Author: Marek Bykowski at
marekx.bykowski@intel.com

Addressing the L3 cache LockDown issue around CCN-512

Table of contents

1 Goal

ARM bug in XLF ASIC A0 Last Level Cache (LLC) is shown yellow below:

ASIC A0

Total cache/total ways	Number of locked ways	Total locked region size	Locked ways (which ones per partition)	Number of ways per region	To be fixed through RTL
24MB/12	1	2MB	0	1	NO
24MB/12	2	4MB	0,1	1,1	NO
24MB/12	4	8MB	0,1,2,3	1,1,1,1	YES
24MB/12	8	16MB	0,1,2,3,4,5,6,7	2,2,2,2	

Frio cannot squeeze 24MB of LLC so instead 16MB LLC gets produced. However a 16MB-cache is a 16-way associative in contrast to a 12-way associative the ASIC features. The different number of ways prohibit the bug re-creation from ASIC to FPGA due to a different ratio of the locked to unlocked ways. To address the limitation ARM suggested that we generate an FPGA with 4 upper ways disabled. LLC resulting from it is shown below:

Frio B0/B1

Total cache/total ways	Number of locked ways	Total locked region size	Locked ways (which ones per partition)	Number of ways per region	To be fixed through RTL
12MB/12	1	1MB	0	1	NO
12MB/12	2	2MB	0,1	1,1	NO
12MB/12	4	4MB	0,1,2,3	1,1,1,1	YES
12MB/12	8	8MB	0,1,2,3,4,5,6,7	2,2,2,2	

The goal of this task is twofold:

1) to show up a 4-way locking behaves abnormally on B0
2) to show the ARM RTL fix addresses the issue of 1) on B1

To achieve that we do the L3 LockDown measurements for all the possible LockDown scenarios:

- No lock
- ~~1/12 way locking~~ – not measured, truly neglected
- 2/12 way locking
- 4/12 way locking
- 8/12 way locking

2 Assignment realization

To test the LLC LockDown we thought of 2x measurement mechanisms, namely Shalley's PMU (Performance Monitor Unit) and LmBench.

Shalley's PMU offers a number of events to be monitored, among them, cache misses and accesses that can be used to calculate the cache miss/hit ratio. However these events comprise all the misses/accesses within each of the HN-Fs including the Locked and Unlocked ways making them not really suitable to measure the cache performance of L3 LockDown as lacking the ability to differentiate.

The method that is left then is LmBench “bw_mem rd”. bw_mem except rd (read) offers copy, write, also LmBench features other benchmarking that maybe considered to use as measuring the latencies. However as “bw_mem rd” helped finding the L3 LockDown bug for ASIC A0 it was selected as a measure for FPGA B0/B1 too.

“bw_mem rd” works by summing up data read from [addr x to addr x+<size>] swapping through the heap of the process with a 32-byte step (it is in opposed to “bw_mem fdr” that walks over every word). First iteration line fills in the caches offering the subsequent iterations reading from the caches. LmBench measures then the time it takes it to walk through the region and computes the <size> of the region in MB by the time it took it as a function of the region <size>. bw_mem is run from a shell script looping through the sizes:

```
sizeList="512 1m 2m 3m 4m  5m  6m  7m  8m  9m  10m 11m 12m 13m 14m 15m 16m”
for size in $sizeList; do
      bw_mem -P 1 $size rd
done
```

A few notes:
L3 (LLC) allocation policy is exclusive for data lines, with the policy changing to inclusive if the data is shared by other RN-Fs. In addition A53 features data side L1 exclusive of L2, with L2 being a victim cache (with exceptions for the inner transient in page attributes or for non-temporal loads/stores that LmBench doesn’t do, not sure of the page attributes though). In theory such caches should cover up to L1 (32KB) + L2 (1M) + L3 (12M) = over 13M, in which over that point the locations should be served from the main memory.

