# Programmers Model

## Transitions for Way Group State

Each waygroup can exist in one of three states. It can be enabled for Cache, enabled for RAM, or disabled. When disabled, it may be powered off. This section will describe the transitions between these states.



Figure 15: Each WayGroup supports 3 operation states

As seen in this figure, while 3 state exists, the transitions should always go through the Disabled state. Directly changing Cache Enabled to RAM enabled or vice versa is not supported.

### Cache Mode to Disabled Mode Transition

To transition a waygroup or multiple waygroups from cache mode to disabled mode, the first step is to disabled allocation for those waygroups. This will prevent any new lines from being added to the waygroups. This can be achieved by writing to the Global Allocation Way Group Enable register. By marking the appropriate waygroups as zero, no new lines will allocate into that cache.

After allocation is disabled, the lines present in that waygroup must be flushed and invalidate. The flush will write back any dirty data to memory, while the invalidation will prevent any new requests to an address from writing into the cache in that location. The flush engine is controlled by the LLC Way Flush register. The waygroups that you want to flush can be written to the wayflush register, and it will trigger the flush engine. Once the flush process has completed, the way flush control register will reset to zero. Since the flush sequence can take a significant amount of time, the status can be periodically polled by reading the register to determine if any of the bits are still set. If all are zeroed out, the flush engine has completed.

Once the flush engine has completed, those ways will be guaranteed to be empty and no new lines will be added. During this time, requests will still perform cache lookups for these waygroups, but at this point, they will always miss.

At this point, the Cache Way Enable register can be programmed to indicate that these ways are no longer enabled for Caching. Once they are disabled, new cache accesses will stop reading the cache tags and will ignore the contents of those pins. At this point, the corresponding tag arrays and data arrays can be powered off.

### RAM Mode to Disabled Mode

To transition a waygroup from RAM mode to the Disabled mode, the hardware transition is straightforward. A write to the RAM Way Enable control register can mark the waygroup as disabled.

Software may need more elaborate control sequences. When the waygroup transitions to disabled, the prior content of the RAM is lost. If that data needs to be preserved, a copy sequence must be performed by software to move the data from the RAM to a location in memory. Before this sequence happens, software should coordinate traffic so that new requests to the RAM region are not expected.

### Disabled Mode to Cache Mode

To transition from disabled mode to cache mode, the LLC must go through a sequence where the tag RAMs are invalidated, where they are enabled for use, and then finally they are enabled for allocation.

The first part of this sequence must ensure that the corresponding tag arrays are invalidated. If the ways were previously invalidated and the state was maintained (there was no power sequences, for instance), no further action is required. If the tags are in an unknown state, or have just powered on, an invalidation sequence must occur. The LLC Tag Invalidation Engine control register can be used to initiate an invalidation of the tags for the waygroups in question. The registers can be polled for status to determine when it has completed.

Once invalidated, the cache ways can be enabled for cache mode accesses by writing to the Cache Way Enable register. Once set, those waygroups will be looked up by any cache access. However, the entries will be invalidated until the allocation enables are set up. The allocation controls can be set before the Cache Way Enable register is set because allocation is limited to waygroups that are enabled.

### Disabled Mode to RAM Mode

To transition a disabled waygroup to RAM mode requires three steps. First, the data contents must be overridden. Second, security permissions must be determined for the RAM waygroups, and finally, the waygroups should be set to RAM mode.

The data invalidation must happen in order to ensure that any data that was previously stored into the data array (either cache data or RAM data) is completely invalidated. Without this step, it is possible for the RAM mode to have visibility of unrelated data, including potentially secure data. Invalidation is needed to ensure security is preserved.

Data invalidation happens by writing to the Data Invalidation Engine Control register, indicating which waygroups should be invalidated. This register can also be polled for status.

Once invalidation has completed, but before the RAM mode enable occurs, the security permission must be decided. This can be controlled by writing to the Scratchpad Ram Way Group Security register. This register has one bit per way group, with a value of 1 requiring accesses to be secure, and a value of 0 requiring non-secure.

Finally, the RAM mode can be enabled by setting the Scratchpad RAM Way Group Enable register.

## Register-based Access of RAMs

Both the Data RAMs and Tag RAMs are accessible through register-based accesses. This allows a backdoor method of reading or writing the arrays, which can be useful for debug, DFT follow up, or even for controlling stimulus in a test.

The register-based accesses use two sets of registers. It uses the indirect content registers, and the indirect trigger register. The content registers hold data to be written to the RAMs, or data read from the RAMs. The indirect trigger actually performs the access and indicates RAM address (index and way).

There are 4 kinds of accesses applicable to each of the LLC RAMs. The operations are read, write raw, write+ecc, and read-modify-write.

The read request is triggered by a write to the indirect trigger indicating it should perform a read. It will read the specified RAM location and copy the contents of that RAM into the indirect content registers. The completion of the trigger access occurs only after the RAM access has occurred, so there is no need to poll for completion. Once the trigger access is done, software can load the content registers to view the data in the RAM.

