

## FEATURE ARTICLE

# Monitoring Your Micro

Daniel Mann



If you're going to keep your system running at its peak performance, it helps to be able to monitor the microprocessor. Most processors use counters to measure performance, but Daniel and Jim have a better way of gathering and analyzing performance data.

Most desktop processors, such as the K6 and Pentium, are equipped with performance monitoring counters. These counters permit processor performance parameters to be monitored and measured. Such information is useful for performance tuning. Current techniques typically use two counters that simultaneously record the occurrence of pre-specified events. When one of the counters overflows, counting stops and an interrupt is generated. Post-processing software is used to analyze the gathered data.

In this article, we'll describe a new technique for gathering and analyzing performance data with a microprocessor or microcontroller. The technique avoids the limitations imposed by fixed-size counters, which eventually overflow. This method is less intrusive and suitable for monitoring a wide range of performance parameters.

Also, a performance improving DRAM interface buffer is described, as well as the monitoring technique used to examine and tune its operation.

## PERFORMANCE MONITORING COUNTERS

Two large counters, about 40 bits or more, are usually provided for event counting. These counters can be read and written from within the register address space. The counters can be configured to measure such parameters as the number of data reads that hit in the cache. In this case, the second counter can be programmed to record the number of data reads performed. The ratio of these two numbers gives the cache hit rate for reads.

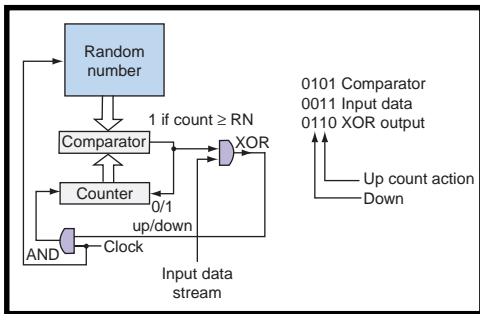
When one of the counters reaches its limit, the overflow signal can be used to stop all counting and generate an interrupt. The software interrupt handler then records the counter values and completes post data processing and any other support work necessary.

The size of the counters is important. The larger the counter, the less frequently an interrupt is generated. Interrupts are undesirable because they intrude on normal processor operation. A larger counter also results in greater data averaging. Any temporary fluctuation in the cache hit rate is not observed. This may or may not be what is required.

Before performance monitoring can be accomplished, an interrupt handler must be installed to deal with counter overflow. Of course, overflow can be avoided by the use of extremely large counters. But, such a technique may be expensive to implement, unreliable, or fail to produce the desired statistical analysis.

## DATA GATHERING

The type of performance data that is being measured is random in nature, such as the cache hit rate or the number of cycles to read memory. These parameters vary during program execution. Measuring random



**Figure 1**—Here you can see a possible Addie configuration where a counter is compared with a random number.

data enables a precise value to be generated. These precise values take the form of averages, such as the average cache hit rate.

Measured performance parameters are a good estimate of future performance. Actual performance at any instant may vary widely from the measured estimate. The typical use of two large counters does not make any attempt to measure this deviation.

Consider, for example, that an on-chip cache may successfully provide the required data at each memory access. The sequence of hit-and-miss data can be represented by a simple one or zero bitstream. The probability of a one is the same as the probability of a hit occurrence. A stochastic Addie can be used to integrate the probability stream and determine the relevant probability.

Figure 1 shows a possible Addie configuration. A counter is compared with a random number. If the counter is greater than the random number, a one is generated. Large counter values are more likely to produce an output of one from the comparator than smaller counter values.

The counter output is compared with the 1/0 datastream of interest—the cache hit information. These two stochastic datastreams are compared to see which one has the highest probability of being one. This sounds more difficult than it is, only an XOR gate is required. When the datastreams differ, there is a difference in probability. This information is fed back to increase or decrease the counter value. Consequently, with each new comparison, the counter is

adjusted to produce a probability stream (from the comparator), which matches the input datastream.

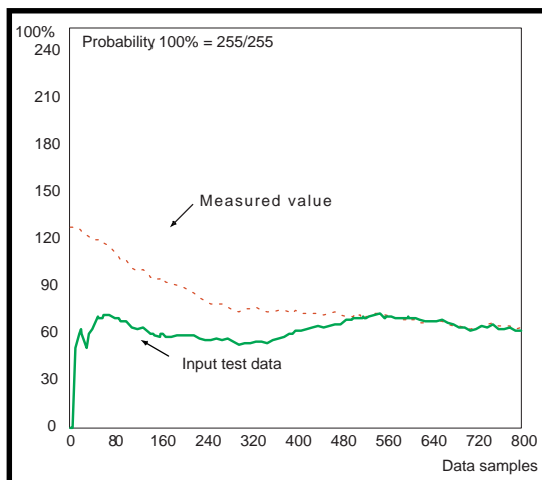
The Addie device effectively integrates the probability stream. Hence, converting the probability stream into a digital value (held in the counter) represents the probability of the parameter being measured.