Cortex A-53 features L1 D Prefetch. L1 D Prefetch is an L1 data cache stride prefetch that monitors cache misses at the fixed stride pattern and upon detection prefetches the locations to L1. By reset 2x linefills to consecutive cache lines activate the prefetch. Then the prefetch monitors the caches misses at a stride of 4x cache lines +/- 1 (=256 +/- 64 bytes) and preloads if the condition is met. Its operation may affect the measurements of LmBench therefore we have disabled it. The register configuring and controlling it is CPUACTLR_EL1. As “CPUACTLR_EL1 can be written only when the system is idle. ARM recommends that you write to this register after a powerup reset, before the MMU is enabled, and before any ACE or ACP traffic begins.” the altering of the register happened from within SPL Uboot and Secure Monitor at times cpu0 gets branched to SPL Uboot (MMU isn’t enabled by then) and for cpu1 through N-1 after they got released out of reset which is takes place in EL3, secure state from inside the Trusted Firmware (N is a number of cpus, for Super Frio it is 12, for ASIC it is 32… most likely).

The Super Frio we run the tests on is 12 cpus / 3x ARM Clusters. The 3x RN-Fs of ARM are all added in to the snoop domain in Shelley so that they can and will send and receive the snoops. It should be noted it may affect the measurements as some of the locations can get invalidated as a result of cache line state (MOESI/MESI) coherency protocol operations.

3 Showing up the bug on B0

3.1 System configuration

- L3 is 12MB with 12 ways

A way is 1MB.

From CCN-512 TRM:
"For non-3MB configurations, the L3 and snoop filter are 16-way set-associative. When the CCN-512 is configured with a 3MB partition then the L3 and snoop filter are 12-way set-associative." By disabling 4 upper ways we achieved what is described below.

From ARM for FPGA:
"defparam oly_hnf_nodeXX.L3_NUM_WAYS_PARAM = 16;
to
defparam oly_hnf_nodeXX.L3_NUM_WAYS_PARAM = 12;

This will result in a 16MB*(12/16)=12MB total LLC, and allow to expose the 4-way lock with 12 total ways issue, and then show that the RTL update addresses the issue."

- L1 D Prefetch disabled

- Configuring the way locking is done from within SPL UBoot running EL3 which is only present in secure state. This is due to the requirements imposed from Shelley on writing the way locking registers (hnf_l3_lock_ways for a number of ways to lock, hnf_l3_lock_base0, hnf_l3_lock_base1, hnf_l3_lock_base2, and hnf_l3_lock_base3 are 4x lock base addresses), namely they can only be accessed from within a secure state, L3 must be flushed before writing the registers and no non-configuration accesses to HN-Fs may be in-flight while the write to the registers is occurring. See "ARM® CoreLink™ CCN-512 Cache Coherent Network Technical Reference Manual" for details.

