SDRAM Performance Monitors with the Élan™SC520 Microcontroller

AMD

Application Note

by Daniel Mann and James Magro

This application note describes the concept, design considerations, functions, and use of the SDRAM performance monitors with the Élan™SC520 microcontroller.

OVERVIEW

The Élan™SC520 microcontroller includes two non-intrusive Adaptive Digital Element (ADDIE) performance monitor resources for monitoring a variety of SDRAM controller parameters. Features of the performance monitors include:

- Non-intrusive (hardware)
- Probability resolution of 0.39%
- Less than 4% error
- Monitoring of the following SDRAM controller parameters:
  - Write buffer hits (implying merge or collapse)
  - Read merge hits (from the write buffer)
  - Write buffer full occurrence
  - Read buffer hits
  - SDRAM page and bank misses
- Two ADDIE resources that can be configured to concurrently monitor any two SDRAM controller parameters

Registers

The ADDIE performance monitors are controlled by the performance monitor control registers, as shown in Table 1.

Refer to Appendix A, “SDRAM Performance Monitor Registers” for more information about the performance monitor registers.

Table 1. Performance Monitor Control Registers (Memory Mapped)

Register Name	MMCR Offset Address	Function
Performance Monitor Control	44h	Selects parameter to monitor for both ADDIE 0 and ADDIE 1.
Performance Monitor Data 0	48h	Returns ADDIE 0 performance data.
Performance Monitor Data 1	49h	Returns ADDIE 1 performance data.

Publication# **23380** Rev: **A** Amendment/**0**
Issue Date: **April 2000**

Block Diagram

The ADDIE performance monitors are integrated into the SDRAM controller's subsystem as shown in Figure 1. The performance steering logic allows the appropriate signals to be routed from the various components of the SDRAM controller's sub-system based on the user's configuration. Each ADDIE is capable of functioning independently.

The performance monitor is implemented with two ADDIE devices with multiplexor circuitry. The two ADDIE devices select the parameter to be monitored with a configuration register.

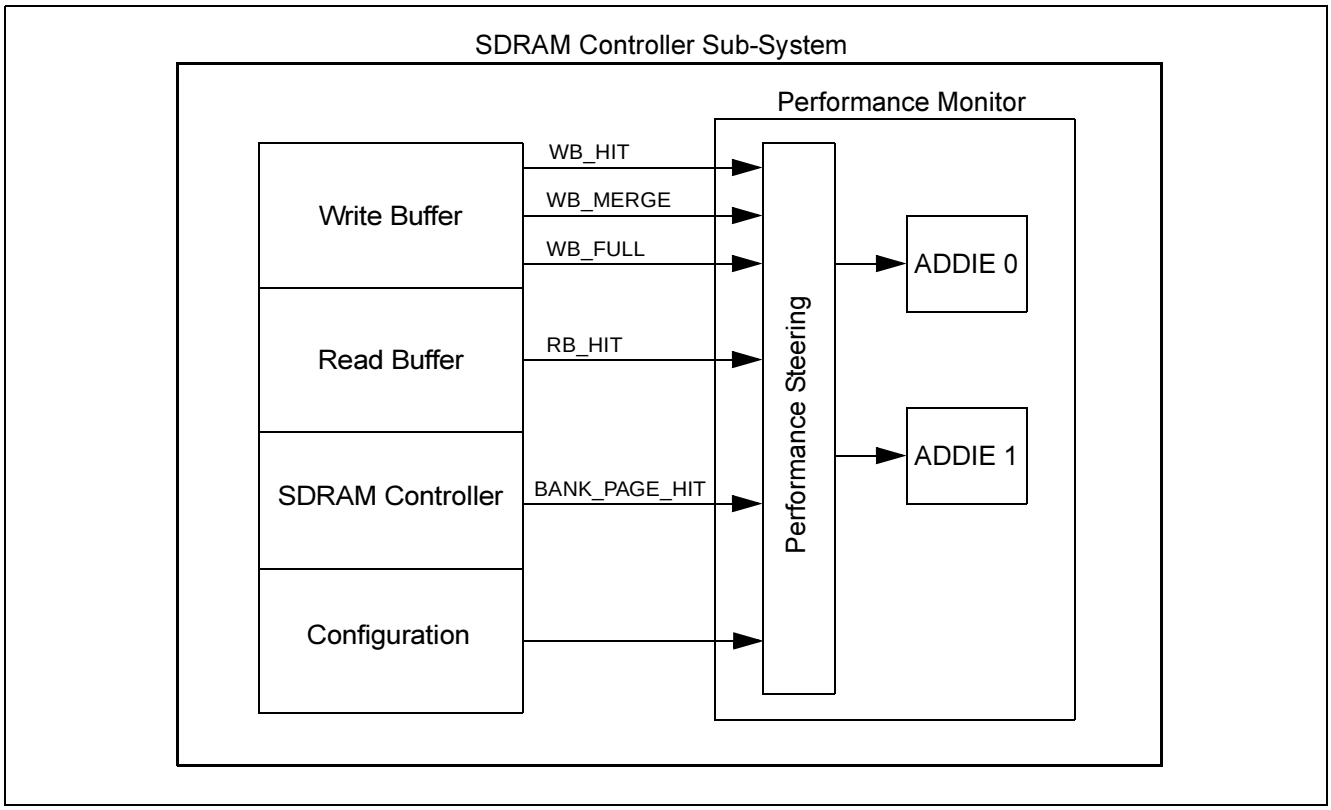

Figure 1. Block Diagram of SDRAM Subsystem with ADDIE Performance Monitors

ADDIE PERFORMANCE MONITORING PARAMETERS

The performance monitors are capable of independently monitoring the following parameters:

- Write buffer hits (implying merge or collapse)
- Read merge hits (from the write buffer)
- Write buffer full occurrence
- Read buffer hits
- SDRAM page and bank misses

Each parameter is explained below in detail.

Write Buffer Hit Monitoring

The ÉlanSC520 microcontroller's SDRAM controller contains a 32-rank write buffer with each rank providing four bytes of write data storage. Combined, these ranks provide up to eight cache lines of write data storage for transfers initiated by either the $Am5_x86$® CPU, PCI host bridge (on behalf of a PCI master for writing data to SDRAM), or GP bus DMA. A write transfer initiated by either the $Am5_x86$ CPU or host bridge master can be a single 32-bit DWORD or a burst of up to four DWORDs during a single tenure. Each DWORD write transaction can be either four bytes or less. A DWORD write transfer of three bytes or less is referred to as a partial DWORD. The $Am5_x86$ CPU does not burst partial DWORDs and only bursts write data during a cache copy-back or write-back. However, the PCI host bridge can burst transfer with all complete DWORDs, all partial DWORDs, or a combination of both. GP bus DMA write transfers are never larger than two bytes.

The ÉlanSC520 microcontroller's write buffer supports standard FIFO buffering and also supports a write data merge and collapse feature. Write merging occurs when a sequence of individual writes are merged into a single DWORD.

Merging implies that the same byte location is not written more than once. For example, four individual byte transfers from address 0, 1, 2, and 3 are merged in the write buffer to form a single DWORD, thus converting four independent byte transfers into a single DWORD transfer to SDRAM.

