Hierarchical Instruction Cache
User Manual

April 2018
IP Version 0.0.1

Igor Loi
igor.loi@unibo.it

Department of Electrical and Information Engineering
University of Bologna, Italy

Document Revisions

Rev.	Date	Author	Description
0.0.1	17/04/16	Igor Loi	First draft of the user manual.

Table of Contents

1 Introduction

The hierarchical instruction cache is an optimized IP used to perform instruction caching in a multi core platform. Code to be executed is normally stored in the main memory (e.g. L2), and this IP is interposed in between processing elements (PEs) and main memory. The cache fetch interface is customized for the RISCV 128 instruction interface, meaning that the the cache delivers data with granularity of a cache line (normally 128 bit). The core itself will use take care to deliver compressed and aligned instruction to the decode stage.

Figure 1 shows a top-level view of the instruction cache within a typical PULP cluster, detailing also the most significant internal modules of the IP.

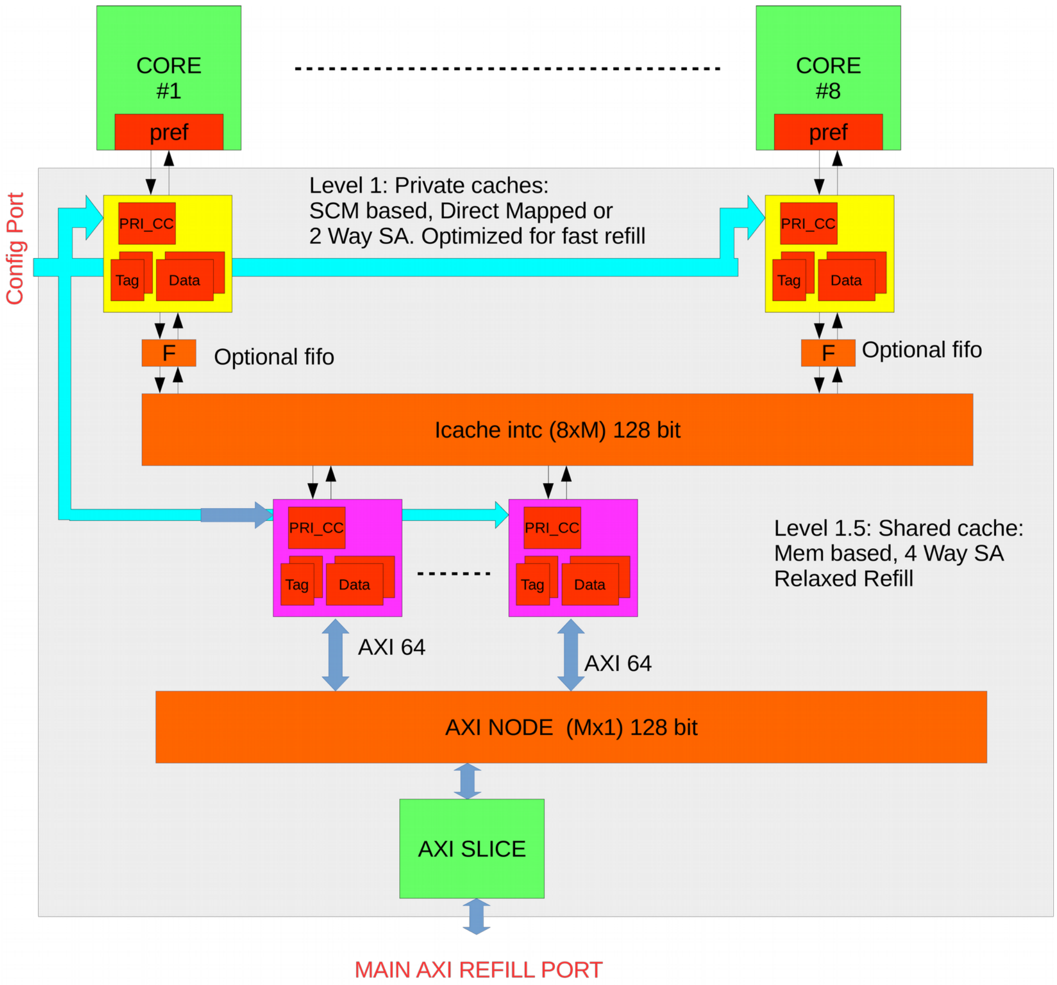

Figure 1: top-level view of the Hierarchical Instruction icache IP

The IP is flexible and offers several degrees of freedom to tune the parameters based on the user needs. The cache is organized in a hierarchical way, and based on two levels. The first L1 level is tightly coupled with the processor and it is private, and normally is tuned to a small capacity (eg half KB), while the second level is shared and tuned to higher capacity (eg 4KB or more). Second level is shared though a lightweight, fast interconnect (icache_intc in the Figure), and this level implemented with multi-banking approach to better spread L1.5 accesses.