The write request is initiated by a write to the indirect trigger, and it copies the values in the indirect content registers into the RAM. There are two variants of the write request. The first will write the entire array, including ECC, based on the indirect content register. The second variant will write the non-ECC data with the indirect content register but will perform an ECC generation to determine the checkbits to set for the ECC portion of the RAM.

The read-modify-write sequence performs an atomic access to the RAM where it reads the content, XORs the content with the indirect content registers, and writes it back. This allows an atomic method of flipping a specified number of bits. This is useful for forcing the generation of single or double bit errors. The read-modify-write can be triggered during normal operation to introduce single or double bit errors. The operation is atomic, so it will modify the existing content of the RAM without any chance of that entry being modified in the middle of the sequence.

## LLC Allocation Controls

There are several programmable registers that can change the allocation properties of the LLC. These registers correspond to the following NocStudio LLC properties:

* llc\_class\*\_alloc\_waygroups
* llc\_class\_read\_allocate
* llc\_class\_read\_allocate\_use\_arcache
* llc\_class\_write\_allocate
* llc\_class\_write\_allocate\_use\_awcache

The NocStudio properties set the initial values of these registers, but the registers can be reprogrammed at any time to modify the allocation behavior of the system. These registers control which ways an LLC Allocation Class is allowed to allocate into, and whether they should allocate on reads or writes.

These allocation controls are modified by which ways are enabled for cache mode. If a way is disabled for cache mode, programming it to be allocated into will have no effect. This means these properties do not need to be modified as waygroups are enabled or disabled for cache use.

## LLC Host Registers

### LLC\_ALLOC\_ARCACHE\_EN

This register holds one bit for each of the LLC Allocation Classes. If the bit is marked as one, this indicates that read allocation for that LLC Allocation Class is controlled by the ARCACHE bits. If marked as zero, it means the allocation will be controlled by the llc\_alloc\_rd\_en register. The default of this register is configured within NocStudio.

Attribute: RW

Security: Non secure

Bit field description:

* **ARCACHE\_Allocation\_Enable** [7:0] - Read allocation for that LLC Allocation Class is controlled by the ARCACHE bits/llc\_alloc\_rd\_en register.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | ARCACHE\_Allocation\_Enable | | | | | | | |

Table 1 LLC\_ALLOC\_ARCACHE\_EN register

### LLC\_ALLOC\_AWCACHE\_EN

This register holds one bit for each of the LLC Allocation Classes. If the bit is marked as one, this indicates that write allocation for that LLC Allocation Class is controlled by the AWCACHE bits. If marked as zero, it means the allocation will be controlled by the llc\_alloc\_wr\_en register. The default of this register is configured within NocStudio.

Attribute: RW

Security: Non secure

Bit field description:

* **AWCACHE\_Allocation\_Enable** [7:0] - Write allocation for that LLC Allocation Class is controlled by the AWCACHE bits/llc\_alloc\_wr\_en register.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | AWCACHE\_Allocation\_Enable | | | | | | | |

Table 2 LLC\_ALLOC\_AWCACHE\_EN register

### LLC\_ALLOC\_RD\_EN

This register holds one bit for each of the LLC Allocation Classes. The use of this register is controlled by the llc\_alloc\_arcache\_en register, which indicates whether ARCACHE bits should be used for allocation, or whether this register should decide on allocation. If this register is used for an LLC Allocation Class, a value of one will indicate that reads should allocate into the LLC. A value of zero indicates that reads should not allocate.

Attribute: RW

Security: Non secure

Bit field description:

* **Read\_Allocation\_Enable** [7:0] - Reads should/should not allocate into the LLC

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | Read\_Allocation\_Enable | | | | | | | |

Table 3 LLC\_ALLOC\_RD\_EN register

### LLC\_ALLOC\_WR\_EN

This register holds one bit for each of the LLC Allocation Classes. The use of this register is controlled by the llc\_alloc\_awcache\_en register, which indicates whether AWCACHE bits should be used for allocation, or whether this register should decide on allocation. If this register is used for an LLC Allocation Class, a value of one will indicate that writes should allocate into the LLC. A value of zero indicates that writes should not allocate.

Attribute: RW

Security: Non secure

Bit field description:

* **Write\_Allocation\_Enable** [7:0] - Writes should/should not allocate into the LLC.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | Write\_Allocation\_Enable | | | | | | | |

Table 4 LLC\_ALLOC\_WR\_EN register

### LLC\_CACHE\_WAY\_ENABLE

This register indicates whether a way is enabled for cache access. If enabled, a cache lookup will read the associated Tag values and perform an address comparison. If disabled, the Tags won't be accessed and the contents of the Tags won't be compared.

This allows the Tags to be powered down or the lines to be used for RAM access. The register has one bit per way, allowing each way to be individually enabled or disabled. A value of 1 indicates that the way is enabled, while a value of 0 indicates it is disabled. All ways are enabled by default. Before disabling a cache way, the way must be disabled in the llc\_global\_alloc register, and the contents should be flushed.