Collapsing is similar to merging with the exception that the same byte locations can be written more than once. For example, an $Am5_x86$ CPU cache snoop, that is a result of a PCI host bridge burst of four DWORD transfer to SDRAM, first requires a cache line write-back (as a result of a hit in the $Am5_x86$ CPU's write-back cache). First, the cache line write-back data is written to the write buffer, followed by the PCI host bridge transfer to the same four DWORD addresses. Because the write buffer supports collapsing, the cache line that was written to the write buffer by the CPU due to a write-back is over-written with the write data from the PCI host bridge transfer, thus collapsing on the cache's write-back data. Instead of eight DWORD transfers to SDRAM, the collapse feature of the write buffer only requires four DWORD transfers to SDRAM. The write buffer is intended to de-couple the master's write traffic from incurring the overhead associated with SDRAM refresh cycles and page/bank misses.

In addition to possibly reducing the total number of write transactions to SDRAM, the merge/collapse feature helps to assemble independent partial DWORD transfers into complete DWORDs, so the overhead associated with error correcting code (ECC) read-modify-write cycles can be reduced. Read-modify-write transfers are required for ECC support when a partial DWORD write occurs to SDRAM. Complete DWORD writes do not require a read-modify-write function.

To provide the merge and collapse features, the write buffer incorporates a content-addressable memory (CAM). The CAM performs a look-up of DWORD addresses that currently exist within the write buffer with an address presented by the $Am5_x86$ CPU, PCI host bridge, or GP bus DMA that requests the write transfer. During a CPU or PCI host bridge burst transfer, *each* DWORD address during the transfer is searched within the CAM.

Either of the two performance monitor resources can be configured to provide a hit average of the number of write buffer hits that occurred during either an $Am5_x86$ CPU, PCI host bridge, or GP bus DMA write transfer. A hit implies that a merge or collapse of write data occurred. Each DWORD write transfer to the write buffer, either complete or partial, is monitored independently of each transfer being a single DWORD or a burst of two, three, or four DWORDs. A ratio of write buffer HIT/MISS is provided by the performance monitors.

Write Buffer Hit Monitoring Analysis

The write buffer provides a write data merge and collapse feature where a write of data to a DWORD that currently exists in the write buffer can be merged or collapsed into the same location in the buffer. This feature is used to reduce the overall number of consecutive write transactions to the same location in SDRAM. This feature also attempts to merge partial DWORD write transfers into complete DWORDs in the effort to reduce the number of ECC read-modify-write cycles to SDRAM.

Either performance monitor can be used to provide a write buffer hit average. This information can be used to specify an optimal watermark setting for the write buffer. A higher watermark setting allows more write data to accumulate in the write buffer before write-backs are initiated to SDRAM. This provides a greater chance for data merging and collapsing to take place and lessens the occurrences of writes from the write buffer interrupting the read accesses. A lower watermark setting causes the write buffer to request SDRAM accesses with fewer DWORDs in the buffer. This causes the write buffer to possibly interfere with read accesses, but lessens the occurrence of overflowing the write buffer during complete DWORD transfers when write data merging and collapsing is less likely to occur.

A lower hit average with a high watermark setting might imply that most write data is a complete DWORD, thus merging and collapsing is not occurring as often. In this condition, a lower watermark setting is recommended to prevent write buffer full occurrence. A higher hit average with a low watermark setting might imply that a high occurrence of write data merging or collapsing exists. In this condition, a higher watermark setting is recommended to allow the write buffer to absorb the write activity, without interrupting read accesses.

Write Buffer Read Merge Monitoring

As mentioned in "Write Buffer Hit Monitoring Analysis", the write buffer merges and collapses data. Because the write buffer's CAM makes the merge and collapse features during write transfers possible and only a single entry will exist for any given DWORD address that currently exists in the write buffer, data read merging from the write buffer is possible. This implies that a read request by either the $Am5_x86$ CPU, PCI host bridge, or GP bus DMA to an address that currently exists in the write buffer does not require the write buffer to first be flushed to SDRAM to maintain data coherency prior to the read request being satisfied. Standard first-in first-out (FIFO) buffering techniques without a CAM function require that the buffer be flushed prior to every read because the data, in the buffer that pertains to the data being requested during the read, is not known. Instead of the possible

large overhead associated with a write buffer flush prior to every read request, the write buffer performs a read merge data from the write buffer's contents as data is returned from SDRAM. Read merge occurs when a read cycle hits a dirty DWORD that currently exists in the write buffer and the read data returned from SDRAM is replaced, or merged with existing bytes from the write buffer. Read merging from the write buffer occurs at the byte resolution.

For example, an $Am5_x86$ CPU cache snoop that is a result of a PCI host bridge read request requires a cache line write-back that is a result of a hit in the $Am5_x86$ CPU's write-back cache. First, the cache line write-back data is written to the write buffer, followed by the PCI Host Bridge read request to SDRAM, possibly to more than one of the same four DWORD addresses written during the cache line write-back. The cache line write-back data is written to the write buffer, but the read request from the PCI host bridge can be satisfied by the SDRAM controller prior to the cache line write data being actually written to SDRAM. Because the write buffer supports read merging, the SDRAM data is replaced by the more recent data from the write buffer that was just written during the cache line write-back. This feature enables the SDRAM controller to service the demand of read requests overwrite cycles, without the penalty of flushing the write buffer first.

Either of the two performance monitor resources can be configured to provide a read merge average of the number of DWORD read transfers during either an $Am5_x86$ CPU, PCI host bridge, or GP bus DMA read request that results in a read merge from the write buffer. A read merge implies that at least one of the four bytes within a DWORD resulted in a read merge during *each* DWORD of a read request. Each DWORD of a read transfer is monitored independently of each read transfer being a single DWORD, or a burst of two, three, or four DWORDs. A ratio of write buffer read-merge/no-read-merge is provided by the performance monitors.

Write Buffer Read Merge Monitoring Analysis

Conventional write buffer architectures do not provide the read merge function, and require the entire buffer to be flushed to SDRAM if a read access occurs to any data that currently exists in the buffer. When this occurs, the read access incurs the overhead associated with the write-back of *all* buffer contents to SDRAM instead of just the data that is needed to maintain data coherency. However, because the ÉlanSC520 microcontroller's write buffer provides merging and collapsing, this forced flush on the occurrence of a read access that currently exists in the write buffer is *not* necessary. The read access continues around the associated write data, and the more current write data is merged in from the write buffer as the read data is being returned to the requesting master from SDRAM.

Either performance monitor can be configured to provide a read merge average. This average can be used to determine if the read-around-write feature of the write buffer provides performance advantages.

Write Buffer Full Monitoring

The write buffer is intended to de-couple the master's write traffic from incurring the overhead associated with SDRAM refresh cycles, page bank misses, and ECC read-modify-write cycles as a result of incomplete DWORD writes. When the write buffer is not full, write data is posted in zero wait states. However, due to the various delays mentioned above, data can be posted to the write buffer faster than data is written to SDRAM. This can result in a full write buffer such that no more data can be posted until data is written from the write buffer to SDRAM.

The write buffer's watermark setting can 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 enabling a write data merge or collapse to occur, but delays requesting the SDRAM for service for a write-back of write-buffered data to SDRAM. A lower watermark setting requests SDRAM service when less data is stacked into the write buffer.

Either of the two performance monitor resources can be configured to provide a write buffer full average of the number of write attempts by either an $Am5_x86$ CPU, PCI host bridge, or GP bus DMA write request that experienced the write buffer in a full state. A write buffer full state implies that an attempt to write a DWORD into the write buffer was stalled. Each DWORD of a write transfer is monitored independently of each write transfer being a single DWORD, or a burst of two, three, or four DWORDs. A ratio of write buffer full/not-full is provided by the performance monitors.

