Vendors of high-speed double-data-rate (DDR) SDRAMs commonly specify their devices by their peak data transfer rate. For example, if a given vendor dubs a device a DDR3-1600, it means the vendor has specified the SDRAM component will transfer data at a peak rate of 1,600 mega-transfers per second.

While the devices can indeed reach their specified transfer rate, in practice they can’t sustain it for real-world workloads. That’s because row-address conflicts, data-bus turnaround penalties and write recovery all can degrade the device’s peak transfer rate.

To make matters worse, the negative impact from these assorted degradations has increased in lockstep with each new generation of faster SDRAM. For this reason, memory controllers employing simple in-order scheduling algorithms will likely achieve sustained throughput that’s substantially below the specified peak. However, more-advanced reordering scheduling techniques can overcome these degrading factors, ensuring that your memory will deliver excellent sustained transfer rates for real-world applications.
With the Virtex®-6, Xilinx® has introduced the first reordering DDR SDRAM memory controller optimized for FPGAs. The new controller enables Virtex-6 users to capitalize on the improved capacity, performance and power efficiency of the latest-generation DDR SDRAM technology.

On the surface, a DDR SDRAM component is simply a read-write memory. In reality, however, modern DDR SDRAMs are complex devices. DDR SDRAM controllers must generate very precise sequences of addresses, commands and data while observing a myriad of timing requirements. High performance requires command pipelining at the minimum allowed timings.

**Ins and Outs of DDR SDRAM**

What is the nature of the degradations that affect memory performance, and why does increasing the peak transfer rate exacerbate their impact?

As illustrated in Figure 1, a basic DDR SDRAM access has the memory controller sending a row address with the activate command to the memory, waiting for the RAS-to-CAS delay interval (defined as the number of clock cycles between the row and column address strobes) and then sending the column address and either a read or a write command. Completing the request requires waiting for the CAS-latency interval and then sampling data for reads or supplying data for writes. When the data transfer is complete, the controller issues a precharge command to close the active row. Following the precharge interval, the controller may issue another activate command.

After that, it may issue multiple read or write commands without a precharge-activate sequence. This is commonly referred to as fast-page-mode access.

Fast page mode is very efficient, since it avoids the time-consuming and power-hungry activate-precharge sequence, but the accesses must be to the same row address. If the workload contains a pattern of accesses to different row addresses, then the activate-precharge sequence must take place between each access. In this case, sustained throughput will be relatively low, very significantly less than the peak transfer rate.

DDR SDRAMs are “banked,” or broken into some number of equal-size, quasi-independent sections. DDR3 DRAMs have eight banks. Accesses to different banks may be overlapped. For example, if the workload contains a read to bank 1 followed immediately by a read to bank 2, the memory controller is allowed to send the row address and activate to bank 1, then the row address and activate to bank 2. After the RAS-to-CAS delay interval, the controller can then send the column address and command for the first read followed by the column address and command for the second read, wait for the CAS latency and then transfer two seamless data bursts. Given a workload that sequentially steps through the banks, this

---

**Figure 1 — Basic DDR DRAM read cycle**

**Figure 2 — Reordering to avoid page conflict**
In addition to overcoming much of the degradation associated with row-address conflicts, a reordering memory controller can also deal with degradation due to write recovery and bus turnaround.

process may be extended in such a way as to sustain the peak transfer rate.

The best way to achieve good DRAM performance is to manipulate the workload such that it falls into these fast-page or rolling-bank modes. Often, however, this is not possible. Processors typically generate quasi-random workloads that are not easily manipulated in this way.

If the workload contains adjacent accesses to different row addresses on the same bank (row-address conflict), followed by an access to a different bank, a simple in-order memory controller will serialize all three accesses, incurring a substantial efficiency penalty due to the precharge-activate for the second access. In contrast, a reordering memory controller is able to send the activate for the first access and then the activate for the third access, overlapping these accesses and thereby improving efficiency.

Figure 2 illustrates this concept. The pattern is shown executed in order and then with the second request moved in front of the third. As you can see, the reordered sequence completes before the in-order sequence.

In addition to overcoming much of the degradation associated with row-address conflicts, a reordering memory controller can also address degradation due to write recovery and bus turnaround. At the end of a write cycle, internally the DRAM is busy actually writing the data into the array. While adjacent write accesses can proceed at peak rate, a write access followed by a read access must wait for the write to complete in the DRAM array. This gives rise to the write-recovery specification.

DDR SDRAMs utilize a bidirectional shared-bus structure. This bus has the controller on one end and typically two to four dual in-line memory modules (DIMMs) on the other end. The electrical length of this bus is about 5 inches. This translates to a bus propagation time of approximately 1 nanosecond. Whenever a bus driver stops driving, it propagates a glitch on the bus. Nominally, two bus propagation times are required for this glitch to settle out.

Built into the DDR SDRAM protocol is a data preamble consisting of one DDR SDRAM clock or two unit intervals. Data is not transferred during the preamble. The preamble can be suppressed if the same driver drives the data for two adjacent accesses. In fact, the preamble must be suppressed if the peak transfer rate is to be achieved.