This register requires secure access, since it disables ways of the cache, potentially modifying the contents of secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Cache\_Way\_Enable** [31:0] - Cache Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Cache\_Way\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 5 LLC\_CACHE\_WAY\_ENABLE register

### LLC\_CLASS\_ALLOC

The class allocation control registers are used to specify which associative ways can be written to. Each master in the system belongs to an LLC class, and each class allocation control register indicates which ways that class of agents can allocate into.

These registers can be used to provide dedicated associativity for different agents or groups of agents. The default value of these registers indicates that all ways are accessible by all agents, with a value of one indicating allocation is allowed. Setting the value to zero will disable allocation for an agent.

It is permissible to turn off allocation for all ways, which will prevent any accesses from that class from allocating into the cache.

Note that the llc\_global\_alloc register can override these values. If global allocation is disabled for a way, none of the agents can allocate into those ways regardless of what the llc\_class\_allocate registers indicate.

Attribute: RW

Security: Non secure

Bit field description:

* **WGE\_0** [7:0] - Class 0 Allocation Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | WGE\_0 | | | | | | | |

Table 6 LLC\_CLASS\_ALLOC register

### LLC\_DATA\_INV\_CTL

This register controls a state machine that can invalidate the contents of data array banks. Each way has a corresponding bit in this register. When the register is written, the invalidation engine will kick off and invalidate the data for each of the specified ways. Writing a value of 1 to a bit indicates that the corresponding way should be invalidated. Writing a value of 0 to a bit indicates that the content of that way shouldn't be invalidated.

The register can be read to determine the current status of the data bank invalidation sequence. When hardware has completed the invalidation sequence for a way, it will change the value of that register bit from 1 to 0. If the entire register has a value of 0, then the invalidation engine has completed. The reset value of this register is zero.

Invalidation of the data array is needed when switching between cache mode and scratchpad RAM mode, since the RAM mode allows direct access to the data. Any secure data that was stored in the cache may be visible to RAM mode accesses unless it is invalidated first. Similarly, if security permissions are removed for the Scratchpad RAM, the prior contents should be invalidated before removing the security check.

An invalidation sequence should be completed before a second sequence is requested.

This register requires secure access, since it invalidates data array, potentially modifying the contents of secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Data\_Bank\_Invalidation\_Enable** [31:0] - Data Bank Invalidation Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data\_Bank\_Invalidation\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 7 LLC\_DATA\_INV\_CTL register

### LLC\_ECC\_DATA\_ADDR

This is a status register that tracks the address of ECC errors that occur in the Data ram. If only one of either the SB bit or the DB bit is set in the ecc\_data\_info register, this address contains the address for that error. If both the SB and the DB ecc error bits are set in the ecc\_data\_info register, this register contains the address of the DB error.

Attribute: RW

Security: Non secure

Bit field description:

* **Address** [59:0] - Address of first detected error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | Address | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 8 LLC\_ECC\_DATA\_ADDR register

### LLC\_ECC\_DATA\_INFO

This is a status register that tracks ECC errors that occur in the Data array. The register will track the number of ECC errors, as well as whether single-bit or double-bit errors have been detected. If the SB bit is set, at least one single bit error has been detected. If the DB bit is set, at least one double-bit error has been detected.

Additionally, the register tracks information about the first error detected. It stores the index of the tag array that had the error, as well as the way group. It also tracks which half of the cache line failed, which is needed to identify the sub-bank that failed. If a double-bit error occurs after a single-bit error has already been recorded, the double-bit error will overwrite the content of the register. This is because double-bit errors are fatal, and the information about how a fatal error is more important that the information about a non-fatal error.

The register can be read for status, but can also be written. If the SB and DB bit are written with zeros, the sampling of the first detected error will happen as described above.

Attribute: RW

Security: Non secure

Bit field description:

* **Index** [45:32] - index of first detected error
* **ECC\_Count** [31:16] - number of ECC errors found
* **hlf** [9] - Which half of cache line reported error
* **way** [8:2] - Way group of first detected error
* **db** [1] - Detected double or multi bit error
* **sb** [0] - Detected single bit error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | Index | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC\_Count | | | | | | | | | | | | | | | | u | | | | | | hlf | way | | | | | | | db | sb |

Table 9 LLC\_ECC\_DATA\_INFO register

### LLC\_ECC\_DISABLE

This register allows ECC to be disabled for either the Data arrays or the Tag arrays. These are independently controlled. A bit value of 1 indicates that ECC is disabled. A bit value of 0 indicates ECC is enabled, if present. The register value resets to value 0, meaning ECC is enabled.

Attribute: RW

Security: Non secure

Bit field description:

* **D** [1] - Disable Data ECC Check/Correct
* **T** [0] - Disable Tag ECC Check/Correct

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | D | T |

Table 10 LLC\_ECC\_DISABLE register

### LLC\_ECC\_TAG\_ADDR

This is a status register that tracks the address of ECC errors that occur in the Tag ram. If only one of either the SB bit or the DB bit is set in the ecc\_tag\_info register, this address contains the address for that error. If both the SB and the DB ecc error bits are set in the ecc\_tag\_info register, this register contains the address of the DB error.