Write Buffer Full Monitoring Analysis

The write buffer provides 32 DWORDs of storage for master write accesses to SDRAM. It is intended to absorb the overhead associated with page and bank misses during write cycles and ECC read-modify-write cycles. The write buffer experiences a full condition when master accesses write data into the write buffer at a faster rate than data can be removed from the write buffer into SDRAM. When full, master accesses will experience delays associated with write accesses to SDRAM.

Either performance monitor can be used to monitor the write buffer full average, which can be used to determine the best write buffer watermark setting and also provide information regarding write overhead associated with page and bank misses, and ECC read-modify-write. The write buffer's watermark settings are

provided for the user to throttle the write request interruptions to SDRAM. Because read-around-write accesses are provided when the write buffer is enabled, the watermark setting is provided to specify how full the write buffer can become before requesting service to write data back to SDRAM. The write buffer requests SDRAM access when the configured watermark is reached. The write buffer provides four watermark settings. A higher watermark can be used to delay write accesses from interfering with read accesses by stalling write activity to SDRAM, which also provides a greater chance of data write merging and collapsing where a large amount of partial DWORD writes are expected. A low watermark setting can be used when the expected result is that most writes to SDRAM are complete DWORDs, thus a reduced chance of write buffer merging and collapsing and the ECC read-modify-write cycles are not necessary. The user should configure the write buffer's watermark setting to a position that yields the lowest write buffer full average.

Read Buffer Hit Monitoring

The ÉlanSC520 microcontroller's SDRAM controller contains eight 4-byte read data buffers. When combined, these buffers comprise the read buffer and are designed to hold two cache lines of data that are returned from SDRAM for transfers initiated by either the Am5$_x$86 CPU, PCI host bridge (on behalf of a PCI master), or GP bus DMA. During any read request to SDRAM, an entire cache line is always read into the read buffer, independent of each read transfer being a single DWORD, or a burst of two, three, or four DWORDs. To maintain data coherency, any write transfer to SDRAM on behalf of any master or the write buffer that matches a cache line in the read buffer, results in both cache lines of the read buffer being invalidated. When the read prefetch feature is enabled and the read request is a burst type (i.e., two or more DWORDs), the entire cache line of the request *and* the cache line following the requested cache line is also fetched from SDRAM. The cache line following a single DWORD read request is never prefetched; however, the remainder of the requested DWORD's cache line is stored in the read buffer.

For example, if the Am5$_x$86 CPU or PCI host bridge requests a single DWORD of data from SDRAM, the SDRAM controller fetches the entire cache line of data on behalf of this single DWORD read request. The requested DWORD is forwarded to the requesting master, with the requested DWORD and remaining DWORDs of that cache line being stored in the read buffer. Any future accesses to the same cache line within the read buffer result in read buffer hits.

Either of the two performance monitor resources can be configured to provide a read buffer hit average of the number of DWORD read transfers during either an Am5$_x$86 CPU, PCI host bridge, or GP bus DMA read request that results in a hit to the read buffer. A read buffer hit implies that at least one of the four bytes within a DWORD resulted in a read buffer hit. The performance monitor counts read buffer hits on the basis of an atomic read request during the same bus tenure, independent of burst length (i.e., complete cycle, regardless of the amount of read data requested during the same burst tenure). Therefore, each read *request* is monitored, instead of each DWORD transferred, during that read request tenure. This occurs because the read buffer always stores an entire cache line of read data from SDRAM, independent of the number of DWORDs requested during the read request. A read request of two, three, or four DWORDs that hit within the read buffer are counted by the performance monitors as only one hit to the read buffer because the read buffer always maintains an entire cache line of data, and a hit of one DWORD during a read burst request is guaranteed to hit the remaining data in that cache line during that same bus tenure. This occurs instead of unfairly counting four read buffer hits during a burst of four DWORDs requested because the entire cache line would result in a hit. Four *independent* read requests of one DWORD each result in four independent read buffer hits by the performance monitor because each read transfer is an individual read request. A ratio of read buffer HIT/MISS is provided by the performance monitors.

Read Buffer Hit Monitoring Analysis

The ÉlanSC520 microcontroller's SDRAM controller maintains two cache lines of read data within the read buffers. These buffers are always enabled. For any read request, an entire cache line of data is stored in the read buffer. When the SDRAM controller's read prefetch feature is enabled and the read request is a burst type (i.e., two or more DWORDs), the entire cache line that contains the requested data *and* the cache line following the requested cache line are also fetched from SDRAM. The read prefetch feature is intended to accelerate SDRAM read accesses if read requests are sequential, so that data is supplied to the master while the next cache line from SDRAM is concurrently prefetched, relying on sequential accesses. The read buffers prefetch feature relies on long tenure of a single requesting master, which can be affected by the enable state of the Am5$_x$86 CPU's cache, program flow, mastership changes, etc. However, like all buffering techniques that speculatively prefetch, the anticipated prefetched line might not be used, possibly resulting in overhead associated with the unused prefetch.

Either performance monitor can be used to provide a read buffer hit average. This information can be used to provide feedback to determine if the read prefetch

feature of the read buffer should be enabled. A high read buffer hit average when read prefetching is enabled implies that the read prefetch data is being used frequently. A low read buffer hit average with the read prefetch feature enabled can imply that the read prefetched data is not being utilized; thus, disabling the read prefetch feature can be desirable.

SDRAM Page and Bank Miss Monitoring

SDRAM devices support either two or four internal banks. The page width of the internal banks are defined by the device's symmetry. The ÉlanSC520 microcontroller's SDRAM controller supports SDRAM devices with 8-, 9-, 10-, or 11-bit column address widths resulting in either 1-, 2-, 4-, or 8-KB page width for the ÉlanSC520 microcontroller's 32-bit data bus width.

Overhead is associated with opening an internal bank. Therefore, the more often an open internal bank is utilized, the greater the overall performance. An internal bank's page is left open after each access. A page miss occurs when a master access is made to an internal bank page that is not open within that same internal bank. The penalty incurred results in a delay associated with closing the currently open page and opening the requested page of the requested internal bank. A bank miss occurs when a master accesses an internal bank where no current open pages exist; for example, after a refresh cycle. The penalty incurred results in a delay associated with opening the requested page.

Either of the two performance monitor resources can be configured to provide a page and bank miss average of the number of read, write, or write buffer transfers resulting in a page bank miss to SDRAM. The performance monitor scores SDRAM page and bank accesses on the basis of an atomic request, during the same bus tenure, independent of burst length (i.e., complete cycle regardless of the amount of data requested during the same burst tenure.) Therefore, each request is monitored, instead of each DWORD transferred, during that requested cycles tenure. This occurs because after an access, the page within each bank of the SDRAM devices remains open, independent of the number of DWORDs requested during the cycle request. An $Am5_x86$ CPU, PCI host bridge, or GP bus DMA request of two, three, or four DWORDs that miss within an open page are counted by the performance monitors as only one miss to the page because the remaining DWORDs during a burst request are guaranteed to result in a page hit during that same bus tenure. Thus, the first access is counted independently of the amount of data transferred in that single cycles bus tenure. This occurs instead of unfairly counting one page bank miss and three following page bank hits during a burst of four DWORDs because the remaining three transfers always result in a page hit. Four independent read or write requests of one DWORD each result in four independent hit/misses by the performance monitor because each read or write transfer is an individual request. The write buffer always writes single DWORDs; therefore, each write buffer write is counted independently. This occurs because the write-buffer write-backs are a single DWORD and provide a read-around-write function. For example, one DWORD of an entire cache line written into the write buffer is written into SDRAM while the remaining DWORDs of the same cache line remain in the write buffer. If a higher priority read request demands access to SDRAM over the write buffer's access, the dynamics of the page bank relationship are changed, resulting in a different page bank miss occurrence over the same scenario where a write burst occurs, but with the write buffer disabled. A ratio of page-bank-miss/page-bank-hit is provided by the performance monitors.

