100% non-stop system availability has emerged as a critical requirement for server systems handling an increasing number of real time transaction processing functions, from finance to e-commerce. Since these systems typically contain large amounts of memory, particular attention has been paid to ensure the integrity of the data held in the DRAM array. System architectures and algorithms have been developed that can detect and correct multiple bit errors such that a system can continue to operate even if an entire DRAM device fails. This capability is known in industry parlance as “chipkill”. This paper will explore an example chipkill system using today’s SDRAM memory devices and how this architecture can be improved using Rambus® DRAMs.

**Traditional Chipkill in SDRAM Systems**

Traditional Hamming error correction code (ECC) has been in widespread use for many years. This approach involves attaching a number of checksum or syndrome bits along with the corresponding data as it is being written to memory. On reads, the controller again generates syndrome bits based on the received data pattern and compares it against the syndrome bits stored from the write operation. Depending on the result, bit errors in the data or syndrome bits can be detected and/or corrected. The number of syndrome bits needed increases with the size of the data word and the number of error bits that must be detected and corrected in one data word. In most ECC based systems 64 data bits are used along with 8 additional syndrome bits, resulting in a total word size of 72 bits. Rambus DRAM supports the same ECC approach using the x18 device. Internally, the x18 RDRAM® operates as a 144 bit data path, 128 bits of which can be used for data, the remaining 16 bits can be used for syndrome (9 are needed for ECC) or other functions.

Unfortunately, traditional ECC can only detect double bit errors and correct single bit errors across the data word. Multibit errors contained in the data word are uncorrectable using this technique.

Chipkill extends the functionality of the ECC bits by architecturally partitioning the memory array. The premise is to spread out the ECC-enhanced data word across many DRAMS such that any individual DRAM contributes only one bit. Conventional ECC can thus be applied even if an entire DRAM fails. The primary drawback with this approach is that the system requires a minimum of 72 DRAMs, in the case of the 72 bit ECC-enhanced word. However, this assumes x1 DRAMs which are no longer available. Using the more widely available x4 DRAM configuration results in increasing the ECC word size and number of ECC checkers by a corresponding multiple to a total of 288 data bits and four ECC checkers respectively (Figure 1). Each DRAM contributes 1 ECC bit to one of the 4 ECC checkers. There are a number of drawbacks with this system implementation: the minimum data transfer size is 32 bytes from a memory controller with approximately 500 pins, including data, address/control, power and ground. Assuming 100 Mhz SDRAMs (which may not be available in such narrow configurations and low densities), peak bandwidth is 3.2 GB/s; however useable bandwidth is typically much less since the SDRAM array effectively has four pages, resulting in a large percentage of stalled memory requests due to bank conflicts. To overcome this limitation, system designers have doubled or quadrupled the number of memory ranks. While sustained bandwidth rates increase with this approach, the minimum memory size and upgrade granularity increase a corresponding two or four times, to 144 or 288 DRAM devices respectively. For a 72 DRAM implementation, RAS must be activated simultaneously for all 72 DRAMs, resulting in large power consumption.

**Figure 1: Chipkill Implementation using x4 DRAMs**

---

Rambus®: Protecting High-Availability Systems
Rambus DRAM Feature: Interleaved Data Mode

Implementing a traditional chipkill technique for a Rambus DRAM based system is feasible but complicated by the fact that a single RDRAM returns 16 bytes of data (one dualoct) across all its data pins. To spread out the ECC syndrome bits across the required number of RDRAMs involves a combination of many Channels operating in parallel and/or multiple requests to several RDRAMs on any given Channel. The trade-off for chipkill support is reduced memory bandwidth and/or added memory latency. To overcome these issues, Rambus has implemented a new feature to be included in the 256Mbit memory generation and beyond. This feature is completely transparent and compatible with existing RDRAMs and memory controller implementations. Termed Interleaved Data Mode (IDM), this function enables chipkill detection and correction using a similar approach as today’s SDRAM based systems with no loss in memory bandwidth or increased latency.

