_csr_priority1_cnt_disable
6:4	RW	110b	Core	**L3SQ Priority 1 Pool Counter (SQPRI1CNT):** L3SQ Priority 1 Pool Counter (SQPRI1CNT): The count of cycles will be selected from priority1 pool before switching to other priority pools. Count is used as the power of 2. 000 = 1 request 001 = 2 requests 010 = 4 requests 011 = 8 requests .. 111 = 128 requests sarbcf_csr_priority1_cnt[2:0]
3	RW	0b	Core	**L3SQ Priority 0 Pool Count Disable (SQPRI0CNTDIS):** L3SQ Priority 0 Pool Count Disable (SQPRI0CNTDIS): When set, priority0 pool becomes unlimited. And priority0 pool count value should not be used in reset of the remaining counters. 0 = Priority 0 pool count is enabled (default)

Bit	Access	Default Value	RST/PWR	Description
				1 = Priority 0 pool count is disabled sarbcf_csr_priority0_cnt_disable
2:0	RW	111b	Core	**L3SQ Priority 0 Pool Counter (SQPRI0CNT):** L3SQ Priority 0 Pool Counter (SQPRI0CNT): The count of cycles will be selected from priority0 pool before switching to other priority pools. Count is used as the power of 2. 000 = 1 request 001 = 2 requests 010 = 4 requests 011 = 8 requests .. 111 = 128 requests (default) sarbcf_csr_priority0_cnt[2:0]

8.1.5 L3SQCREG3 - L3 SQC registers 3

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B018-B01Bh
Default Value: 00001ABFh
Access: RO; RW;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:30	RO	00b	Core	**Reserved (RSVD):** Reserved
29:28	RW	00b	Core	**SOLunit Priority Value (SQSOLPRIVAL):** SOLunit Priority Value (SQSOLPRIVAL): Identifies the priority value for all cycles that are initiated by SOLunit. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 1 10 = Priority 2 11 = Priority 3 sarbcf_csr_sol_priority[1:0]
27:26	RW	00b	Core	**GSunit Priority Value (SQGSPRIVAL):** GSunit Priority Value (SQGSPRIVAL): Identifies the priority value for all cycles that are initiated by GSunit. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 1

Bit	Access	Default Value	RST/PWR	Description
				10 = Priority 2 11 = Priority 3 sarbcf_csr_gs_priority[1:0]
25:24	RW	00b	Core	**TEunit Priority Value (SQTEPRIVAL):** TEunit Priority Value (SQTEPRIVAL): Identifies the priority value for all cycles that are initiated by TEunit. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 1 10 = Priority 2 11 = Priority 3 sarbcf_csr_te_priority[1:0]
23:22	RW	00b	Core	**CLunit Priority Value (SQCLPRIVAL):** CLunit Priority Value (SQCLPRIVAL): Identifies the priority value for all cycles that are initiated by CLunit. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 110 = Priority 2 11 = Priority 3 sarbcf_csr_cl_priority[1:0]
21:20	RW	00b	Core	**TSunit Priority Value (SQTSPRIVAL):** TSunit Priority Value (SQTSPRIVAL): Identifies the priority value for all cycles that are initiated by TSunit. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 1 10 = Priority 2 11 = Priority 3 sarbcf_csr_ts_priority[1:0]
19:18	RW	00b	Core	**SFunit Priority Value (SQSFPRIVAL):** SFunit Priority Value (SQSFPRIVAL): Identifies the priority value for all cycles that are initiated by SFunit. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 1 10 = Priority 2 11 = Priority 3 sarbcf_csr_sf_priority[1:0]
17:16	RW	00b	Core	**SVSM Priority Value (SQSVSMPRIVAL):**

