Application Note 195

ARM11 performance monitor unit

Document number: ARM DAI 195B
Issued: 15th February, 2008

ARM®

Application Note 195
ARM11 performance monitor unit

Release information

Change history

Date	Issue	Change
December, 2007	A	First release
February, 2008	B	Updated branch mis-prediction stall count.

Proprietary notice

Confidentiality status

This document is Open Access. This document has no restriction on distribution.

Feedback on this Application Note

If you have any comments on this Application Note, please send email to errata@arm.com giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

ARM web address

`http://www.arm.com`

Table of Contents

1. Performance Monitor Registers

The ARM11 family of processors contain three 32-bit registers within the system co-processor (CP15), which can be used to obtain performance metrics.

One register is a clock cycle counter (CCNT), which can be enabled to count every cycle, or every 64th cycle. Two registers are event counters (PMNC0, PMNC1), which can be enabled to count events of a specified type, selected from the list given in the Technical Reference Manual tables, or Table 2 below. Enabling these counters does not impact system performance in any way.

The counter registers are programmed using the Performance Monitor Control Register, which

- controls which events PMNC0 and PMNC1 monitor
- detects which counter overflowed
- enables and disables interrupt generation
- controls CCNT counting every cycle or every 64th cycle
- resets all counters to zero
- enables the entire performance monitoring mechanism

An interrupt can be generated from the performance monitor when a counter overflows (nPMUIRQ and, on ARM1156, nPMUFIQ). The interrupt is cleared by writing to the overflow flag for the counter (control register bits 10:8) when bit 0 is set. To make the counter overflow at a specific value e.g. after counting to 1000, the counters PMNC0/PMNC1 can be initialised by writing a value of “232 – value” to them before starting the test. When a counter overflows, it wraps around to zero and continues counting.

When the control register bit 11 (X) is set, events are exported to an external bus (EVNTBUS), which can be monitored by an embedded trace macro cell. Refer to the TRM for EVNTBUS bit mapping to events.

There is an FAQ with supporting software routines for using the PMU at the following url:

http://www.arm.com/support/faqdev/17587.html

2. Performance Monitor Control Register

The purpose of the Performance Monitor Control Register is to control the operation of the cycle counter (CCNT) and the two event counters (PMNC0, PMNC1). It is a 32-bit read/writeable register, accessible in user and privileged modes and is common to ARM1176 secure/non-secure worlds. Commands used to access the register are:

- MRC p15, 0, <Rd>, c15, c12, 0 ; Read Performance Monitor Control Register
- MCR p15, 0, <Rd>, c15, c12, 0 ; Write Performance Monitor Control Register

31	30	29	28	27-20	19-12	11	10	9	8	7	6	5	4	3	2	1	0
SBZ	FCC	FC1	FC0	EvtCount0	EvtCount1	X	OCC	OC1	OC0	SBZ	ECC	EC1	EC0	D	C	P	E

Table 1. Performance monitor control register bit functions

Bits	Field	Function
31	SBZ	Should be zero on write / unpredictable on read
30	FCC / SBZ	Enable CCNT FIQ interrupt reporting (ARM1156 only. On ARM1136/1176, field is SBZ)
29	FC1 / SBZ	Enable PMNC1 FIQ interrupt reporting (ARM1156 only. On ARM1136/1176, field is SBZ)
28	FC0 / SBZ	Enable PMNC0 FIQ interrupt reporting (ARM1156 only. On ARM1136/1176, field is SBZ)
27 – 20	EvtCount0	PMNC0 event type (see table 2)
19 – 12	EvtCount1	PMNC1 event type (see table 2)
11	X	Enable export of the events to event bus for external monitoring (ETM trace).
10	OCC	CCNT overflow flag. Read: 0 = no overflow (reset value) / 1 = overflow has occurred. Write: 0 = no effect / 1 = clear this bit and clear interrupt line
9	OC1	PMNC1 overflow flag. Read: 0 = no overflow (reset value) / 1 = overflow has occurred. Write: 0 = no effect / 1 = clear this bit and clear interrupt line
8	OC0	PMNC0 overflow flag. Read: 0 = no overflow (reset value) / 1 = overflow has occurred. Write: 0 = no effect / 1 = clear this bit and clear interrupt line
7	SBZ	Should be zero on write / unpredictable on read
6	ECC	Enable CCNT interrupt
5	EC1	Enable PMNC1 interrupt
4	EC0	Enable PMNC0 interrupt
3	D	Cycle count divider: 1 = Counts every 64th clock cycle / 0 = Counts every clock cycle.
2	C	CCNT Reset. Read: unpredictable. Write: 0 = no action / 1 = reset counter to zero.