Attribute: RW

Security: Non secure

Bit field description:

* **Address** [59:0] - Address of first detected error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | Address | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 11 LLC\_ECC\_TAG\_ADDR register

### LLC\_ECC\_TAG\_INFO

This is a status register that tracks ECC errors that occur in the Tag array. The register will track the number of ECC errors, as well as whether single-bit or double-bit errors have been detected. If the SB bit is set, at least one single bit error has been detected. If the DB bit is set, at least one double-bit error has been detected.

Additionally, the register tracks information about the first error detected. It stores the index of the tag array that had the error, as well as the way group. If a double-bit error occurs after a single-bit error has already been recorded, the double-bit error will overwrite the content of the register. This is because double-bit errors are fatal, and the information about how a fatal error is more important that the information about a non-fatal error.

The register can be read for status, but can also be written. If the SB and DB bit are written with zeros, the sampling of the first detected error will happen as described above.

Attribute: RW

Security: Non secure

Bit field description:

* **Index** [45:32] - Index of first detected error
* **ECC\_Count** [31:16] - Number of ECC errors found
* **way** [6:2] - Way group of first detected error
* **db** [1] - Detected double or multi bit error
* **sb** [0] - Detected single bit error

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | Index | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ECC\_Count | | | | | | | | | | | | | | | | u | | | | | | | | | way | | | | | db | sb |

Table 12 LLC\_ECC\_TAG\_INFO register

### LLC\_EVENT\_COUNTER\_0

This register is the first of two event counters. When its event counter mask control enables certain events to be counted, they will increment this counter. When the counter overflows, in can produce an interrupt if the interrupt mask is enabled. This can be used to trap to software after a number of specified events has occurred.

The counter can be read or written. Writing the value can initialize the counter to a larger value which can speed up the point at which counter will overflow and the interrupt will be triggered.

Attribute: RW

Security: Non secure

Bit field description:

* **Event\_Counter\_Value\_0** [31:0] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Event\_Counter\_Value\_0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 13 LLC\_EVENT\_COUNTER\_0 register

### LLC\_EVENT\_COUNTER\_1

This register is the first of two event counters. When its event counter mask control enables certain events to be counted, they will increment this counter. When the counter overflows, in can produce an interrupt if the interrupt mask is enabled. This can be used to trap to software after a number of specified events has occurred.

The counter can be read or written. Writing the value can initialize the counter to a larger value which can speed up the point at which counter will overflow and the interrupt will be triggered.

Attribute: RW

Security: Non secure

Bit field description:

* **Event\_Counter\_Value\_1** [31:0] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Event\_Counter\_Value\_1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 14 LLC\_EVENT\_COUNTER\_1 register

### LLC\_EVENT\_COUNTER\_MASK\_0

This register is used to program its corresponding event counter. Each bit of this register enables the performance counter to increment if the event occurs. A value of 1 for a bit indicates that this event should be counted.

If multiple bits are set to 1, the logic will only count if all of the corresponding events occur on the same cycle. This allows for combinations of events, such as a cache miss that causes an eviction. The events that can be counted are all related to cache accesses and occur in the same cycle of the pipeline, so they can be combined easily.

When an event satisfies all of the requirements, the llc\_event\_counter\_0 register will be updated. A value of 1 indicates that the event is selected for counting. A value of 0 the event is not selected. By default, this register will be set to 0 for all bits, indicating no event counting should occur.

Attribute: RW

Security: Non secure

Bit field description:

* **e27** [27] - Allocate conflict
* **e26** [26] - Indx after indx tmp conflict
* **e25** [25] - Wr after alloc conflict
* **e24** [24] - Wr after evict conflict
* **e23** [23] - Rd after rd alloc conflict
* **e22** [22] - Rd alloc after wr conflict
* **e21** [21] - Rd after evict
* **e20** [20] - Tag update
* **e19** [19] - Request CleanUnique
* **e18** [18] - External snoop
* **e17** [17] - Excl wr fails store conditional
* **e16** [16] - Exclusive write
* **e15** [15] - Atomic op
* **e14** [14] - Write through
* **e13** [13] - Rd dealloc on dirty
* **e12** [12] - Rd dealloc
* **e11** [11] - Partial line wr
* **e10** [10] - Fetch due to partial wr
* **e9** [9] - Retry access
* **e8** [8] - Retry needed
* **e7** [7] - Eviction
* **e6** [6] - Cache maint op
* **e5** [5] - Partial write
* **e4** [4] - Cache miss
* **e3** [3] - Cache hit
* **e2** [2] - Scratchpad access
* **e1** [1] - Cache write
* **e0** [0] - Cache read

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | e27 | e26 | e25 | e24 | e23 | e22 | e21 | e20 | e19 | e18 | e17 | e16 | e15 | e14 | e13 | e12 | e11 | e10 | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table 15 LLC\_EVENT\_COUNTER\_MASK\_0 register

### LLC\_EVENT\_COUNTER\_MASK\_1

This register is used to program its corresponding event counter. Each bit of this register enables the performance counter to increment if the event occurs. A value of 1 for a bit indicates that this event should be counted.