Bit	Access	Default Value	RST/PWR	Description
				SVSM Priority Value (SQSVSMPRIVAL): Identifies the priority value for all cycles that are initiated by SVSM. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 1 10 = Priority 2 11 = Priority 3 sarbcf_csr_svsm_priority[1:0]
15:14	RW	00b	Core	**SARB Priority Value (SQSARBPRIVAL):** SARB Priority Value (SQSARBPRIVAL): Identifies the priority value for all cycles that are initiated by State Arbiter (SARB). Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 (default) 01 = Priority 1 10 = Priority 2 11 = Priority 3 sarbcf_csr_sarb_priority[1:0]
13:12	RW	01b	Core	**SBE Priority Value (SQSBEPRIVAL):** SBE Priority Value (SQSBEPRIVAL): Identifies the priority value for all cycles that are initiated by SBE. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 01 = Priority 1 (default) 10 = Priority 2 11 = Priority 3 sarbcf_csr_sbe_priority[1:0]
11:10	RW	10b	Core	**IC$ Priority Value (SQICPRIVAL):** IC$ Priority Value (SQICPRIVAL): Identifies the priority value for all cycles that are initiated by Instruction Cache (IC$). Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 01 = Priority 1 10 = Priority 2 (default) 11 = Priority 3 sarbcf_csr_ic_priority[1:0]
9:8	RW	10b	Core	**TDL Priority Value (SQTDLPRIVAL):** TDL Priority Value (SQTDLPRIVAL): Identifies the priority value for all cycles that are initiated by TDL. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0

Bit	Access	Default Value	RST/PWR	Description
				01 = Priority 1 10 = Priority 2 (default) 11 = Priority 3 sarbcf_csr_tdl_priority[1:0]
7:6	RW	10b	Core	**DCunit Priority Value (SQDCPRIVAL):** DCunit Priority Value (SQDCPRIVAL): Identifies the priority value for all cycles that are initiated by DC. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 01 = Priority 1 10 = Priority 2 (default) 11 = Priority 3 sarbcf_csr_dc_priority[1:0]
5:4	RW	11b	Core	**DAPR Priority Value (SQDAPRPRIVAL):** DAPR Priority Value (SQDAPRPRIVAL): Identifies the priority value for all cycles that are initiated by DAPR. Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 01 = Priority 1 10 = Priority 2 11 = Priority 3 (default) sarbcf_csr_dapr_priority[1:0]
3:2	RW	11b	Core	**MTunit Priority Value (SQMTPRIVAL):** MTunit Priority Value (SQMTPRIVAL): Identifies the priority value for all cycles that are initiated by Sampler (MT). Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 01 = Priority 1 10 = Priority 2 11 = Priority 3 (default) sarbcf_csr_mt_priority[1:0]
1:0	RW	11b	Core	**LSQCunit Priority Value (SQPRIVAL):** LSQCunit Priority Value (SQPRIVAL): Identifies the priority value for all cycles that are initiated by Super Queue (L3 Evictions). Priority is used in the L3 Super Queue (L3SQ). 00 = Priority 0 01 = Priority 1 10 = Priority 2 11 = Priority 3 (default)

Bit	Access	Default Value	RST/PWR	Description
				sarbcf_csr_lsqc_priority[1:0]

8.1.6 L3CNTLREG1 - L3 Control Register1

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B01C-B01Fh
Default Value: 8C47FF80h
Access: RW; RO;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:28	RW	1000b	Core	**Data Fifo Depth Control (DFIFODC):** Data Fifo Depth Control (TS mode) Stall Control: POR is 1000b. Flexing for Hitting the stall Validation scenarios Maximum available setting for h/w is "1000" sarbcf_csr_lc_datafifo_depth[3:0] Due to w/a of bug#3665919, the depth has to be set to “0011”
27:24	RW	1100b	Core	**Data Clock off time (DCLKOFFT):** Data Clock off time (DATACLKOFF): Data Clock off time - Data block is shut off after these many number of clocks programmed in this register bits. sarbcf_csr_lc_dataclkoff_time[3:0]
23:20	RW	0100b	Core	**TAG CLK OFF TIME (TAGCLKOFF):** TAG CLK OFF TIME (TAGCLKOFF): TAG Clock Off time. This is the time, which Clock gating Logic check before it turn off the clock. sarbcf_csr_lc_tagclkoff_time[3:0]
19	RW	0b	Core	**L3 Aging Disable Bit (L3AGDIS):** L3 Aging Disable Bit (L3AGDIS): Aging Disable sarbcf_csr_lc_agingdis
18:15	RW	1111b	Core	**Fill aging (L3AGF):** Fill aging (L3AGF): Aging Counter for Fill sarbcf_csr_lc_fill_aging_cnt[3:0]
14:11	RW	1111b	Core	**Aging Counter for Read 1 Port (L3AGR1):** Aging Counter for Read 1 Port (L3AGR1): Aging Counter