SDRAM Page and Bank Miss Analysis

The overall performance of SDRAM is directly impacted by the overhead associated with a page or bank miss. The more often an access occurs to an open page (spatially local to that page), the faster data is returned to the requesting master during a read access or written to SDRAM during a master write access. Master accesses that are sequential will hit within an open SDRAM page and yield higher SDRAM access performance than master accesses that result in heavy thrashing of the pages.

Many system parameters and configurations contribute to the dynamics of SDRAM page and bank misses. Some of these are program flow, $Am5_x86$ CPU cache enable, $Am5_x86$ CPU cache write-through vs. write-back, number of GP bus DMA channels active, and number of PCI masters and their burst size. For example, read accesses initiated by the $Am5_x86$ CPU's prefetcher are typically sequential until the program flow changes as a result of a program branch, and can utilize an open SDRAM page more frequently. $Am5_x86$ CPU write accesses tend to be directly program-dependent and not predictable and can result in SDRAM page thrashing. These dynamics change when the $Am5_x86$ CPU's cache is enabled and when the dynamics are in write-though or write-back mode. PCI read transfers are linear, and those that request a large burst utilize an open page. Even though the dynamics associated with program flow and master accesses heavily dictate the page and bank miss rates, the user has control over some SDRAM parameters that can lessen the impact associated with system dynamics. These dynamics are as follows:

- Adjustable page widths by selecting devices with either 8-, 9-, 10-, or 11-bit column addresses

- Adjustable count of internal banks supported within the SDRAM devices, either two or four internal banks
- Adjustable refresh rates
- Adjustable write buffer watermarks when the write buffer is enabled

Either of the two performance monitor resources can be configured to provide a page and bank miss average. This information can be used to provide feedback on which SDRAM controller configuration or SDRAM device architecture results in the best overall SDRAM page performance.

Because the SDRAM controller supports SDRAM device architectures yielding a page width from 1 KB to 8 KB, performance monitoring can determine that SDRAM devices with a larger page organization yield better page performance than a device with a smaller page size. The ÉlanSC520 microcontroller provides page size selection via the SDRAM controller's configuration registers. However, the SDRAM device dictates the page size, and the SDRAM controller must be configured to exactly match that devise's page size.

The number of internal banks supported can also affect the overall page performance of the device. Since SDRAM devices support either two or four bank architectures, the performance monitor can be used to determine which SDRAM internal bank architecture yields better page performance.

An SDRAM refresh cycle closes all pages of all banks within the SDRAM devices. As a result, any access to a bank after a refresh occurs will result in a bank miss, thus incurring the associated overhead penalty. Therefore, the refresh rate directly impacts system SDRAM performance. A faster refresh rate closes the SDRAM pages more often than a slower rate. The refresh rate is dictated by the SDRAM device and is a function of the number of rows contained in the device. Some devices may allow for a slower refresh rate because the column width is wide. The performance monitor can be used to select the SDRAM controller's refresh rate to determine the effects of the refresh cycle on page performance. Keep in mind that a device can be over-refreshed, but never under-refreshed without the risk of loosing data integrity.

When the write buffer is enabled, read accesses have priority over writes, thus providing a mechanism for satisfying read requests faster. The write buffer and the associated watermark setting can change the SDRAM page dynamics over the current condition and setting if the write buffer is enabled and strict access ordering to the SDRAM is preserved. This implies that posted writes that influence the page state of the devices have not yet occurred to the SDRAM. Thus, a subsequent read access can utilize a page that is currently open instead of possibly experiencing a closed page due to the write access, if the write buffer is not enabled. The write buffer's watermark setting has a direct impact on when data is written to SDRAM. A low watermark setting causes the write buffer to request a write-back earlier than a higher watermark setting. The performance monitor can be used to monitor the page performance based on the write buffer's enable state or watermark settings.

Software Considerations

A software handler is required to configure the performance monitors, read the monitors' performance data, and calculate the average percentage of the occurrence of the parameters being monitored. The ADDIE devices provide an 8-bit value that represents the overall average. This hexadecimal value must be divided by 255 (which represents the maximum value) to determine the percentages.

RESET

The performance monitors are reset during a power-on system reset event or programmable reset event. As a result of these system reset events, all monitoring information is lost. However, the configuration register that selects the parameter to be monitored is maintained through a programmable reset. The performance monitors do not support a manual reset feature.

INITIALIZATION

Both performance monitors are disabled with a system reset. They are enabled by accessing a configuration register that selects the parameter that is to be monitored. When the parameter being monitored is altered by accessing a configuration register, some time is necessary for the ADDIE to begin to track the newly configured parameter. Disabling the performance monitor causes the monitor status to freeze and retain its value prior to being disabled.

NON-INTRUSIVE PERFORMANCE MONITORING

Most desktop processors are equipped with performance monitoring counters. These counters permit processor performance parameters to be monitored and measured, which is useful for performance tuning. Current techniques typically utilize two counters, which 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.

This section describes 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 more suitable for monitoring a wide range of performance parameters.

Performance Monitoring Counters

Typically two large counters (i.e., 40 bits or more) are provided for event counting. These counters can be read and written from within the register address space. The counters can be configured to measure parameters such 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 actual 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. Such interrupts are undesirable as they intrude into normal processor operation. A larger counter also results in greater data averaging. Any temporary fluctuation in cache hit rate is not observed, and this might be required.

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

Data Integration

The type of performance data 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 data enables a precise value to be generated for the data examined. These precise values appear as 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 measure this deviation.

For example, at each memory access, an on-chip cache can successfully provide the required data. The sequence of hit and miss data can be represented by a simple 1 or 0 bit stream, the probability of a 1 being the probability of a hit occurring. A stochastic ADDIE can be used to integrate the probability stream and determine the relevant probability.

Figure 2 shows a possible ADDIE configuration. A counter is compared with a random number; if the counter is bigger than the random number, a 1 is generated. Large counter values are more likely to produce a 1 output from the comparator than small counter values.