Table 1. Performance monitor control register bit functions (continued)

1	P	PMNC1 and PMNC0 reset. Read: unpredictable. Write: 0 = no action / 1 = reset counter to zero.
0	E	Enable all counters. Must be set to clear interrupt request.

Table 2. Performance Monitor Event Types

EvtCount	Event Definition
0x0	Instruction cache miss. Instruction cache miss to a cacheable location, which requires a fetch from external memory
0x1	Stall because instruction buffer cannot deliver an instruction. This could indicate an Instruction Cache miss or an Instruction MicroTLB miss. This event occurs every cycle in which the condition is present.
0x2	Stall because of a data dependency. This event occurs every cycle in which the condition is present.
0x3	Instruction MicroTLB miss (unused on ARM1156).
0x4	Data MicroTLB miss (unused on ARM1156).
0x5	Branch instruction executed, branch might or might not have changed program flow.
0x6	Branch mis-predicted.
0x7	Instructions executed.
0x9	Data cache access, not including Cache operations. This event occurs for each non-sequential access to a cache line, for cacheable locations.
0xA	Data cache access, not including Cache Operations. This event occurs for each non-sequential access to a cache line, regardless of whether or not the location is cacheable.
0xB	Data cache miss, not including Cache Operations.
0xC	Data cache write-back. This event occurs once for each half line of four words that is written back from the cache.
0xD	Software changed the PC. This event occurs any time the PC is changed by software and there is not a mode change. For example, a MOV instruction with PC as the destination triggers this event. Executing a SWI from User mode does not trigger this event, because it incurs a mode change.
0xF	Main TLB miss (unused on ARM1156).
0x10	Explicit external data or peripheral access. This includes cache refill, non-cacheable and write-through accesses. It does not include write-backs or page table walks.
0x11	Stall because of Load Store Unit request queue being full. This event occurs each clock cycle in which the condition is met. A high incidence of this event indicates the LSU is often waiting for transactions to complete on the external bus.
0x12	The number of times the Write Buffer was drained because of a Data Synchronization Barrier command or Strongly Ordered operation.
0x13	The number of cycles which FIQ interrupts are disabled (ARM1156 only).
0x14	The number of cycles which IRQ interrupts are disabled (ARM1156 only).

Table 2. Performance Monitor Event Types (continued)

0x20	ETMEXTOUT[0] signal was asserted for a cycle.
0x21	ETMEXTOUT[1] signal was asserted for a cycle.
0x22	ETMEXTOUT[0] or ETMEXTOUT[1] was asserted. If both ETMEXTOUT[0] and ETMEXTOUT[1] signals are asserted then the count is incremented by two.
0x23	Procedure call instruction executed. The procedure return address was pushed on to the return stack (ARM1176 only).
0x24	Procedure return instruction executed. The procedure return address was popped off the return stack (ARM1176 only).
0x25	Procedure return instruction executed and return address predicted. The procedure return address was popped off the return stack and the core branched to this address (ARM1176 only).
0x26	Procedure return instruction executed and return address predicted incorrectly. The procedure return address was restored to the return stack following the prediction being identified as incorrect (ARM1176 only).
0x30	Instruction cache Tag or Valid RAM parity error (ARM1156 only).
0x31	Instruction cache RAM parity error (ARM1156 only).
0x32	Data cache Tag or Valid RAM parity error (ARM1156 only).
0x33	Data cache RAM parity error (ARM1156 only).
0x34	ITCM error (ARM1156 only).
0x35	DTCM error (ARM1156 only).
0x36	Procedure return address popped off the return stack (ARM1156 only).
0x37	Procedure return address popped off the return stack has been incorrectly predicted by the PFU (ARM1156 only).
0x38	Data cache Dirty RAM parity error (ARM1156 only).
0xFF	An increment each cycle.
Other values	Reserved. Unpredictable behaviour.