If multiple bits are set to 1, the logic will only count if all of the corresponding events occur on the same cycle. This allows for combinations of events, such as a cache miss that causes an eviction. The events that can be counted are all related to cache accesses and occur in the same cycle of the pipeline, so they can be combined easily.

When an event satisfies all of the requirements, the llc\_event\_counter\_1 register will be updated. A value of 1 indicates that the event is selected for counting. A value of 0 the event is not selected. By default, this register will be set to 0 for all bits, indicating no event counting should occur.

Attribute: RW

Security: Non secure

Bit field description:

* **e27** [27] - Allocate conflict
* **e26** [26] - Indx after indx tmp conflict
* **e25** [25] - Wr after alloc conflict
* **e24** [24] - Wr after evict conflict
* **e23** [23] - Rd after rd alloc conflict
* **e22** [22] - Rd alloc after wr conflict
* **e21** [21] - Rd after evict
* **e20** [20] - Tag update
* **e19** [19] - Request CleanUnique
* **e18** [18] - External snoop
* **e17** [17] - Excl wr fails store conditional
* **e16** [16] - Exclusive write
* **e15** [15] - Atomic op
* **e14** [14] - Write through
* **e13** [13] - Rd dealloc on dirty
* **e12** [12] - Rd dealloc
* **e11** [11] - Partial line wr
* **e10** [10] - Fetch due to partial wr
* **e9** [9] - Retry access
* **e8** [8] - Retry needed
* **e7** [7] - Eviction
* **e6** [6] - Cache maint op
* **e5** [5] - Partial write
* **e4** [4] - Cache miss
* **e3** [3] - Cache hit
* **e2** [2] - Scratchpad access
* **e1** [1] - Cache write
* **e0** [0] - Cache read

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | e27 | e26 | e25 | e24 | e23 | e22 | e21 | e20 | e19 | e18 | e17 | e16 | e15 | e14 | e13 | e12 | e11 | e10 | e9 | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table 16 LLC\_EVENT\_COUNTER\_MASK\_1 register

### LLC\_FORCE\_PARTIAL\_WR\_ALLOC

This register holds one bit for each of the LLC Allocation Classes. When set to 0, all partial write misses will follow normal write allocation rules (llc\_alloc\_wr\_en or awcache). When set to 1, all partial write misses will be forced to allocate (fetch first followed by merge). All partial write hits will be merged into the cache regardless of the configuration.

Attribute: RW

Security: Non secure

Bit field description:

* **Force\_Partial\_Write\_Allocation** [7:0] - Force\_Partial\_Write\_Allocation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | Force\_Partial\_Write\_Allocation | | | | | | | |

Table 17 LLC\_FORCE\_PARTIAL\_WR\_ALLOC register

### LLC\_GLOBAL\_ALLOC

This register controls whether lines can be allocated into a way by any agent.

If a way is disabled from allocation in this register, no agents can allocate even if the llc\_class\_allocate registers are set. This register is used as part of a sequence to remove ways from use by the cache for either Scratchpad RAM usage, or for power gating. By removing allocation ability, a flush engine can remove the existing contents of the line without fear that new entries will be added during or after the flush.

The default value of this register enables allocation for all ways of the cache, and so each bit corresponding to a way is set to 1. To disable allocation, the bits should be set to 0.

Attribute: RW

Security: Non secure

Bit field description:

* **Global\_Allocation\_Way\_Enable** [31:0] - Global Allocation Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Global\_Allocation\_Way\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 18 LLC\_GLOBAL\_ALLOC register

### LLC\_INDIRECT\_RAM\_CONT

This is the indirect access RAM content register. It is used in conjunction with the indirect access trigger register. On an indirect read, data is written to this register. On an indirect write, content from this register is written into the RAM. On a read-modify-write, content from this register is used for the XOR function.

Since the RAM data width may be larger than 64 bits, multiple registers are used to hold the data. Any bits beyond the data width are unused.

This register requires secure access, since it can be used to modify or observe the contents of data.

Attribute: RW

Security: Secure access only

Bit field description:

* **RAM\_content** [63:0] -

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| RAM\_content | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RAM\_content | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 19 LLC\_INDIRECT\_RAM\_CONT register

### LLC\_INDIRECT\_TRIGGER

This register is the indirect access trigger. Indirect access is a mechanism that allows register-based access to the RAM arrays. This can be used for testing RAM bits or reading content on an error condition.

The indirect access is based on a content+trigger mechanism. For writes, the content register is written first to accumulate the data that should be written. Once the content is ready, the trigger register is used to kick off the hardware write mechanism. For reads, the trigger register kicks off a read, and provides data by placing the result into the content registers where it can be accessed.

The indirect access supports 4 sub-commands.