This method of measuring probability is different from the dual counter method. There is no potential overflow. There is no need for an overflow interrupt handler. And, the counter can be read at any time to give a measure of the current probability.

### AN 8-BIT ADDIE

As the number of bits used by the counter and random number generator increase, the probability resolution improves. For example, an 8-bit counter provides a probability resolution of 0.39% (1/255). However, increasing the resolution slows down the integration process. This results in a greater number of samples required before a good estimate of probability can be obtained. Figure 2 shows an 8-bit Addie used to measure a synthetic data stream with a 25% probability.

As you can see, the Addie starts with an initial value of 50%. As the input data is sampled, the Addie's counter value heads towards the expected value. After about 500



**Figure 2**—An 8-bit Addie is used to measure a synthetic datastream with a 25% (64/255) probability.

samples, the Addie closely tracks the input datastream. This indicates that, after as little as 500 samples, the counter could be read to produce an estimate of the performance parameter being measured.

During the generation of the presented data, both the input and Addie datastreams are averaged over 250 samples. Keeping this window small helps to detect temporary fluctuations in both the generated test data and Addie response.

Figure 3 shows the 8-bit Addie used to measure a 78% probability. The 8-bit counter was initially at a value of 128, which represents a probability of 50%. Once again, after about 500 samples the measuring instrument converges on the desired value.

### RANDOM NUMBER GENERATION

The Addie operation requires a random number generator (see Figure 4). A pseudo-random number generator is ideal for this task. A maximum length (m-sequence) can be produced by feeding back selected stages of an n-stage shift register. The required stages are modulo-2 combined and used to produce the input signal for the first stage.

The Addie can be encouraged to converge on the required value by using random numbers that correlate. This can be achieved by selecting consecutive stages of the shift register and inverting the most-significant bit (MSB). Hence, the MSB of the current random number will be inverted, becoming the next-to MSB of the next random number to follow.

The test data shown used a 31-bit shift register with feedback taken from stages 3 and 31. The top 8 or 12 bits were used to form the required random number. A smaller (less than 31 bits) shift register can also be used.

### DRAM MEMORY INTERFACE BUFFER

Microcontrollers often incorporate memory interface circuitry, which eliminates the glue logic required with most microprocessor-based systems. AMD's

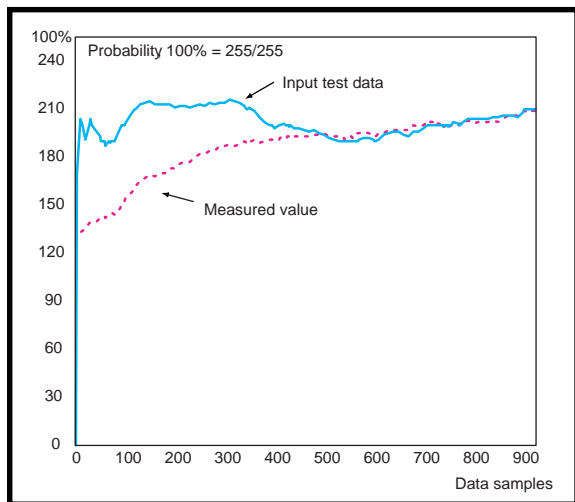


Figure 3—Here the 8-bit Addie is used to measure a 78% (200/255) probability.

Embedded Processor Division is currently developing an advanced memory interface technology. The interface will include two buffering techniques (the write and read buffer) that optimize the effects of DRAM performance on overall system performance.

The write buffer provides a mechanism for writes to DRAM achieved with a zero memory wait-state. For those of you who are unfamiliar with a memory wait-state, it is the number of additional clock cycles required to access a memory resource. Fast SRAM has fewer wait-states than slower DRAM.

The write buffer effectively decouples write activity from incurring the normal DRAM latency penalty. This, in effect, also allows the DRAM to remain free to satisfy a higher demand, such as a data read request. In addition, the write buffer provides write merge and write collapse functions. These techniques enable better usage of the buffer's limited storage and reduce the total number of transactions to DRAM. Data read merging is additionally supported to maintain data coherency.

The read buffer provides storage for eight words (32 bits) read from DRAM. The memory is divided into four word blocks known as cache lines. When the processor requires a value currently held in memory, the whole cache line (or block of memory) is read into the processor. Consequently, the read buffer is better de-

scribed as consisting of two cache lines. The read buffer supports an optional read-ahead function, which ensures the prefetching of the cache line, where the address immediately follows the requested cache line. This feature is provided in anticipation of future accesses to the pre-etched memory block (spatial locality).