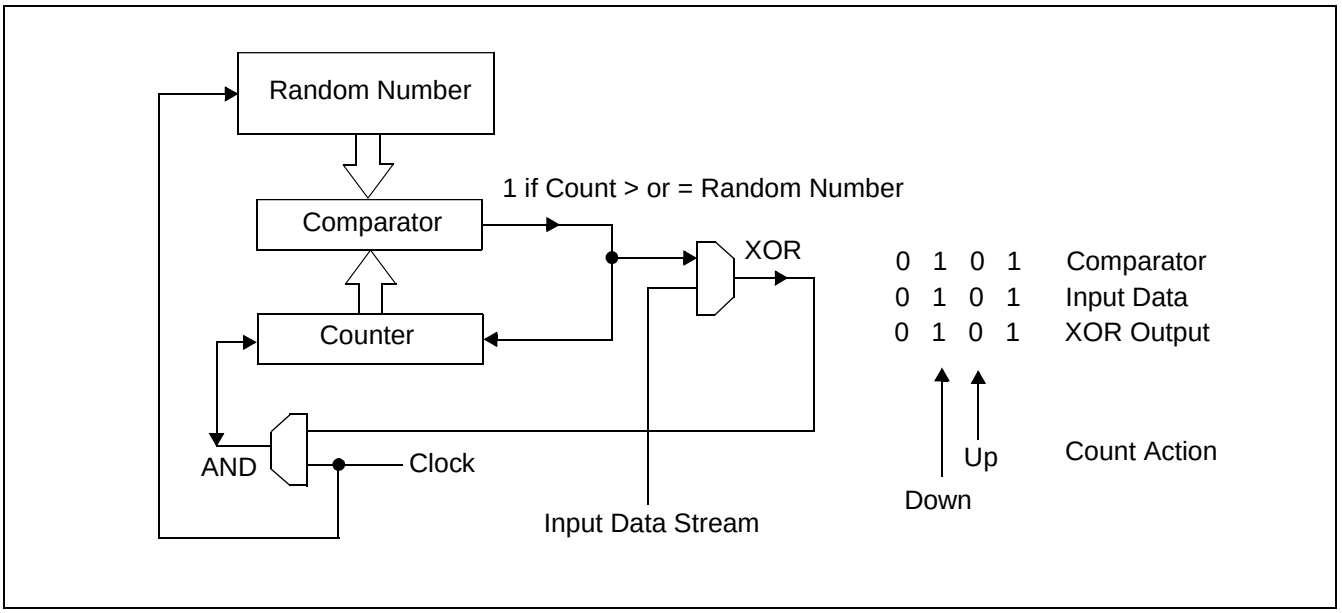

Figure 2. Block Diagram of the ADDIE

The counter output is compared with the 1/0 data stream type—the cache-hit information. These two stochastic data streams are compared to determine which one has the highest probability of being 1. Determining this probability appears difficult; however, only an XOR gate is required. When the data streams

differ, a difference in probability exists. This information is fed back into the data stream 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 data stream.

The ADDIE device effectively integrates the probability stream. The result is that the probability stream is converted into a digital value that is held in the counter, which represents the probability of the parameter being measured.

This method of measuring probability is very different from the dual-counter method. No potential overflow exists, and an overflow interrupt handler is not required. The counter can be read at any time to give a measure of the current probability.

8-bit ADDIE

As the number of bits used by the counter and random number generator increase, the probability resolution is improved. For example, an 8-bit counter provides for 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 being required before a good estimate of probability can be obtained. Figure 3 shows an 8-bit ADDIE used to measure a probability of 25%.

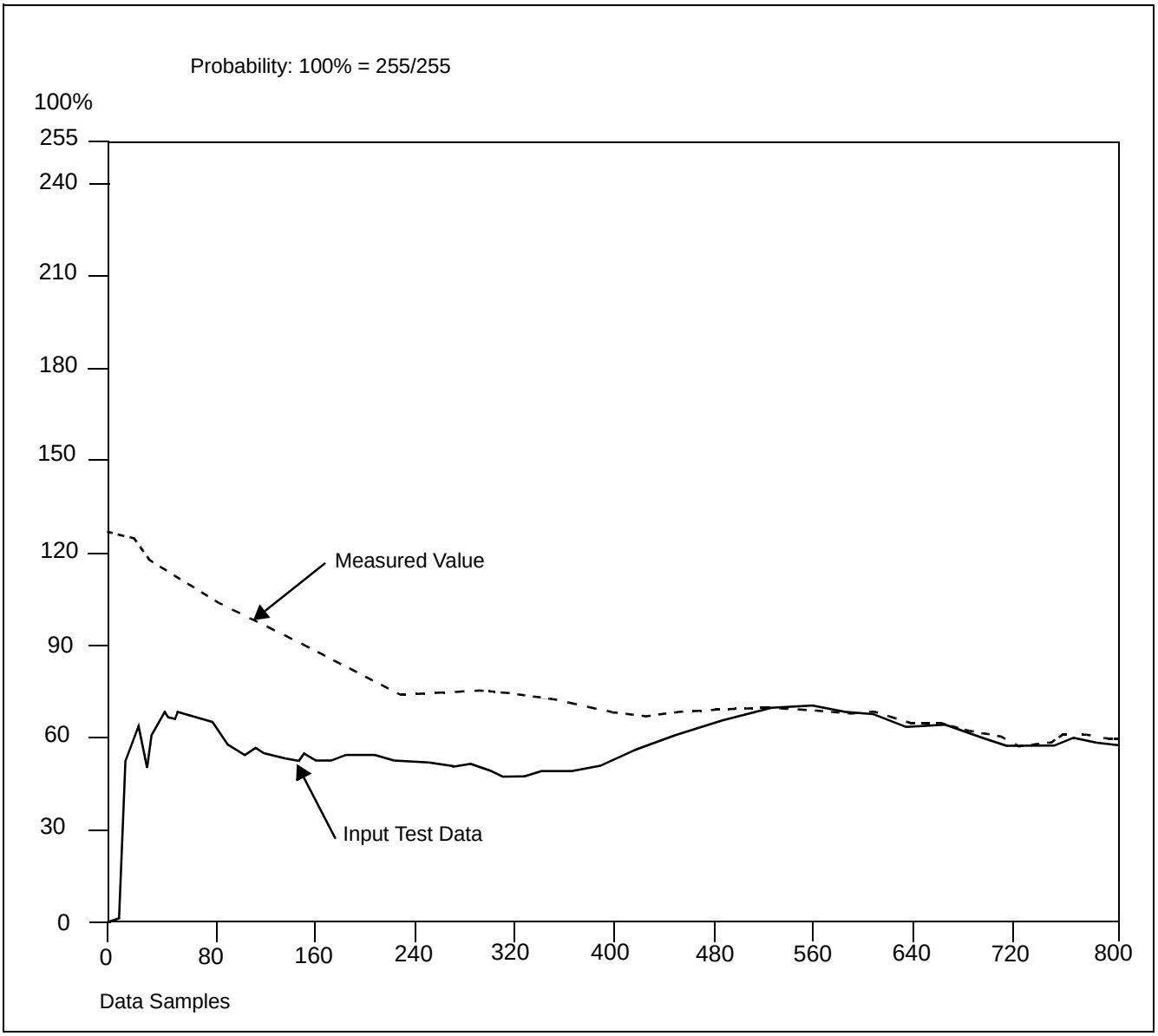

Figure 3. 8-Bit ADDIE Measuring 25% (64/255)

The ADDIE starts with an initial value of 50%. As the input data is sampled, the ADDIE's counter value moves toward the expected value. After about 500 samples, the ADDIE closely tracks the input data stream. This indicates that after as little as 500 samples, the counter is read to produce an estimate of the performance parameter being measured.

The data shown in Figure 3 indicates the average of over 250 inputs and ADDIE streams. Keeping this ratio small helps one to observe temporary fluctuations in both the generated test data and ADDIE response.

Figure 4 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%. After about 500 samples, the measuring instrument converges on the desired value.

