Low-latency SDRAM Controller for Shared Memory in MPSoC
Pei-Jun Ma, Jia-Liang Zhao *, Kang Li, Ling-Fang Zhu, Jiang-Yi Shi
Key Laboratory of Wide Band-gap Semiconductor Materials and Devices of Ministry of Education,
School of Microelectronics, Xidian University, Xi’an 710071, China
* achao86@qq.com

Abstract
In a memory structure shared by multiple processors based on Multiprocessor Systems on Chip (MPSoC), the efficiency of memory bus access becomes the bottleneck of the overall system efficiency. This paper presents a low-latency SDRAM controller structure integrated in MPSoC, which controls the off-chip SDRAM memory. Consecutive same row optimization and odd-even bank optimization are used to eliminate precharge time and active to read/write execution in memory access. Burst mode supported by data transmit block improves the efficiency of the memory bus. Simulation results show that memory performance improves maximally by 56% compared to pre-optimized, making it meet the high throughput requirements of shared-memory controller in MPSoC.

1. Introduction
Since lots of difficulties lie in the integration of on-chip memory larger than 4M with the current Microelectronic Technique, it is more likely to choose high-capacity off-chip DRAM, which is controlled by the on-chip memory controller. For the case of MPSoC, especially high throughout network processor, the efficiency of memory controller is a key factor in the system performance.

This paper made in-depth analysis of memory access in different situations, and presents a new controller structure of shared SDRAM in MPSoC. By using of consecutive same row Optimization, memory avoids duplicate row activation and precharge operations. With the further use of Odd-even Optimization, precharge time is hided so as to speed up the memory access. High throughput performance of memory access is obtained.

2. SDRAM controller
2.1 Structure of SDRAM controller
The main modules within the SDRAM controller, as show in Figure 1, include reference queues, arbitrator, instruction decoder and address generator, burst mode data transmit module, data-command interface logic, mode registers, etc.

Mode registers configure different capacities of SDRAM. All the SDRAM parameter values are also stored in the registers. After initialization, references are classified and pushed into different reference queues. Using the hierarchical priority arbitration algorithm, the arbiter grants different processor its own priority. Once a reference is arbitrated out of a queue, it is decoded and address to be accessed is generated. Then the commands and the address, together with data, are transmitted to SDRAM interface logic. Data path adopts burst mode data transmit mechanism based on instruction control. It means that one reference can access multiple memory addresses, avoiding time consuming through frequent read or write operations.

2.2 Design of FSM
Figure 2 illustrates the timing of SDRAM single read without auto precharge (Page Miss). tRCD is active to read or write execution; tCL is read access time from clock; tRP is precharge command period; and tRAS is active to precharge command period.
If controller continuously read n quadwords, and tBL is the execution of burst access, then the total execution time before the data is read out is,

$$t_{\text{DELAY}} = n \times (t_{AP} + t_{RCD} + t_{CL} + t_{CL})$$  \hspace{1cm} (1)$$

If the bank to be addressed is in idle state (Page Hit), row activation command could be sent out directly, and then, column access command. Before data is read out, the execution time is \((t_{RCD} + t_{CL})\).

If the row to be addressed has already been activated (Page Fast Hit), which means command can be sent out directly to read data. The execution time is \(t_{RCD}\).

According to analysis above, this paper improves the FSM so as to support all these three access cases, as shown in Figure 3.

In Page Miss case, the FSM state’s jumping direction is from idle to row access, tRCD execution, column access, precharge, and back to idle state. While in the Page Fast Hit case, row access and tRCD execution and precharge states are omitted; as in Page Hit case, precharge operation is eliminated.

### 2.3 Consecutive same row optimization

Figure 4 roughly shows the timing of four read accesses when consecutive same row optimization is adopted. As you can see, the row activation and precharge operations of the last three read accesses are omitted.

The arbiter and interface logic cooperate to implement this function. When the arbiter selects the following instruction, the arbiter also compares the address of the forthcoming instruction and the previous address which has been stored. If these two addresses are in the same row within the same bank, it means the row which is about to be accessed is activated. And the consecutive same row optimization enable signal is set valid. Under the influence of this enable signal, FSM will keep the state jumping among the following column accesses until there is an instruction not accessing the same row. Then the state will jump to precharge state so as to close the page. Figure 3 shows the details.

