XAPP657 (v1.0) August 15, 2002

Virtex-II Pro RAID-5 Parity and Data Regeneration Controller

Author: Steve Trynosky

Summary

Redundant Array of Independent Disks (RAID) is an acronym first used in a 1988 paper by University of California Berkeley researchers Patterson, Gibson, and Katz[1]. A RAID array is a disk array where part of the physical storage capacity is used to store redundant information about user data stored on the remainder of the storage capacity. The redundant information enables regeneration of user data in the event that one of the array's member disks or the access path to it fails.

Data regeneration is an important function in RAID controllers and is best performed by dedicated hardware under the control of a microprocessor. The Virtex-II Pro™ FPGA can perform both the hardware and software functions required for a RAID parity generator and data regeneration controller. This reference design uses burst mode SYNCBURST™ SRAM memory accesses and an internal block SelectRAM+™ memory to provide an extremely efficient hardware design in a Virtex-II Pro FPGA.

Introduction

RAID provides a method of accessing multiple individual disks as if the array was one large disk. Multiple levels of RAID architecture are defined, each with different cost, performance, and availability profiles. The reference design is targeted at RAID-5 architectures, however it can be used in any system using parity information for data regeneration. Figure 1 shows a RAID-5 array system. From the perspective of the host computer, the storage system appears as one large disk. From the perspective of the disk array controller, the storage system contains six individual disks. A RAID disk system can have any number of disks in the array, between two and eight disks are typical. RAID-5 controllers can service multiple I/O requests independently. A RAID-5 system can withstand a single disk failure without losing data or access to the data.

Array management software is responsible for logical to physical mapping of user data onto the storage medium. The array management software also controls the data striping algorithm and parity management as new data is written to the disk, or files are deleted from the disk. Data regeneration, in case of a disk failure, is controlled by the application software.

Figure 1 shows how data and parity are horizontally stripped across a RAID-5 storage array. The figure depicts a single array of disks. A storage system can have more than one array of disks. The depth of the data stripe is controlled by the array management software and can be tuned for system performance needs. Each data element, D(n) or P, is a logical block of data where the host application software controls the size. Parity information, for each stripe, "P" is updated each time one of the horizontal data elements is updated on the disk. The parity information rotates around to prevent a single drive from becoming the bottleneck for other disk I/O accesses.

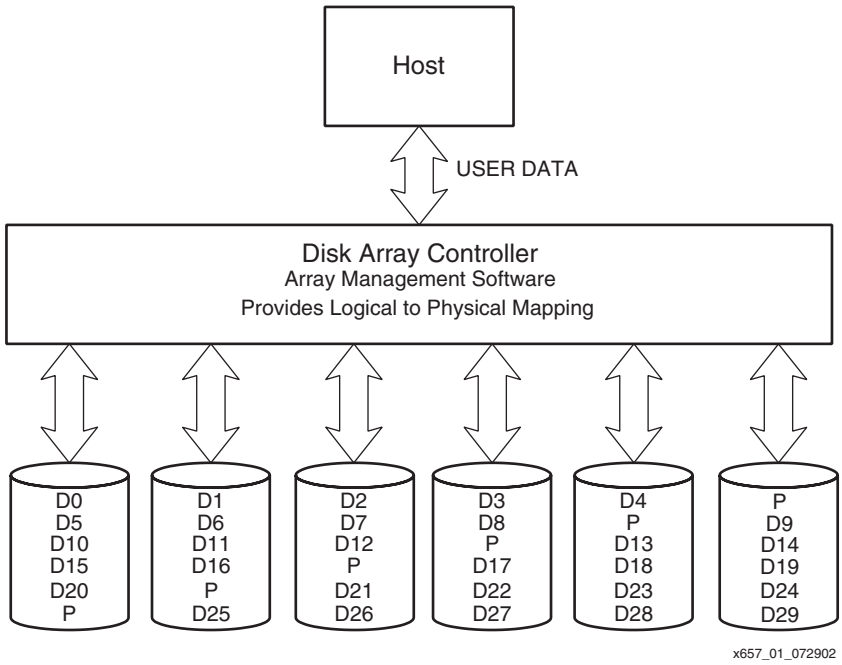

Figure 1: RAID-5 Data and Parity Striping

Write performance of RAID-5 is poor. Each write requires four I/O accesses, two reads, and two writes. First old data and parity are read from separate disks into the storage controller memory. New parity is calculated by the controller hardware using old data, old parity, and new data information. Once data parity has been calculated, the new data and new parity information is written to separate disks. The "write penalty" associated with RAID-5 is minimized by performing parity management as a background operation and utilizing resources such as cache memory, multiple data paths, and providing multiple microprocessors to handle the data traffic.