The performance of the memory interface buffer is monitored by two Addie units. Through a multiplexer, each Addie can be configured to monitor a range of different performance parameters. The following sections describe some of these parameters and show how the performance data obtained can be used to further optimize and analyze system performance.

## PERFORMANCE-ENHANCING FEATURES

The write buffer can hold up to 32 words (4 bytes per word) of data waiting to be sent to the system DRAM. Each write request results in a snoop of the write buffer contents. The snoop is used to determine if the data associated with the write request already exists in the buffer. The term "associated" refers to the address of the new data, matching up to an address tagged to data waiting for transfer to DRAM. This feature allows write data to be merged (or collapsed) with data that already exists in the write buffer.

Merging data writes results in a reduced number of overall writes to DRAM. It also results in more efficient use of the write buffer storage.

The write buffer supports read merging. Read merging occurs when a read request "hits" (is associated with) a word that currently exists in the write buffer. In such a case, the data returned from DRAM is replaced, or merged, with existing bytes from the

write buffer (see Figure 5).

Without the snooping capability and read merge logic, the entire contents of the write buffer would have to be flushed prior to any read cycle. This would be required to prevent reading old data from DRAM that is about to be updated by data held in the write buffer. Snooping results in less overhead, while maintaining data coherency. With some write buffers, snooping is merely used to determine if flushing is required. A performance advantage is achieved by activating read merge logic when snooping detects a write buffer hit.

The write buffer also supports a read-around-write feature. This allows read requests to DRAM to occur ahead of (or around) write requests waiting in the write buffer. Allowing read requests to be processed without additional delays reduces processor stalling and, in turn, improves system performance.

## WRITE BUFFER "HIT" MONITORING

The write buffer supports a watermark limit setting. The write buffer accumulates data waiting to be transferred to DRAM until the watermark limit is reached. At that point, priority is given to completing data transfer from the write buffer to DRAM.

A higher watermark setting allows more write data to accumulate in the write buffer before write-backs are initiated to DRAM. This provides a greater chance for data merging and collapsing to take place and lessens the occurrence of writes from the write buffer delaying read accesses.

A lower watermark setting causes the write buffer to request DRAM accesses with fewer data in the buffer.

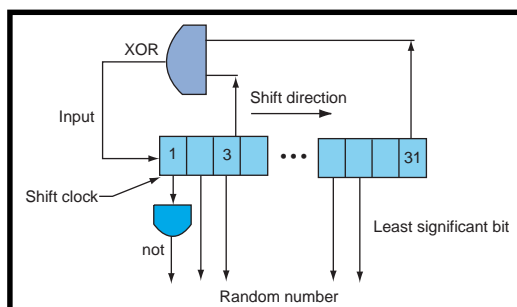
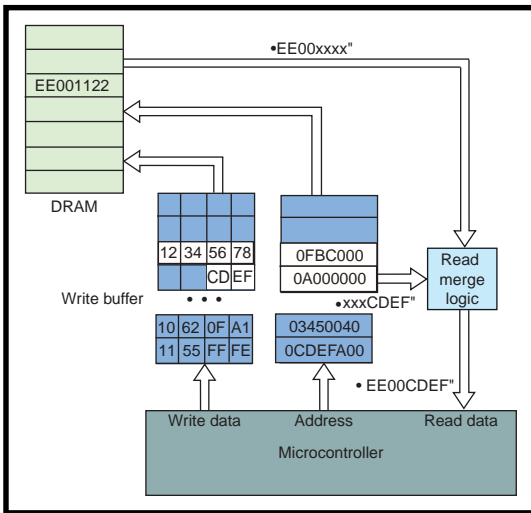


Figure 4—A random number generator is required for the Addie operation.



**Figure 5**—Here's a memory interface buffer. The data returned from DRAM is replaced or merged with existing bytes from the write buffer.

This results in a greater likelihood of the write buffer competing with read accesses to the DRAM and lessens the occurrence of write buffer overflow. This results in the write buffer gaining highest priority access to the DRAM.

A low hit average, even with a high watermark setting, may imply that most write data is simply queuing in the write buffer, and merging and collapsing is not taking place often. If this is the case, a lower watermark setting is advisable to prevent full write buffer occurrence. A higher hit average with a low watermark setting may imply that there is a high occurrence of write data merging or collapsing. If this is the case, a higher watermark setting is advisable to allow the write buffer to absorb the write activity, without adversely affecting read accesses.