3. Event 0x0: Instruction cache miss

The instruction cache must be enabled for this event to be counted. Each instruction cache miss triggers an instruction fetch, which is a burst of four double words (8 words) to fill a cache line.

If the I-Cache miss count appears to be excessive, the most likely reason is that branch prediction is not enabled, as it is at reset. This results in the branch shadow at the end of each code loop being fetched every time around the loop. If the branch shadow extends into a new cache line, which has not yet been executed, then the cache line will continually be discarded after it is fetched, meaning that a new cache line fill will take place next time around the loop. This behaviour is eliminated with branch prediction enabled, because the processor will not fetch the branch shadow, if it predicts correctly.

If branch prediction is already enabled and the I-Cache miss count is still high, this may be the result of alignment of subroutine return instructions. With branch prediction on, subroutine returns are predicted using the return stack, but in this case the prediction is recognized in the cycle following the request to fetch the branch shadow. This is in time to cancel the requested bus access for a potential cache line fill (assuming the alignment of the code would trigger a line fill on the branch shadow), but the I-Cache miss counter has already been updated by this time - so in this case, the 'miss' count is incremented but no external bus access takes place.

Some instruction sequences will generate multiple I-cache misses in the same cache line, but result in a single cache line fill. Thus the I-Cache miss count in the performance monitor is larger than the number of cache line fills seen on the bus.

4. Events 0x1, 0x2, 0x11: Stall cycles

Event 0x1 is a stall due to the instruction buffer not delivering an instruction. Event 0x2 is a stall due to a data dependency, which implies that the CPU is trying to execute an instruction whose operands have not yet arrived from memory, therefore the stall cycles depend on the arrival of data. Event 0x11 is a stall due to the load-store unit being fully occupied with outstanding transactions, and therefore being unable to accept another memory access request.

Branch mis-prediction, which causes a pipeline flush, will increment the instruction stall counter by one (event 0x1), although it incurs a 4 clock cycle penalty.

Page table walks are not counted as instruction or data stalls (events 0x1, 0x2) – they are counted as main TLB misses (event 0xF).

A data dependency does not cause an instruction stall (event 0x1). A data dependency may occur with or without the LSU queue being full. LSU queue being full may occur with or without a data dependency. Note that it is possible for one cycle to be counted in more than one performance monitor event; therefore the total number of stall cycles is not necessarily the sum of the stall cycles in different classes.

5. Events 0x5, 0x6, 0xD: Branches

Event 0x5 counts branch instruction execution. Event 0x6 counts branch mis-prediction which necessitates a pipeline flush. Event 0xD counts any change to the program counter which does not involve a mode change – this includes taken branches.

After reset, all ARM11 branch prediction features are disabled. The processor assumes a linear program flow with all branches assumed to be not taken (even if they are unconditional). Branches are still counted as mis-predicted under these conditions, when they are taken and change the program flow.

Turning on program flow prediction via the Z bit in the CP15 control register enables all three forms of prediction: static branch prediction, dynamic branch prediction and return stack usage. These three features are individually enabled by the auxiliary control register.

Branch prediction ratio can be measured as 1 – event0x6 / event0x5.