Reference Design

This application note concentrates on the hardware design elements required to implement the parity drive information and data regeneration hardware. The disk controller and storage array software is not part of this application note. The storage array software and disk controller hardware are responsible for moving data and parity information to/from a memory buffer to the RAID-5 storage array. The Virtex-II Pro FPGA contains an IBM® PowerPC® 405 microprocessor block to execute storage array management software.

A block diagram of the reference design is shown in Figure 2. The reference design includes an interface to an external SSRAM memory for user data and parity information storage. The SSRAM memory organization is 512K x 32 bits. The design does not include a disk controller to SSRAM memory interface. The design assumes the external SSRAM memory contains the user data required to either generate parity or to regenerate the user data from other user data elements and parity information

The Virtex-II Pro FPGA design contains an external SSRAM memory controller, a dual port block RAM controller, a 2K x 32 bit block RAM with an exclusive OR logic function, control state machines, and a bidirectional to unidirectional data bus conversion module.

The primary functions of the reference design are to provide hardware to:

- Calculate new parity as new data, old data, and old parity arrives at the controller memory
- Regenerate data in case of a disk failure in a storage array
- Control an external SSRAM memory buffer

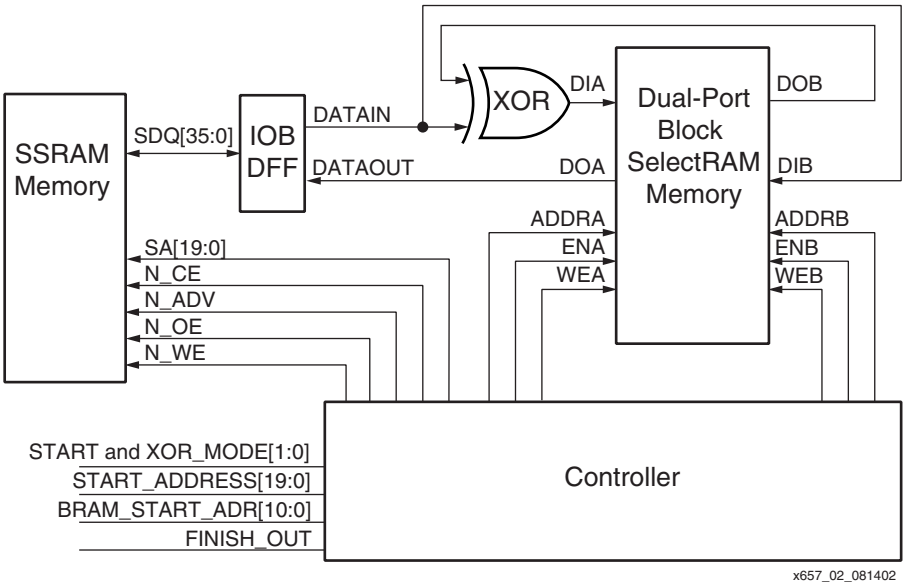

Figure 2: **RAID-5 Data/Parity Controller Block Diagram**

The controller hardware calculates new parity or regenerated data using data contained in the SSRAM memory. A sequence of memory accesses and exclusive OR operations are performed, using the block RAM as a temporary storage buffer. The final operation moves the parity information or regenerated data from the block RAM into the SSRAM to complete the function.

The following formula shows how to calculate NEW_PARITY, for data element D3.