The main refill plug of the IP implements the AXI4 protocol, making the IP extremely efficient form a point of view of performance (multiple outstanding refill is supported).

1.1 Scope and Purpose

This Document aims at describing the functional specification of the Hierarchical Instruction Cache. In particulat, it includes:

- Main features of the IP
- IP parameters
- Pinout of the IP
- Application Programming Interface

1.2 Acronyms and Synonyms

PULP: Parallel processing Ultra-Low-Power platform

TCDM: Tightly Coupled Data Memory

PE: Processing Element

SCM: Standard Cell Memories

FSM: Finite State Machine

LFSR: Linear Feedback shift register

1.3 Delivery

Library	Verilog File Name	Description
hier_icache_lib	TOP/icache_hier_top.sv	TOP module that instantiates L1, L1.5 and the interconnects
hier_icache_lib	L1_CACHE/pri_icache.sv	L1 cache top module, that instantiates L1 cache controller, DATA and TAG memories, and Decoupling FIFOs
hier_icache_lib	L1_CACHE/pri_icache_controller.sv	L1 Cache controller, the main engine for the private stage

hier_icache_lib	L1.5_CACHE/icache.sv	L1.5 cache top module, that instantiates L1.5 cache controller, DATA and TAG memories, and AXI plugs
hier_icache_lib	L1.5_CACHE/icache_controller.sv	L1 Cache controller, the main engine for the shared stage
hier_icache_lib	L1.5_CACHE/AXI4_REFILL_Resp_Deserializer.sv	AXI plug that collects refill responses to create an atomic transaction to be committed in the icache DATA memory
hier_icache_lib	L1.5_CACHE/LFSR_L2_Way_Repl.sv	Linear Feedback Shift register (8bit) used to generate a pseudo-random number for replacement policy purpose.
hier_icache_lib	L1.5_CACHE/ram_ws_rs_data_scm.sv	Wrapper for the DATA memory: supports both SCM or SRAM based storage
hier_icache_lib	L1.5_CACHE/ram_ws_rs_tag_scm.sv	Wrapper for the TAG memory: supports both SCM or SRAM based storage
hier_icache_lib	L1.5_CACHE/REP_buffer_4.sv	AXI slice for the Address Read (AR) Channel
hier_icache_lib	L1.5_CACHE/RefillTracker_4.sv	FIFO with CAM capability, to track outstanding transactions, and gather the information needed to commit back refills on the DATA/TAG memory
hier_icache_lib	CTRL_UNIT/hier_icache_ctrl_unit.sv	Instruction cache Control Unit

Table 1: Hierarchical instruction cache delivery

1.4 Hierarchical Instruction Cache Architecture

1.4.1 L1 Stage – L1.5 Stage orchestration

The L1 stage is composed by a private instruction cache, tightly coupled to the processor fetch interface, that is tuned to perform fast HIT access, and with simple and lean interface to reduce as much as possible the timing pressure that is common in low-latency parallel architectures.

The L1 usually is tuned to offer small cache capacity (to reduce overall cache footprint), and for those cache capacity, an approach based on SCM is preferable. The main purpose of the L1 is to act as filter cache, to serve in a efficient way fetch request. If tuned properly, most of the fetches will be cached locally, therefore only a small fraction of the transaction will be redirected to the L1.5 (L1 Miss).

In case of L1 Miss the fetch is processed in the L1.5 cache sub-system, that will respond in one cycle in case of L1.5 Hit.

L1	L1.5	Refill latency
HIT	--	1 cycle
MISS	HIT	2-3-4 cycles (depending on decoupling FIFO options)
MISS	MISS	Refill from L2 latency (round-trip) + 4

Table 2: Refill Latencies

1.4.1 Private icache : L1 Stage

The private stage is basically a set associative read-only cache, composed by a simple cache controller, and TAG/DATA memories. The cache supports only one outstanding transaction (per core), this allowed to simplify both complexity and size (FIFOS and aother tracking elements are not needed) of the L1 stage.