1. Read Raw data. When triggered, a read to the RAM array will be performed and the resulting data, without ECC correction, will be copied into the content register.
2. Write Raw Data. When triggered, the content register values will be written into the RAM. This will include the ECC bits if present.
3. Write Data with Generated ECC. When triggered, this will write to the RAM entry. The content register will be used to specify the data to be written. However, if ECC hardware is present, the ECC bits will be generated based on the data instead of coming from the content register. This allows the RAM entry to be written with correct ECC value without needing to calculate it first.
4. Read-Modify-Write. This command will perform a specific kind of read-modify-write operation on a RAM entry. It will read the content of the RAM, XOR that content with the indirect content register, and write the combined value into the same RAM entry. This can be used to introduce single or double bit errors into the directory to test error detection and handling. The content register will not be modified during this operation, so it can be used to introduce errors into multiple lines.

Each of the indirect access commands can be issued during normal operation, but the Write commands can have side-effects that break coherency functionality. The Read Raw is not disruptive, and the Read-Modify-Write can be performed atomically so single-bit errors can be introduced while maintaining functionality. The indirect access trigger registers are readable and writeable. To trigger the RAM access, this register must be written. Reads will not have side-effects and will only return the current value of the trigger register.

The trigger register has a number of fields that must be set correctly. The CMD field indicates which kind of indirect access to perform. The WAY field indicates which way group to access. The TYP field indicates whether the Data array or Tag array should be accessed. If the Data array is accessed, the hlf bit indicates which sub-bank is accessed. The RAM index indicates the entry to access within the RAM.

This register requires secure access, since it can be used to modify or observe the contents of data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Index** [31:11] - Index of RAM to access
* **way** [10:9] - Way position in the waygroup
* **hlf** [8] - Which half of cache line reported error
* **waygroup** [7:3] - Way group of first detected error
* **typ** [2] -   
  0: Tag array access  
  1: Data array access
* **cmd** [1:0] -   
  00: Read raw array content including any ECC bits and copy to RAM Content register  
  01: Write RAM content register directly into array  
  10: Write RAM content minus ECC bits to array, use ECC generation logic to set ECC bits in array  
  11: Read-modify-write. Read array content, XOR with RAM content register, and write modified data into array

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Index | | | | | | | | | | | | | | | | | | | | | way | | hlf | waygroup | | | | | typ | cmd | |

Table 20 LLC\_INDIRECT\_TRIGGER register

### LLC\_INTERRUPT\_ERR

This is a status register that tracks the interrupt generating events. This includes multi-bit ECC error, single-bit ECC error, RAM mode disallowed accesses, and event counter overflow. When these events occur, this register is updated and will hold the bit value until cleared. It can be cleared by writing to the register. To allow per-bit clearing control, the write value should use a value of 1 when it doesn't want to make a change, or a bit value of 0 when it wants to clear.

Attribute: WZC

Security: Non secure

Bit field description:

* **e8** [8] - Data Parity Error
* **e7** [7] - Register Parity Error
* **e6** [6] - Event Counter Overflow
* **e5** [5] - Scratchpad Security Failure status
* **e4** [4] - Scratchpad RAM Disabled status
* **e3** [3] - Data ECC Double Bit Error status
* **e2** [2] - Data ECC Single Bit Error status
* **e1** [1] - Tag ECC Double Bit Error status
* **e0** [0] - Tag ECC Single Bit Error status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table 21 LLC\_INTERRUPT\_ERR register

### LLC\_INTERRUPT\_FATAL\_MASK

This register determines which errors in llc\_interrupt\_err should be considered fatal. 1 is fatal and 0 is non-fatal.

Attribute: RW

Security: Non secure

Bit field description:

* **e8** [8] - Data Parity Error
* **e7** [7] - Register Parity Error
* **e6** [6] - Event Counter Overflow
* **e5** [5] - Scratchpad Security Failure status
* **e4** [4] - Scratchpad RAM Disabled status
* **e3** [3] - Data ECC Double Bit Error status
* **e2** [2] - Data ECC Single Bit Error status
* **e1** [1] - Tag ECC Double Bit Error status
* **e0** [0] - Tag ECC Single Bit Error status

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | e8 | e7 | e6 | e5 | e4 | e3 | e2 | e1 | e0 |

Table 22 LLC\_INTERRUPT\_FATAL\_MASK register

### LLC\_INTERRUPT\_MASK

This register is used for determining what kind of events can trigger an interrupt from the LLC.

A bit value of 1 indicates that the event will not send an interrupt. A bit value of 0 means the event will cause an interrupt. The default values are listed in the bit field description.

Attribute: RW

Security: Non secure

Bit field description:

* **m8** [8] -   
  1'b1: Data parity error interrupt disabled.  
  1'b0: Data parity error interrupt enabled (default).
* **m7** [7] -   
  1'b1: Register parity error interrupt disabled.  
  1'b0: Register parity error interrupt enabled (default).
* **m6** [6] -   
  1'b1: Event Counter Overflow interrupt disabled (default).  
  1'b0: Event Counter Overflow interrupt enabled.
* **m5** [5] -   
  1'b1: Scratchpad Security Check Failure interrupt disabled.  
  1'b0: Scratchpad Security Check Failure interrupt enabled (default).
* **m4** [4] -   
  1'b1: Scratchpad RAM Access while Disabled interrupt disabled.  
  1'b0: Scratchpad RAM Access while Disabled interrupt enabled (default).
* **m3** [3] -   
  1'b1: Data ECC Double Bit Error interrupt disabled.  
  1'b0: Data ECC Double Bit Error interrupt enabled (default).
* **m2** [2] -   
  1'b1: Data ECC Single Bit Error interrupt disabled (default).  
  1'b0: Data ECC Single Bit Error interrupt enabled.
* **m1** [1] -   
  1'b1: Tag ECC Double Bit Error interrupt disabled.  
  1'b0: Tag ECC Double Bit Error interrupt enabled (default).
* **m0** [0] -   
  1'b1: Tag ECC Single Bit Error interrupt disabled (default).  
  1'b0: Tag ECC Single Bit Error interrupt enabled.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | m8 | m7 | m6 | m5 | m4 | m3 | m2 | m1 | m0 |