Bit	Access	Default Value	RST/PWR	Description
				for Read 1 Port sarbcf_csr_lc_rd1_aging_cnt[3:0]
10:7	RW	1111b	Core	**L3 Aging Counter for R0 (L3AGR0):** L3 Aging Counter for R0 (L3AGR0): Aging Counter for R0 Port sarbcf_csr_lc_rd0_aging_cnt[3:0]
6:3	RW	0000b	Core	**Number of NOPs (L3NOP):** Number of NOPs (L3NOP): Number of NOPs to be inserted between the Tag commands. sarbcf_csr_lc_num_nop[3:0] Due to w/a of bug#3665919, the depth has to be set to "0001"
2	RW	0b	Core	**OP0/OP1 Disable (L3OPDIS):** OP0/OP1 Disable (L3OPDIS): This bit is used to enable the feature of inserting the number of cycles between the tag pipeline operation. sarbcf_csr_lc_op0op1_disable Due to w/a of bug#3665919, the OP1 has to be set to "1"
1	RW	0b	Core	**L3 OP1 Disable Mode (L3OP1DIS):** L3 OP1 Disable Mode (L3OP1DIS): OP1 in L3 can be disabled which means there will be one Command transferred to the Tag pipeline in 1X Domain sarbcf_csr_lc_op1_disable Note: If this bit is set Aging mode needs to be disabled as well.
0	RO	0b	Core	**Reserved (RSVD):** Reserved

8.1.7 L3CNTLREG2 - L3 Control Register2

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B020-B023h
Default Value: 00080040h
Access: RW; RO;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:28	RO	0000b	Core	**Reserved (RSVD):**
27	RW	0b	Core	**DC way assignment SLM Behavior (DCWASLMB):** DC way assignment SLM Behavior: In shared local memory mode DC is assigned into low b/w mode which requires it to be assigned non-matched ways of bigger banks and it will be hashed to 2 banks. sarbcf_csr_dc_slm_lowbw
26:21	RW	000000b	Core	**DC Way Assignment (DCWASS):** DC Way Assignment: Number of ways allocated for DC. Note this allocation is only for DC data types. 000000: 0KB000001: 8KB (4KB for GT1) 000010: 16KB (8KB for GT1) 111111: 504KB (252KB for GT1)Note: This field must be 0KB is All L3 Client Pool is non-zero. sarbcf_csr_dc_size[5:0]
20	RW	0b	Core	**RO Client Pool SLM Behavior (ROCPSLMB):** RO Client Pool SLM Behavior: In shared local memory mode Read Only Clients are assigned into low b/w mode which requires it to be assigned non-matched ways of bigger banks and it will be hashed to 2 banks. sarbcf_csr_ro_slm_lowbw
19:14	RW	100000b	Core	**Read Only Client Pool (RDOCPL):** Read Only Client Pool: Number of ways allocated for ROnly L3 clients. This is a combined pool for all RO clients. 000000: 0KB 000001: 8KB (4KB for GT1) 000010: 16KB (8KB for GT1) 111111: 504KB (252KB for GT1) Note: If all ROClient pool is non-zero, than Inst/state, Const and Texture client allocation should have 0KB allocation. sarbcf_csr_ro_size[5:0]
13:8	RW	000000b	Core	**All L3 Client Pool (ALL3CLPL):** All L3 Client Pool: Number of ways allocated for all L3 clients. This is a combined pool for all L3 clients.