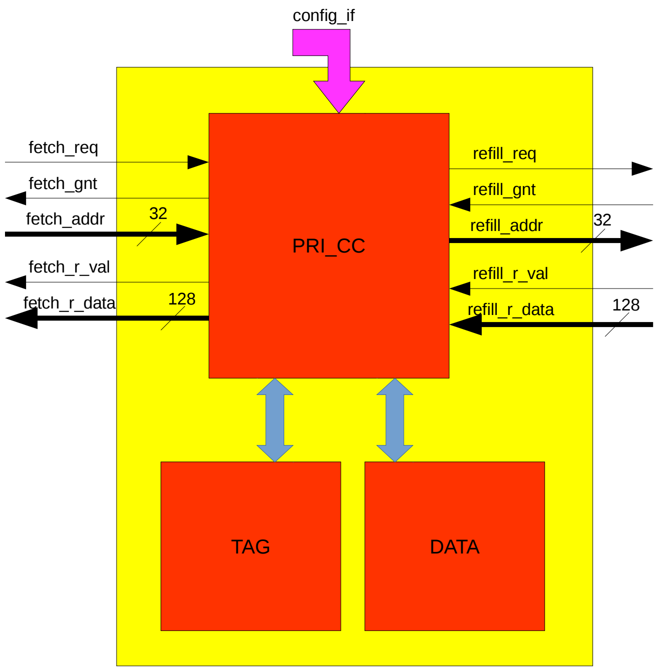

Figure 2: L1 stage overview

The Cache controller is a FSM and though the configuration interface, the cache can be enabled/Disabled, or can issue FLUSH or selective FLUSH request:

STATE	Description
DISABLED	Cache is bypassed, all transaction are fly-through the L1 stage and directed to main memory (L2). No service operations are available (FLUSH). Performance counters are not available when cache is Bypassed
ENABLED	Cache is Enabled, every fetch request is decoded to perform TAG LOOKUP and perform the needed

	operations. FLUSH and SELECTIVE FLUSH are available only when cache is this state.
FLUSH	The cache controllers enters in FLUSH MODE, every cache line is invalidated
FLUSH_ID	The cache invalidates the address specified along the FLUSH_ID request

Table 3: Supported Modes

After reset the Cache controllers enters in BYPASSED mode, and if enable is triggered, the Cache controllers Invalidated all cache lines, and goes in the ENABLED mode. During the Invalidation, the cache is busy and not accessible. To ensure proper functionality, the transition DISABLED → ENABLED and vice versa is performed only when pending transactions are concluded.

1.4.2 Instruction Cache Interconnect

All the Misses generated in the L1 Stage, or fly-though transactions are routed to the L1.5 stage though a read-only MxN interconnect where M stands for the number of processors, and M for the number of BANKS for L1.5 Stage. Requests and responses cross the crossbar within the same cycle they are asserted. For further information, please refer to the LOG interconnect manual.

1.4.3 L1.5 Stage

The Second Stage is decomposed in a Multi-bank fashion (See Figure 1), to better spread accesses from L1. In case two or more L1 misses, the L1 refills are redirect on L1.5 though the interconnect, with potential port collision. In case of Collision, only one is routed, while the rest are stalled. By increasing the Banking factor (Number of L1.5 cache banks vs number of L1 private caches) the probability to have collision is reduced.