- $NEW\_PARITY = OLD\_DATA_{(D3)} \oplus OLD\_P \oplus NEW\_DATA_{(D3)}$

The second formula shows how to calculate REGENERATED_DATA, for data element D4, assuming the storage array has six disks per array:

- $REGENERATED_{(D4)} = D0 \oplus D1 \oplus D2 \oplus D3 \oplus PARITY$

The design uses burst-memory cycles to/from the SSRAM utilizing the internal dual port block RAM in the Virtex-II Pro FPGA for temporary storage. Each SSRAM memory buffer location is accessed only one time. This makes the absolute minimum number of memory cycles required to perform parity generation or data regeneration.

The reference design assumes all data blocks (Figure 1) are 512-bytes deep. Each data transfer from SSRAM to/from the block RAM is 512 bytes. One-hundred and twenty-eight memory cycles are required, per data block, to complete a burst cycle. The controller performs either a burst read or a burst write, to the SSRAM, depending upon XOR_MODE control inputs to the design.

Taking the regeneration calculation above, the design performs a series of memory operations as follows:

1. Read SSRAM D0 block into the block RAM without any XOR data operation.
2. Read the SSRAM D1 block and the block RAM (D0), perform a 32-bit XOR and write the result into the other port of the block RAM element. The block RAM now contains D0 ⊕ D1.
3. Read the SSRAM D2 block and the block RAM (D0 ⊕ D1), perform a 32-bit XOR and write the result into the other port of the block RAM element. The block RAM now contains D0 ⊕ D1 ⊕ D2.
4. Read the SSRAM D3 block and the block RAM (D0 ⊕ D1 ⊕ D2), perform a 32-bit XOR and write the result into the other port of the block RAM element. The block RAM now contains D0 ⊕ D1 ⊕ D2 ⊕ D3.

5. Read the SSRAM P block and the block RAM (D0 ⊕ D1 ⊕ D2 ⊕ D3), perform a 32-bit XOR and write the result into the other port of the block RAM element. The block RAM now contains the regenerated D4 (D0 ⊕ D1 ⊕ D2 ⊕ D3 ⊕ P).
6. Transfer the regenerated D4 from block RAM into the SSRAM memory.

As can be seen from the sequence of operations detailed above, and referring to the data path between the SSRAM and block RAM in Figure 2, all four ports of the block RAM are utilized at different times. In step #1, the data read from the SSRAM is written directly into the DIB port of the block RAM. In steps #2 through #5, the SSRAM data is exclusive OR'ed with the contents of the DOB port and the result is written into the DIA port of the block RAM. In step #6 the DOA port is read and written directly into the SSRAM memory. The reference design controls hardware functions for both the SSRAM and block RAM through the two user interface inputs, XOR_MODE[1:0]. See **User Interface** section for more information.

The reference design utilizes burst mode memory cycles to minimize the time required to perform RAID-5 parity and data recovery calculations, thus permitting the maximum number of parity or data regeneration calculations per second. Burst mode uses a combination of external address and internal address of the SSRAM memory. A four-word burst can be performed between successive external address commands. The block RAM in the FPGA permits burst mode memory accesses with exclusive OR operations to be performed on the data elements. Traditional "external memory only" designs have two drawbacks:

- The memory controller must use inefficient, single address, non burst-mode memory accesses to retrieve individual elements of the user data buffers.
- The parity or data regeneration calculation can not begin until all data and parity elements are contained in the SSRAM memory buffer.

Consider the previously discussed data regeneration calculation for D4. Assume each 512-byte data and parity block is stored in the SSRAM at the locations defined in Table 1. A design without an internal block RAM component would replace the memory with a 32-bit "D" flip-flop (DFF) to store intermediate calculations, as shown in Figure 3. The SSRAM memory controller has to perform the following memory accesses:

For n = 0 to 127

1. Read data element D0(n) at location nx0, and store in an internal 32-bit DFF.
2. Read data element D1(n) at location nx80, XOR with DFF contents and store in DFF.
3. Read data element D2(n) at location nx100, XOR with DFF contents and store in DFF.
4. Read data element D3(n) at location nx180, XOR with DFF contents and store in DFF.
5. Read data element P(n) at location nx200, XOR with DFF contents and store in DFF.
6. Write regenerated data D4(n) (32-bit DFF contents) to SSRAM at location nx280. Next n.