3.2 Images, source code, system/board used

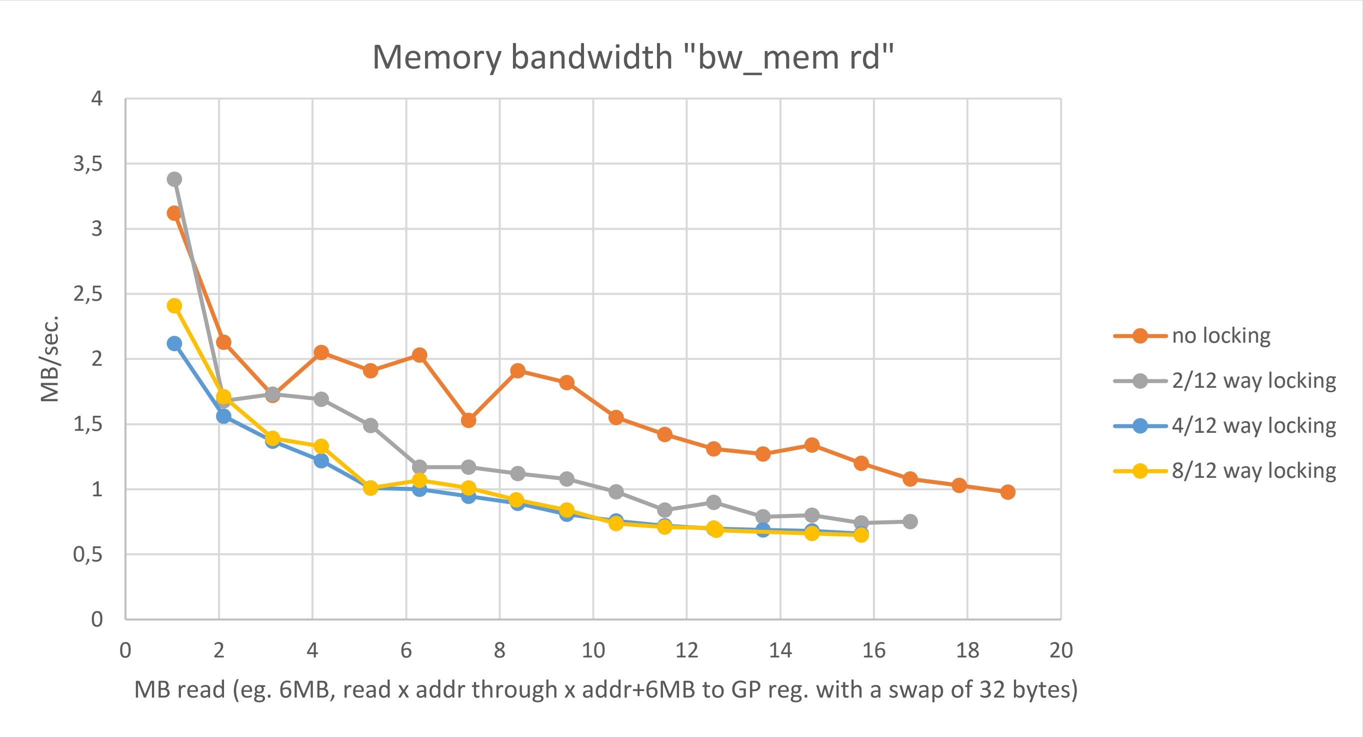
S7A3 Load script: /home/validation/al_common/XLF/util/load_a3_xlf_ph4.3e
Slot 2-14 Load script: "/home/validation/al_common/XLF/util/sa_fpga_load_baseline_xlf_ph4.3h"
FPGA images from FPGA team.

RTE: /home/validation/Lionfish/ncp_rte/ncp_ep8572_1.4.20.3.2
Linux: /tools/AGRreleases/axxia/linux_4.1_axxia_rt_1.71/axxia-arm64-xlf-dev/axc6712-emu
RTE and Linux from HW validation team (forwarded on me from Masoud).

Uboot: github.com/axxia/axxia_u-boot_private.git on l3-lockdown-bug
ATF: github.com/axxia/axxia_atf_private.git on l3-lockdown-bug
Changes in Uboot and ATF to L1 D Prefetch disabling.

System/board used: Super Frio frio-037a-ep2p featuring the 12 ARM cpus.

3.3 Results



3.4 Conclusions

8/12 way-locking seems operating correctly.

2/12 way-locking and 4/12 way-locking **show "abnormality"**. For 4/12 way-locking the explanation of the abnormality was given by ARM for ASIC A0, in AX9Y-184 excerpted below, so that we can infer it is the same cause for Frio B0, that is for:

- 2/12 way-locking only 6 ways remain available to LFSR replacement logic (instead of 10)
- 4/12 way-locking only 4 ways (instead of 8).

"[From Mark Brandyberry - Arm Technical Support]
Please quote reference number TAC710877 when referring to this issue.

Hi Jay,
The design team has found an issue with the CCN-512 way locking feature:

We've replicated the behavior the partner is seeing, and found an issue with the CCN L3 replacement policy when locking 1, 2 or 4 ways. For example, the LFSR replacement logic should chose among the remaining 8 ways when 4 ways are locked, but only chooses between 4.

Note that find-first-invalid is the priority for replacement, so this issue only occurs when all ways are valid, which may decrease the impact of this behavior since the L3 is an exclusive cache.

There is no work-around, if 1, 2, or 4 way locking is needed."