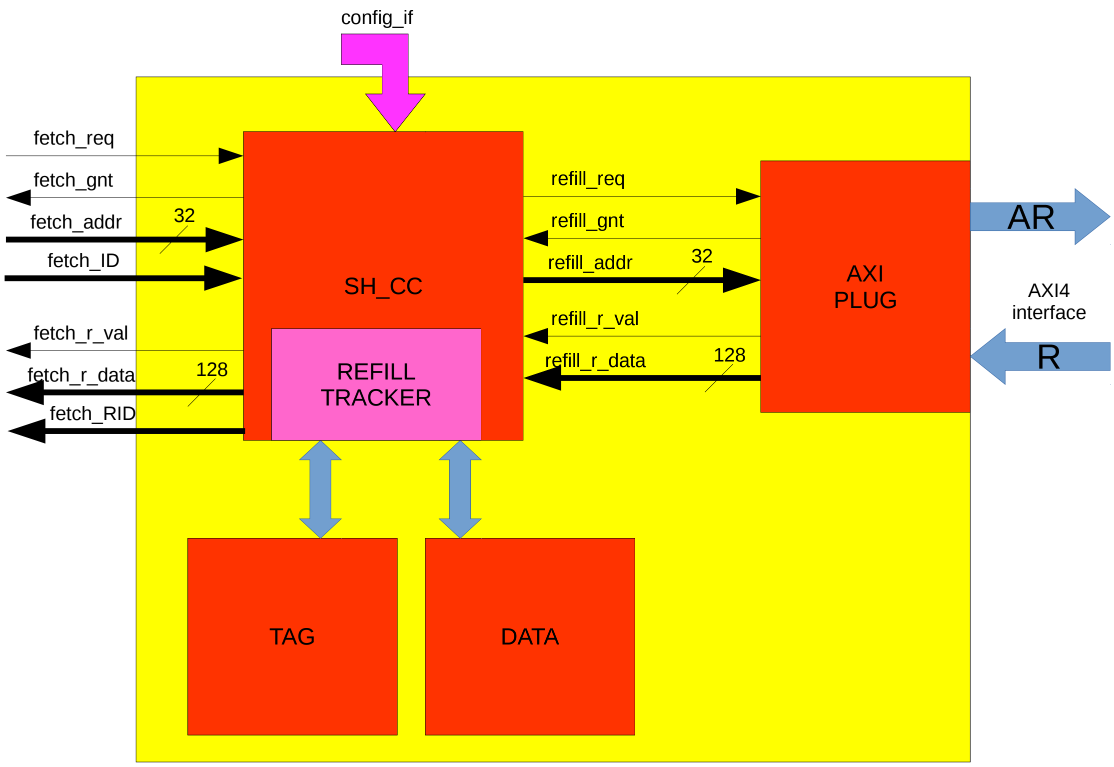

Figure 3: L1.5 Cache Bank overview

The L1.5 Cache Bank is part of the shared instruction cache, derived from the SP (Single Port) variant. The whole L1.5 Cache is composed then from the instruction cache interconnect, the AXI node and the several L1.5 Cache Banks. Those banks are designed to operate with fetch request coming from different L1 caches, therefore it is needed to support multiple outstanding transactions. To track all the pending refills, The Shared Cache Controller (SH_CC) uses a REFILL TRACKER, which is a FIFO where L1.5 misses are pushed, and then using the transaction ID, the information are retrieved , and the SH_CC can commit back the refill and respond properly to the MISS.

Before pushing a REFILL request the SH_CC checks if there is any pending refill at that particular address. To do so, the REFILL TRACKER searches the TAG/SET_ID in this BLOCK, and only in case it is not present, will be pushed in.

In case of double miss on the same address, the SH_CC enters in a state called CRITICAL_REFILL, and stalls any other incoming transaction, until the CRITICAL_REFILL is served.

STATE	Description
DISABLED	Cache is bypassed, all transaction are fly-through the L2 stage and directed to main memory (L2). No service operations are available (FLUSH). Performance counters are not available when cache is Bypassed
ENABLED	Cache is Enabled, every fetch request is decoded to perform TAG LOOKUP and perform the needed operations. FLUSH and SELECTIVE FLUSH are available only when cache is this state.
FLUSH	The cache controllers enters in FLUSH MODE, every cache line is invalidated
FLUSH_ID	The cache invalidates the address specified along the FLUSH_ID request
INVALIDATE	The cache controller iterates on every cache line to set the VALID bit to 0

Table 4: Supported Modes in L1.5

After reset, the SH_CC enters in DISABLED mode and all the traffic is fly-through. User can enable the cache using the configuration interface. Since the L1.5 stage is split in several banks, the procedure to enable/bypass/flush must consider the whole ensemble of cache banks. This is performed by the instruction cache control unit, that in the specific case of Hierarchical Instruction Cache, will orchestrate the different service operations across L1 and L1.5 stage.