Table 1: **User Data Storage Addresses**

User Data Elements (512 bytes per)	First Data Element SSRAM Address (Hex)	Last Data Element SSRAM Address (Hex)
D0	0	7F
D1	80	FF
D2	100	17F
D3	180	1FF
P	200	27F
Regenerated D4	280	2FF

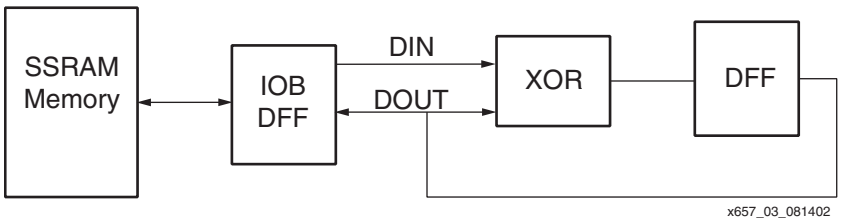

Figure 3: **Data Path Without Xilinx Block RAM Element**

Obviously, the parity or regeneration calculation can not start until all elements are in the SSRAM. Burst-mode memory accesses are not used in an "external memory-only design", as each calculation must select an element from each user data element to exclusive OR with another data element not in an adjacent memory location. This complicates the design of the address path for the memory controller. A SSRAM read operation requires two clocks to access data. The address is registered on the first clock and the data is output on the second clock. The performance advantages of pipelining reads can not be utilized for this type of design.

These limitations are overcome with this reference design. Using burst-mode read cycles, the one-clock cycle latency can be pipelined in a burst cycle where a 128-word read burst requires only 129 clocks to complete. Internal block RAM permits a calculation to be suspended if waiting for data, or to begin a calculation before all the data and parity elements arrive from the disk system to the SSRAM memory. Since the data blocks contain only 512-bytes and the block RAM is 8K-bytes, sixteen independent operations can be in different stages of completion at one time. Or the data stripe depth could be increased, to say 8K bytes.

Port Descriptions

The reference design provides an interface to an external SSRAM memory. User interface inputs and outputs are used by the testbench to control the parity or data regeneration function. This design is intended to be integrated into a larger system design containing a microprocessor for storage management and data transfer control plus disk controller hardware to transfer data to/from the SSRAM and disk array elements.

SSRAM I/O Signals

The design interfaces a Virtex-II Pro FPGA to a Micron Semiconductor 8 Mb SYNCBURST SRAM device. The Micron Semiconductor part number is MT58L512V18FS-7.5 or equivalent. The 113 MHz SSRAM device timings result in a 100 MHz memory bus speed. The Micron Semiconductor web page http://www.micronsemi.com/ contains a data sheet for the device.

Table 2: **SSRAM I/O Signal Usage**

Port Name	Direction	SSRAM Name	Description
SA[18:0]	Output	SA[18:0]	Synchronous SRAM address
SDQ[35:0]	I/O	DQ[17:0]	Note 6. SSRAM data I/O
SRAM_CLK	Output	CLK	Clock
N_CE	Output	$\overline{CE}$	Chip enable
N_ADV	Output	$\overline{ADV}$	Address advance
N_OE	Output	$\overline{OE}$	Output enable
N_WE	Output	$\overline{BWA}$: $\overline{BWB}$	Note 1. Synchronous byte writes
		$\overline{BWE}$	Note 2. Byte write enable
		$\overline{GW}$	Note 2. Global write
		$\overline{CE2}$: CE2	Note 2. Chip enable
		$\overline{ADSC}$	Note 3. Address status controller
		$\overline{ADSP}$	Note 2. Address status processor
		MODE	Note 4. Linear/interleaved addressing.
		ZZ	Note 5. Low power mode for SSRAM.

Notes:

1. Testbench connects FPGA output N_WE to all four byte write enable SSRAM inputs.
2. Not controlled by FPGA. Testbench asserts/deasserts by connection to V_{DD}/GND level.
3. Testbench connects N_CE to N_ADSC.
4. The design uses linear burst mode.
5. This SSRAM input is deasserted in the testbench. The design does not support SSRAM standby mode.
6. The memory model used in the testbench combines the data and parity bits into an 18-bit bus.