3.5 Issues/questions

- I obtained Uboot and ATF git tag to work with from the HW validation, on top of which I put the necessary changes of mine. The tag is not the latest Uboot neither ATF but close to. Probably I should switch to the latest Uboot and ATF.
- Why don't we use the latest kernel 4.9 for Frio B0?
  - Latest rt kernel 4.9 doesn't boot on Frio B0 but none-rt kernel 4.9 boots fine
- FPGA bring up time is about 1H?
  - udevd population takes the longest, about 15/20minutes
  - it seems shorter on none rt kernel 4.9
- More LmBench tests may be run, eg. bw_mem cp, wr and latencies

Appendix A Results across preempt rt kernel 4.1 and preempt kernel 4.9

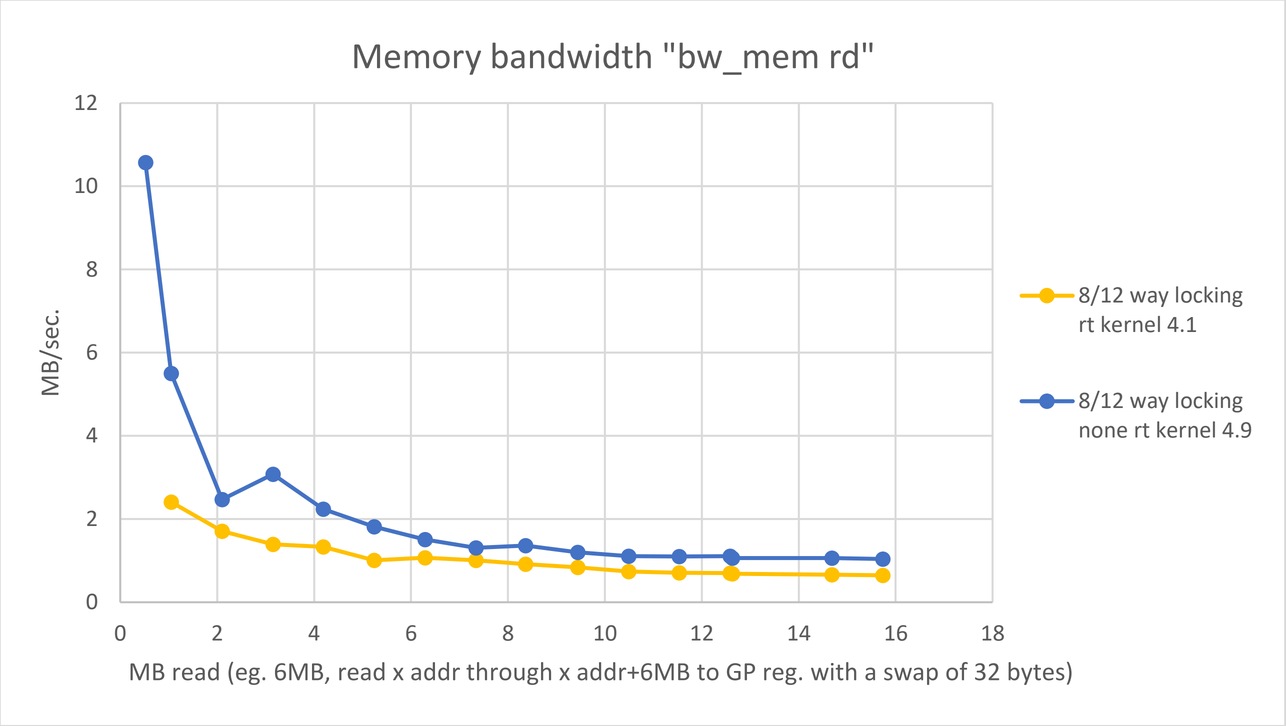
We aimed at updating to the latest, rt kernel 4.9, at the time of this write-up. However it failed booting on B0 but none rt kernel 4.9 has booted successfully:

/tools/AGRreleases/axxia/linux_4.1_axxia_rt_1.71/axxia-arm64-xlf-dev/axc6712-emu

to

Branch: standard/axxia-dev/base, latest at the time of this wrap-up (hash: c03196abf81)
Repo: github.com/axxia/axxia_yocto_linux_4.9_private.git
Kernel config: arch/arm64/configs/axxia_xlf_rt_defconfig
Device-Tree: arch/arm64/boot/dts/intel/axc6712-emu.dts

A note: linux_4.1_axxia_rt_1.71 kernel throws an exception for LmBench whereas none rt kernel 4.9 didn't. The exception is "bw_mem[1890]: unhandled level 2 translation fault (11) at 0x00000000, esr 0x92000006" with a core dump. However everything seemed going fine the return from the exception handler. Most probably there was no region mapped LmBench reached out an address from and it got added in through the handler: exception is taken to EL1, ESR_EL1 (exception syndrome register) is written by HW processor that is then read out in the handler. Upon learning missing pages for the memory the pages are added then in.