The axi plug is in charge to create AXI refill request both in case of miss and in case of fly-through transactions. Being a read only cache, only the AR and R channels are used, why the AW, W and B are simply not used (tied to zero and disconnected). The AXI channel data-with is 64bit so in case of cache line refill or fly-through, 128 bit or multiple , will be fetched from this AXI interface. The aims of this plug is to take care or data-with adaptation, and protocol conversion.

The replacement policy is is pseudo-random. In case all ways are in use, the SH_CC invalidate one random way using a code generated with a LFSR.

1.4.4 Instruction Cache Control Unit (ICCU)

The Hierarchical Instruction Cache is controlled by a peripheral that is memory mapped and allows to ENABLE/DISABLE/FLUSH/INVALIDATE the cache, to clear and enable the performance counters, and to read the number of HIT/MISS/TRANSACTION on L1 and HIT/MISS//TRANSACTION/CONGESTION on L1.5. This peripheral is accessible through a peripheral interface (req-gnt based flow control).

The Service Operations like ENABLE/DISABLE/FLUSH/INVALIDATE are blocking, in the sense that the peripheral will reply with a valid response (r_valid) only at the end of the operation (not deterministic latency). In this time frame the master that made the request is blocked waiting the response. Operation like COUNTER RESET, COUNTER START and COUNTER READ will return a response the cycle after the request is granted.

The peripheral support only ENABLE/DISABLE as monolithic cache, in the sense that legal state are L1+L1.5 all enabled and L1+L1.5 all disabled. Having L1 in a different state than L1.5 is not supported at the moment.

2 Hierarchical Instruction Cache IP Parameters and Top-level Interface

This section describes the parameters of the IP and its top-level interface.

2.1 Parameters

The IP is designed to be configurable a design-time. Table 5 reports a list of the parameters that can be used to tweak the Hierarchical instruction Cache

Parameter Name	Default Value	Description
NB_CORES	8	Number of fetch ports that will be connected to core fetch interface
SH_NB_BANKS	2	Number of shared banks of the L1.5 stage. Must be power of 2 and > 0
SH_NB_WAYS	4	Set Associativity for L1.5. Must be power of and > 0. If set to 1, cache is Direct mapped
SH_CACHE_SIZE	4096	Size in Byte for the L1.5
SH_CACHE_LINE and *PRI_CACHE_LINE*	1	Number of Words per cache Line. A single word is large as the FETCH_DATA_WITH which is set to 128bit.
PRI_NB_WAYS	2	Associativity for L1 (2-4-8-16 supported)
PRI_CACHE_SIZE	512	Size in Byte for L1 Private cache (per core)
AXI_ID	6	WIDTH for the Field AR_ID and R_ID
AXI_ADDR	32	Number of ID bits in the peripheral interconnect port.
AXI_USER	1	If ‘1’, it activates the time-multiplexing scheme that saves ½ of the multipliers with 50% cost in terms of peak theoretical throughput.
AXI_DATA	1	If ‘1’, it enables support for linear convolution mode (bypasses
USE_REDUCED_TAG	TRUE	Will cache only a portion of the memory space (L2_SIZE) instead of 32bit (4GB)
L2_SIZE	512*1024	Size of the L2 used in case the parameter USE_REDUCED_TAG is set to true

Table 5: IP parameters.

2.2 Top level interface

The Hierarchical instruction cache is composed by 3 groups of signal:

1. Fetch interfaces connected to cores

2. Refill interface (AXI port)
3. Configuration interfaces

The Fetch interface is composed by a simple protocol req → gnt, where the core puts the address, the request, and wait for the grant. As soon the grant is sampled high, the core can remove the address, and will expect a valid response in the following cycles.

Name	Width – Direction	Description
fetch_req_i	NB_CORES – INPUT	Fetch request (one bit per core)
fetch_gnt_o	NB_CORES – OUTPUT	Fetch grant (one bit per core)
fetch_addr_i	NB_CORES*FETCH_ADDR_WIDTH - INPUT	Fetch address (FETCH_ADDR_WIFTH bit per core)
fetch_r_valid_o	NB_CORES – OUTPUT	Fetch response valid (one bit per core)
fetch_r_data_o	NB_CORES*FETCH_DATA_WIDTH – OUTPUT	Fetch response data (usually 128 bit per core)

The refill interface is based on AXI4. The AR channel is 64 bit wide and it is used to make refill request, while the R channel is 64 bit wide is will receive the data read from L2 memory.