User Interface

The user interface includes a SSRAM start address, a block RAM start address, a XOR function select, a reference clock, start and a FINISH_OUT signal. Bit definitions are provided in Table 3 and Table 4. Refer to the testbench file for proper sequencing of control inputs. The design provides a FINISH_OUT signal to indicate when a 512-byte burst memory operation has completed, Once the testbench has asserted the START signal, it waits for assertion of FINISH_OUT before changing any of the control inputs.

The testbench drives the user interface inputs to the FPGA. In a system environment this reference design would be integrated into a microprocessor controlled system with a DISK controller interface to the SSRAM. The user interface signals would then be controlled by the microprocessor using memory mapped I/O registers. The FINISH_OUT signal could be used as a hardware interrupt mechanism to inform the microprocessor when the hardware has completed the current calculation. The microprocessor would also manage the memory buffer contents and perform multiple RAID-5 data regeneration and parity calculations, potentially using the BRAM_START_ADR to have concurrent operations in various stages of completion.

Table 3: **User Interface Signal Definitions**

Port Name	Direction	Description
BRAM_START_ADR[10:0]	Input	Defines the starting location to use for data/parity block storage within the block RAM buffer. A maximum of sixteen operations can be in progress using one of sixteen available block RAM buffers (using default 512-byte block size).
BUS_REQUEST	Output	Internal signal requesting SSRAM burst memory access.
CLK	Input	Clock input to the Digital Clock Managers (DCM)
FINISH_OUT	Output	Asserted when a burst operation has completed. The testbench waits for FINISH_OUT to be asserted before setting up the next calculation.
START_ADDRESS[19:0]	Input	Defines the starting location of a particular data/parity block within the SSRAM buffer.
SSRAM_CLK	Output	DCM2 clock output to SSRAM clock
SSRAM_CLKFB	Input	Board de-skew clock feedback input to DCM2
START	Input	Start RAID controller operation as defined by XOR_MODE. Once the user interface inputs are specified, assert and deassert the START signal to commence RAID calculation.
XOR_MODE[1:0]	Input	See Table 4 for detailed bit definitions

The user interface to the controller includes two inputs (XOR_MODE[1:0]) for determining the XOR operation performed. Table 4 defines the XOR_MODE bit values and the resulting SSRAM and block RAM function performed by the controller hardware. Refer to Figure 2 to understand the system data flow function controlled by the XOR_MODE inputs.

Table 4: **XOR_MODE Control Bit Definitions**

XOR_MODE[1:0] Binary Value	SSRAM Memory Operation	Block RAM Operation
00	Idle: no operation	Idle: no operation
01	Read SSRAM starting at user interface defined "Start Address"	Write SSRAM data to block RAM (Port B) at user interface defined "BRAM Start Address", without XOR
10	Read SSRAM starting at user interface defined "Start Address"	Read block RAM (Port B) at user interface defined "BRAM Start Address", XOR with SSRAM data, and write result back to block RAM (Port A)
11	Write contents of block RAM to SSRAM starting at user interface defined "Start Address"	Read block RAM (Port A) starting at user interface defined "BRAM Start Address"

Design Elements

The reference design includes a testbench, Verilog and VHDL design files, and a user constraint file. The testbench initializes a small portion of the external SSRAM prior to starting any RAID parity control functions.

TOP Module

This design element provides an interface to the external SSRAM and testbench controls. The user interface provides a means to select the starting address of the external SSRAM and the starting address for the block RAM. The transfer size of the data block is fixed at 512 bytes for this design. The XOR_MODE control inputs determine SSRAM access mode (read/write) and

how the block RAM memory ports are utilized (Port B write only, Port B read XOR and Port A write, or Port A read only). All testbench inputs are registered at the IOB.

ADRCNTR Module

This module provides the memory address and transfer counters used in the design. The counter functions: LOAD, COUNT_UP, and COUNT_DOWN, are controlled by the RAIDCTL state machine.

For the SSRAM memory interface, a 20-bit load-able counter is provided to address the SSRAM. The SSRAM counter is loaded through the user interface signals START_ADDRESS. The data blocks are stored throughout the SSRAM memory buffer, and the user interface selects the locations in memory for the RAID operations.