Preempt but none rt kernel 4.9 performances better than preempt rt kernel 4.1. Basic differences across is almost everything can be preempted (eg. IRQ handlers, Spinlocks unless raw_) rt kernel, which would suggest there should be more CPU to LmBench and the results should be better (but aren't). On the other hand such a conclusion could be unjustified as we are comparing different kernels.

4 Showing up ARM RTL fix through ECO addresses the issue from B0 on B1

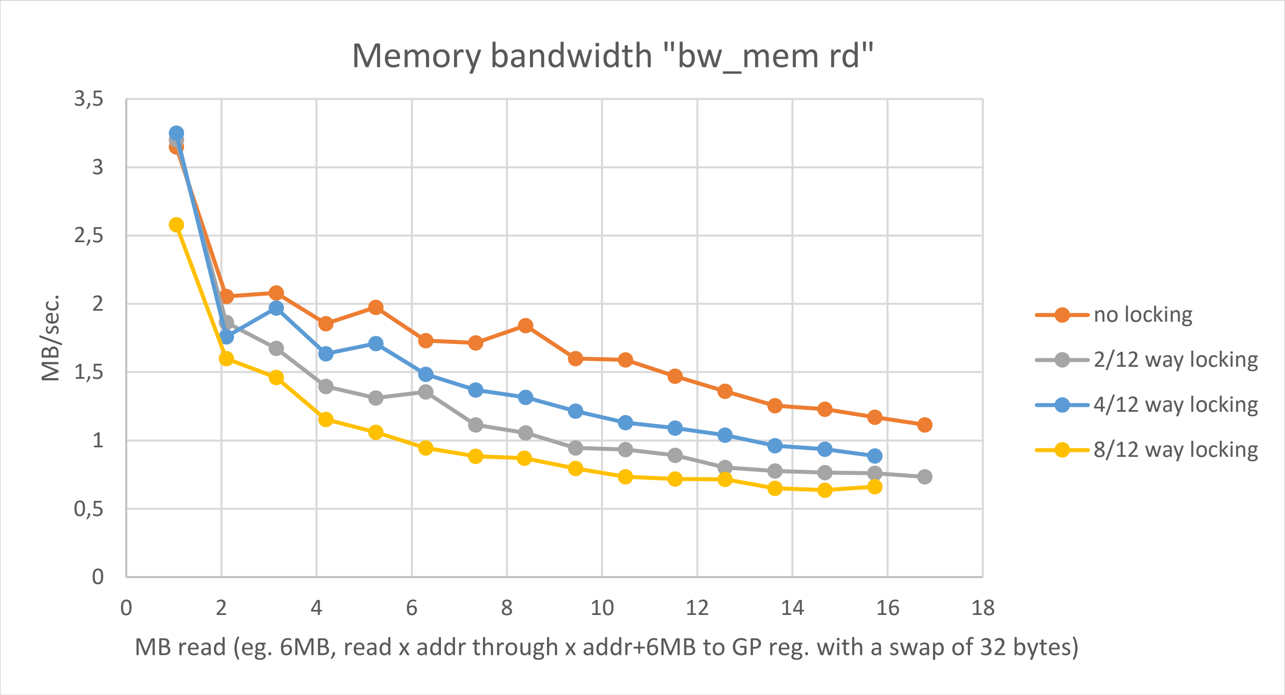
4.1 System configuration

Same as in 3.1

4.2 Images, source code, system/board used

Same as in 3.2 but FPGA images with the Shelley fix in:
S7A3 Load script: /home/validation/al_common/XLF/util/load_a3_xlf_ph5.0
Slot 2-14 Load script: "/home/validation/al_common/XLF/util/sa_fpga_load_baseline_xlf_ph5.0"