Bit	Access	Default Value	RST/PWR	Description
				000000: 0KB 000001: 8KB (4KB for GT1) 000010: 16KB (8KB for GT1) 111111: 504KB (252KB for GT1) Note: If all L3 Client pool is non-zero, than all L3 client pools should have 0KB allocation. sarbcf_csr_l3_size[5:0]
7	RW	0b	Core	**URB SLM Behavior (URBSLMB):** URB SLM Behavior: In shared local memory mode URB is assigned into low b/w mode which requires it to be assigned non-matched ways of bigger banks and it will be hashed to 2 banks. sarbcf_csr_urb_slm_lowbw
6:1	RW	100000b	Core	**URB Allocation (URBALL):** URB Allocation: Number of ways allocated for URB usage 000000: 0KB 000001: 8KB (4KB for GT1) 000010: 16KB (8KB for GT1) 111111: 504KB (252KB for GT1) sarbcf_csr_urb_size[5:0]
0	RW	0b	Core	**SLM Mode Enable (SLMMENB):** SLM Mode Enable: When enabled, a 128KB region of L3 is reserved for SLM. This allocation is done on 2 banks with 64KB per half-slice. 0: SLM is disabled 1: SLM is enabled Note: For GT1, there is only one 64KB allocation for single half-slice. sarbcf_csr_slm_mode

8.1.8 L3CNTLREG3 - L3 Control Register3

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B024-B027h
Default Value: 00000000h
Access: RO; RW;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:22	RO	0000000000b	Core	**Reserved (RSVD):**
21	RW	0b	Core	**Textures Way Allocation SLM Behavior (TWALSLMB):** Textures Way Allocation SLM Behavior: In shared local memory mode Textures is assigned into low b/w mode which requires it to be assigned non-matched ways of bigger banks and it will be hashed to 2 banks. sarbcf_csr_tex_slm_lowbw
20:15	RW	000000b	Core	**Textures Way Allocation (TXWYALL):** Textures Way Allocation: Number of ways allocated for Textures. 000000: 0KB 000001: 8KB (4KB for GT1) 000010: 16KB (8KB for GT1) 111111: 504KB (252KB for GT1) Note: This field must be 0KB is All L3 Client Pool or Read-Only Client Pool is non-zero. sarbcf_csr_tex_size[5:0]
14	RW	0b	Core	**Constants Way Allocation SLM Behavior (CWASLMB):** Constants Way Allocation SLM Behavior: In shared local memory mode Instruction/state is assigned into low b/w mode which requires it to be assigned non-matched ways of bigger banks and it will be hashed to 2 banks. sarbcf_csr_const_slm_lowbw
13:8	RW	000000b	Core	**Constants Way Allocation (CTWALL):** Constants Way Allocation: Number of ways allocated for Constants. 000000: 0KB 000001: 8KB (4KB for GT1) 000010: 16KB (8KB for GT1) 111111: 504KB (252KB for GT1) Note: This field must be 0KB is All L3 Client Pool or Read-Only Client Pool is non-zero. sarbcf_csr_const_size[5:0]

Bit	Access	Default Value	RST/PWR	Description
7	RW	0b	Core	**Instruction/State Way Allocation SLM Behavior (ISWASLMB):** Instruction/State Way Allocation SLM Behavior: In shared local memory mode Instruction/state is assigned into low b/w mode which requires it to be assigned non-matched ways of bigger banks and it will be hashed to 2 banks. sarbcf_csr_is_slm_lowbw
6:1	RW	000000b	Core	**Instruction/State Way Allocation (ISWYALL):** Instruction/State Way Allocation: Number of ways allocated for Instruction/State usage 000000: 0KB 000001: 8KB (4KB for GT1) 000010: 16KB (8KB for GT1) 111111: 504KB (252KB for GT1) Note: This field must be 0KB is All L3 Client Pool or Read-Only Client Pool is non-zero. sarbcf_csr_is_size[5:0]
0	RO	0b	Core	**Reserved (RSVD):** Reserved

8.1.9 L3SLMREG - L3 SLM Register

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B028-B02Bh
Default Value: 40000000h
Access: RO; RW;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31	RW	0b	Core	**Disable Periodic SLM/SQ slot allocation (DPSLMALL):** Disable Periodic SLM/SQ slot allocation: When cfg_lslm_livelock_fairarb_dis=1 lslm unit will always have the higher priority and lslm_lsqc_block to lsqcunit is asserted as long as there are requests in SLM FIFO sarbcf_csr_lslm_livelock_fairarb_dis
30:27	RW	1000b	Core	**LSLM_SQ_PENDING_MAX (LSLMSQPEND):** If lslmunit has read data to be sent to lcbrunit this cfg register specifies the maximum number of clocks for which LSLMunit can block SQ request from being sent to lcbrunit