Each port of the dual-port block RAM has an 11-bit load-able counter. The counters behave differently, depending upon the XOR_MODE[1:0] user interface inputs. The two address counters are loaded through the user interface signals BRAM_START_ADR. The design supports overlapping, but non-concurrent, RAID operations.

An 11-bit transfer counter is used to control the length of any XOR operation. The transfer size is fixed at 128 words (512 bytes). The transfer counter counts down under control of the RAIDCTL state machine.

CLK_RST_STARTUP Module

The CLK_RST_STARTUP module provides a digital clock manager (DCM) and reset circuit for the design. The input clock is buffered for the internal design elements. An internal reset signal is deasserted when the DCM locks on the input reference clock.

A second DCM is used to de-skew the SSRAM_CLK output. The first DCM is used for internal FPGA logic and the second is used to generate the clock to the SSRAM modules.

DATABUS Module

The SSRAM has a 36-bit bidirectional data bus interface. Only 32-bits are used during a read from SSRAM. Write cycles to SSRAM force the "unused" parity bits (35, 26, 17, and 8) to ground. The DATABUS module provides registered I/O with 3-state control in the IOB cells of the FPGA. The registered I/O is critical for meeting bus timings for the reference design. Data read from the SSRAM is registered prior to presenting to the XOR_BRAM module, and data read from the XOR_BRAM module is registered prior to writing to the SSRAM.

RAID_CNTL Module

This state machine controls all of the memory address counters and transfer counters in the ADRCNTR module. The state machine provides a LOAD signal to load all counters in the design. Separate COUNT_UP signals are provide for the SSRAM memory address, the Port A block RAM address, and the Port B block RAM address. A COUNT_DOWN signal is provided for the transfer counter. The XOR_MODE[1:0] user interface inputs control the function provided by the state machine. Three separate control functions are provided in the design to accommodate the parity creation or data regeneration functions required.

The state machine controls the dual-port block RAM. Control signals (enable and write) are provided for each port of the block RAM. The block RAM is configured differently (read versus write) depending on the specific XOR operation performed.

The state machine submits a bus request and read/write request to the SRAMCTL module.

SRAMCTL Module

The SRAMCTL module provides a burst mode only SSRAM memory controller. Burst cycles must consist of either four reads or four writes. All memory writes are 32-bits. Byte writes to the SSRAM are not supported, nor are they required for this design. Multiple bursts of reads or writes can be performed back-to-back to provide highest performance memory access. The

SSRAM controller provides the chip enable, advance, output enable, and write enable signals to the memory.

In order to meet the bus-interface timings at 100 MHz all SSRAM signals are registered (address, data, and control). The data bus signals are registered in the DATABUS module. The address and control signals are registered in the TOP module.

XOR_BRAM Module

The XOR_BRAM module contains a dual port 2K x 32-bit block SelectRAM+ module and a 32-bit XOR function. The design uses a RAMB16_S32_S32 UNISIM component. The Port B data output of the block RAM is logically XOR'ed with the internally registered SSRAM data bus. The output of the 32-bit XOR function is connected to the Port A data input. The address for the two block RAM ports is provided by the ADRCNTR module. The enable and write enable controls are provided by the RAID_CNTL module.

Design Summary

Synplicity, Simplify version 7.03, was used to synthesize the design. The EDIF file was put to the Xilinx ISE4.2I, Service Pack 3, to place and route the reference design. The Virtex-II Pro speeds files were upgraded to version 1.62. The design resources are summarized in Table 5.

Table 5: **RAID Parity and Data Regeneration Controller Design Summary**

Device	LUTs	Slice FFs	RAMB16	DCMs	Ports	Performance	
						Internal	Memory
XC2VP4-7FG256	316	150	4	2	100	188 MHz	100 MHz

Notes:

1. Using ISE4.2i Service Pack 3 with Virtex-II Pro speed files version 1.62
2. Ports: 36 inputs; 27 outputs; 36 bidirectional
3. Internal clock performance of 142 MHz.
4. External SSRAM memory bus operation at 100 MHz.