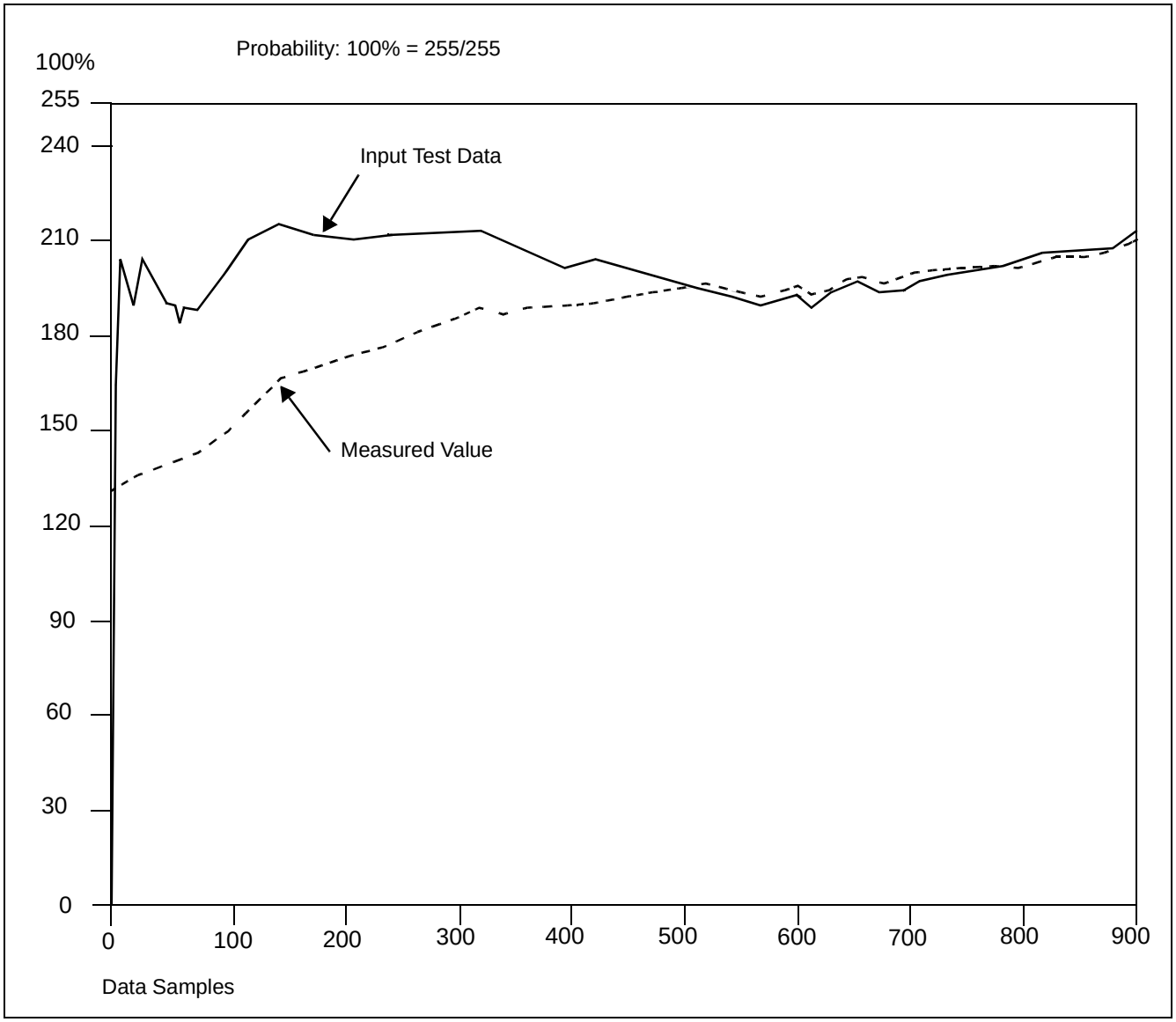

Figure 4. 8-Bit ADDIE Measuring 78% (200/255)

12-bit ADDIE

The time taken by the ADDIE's counter to move from its current position to the desired value is dependent on the number of bits used by the counter. An 8-bit counter takes only 256 clock pulses to increment through its full range. A 12-bit counter takes 4096 clock pulses to cover its complete range. The extra 4 bits of probability resolution requires 16 times more clock pulses to scan its complete counting range. This condition directly relates to the number of samples required to measure a performance parameter. Greater resolution requires a greater number of samples to achieve a measurement with similar results. Figure 5 shows a 12-bit ADDIE used to measure a 78% probability. The counter was initiated with a 50% probability. The results show that about 10,000 samples where required before the input and measured probability streams converge.

The larger ADDIE offers greater measurement resolution. However, for measuring on-chip performance parameters, an 8-bit counter appears adequate. The smaller ADDIE provides the benefit of more quickly converging on the required value and better tracking local fluctuations of the input data stream.