Conceptually, IDM enables a group of RDRAMs on a single Channel to respond to a single request. This is similar to a wide SDRAM datapath employing multiple devices acting in parallel. Each RDRAM in the group receives and transmits information on unique data pin(s) for the entire CAS cycle. So instead of a single RDRAM returning all 16 bytes, multiple RDRAMs in parallel transmit/receive data on separate pins to make up the dualoct. Like the SDRAM approach, this solution enables the ECC word to be distributed efficiently across multiple devices while still maintaining a peak memory bandwidth of 1.6 GB/s per Channel with no added latency. In addition, when IDM is not enabled, the devices function as normal RDRAMs.

IDM is enabled during initialization by a register bit setting. When enabled, eight RDRAM devices on a Channel will respond in parallel to row and column requests. Each of eight RDRAMs will transmit and receive information on unique data pins, determined by a mapping function of the device identification field in the row and column packets.

The address to pin mapping is handled by the RDRAM, hence the row and column packet formats themselves are unchanged in IDM. With the x16 RDRAMs, each device will receive/transmit two data bytes across two unique pins (one from DQA, the other from DQB). With the x18 RDRAMs, six devices will receive/transmit two data bytes across two unique pins, while the other two devices will receive/transmit three bytes across three pins. Levelization logic in the RDRAM assures that all data bits transmitted by the 8 RDRAMs are received by the controller at the same time. (see Figure 2 below). Byte masking is not supported in IDM.

Chipkill for Rambus DRAM Systems using IDM

Let’s examine the same chipkill enabled SDRAM system with a comparable x16 RDRAM system using IDM. Although 72 RDRAM devices are still required, there are several system implementation choices to choose from. The low cost method would involve 3 Rambus Channels, each with 24 RDRAMs. This would yield a peak bandwidth of 4.8 GB/s and consume ~210 pins on the memory controller. This is 50% more peak bandwidth and less than half the controller pins versus a 100MHz SDRAM solution. Sustained bandwidth for the Rambus solution will be greater than that due to the much larger number of available banks (32 versus 4 for SDRAM) and hence the less likelihood of bank conflict. Alternatively, if performance or memory foot-

Figure 2: IDM mode. Logical pin mapping for 8 RDRAM parallel operation. Each device transmits on 2 unique pins
(For x18 device, 2 devices transmit on 3 pins)
print was a primary concern, a 9 Rambus Channel controller can be designed (Figure 4), yielding 12.8 GB/s peak bandwidth in 600 pins, 100 pins more than the 72-device SDRAM solution while providing 3 times the bandwidth. This solution has the added benefit of enabling detection and correction of stuck at pin faults, by distributing the error code across multiple Channels, as well as having a potential maximum memory footprint of 9GB. In both examples, the minimum data transfer size is 128 bytes, since each RDRAM in IDM is effectively transferring two bytes of data.

As an added benefit, because the same RDRAM can be used in non chipkill ECC based systems as well as chipkill enabled systems, Rambus technology is the only memory solution that enables a single, scalable memory architecture design for high reliability systems. Lower end server/high end desktop systems can be chipkill “ready”, supporting ECC initially and enabling the chipkill option when a sufficient number of RDRAMs are added. This cannot be done with SDRAMs today as lower end systems use the x8 and x16 organization for minimum memory granularity, while chipkill requires using x4 devices along with a 500+ pin memory interface on the controller.

Conclusion

With IDM, system companies can implement a chipkill enabled RDRAM system using the same techniques as today’s SDRAM based solution without sacrificing Rambus technology’s inherent advantages over SDRAM in terms of pin count and performance. Moreover, IDM enables lower end systems to be chipkill “ready” since the same RDRAM and memory system architecture is used in either case. Only Rambus memory technology offers a scalable approach to DRAM reliability in a system operating environment that must be available 24 hours a day, 7 days a week and 365 days a year.