Bit	Access	Default Value	RST/PWR	Description
				Default value = 8 sarbcf_csr_lslm_sqpend_max[3:0]
26:0	RO	0000000h	Core	**Reserved (RSVD):**

8.1.10 GARBCNTLREG - Arbiter Control Register

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B02C-B02Fh
Default Value: 29000000h
Access: RO; RW;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31				
30	RW	0b	Core	**Disables hashing function (DISHHF):** Disables hashing function to generate bank_id[1:0] for L3$ bank accessing, and forces the use of address[7:6] for bank_id[1:0]. 0 : (default) Hash function enabled to generate L3$ bank IDs. 1 : L3$ address[7:6] used as L3$ bank IDs. sarbcf_csr_l3bankidhashdis
29:28	RW	10b	Core	**Arbitration priority order between RCC and MSC (APORM):** Arbitration priority order between RCC and MSC. 00/11: Invalid; default setting used 10 : Default setting; RCC < MSC (i.e., MSC has higher priority) 01: RCC > MSC (i.e., RCC has higher priority) sarbcf_csr_rcc_msc_pri[1:0]
27:22	RW	100100b	Core	**Arbitration priority order between RCZ, STC, and HIZ (APORSH):** Arbitration priority order between RCZ, STC, and HIZ. 100100 : Default setting; RCZ < STC < HIZ (i.e., RCZ has lowest priority; HIZ has highest priority) 100001 : RCZ < HIZ < STC0 11000 : STC < RCZ < HIZ0 10010 : STC < HIZ < RCZ 001001 : HIZ < RCZ < STC 000110 : HIZ < STC < RCZ

Bit	Access	Default Value	RST/PWR	Description
				Note: Others settings are invalid, and result in use of default. sarbcf_csr_rcz_stc_hiz_pri[5:0]
21:19	RW	000b	Core	**Write data port arbitration priority between Z client writes and L3$ evictions (WDPAGAPZ):** Write data port arbitration priority between Z client writes and L3$ evictions. 000 : L3:Z=1:0 - i.e., L3$ evictions > Z writes; Fixed priority; Default setting. 001 : L3:Z=1:1 - i.e., L3$ evictions = Z writes; Round-robin priority 010 : L3:Z=2:1 - i.e., Z writes will have higher priority after 2 L3$ evictions have been continuously granted earlier 011 : L3:Z=3:1 - i.e., Z writes will have higher priority after 3 L3$ evictions have been continuously granted earlier ... 111 : L3:Z=7:1 - i.e., Z writes will have higher priority after 7 L3$ evictions have been continuously granted sarbcf_csr_wdpagapz[2:0]
18:16	RW	000b	Core	**Write data port arbitration priority between C client writes and Z/L3$ writes/evictions (WDPAGAPC):** Write data port arbitration priority between C client writes and Z/L3$ writes/evictions. 000 : L3:Z=1:0 – i.e., Z/L3$ writes/evictions > C writes; Fixed priority; Default setting. 001 : L3:Z=1:1 – i.e., Z/L3$ writes/evictions = C writes; Round-robin priority 010 : L3:Z=2:1 – i.e., C writes will have higher priority after 2 Z/L3$ writes/evictions have been continuously granted earlier 011 : L3:Z=3:1 – i.e., C writes will have higher priority after 3 Z/L3$ writes/evictions have been continuously granted earlier ... 111 : L3:Z=7:1 – i.e., C writes will have higher priority after 7 Z/L3$ writes/evictions have been continuously granted earlier sarbcf_csr_wdpagapc[2:0]
15:0	RO	0000h	Core	**Reserved (RSVD):**