The configuration interface is adopting the req-gnt protocol for peripheral interconnect:

Name	Width – Direction	Description
speriph_slave_req_i	1 – INPUT	Config request
speriph_slave_gnt_o	1 – OUTPUT	Config grant
speriph_slave_addr_i	32 - INPUT	Config address
speriph_slave_wen_i	1 – OUTPUT	Config Write enable active low
speriph_slave_wdata_i	32 – INPUT	Config Write Data
speriph_slave_be_i	4 – INPUT	Config Byte enable

speriph_slave_id_i	ID_WIDTH - INPUT	Config ID Width (depends on the PERIPH interco).
speriph_slave_r_valid_o	1 – OUTPUT	Config Response valid
speriph_slave_r_opc_o	1 – OUTPUT	Config Response Error
speriph_slave_r_id_o	ID_WIDTH – OUTPUT	Config Response ID
speriph_slave_r_rdata_o	32 – OUTPUT	Config Response DATA

3 Configuration register file map

The configuration port is visible in the cluster memory map. In the latest version of PULP at the time of this writing, the ICCU registers are mapped in the space from address 0x10201400 to 0x102017FC;

3.1 Configuration registers

The Table below reports the available registers

Reg. Offset	Name	R/W	Description
0x00	*ENABLE/DISABLE*	W/O	Enable by writing 0xFFFFFFFF, disable the cache by writing 0X00000000
0x04	*FLUSH*	W/O	FLUSH L1+L1.5 cache by writing 0xFFFFFFFF
0x08	*FLUSH_L1_ONLY*	W/O	FLUSH L1 cache only by writing 0xFFFFFFFF
0x0C	*SEL_FLUSH_CACHE*	W/O	
0x10	*RESET_COUNTERS*	W/O	Reset the performance counters by writing 0xFFFFFFFF
0x14	*START/STOP COUNTERS*	W/O	If write 0 stop the counters, if write 0xFFFFFFFF starts the counters
0x18-0x2C	-	-	RESERVED
0x30	*HIT L1 CORE 0*	R/O	
0x34	*TRANS L1 CORE 0*	R/O	
0x38	*MISS L1 CORE 0*	R/O	
0x3C	*CONG L1 CORE 0*	R/O	
0x40	*HIT L1 CORE 1*	R/O	
0x44	*TRANS L1 CORE 1*	R/O	
0x48	*MISS L1 CORE 1*	R/O	
0x4C	*CONG L1 CORE 1*	R/O	

Reg. Offset	Name	R/W	Description
0x50	*HIT L1 CORE 2*	R/O	
0x54	*TRANS L1 CORE 2*	R/O	
0x58	*MISS L1 CORE 2*	R/O	
0x5C	*CONG L1 CORE 2*	R/O	
0x60	*HIT L1 CORE 3*	R/O	
0x64	*TRANS L1 CORE 3*	R/O	
0x68	*MISS L1 CORE 3*	R/O	
0x6C	*CONG L1 CORE 3*	R/O	
0x70	*HIT L1 CORE 4*	R/O	
0x74	*TRANS L1 CORE 4*	R/O	
0x78	*MISS L1 CORE 4*	R/O	
0x7C	*CONG L1 CORE 4*	R/O	
0x80	*HIT L1 CORE 5*	R/O	
0x84	*TRANS L1 CORE 5*	R/O	
0x88	*MISS L1 CORE 5*	R/O	
0x8C	*CONG L1 CORE 5*	R/O	
0x90	*HIT L1 CORE 6*	R/O	
0x94	*TRANS L1 CORE 6*	R/O	
0x98	*MISS L1 CORE 6*	R/O	
0x9C	*CONG L1 CORE 6*	R/O	
0xA0	*HIT L1 CORE 7*	R/O	
0xA4	*TRANS L1 CORE 7*	R/O	
0xA8	*MISS L1 CORE 7*	R/O	
0xAC	*CONG L1 CORE 7*	R/O	
0xB0	*HIT L1.5 BANK 0*	R/O	
0xB4	*TRANS L1.5 BANK 0*	R/O	
0xB8	*MISS L1.5 BANK 0*	R/O	
0xBC	*HIT L1.5 BANK 1*	R/O	Available if BANK > 1, otherwise will return 0xBADACCE5
0xC0	*TRANS L1.5 BANK 1*	R/O	Available if BANK > 1, otherwise will return 0xBADACCE5
0xC4	*MISS L1.5 BANK 1*	R/O	Available if BANK > 1, otherwise will return 0xBADACCE5
..			