6. Event 0x7: Instructions executed

This is a count of instructions which reach the execute stage of the pipeline. Instructions which are fetched but not executed are not included. Instructions are counted which fail their condition codes (as well as those which pass condition codes).

To build a picture of what the CPU is doing in every cycle, you might expect that you could add instructions executed (event 0x7) to the number of stalls cycles (events 0x1, 0x2, 0x11) to arrive at the total cycle count. But this does not work because:

- Some instructions take multiple CPU cycles to execute and are only counted once by the performance monitor
- Stall cycles can be counted in more than one event category

7. Events 0x9 and 0xA: Cacheable and non-cacheable data-side accesses

Event 0x9 counts non-sequential data cache accesses, to cache lines, for cacheable locations. Event 0xA counts non-sequential data cache accesses, to cache lines, regardless of whether the access is cacheable or non-cacheable. Subtracting the event 0x9 count from the event 0xA count gives the number of non-cacheable accesses.

These events count non-sequential cache line accesses: when an LDM/STM causes multiple accesses within a cache line, it is counted as a single access; however, when an LDM/STM causes accesses across two cache lines, it is counted as two accesses.

A TCM access is treated as a cacheable access and is counted as an event 0x9 and 0xA. Every data side load or store is looked up in the cache or TCM – this is true for any memory type (normal, device, strongly ordered) and for cacheable/non-cacheable memory accesses – so event 0xA gives the total number of accesses (cache hits plus cache misses). Speculative accesses are included in the count, causing a higher count than expected. For example, a level-2 memory write followed by a TCM read is counted as 3 accesses due to speculation.

Page table walks are excluded from the event count since they never generate a cache access. Also, CP15 control co-processor cache operations are excluded from the event count.

8. Estimating cache hit ratio

We do not have counters of pure cache hits and cache misses, so it is not possible to measure the exact cache hit/miss ratio. But estimates can be made using the following event counts.

For the data cache, use event 0x9 for the total data cache accesses, since this count does not include non-cacheable data accesses. Use event 0xB for the number of accesses that miss the cache. Event 0xB, data cache miss, counts all non-CP15 operations which miss in the cache and DTCM and go to the external AHB interface, for both writes and reads. Note, however, that only the initial non-sequential access is counted: multiple reads/writes to the same cache line are counted only as a single access.

For the instruction cache, use event 0x0 for the number of I-Cache misses to cacheable locations. This does not account for any non-cacheable instruction accesses. Again, only non-sequential misses are counted. Be aware that there will be speculative instruction cache lookups (owing to branch prediction/mis-prediction). A cache miss causes a linefill of eight 32-bit instructions. There

is no count of I-cache accesses, so calculating the I-cache hit ratio is not possible. Event 0x7 gives the number of instructions executed.

9. Using level 2 cache controller statistics

The ARM level 2 cache controller (e.g. L210/L220) has its own event monitor bus. The L210 has the capability to count the following events (as well as some others):

- CPU instruction read request to L210 cache
- CPU instruction read hit in L210 cache
- CPU data read request to L210 cache
- CPU data read hit in L210 cache
- CPU data write request to L210 cache, not write-through
- CPU data write request to L210 cache, write-through
- CPU data write hit in L210 cache

It is best to regard these counts as different from ARM11 PMU counts, since there is not a 1:1 correlation between counters.

The number of L1 instruction cache misses (event 0x0) is greater than the number of L2 instruction read requests. Instruction cache misses may be counted without any L2 fetch taking place (refer to section 3 above). Two instruction cache misses (in the same cache line) may lead to a single cache line fill. The L2CC slave buffers also affect the figures. When a L2CC slave port receives a read request, it copies a cache line into a small (32 byte) local buffer. Having this buffer in the slave interface speeds up subsequent accesses to the same cache line and reduces contention for accessing the L2CC memory blocks. When the slave port receives a read access, which can be served from this small buffer, then the L2 event monitor does not count the read request. This means that multiple accesses to the same cache line may incur only a single L2 read request event.