Take continuously executing \(n\) read instructions as an example, after consecutive same row optimization is adopted, the total execution time is,

$$t_{\text{DELAY}} = (t_{RCD} + t_{AP}) + n \times (t_{CL} + t_{CL})$$  \hspace{1cm} (2)$$

Compared to equation (1), there is no doubt that this optimization can save much time, by \((n-1) \times (t_{RCD} + t_{AP})\) time exactly.

### 2.4 Odd-even bank optimization

What if the instructions are not continuous access to the same row within the same Bank? In order to improve memory access efficiency, a more proactive optimization method is required. Figure 5 briefly illustrate the timing of reading 3 quadwords from bank 1 and writing 2 quadwords to bank 2 when using odd-even bank optimization.

It seems in Figure 5 that the precharge time of bank 1 is eliminated. In fact, when column access command is issued in the first read instruction access bank 1, the controller automatically signal an auto precharge. When the following implementation of write access is going on in bank 2, bank 1 is precharged automatically.
bank and even bank by turns. Meanwhile, odd-even bank optimization enable signal is set valid. Under the influence of this signal, FSM will jump to the next row access after column access, instead of jumping to precharge state, as shown in Figure 5. And with odd-even bank optimization, the total execution time of continuously “n” read instruction is,

$$t_{\text{delay}} = t_{\text{pre}} + n \times (t_{\text{RCD}} + t_{\text{CL}} + t_{\text{RAS}})$$  \hspace{1cm} (3)

Compared to equation (1) again, we can see considerable amount of time is saved after optimization, by \((n-1) \times t_{\text{pre}}\) exactly.

3. Verification and Analysis

This paper provides RTL description of the design in Verilog HDL and functional verification in Modelsim6.5 environment, using Micron SDRAM model MT48LCM4A2. The simulation waveforms are shown in Figure 6 and Figure 7.

![Figure 6. Timing waveforms of reading 8 quadwords with consecutive same row optimization](image1)

![Figure 7. Timing waveforms of writing 8 quadwords with odd-even bank optimization](image2)

Statistics is collected to make a comparison of the performance before and after these two optimizations are adopted. The execution time of writing one to four quadwords, is listed separately in Table 1 and Table 2. The unit of the statistics is the clock period of off-chip SDRAM memory.

### Table 1. Performance Improvement with consecutive same row optimization

<table>
<thead>
<tr>
<th>Case</th>
<th>Ref.</th>
<th>Pre-optimized</th>
<th>Optimized</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write (1 qw)</td>
<td>16</td>
<td>7</td>
<td></td>
<td>56.25%</td>
</tr>
<tr>
<td>Write (2 qw)</td>
<td>17</td>
<td>8</td>
<td></td>
<td>52.94%</td>
</tr>
<tr>
<td>Write (3 qw)</td>
<td>18</td>
<td>9</td>
<td></td>
<td>50.00%</td>
</tr>
<tr>
<td>Write (4 qw)</td>
<td>19</td>
<td>10</td>
<td></td>
<td>47.37%</td>
</tr>
</tbody>
</table>

### Table 2. Performance Improvement with odd-even bank optimization

<table>
<thead>
<tr>
<th>Case</th>
<th>Ref.</th>
<th>Pre-optimized</th>
<th>Optimized</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write (1 qw)</td>
<td>12</td>
<td>6</td>
<td></td>
<td>50.00%</td>
</tr>
<tr>
<td>Write (2 qw)</td>
<td>13</td>
<td>7</td>
<td></td>
<td>46.15%</td>
</tr>
<tr>
<td>Write (3 qw)</td>
<td>14</td>
<td>8</td>
<td></td>
<td>42.86%</td>
</tr>
<tr>
<td>Write (4 qw)</td>
<td>15</td>
<td>9</td>
<td></td>
<td>40.00%</td>
</tr>
</tbody>
</table>

With current design of SDRAM controller, consecutive same row optimization improves the overall efficiency by 50% or even more. While the odd-even bank optimization method improves the performance by 40% to 50%. These two optimization results show fairly obvious performance enhancement, which fully achieves the desired target.

4. Conclusion

This paper has done in-depth study of memory access to shared memory in MPSoC, and presents a SDRAM controller structure. Multi-processor parallel access to memory is achieved with the use of multiple reference queues and arbiter. And by using consecutive same row optimization and odd-even bank optimization, high performance of memory access is easily gained. The design can be applied in a variety of areas where high throughput data processing is required, such as networking, communications, video and image processing. Surely it has a very broad application prospects.

Acknowledgments

Support for this work was provided by The National Natural Science Foundation of China (60506020) and Natural Science Foundation of Shannxi Province of China (SJ08-ZT13).

References