The VHDL and Verilog reference designs are available on the Xilinx FTP site:
ftp://ftp/xilinx.com/publications/xapp/xapp657.zip

Memory Interface Timings

The reference design interfaces to a Micron Semiconductor 8 Mb SYNCBURST SRAM. The SSRAM is a 3.3V V_{DD} device with a 2.5V I/O interface. The part number is MT58L512V18FS-7.5 or equivalent. In order to achieve a 100 MHz bus speed a 113 MHz SSRAM device is utilized.

The data sheet for the device can be found at **http://www.micronsemi.com**. Timing requirements for the Virtex-II Pro FPGA to SSRAM interface are shown in Table 6. The design constraint file includes constraints for both SSRAM inputs and outputs. All timing constraints were met.

Table 6: **SSRAM Memory Interface Timing Specifications**

Description	-7.5 A		Units
	MIN	MAX	
Clock Frequency		113	MHz
Output Times			
Clock to Output Valid		7.5	ns
Clock to Output Invalid	1.5		ns
$\overline{OE}$ to Output Valid / High-Z		4.2	ns
Setup Times			
Address, chip enable, address status, advance, byte write enables, data inputs	1.5		ns
Hold Times			
Address, chip enable, address status, advance, byte write enables, data inputs	0.5		ns

Reference Design Simulation

Mentor Graphics Corporation ModelSim SE, version 5.5e, was used for simulation of both the Verilog and VHDL designs. The SE version supports mixed language simulation. Two test benches are provided with this reference design: one for the Verilog and one for the VHDL design. The testbench initializes a portion of the SSRAM memory as shown in Table 7.

Table 7: **Testbench Initialized SSRAM Memory Contents**

Address	Data/Parity Block	Initialization or SSRAM Buffer Usage
000:07F	D0	Incrementing data pattern, each byte lane in a word incremented by 1.
080:0FF	D1	Incrementing data pattern, each byte lane in a word incremented by 1.
100:17F	D2	Fixed data pattern 0xDEADBEEF.
180:1FF	D3	Decrementing data pattern, each byte lane in a word decremented by 1.
200:27F	D4	Decrementing data pattern, each byte lane in a word decremented by 1.
280:2FF	P	Initialized to 0x00000000. Calculated parity stored here in Test 1.
300:37F	Regenerated D4	Initialized to 0x00000000. Regenerated D4 block stored here in Test 2.
380:3FF	Compare D4 blocks	Initialized to 0x00000000. Compare D4 to Regenerated D4 stored here in Test 3.

The testbench performs three separate tests.

Test 1: Create the parity block given five data elements. Since the testbench assumes that none of the data blocks or parity block has been written to previously, it performs five XOR operations to create the parity block. Typically, when calculating parity, the system takes the old parity, old data, and new data and exclusive OR's these blocks together each time a data element changes.

Test 2: Regenerate data block D0, given D1, D2, D3, D4, and P (previous calculation).

Test 3: Compare the original data block D0 with regenerated data block D0.

For the first calculation, the testbench uses BRAM_START_ADR of 0x000. For the second, data regeneration, calculation the testbench uses a BRAM_START_ADR of 0x100. For the data compare operation, the block RAM starting address is 0x180. This demonstrates the capability of the hardware to support multiple non-concurrent operations. The ability to support

multiple operations is important in disk systems as the latency to retrieve data/parity from the disk arrays can take milliseconds to complete.

Comparison

Data Regeneration in Software Versus Hardware

Parity calculations and data block regeneration can also be accomplished using a software routine executing on the PPC405 RISC CPU in a Virtex-II Pro FPGA. The "C" language routine "parity" calculates a regenerated 512-byte data block, given four member data blocks and a parity block. A portion of the software routine is shown below.

```
int main()
{
int I;  //loop counter
mtdcr(0x300,0x00000000);
for( I = 0; I < 128; I++)
{
results[I] = packet0[I] ^ packet1[I] ^ packet2[I] ^ packet3[I] ^ packet4[I];
}
mtdcr(0x300,0xffffffff);
__asm__ ("trap");
    return 0;
}/*end of main*/
```

Five 512-byte packets are defined, in the processor memory, and the results are computed taking the exclusive OR of each packet element. The routine runs on the Embedded Reference System from the Virtex-II Pro Developer's Kit. Time to complete the calculation, using a 300 MHz PPC405 processor block and a 100 MHz processor local bus (PLB) attached block RAM memory interface, shown in Table 8. In addition, the data-side and instruction-side On-Chip Memory Controller (OCM) interfaces were used to compare PLB versus OCM performance. **XAPP644** has information regarding PLB and OCM functionality and performance.