Reg. Offset	Name	R/W	Description
0x104	*HIT L1.5 BANK 7*	R/O	Available if BANK > 4, otherwise will return 0xBADACCE5
0x108	*TRANS L1.5 BANK 7*	R/O	Available if BANK > 4, otherwise will return 0xBADACCE5
0x10C	*MISS L1.5 BANK 7*	R/O	Available if BANK > 4, otherwise will return 0xBADACCE5

Table 1: Control registers and generic configuration registers.

3.1.1 ENABLE / DISABLE register

Write-only register; once written the ICCU will perform global icache enable/disable. If the bit is 1 the enable command is propagated to the right L1.5Bank or L1 private cache bank.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not Used															
R/O															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Enable for L1.5 Bank 7 → Bank0								**Enable for L1 Core 7 → Core 0**							
W/O								W/O							

Bit #	R/W	Description
31:0	R/O	Not used
15:8	W/O	Enable L1.5 icache for BANK 7 to BANK 0 (BANK 0 is LSB)
7:0	W/O	Enable L1 icache for CORE 7 to CORE 0 (Core 0 is LSB)

Table 2: ENABLE / DISABLE register bit fields.

3.1.2 FLUSH register

Write-only register; once written the ICCU will perform global icache FLUSH operation. If the bit is 1 the flush command is propagated to the right L1.5Bank or L1 private cache bank.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not Used															
R/O															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLUSH for L1.5 Bank 7 → Bank0								**FLUSH for L1 Core 7 → Core 0**							
W/O								W/O							

Bit #	R/W	Description
31:0	R/O	Not used
15:8	W/O	FLUSH L1.5 icache for BANK 7 to BANK 0 (BANK 0 is LSB)
7:0	W/O	FLUSH L1 icache for CORE 7 to CORE 0 (Core 0 is LSB)

Table 3: FLUSH register bit fields.

3.1.3 FLUSH L1 only

Write-only register; once written the ICCU will perform L1 icache FLUSH. If the bit is 1 the flush command is propagated to the right L1 private cache bank.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not Used															
R/O															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not Used								**Enable for L1 Core 7 → Core 0**							
R/O								W/O							

Bit #	R/W	Description
31:8	R/O	Not used
7:0	W/O	FLUSH L1 icache for CORE 7 to CORE 0 (Core 0 is LSB)

Table 4: FLUSH L1 only register bit fields.

3.1.4 Selective FLUSH

Write-only register; once written the ICCU will invalidate the address specified within the daa field L1 icache FLUSH

31	30	29	28	27	26	25	24	23	..	..	..	..	2	1	0
Address to be flushed															
W/O															

Bit #	R/W	Description
31:0	W/O	Address to be FLushed

Table 5: Selective FLUSH only register bit fields.

3.1.5 CLEAR Counters

Write-only register; once written the ICCU will perform clear all performance counters. If the bit is 1 the reset command is propagated to the right L1.5Bank or L1 private cache performance counter register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not Used															
R/O															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Clear for L1.5 Bank 7 → Bank 0								**Clear for L1 CORE 7 → CORE0**							
R/O								W/O							

Bit #	R/W	Description
31:8	R/O	Not used
15:8	W/O	Clear performance counters for bank 7 to bank 0 (Core 0 is LSB)
7:0	W/O	Clear performance counters for CORE 7 to CORE 0 (Core 0 is LSB)

Table 6: Clear register bit fields.

3.1.6 Start/Stop Counters

Write-only register; once written the ICCU will perform clear all performance counters.If the bit is 1 the Start command is propagated to the right L1.5Bank or L1 private cache performance counter register. If bit is 0 the register are not updated (frozen).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Not Used															
R/O															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Clear for L1.5 Bank 7 → Bank 0								**Clear for L1 CORE 7 → CORE0**							
R/O								W/O							

Bit #	R/W	Description
31:8	R/O	Not used
15:8	W/O	Clear performance coutners for bank 7 to bank 0 (Core 0 is LSB)
7:0	W/O	Clear performance coutners for CORE 7 to CORE 0 (Core 0 is LSB)

Table 7: Clear register bit fields.