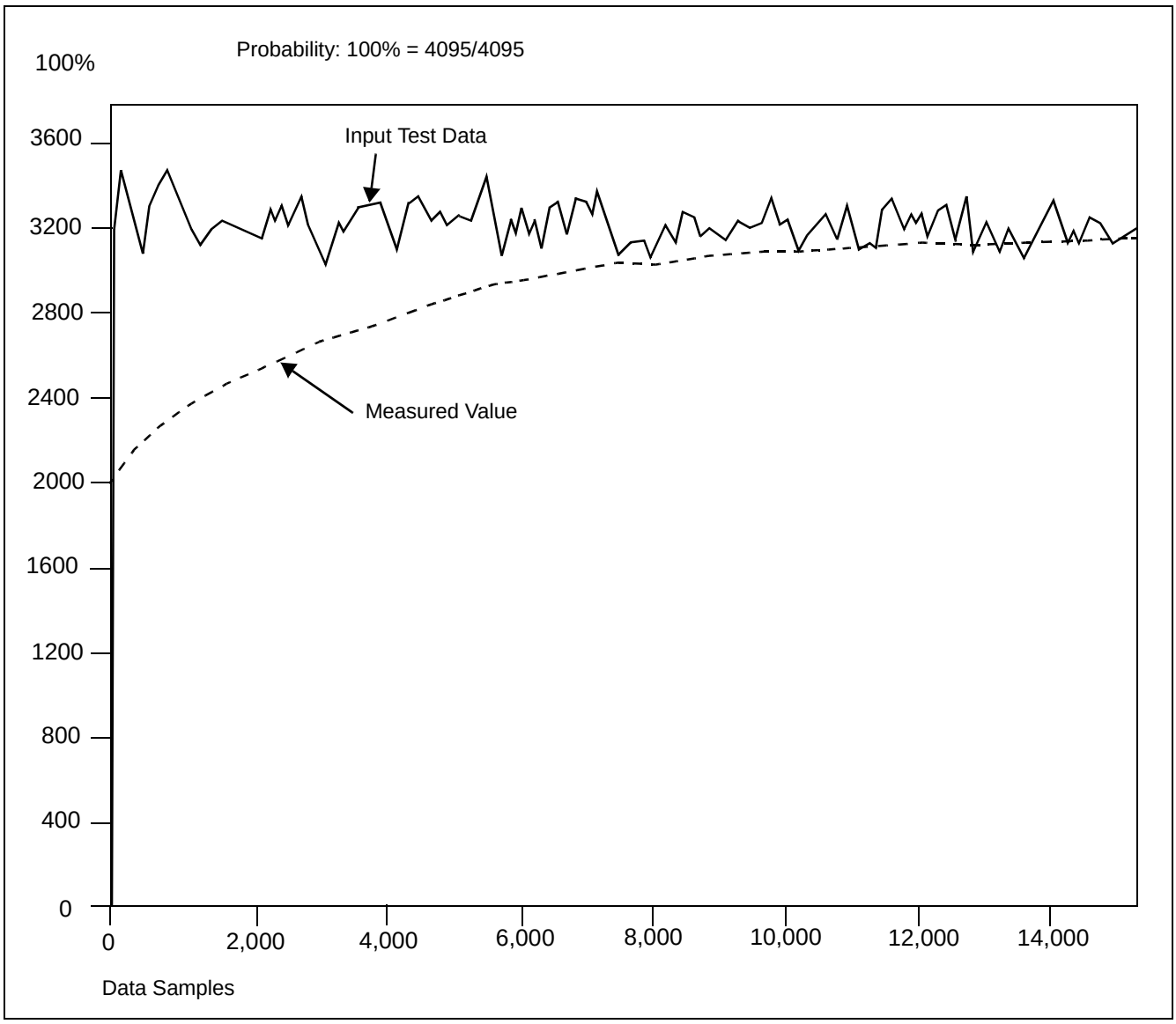

Figure 5. 12-Bit ADDIE Measuring 78% (3200/4095)

PARAMETER VALUES AS PROBABILITIES

The performance parameters described above are easy to represent by a single bit, for example, cache-hit information. However, other parameters need more than one bit, such as the number of cycles needed to access external memory or the number of clock pulses while the pipeline is stalled. These parameters must be converted into a pulse stream consisting of several bits.

Measuring performance parameters is not a function of the ÉlanSC520 microcontroller. However, the example shown in Figure 6 can be used to measure a performance parameter that ranges between the value 0 and 4 and requires four bits.

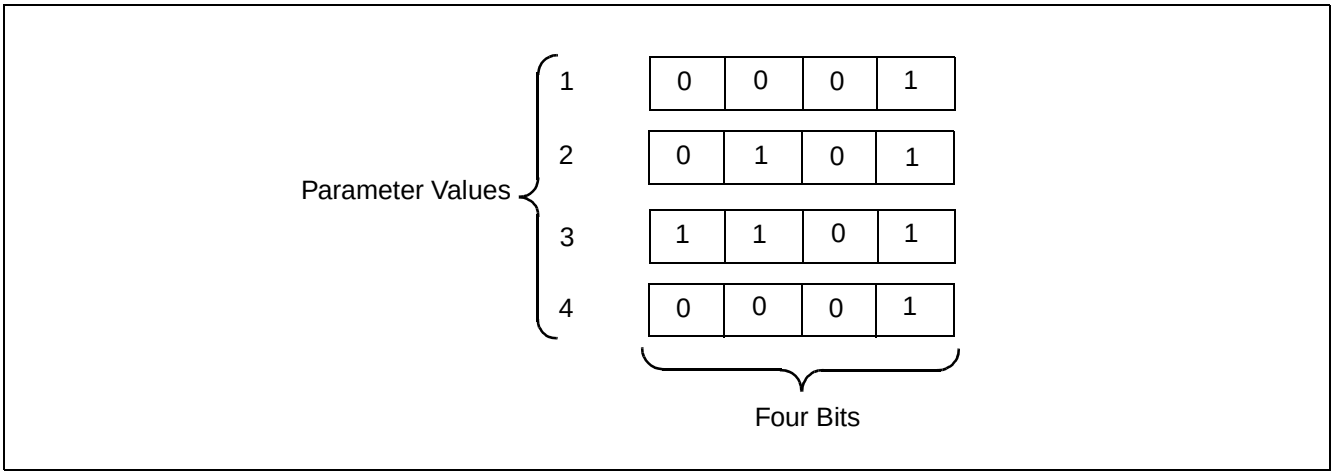

Figure 6. Performance Parameters

For example, if a parameter is measured to be three cycles, then three bits of the 4-bit pulse stream must be set. This pulse stream is serially clocked into the ADDIE. There is no synchronous requirement for presenting data to the ADDIE. New parameter data can be sampled at any time and clocked into the ADDIE convertor.

Different parameters have differing value ranges. The example in Figure 6 has a value range of 0 to 4; another parameter might have a value range 0 to 16. All parameters are not required to be restricted to the same data range. Adjusting the number of data bits applied to each parameter's data range efficiently utilizes the ADDIE's restricted probability resolution.

An 8-bit ADDIE can be used to measure a parameter with the value range of 0 to 4, then later used to measure a parameter with the data range of 0 to 16. In all cases, the ADDIE determines the probability of the data stream being 1. For a value range of 0 to 4, a probability value of 75% represents a measured value of 3 (100% being 4).

For example, when generating a sample value 3, any three bits can be set. The order is not important. However, dispersing the setting of the bits can have a small advantage in reducing the variance in counter operation.

RANDOM NUMBER GENERATION

The ADDIE operation requires a random number generator. 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. For an example of a random number generator, refer to Figure 7 on page 13.

The ADDIE can be prompted to converge on the required value by using random numbers that correlate. This condition is achieved by selecting consecutive stages of the shift register and inverting the most significant bit (MSB). Hence, the MSB of the current random number is inverted and becomes the bit occurring just before the MSB of the next random number.

The test data shown in Figure 7 on page 13 uses a 31-bit shift register with feedback taken from stages 3 and 31. The top eight or 12 bits were used to form the required random number. A smaller (less than 31-bit) shift register can also be used.

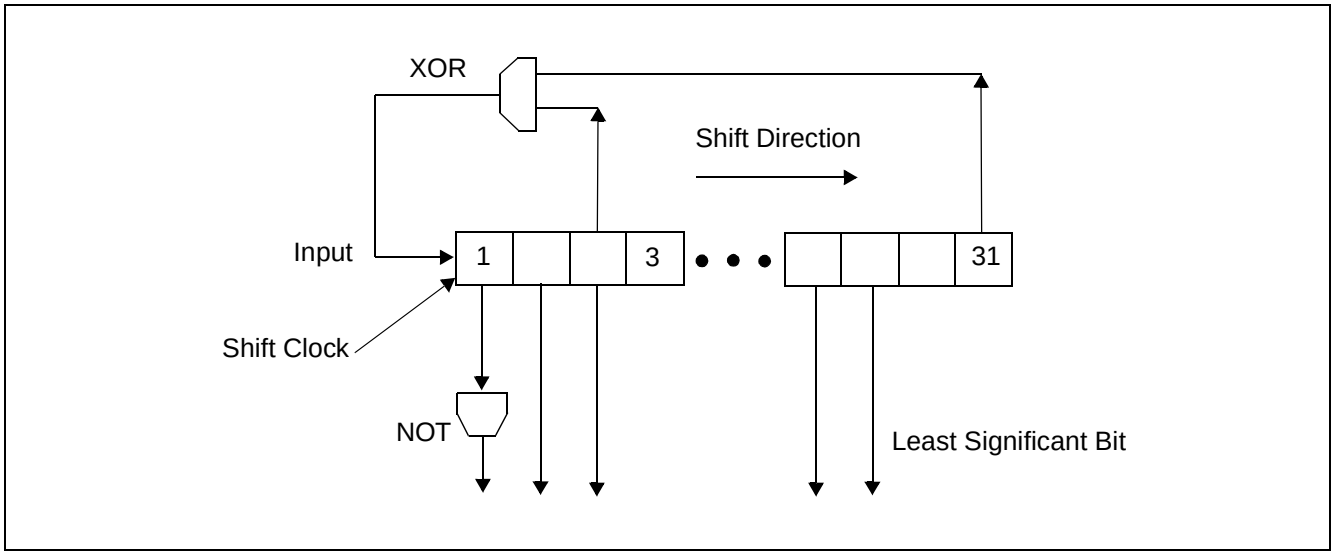

Figure 7. Random Number Generator

SHARING AN ADDIE

An ADDIE is not required to be allocated to each parameter being measured. A more practical system uses a single ADDIE that feeds a bit stream via an input multiplexer. This enables a single parameter to be selected for measurement shown in Figure 8.