8.1.11 L3SQCREG4 - L3 SQC register 4

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B034-B037h
Default Value: 08000000h
Access: RWHC; RO; RW;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31	RO	0b	Core	**Reserved (RSVD):**
30	RW	0b	Core	**L3SQ Mode select for FF32 cycles on Crossbar 0/2 (SQFF32MODE):** L3SQ Mode select for FF32 cycles on Crossbar 0/2 (SQFF32MODE): Selects the mode in which SQ arbitrates FF32 cycles versus Half slice clients during Crossbar 2 arbitration. Qualified with the assertion of the crossbar 0/2 enable (SQFF32EN). In mode 0, crossbar 0/2 arbitration operates normally, and is determined strictly by the priority bits assigned by client (SQ*PRIVAL). In mode 1, FF32 priority values are determined by the FF32 priority override (SQFF32PRIOVER) when an indication is observed from GAFS (greater than 32 FF cycles are pending) 1 = all FF32 cycles will receive the overridden value instead of the default client priority value when indicated by GAFS counter 0 = all FF32 cycles will receive normal priority values (default)
29:28	RW	00b	Core	**L3SQ Priority Override for FF32 cycles on Crossbar 0/2 (SQFF32PRIOVER):** L3SQ Priority Override for FF32 cycles on Crossbar 0/2 (SQFF32PRIOVER): Identifies the priority value for all FF32 cycles that are initiated by GAL3. Priority is used in the L3 Super Queue (L3SQ). Qualified with the assertion of the crossbar 0/2 enable (SQFF32EN) and the assertion of ff32 mode select (SQFF32MODE). 00 = Priority 0 (default) 01 = Priority 1 10 = Priority 2 11 = Priority 3
27	RW	1b	Core	**L3SQ URB Read CAM Match Disable (SQURBRDCAMDIS):** L3SQ URB Read CAM Match Disable (SQURBRDCAMDIS): Disables the L3SQ Cam Match ability for URB Reads. By disabling, this allows a performance mode where URB reads are not dependent upon one another but only on any previous URB writes to the same address. This allows many URB reads to the same cacheline at any given time instead of serializing the requests. 1 = URB Read CAM matching is disabled; multiple URB reads to the same cacheline are allowed to be concurrent(default) 0 = URB Read CAM matching is enabled; multiple URB reads to the same

Bit	Access	Default Value	RST/PWR	Description
				cacheline are serialized
26	RWHC	0b	Core	**LSQC reset fcount (LSQCRFCNT):** self clearing register bit - Write to this register generates 1 clock pulse sarbcf_csr_lsqc_rst_fcount to lsqc and also used to clear the register sarbcf_csr_lsqc_rst_fcount_lvl is output of configdb.
25	RWHC	0b	Core	**LSLM1 reset fcount (LSLM1RFCNT):** self clearing register bit - Write to this register generates 1 clock pulse sarbcf_csr_lslm1_rst_fcount goes to lslm1 and also used to clear this register bit sarbcf_csr_lslm1_rst_fcount_lvl is ouput of configdb
24	RWHC	0b	Core	**LSLM3 reset fcount (LSLM3RFCNT):** self clearing register bit - Write to this register generates 1 clock pulse sarbcf_csr_lslm3_rst_fcount goes to lslm3 and used to clear this register too sarbcf_csr_lslm1_rst_fcount_lvl is output of configdb.
23:0	RO	000000h	Core	**reserved (RSVD):**

8.1.12 SCRATCH1 - SCRATCH1

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B038-B03Bh
Default Value: 00000000h
Access: RWHC; RO; RW;
Size: 32 bits

Bit	Access	Default Value	RST/PWR	Description
31:1	RW	00000000h	Core	**SCRATCH (SCRATCH):**
0	RW	0b	Core	**L3 Starvation Limit:** sarbcf_csr_hs_l3starvlimit - 0 : GAHS* could get starved if HDC reads are streaming 1 : GAHS* arbitrated at higher priority than HDC* read req serviced in 7 clocks