The performance impact of these write-recovery and data-bus degradations can be significant. A simple in-order memory controller is at the mercy of the workload. Workloads with alternating read-write patterns will exhibit very substantially less than peak transfer rates. But a reordering memory controller is able to group accesses based on the bus driver. In a simple single-rank configuration, it groups reads and writes together and issues them in bursts. In a multirank
configuration, it may organize accesses such that reads for each rank are grouped and issued in a burst.

Figure 3 illustrates the impact of request reordering to avoid bus-turn-around and write-recovery penalties. In this example, sequential reads to bank 0 are interleaved with a sequential stream of writes to bank 1. Two DMA machines reading and writing to memory could very easily generate this workload.

The in-order case loses considerable bus bandwidth with each changeover from read to write and write to read. The reordered case can avoid these penalties to attain significantly higher throughput.

Transaction reordering always raises the issues of data corruption and starvation. Promoting a write in front of a read to the same address corrupts the contents of a memory and must be prevented. Endlessly deferring a read access leads to starvation and must also be avoided.

Reordered data returning from the memory controller may be an issue for some applications. It’s easy to rectify this problem by fitting the memory controller with a reorder-back-to-issue order buffer.

For some corner cases, reordering may increase latency for some read accesses. However, since reordering improves efficiency and therefore reduces memory controller occupancy, the result is to improve average read latency.

Table 1 illustrates the progression of DDR SDRAM technology. Some interesting specifications are provided for DDR1, DDR2 and DDR3. The speed grades chosen are intended to represent midlife, high-volume production.

The transfer rates are increasing geometrically, while the core device characteristics such as RAS cycle are improving at a much slower rate or, in the case of write recovery, not at all. The relative penalty normalized to unit interval increases dramatically with each generation.

Other DDR SDRAM parameters have similar scaling issues. The above three were chosen for illustrative purposes.

**Table 1** – DDR SDRAM transfer rates are increasing geometrically; core device characteristics are not.

<table>
<thead>
<tr>
<th></th>
<th>DDR1-6</th>
<th>DDR2-25</th>
<th>DDR3-1SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak transfer rate (Megatransfers/s)</td>
<td>333</td>
<td>667</td>
<td>1333</td>
</tr>
<tr>
<td>Unit intervals (nanoseconds)</td>
<td>3</td>
<td>1.5</td>
<td>0.75</td>
</tr>
<tr>
<td>RAS cycle (ns/ULs)</td>
<td>60</td>
<td>20</td>
<td>55</td>
</tr>
<tr>
<td>Write recovery (ns/ULs)</td>
<td>15</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>2×BUS_PROP (ns/ULs)</td>
<td>2.0</td>
<td>0.67</td>
<td>2.0</td>
</tr>
</tbody>
</table>

**Table 2** – In two example workloads, memory controller efficiency improves when reordering is turned on.

<table>
<thead>
<tr>
<th></th>
<th>In-order</th>
<th>Reordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU-DMA streams</td>
<td>36%</td>
<td>76%</td>
</tr>
<tr>
<td>Alternating reads/writes</td>
<td>25%</td>
<td>63%</td>
</tr>
</tbody>
</table>

In-Order and Reordering Modes

Let’s look at two example workloads that will demonstrate the capability of the new Virtex-6 DDR SDRAM memory controller. Because this device is able to operate in both in-order and reordering modes, the impact of the reordering is readily visible.

The first workload models two masters. One is a CPU executing a code sequence that is stepping through a matrix. The “stride” of this stepping through the matrix generates a memory request pattern that targets a single bank, but increments the row address with each request. This so-called “unfortunate stride” creates a stream of read requests to a single DRAM bank, but different row addresses. The other master is a DMA engine generating a simple sequential stream of read accesses to a different DRAM bank.

The in-order and reordering cases will actually execute a different overall stream of requests. The row cycle time of the DRAM limits the CPU stream completion rate. Meanwhile, the DMA workload can proceed independently. For the in-order case, the DMA stream becomes serialized with the CPU stream. For the reordering case, the DMA stream fills in the otherwise unused DQ bus cycles with useful work. Since the throughput of the CPU stream is limited by the DRAM row cycle time, it won’t improve much. However, the DMA stream will execute at a higher rate when reordering is enabled.

The second workload models two DMA engines, one generating a sequential stream of reads, the other a sequential stream of writes. It is assumed that the memory controller receives the read and write requests on an alternating basis. To avoid other conflicts, all reads target bank 0, row 0, and all writes target bank 1, row 0.

We ran each workload with reordering turned off, then reran them with reordering turned on. We used default memory controller parameters, with the exception of turning off refresh, ZQ and other DRAM maintenance functions.

Table 2 summarizes the efficiency, measured over a period of steady-state operation, for the two example workloads. We computed efficiency as the number of unit intervals containing payload divided by the total number of unit intervals in the period. When reordering is turned on, the memory controller’s efficiency improves significantly.

The trend of nonscaling DDR SDRAM parameters is expected to intensify as peak data rates continue to increase. Future memory controllers will need to implement continuously improving scheduling algorithms to extract the performance potential of future DDR SDRAM generations.