To determine the starting and ending points of the "parity" routine, a "move to dcr" instruction is placed at the beginning and end of the routine. On the wave window of the MTI simulation, the time to run the routine is simply the time measured between the two DCR bus operations. The trap instruction halts execution at the end of the function.

Table 8: **Software Data Regeneration Completion Times**

Case	Time to Complete (µs)	Instruction Cache	Data Cache	Data Side OCM
1	52.7	ON	OFF	OFF
2	44.0	OFF	OFF	ON
3	18.1	ON	OFF	ON

Case #1 uses all PLB attached memory for the calculation. Instruction performance is optimized by enabling the PPC405 instruction cache. Case #2 achieves a slight performance improvement using the PPC405 data-side OCM memory, even though the instruction cache is disabled. Case #3 results in the best software solution achieved using the PPC405 instruction cache and data-side OCM for the data buffer. Unfortunately, case #3 requires a hardware mechanism to get the data from a larger memory buffer into the 8 KB data-side OCM memory. PLB attached memory does not have this issue as PLB memory can be external memory.

The time to complete a data regeneration calculation, using the reference design is 8.44 µs. This is the time to complete six, 128-word burst SSRAM cycles. The corresponding time, using PLB attached block RAM, also running at 100 MHz, with the instruction cache turned on is 52.7 µs.

Conclusion

The reference design supports a maximum 8 KB data stripe, limited to the depth of the block RAM instantiated in the design. Larger data stripes can be accommodated by changing the block RAM memory instantiated in the design and modification of the address and transfer counters in the design. Table 9 shows the maximum data stripe size available with different sizes of block RAM in a Virtex-II Pro FPGA and the number of parallel operations supported by a 512-byte data stripe design. For smaller stripe sizes, the hardware can be utilized to perform multiple, non-concurrent operations. The disk seek time and disk access times can be much longer than the time it takes to compute new parity information for a data stripe. Microprocessor controlled parity and data recovery algorithms can utilize this parallel capability to start processes while disk I/O operations are retrieving data from multiple disk arrays.

Table 9: **Block SelectRAM Size Versus Maximum Data Stripe Size**

Block RAM Resources		Maximum Data Stripe Size	Number of Independent 512-byte Operations
Organization	Quality		
512 x 36	1	2KB	4
1K x 18	2	4KB	8
2K x 9	**4**	**8KB**	**16**
4K x 4	8	16KB	32
8K x 2	16	32KB	64
16K x 1	32	64KB	128

Notes:
1. The reference design uses four 2K x 9 block RAMs.

In microprocessor systems, hardware and software tradeoffs are always evaluated. This application note shows the difference between a software solution versus a hardware solution. In the case of RAID parity and data regeneration, given a memory element running at 100 MHz, a hardware assist will significantly out perform a software only solution.

Utilizing burst-mode memory accesses, the reference design minimizes the number of times memory is accessed for both new parity calculation and regenerated data calculations.

Once the RAID controller hardware is enabled, it can sustain a burst to the SSRAM for read and write operations. The transfer rate, assuming a 32-bit wide data path of 512 bytes, is over 375,000,000 bytes per second for reads and over 380,000,000 bytes per second for writes. Setup time for the user interface is not included in this performance number. A Virtex-II Pro Platform FPGA contains both the microprocessor and logic resources to permit both the microprocessor and hardware assist logic to be contained in one device.

References

1. "The RAIDbook: A Source Book for Disk Array Technology", Fourth Edition, published by the RAID Advisory Board.
2. Virtex-II Pro Platform FPGA Documentation on the web at: http://www.xilinx.com/publications/products/v2pro/handbook/index.htm

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/15/02	1.0	Initial Xilinx release.