Table 23 LLC\_INTERRUPT\_MASK register

### LLC\_OPERATION\_CONTROL

This register is used to change the behavior of the LLC in various ways, as listed in the description for each bit. The default values are listed in the bit field description.

Attribute: RW

Security: Non secure

Bit field description:

* **FAS** [1] -   
  1'b1: Force address serialization--only 1 operation can issue to memory for an address.   
  1'b0: Do not force address serialization. Default value.
* **LRU** [0] -   
  1'b1: Enable LRU updates on a cache hit.  
  1'b0: Disable LRU updates on a cache hit.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | FAS | LRU |

Table 24 LLC\_OPERATION\_CONTROL register

### LLC\_RAM\_ADDRESS\_BASE

This register indicates the system address offset of the RAM mode. The address range should always be allocated as the full size of the LLC capacity rounded up to a power of 2, and the address offset must be programmed to a naturally aligned address for that size. The default value of this register is the address range base specified during NoC construction.

This register requires secure access, since it controls the address range for RAM storage, implicitly changing the contents of that range.

Attribute: RW

Security: Secure access only

Bit field description:

* **Address**[63:0] - Address

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Address | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 25 LLC\_RAM\_ADDRESS\_BASE register.

### LLC\_RAM\_WAY\_ENABLE

This register is used to enable cache ways to be used as a Scratchpad RAM instead of a cache. The register indicates which of the ways should be used as a RAM instead of a cache. It is possible to use some of the LLC as a cache, and some as a RAM, by selecting which ways are used by each. By default, this register is set to 0 so that all ways are used as cache. To set this register, the lines must be removed from cache usage by the llc\_cache\_way\_enable register. Any cache contents should be flushed before enabling the RAM mode.

This register requires secure access, since it enables RAM storage, allowing previous content of the cache to be visible to scratchpad reads.

Attribute: RW

Security: Secure access only

Bit field description:

* **Scratchpad\_Ram\_Way\_Group\_Enable**[31:0] - Scratchpad Ram Way Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Scratchpad\_Ram\_Way\_Group\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 26 LLC\_RAM\_WAY\_ENABLE register

### LLC\_RAM\_WAY\_SECURE

This register allows the Scratchpad RAM to have trust-zone security checking. Each way can be individually controlled. If the security bit is set, only secure accesses (those with AxPROT [1] set to secure) can access that address. Non-secure accesses will be responded to with an error, and an interrupt will be triggered if the interrupt is enabled.

A value of 1 indicates secure accesses are required, while a value of 0 indicates no security check is needed (secure or non-secure accesses are enabled). By default, this register is 0, so no security checking occurs.

This register requires secure access, since it controls the security check for scratchpad RAM storage.

Attribute: RW

Security: Secure access only

Bit field description:

* **Scratchpad\_Ram\_Way\_Group\_Security**[31:0] - Scratchpad Ram Way Security

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Scratchpad\_Ram\_Way\_Group\_Security | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 27 LLC\_RAM\_WAY\_SECURE register

### LLC\_READ\_DEALLOCATE

This register holds one bit for each of the LLC Allocation Classes. When set to 1, the read hits for the LLC class will result in the cacheline being invalidated. Dirty cachelines will either be discarded or flushed depending on the setting in the LLC\_READ\_DISCARD\_DIRTY register. When set to 0, normal cache read behavior will apply.

Attribute: RW

Security: Non secure

Bit field description:

* **LLC\_READ\_DEALLOCATE** [7:0] - Per Allocation Class Read-And-Invalidate Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | LLC\_READ\_DEALLOCATE | | | | | | | |

Table 28 LLC\_READ\_DEALLOCATE register

### LLC\_READ\_DISCARD\_DIRTY

This register holds one bit for each of the LLC Allocation Classes. When set to 1, and the corresponding bit (class) in LLC\_READ\_DEALLOCATE is also set to 1, read hits on a dirty line will be discarded without write back.

Attribute: RW

Security: Non secure

Bit field description:

* **LLC\_READ\_DISCARD\_DIRTY**[7:0] - Per Allocation Class Control For Dirty Line When Read-And-Invalidate (Read Deallocate)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | LLC\_READ\_DISCARD\_DIRTY | | | | | | | |

Table 29 LLC\_READ\_DISCARD\_DIRTY register

### LLC\_SHDW\_PIPE\_STATUS

This register reports incoming request activity and resource availability for fulfilling those requests. Insight into pipeline stalls may be gleaned by polling this register and looking for chronically active request sources that do not have resource availability.