4.3 Results

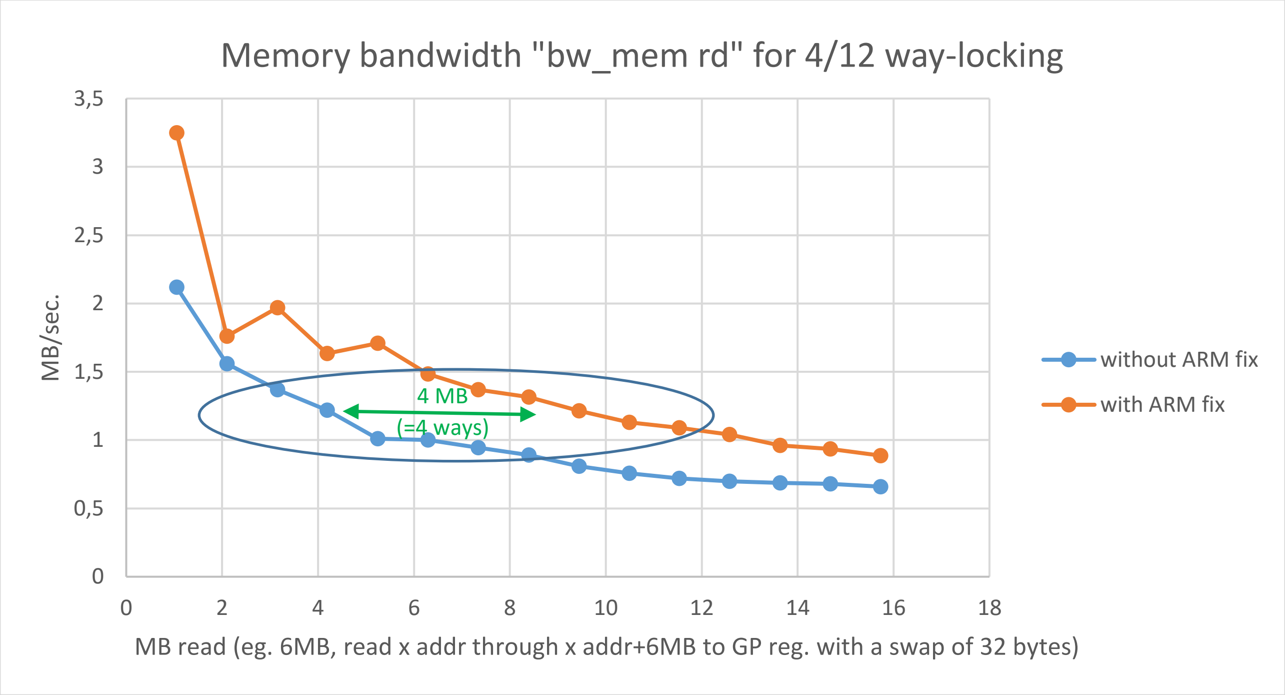


4.4 Conclusions

The ARM RTL update seems addressing the 4/12 way-locking. The bandwidth drops to as if it was fetched from the memory for reads from 8 to 10MB.

4.5 Issues / questions

5 Comparing the 4 way-locking between B0 (without fix) and B1 (with fix)



LLC 4 way-locking without and with the ARM fix is presented in the tables below.

FPGA B0 versus B1 (with reduced ways from 16 to 12, each ways is 1MB)

FPGA	LLC ARM Fix	# ways locked (= MB)	# ways remained to LFSR (=MB)	# ways gone to waste (=MB)
B0	no	4 (=4 MB)	4 (=4 MB)	4 (=4 MB)
B1	**yes**	**4 (=4 MB)**	**8 (=8 MB)**	**0 (=0 MB)**

ASIC B0 versus B1 (12 ways, each way is 2MB)

ASIC	LLC ARM Fix	# ways locked (= MB)	# ways remained to LFSR (=MB)	# ways gone to waste (=MB)
B0	No	4 (=8 MB)	4 (=8 MB)	4 (=8 MB)
B1	**yes**	**4 (=8 MB)**	**8 (=16 MB)**	**0 (=0 MB)**