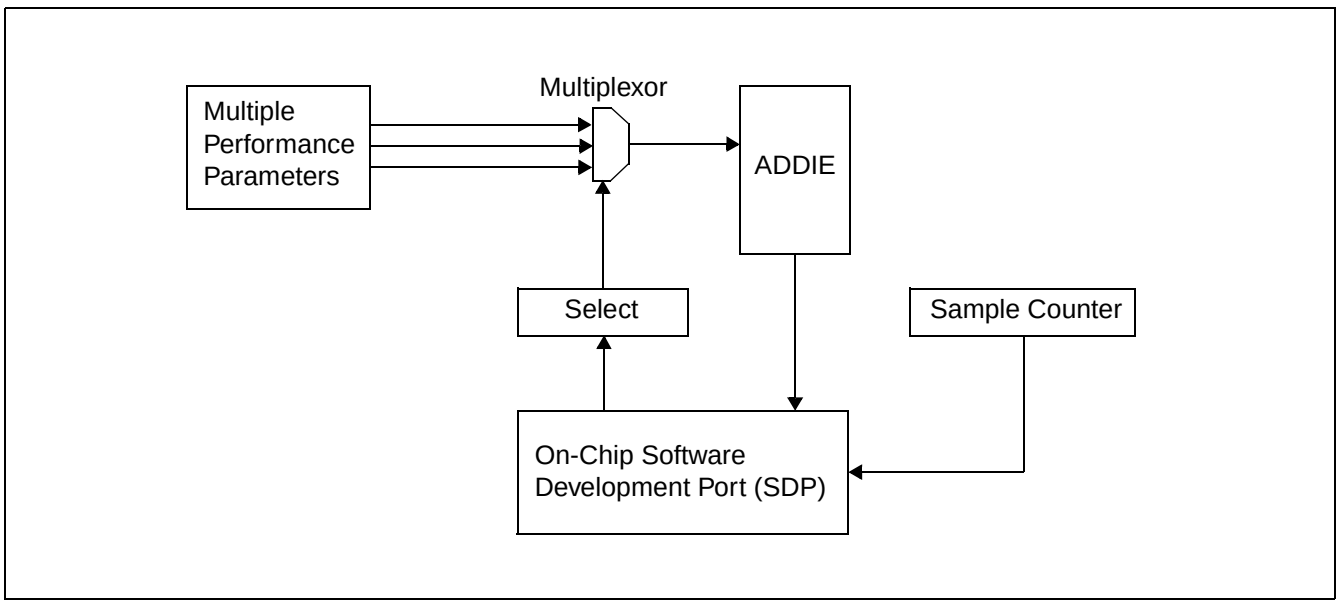

Figure 8. System Block Diagram

The register-controlling parameter selection and the ADDIE counter should be accessible by the processor; they might be mapped into a register, I/O, or memory address space. Additionally, these registers should be accessible from the on-chip software development port (SDP). Via the SDP, a host computer can connect with a target system and unobtrusively examine performance parameters. An instrument is not required for the target system's application or operating system software, and an interrupt handler is not required.

After setting the select register, an 8-bit ADDIE requires that about 500 or more samples be gathered before the ADDIE is assumed to have tracked the new input data stream. To assist with this, a sample counter is used and is reset when the register selecting parameter input is updated. The sample counter is examined via the SDP before accessing the ADDIE counter.

SDRAM Performance Monitor Registers

The synchronous DRAM controller, which includes the write buffer and read buffer, contains two performance monitor resources. These performance monitors are designed using an adaptive digital element (ADDIE). Any of the following ADDIE monitors can be used to monitor the following parameters: SDRAM page misses, write buffer hits, write buffer full, write buffer read merge hits and read buffer hits.

Table 2 shows the performance monitor registers.

Table 2. Performance Monitor Registers with MMCR Offsets

Register Name	MMCR Offset
Performance Monitor Control	44h
Performance Monitor Data 0	48h
Performance Monitor Data 1	49h

Performance Monitor Control Register

MMCR Offset 44h
Memory-Mapped

(Preserved on Programmable Reset)

	7	6	5	4	3	2	1	0
Bit	Reserved	ADDIE1_SEL			Reserved	ADDIE0_SEL		
Reset	0	0	0	0	0	0	0	0
R/W	RSV	R/W			RSV	R/W		

Register Description

This register selects the source to monitor for each of the two ADDIE machines.

Bit Definitions

Bit	Name	Function
7	Reserved	**Reserved** This field should be written to a 0 for normal system operation.
6–4	ADDIE1_SEL	**ADDIE 1 Selector** These bits select the source to monitor using ADDIE 1 000 = ADDIE 1 disabled 001 = SDRAM page-bank misses 010 = Write buffer hits 011 = Write buffer read merge hits 100 = Write buffer full 101 = Read buffer hits 110 = Reserved 111 = Reserved
3	Reserved	**Reserved** This field should be written to a 0 for normal system operation.
2–0	ADDIE0_SEL	**ADDIE 0 Selector** These bits select the source to monitor using ADDIE 0 000 = ADDIE 0 disabled 001 = SDRAM page-bank misses 010 = Write buffer hits 011 = Write buffer read merge hits 100 = Write buffer full 101 = Read buffer hits 110 = Reserved 111 = Reserved

Performance Monitor Data 0 Register

MMCR Offset 48h
Memory-Mapped

(*Not* Preserved on Programmable Reset)

	7	6	5	4	3	2	1	0
Bit	ADDIE0_DATA[7:0]							
Reset	1	0	0	0	0	0	0	0
R/W	R							

Register Description

This register returns the performance monitor data for ADDIE machines 0.

Bit Definitions

Bit	Name	Function
7–0	ADDIE0_DATA	**ADDIE 0 Data** These bits return the ADDIE 0 performance data. This hexadecimal value represents the monitor value. To determine percentages, divide by 255 and multiply by 100. The resulting percentage represents the hit or miss percentage of the parameter being monitored.

Performance Monitor Data 1 Register

MMCR Offset 49h
Memory-Mapped

(*Not* Preserved on Programmable Reset)

	7	6	5	4	3	2	1	0
Bit	ADDIE1_DATA [7:0]							
Reset	1	0	0	0	0	0	0	0
R/W	R							

Register Description

This register returns the performance monitor data for ADDIE machines 1.

Bit Definitions

Bit	Name	Function
7–0	ADDIE1_DATA	**ADDIE 1 Data** These bits return the ADDIE 1 performance data. This hexadecimal value represents the monitor value. To determine percentages, divide by 255 and multiply by 100. The resulting percentage represents the hit or miss percentage of the parameter being monitored.