Attribute: R

Security: Non secure

Bit field description:

* **QREQS** [41:32] - Vector showing qualified request status to the LLC from various sources. These are active requests qualified by resource availability. Only when all required resources for a particular request source are 1 will the filtered request status be 1 (if the unfiltered request status is 1).

[9] evt  
 [8] extsnp  
 [7] tagupdate  
 [6] flush\_req  
 [5] rd\_sprt1  
 [4] rd\_sprt0  
 [3] retry\_rd  
 [2] wr\_sprt1  
 [1] wr\_sprt0  
 [0] retry\_wr

* **REQS** [25:16] - Vector showing request status to the LLC from various sources.

[9] evt  
 [8] extsnp  
 [7] tagupdate  
 [6] flush\_req  
 [5] rd\_sprt1  
 [4] rd\_sprt0  
 [3] retry\_rd  
 [2] wr\_sprt1  
 [1] wr\_sprt0  
 [0] retry\_wr

* **PS** [14] - TAC pipeline stall signal status. All incoming requests to lookup arbiter are held off while this signal is active, regardless of resource availability.
* **RSRCS** [13:0] - Vector showing the availability status of the various resources of the LLC.

[13] acem\_ar  
 [12] chim\_ar  
 [11] chim\_aww  
 [10] mprt\_aww  
  [9] mprt\_ar  
  [8] missq\_wr  
  [7] missq\_rd  
  [6] dac\_cmd  
  [5] dac\_wctl  
  [4] tac  
  [3] retry  
  [2] wrsp  
  [1] evict  
  [0] atomics

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | QREQS | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | REQS | | | | | | | | | | u | PS | RSRCS | | | | | | | | | | | | | |

Table 30 LLC\_SHDW\_PIPE\_STATUS register

### LLC\_SPARE

This register contains writable spare bits. Software may read and write these bits. They retain any value written to them (between reset events), but otherwise they have no functional effect.

Attribute: RW

Security: Non secure

Bit field description:

* **BITS** [31:0] - Spare Bits

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BITS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 31 LLC\_SPARE register

### LLC\_TAG\_INV\_CTL

The Tag Invalidation Control register triggers a state machine that will invalidate the contents of one or more way groups of the cache. The register has one bit per way group, and the bit vector written into this register will invalidate the corresponding way groups. A value of 1 will indicate that the corresponding way group should be invalidated. A value of 0 will indicate that the way group should not be invalidated. This per-way group control allows portions of the cache to be powered down and restarted later, with the ability to reset just the way groups that were powered down and powered back on.

A write to the register will kick off the invalidation engine, invalidating the specified way groups. A read of the register will indicate whether the invalidation is in progress. When the invalidation engine has completed, the bit vector will transition to a value of zero. So a read value of zero will indicate that the state machine has completed. The reset value of this register is zero. An invalidation sequence should be completed before a second sequence is requested.

This register requires secure access, since it invalidates tags, potentially modifying the contents of secure data.

Attribute: RW

Security: Secure access only

Bit field description:

* **Tag\_Way\_Group\_Invalidation\_Enable** [31:0] - Tag Way Group Invalidation Enable

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Tag\_Way\_Group\_Invalidation\_Enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 32 LLC\_TAG\_INV\_CTL register

### LLC\_WAY\_FLUSH

This register controls a state machine that flushes specified ways of the cache. The intent of this engine is to remove all content from the specified ways, pushing any dirty data that may exist to memory. It also invalidates clean lines. The Way Flush engine should be run while the llc\_cache\_way\_enable is still on for those ways so the contents are still accessible, but the llc\_global\_alloc register should have disabled the way for allocation. This ensures that as lines are removed from the cache, they won't unintentionally get added again. Clean lines are invalidated to ensure that dirty line writes do not write into the ways being flushed.

Writing the register will kick off the flush engine. If the write value specifies a bit value of 1, then that way group will be flushed. If the bit value written is zero, that way will not be flushed. When the sequence is completed, hardware will transition the bit values to zero. A register value of 0 indicates the state machine has complete. The default value for this register is zero.

A flush sequence should be completed before a second sequence is requested.

This register requires secure access, since it flushes cache lines.

Attribute: RW

Security: Secure access only

Bit field description:

* **Way\_Flush\_Control** [31:0] - Way Flush Control

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Way\_Flush\_Control | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Table 33 LLC\_WAY\_FLUSH register

### LLC\_WRITETHROUGH\_EN

This register holds one bit for each of the LLC Allocation Classes. When set to 1, the write accesses that allocate will be write-through. When set to 0, write accesses will be write-back.

Attribute: RW

Security: Non secure

Bit field description:

* **LLC\_WRITETHROUGH\_EN** [7:0] - Write should be write-through

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| u | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| u | | | | | | | | | | | | | | | | | | | | | | | | LLC\_WRITETHROUGH\_EN | | | | | | | |

Table 34 LLC\_WRITETHROUGH\_EN register