8.1.13 L3B0REG1 - L3 bank0 reg1 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B074-B077h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error.
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error.
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.14 L3B0REG2 - L3 bank0 reg2 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B078-B07Bh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error.
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error.
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.15 L3B0REG3 - L3 bank0 reg3 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B07C-B07Fh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.16 L3B0REG4 - L3 bank0 reg4 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B080-B083h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.17 L3B0REG5 - L3 bank0 reg5 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B084-B087h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.18 L3B0REG6 - L3 bank0 reg6 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B088-B08Bh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.19 L3B0REG7 - L3 bank0 reg7 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B08C-B08Fh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.20 L3B1REG0 - L3 bank1 reg0 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B090-B093h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.21 L3B1REG1 - L3 bank1 reg1 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B094-B097h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.22 L3B1REG2 - L3 bank1 reg2 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B098-B09Bh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.23 L3B1REG3 - L3 bank1 reg3 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B09C-B09Fh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.24 L3B1REG4 - L3 bank1 reg4 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0A0-B0A3h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.25 L3B1REG5 - L3 bank1 reg5 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0A4-B0A7h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.26 L3B1REG6 - L3 bank1 reg6 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0A8-B0ABh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.27 L3B1REG7 - L3 bank1 reg7 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0AC-B0AFh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.28 L3B2REG0 - L3 bank2 reg0 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0B0-B0B3h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.29 L3B2REG1 - L3 bank2 reg1 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0B4-B0B7h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.30 L3B2REG2 - L3 bank2 reg2 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0B8-B0BBh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.31 L3B2REG3 - L3 bank2 reg3 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0BC-B0BFh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.32 L3B2REG4 - L3 bank2 reg4 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0C0-B0C3h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.33 L3B2REG5 - L3 bank2 reg5 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0C4-B0C7h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.34 L3B2REG6 - L3 bank2 reg6 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0C8-B0CBh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.35 L3B2REG7 - L3 bank2 reg7 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0CC-B0CFh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.36 L3B3REG0 - L3 bank3 reg0 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0D0-B0D3h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.37 L3B3REG1 - L3 bank3 reg1 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0D4-B0D7h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.38 L3B3REG2 - L3 bank3 reg2 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0D8-B0DBh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.39 L3B3REG3 - L3 bank3 reg3 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0DC-B0DFh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.40 L3B3REG4 - L3 bank3 reg4 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0E0-B0E3h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.41 L3B3REG5 - L3 bank3 reg5 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0E4-B0E7h
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.42 L3B3REG6 - L3 bank3 reg6 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0E8-B0EBh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.43 L3B3REG7 - L3 bank3 reg7 log error

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B0EC-B0EFh
Default Value: 00000000h
Access: RW; RO;
Size: 32 bits

The ERROR LOG registers of L3 will maintain the bad row information for each of the 16KB subbank groups. The LOG will be programmed by driver before any workloads are submitted. The contents of the LOG register will be context Save&Restored by h/w around rc6 events.

Bit	Access	Default Value	RST/PWR	Description
31:21	RW	000h	Core	**Row Number for Error1 (RNUMERR1):** Row Number for Error1: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
20:17	RO	0000b	Core	**Reserved (RSVD):**
16	RW	0b	Core	**Valid Error 1 (VLDERR1):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.
15:5	RW	000h	Core	**Row Number for Error0 (RNUMERR0):** Row Number for Error0: The physical row number where the parity error has been detected. The number of rows vary between 4K vs 8K/16K subbanks which requires 10bits vs 11bits respectively. This field contains the row# with the error
4:1	RO	0000b	Core	**Reserved (RSVD):**
0	RW	0b	Core	**Valid Error 0 (VLDERR0):** Valid Error: The error located in field 15:5 is valid and corresponding logical 16KB group should bypass this row.

8.1.44 SARBCSR - SARB config save msg

B/D/F/Type: 0/0/0/SARBunit_Config
Address Offset: B1FC-B1FFh
Default Value: 00000000h
Access: RWHC; RO;
Size: 32 bits

This register is not context saved

Bit	Access	Default Value	RST/PWR	Description
31:2	RO	00000000h	Core	**Reserved (RSVD):**
1	RWHC	0b	Core	**Context restore ack (CTXRSTRACK):** A write from cs to this bit along with mask bit 17 will prompt sarb to ack ctx restore ack . sarb_ctx_restore - ctx restore from cs clr_sarb_ctx_restore - sarb clr this bit.
0	RWHC	0b	Core	**Context save bit (SARBCS):** A write from cs to this bit along with mask bit 16 will prompt sarb to start context save to cs. sarb_ctx_save - ctx save from cs clr_sarb_ctx_save - sarb clr this bit once ctx save sm kicks in .

Revision History

Revision Number	Description	Revision Date
1.0	First 2012 OpenSource edition	May 2012

§§