An Addie performance monitor may be used to provide a write buffer hit measure. This information can be used to specify an optimal watermark setting for the write buffer.

### WRITE BUFFER "READ MERGE" MONITORING

A performance monitoring Addie can be configured to provide a "read merge" measure. A read merge implies that at least one byte of a read request was provided by a write buffer entry. A ratio of write buffer read merge to no-read merge can be provided by the Addie. This gives a mea-

sure of reads occurring ahead of writes still pending in the write buffer.

### WRITE BUFFER FULL MONITORING

When the write buffer is not full, write data is posted with zero wait states. However, occasionally data may be posted to the write buffer faster than data is written to DRAM. This may result in a full write buffer, preventing data from being posted until it is written out of the write buffer to DRAM.

The write buffer's watermark setting will play a role in how often the write buffer becomes full. A higher watermark setting causes data to sit in the write buffer longer, possibly allowing write data merging or collapsing to occur. A lower watermark setting requests DRAM service when less data is waiting in the write buffer.

A performance monitor may be configured to provide a write buffer full measure. This information could be used to select the watermark setting that produces the best system performance.

### READ BUFFER "HIT" MONITORING

An Addie performance monitor can be configured to provide a read buffer "hit" measurement. A read buffer hit implies that at least one byte of a read request was satisfied from read buffer data.

The read buffer always stores an entire cache line of data read from DRAM, independent of the number of words requested during the read request. A subsequent read request that hits (is satisfied by) the read-buffer is scored as a single hit by the performance monitor.

The Addie monitors read buffer hits on the basis of an atomic read request during the same bus tenure, independent of burst length. Therefore, each read request is monitored, rather than each 32-bit word transferred. It is done in this way, rather than unfairly scoring four read buffer hits during a burst of four 32-bit

words. Such a burst request would naturally fetch an entire cache line. However, four independent read requests will result in four independent read buffer hits by the performance monitor because each read transfer was an individual read request.

The read buffer supports an optional read-ahead feature. However, like all buffering techniques that support speculative prefetch, the anticipated prefetched line may not be used, possibly resulting in overhead associated with the unused prefetch.

The Addie provides a measure of read buffer hit success. This information can be used to provide feedback as to whether or not the read prefetch feature of the read buffer should be enabled. A high read buffer hit average would imply that the read prefetch data has a high occurrence of being used. A low read buffer hit average would imply that the read prefetched data is not being used, thus disabling the read prefetch feature.

### BETTER THAN COUNTERS

This discussion does not exhaustively describe the performance monitoring capabilities of the Addie methodology. However, it is clear that the Addie device is well-suited to carry out continuous performance measurement when incorporated into a microprocessor or microcontroller. The operation of advanced write buffer technology is not normally easy to observe. The Addie enables its operation to be tuned for best system performance.

AMD is working on new debug-port technology designed to support all stages of embedded software development. This technology, known as AMDebug, incorporates a JTAG communication port for controlling and monitoring processor operation. Through the AMDebug port, Addie operation can be administered and performance measurements can be obtained.

Using the Addie, there is no need to instrument the target system's application or operating system software (i.e., no interrupt handler is required). The method is superior to the traditional approach of using mul-

tiple counters. An 8-bit Addie requires about 500 or more samples gathered before it can track the desired input datastream. The technology of the Addie will likely lead engineers to a greater understanding of how their microprocessor is performing and where system bottlenecks exist. 📌

*Daniel Mann is on the Technical Staff of AMD's Embedded Processor Division. He has worked extensively on developing on-chip technology supporting microprocessor software development and performance measurement. He is currently working on Box development. You can contact him at [daniel.mann@amd.com](mailto:daniel.mann@amd.com).*

*Jim Margo is on the Technical Staff of AMD's Microprocessor Products Division. He has worked on various DRAM controller designs, which include the AMD Elan SC520 SDRAM controller, read buffer, and write buffer. He is currently developing the high-performance Double Data Rate (DDR) DRAM controller in the AMD-761 chipset for the AMD Athlon processor. You can contact him at [jim.margo@amd.com](mailto:jim.margo@amd.com).*

## REFERENCES

[1] A. J. Miller, A.W. Brown, P. Mars, *A Study of an Output Interface for a Digital Stochastic Computers*, Int. J. Electronics, 1974, Vol. 37, No. 5, 637–655.

Circuit Cellar, the Magazine for Computer Applications.  
Reprinted by permission. For subscription information,  
call (860) 875-2199, [subscribe@circuitcellar.com](mailto:subscribe@circuitcellar.com) or  
[www.circuitcellar.com/subscribe.htm](http://www.circuitcellar.com/subscribe.htm).